

Desarrollo Backend

Bienvenid@s

Manejo de Cookies y Sesiones

Clase 19



Pon a grabar la clase



Temario

- Qué son las cookies
- Creación y gestión de cookies
 - Ejemplo funcional
 - Configuración de cookies
 - Leer una cookie
 - Eliminar una cookie
 - Consideraciones de seguridad y privacidad
- Prácticas asociadas al manejo de cookies
 - Espacio de prácticas (*consigna*)
 - Puesta en común de la práctica



Qué son las cookies

Qué son las cookies

Las cookies son pequeños archivos de texto que se utilizan para almacenar información en el navegador web del usuario. Estos archivos se generan y envían desde el servidor web al navegador durante la interacción entre el usuario y un sitio web.

Una vez que se almacenan en el navegador, pueden ser enviadas de vuelta al servidor con cada solicitud realizada por el usuario.



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Qué son las cookies

Desempeñan un papel importante en el desarrollo web, ya que permiten que los sitios web recuerden información sobre el usuario y su interacción con el sitio.

Esto incluye datos como preferencias de idioma, información de inicio de sesión, artículos agregados al carrito de compras, entre otros.



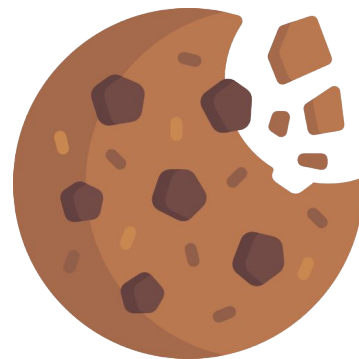
UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

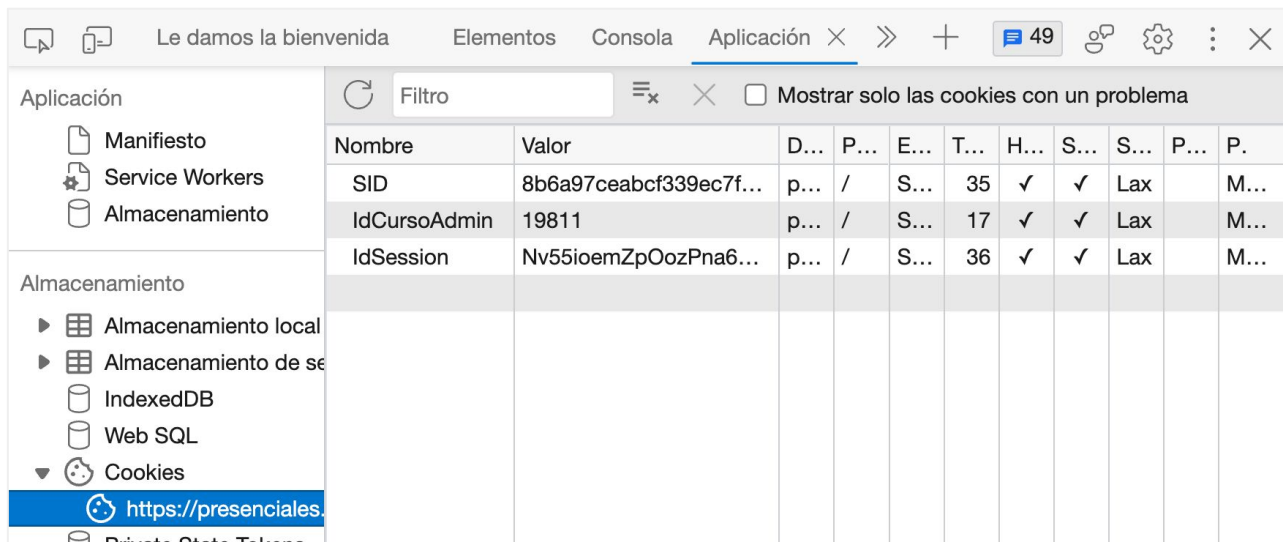
Qué son las cookies

Constan de un nombre, un valor, y otros atributos opcionales, como la fecha de expiración y la ruta del sitio web que las creó. Estos atributos ayudan a controlar cómo se almacenan y acceden las cookies.

Además, pueden ser **utilizadas para diversos propósitos** como **personalizar la experiencia del usuario**, realizar **seguimiento de actividades y comportamientos**, mostrar **anuncios dirigidos**, recordar **información de inicio de sesión** y mantener el **estado de la sesión**.



