

CLASE 10

SEMINARIO DE LENGUAJES JS

Presentación del framework Express de NodeJS

AGENDA

1. Js - Server Side
2. Modelo Mvc
3. Stack Mean Y Mern
4. Express Js
5. Módulos En Js - Npm
6. Express Js - Instalación
7. Creando Un Servidor
8. Express - Rutas
9. Express - Rutas Dinámicas
10. Express - Middleware
11. Express - Middleware Express.static
12. Nodemon
13. Referencias

JS - SERVER SIDE

Cuando programamos una aplicación Web server-side, estamos programando una aplicación que reciba peticiones HTTP del navegador o cliente y retorne datos.

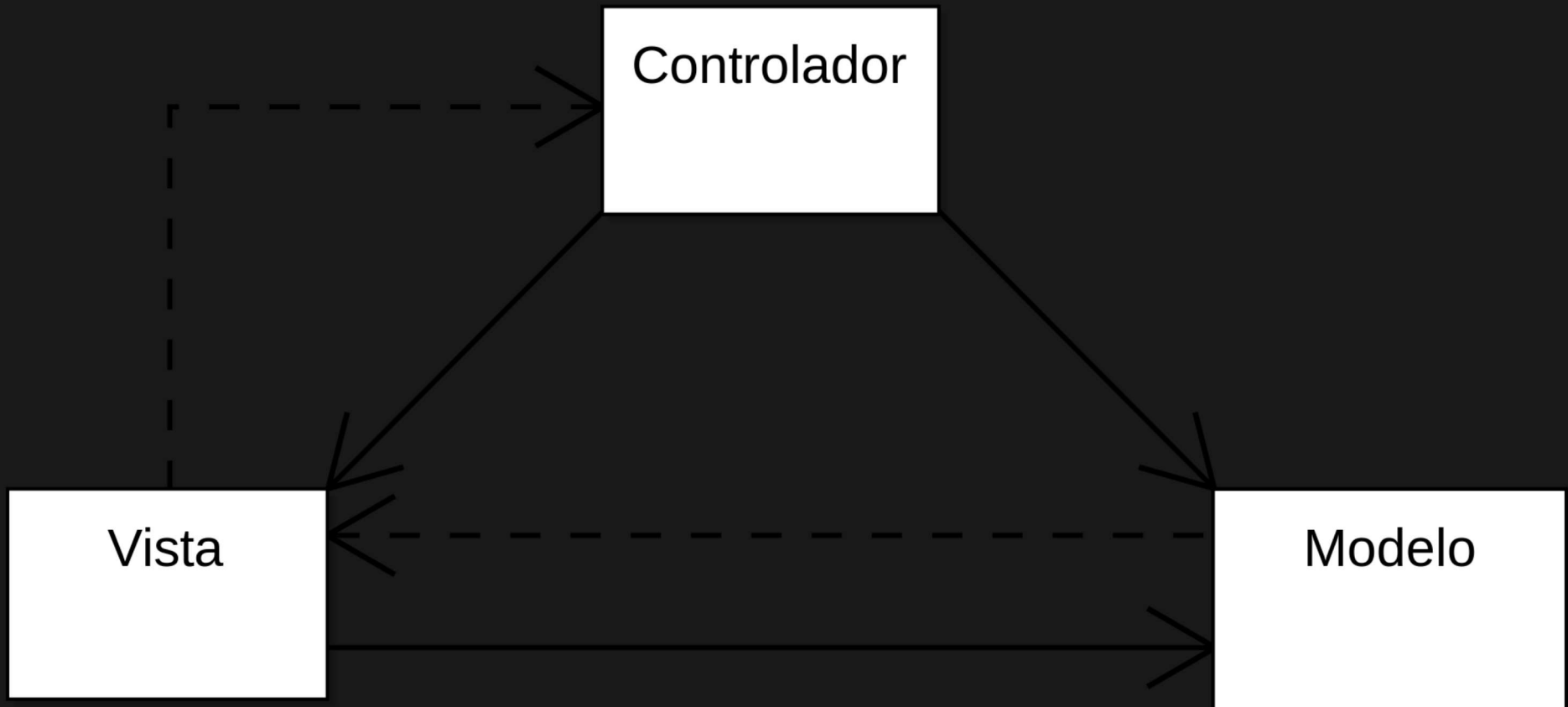
La aplicación determina qué datos retornar según la acción a ejecutar de acuerdo a la estructura de la URL y la información indicada en la misma (opcional) según los métodos GET y POST .

Según la acción, puede acceder a una base de datos o a un archivo para atender la petición y retornar en un HTML los datos solicitados.

