

SEMINARIO DE
LENGUAJES

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.

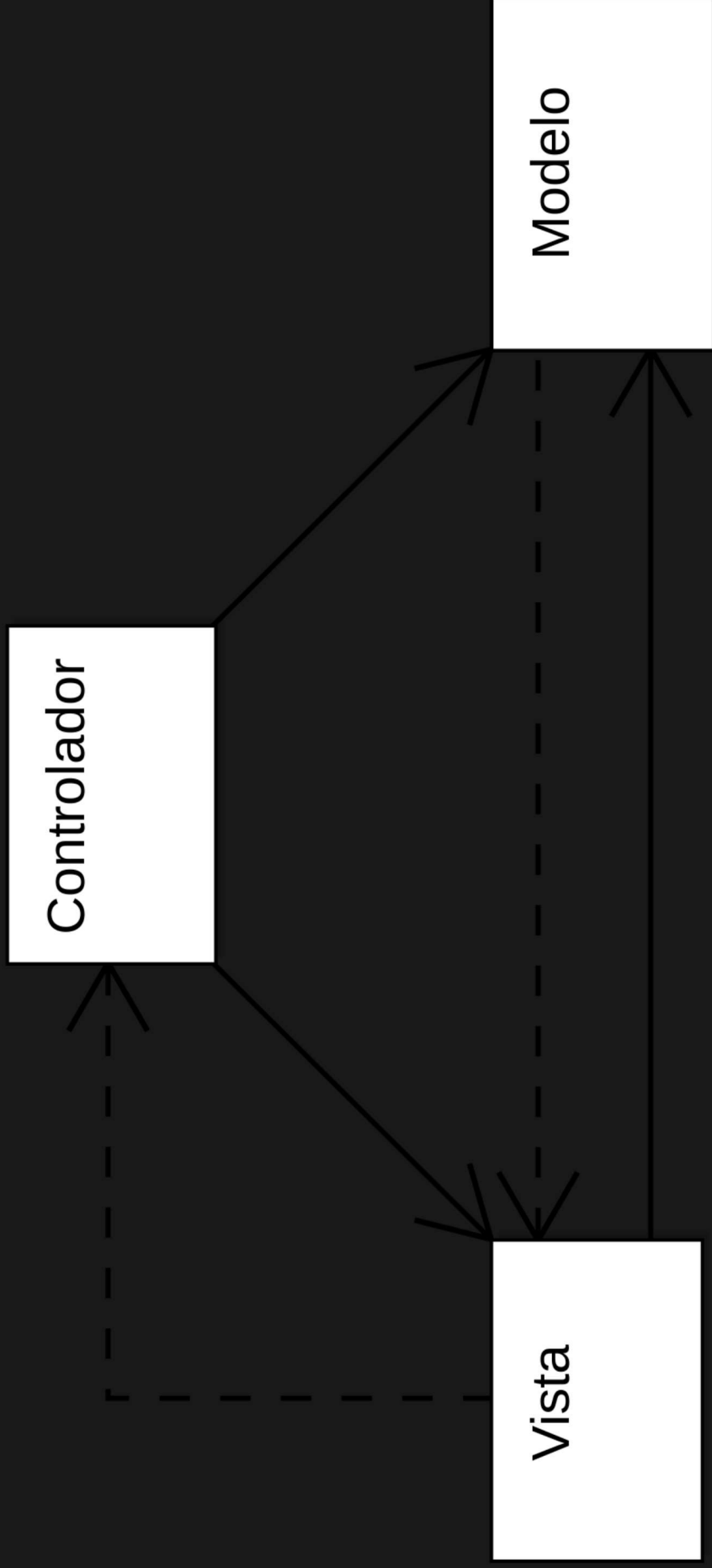
La aplicación determina qué datos retornar según la acción a ejecutar de acuerdo a la estructura de la URL y la información indicada en la misma (opcional) según los métodos GET y POST.

