

## Apuntes SWEB:

### Tabla de contenido

Pasos para resolver el Apartado 1: .....	1
<a href="https://www.geeksforgeeks.org/how-to-create-global-variables-accessible-in-all-views-using-express-nodejs/">https://www.geeksforgeeks.org/how-to-create-global-variables-accessible-in-all-views-using-express-nodejs/</a> .....	2
1. Instala dotenv para manejar variables de entorno .....	9
2. Crea un archivo .env en la raíz del proyecto .....	9
3. Crea un archivo config.js para cargar la configuración .....	10
4. Actualiza app.js para usar la configuración centralizada .....	¡Error! Marcador no definido.
5. Acceder a la configuración en otros archivos.....	¡Error! Marcador no definido.
Beneficios de esta configuración.....	¡Error! Marcador no definido.
Prueba final.....	¡Error! Marcador no definido.
1. Estructura general de Express .....	¡Error! Marcador no definido.
2. Detalle de cada carpeta y archivo .....	¡Error! Marcador no definido.
1. /bin/www .....	¡Error! Marcador no definido.
2. /routes .....	¡Error! Marcador no definido.
3. /views .....	¡Error! Marcador no definido.
4. /public .....	¡Error! Marcador no definido.
5. /database .....	¡Error! Marcador no definido.
6. app.js .....	¡Error! Marcador no definido.
7. config.js .....	¡Error! Marcador no definido.
Flujo de la Aplicación .....	¡Error! Marcador no definido.
Resumen.....	¡Error! Marcador no definido.

CSS: <https://www.w3schools.com/css/default.asp> o bootstrap

<https://developer.mozilla.org/es/docs/Web/HTTP/Cookies>

(registro) <https://www.freecodecamp.org/espanol/news/como-construir-una-forma-de-registro-con-etiquetas-flotantes-y-transiciones-usando-solamente/>

(localStorage) <https://www.freecodecamp.org/espanol/news/como-usar-localstorage-en-javascript/>

(cookies) <https://expressjs.com/en/starter/examples.html>

<https://www.digitalocean.com/community/tutorials?q=login>

<https://www.geeksforgeeks.org/nodejs/?ref=outind>

### Pasos para resolver el Apartado 1:

De momento no tenemos del todo claro cuál va a ser el nombre definitivo de la tienda y ahora mismo lo tenemos **hardcodeado en cada ruta**. Si quisiéramos cambiar el nombre tendríamos que modificar esos archivos y cualquier futura ruta que creemos. **Modifique la aplicación para definir en un único sitio el nombre de la tienda** (ahora mismo “Embutidos León”), que este sea accesible en las vistas y elimine el nombre que aparece ahora en las rutas.

<https://www.geeksforgeeks.org/how-to-create-global-variables-accessible-in-all-views-using-express-nodejs/>

### 1. Crear un archivo de configuración:

- Crea un archivo llamado `config.js` en la raíz del proyecto.
- Define el nombre de la tienda en este archivo:

```
module.exports = {  
  shopName: "Embutidos León",  
};
```

### 2. Hacer accesible el nombre en las vistas:

- En el archivo principal del servidor (`app.js`), importa el archivo `config.js` y utiliza un middleware para hacerlo accesible en todas las vistas:

```
const config = require('./config');  
  
app.use((req, res, next) => {  
  res.locals.title = config.shopName;  
  next();  
});
```

### 3. Actualizar las vistas:

- En cada archivo de vista donde se utiliza el nombre de la tienda, reemplázalo por la variable `title` (viene así por defecto):

```
<%- include("header", {}) %>  
<div class="container">  
  <h1><%= title %></h1>  
  <p>Bienvenido a la tienda <%= title %></p>  
  <p>Aquí podrá encontrar los mejores embutidos y productos gastronómicos  
de la región.</p>  
  <p>Visite <a href="/tienda">nuestra tienda</a> para ver los productos  
que tenemos.</p>  
    
</div>  
<%- include("footer", {}) %>
```

### 4. Eliminar el nombre hardcodeado en las rutas:

- Si tienes rutas que están usando el nombre de la tienda, reemplázalo por la variable global accesible desde `res.locals.shopName`. Por ejemplo en `index.js`:

