

# Express.pdf



**Pops\_B**



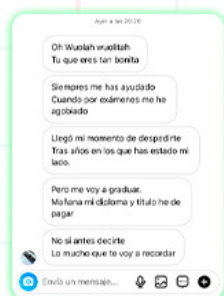
**Aplicaciones Web**



**4º Grado en Ingeniería del Software**



**Facultad de Informática  
Universidad Complutense de Madrid**



**Que no te escriban poemas de amor  
cuando terminen la carrera**

**WUOLAH**

*(a nosotros por  
suerte nos pasa)*

# La escuela de Ciberseguridad más grande del mundo.

La formación más completa y transversal que demanda el mercado.

IMEF x Deloitte.  
Smart Education

## FRAMEWORK EXPRESS

Separa vista de **controlador**

↳ Gestiona todas las peticiones  
Distribuido en mini-controladores

1 - Crear un proyecto

2 - Añadir express

`npm init`

`npm install express --save`

**app.js**

```
"use strict";

const express = require("express");
const app = express();
app.get("/", function(request, response) {
  response.status(200);
  response.type("text/plain; charset=utf-8");
  response.end("Esta es la página raíz");
});
app.get("/users.html", function(request, response) {
  response.status(200);
  response.type("text/plain; charset=utf-8");
  response.end("Aquí se mostrará la página de usuarios");
});
app.listen(3000, function(err) {
  if (err) {
    console.error("No se pudo inicializar el servidor: "
      + err.message);
  } else {
    console.log("Servidor arrancado en el puerto 3000");
  }
});
```

devuelve un servidor http

Manejadores de ruta

Se crea uno por cada  
ruta que se quiera atender  
Define la respuesta que hay  
que dar a cada petición

Sabemos que es  
difícil definir tu  
futuro profesional  
¿Te ayudamos?

Máster en  
Ciberseguridad

Más info



WUOLAH

## Redireccionar una solicitud

El servidor responde al cliente (navegador) que el recurso solicitado está ubicado en otro sitio. Al recibir la respuesta de reubicación, el cliente vuelve a hacer una petición a la nueva ubicación.

Se envían mediante `response.redirect(url)`:

```
app.get("/usuarios.html", function(request, response) {  
  response.redirect("/users.html");  
});
```

Al acceder a `http://localhost:3000/usuarios.html` se obtendrá la siguiente respuesta HTTP,

```
HTTP/1.1 302 Found  
X-Powered-By: Express  
Location: /users.html  
...
```

Código 302 ⇒ Redirección

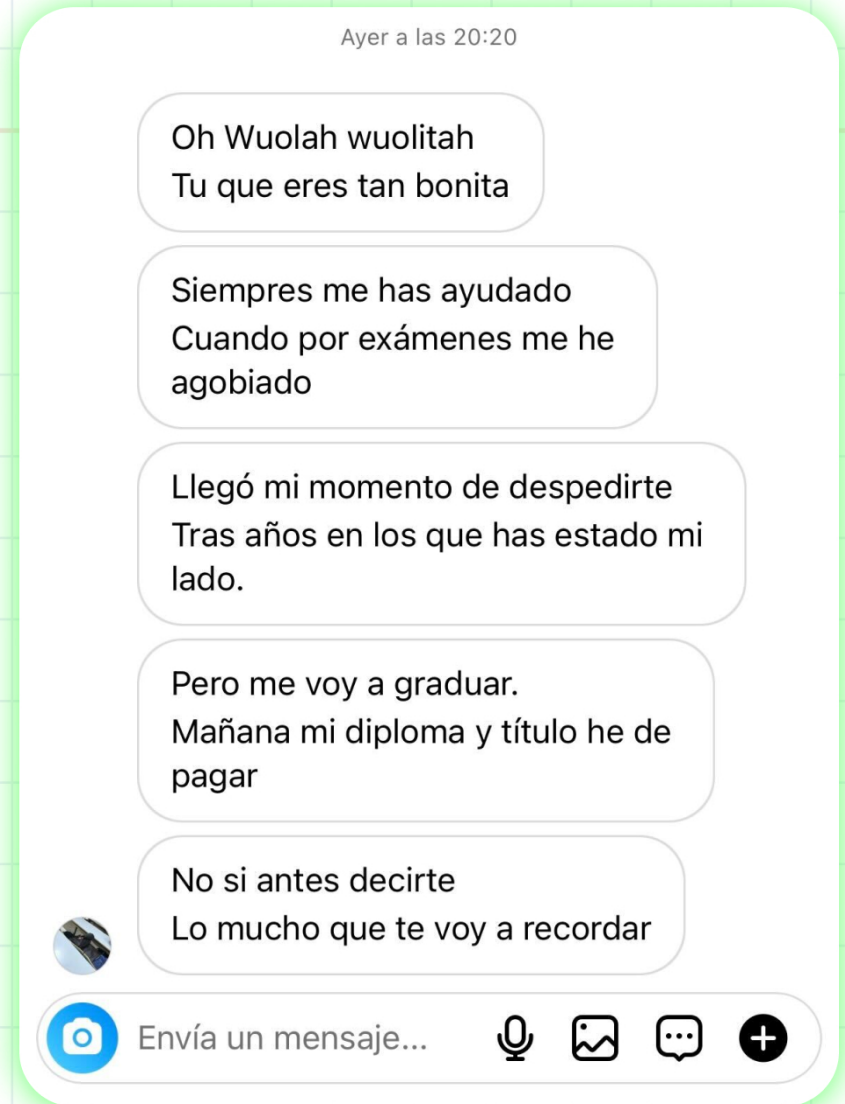
y el navegador «saltará» automáticamente a `users.html`.

