



UNIREMINGTON®
CORPORACIÓN UNIVERSITARIA REMINGTON
RES. 2661 MEN JUNIO 21 DE 1996

LENGUAJES DE PROGRAMACIÓN II

Desarrollo de Software

Módulo de Aprendizaje

2
0
2
2



Esta obra es publicada bajo la licencia
Creative Commons.Reconocimiento-
No Comercial-Compartir Igual 2.5 Colombia.

© Corporación Universitaria Remington

Medellín, Colombia
Derechos Reservados ©2022
Primera edición
2022

Lenguajes de Programación II
Gloria Amparo Lora Patiño
Facultad de Ingenierías

Comité académico

Decano o director de Unidad o Programa
Jorge Mauricio Sepúlveda
jsepulveda@uniremington.edu.co

David Ernesto González Parra
Director de Educación a Distancia y Virtual
dgonzalez@uniremington.edu.co

Francisco Javier Álvarez Gómez
Coordinador de Virtualidad EAD-V
falvarez@uniremington.edu.co

Edición y Montaje
Dirección de Educación a Distancia y Virtual
Equipo de diseño gráfico

www.uniremington.edu.co
virtual@uniremington.edu.co



Derechos reservados: El módulo de estudio del curso de **Lenguajes de Programación II** es propiedad de la Corporación Universitaria Remington; las imágenes fueron tomadas de diferentes fuentes que se relacionan en los derechos de autor y las citas en la bibliografía. El contenido del módulo está protegido por las leyes de derechos de autor que rigen al país. Este material tiene fines educativos y no puede usarse con propósitos económicos o comerciales. El autor(es) certificó (de manera verbal o escrita) No haber incurrido en fraude científico, plagio o vicios de autoría; en caso contrario eximió de toda responsabilidad a la Corporación Universitaria Remington y se declaró como el único responsable.



Esta obra es publicada bajo la licencia Creative Commons.
Reconocimiento-No Comercial-Compartir Igual 2.5 Colombia

TABLA DE CONTENIDO

Pág.

1	PROPÓSITO GENERAL	5
1.1	PROBLEMA PARA SOLUCIONAR O PREGUNTA PROBLEMATIZADORA	5
1.2	OBJETIVOS	5
1.2.1	General	5
1.2.2	Específicos	5
2	UNIDAD 1 CONCEPTOS DE DESARROLLO WEB.....	6
2.1	OBJETIVOS	6
2.1.1	General	6
2.1.2	Específicos	6
2.2	RELACIÓN DE CONCEPTOS.....	6
2.2.1	Mapa Mental o Conceptual.....	6
2.2.2	Lista de Conceptos	6
2.3	PRESENTACIÓN DE LA UNIDAD	7
2.4	DESARROLLO TEMÁTICO	7
2.4.1	Tema 1 Conceptos básicos de la programación web>	7
2.4.2	Tema 2 Servicios web Xml y Json	19
2.4.3	Tema 3 Arquitecturas web	21
2.4.4	Arquitectura de microservicios	26
2.5	RECURSOS.....	28
2.5.1	Documentación Referenciada	28
2.5.2	Material de profundización	28
2.6	EJERCICIOS DE PROFUNDIZACIÓN	29
2.7	PISTAS DE APRENDIZAJE	30
3	UNIDAD 2 INTRODUCCIÓN A LAS INTERFACES DE USUARIO WEB.....	31
3.1	OBJETIVOS	31
3.1.1	General	31
3.1.2	Específicos	31
3.2	RELACIÓN DE CONCEPTOS.....	31
3.2.1	Mapa Mental o Conceptual.....	31
3.2.2	Lista de Conceptos	31

3.3	PRESENTACIÓN DE LA UNIDAD	32
3.4	DESARROLLO TEMÁTICO	32
3.4.1	Tema 1 Elementos básicos que componen una interfaz de usuario	32
3.4.2	Tema 2 Lenguajes de maquetado HTML, CSS, JavaScript	36
3.4.3	Tema 3 Lenguajes de maquetado HTML, CSS, JavaScript	61
3.5	RECURSOS	72
3.5.1	Documentación Referenciada	72
3.5.2	Material de profundización	73
3.6	EJERCICIOS DE PROFUNDIZACIÓN	73
3.7	PISTAS DE APRENDIZAJE	74
4	UNIDAD 3 FRAMEWORK PARA EL DESARROLLO WEB	75
4.1	OBJETIVOS	75
4.1.1	General	75
4.1.2	Específicos	75
4.2	RELACIÓN DE CONCEPTOS	75
4.2.1	Mapa Mental o Conceptual	75
4.2.2	Lista de Conceptos	75
4.3	PRESENTACIÓN DE LA UNIDAD	76
4.4	DESARROLLO TEMÁTICO	76
4.4.1	Tema 1 Framework JavaScript para FrontEnd	76
4.4.2	Tema 2 Consumo API usando un framework	85
4.5	RECURSOS	88
4.5.1	Documentación Referenciada	88
4.5.2	Material de profundización	89
4.6	EJERCICIOS DE PROFUNDIZACIÓN	89
4.7	PISTAS DE APRENDIZAJE	90
5	VIDEO LECCIÓN	91
6	BIBLIOGRAFÍA	92
6.1	BASES DE DATOS INSTITUCIONALES	92
6.2	CIBERGRAFÍA	93