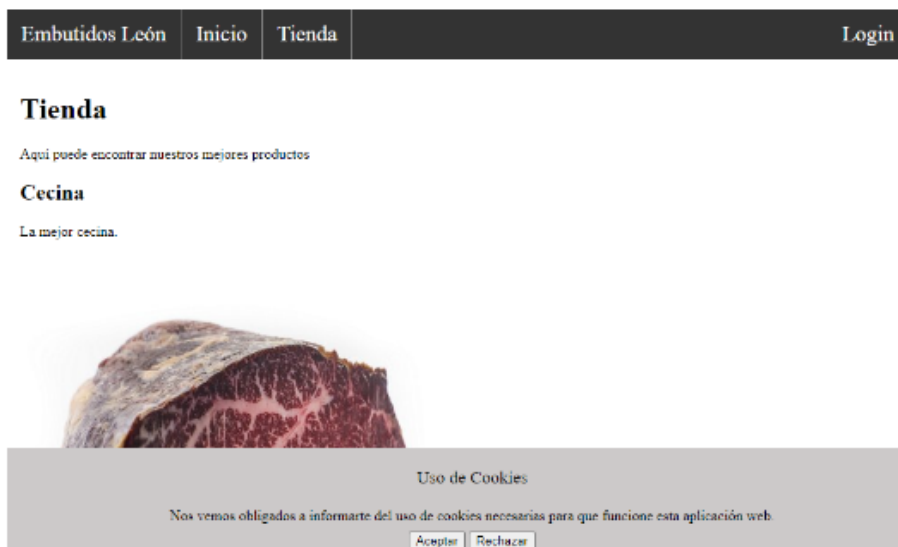
```
1. const express = require('express');  
2. const router = express.Router();  
3.  
4. /* GET home page. */  
5. router.get('/', function(req, res, next) {  
6.   res.render('index', {user:req.session.user, title:res.locals.title});
```

```
7. });  
8.  
9. module.exports = router;  
10.
```

## Apartado 2

Hay una cuestión muy importante a la que nos obliga la normativa europea y es sobre el uso de cookies en la web. Tenemos que informar al usuario de que estamos usando cookies en nuestra página y que las acepte. Para ello complete las siguientes tareas:

1. Cree una sección que aparezca en la parte inferior de todas las páginas con las siguientes características visuales:
  - Que aparezca un título con un mensaje informando al usuario del uso de cookies.
  - Que tenga un botón de aceptar y otro de rechazar.
  - Que ocupe todo el ancho de la pantalla, tenga el texto y los botones centrados y que esté por encima de cualquier contenido que haya en la página. Tiene que estar siempre visible.
  - Usa CSS puro para crear esta sección, no uses ningún framework.
  - Puedes encontrar una referencia de cómo tendría que quedar en el anexo:



<https://programadorwebvalencia.com/Javascript-ejemplo-aviso-o-cartel-de-cookie/>

Accedo al footer.ejs:

Creo los botones y html:

```
</body>  
<section class="cookies">  
  <h2 class="cookies__titulo">¿Aceptas nuestras Cookies?</h2>  
  <p class="cookies__texto">Usamos cookies para mejorar tu experiencia en la web.</p>  
  <div class="cookies__botones">  
    <button class="cookies__boton cookies__boton--si">Si</button>  
    <button class="cookies__boton cookies__boton--no">No</button>
```

```
</div>
</section>
```

Luego aplico estilos caseros:

```
<style>
  .cookies {
    position: fixed;
    bottom: 0;
    left: 0;
    right: 0;
    background: black;
    color: white;
    font-family: arial;
    text-align: center;
  }
  /*ESTO LO COMENTO PORQUE NO LO PIDE EN SU IMAGEN
  .cookies__boton {
    background: initial;
    border: 2px solid white;
    padding: 1rem;
    font-size: 1rem;
    color: white;
    width: 5rem;
    text-align: center;
  } */

  button {
    padding: 1rem;
    width: 5rem;
    margin-bottom: 15px;
  }
</style>
```

2. Si el usuario le da a cancelar, redirígele a la página de google.es.

Le añado la referencia de Google:

```
<button class="cookies__boton cookies__boton--si"
onclick="location.href='https://www.google.es'">Rechazar</button>
```

3. Si el usuario acepta las cookies, realice lo siguiente:

- Cierre el mensaje.
- Guarde información en la sesión para que, si sigue visitando otras páginas, no se vuelva a mostrar el mensaje.