**Que no te escriban  
poemas de amor  
cuando terminen la  
carrera** ▶▶▶▶▶▶

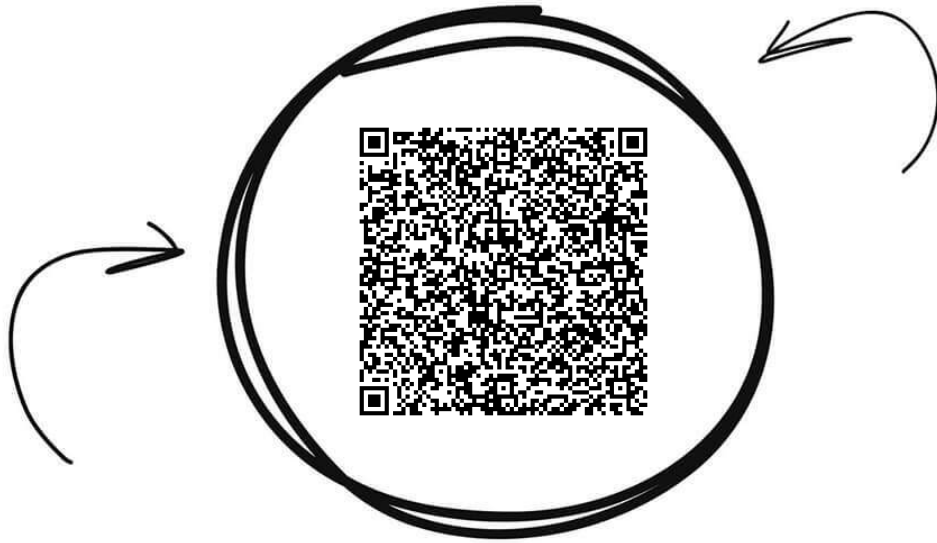
(a nosotros por suerte nos pasa)



**WUOLAH**



## Aplicaciones Web



Banco de apuntes de la UCM

**WUOLAH**



**Comparte estos flyers en tu clase y consigue más dinero y recompensas**

- 1** Imprime esta hoja
- 2** Recorta por la mitad
- 3** Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
- 4** Llévate dinero por cada descarga de los documentos descargados a través de tu QR





## Como presentar el contenido html con express

Las páginas **estáticas** son aquellas cuyo contenido es fijo, no utilizan **response.write()**, sino que se almacena el contenido en archivos independientes en el servidor.