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

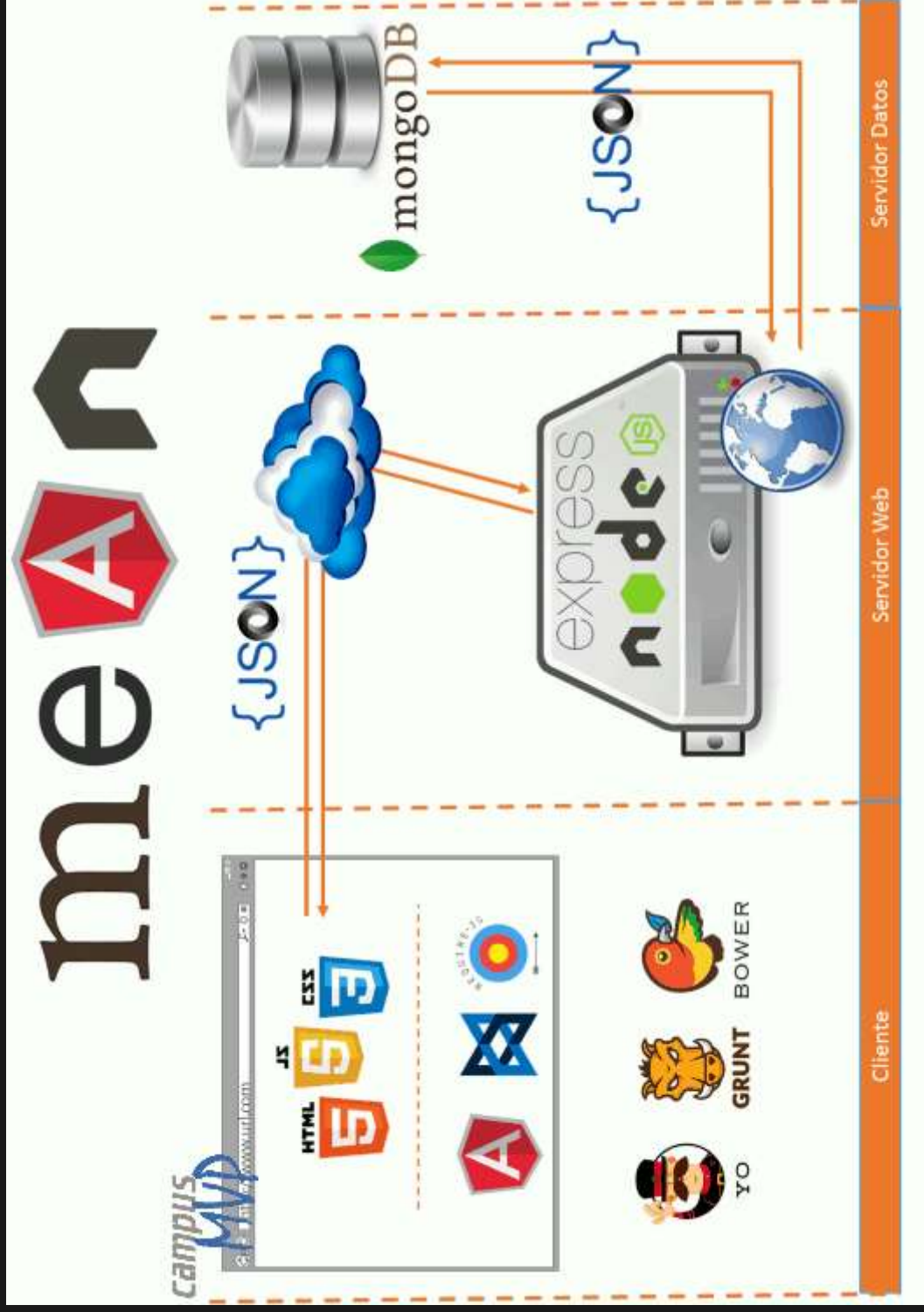
- Patrón de arquitectura de software para separar los datos y la lógica de negocio de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.
- Modelo: Es la representación de la información.
- Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones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