- Si el usuario está logeado, guarde en su perfil (en la base de datos) información de que ha aceptado las cookies.

Copiamos y pegamos la lógica del JS:

```
<!-- No borrar, aquí se generarán todas las etiquetas <script> si acepta el
usuario -->
<div id="nuevosScripts"></div>

<script defer>
let cookies = () => {

//=====
// COOKIES
//=====

//-----
// Configuración
//-----
const urlsScriptsCookies = ['https://analytics.google.com',
'https://facebook.com'];

function contenidoScriptsCookies () {
    //////////// ¿Google Analíticas? ////////////
    //////////// ¿Facebook Pixel? ////////////
    //////////// ¿Admob? ////////////
    //////////// etc ////////////
}

//-----
// Variables
//-----
const seccionCookie = document.querySelector('section.cookies');
const cookieSi = document.querySelector('.cookies__boton--si');
const cookieNo = document.querySelector('.cookies__boton--no');
const nuevosScripts = document.querySelector('#nuevosScripts');

//-----
// Funciones
//-----

/**
 * Método que oculta la sección de Cookie para siempre
 */
function ocultarCookie() {
    // Borra la sección de cookies en el HTML
    seccionCookie.remove();
}
```

```
/**
 * Método que marca las cookies como aceptadas
 */
function aceptarCookies() {
    // Oculta el HTML de cookies
    ocultarCookie();
    // Guarda que ha aceptado
    localStorage.setItem('cookie', true);
    // Tu código a ejecutar si aceptan las cookies
    ejecutarSiAcepta();
}

/**
 * Método que marca las cookies como denegadas
 */
function denegarCookies() {
    // Oculta el HTML de cookies
    ocultarCookie();
    // Guarda que ha aceptado
    localStorage.setItem('cookie', false);
    // Redirige a Google
    window.location.href = 'https://www.google.es';
}

/**
 * Método que ejecuta tu código si aceptan las cookies
 */
function ejecutarSiAcepta() {
    // Crea los <script>
    urlsScriptsCookies.forEach((url) => {
        const nuevoScript = document.createElement('script');
        nuevoScript.setAttribute('src', url);
        nuevosScripts.appendChild(nuevoScript);
    });
    // Lanza los códigos
    contenidoScriptsCookies();
}

/**
 * Método que inicia la lógica
 */
function iniciar() {
    // Comprueba si en el pasado el usuario ha marcado una opción
    if (localStorage.getItem('cookie') !== null) {
        if (localStorage.getItem('cookie') === 'true') {
            // Aceptó
            aceptarCookies();
        } else {
            // No aceptó
            denegarCookies();
        }
    }
}
```

```

    }
  }

  //-----
  // Eventos
  //-----
  cookieSi.addEventListener('click',aceptarCookies, false);
  cookieNo.addEventListener('click',denegarCookies, false);

  return {
    iniciar: iniciar
  }
}

// Activa el código. Comenta si quieres desactivarlo.
cookies().iniciar();

</script>

```

Posteriormente, añadimos lógica para guardar en la bbdd:

```

async function aceptarCookies() {
  // Oculta el HTML de cookies
  ocultarCookie();
  // Guarda que ha aceptado
  localStorage.setItem('cookie', true);
  // Lógica para guardar en la base de datos si está logeado
  const isUserLoggedIn = document.body.dataset.userLoggedIn ===
'true';
  if (isUserLoggedIn) {
    try {
      const response = await fetch('/cookies/accept', { method:
'POST' });
      if (!response.ok) {
        console.error('Error al guardar la aceptación de
cookies en la base de datos');
      }
    } catch (error) {
      console.error('Error:', error);
    }
  }
  // Tu código a ejecutar si aceptan las cookies
  ejecutarSiAcepta();
}

```

**Backend para guardar cookies en la base de datos:** Si el usuario está logeado, guarda la decisión de aceptar cookies en la base de datos. Crea una ruta en tu backend:

Añadimos esta funcionalidad:

```

users.data[username] = {username, hash, last_Login: new Date().toISOString,
  cookiesAccepted: false, // Agregamos esta propiedad};

```

```
    };\n  });\n};
```

Y al final del mismo archivo user.models.js:

```
// Método para marcar cookies como aceptadas\nusers.acceptCookies = function (username) {\n  if (!users.data.hasOwnProperty(username)) {\n    throw new Error(`El usuario ${username} no existe.`);\n  }\n  users.data[username].cookiesAccepted = true;\n};\n\n// Método para verificar si las cookies han sido aceptadas\nusers.hasAcceptedCookies = function (username) {\n  if (!users.data.hasOwnProperty(username)) {\n    return false;\n  }\n  return users.data[username].cookiesAccepted;\n};
```

Ahora, necesitamos una ruta que llame al método acceptCookies del modelo cuando un usuario acepta cookies.

### Crear la ruta routes/cookies.js:

```
const express = require('express');\nconst router = express.Router();\nconst database = require('../database');\n\n// Ruta para aceptar cookies\nrouter.post('/accept', (req, res) => {\n  if (req.session.user && req.session.user.username) {\n    try {\n      database.user.acceptCookies(req.session.user.username);\n      res.status(200).json({ message: 'Cookies aceptadas' });\n    } catch (err) {\n      console.error(err);\n      res.status(500).json({ error: 'No se pudieron aceptar las cookies' });\n    }\n  } else {\n    res.status(401).json({ error: 'Usuario no autenticado' });\n  }\n});\n\nmodule.exports = router;
```

Añadimos la nueva ruta en app.js:



```
const cookiesRouter = require('./routes/cookies');

// Otras rutas...

app.use('/cookies', cookiesRouter);
```

4. Cuando un usuario hace login en nuestra página, si en la sesión actual ya ha aceptado cookies o en su perfil hemos guardado previamente información de que ha aceptado las cookies, no debería mostrarse el mensaje. En caso contrario sí que debería aparecer.

Para ver resultado casi perfecto, acceder al repo...

Sockets.io

[https://www.luisllamas.es/como-usar-socketio-con-nodejs/?utm\\_source=programandoweb&utm\\_medium=social&utm\\_campaign=nodejs](https://www.luisllamas.es/como-usar-socketio-con-nodejs/?utm_source=programandoweb&utm_medium=social&utm_campaign=nodejs)  
[https://programandoweb.net/sencillo-chat-con-node-js-express-socket-htaccess/?utm\\_source=programandoweb&utm\\_medium=social&utm\\_campaign=nodejs](https://programandoweb.net/sencillo-chat-con-node-js-express-socket-htaccess/?utm_source=programandoweb&utm_medium=social&utm_campaign=nodejs)

Para **centralizar la configuración** y utilizar variables de entorno, vamos a seguir estos pasos. Esto mejora la seguridad y el mantenimiento de tu proyecto.

---

## 1. Instala `dotenv` para manejar variables de entorno

En la terminal, ejecuta el siguiente comando para instalar **dotenv**:

```
npm install dotenv
```

---

## 2. Crea un archivo `.env` en la raíz del proyecto

Aquí es donde almacenarás los valores críticos como el puerto, nombre de la tienda y credenciales de la base de datos.

**Contenido de `.env`:**

```
PORT=4000
SHOP_NAME=Embutidos León
DB_USER=admin
DB_PASSWORD=supersecurepassword
DB_HOST=localhost
DB_NAME=carrito_db
```

```
SECRET_KEY=UnaFraseMuySecreta
```

---

### 3. Crea un archivo `config.js` para cargar la configuración

Este archivo centraliza todas las configuraciones y las lee desde el archivo `.env`.

**config.js:**

```
require('dotenv').config();

module.exports = {
  port: process.env.PORT || 3000,
  shopName: process.env.SHOP_NAME || 'Tienda Default',
  db: {
    user: process.env.DB_USER || 'root',
    password: process.env.DB_PASSWORD || '',
    host: process.env.DB_HOST || 'localhost',
    name: process.env.DB_NAME || 'test_db'
  },
  sessionSecret: process.env.SECRET_KEY || 'defaultSecret'
};
```