MODELO MVC

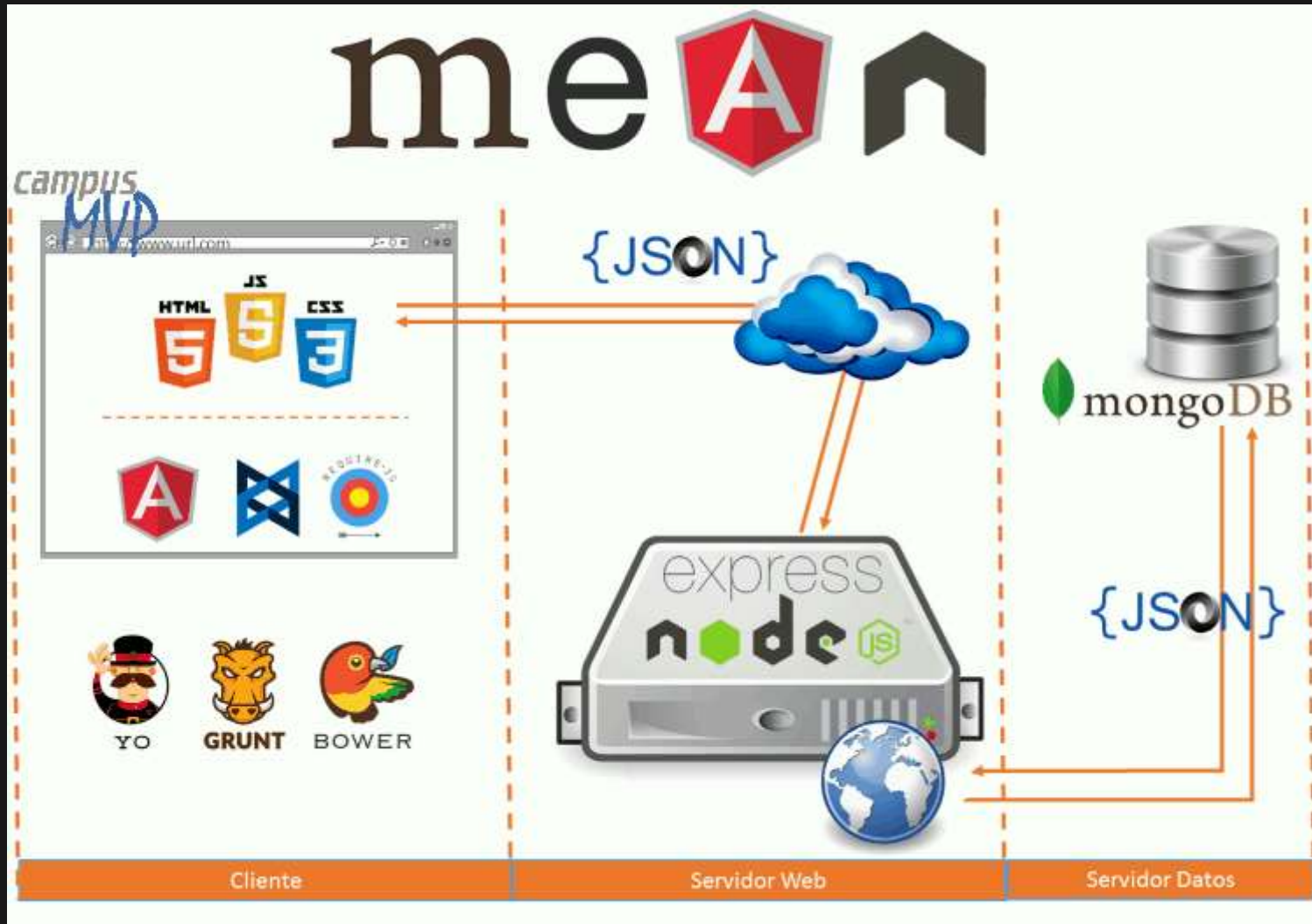
- Patrón de arquitectura de software para separar los datos y la lógica de negocio de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.
- Modelo: Es la representación de la información.
- Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' . Intermediario.
- Vista: Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar

MODELO MVC²

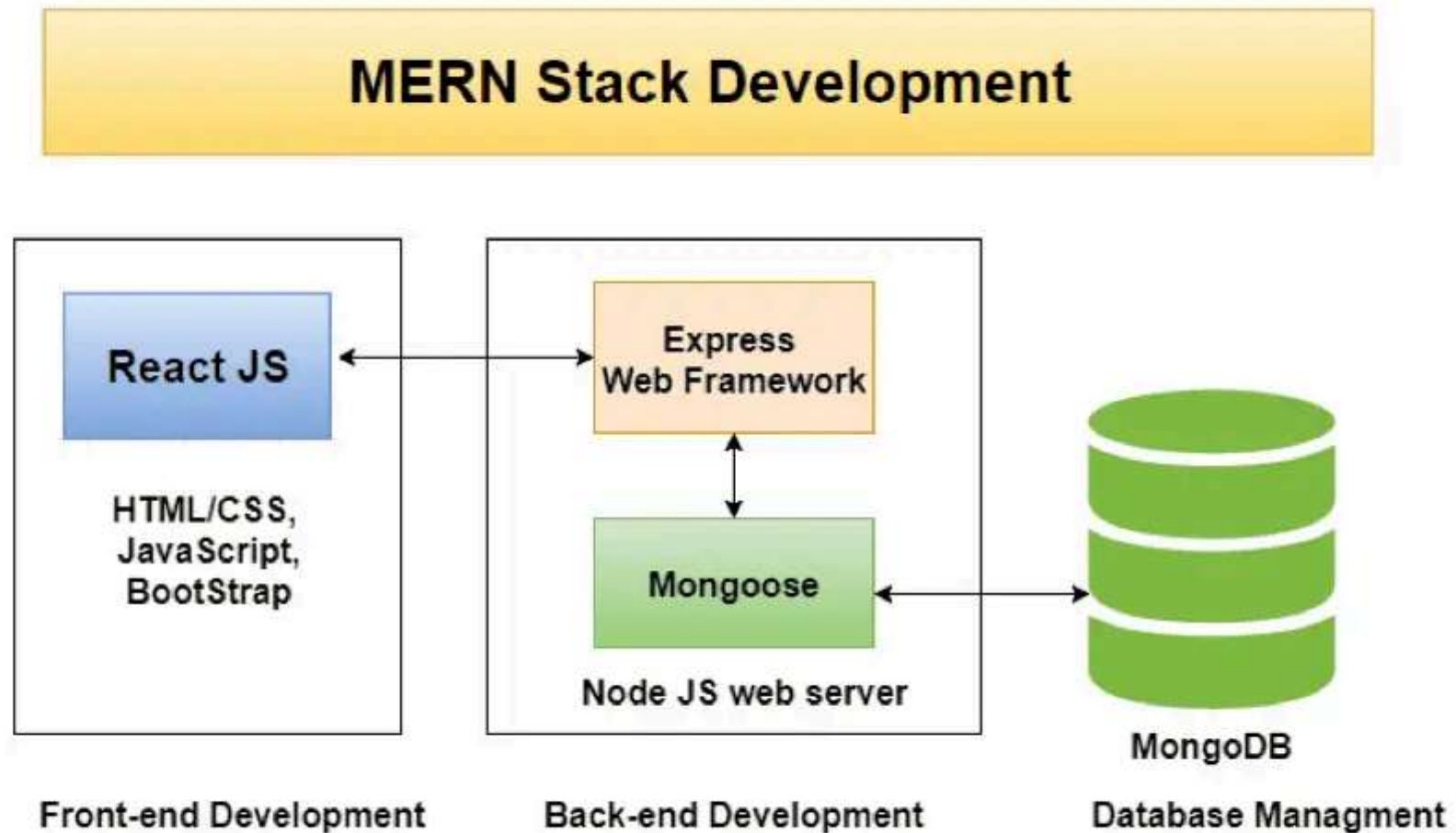


STACK MEAN Y MERN

Permiten crear aplicaciones Web utilizando todas herramientas JS.



STACK MEAN Y MERN²



EXPRESS JS

- Es el framework más popular de Node, liberado en 2010, actualmente en la versión 4.19.2
- Facilita la creación de aplicaciones Web, del lado del servidor.
- Construido a partir de NodeJS.
- Modular a través de npm, evita verbosity.
- Single thread, asíncronico
- Javascript Universal o Isomórfico. Se comparten librerías, entre el cliente y el servidor. Por ejemplo timing, loadge, o fechas optimizando el rendimiento de las aplicaciones.

Express vs Fastify vs Hapi vs Koa vs Nestjs
Análisis comparativo de frameworks según
performance (2023)

EXPRESS JS ²

- Permite escribir manejadores de rutas (URLs) para los diferentes verbos HTTP.
- Se integra con renderizaciones de vistas para generar respuestas a través de introducir datos en plantillas.
- Añade procesamiento de peticiones middleware en cualquier punto dentro del manejo de la petición.
- Aplicaciones MVC. Model View Controller.
- Es muy sencillo de aprender los conceptos y utilizar un servidor Web.
- Es no dogmático (unopinionated).

EXPRESS JS ³

Posee métodos para especificar que función usar dependiendo del verbo usado en la petición y la estructura de la URL (**ruta**).

También tiene métodos para especificar plantillas (**views**) que se usaran para armar el documento HTML con la respuesta.

Los **middlewares** se utilizan para añadir funcionalidad como sesiones de usuario, cookies, logging, etc.

MÓDULOS EN JS - NPM

Un módulo es una librería de JS que puede ser importada en otro código usando `require()`
También podemos crear nuestros propios módulos, cuanto más modular más óptimo para reusar y encapsular.