Por ejemplo, si se tiene el fichero **bienvenido.html** situado en el directorio **public** con el siguiente contenido:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Lista de usuarios</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>¡Bienvenido!</h1>
  </body>
</html>
```

Se podría hacer:

```
app.get("/", function(request, response) {
  response.sendFile(path.join(__dirname, "public", "bienvenido.htm
});
```

En **sendFile()** el path debe ser absoluto.

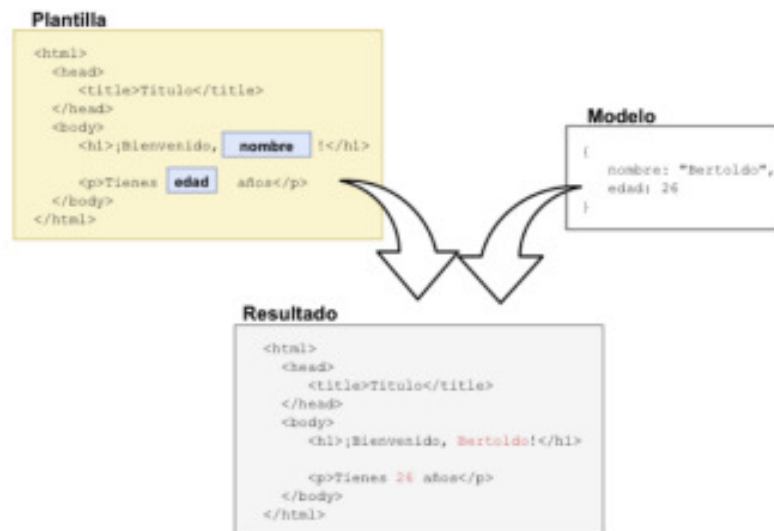
Las páginas **dinámicas** si dependen de la petición realizada. Para representar el contenido se usan **plantillas con ejs**

## PLANTILLAS CON EJS

### Plantillas

Generamos nuestros contenidos dinámicos, que son documentos html con 'huecos' que se rellenan dinámicamente con el contenido actual de la BBDD o que se genere de otra manera.

Los huecos son rellenos por el procesador de plantilla.



Que no te escriban poemas de amor  
cuando terminen la carrera ▶▶▶▶▶

(a nosotros por  
suerte nos pasa)



WUOLAH

Oh Wuolah wuolilah  
Tu que eres tan bonita

Siempre me has ayudado  
Cuando por exámenes me he  
agobiado

Llegó mi momento de despedirte  
Tras años en los que has estado mi  
lado.

Pero me voy a graduar.  
Mañana mi diploma y título he de  
pagar

No si antes decirte  
Lo mucho que te voy a recordar

## EJS

### Marcadores

↳ Programa: Contienen sentencias JS. Suelen utilizarse con bucles, condicionales...

```
<% if (!usuario.nombre) { %>
  <p>No estás identificado.</p>
<% } else { %>
  <p>Bienvenido, <%= usuario.nombre %>!</p>
<% } %>
```

↳ Expresión: Evalúan la expresión y la convierten en cadena.

En el ejemplo anterior:

```
var usuarios = ["Javier Montoro", "Dolores Vega", "Beatriz Nito"];

app.get("/users.html", function(request, response) {
  response.status(200);
  response.type("text/html");
  response.write("<html>");
  response.write("<head>");
  response.write("<title>Lista de usuarios</title>");
  response.write("<meta charset='utf-8'>");
  response.write("</head>");
  response.write("<body><ul>");
  usuarios.forEach(function(usuario) {
    response.write("<li>${usuario}</li>");
  });
  response.write("</ul></body>");
  response.end("</html>");
});
```

Se crea la plantilla `views/users.ejs`:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Lista de usuarios</title>
    <meta charset="utf-8">
  </head>
  <body>
    <ul>
      <% users.forEach(function(user) { %>
        <li><%= user %></li>
      <% }); %>
    </ul>
  </body>
</html>
```

COMO CREAR LA  
PLANTILLA USANDO  
MARCADORES

### Indicar el motor de plantillas

1- Crear el objeto aplicación

```
const path = require("path");
const express = require("express");
const app = express();
```

2- Configurar EJS como motor de plantillas

```
app.set("view engine", "ejs");
```

3- Definir el directorio de plantillas

```
app.set("views", path.join(__dirname, "views"));
```

COMO CONFIGURAR EL  
MOTOR DE PLANTILLAS  
JUNTO CON EL DIRECTORIO

WUOLAH



## Enviar al cliente una plantilla

Usamos el método `render()`, que llama al motor de plantillas, le indica que plantilla ha de coger, el modelo de cadena de datos para rellenar esa plantilla y se envía al cliente el resultado.

```
var usuarios = ["Javier Montoro", "Dolores Vega", "Beatriz Nito"];  
  
app.get("/users.html", function(request, response) {  
  response.status(200);  
  response.render("users", { users: usuarios });  
});
```