MODELO MVC²



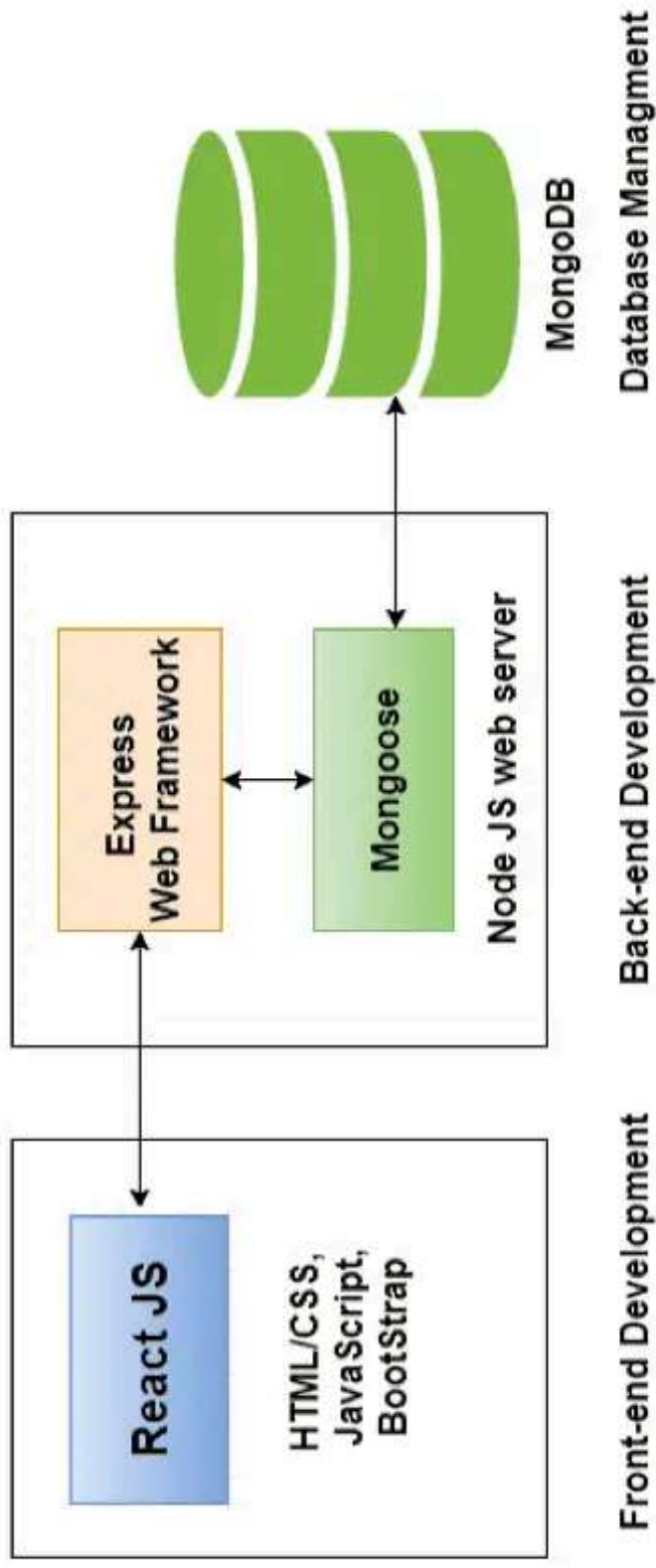
STACK MEAN Y MERN

Permiten crear aplicaciones Web utilizando todas herramientas JS.



STACK MEAN Y MERN²

MERN Stack Development



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 verbosidad.
- Single thread, asíncrono
- 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 JS²

- 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 JS³

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 screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The Explorer sidebar displays a file tree for a project named "JS21-EXPRESS". The files listed are: debug, decompress-response, deep-extend, defer-to-connect, depd, destroy, dot-prop, duplex3, ee-first, emoji-regex, encodeurl, end-of-stream, escape-goat, escape-html, etag, express, lib, node_modules, History.md, index.js, LICENSE, package.json (highlighted), and Readme.md. The package.json file is open in the main editor area, showing its content. The top of the editor has a menu bar with File, Edit, Selection, View, Go, Run, Terminal, and Help. The top of the editor also shows the path: package.json - js21-express - Visual Studio Code. The package.json content is as follows:

```
{
  "name": "express",
  "description": "Fast, unopinionated, minimalist web framework",
  "version": "4.17.1",
  "author": "TJ Holowaychuk <tj@vision-media.ca>",
  "contributors": [
    "Aaron Heckmann <aaron.heckmann+github@gmail.com>",
    "Ciaran Jessup <ciaranj@gmail.com>",
    "Douglas Christopher Wilson <doug@somethingdoug.com>",
    "Guillermo Rauch <rauchg@gmail.com>",
    "Jonathan Ong <me@jongleberry.com>",
    "Roman Shtylman <shtylman+expressjs@gmail.com>",
    "Young Jae Sim <hanul@hanul.me>"
  ],
  "license": "MIT",
  "repository": "expressjs/express",
  "homepage": "http://expressjs.com/",
  "keywords": [
    "express",
    "framework",
    "sinatra",
    "web",
    "rest",
    "restful",
    "router",
    "app",
    "api"
  ],
}
```