EXPRESS JS - INSTALACIÓN

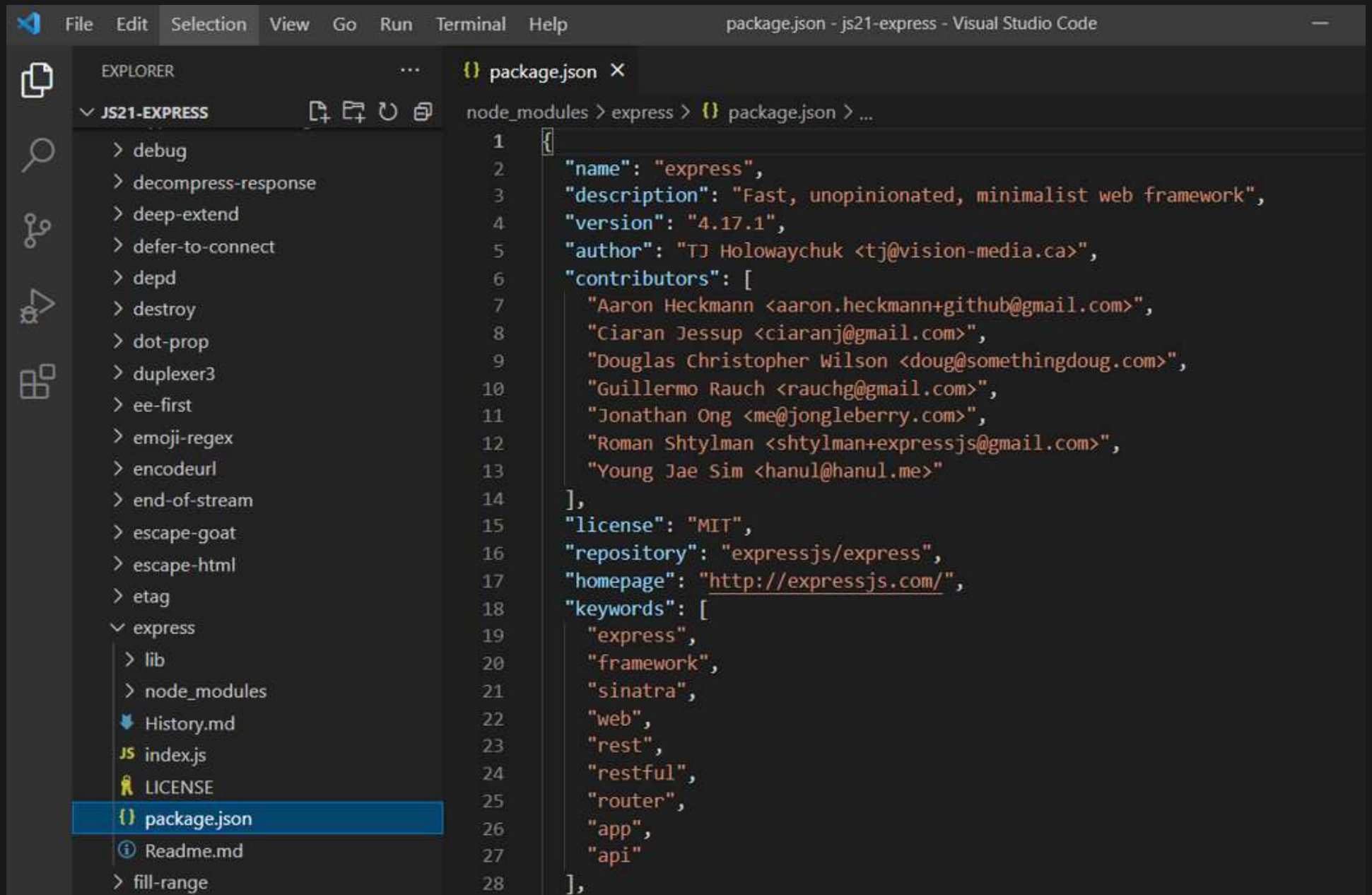
Es necesario instalarlo ejecutando npm

```
npm install express --save
```

Node incluye módulos de todo tipo en su instalación. Express es necesario instalarlo y se puede modificar el archivo package.json

La opción --save incluye Express dentro de las dependencias del archivo package.json para registrarlo correctamente.

EXPRESS JS - INSTALACIÓN ²



The image shows a screenshot of the Visual Studio Code editor. The Explorer sidebar on the left displays the file structure of a project named 'JS21-EXPRESS'. The 'express' directory is expanded, showing files like 'lib', 'node_modules', 'History.md', 'index.js', 'LICENSE', 'package.json', 'Readme.md', and 'fill-range'. The 'package.json' file is selected and its content is displayed in the main editor area. The file path in the editor is 'node_modules > express > package.json'. The code in 'package.json' is as follows:

```
1 {  
2   "name": "express",  
3   "description": "Fast, unopinionated, minimalist web framework",  
4   "version": "4.17.1",  
5   "author": "TJ Holowaychuk <tj@vision-media.ca>",  
6   "contributors": [  
7     "Aaron Heckmann <aaron.heckmann+github@gmail.com>",  
8     "Ciaran Jessup <ciaranj@gmail.com>",  
9     "Douglas Christopher Wilson <doug@somethingdoug.com>",  
10    "Guillermo Rauch <rauchg@gmail.com>",  
11    "Jonathan Ong <me@jongleberry.com>",  
12    "Roman Shtylman <shtylman+expressjs@gmail.com>",  
13    "Young Jae Sim <hanul@hanul.me>"  
14  ],  
15  "license": "MIT",  
16  "repository": "expressjs/express",  
17  "homepage": "http://expressjs.com/",  
18  "keywords": [  
19    "express",  
20    "framework",  
21    "sinatra",  
22    "web",  
23    "rest",  
24    "restful",  
25    "router",  
26    "app",  
27    "api"  
28  ],
```


CREANDO UN SERVIDOR

C: > Users > Cespi > JS-node > JS index.js > [m] morgan

```
1  const http = require('http')
2
3  const server = http.createServer((req, res) => {
4    res.status = 200;
5    res.setHeader = ('Content-type', 'text/plain')
6    res.end('Hello World');
7  });
8
9  server.listen(3000, () =>
10    console.log ('Server on port 3000')
11  );
12
```

```
12
13  const express = require('express');
14  const app = express();
15
16  app.listen(3000, () =>
17    console.log('Server on port 3000')
18  );
19
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

CREANDO UN SERVIDOR ²

```
{ } package.json JS index.js
```

```
JS index.js > ...  
1  const express = require('express');  
2  const app = express();  
3  
  
app.listen(5000, () => {  
  console.log('Servidor en el puerto 5000');  
})
```

Módulo que voy a utilizar. Internamente utiliza el módulo http de NodeJS

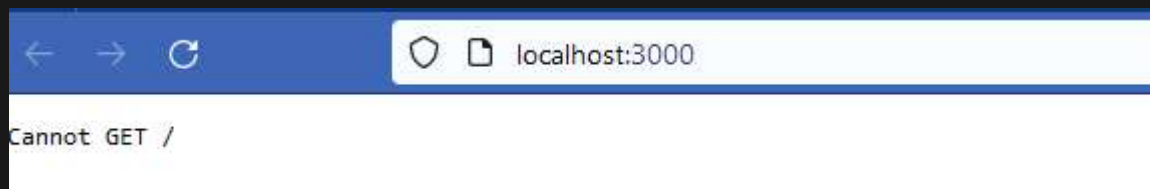
Retorna un objeto servidor

Seteo el puerto donde va a escuchar el servidor.