- 1 Busca la plantilla "view/users.ejs"
- 2 La variable 'users' que hay dentro de esta plantilla tomará
- 3 el valor del array 'usuarios'

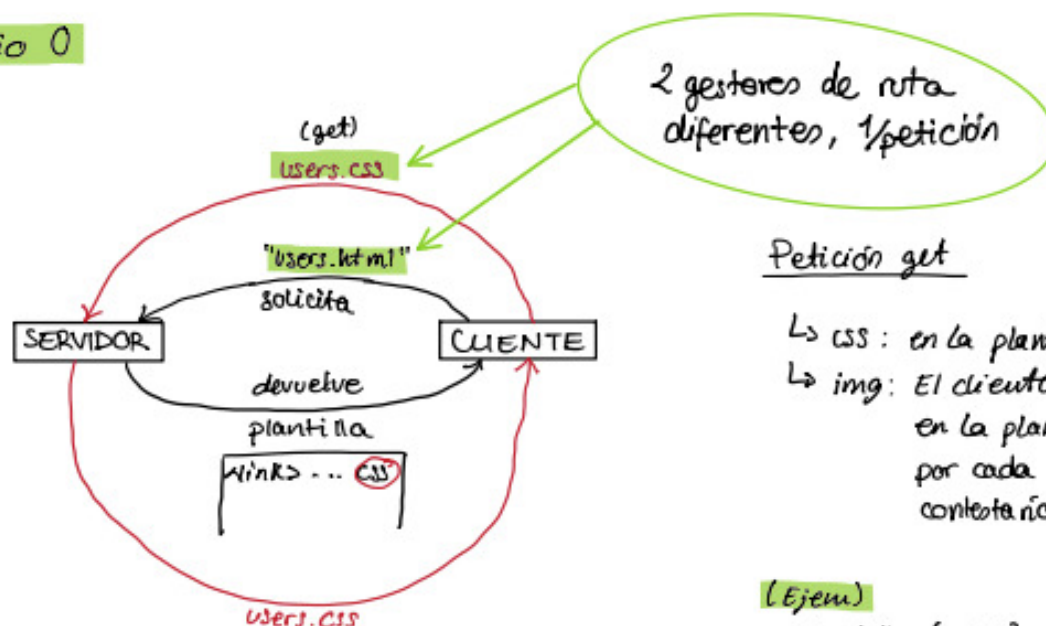
Plantilla `views/users.ejs`:

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Lista de usuarios</title>  
    <meta charset="utf-8">  
  </head>  
  <body>  
    <ul>  
      <% users.forEach(function(user) { %>  
        <li><%= user %></li>  
      <% }); %>  
    </ul>  
  </body>  
</html>
```

`http://localhost:3000/users.html`



## Ejercicio 0



### Petición get

- ↳ css: en la plantilla
- ↳ img: El cliente al detectar las img en la plantilla, haría una petición por cada una de ellas, y el servidor contestaría cada petición.

### (Ejem)

plantilla (css x2, img x5)

- petición inicial
  - 2 peticiones css
  - 5 peticiones img
- ↓
- 8 peticiones  
8 manejadores de ruta (muchas)

¿Cómo se resuelve?  
paquete static

### Estructura

- 1) Crear la aplicación → app
- 2) Establecer el motor de plantillas → app.set<
- 3) En "views" → creamos la plantilla (<link> → css)
- 4) Configurar los manejadores de ruta

Ⓐ `router.get('/users', (req, res) => {  
 res.render('users', {modelo de datos});  
})`

Ⓑ manejador para servir el fichero css

```

1  "use strict";
2
3  const path = require("path");
4
5  const express = require("express");
6  const { response } = require("express");
7  const app = express();
8
9  app.set("view engine", "ejs");
10 app.set("views", path.join(__dirname, "views"));
11
12
13 let usuarios = [
14   { nombre: "Carmen San Juan", numero: 8976 },
15   { nombre: "Adrián Lucas", numero: 8977 },
16   { nombre: "Natalia Rodríguez", numero: 8978 }
17 ]
18
19 app.get("/usuarios", function (request, response) {
20   response.render("usuarios", { usuarios: usuarios });
21 });
22
23 app.get("/users", function (request, response) {
24   response.redirect("/usuarios");
25 });
26
27 app.get("/socios", function (request, response) {
28   response.redirect("/usuarios");
29 });
30
31 app.get("/public/css/usuarios.css", function(request, response){
32   response.sendFile(path.join(__dirname, "public", "css", "usuarios.css"));
33 });
34
35 app.listen(3000, function(err) {
36   if (err) {
37     console.error("No se pudo inicializar el servidor: " +
38       err.message);
39   } else {
40     console.log("Servidor arrancado en el puerto 3000");
41   }
42 });

```