1 PROPÓSITO GENERAL

1.1 PROBLEMA PARA SOLUCIONAR O PREGUNTA PROBLEMATIZADORA

Actualmente la tecnología enfocada al desarrollo Web, ha aumentado la demanda de programadores en este campo, generando así oportunidades de empleo para millones de personas alrededor del mundo. Además, esto ha dado pie al nacimiento y el crecimiento de muchas empresas de tipo tecnológico, que prestan servicios a otras empresas.

1.2 OBJETIVOS

1.2.1 General

Conocer los principios básicos de una aplicación web, para la construcción soluciones de software orientadas al internet mediante el uso de herramientas propias del entorno seleccionado.

1.2.2 ESPECÍFICOS

- Identificar las tecnologías y conceptos que se deben de tener presentes al desarrollar un sitio web.
- Conocer las características de los lenguajes de programación para la creación de sitios web.
- Implementar el uso de framewok para el desarrollo de sitios web.

2 UNIDAD 1 CONCEPTOS DE DESARROLLO WEB.

2.1 OBJETIVOS

2.1.1 General

Identificar las tecnologías y conceptos que se deben de tener presentes al desarrollar un sitio web.

2.1.2 Específicos

- Identificar los conceptos básicos de la programación web.
- Conocer los servicios web JSON, SOAP o XML-RPC.
- Identificar las principales arquitecturas Web más usadas del desarrollo Web.

2.2 RELACIÓN DE CONCEPTOS

2.2.1 Mapa Mental o Conceptual



Ilustración 1 Mapa Conceptual Relación Conceptos Unidad 1

2.2.2 Lista de Conceptos

Backend: Parte lógica y de desarrollo que una página web usa para que esta funcione. Son las acciones, dentro del código, que NO son visibles para el usuario. La función principal es acceder a la información que el usuario solicita.

Frontend: Parte de una aplicación que interactúa con los usuarios, es conocida como el lado del cliente.

Http: De sus siglas en inglés "Hypertext Transfer Protocol", es el nombre de un protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos HTML.

XML: O "eXtensible Markup Language" es un lenguaje basado en etiquetas, el cual se utiliza para llevar datos de forma organizada. Se conoce como extensible porque el conjunto de etiquetas no es finito, ya que se pueden crear según la necesidad.

Rest-API: Una API REST (también conocida como API RESTful) es una interfaz de programación de aplicaciones (API o API web) que se ajusta a las restricciones del estilo arquitectónico REST y permite la interacción con los servicios web RESTful.

2.3 PRESENTACIÓN DE LA UNIDAD

Actualmente el desarrollo y creación de sitios web se impone como herramienta tecnológica para unir regiones, crear negocios, mostrarse ante el mundo y así infinidad de aplicaciones de acuerdo con las perspectivas y alcances de los usuarios. La programación web es la herramienta más importante del Internet ya que permite generar un diálogo constante, dinámico y amigable entre los usuarios y la información que se encuentra en los sitios web, es por esto por lo que es un pilar fundamental en la formación ingenieril.

2.4 DESARROLLO TEMÁTICO

Los lenguajes de programación del lado del cliente se utilizan para integrarse en los sitios web, existen multitud de plataformas que ayudan a crear sitios atractivos para los usuarios finales. Dichas plataformas se basan en una serie de lenguajes que se interpretan de una forma universal para que, desde cualquier parte del mundo, se pueda cargar una web y se visualice su contenido de forma idéntica. Estos lenguajes pueden ser interpretados por parte del servidor o por parte del cliente, según la necesidad.

2.4.1 TEMA 1 CONCEPTOS BÁSICOS DE LA PROGRAMACIÓN WEB>

2.4.1.1 COMO FUNCIONA UNA PÁGINA WEB

Cuando se accede a un sitio web desde un navegador específico desde el dispositivo suceden varios interrogantes:

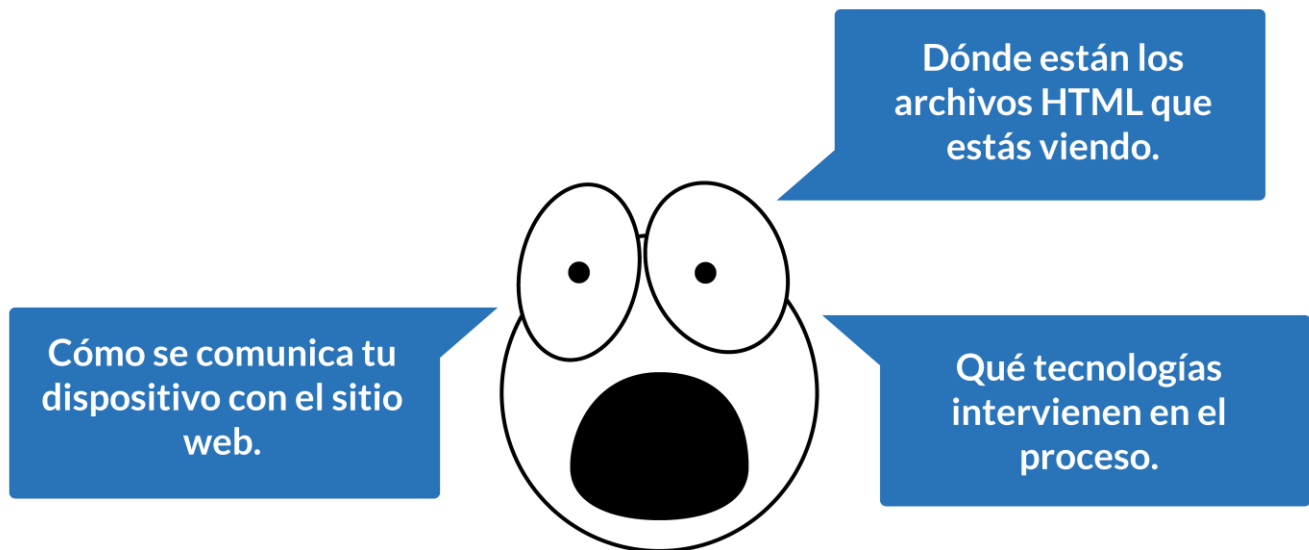


Ilustración 2 Porque aprender lenguajes Web

Para que un sitio web sea un sistema engranado para que funcione correctamente, tiene las siguientes partes:

- **Desarrollo web**

El desarrollo web se encarga de dar funcionalidad y se puede dividir en dos grandes bloques Front-end (se ejecuta en el cliente y es lo que ve el usuario) y Back-end (se ejecuta en el servidor).

Front End es la parte de una aplicación que interactúa con los usuarios, **es conocida como el lado del cliente**, en esta parte se ve más la fase artística y creativa de una página web, es toda la parte visual de la aplicación.

Back end es toda la parte lógica y de desarrollo que una página web usa para que funcione a la perfección, se debe aclarar que el Backend son las acciones, dentro del código, que NO son visibles para el usuario. La función principal es acceder a la información que el usuario solicita. Por ejemplo, que pueda ingresar con su cuenta registrada

Ejemplo:

- Frontend: Hacer que en la zona de la página aparezca una caja visible para que los usuarios puedan colocar acá su correo electrónico.