```
pamadeo@LAPTOP-4E49AE2Q: /mnt/c:/Users/pamadeo/Desktop/15/JS/21-express: $ node index.js  
Servidor en el puerto 5000
```

CREANDO UN SERVIDOR ³

Haciendo una petición desde un cliente a nuestro servidor obtenemos:



¿POR QUÉ?

CREANDO UNA RESPUESTA

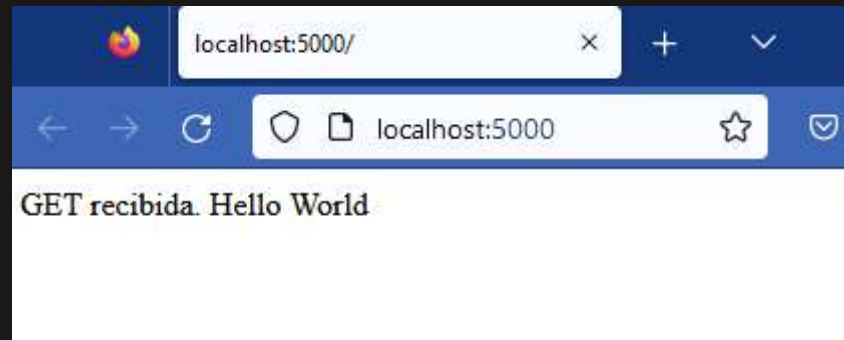
```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Seminario JS!');
}).listen(8080);
```

JS index.js > ...

```
1  const express = require('express');
2  const app = express();
3
4  app.get('/', (req, res) => {
5    |  res.send('GET recibido');
6  });
7
```

EXPRESS - RUTAS



EXPRESS - ENRUTAMIENTO

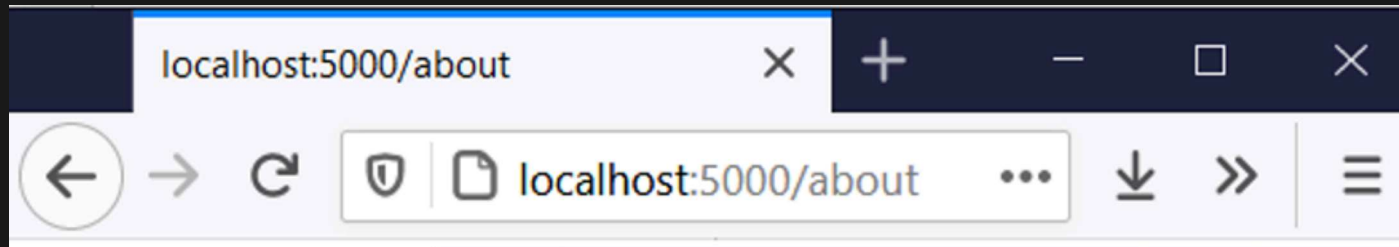
- Las rutas definen la navegación del usuario a partir de los verbos http.
- Determina la respuesta de una aplicación a una solicitud de cliente, establecida en la URL.
- Es la base de toda aplicación Web.
- Es necesario configurar un manejador.

EXPRESS - ENRUTAMIENTO ²

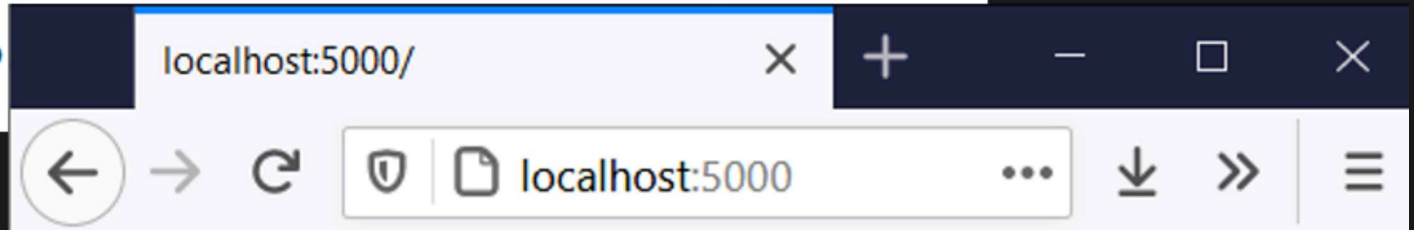
Soporte a los métodos de direccionamiento que se corresponden con los métodos HTTP: get, post, put, head, delete, options, trace, copy, lock, mkcol, move, purge, propfind, proppatch, unlock, report, mkactivity, checkout, merge, m-search, notify, subscribe, unsubscribe, patch, search y connect.

EXPRESS - ENRUTAMIENTO ³

```
} package.json  JS index.js  ●  JS application.js
JS index.js > ...
1  const express = require('express');
2  const app = express();
3
4  app.get('/', (req, res) => {
5    res.send('GET recibido');
6  });
7
8  app.get('/about', (req, res) => {
9    res.send('GET about recibido');
10 });
11
12 app.get('/contact', (req, res) => {
13   res.send('GET contact recibido');
14 });
15
16 app.listen(5000, () => {
17   console.log('Servidor en el puerto 5000');
18 })
```

GET about recibido



GET recibido

EXPRESS - ENRUTAMIENTO ⁴

Es posible incluir expresiones regulares como * y ?