Qué son las cookies



The screenshot shows the Chrome DevTools interface with the 'Aplicación' (Application) tab selected. The left sidebar shows the 'Almacenamiento' (Storage) section, with 'Cookies' expanded and the URL 'https://presenciales.untref.edu.ar' selected. The main panel displays a table of cookies.

Nombre	Valor	D...	P...	E...	T...	H...	S...	S...	P...	P...
SID	8b6a97ceabcf339ec7f...	p...	/	S...	35	✓	✓	Lax		M...
IdCursoAdmin	19811	p...	/	S...	17	✓	✓	Lax		M...
IdSession	Nv55ioemZpOozPna6...	p...	/	S...	36	✓	✓	Lax		M...

Desde cualquier navegador web, podemos acceder a algún sitio web, luego **DevTools > Application**, y en el panel lateral denominado **Almacenamiento**, encontraremos un apartado llamado **Cookies**.

Crear una cookie

Crear una cookie

Dentro de **DevTools > Application > Almacenamiento > Cookies**,
pulsemos la tecla **ESC** (*escape*) para
visualizar en la parte inferior de
DevTools, el panel **Console**.

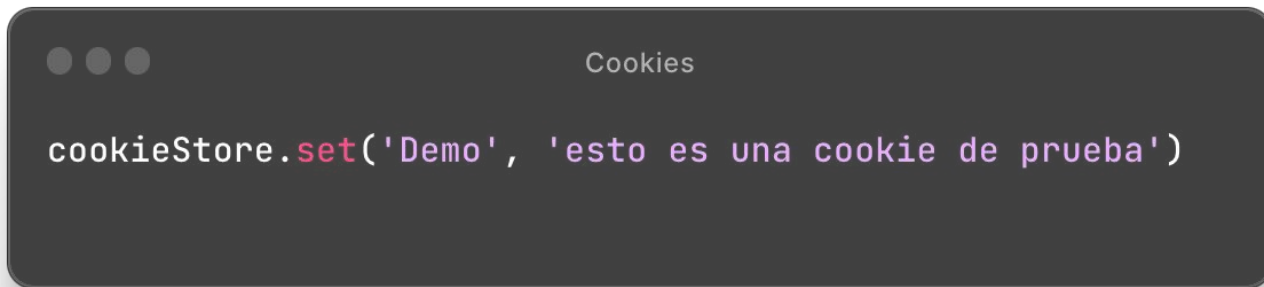
Luego, en este mismo panel,
podremos escribir una cláusula JS
que nos permitirá crear una cookie de
forma rápida y concisa.



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Crear una cookie



```
cookieStore.set('Demo', 'esto es una cookie de prueba')
```

Allí podemos definir un comando simple, para crear de forma rápida una cookie. Ejecutado el mismo, refrescamos el panel superior de cookies, y veremos la misma creada con el nombre y valor indicados.

Crear una cookie

Si comparamos la misma con las otras cookies que puedan existir previamente en este entorno, veremos que las otras opciones no dejan textos expuestos en formato plano, sino que los codifican.

Filtro

Mostrar solo las cookies con un problema

Nom...	Valor	Domain	Path	Exp...	T...	H...	S	S.	P...	P
SID	8b6a97ceabcf339e...	prese...	/	Sesi...	35	✓	✓	L..		M...
IdCu...	19811	prese...	/	Sesi...	17	✓	✓	L..		M...
Demo	esto es una cookie ...	prese...	/	Sesi...	32		✓	S..		M...

Cookie Value

☐ Show URL-decoded

esto es una cookie de prueba

Podemos, para ello, utilizar algún mecanismo como **Base64**. No es privado ni encriptado, pero al menos aplica una mínima codificación sobre la información generada.

Crear una cookie



```
const texto = 'texto a encriptar';  
const textoEnBase64 = Buffer.from(texto).toString('base64');  
  
console.log(textoEnBase64);
```

El módulo **Buffer**, de Node.js, nos proporciona un método llamado **toString()** que acepta el tipo de codificación como argumento, y podemos usarlo con **'base64'** para codificar texto en este estándar.

Crear una cookie



Cookies

```
const textoDecodificado = Buffer.from(textoEnBase64, 'base64').toString();  
  
console.log(textoDecodificado);
```

Luego, para decodificar texto en **base64** podemos utilizar el método **Buffer.from()** con la codificación 'base64' combinando, a posteriori, el método **toString()** sin argumentos para obtener el texto original.