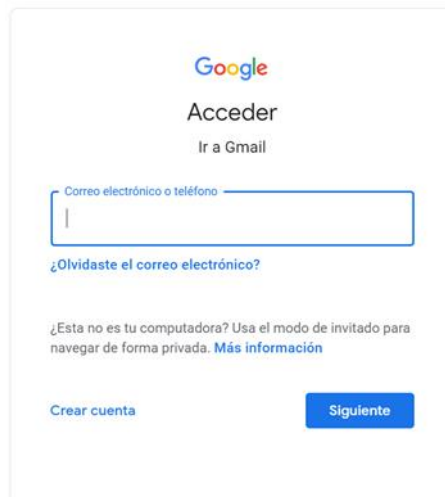


Ilustración 2 Ejemplo de Frontend

- Backend: lograr que, al momento de ingresar su correo, el usuario ingrese correctamente y puedan ingresar cuantas veces quieran a su cuenta. Un ejemplo de esto puede ser:

```
websocket.WebSocketWSGI
def startTimer(ws):
    n_cnt = 0
    global execTimer
    while execTimer:
        print('Timer fired! {}'.format(n_cnt))

        greenthread.sleep(1)
        n_cnt+=1

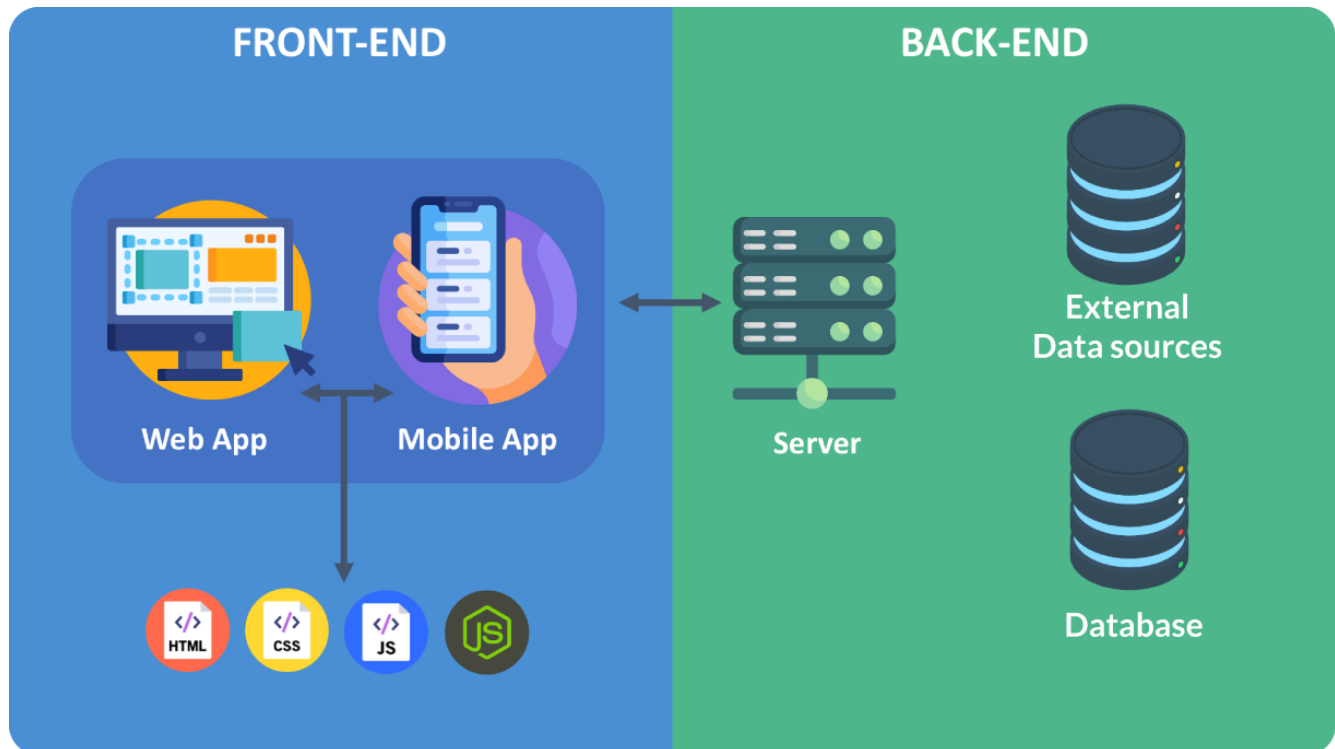
    try:
        ws.send('Timer fired! {}'.format(n_cnt))
    except Exception as e:
        print('Client websocket not available')
        ws.close()
        return

websocket.WebSocketWSGI
def processMessage(ws):
    m = ws.wait()
    print('Message received: {}'.format(m))

websocket.WebSocketWSGI
def saveData(ws):
AL mysite.py startTimer python utf-8[unix] 21% 25/118 ln : 1
```

Esta foto de Autor desconocido está bajo licencia CC BY-SA

Ilustración 3 Ejemplo Backend



Esta foto de Autor desconocido está bajo licencia CC BY-SA

Ilustración 4 Ejemplos de Front end y Back end

Navegador Web

- El navegador es el que te muestra el Front-end. Permite la interacción con la aplicación web, recoge datos y acciones para ser transmitidas al Back-end.
- Un ejemplo es un formulario donde rellenamos los campos y estos son enviados cuando realizamos la acción de pulsar el botón enviar.
- El navegador es capaz interpretar el código HTML, CSS y JavaScript para mostrarnos una versión gráfica del código.