```
app.post('/', function (req, res) {  
  res.send('Got a POST request');  
});
```

```
app.put('/user', function (req, res) {  
  res.send('Got a PUT request at /user');  
});
```

```
app.delete('/user', function (req, res) {  
  res.send('Got a DELETE request at /user');  
});
```

Se pueden utilizar expresiones regulares como *?

```
app.get('/ab*cd', function(req, res) {  
  res.send('ab*cd');  
});
```

```
app.all('/secret', function (req, res, next) {  
  console.log('Accessing the secret section ...');  
  next(); // pass control to the next handler  
});
```

Guide Express JS - Routes

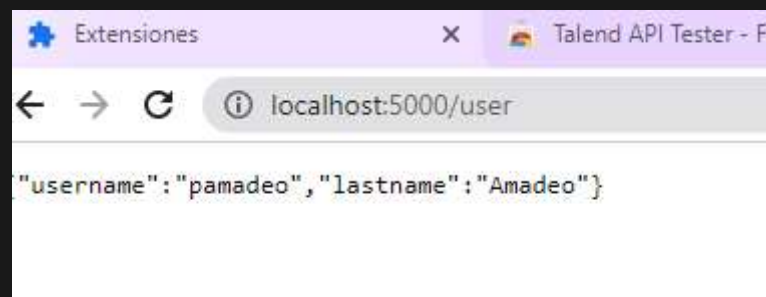
EXPRESS - ENRUTAMIENTO - 5

APP.ALL()

No se deriva de ningún método HTTP. Este método se utiliza para cargar funciones de middleware en una vía de acceso para todos los métodos de solicitud. El manejador se ejecutará para las solicitudes a “/secret”, tanto si utiliza GET, POST, PUT, DELETE, como cualquier otro método de solicitud HTTP.

EXPRESS - ENRUTAMIENTO ⁶

```
app.get('/user', (req, res) => {  
  res.json({  
    username: 'pamadeo',  
    lastname: 'Amadeo'  
  });  
});
```



EXPRESS - ENRUTAMIENTO ⁷

REQ.BODY

Objeto JS que provee express y permite gestionar los datos enviados por el cliente

The screenshot shows a REST client interface for a request named "js-postAPI-user". The method is set to "POST" and the URL is "http://localhost:5000/user". The request body is a JSON object: `{ "username": "tmalaruk", "lastname": "Malaruk" }`. The "Content-Type" header is set to "application/json".

js-postAPI-user

METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]

http://localhost:5000/user

length: 26 byte(s)

QUERY PARAMETERS

HEADERS

Content-Type: application/json

Form

BODY

```
1 { "username": "tmalaruk", "lastname": "Malaruk" }
```

Text

+ Add header Add authorization

Top Bottom 2 Request Copy Download

EXPRESS - ENRUTAMIENTO ⁸

```
app.post('/user', (req, res) => {  
  console.log(req.body);  
});
```

Y también:

```
// Para indicarle a express que le estamos e  
app.use(express.json());
```

Cuando una petición coincida con el content-type entiende que es un json e interpreta el objeto

EXPRESS - RUTAS DINÁMICAS

Permite manipular parámetros en la URL a través del objeto `Request.params`

```
✓ app.post('/user:idUser', (req, res) => {  
  console.log(req.body);  
  console.log(req.params);  
  res.send('POST user recibido');  
});
```

Tomar la información que envía el cliente.

Tomar la información que envía el cliente como parámetro en la petición.



```
Servidor en el puerto 5000  
{ materia: 'JS', facultad: 'Informatica' }  
{ idUser: ':256' }
```

EXPRESS - RUTAS DINÁMICAS ²