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Gestión de cookies

Gestión de cookies

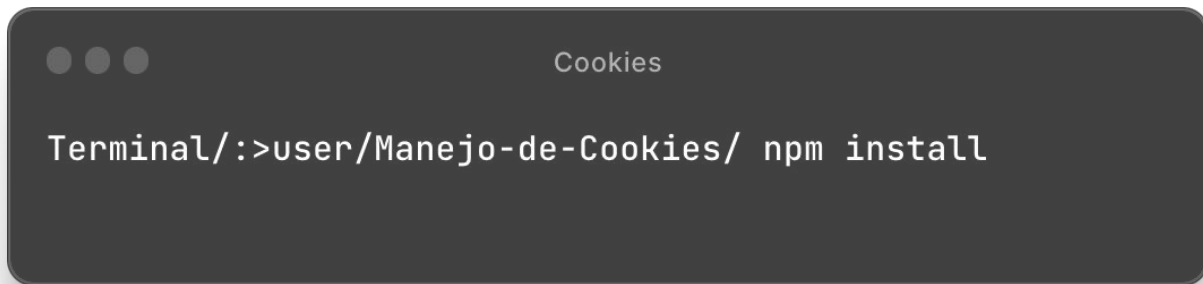
Valor	Descripción
Nombre	Es el identificador único de la cookie. Se utiliza para acceder y manipular la cookie desde el lado del servidor.
Valor	Es la información o datos que se almacenan en la cookie. Puede ser cualquier tipo de dato, como una cadena de texto, un número o incluso una estructura de datos en formato JSON.
Dominio	Es el dominio del sitio web que creó la cookie. La cookie solo se enviará al servidor cuando se haga una solicitud al dominio específico.
Ruta	Es la ruta dentro del dominio en la que la cookie es válida. Por ejemplo, si la ruta es <code>"/admin"</code> , la cookie solo se enviará al servidor cuando la URL comience con <code>"/admin"</code> .
Expiración	Es la fecha y hora en la que la cookie caduca y deja de ser válida. Después de la expiración, el navegador ya no enviará la cookie al servidor.
Segura	Es un valor booleano que indica si la cookie debe enviarse sólo a través de conexiones HTTPS seguras. Esto proporciona una capa adicional de seguridad para proteger la información sensible.
HttpOnly	Valor booleano que indica si la cookie solo debe ser accesible a través del protocolo HTTP y no a través de scripts en el lado del cliente, como JavaScript. Esto ayuda a mitigar ataques de secuestro de cookies.

Aquí, algunos de los valores que componen una cookie.

Gestión de cookies

Veamos proyecto en Node.js, llamado **Manejo-de-cookies**, en donde trabajamos el manejo de Cookies desde una aplicación backend.

Descarga o accede al proyecto publicado en el Campus, ábrelo con VS CODE, e inicializa el mismo para que se instalen las dependencias correspondientes.

A dark-themed terminal window with the title 'Cookies' and three window control buttons (red, yellow, green) in the top-left corner. The terminal text shows the command 'Terminal/:>user/Manejo-de-Cookies/ npm install' being entered.

```
Terminal/:>user/Manejo-de-Cookies/ npm install
```

```
require('dotenv').config()
const express = require('express')
const cookieParser = require('cookie-parser')
const vistaPrincipal = require('./views/views.js')
const PORT = process.env.PORT
const app = express()

app.use(cookieParser())

app.get("/", (req, res) => {
  const datosEnCookie = new Date()
  res.cookie('nodeCookie', datosEnCookie)
  res.status(200).send(vistaPrincipal)
})

app.get('/leer-cookie', (req, res) => {
  const fechaAcceso = req.cookies.nodeCookie || 'No hay registro previo'
  res.status(200).send('Fecha del último acceso: ' + fechaAcceso)
})

app.get('/eliminar-cookie', (req, res) => {
  const fechaAcceso = req.cookies.nodeCookie || 'No hay registro previo'
  if (fechaAcceso !== 'No hay registro previo') {
    res.clearCookie('nodeCookie')
    res.status(200).send('Se eliminó la cookie con el registro: ' + fechaAcceso)
  } else {
    res.status(406).send('No se encontró una cookie para eliminar.')
  }
})

app.listen(PORT, () => {
  console.log('Server listening on port:', PORT)
})
```

Ejemplo funcional completo

Aquí tenemos un ejemplo funcional completo, con tres rutas:

/ (crear una cookie)

/leer-cookie (Leer la cookie)

/eliminar-cookie (eliminar la cookie)