Si miramos una página web de un sitio y, en cualquier navegador, pulsamos botón derecho -> ver código fuente, podremos ver qué es un HTML e incluso el CSS y JavaScript.

```
<!DOCTYPE html><html lang="es"><head><meta charset="UTF-8"><link data-optimized="2" rel="stylesheet" href="https://joseramonbernabeu.com/wp-content/litespeed/css/d0674757
<script id="joser-ready">window.advanced_ads_ready=function(e,a){a=a||"complete";var d=function(e){return"interactive"===e?"loading"!==e:"complete"===e};d(document.readyState)
var Cli_Data = {"nm_cookie_ids":["non_necessary_cookies":["ccpaEnabled":"","ccpaRegionBased":"","ccpaBarEnabled":"","strictlyEnabled":["necessary","obl
var cli_cookiebar_settings = {"animate_speed_hide":"500","animate_speed_show":"500","background":"#a32857","border":"#b1a6a6c2","border_on":"","button_1_button_colour":"#e
var log_object = {"ajax_url":"https://joseramonbernabeu.com/wp-admin/admin-ajax.php"};
/* */</script><script type="text/javascript" src="https://joseramonbernabeu.com/wp-content/plugins/cookie-law-info/public/js/cookie-law-info-public.js" defer ' id="cooki
<script type="text/plain" data-cli-class="cli-blocker-script" data-cli-script-type="non-necessary" data-cli-blocks="true" data-cli-element-position="head">(function(w,d,
new Date().getTime(),event:'gtm.js'));var f=d.getElementsByName(s)[0],
j=d.createElement(s),d1=l?'dataLayer':'&l=+1:':j.async=true;j.src=
'https://www.googletagmanager.com/gtm.js?id='+i+d1;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer','GTM-N8SD4C3');</script><script type="text/plain" data-cli-class="cli-blocker-script" data-cli-script-type="non-necessary" data-
<span id="post-modified-date-29-5172" class="oxy-post-modified-date">Actualizado el 12 de julio de 2020</span></div></div></div></section><section id="section-19-5096" c
if(responsiveVoice.isPlaying()){
    responsiveVoice.cancel();
}else{
    responsiveVoice.speak("Aunque es común usar los términos Web, Red e Internet de forma indistinta, desde el punto de vista técnico son cosas completamen
    });</script></div></div></section><section id="section-10-5096" class="ct-section"><div class="ct-section-inner-wrap"><h2 id="headline-22-5172" class="ct-head
<b class="lwptoc_title">Tabla de Contenidos</b><span class="lwptoc_toggle">
<a href="#" class="lwptoc_toggle_label" data-label="Mostrar/Ocultar">
</span></div><div class="lwptoc_items lwptoc_items-visible"><ul class="lwptoc_itemWrap"><li class="lwptoc_item"><a href="#">como funciona una pagina web</a>
</li><li class="lwptoc_item"><a href="#">arquitectura</a>
</li><li class="lwptoc_item"><a href="#">cliente y servidor</a>
</li><li class="lwptoc_item"><a href="#">desarrollo web</a>
</li><li class="lwptoc_item"><a href="#">desarrollo web</a>
</li><li class="lwptoc_item"><a href="#">navegador web</a>
</li><li class="lwptoc_item"><a href="#">navegador web</a>
</li><li class="lwptoc_item"><a href="#">protocolo http</a>
</li><li class="lwptoc_item"><a href="#">protocolo http</a>
</li><li class="lwptoc_item"><a href="#">dominio de internet</a>
</li><li class="lwptoc_item"><a href="#">dominio de internet</a>
</li><li class="lwptoc_item"><a href="#">dns</a>
```

Ilustración 5 Código fuente de una página de Internet

HTTP

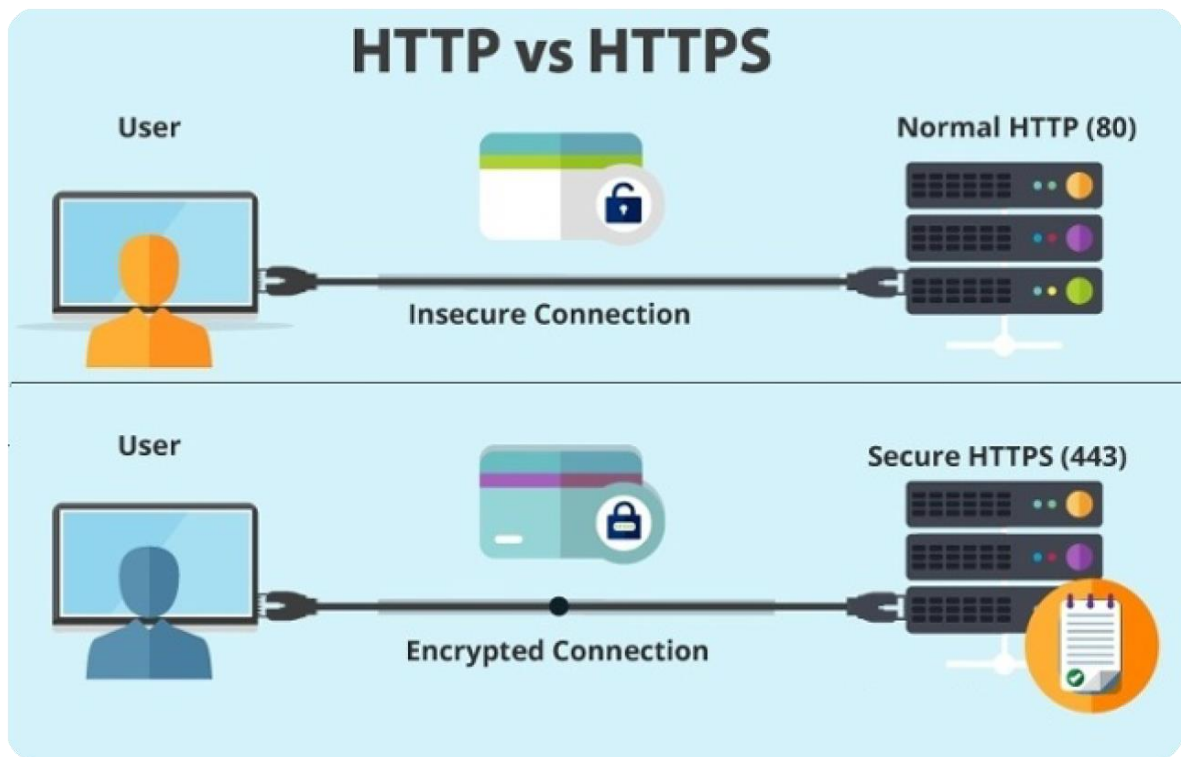
HTTP (HyperText Transfer Protocol) es un protocolo de comunicación, son normas y reglas que permiten la transmisión de archivos HTML. Gracias a él se pueden comunicar el navegador que tienes en tu dispositivo con un servidor que no sabes ni donde está.