```

1  <!DOCTYPE html>
2  <html>
3
4  <head>
5    <title>Lista de usuarios</title>
6    <meta charset="utf-8">
7    <link rel="stylesheet" href="/public/css/usuarios.css" />
8  </head>
9
10 <body>
11   <h1>USUARIOS</h1>
12
13   <table>
14
15     <tr>
16       <th>Nombre</th>
17       <th>Número</th>
18     </tr>
19     <% usuarios.forEach(function(usuarios){ %>
20       <tr>
21         <td>
22           <%= usuarios.nombre %>
23         </td>
24         <td>
25           <%= usuarios.numero %>
26         </td>
27       </tr>
28       <% }); %>
29
30   </table>
31 </body>
32
33 </html>

```

# La escuela de Ciberseguridad más grande del mundo.

La formación más completa y transversal que demanda el mercado.

IMF x Deloitte.  
Smart Education

Sabemos que es  
difícil definir tu  
futuro profesional  
¿Te ayudamos?

Máster en  
Ciberseguridad

Más info



The screenshot shows a web browser window with a toolbar at the top containing various application icons. The address bar displays the URL 'H:/UCM/FDI/AW/2019-2020/express.js...'. The main content area of the browser displays the word 'USUARIOS' in a large, bold, sans-serif font. Below this title is a table with two columns: 'Nombre' and 'Número'. The table contains three rows of data, each with a name and a corresponding number.

Nombre	Número
Carmen San Juan	8976
Adrián Lucas	8977
Natalia Rodríguez	8978

WUOLAH

## MIDDLEWARE

```
const express = require("express");
const app = express();
→ Podría haber set's
let ipsBloqueadas = [ ... ];

① app.use( ... /* logger */ ... );
② app.use( ... /* ip bloqueada? */ ... );
③ app.use( ... /* ip ucm? */ ... );

app.get("/index.html", function(request, response) {
  response.status(200);
  response.type("text/plain; encoding=utf-8");
  response.write(";Hola!");
  if (request.esUCM) {
    response.write("Estás conectado desde la UCM");
  }
  response.end();
});
```

middleware  
gestores de ruta  
(Routers)



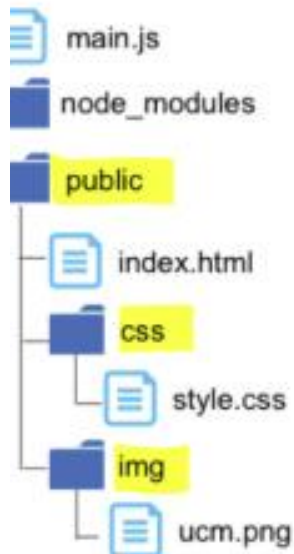


```
let ipsCensuradas = [ "147.96.81.244", "145.2.34.23" ];

app.use(function(request, response, next) {
  // Comprobar si la IP de la petición está dentro de la
  // lista de IPs censuradas.
  if (ipsCensuradas.indexOf(request.ip) >= 0) {
    // Si está censurada, se devuelve el código 401 (Unauthorized)
    response.status(401);
    response.end("No autorizado"); // TERMINA LA RESPUESTA
  } else {
    // En caso contrario, se pasa el control al siguiente middleware
    console.log("IP autorizada");
    next();
  }
});
```

static

main.js



```
"use strict";
const express = require("express");
const path = require("path");

const app = express();