CREANDO UN SERVIDOR

```
C: > Users > Cespi > JS-node > JS index.js > [e] 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`

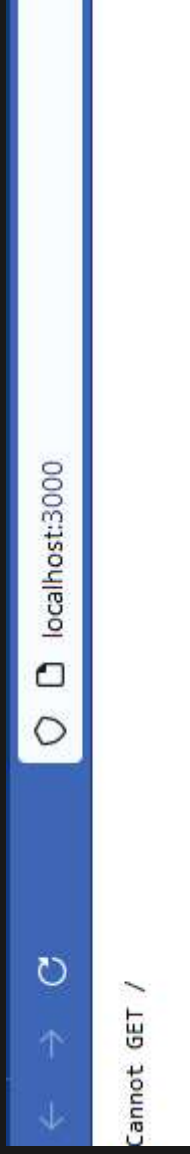
Retorna un objeto servidor

Seteo el puerto donde va a
escuchar el servidor.

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

CREANDO UN SERVIDOR³

Haciendo una petición desde un cliente a nuestro servidor obtenemos:



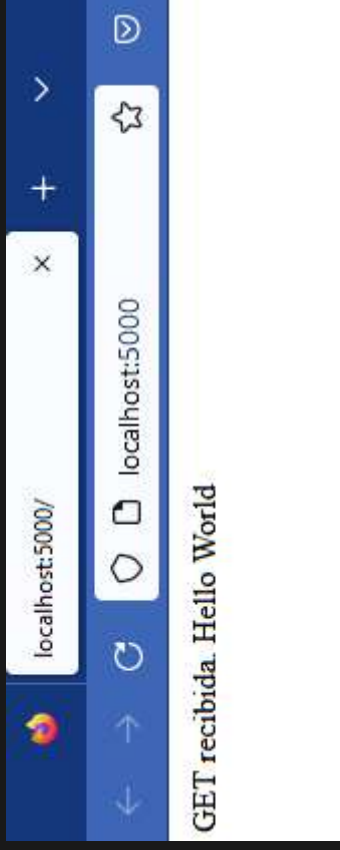
¿POR QUÉ?

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

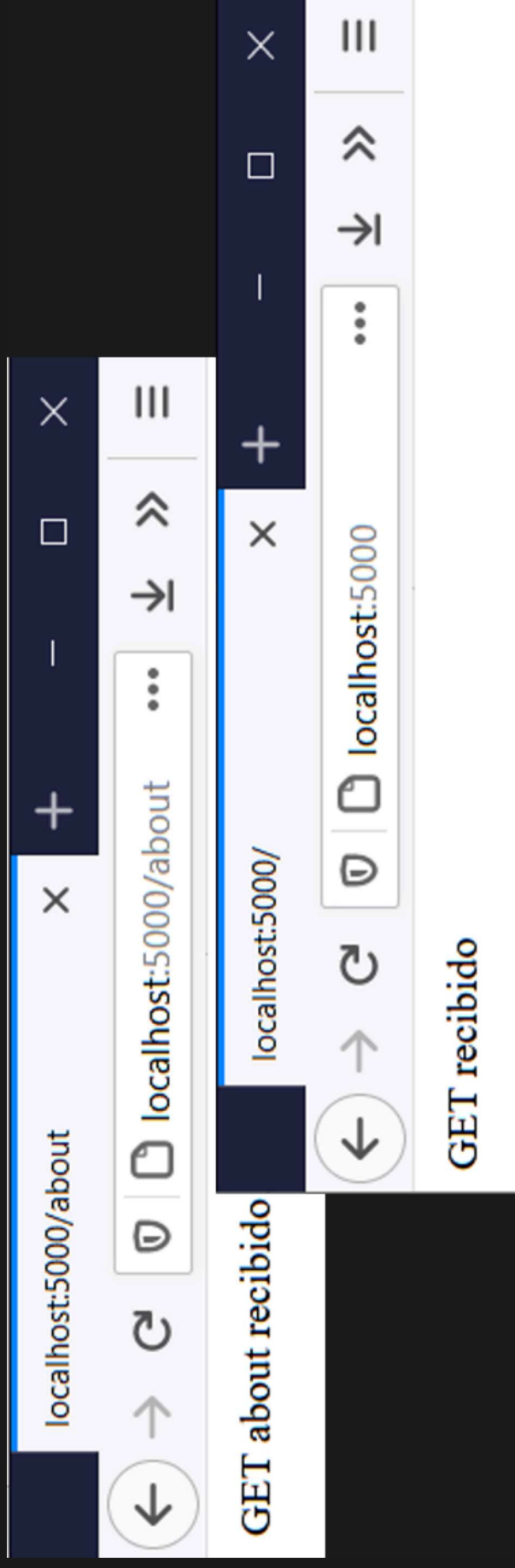
- Las rutas definen la navegación del usuario a partir de los verbos http.
- Determina la respuesta de una aplicación a una solicitud de cliente, establecida en la URL.
- Es la base de toda aplicación Web.
- Es necesario configurar un manejador.

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, mlock, move, purge, proppatch, unlock, report, mkactivity, checkout, merge, m-search, notify, subscribe, unsubscribe, patch, search y connect.

EXPRESS - ENRUTAMIENTO³