- Basado en texto (text-based): a la final lo que se envía es un texto que tiene una determinada sintaxis, en él se especifican los atributos del Request o del Response según el caso.
- Request-Response: utiliza el modelo cliente-servidor, y se tiene en cuenta que el cliente siempre comienza la comunicación.
- La versión segura del protocolo, **HTTPS**, cifra los datos para que, si alguien intercepta la información, no pueda entender qué se está transmitiendo.

Desde hace un tiempo, los navegadores, sobre todo Google Chrome, te avisan cuando vas a entrar a un sitio que potencialmente inseguro.

Así que es imprescindible disponer de un certificado de seguridad e implementar el protocolo HTTPS para cualquier web, sobre todo si se trata de un comercio electrónico o un sitio que transmita información sensible

Por lo tanto, este protocolo es una parte fundamental del sistema que engloba a un sitio web.



Esta foto de Autor desconocido está bajo licencia CC BY-SA

Ilustración 7 Diferencias entre HTTP y HTTPS

- Sin estado (stateless): luego de recibir el Response la comunicación se acaba.
- Basado en texto (text-based): a la final lo que se envía es un texto que tiene una determinada sintaxis, en él se especifican los atributos del Request o del Response según el caso.
- Request-Response: utiliza el modelo cliente-servidor, y se tiene en cuenta que el cliente siempre comienza la comunicación.

2.4.1.2 REQUEST HTTP

El Request es el mensaje que debe enviar el cliente para comenzar la comunicación con el servidor. Este mensaje debe tener el siguiente orden:

1. [Método HTTP] [URI] [Versión HTTP]
2. Cero o más encabezados (headers).
3. Una línea vacía.

4. La información del cuerpo (opcional).

A continuación, se muestra un ejemplo de un Request HTTP:

```
GET /Protocols/rfc2616/rfc2616.html HTTP/1.1
Host: www.w3.org
User-Agent: Mozilla/5.0
```

(línea vacía)

ENCABEZADOS DE UN REQUEST HTTP

Los encabezados HTTP se utilizan para llevar información adicional en el mensaje. A continuación, se muestran los encabezados de un **Request** más comunes con su respectiva descripción:

Tabla 1: Encabezados Request HTTP

HEADER	DESCRIPCIÓN
Accept	Se especifica el tipo de contenido que es aceptado por el cliente como cuerpo del Response.
Accept-Charset	Le dice al servidor qué tipo de charset acepta el cliente en el cuerpo del Response.
Authorization	Es utilizado para enviar credenciales de autenticación al servidor.
Cookie	El cliente retorna las cookies que fueron puestas por el servidor que está siendo llamado.
Content-Length	Lleva el tamaño del cuerpo del Request en bytes.
Content-Type	El tipo de contenido del cuerpo del Request (en el caso de que haya cuerpo).
Host	El nombre del servidor acompañado del número del puerto. Si el puerto es omitido, por defecto será 80.
Referrer	La dirección de la página que está pidiendo la siguiente página al servidor.
User-Agent	Describe el cliente que está enviando el Request.

2.4.1.3 MÉTODOS HTTP

En el ejemplo del Request HTTP se puede ver que el texto comienza con un GET, el cual corresponde al método HTTP. Este método puede ser de varios tipos tal y como se puede ver en la siguiente tabla:

Tabla 2: Métodos HTTP más comunes

MÉTODO	ACCIÓN
GET	Le dice al servidor que retorne el recurso especificado. (Es el que usan los navegadores web)
HEAD	Lo mismo que el GET, pero el Response debe venir sin cuerpo, lo que quiere decir que únicamente vienen los encabezados.
POST	Permite enviar cuerpo en el Request y normalmente se utiliza para crear recursos.
PUT	Permite enviar cuerpo a través del Request y normalmente se utiliza para actualizar todo un recurso en el caso de que exista o crearlo si no.
DELETE	Le dice al servidor que elimine un recurso.
TRACE	Le dice al servidor que retorne el Request como Response.
OPTIONS	Le dice al servidor que retorne una lista de los métodos HTTP que el servidor soporta.
CONNECT	Le dice al servidor como configurar la comunicación con el cliente. Normalmente, este método es usado para configurar SSL para la comunicación a través de HTTPS.
PATCH	Le dice al servidor que la información que va en el cuerpo es para modificar parte de la información de un recurso.