// La variable ficherosEstaticos guarda el
// nombre del directorio donde se encuentran
// los ficheros estáticos:
// <directorioProyecto>/public
const ficherosEstaticos =
    path.join(__dirname, "public");

app.use(express.static(ficherosEstaticos));

app.listen(3000, function(err) {
    if (err) {
        console.error("No se pudo inicializar el servidor"
            + err.message);
    } else {
        console.log("Servidor arrancado en puerto 3000")
    }
});
```

} SIEMPRE

Que no te escriban poemas de amor  
cuando terminen la carrera ▶▶▶▶▶▶▶▶▶▶

(a nosotros por  
suerte nos pasa)



WUOLAH

Oh Wuolah wuolita  
Tu que eres tan bonita

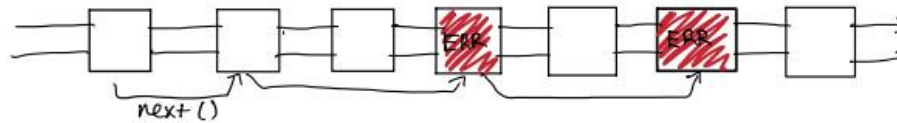
Siempre me has ayudado  
Cuando por exámenes me he  
agobiado

Llegó mi momento de despedirte  
Tras años en los que has estado mi  
lado.

Pero me voy a graduar.  
Mañana mi diploma y título he de  
pagar

No si antes decirte  
Lo mucho que te voy a recordar

manejador de errores



```
// app.js

const express = require("express");
const path = require("path");
const fs = require("fs");

const app = express();

app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "views"));

app.get("/usuarios", function(request, response, next) {
  fs.readFile("noexiste.txt", function(err, contenido) {
    if (err) {
      next(err); // Saltar al manejador de error
    } else {
      request.contenido = contenido;
    }
  });
});
// ... continúa ...
```

```
// Manejador del error

app.use(function(error, request, response, next) {
  // Código 500: Internal server error
  response.status(500);
  response.render("error500", {
    mensaje: error.message,
    pila: error.stack
  });
});

app.listen(3000, function(err) {
  if (err) {
    console.error("No se pudo inicializar el servidor: "
      + err.message);
  } else {
    console.log("Servidor arrancado en el puerto 3000");
  }
});
```

WUOLAH

# PETICIONES Y FORMULARIOS

## GET

Toda la info viaja junto con la url en la línea del navegador

*form\_get.html*

```
<form method="GET" action="procesar_get">
  <div>
    <label for="formNombre">Nombre:</label>
    <input type="text" name="nombre" id="formNombre">
  </div>
  <div>
    <label for="formEdad">Edad:</label>
    <input type="text" name="edad" id="formEdad">
  </div>
  <div>
    <input type="radio" name="sexo" value="H" id="formHombre">
    <label for="formHombre">Hombre</label>
    <input type="radio" name="sexo" value="M" id="formMujer">
    <label for="formMujer">Mujer</label>
  </div>
  <div>
    <input type="checkbox" name="fumador" value="ON"
      id="formFumador">
    <label for="formFumador">Fumador/a</label>
  </div>
  <div>
    <input type="submit" value="Enviar">
  </div>
</form>
```

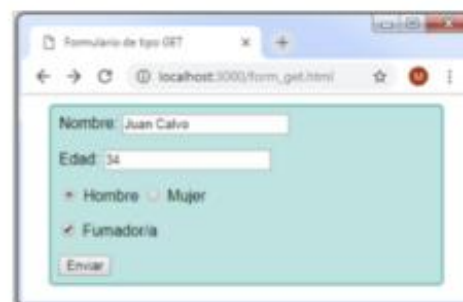
*infoForm.ejs*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Información proporcionada</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <h1>Formulario procesado</h1>
    <table>
      <tr>
        <th>Nombre:</th>
        <td><%= nombre %></td>
      </tr>
      <tr>
        <th>Edad:</th>
        <td><%= edad %></td>
      </tr>
      <tr>
        <th>Sexo:</th>
        <td><%= sexo %></td>
      </tr>
      <tr>
        <th>Fumador:</th>
        <td><%= fumador %></td>
      </tr>
    </table>
  </body>
</html>
```

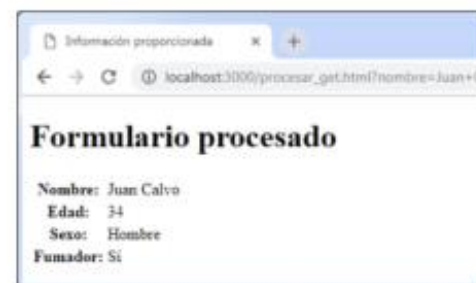
app.js

```
app.get("/procesar_get", function(request, response) {  
  let sexoStr = "No especificado";  
  switch (request.query.sexo) {  
    case "H": sexoStr = "Hombre"; break;  
    case "M": sexoStr = "Mujer"; break;  
  }  
  response.render("infoForm", {  
    nombre: request.query.nombre,  
    edad: request.query.edad,  
    sexo: sexoStr,  
    fumador: (request.query.fumador === "ON" ? "Sí" : "No")  
  });  
});
```