```
} 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 })
```

EXPRESS - ENRUTAMIENTO⁴

Es posible incluir expresiones regulares como * y ?

Se pueden utilizar expresiones regulares como *?

```
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');  
});
```

```
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

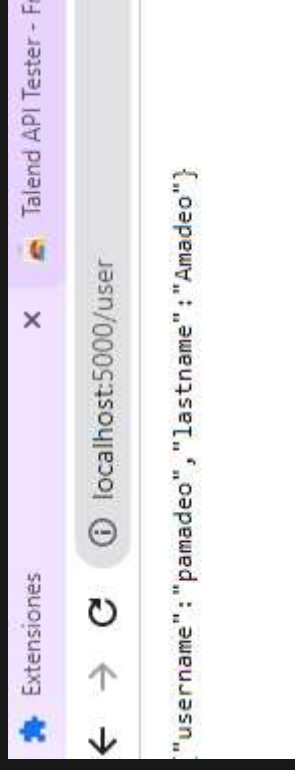
APP.ALL()

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 - ENRUTAMIENTO⁶

```
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 the 'js-postAPI-user' REST client interface. The URL is set to 'http://localhost:5000/user'. The method is 'POST'. The body is a JSON object: { "username": "tmalaruk", "lastname": "Malaruk" }. The content type is set to 'application/json'. The interface includes tabs for HEADERS, QUERY PARAMETERS, and BODY. The BODY tab is active, showing the JSON payload. The length of the body is 26 bytes. The interface also has a 'Send' button and a 'Save' button.

js-postAPI-user

METHOD: POST

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

http://localhost:5000/user

length: 26 byte(s)

HEADERS: Content-Type: application/json

Form

Content-Type: application/json

+ Add header

QUERY PARAMETERS

BODY

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

Text

Save Send

Top Bottom Request Copy Download

EXPRESS - ENRUTAMIENTO⁸

```
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 coincide 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 DINÁMICAS²

The screenshot displays the REST Client interface. On the left, the 'Collections' panel is empty. The 'Request' tab is active, showing a PUT request to `http://localhost:5000/user/256`. The request headers are set to `application/json`. The request body is a JSON object: `{ "materia": "JS", "facultad": "Informatica" }`. The 'Response' tab shows a `200 OK` status with a response time of `0.021s`. The response body is a JSON object: `{ "id": 256 }`.

```

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

```



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

No collected requests. Add by pressing  top right of the request panel.