2.4.1.4 RESPONSE HTTP

El Response es el mensaje que da el servidor como respuesta a un Request. Este mensaje debe tener el siguiente orden:

1. Estado HTTP
2. Cero o más encabezados (headers).
3. Una línea en vacía.
4. El cuerpo del mensaje (opcional).

A continuación, se muestra un ejemplo de un Response HTTP:

```
HTTP/1.1 200 OK
Date: Sat, 22 Nov 2014 12:58:58 GMT
Server: Apache/2
Last-Modified: Thu, 28 Aug 2014 21:01:33 GMT
Content-Length: 33115
Content-Type: text/html; charset=iso-8859-1
(línea vacía)
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> <html xmlns='http://www.w3.org/1999/xhtml'> <head><title>Hypertext Transfer Protocol -- HTTP/1.1</title></head><body>...</body></html>
```

2.4.1.5 ENCABEZADOS DE RESPONSE HTTP

Nótese que en ejemplo anterior el Response tiene un cuerpo, el cual corresponde a un archivo HTML. Al igual que en el Request, este cuerpo es opcional. Además, este Response también tiene una variedad de encabezados, los cuales se explican en la siguiente tabla:

Tabla 3: Encabezados más comunes para el Response

HEADER	DESCRIPCIÓN
Allow	Le dice al cliente cuales métodos HTTP son soportados por el servidor.
Content-Length	Lleva el tamaño del cuerpo del Response en bytes.
Content-Type	El tipo de contenido del cuerpo del Response (cuando hay cuerpo).
Date	Dice la fecha y la hora actual. (formateado en GMT)
Location	Le dice al cliente donde pedir la siguiente URL.
Server	Dominio del servidor que está retornando el Response.
Set-Cookie	Envía cookies al cliente. Múltiples Set-Cookie pueden ser agregados al mismo Response.
WWW-Authenticate	Dice que tipo de autorización debe suministrar el cliente en el header Authorization del Request.

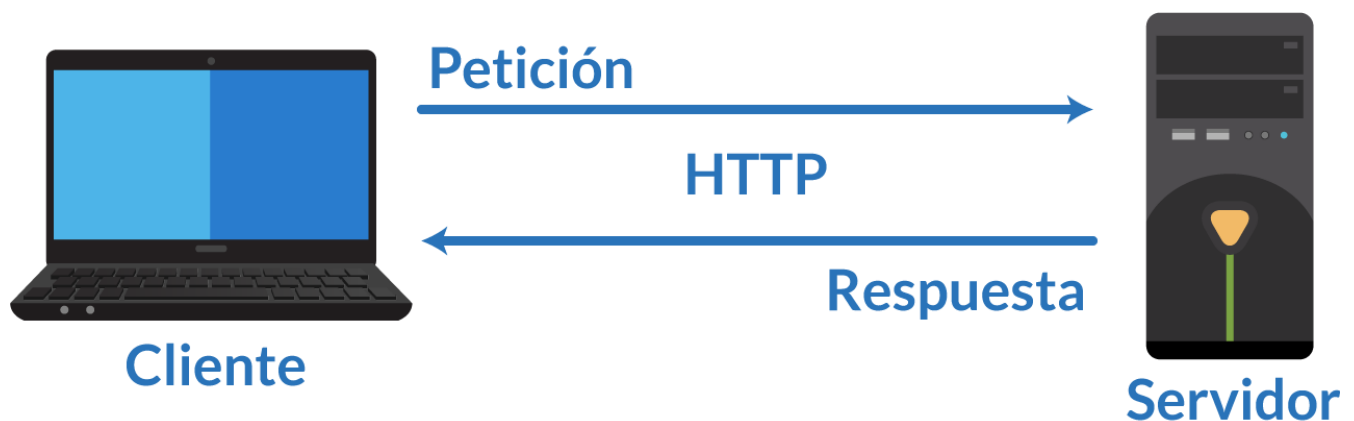
2.4.1.6 ESTADOS HTTP

En el ejemplo del Response se puede notar el texto **200 OK**, el cual corresponde al estado HTTP; gracias a esto se puede saber si la comunicación con el cliente fue exitosa o no. Existen varios tipos de estado, los cuales se describen en la siguiente tabla:

Tabla 4: Descripción de los códigos de estado HTTP

CÓDIGO DE ESTADO	DESCRIPCIÓN
1XX	Son de información. Le dicen al cliente que el servidor ha recibido el Request y está procesándolo.
2XX	Indica que todo ha salido bien, es decir, el servidor recibió el Request y lo ha procesado satisfactoriamente.
3XX	Indica redirección, es decir, le dice al cliente que el Request fue recibido pero que para ser procesado el cliente debe hacer algo más. En la mayoría de los casos se utiliza para indicar que ya no se está trabajando con esa URL sino con otra.
4XX	Indica que el cliente cometió un error, es decir, indica que hay algo malo en el Request. Por ejemplo, el código de estado más conocido es 404 Not Found, donde el servidor le dice al cliente que el recurso que está intentando obtener no se encuentra en la URL.
5XX	Indica que el servidor cometió un error. El código de estado más conocido es el 500 Internal Server Error, el cual indica que hay algún fallo en el servidor.