FORMULARIO ENVIADO



FORMULARIO PROCESADO



WUOLAH

si lees esto me debes un besito



## POST

form-get → `<form method="POST" action="procesar_post">`

app.js → `app.post("/procesar_post"...`

`request.body.data`

### La opción extended

```
<form method="POST" action="procesar_post">
  ...
  <input type="text" name="datosPersonales[nombre]"
    id="formNombre">
  ...
  <input type="text" name="datosPersonales[edad]"
    id="formEdad">
```

extended: false

```
{
  "datosPersonales[nombre]" : "Germán",
  "datosPersonales[edad]" : "Hernando Coello",
  sexo : "H"
}
```

extended: true

```
{
  datosPersonales : {
    nombre : "Germán",
    edad : "Hernando Coello"
  },
  sexo : "H"
}
```

# La escuela de Ciberseguridad más grande del mundo.

La formación más completa y transversal que demanda el mercado.

IMF x Deloitte.  
Smart Education

Sabemos que es  
difícil definir tu  
futuro profesional  
¿Te ayudamos?

Máster en  
Ciberseguridad

Más info

## RUTAS PARAMÉTRICAS

```
app.get("/usuarios/:id", function(request, response) {  
  response.status(200);  
  response.render("usuario", { ident: request.params.id });  
});
```

## views/usuario.ejs

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Saludo</title>  
    <meta charset="utf-8">  
  </head>  
  <body>  
    <h1>¡Bienvenido, <%= ident %>!</h1>  
  </body>  
</html>
```



WUOLAH

## Ejercicio 2

A)

```
19 let usuarios = ["Javier Montoro", "Dolores Vega", "Beatriz Nito"];
20
21 app.get("/users.html", function (request, response) {
22   response.status(200);
23   response.render("users", { users: usuarios });
24 }); // se llama al motor de plantillas y se le pasan los datos
25
26 app.get("/borrar/:indice", function (request, response) {
27   response.status(200);
28   usuarios.splice(request.params.indice, 1); // elem que aparece en ese indice
29   response.redirect("/users.html"); // recargamos la página para reflejar los cambios
30 });
```

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <title>Lista de usuarios</title>
6   <metacharset="utf-8">
7 </head>
8
9 <body>
10   <ul>
11     <%users.forEach(function(user, index) { %>
12       <li>
13         <%=user %>
14         <a href="/borrar/<%=index%>"> Borrar</a>
15       </li>
16       <% }); %>
17   </ul>
18 </body>
19
20 </html>
```

elem actual param 1  
indice elem actual param 2  
param 3 Array completo

B) body - parzer

```
32 app.get("/users.html", function (request, response) {
33     response.status(200);
34     response.render("users8", { users: usuarios });
35 });
36
37 app.post("/borrar", function (request, response) {
38     response.status(200);
39     usuarios.splice(request.body.user, 1);
40     response.redirect("/users.html");
41 });
```

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5     <title>Lista de usuarios</title>
6     <metacharset="utf-8">
7 </head>
8
9 <body>
10     <ul>
11         <%users.forEach(function(user, index) { %>
12             <li>
13                 <%=user %>
14                 <form style="display: inline-block; padding: 5px" method="POST" action="/borrar">
15                     <input type="hidden" name="user" value="<%=index%>"> Se envia información
16                     <input type="submit" value="Borrar"> oculta de cada usuario
17                 </form>
18             </li>
19             <% }); %>
20         </ul>
21 </body>
22
23 </html>
```

## ROUTERS

Sirven para estructurar la aplicación agrupando distintas rutas por temas, y así poder crear un core de aplicación que nos permita escalar la aplicación de forma sencilla

```
const miRouter = express.Router();

miRouter.get("/crear_usuario", function(request, response) {
  console.log("Creando usuario.");
  response.end();
});

miRouter.get("/buscar_usuario", function(request, response) {
  console.log("Buscando usuario");
  response.end();
});

module.exports = miRouter;
```

Es posible incorporar un router en otra aplicación indicando la ruta sobre la que se quiere montar:

`app.use(ruta, router)`