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Ejemplo funcional completo



Este ejemplo funcional, se puede testear de forma directa, utilizando un **navegador web + Developer Tools > Application > Cookies**, para ver cómo se genera la misma, y qué información almacena internamente (*fecha y hora en nuestro caso*)

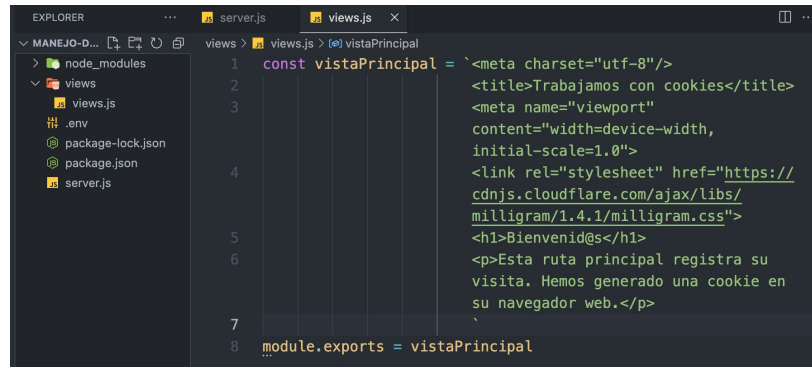
Activa el servidor web desde la Terminal, y accede a las rutas desde el navegador web con DevTools abierto.

Configuración de cookies

En el archivo **server.js** encontramos la referencia a **dotenv**, el framework **Express**, la dependencia **cookie-parser**, y una vista HTML creada con template strings en un archivo JS dedicado, importada como **vistaPrincipal**.
Por último, definimos el Middleware **cookieParser** para que trabaje internamente con nuestros endpoints.

```
require('dotenv').config()
const express = require('express')
const cookieParser = require('cookie-parser')
const vistaPrincipal = require('./views/views.js')
const PORT = process.env.PORT
const app = express()

app.use(cookieParser())
```



```
1 const vistaPrincipal = `<meta charset="utf-8"/>
2 <title>Trabajamos con cookies</title>
3 <meta name="viewport"
4   content="width=device-width,
5   initial-scale=1.0">
6 <link rel="stylesheet" href="https://
7   cdnjs.cloudflare.com/ajax/libs/
8   milligram/1.4.1/milligram.css">
  <h1>Bienvenid@s</h1>
  <p>Esta ruta principal registra su
  visita. Hemos generado una cookie en
  su navegador web.</p>
  `
  module.exports = vistaPrincipal
```

El archivo **views.js** es a modo referencial, para no tener que instalar y crear contenido con una plantilla del tipo EJS.

Configuración de cookies

La función **cookie()** , integrada en el objeto **res** (response), se puede utilizar para configurar cookies en la respuesta que se enviará al cliente, cuando este acceda a la ruta especificada **"/"**.

response.cookie() es un método proporcionado por el framework Express para facilitar la configuración de cookies.

En nuestro ejemplo de código, le enviamos la fecha y hora del instante en el cual es accedida esta URL por una aplicación cliente.

```
app.get("/", (req, res) => {  
  const datosEnCookie = new Date()  
  res.cookie('nodeCookie', datosEnCookie)  
  res.status(200).send(vistaPrincipal)  
})
```

Configuración de cookies

Este método, acepta vario parámetros:

- **nombreCookie:** es el nombre de la cookie que se establecerá.
- **valorCookie:** es el valor que se asignará a la cookie.
- **{ opciones }:** es un objeto opcional que puede contener varias opciones para configurar la cookie, como la fecha de expiración, la ruta, la seguridad, entre otras.

```
res.cookie('nombreCookie', 'valorCookie', { opciones });
```

Configuración de cookies	
Valor	Descripción
Nombre	Es el identificador único de la cookie. Se utiliza para acceder y manipular la cookie desde el lado del servidor.
Valor	Es la información o datos que se almacenan en la cookie. Puede ser cualquier tipo de dato, como una cadena de texto, un número o incluso una estructura de datos en formato JSON.
Domínio	Es el dominio del sitio web que creó la cookie. La cookie solo se enviará al servidor cuando se haga una solicitud al dominio específico.
Ruta	Es la ruta dentro del dominio en la que la cookie es válida. Por ejemplo, si la ruta es "/admin", la cookie solo se enviará al servidor cuando la URL comience con "/admin".
Expiración	Es la fecha y hora en la que la cookie caduca y deja de ser válida. Después de la expiración, el navegador ya no enviará la cookie al servidor.
Segura	Es un valor booleano que indica si la cookie debe enviarse solo a través de conexiones HTTPS seguras. Esto proporciona una capa adicional de seguridad para proteger la información sensible.
HttpOnly	Valor booleano que indica si la cookie solo debe ser accesible a través del protocolo HTTP y no a través de scripts en el lado del cliente, como JavaScript. Esto ayuda a mitigar ataques de secuestro de cookies.