```
14
15 app.post('/user/:idUser', (req, res) => {
16   console.log(req.body);
17   console.log(req.params);
18   res.send('POST user recibido');
19 });
20
21 app.put('/user/:idUser', (req, res) => {
22   res.send(`${req.params.idUser}`);
23 });
24
```

The screenshot shows the RESTED Client interface in a browser. The 'Request' panel is active, showing a PUT request to `http://localhost:5000/user/256`. The headers section shows `Content-Type` set to `application/json`. The request body is set to `JSON` type, with parameters `materia` (value: `JS`) and `facultad` (value: `Informatica`). The 'Response' panel shows a successful `200 OK` response from `http://localhost:5000/user/256`.

EXPRESS - MIDDLEWARE

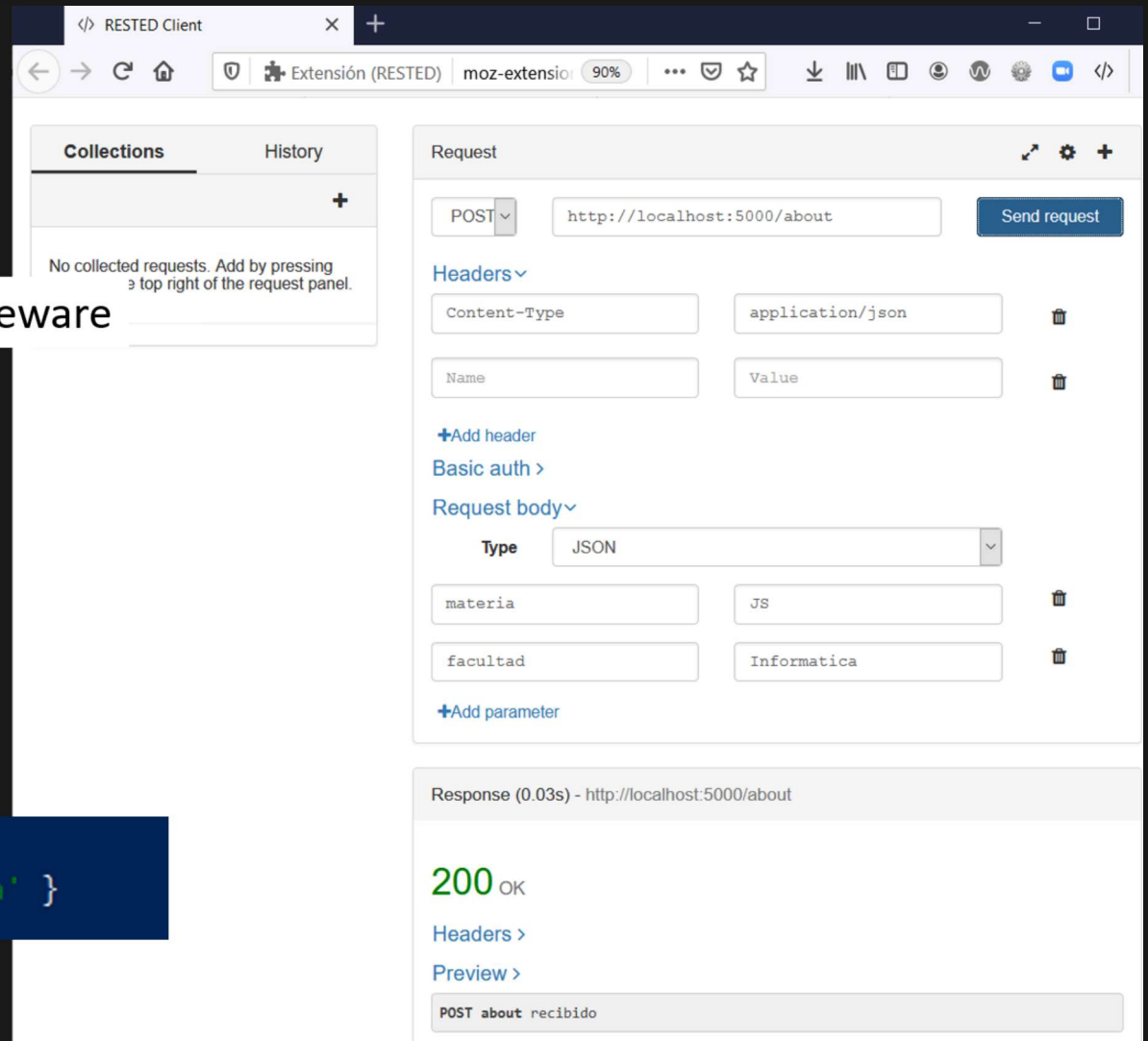
```
app.use(express.json());
```

```
app.get('/', (req, res) => {  
  res.send('GET recibido');  
});
```

```
app.post('/about', (req, res) => {  
  console.log(req.body);  
  res.send('POST about recibido');  
});
```

Middleware

Servidor en el puerto 5000
{ materia: 'JS', facultad: 'Informatica' }



EXPRESS - MIDDLEWARE ²

Es un manejador de peticiones, útil para procesar datos antes de ejecutar las rutas. Por esto es que se incluyen al inicio.

Funciona para todas las rutas definidas en la aplicación.

Por ejemplo, el logueo o la autenticación de usuarios.

Se llaman en el orden que son declaradas, en algunos casos el orden es importante.

EXPRESS - MIDDLEWARE³

```
function logger(res, req, next) {  
  console.log('Petición recibida en logger');  
  next();  
};  
  
app.use(express.json());  
app.use(logger);  
  
app.all('/user', (req, res, next) => {  
  console.log('Recibido en all');  
  next();  
});
```

```
function logger(req, res, next) {  
  // console.log('Petición recibida en logger');  
  console.log(`Ruta ${req.protocol}`);  
  next();  
};
```