Collections **History** **+**


RESTED Client


Extension (RESTED) moz-extensio 90%

Request

POST **Send request**

Headers


Content-Type 


Name 


+Add header

Basic auth >

Request body

Type 

materia 

facultad 

+Add parameter

Response (0.03s) - http://localhost:5000/about

200 OK

Headers >

Preview >

POST about recibido

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();  
};
```

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

EXPRESS - MIDDLEWARE⁴

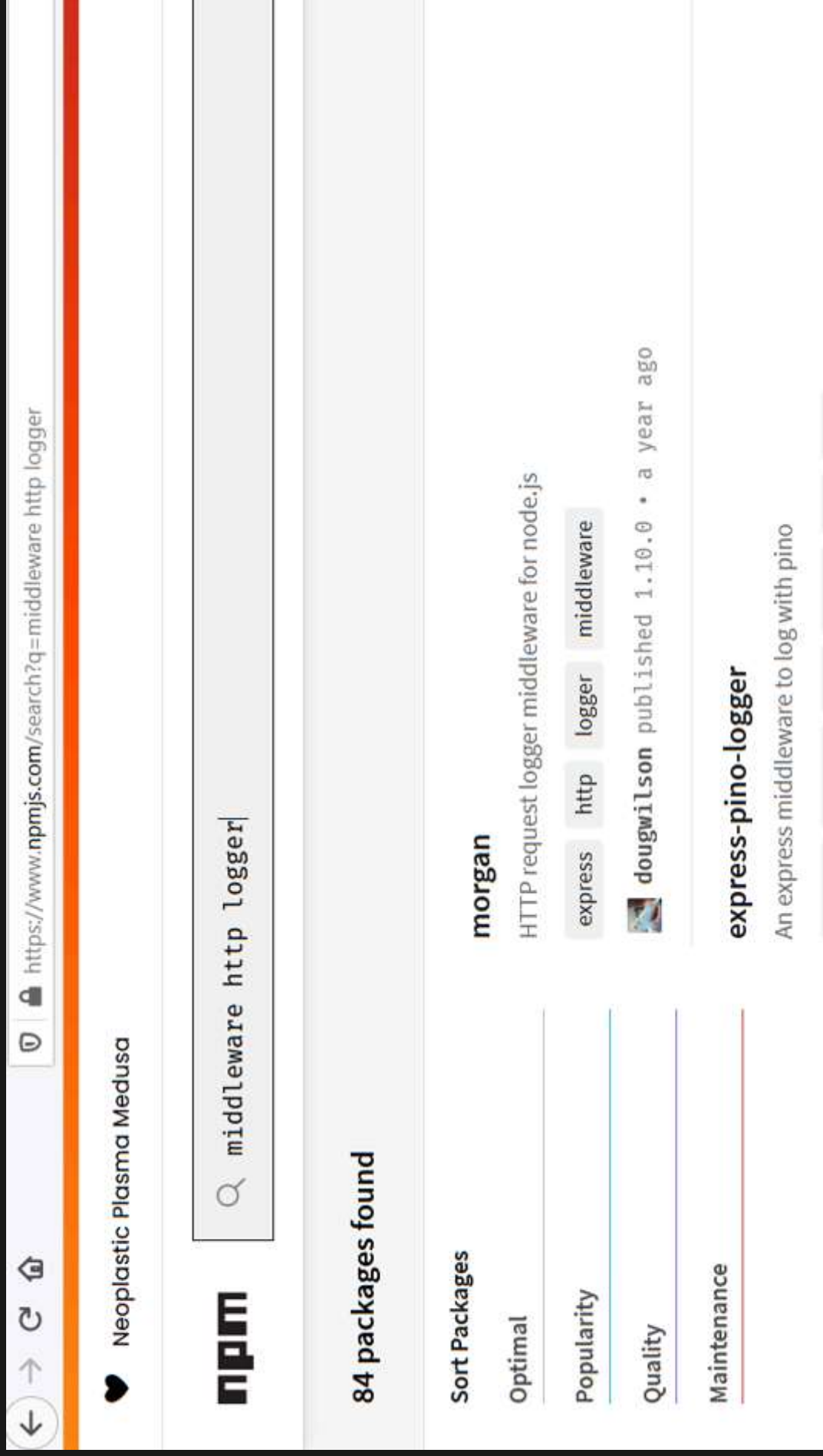
- Módulos de terceras partes, como Morgan
- Para subir imágenes, cambiar datos, autenticación.
- npmjs.com/
- Un ejemplo de middleware es **morgan** para loguear todas las peticiones.

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


A terminal window with a dark background showing logs from the 'nodemon' package. The logs indicate a restart due to changes, followed by the start of 'node index.js'. It shows the server is running on port 5000 and receives a GET request on /user/345 404 2.661 ms - 147.

```
[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 with the search bar containing 'middleware http logger'. Below the search bar, it indicates '84 packages found'. The results are sorted by 'Optimal' and show the following packages:

Sort Packages	Optimal	Popularity	Quality	Maintenance
morgan	HTTP request logger middleware for node.js	express http logger middleware	 dougwilson published 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](#)