```
const app = express();
const miRouter = require("./miRouter");
app.use("/usuarios", miRouter);
```

Montar sobre la ruta /usuarios

En este caso, las rutas definidas en el router estarán accesible a través de las siguientes URLs:

`http://localhost:3000/usuarios/crear_usuario.html`

`http://localhost:3000/usuarios/buscar_usuario.html`



Que no te escriban poemas de amor  
cuando terminen la carrera ▶▶▶▶▶

(a nosotros por  
suerte nos pasa)



WUOLAH

Oh Wuolah wuolita  
Tu que eres tan bonita

Siempre me has ayudado  
Cuando por exámenes me he  
agobiado

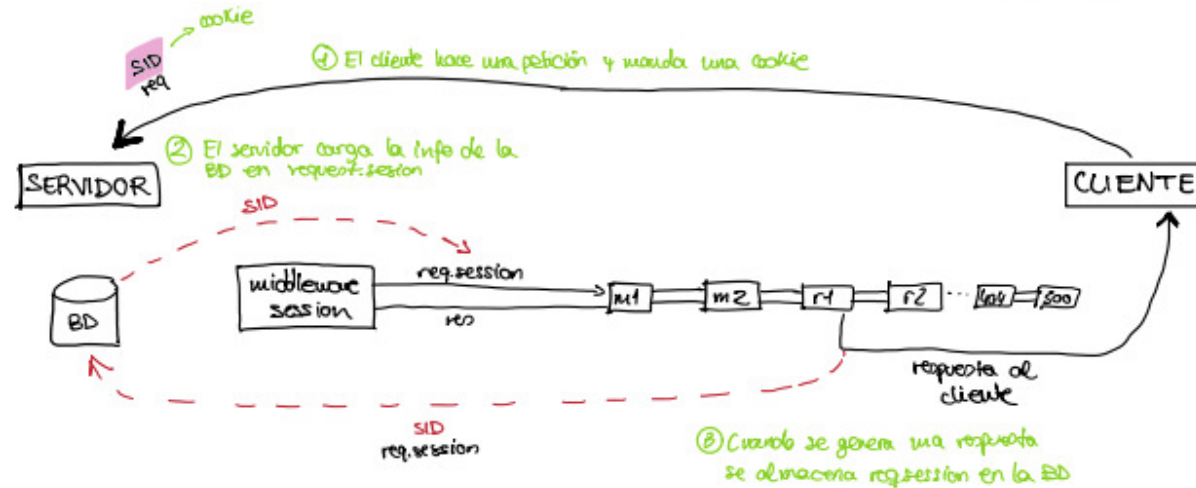
Llegó mi momento de despedirte  
Tras años en los que has estado mi  
lado.

Pero me voy a graduar.  
Mañana mi diploma y título he de  
pagar

No si antes decirte  
Lo mucho que te voy a recordar

## SESIONES

Cuando se termina de atender una petición, toda la información guardada en `request.session` se almacena en el servidor, para que cuando llegue una nueva petición, se recargue toda esa información en `request.session`.



Con el módulo `express-mysql-session` almacenamos la información de sesión en una `BBDD` en vez de en memoria.

WUOLAH

## DISEÑO AVANZADO DE PLANTILLAS

**<% - %>**

```
<p> <%= msg %> </p>
<p> <%- msg %> </p>
```

Si la variable **msg** toma el valor  
"Esto es **<b>importante</b>**"  
tenemos:

```
<p> Esto es &lt;b&gt;importante&lt;/b&gt; </p>
<p> Esto es <b>importante</b> </p>
```

### Subplantillas

```
<body>
  <%- include("header") %>
  <div>
    <ul>
      <% usuarios.forEach(function(u) { %>
        <%- include("userView", { usuario: u }) %>
        <% }); %>
      </ul>
    </div>
    <%- include("footer") %>
</body>
```

views/header.ejs

```
<header>
  Cabecera de la página
</header>
```

views/footer.ejs

```
<footer>
  Pie de página
</footer>
```

views/userView.ejs

```
<li><%= usuario.apellidos %>, <%= usuario.nombre %></li>
```

## Variables globales

app.locals / response.locals

ejem: (...).locals.V1

↳ plantilla utiliza  
solo V1 para llamar  
a la variable

## Diferencias

### VIDA

(¿quién vive?)

### ÁMBITO

app.locals

vida del  
servidor

todos los  
clientes

response.locals

vida de la  
petición

propio del  
cliente  
(la petición  
del cliente)