```
amadeo@LAPTOP-4E49AE2Q:/mnt/c/Users/pulam/JS/js21-express$ node index.js  
Servidor en el puerto 5000  
Ruta http
```

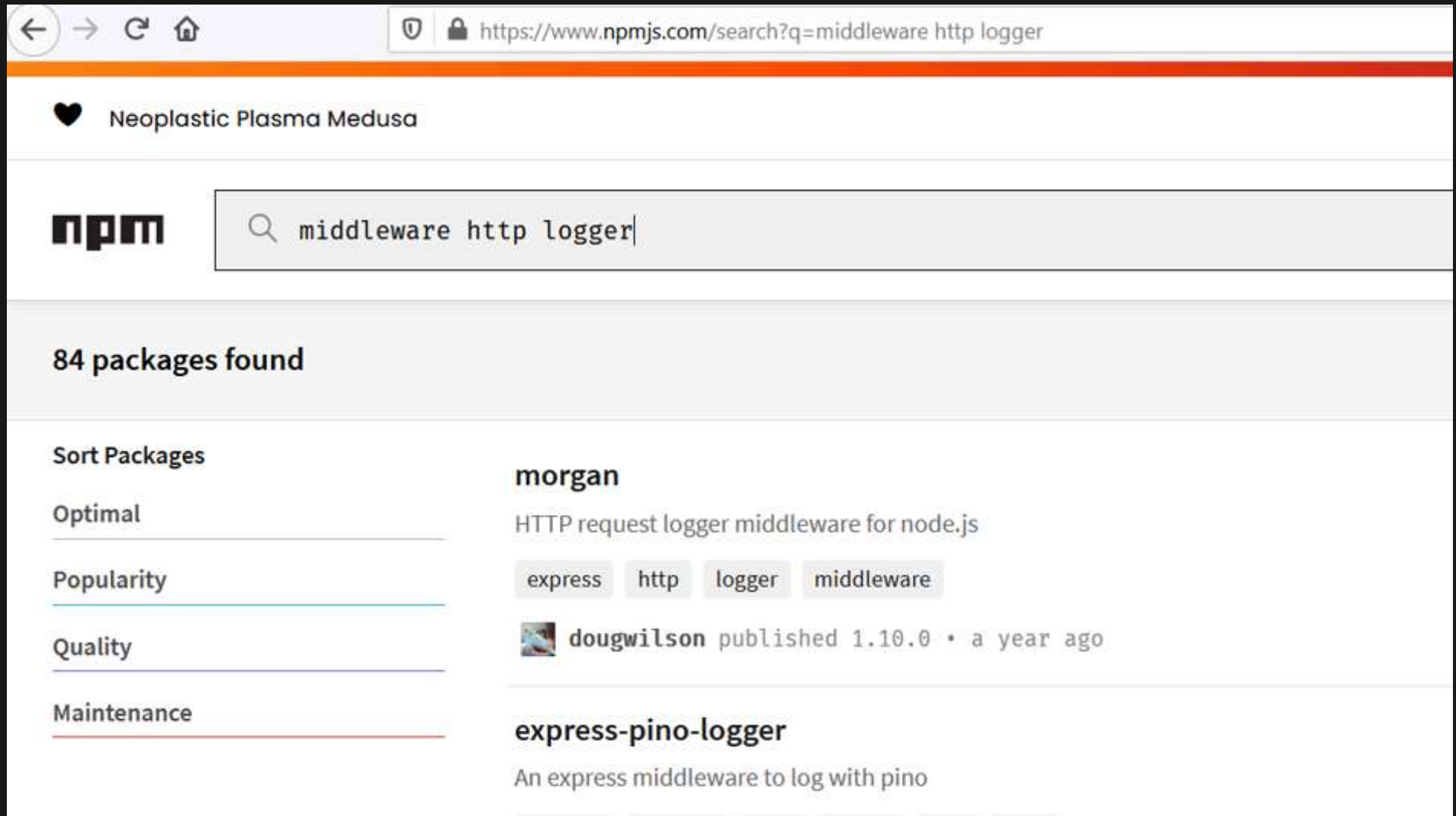
EXPRESS - MIDDLEWARE⁴

- Módulos de terceras partes, como Morgan
- Para subir imágenes, cambiar datos, autenticación.
- npmjs.com/
- Un ejemplo de middleware es [morgan](#) para loguear todas las peticiones.

```
app.use(morgan('dev'));
```

```
[nodemon] restarting due to changes...  
[nodemon] starting `node index.js`  
Server on port 5000  
GET /user/345 404 2.661 ms - 147  
□
```

EXPRESS - MIDDLEWARES DE TERCEROS⁵



The screenshot shows the npm website search results for the query "middleware http logger". The browser address bar shows the URL "https://www.npmjs.com/search?q=middleware http logger". The search bar contains the text "middleware http logger". Below the search bar, it says "84 packages found". On the left side, there is a "Sort Packages" section with options: "Optimal", "Popularity", "Quality", and "Maintenance". The "Popularity" option is selected. The search results list two packages:

- morgan**
HTTP request logger middleware for node.js
Tags: express, http, logger, middleware
Published by dougwilson 1.10.0 • a year ago
- express-pino-logger**
An express middleware to log with pino

Middlewares soportados por el equipo de Express

EXPRESS - MIDDLEWARE EXPRESS.STATIC

Es un middleware que se incluye en el core de Express para gestionar los archivos estáticos como css, el HTML, JS, etc.

EXPRESS - MIDDLEWARE

Existen paquetes de middleware para abordar casi cualquier requerimiento, el tema es decidir cuales son los paquetes adecuados .

No necesariamente existe una única forma correcta de estructurar una aplicación y muchos ejemplos sólo muestran una parte mínima de lo que es necesario hacer para desarrollar una aplicación web.

NODEMON

Vigila el código de JS para alertar cuando este modificado el JS sin necesidad de volver a iniciarlo cada vez.

```
npm i nodemon -D  
npx nodemon index.js
```


REFERENCIAS

npm middlewares

Introducción a Express/Node. MDN

Node- Anatomía de una transacción HTTP

Cascada CSS

BootStrap. Wikipedia