[Todo lo que vimos en esta otra diapositiva.](#)

Lectura de cookies

Cuando una aplicación cliente (*navegador web*, *Cliente REST*), accede a una ruta específica **/leer-cookie**, ante la petición del navegador web a esta ruta, el mismo comparte de forma directa cualquier cookie que haya sido generada por esta aplicación de backend previamente.

Dicha cookie puede ser leída nuevamente por el backend, a través del objeto **req** (request), la propiedad **cookie**, y una subpropiedad generada automáticamente con la clave generada oportunamente a dicha cookie al momento de crearla...

'nodeCookie', en nuestro caso.

A screenshot of a code editor window titled "Cookies". The code is written in JavaScript for an Express.js application. It shows a GET route for "/leer-cookie". Inside the route handler, a constant "fechaAcceso" is assigned the value of "req.cookies.nodeCookie" with a fallback to "'No hay registro previo'". Then, "res.status(200).send()" is called with a string that concatenates "Fecha del último acceso: " and the value of "fechaAcceso".

```
app.get('/leer-cookie', (req, res) => {  
  const fechaAcceso = req.cookies.nodeCookie || 'No hay registro previo'  
  res.status(200).send('Fecha del último acceso: ' + fechaAcceso)  
})
```

Eliminar cookies

El mismo objeto **res** (response), cuenta con una opción que nos permite eliminar una cookie, a través de la clave que definimos previamente en ella, en aquellos escenarios donde no necesitemos más a la misma.

```
app.get('/eliminar-cookie', (req, res)=> {
  const fechaAcceso = req.cookies.nodeCookie || 'No hay registro previo'
  if (fechaAcceso !== 'No hay registro previo') {
    res.clearCookie('nodeCookie')
    res.status(200).send('Se eliminó la cookie con el registro: ' + fechaAcceso)
  } else {
    res.status(406).send('No se encontró una cookie para eliminar.')
  }
})
```

En nuestro ejemplo, cuando el cliente acceda a la ruta de este endpoint, la cookie en cuestión será eliminada.

Aplicaciones

El uso principal de las cookies en los navegadores web, es poder brindar una experiencia personalizada a los usuarios. En éstas, almacenaremos y recibiremos datos pudiendo así recordar información del usuario y/o preferencias.

Entre los datos más comunes, están:

- autenticación
- preferencias de idioma
- carritos de compras
- selección de temas (dark - light)
- entre otras tantas opciones más...

Consideraciones de seguridad y privacidad

Existe una posibilidad de que los navegadores web realicen cambios en el manejo de las cookies en el futuro cercano. La privacidad en línea se ha convertido en un tema importante y los navegadores están tomando medidas para mejorar el control y la protección de la información personal de los usuarios.

Si bien no podemos prever con certeza cómo cambiará el manejo de las cookies en el futuro cercano, los navegadores siguen avanzando hacia un enfoque más centrado en la privacidad y brindan a sus usuarios un mayor control sobre sus datos personales.

Espacio de trabajo

Espacio de trabajo

Deberás implementar la creación y lectura de cookies dentro de una aplicación **Node.js + Express**, que permita acceder a determinada ruta para ver contenido oculto. Para ello, debes validar a un usuario a través del endpoint **/login**, utilizando la extensión **THUNDER-CLIENT** de Visual Studio Code.

El código funcional de los endpoints **"/**, **/login** y **/contenido-oculto**, ya está implementado. Solo debes agregarle la lógica faltante, implementando la creación de una cookie a partir de **POST:/login**, de acuerdo a las indicaciones dejadas mediante comentarios en el código.

Luego, accede mediante otro panel **New Request** de Thunder-Client al endpoint **/contenido-oculto**. Si la cookie fue registrada correctamente, se mostrará en este endpoint contenido HTML.

Tiempo estimado: 30 minutos. 

Espacio de trabajo

Puesta en común del ejercicio.

```
const questions = [ 'dudas', 'consultas', '🤔' ]
```



```
> node gracias.js
```