Ilustración 8 Petición Cliente Servidor



Dominio de Internet

Un dominio de Internet es una red de identificación asociada a un grupo de dispositivos o equipos conectados a la red Internet, su propósito principal de los nombres de dominio en Internet y del sistema de nombres de dominio (DNS), es traducir las direcciones IP de cada nodo activo en la red, a términos fáciles de recordar.

El nombre de dominio nos permite identificar un sitio web de una manera fácil para los humanos, los usuarios de Internet tendrían que acceder a cada servicio web utilizando la dirección IP del nodo (por ejemplo, sería necesario utilizar `http://192.0.32.10` en vez de `http://ejemplo.com`).

Esto quiere decir que cuando buscando en nuestro navegador la URL **`https://www.uniremington.edu.co/soy-estudiante-uniremington/`**, en la mayoría de los casos el navegador debe consultar primero la IP correspondiente al dominio **`www.uniremington.edu.co`** para poder saber con qué servidor de la red se va a comunicar, luego de esto el navegador procede a pedir el recurso que requiere de dicho servidor, que en este caso es lo que se encuentra en la ruta **`/estudiante-uniremington`**.

DNS

El Sistema de nombres de dominio (Domain Name System) es el que se encarga de vincular a cada dominio con una dirección IP (Internet Protocol).

Las direcciones web reales (IP) no son las agradables y fácilmente recordables secuencias que tecleas en la barra de direcciones para encontrar tus sitios web favoritos. En realidad, se trata de secuencias de números del tipo 68.187.32.220.

El DNS no es más que una base de datos distribuida, es decir, hay multitud de bases de datos en todo el mundo que son replicas unas de otras.

DNS funciona de la siguiente manera:

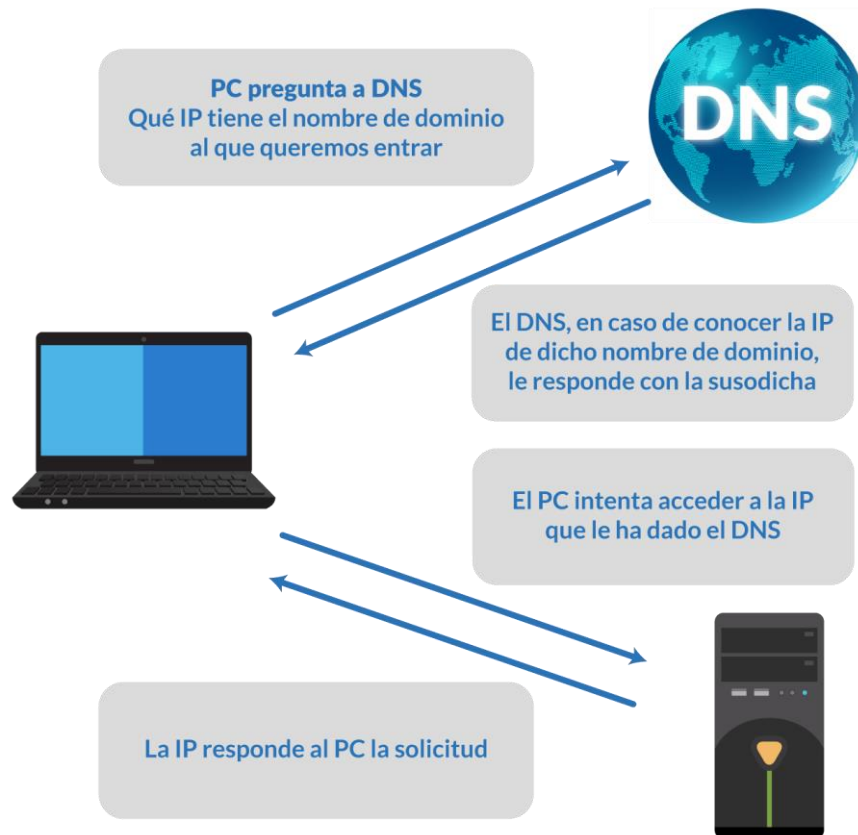


Ilustración 9 Funcionamiento DNS

Servidor web

La principal función de un servidor Web es almacenar los archivos del sitio (HTML, CSS, JavaScript, multimedia, entre otros.), emitirlos por Internet para que así los usuarios lo puedan visitar, encargado de interpretar los lenguajes de programación que se utilizan en el Back-end.

Es un software instalado en un ordenador que procesa una aplicación del lado del servidor, realizando conexiones con el cliente y generando una respuesta en cualquier lenguaje o aplicación del lado del cliente.

El ordenador tiene unas características especiales que lo hacen fiable, seguro, con gran capacidad y potente, cada servidor tiene asignada una dirección IP para que pueda ser fácilmente identificado por el DNS

Este servicio lo ofrecen las compañías de Hosting, que a cambio de una tarifa mensual o anual te ofrecen un servicio para alojar tus sitios web en sus servidores.

Más del 90% de los sitios en Internet utilizan un servidor alquilado a través de una empresa de Web Hosting, sin los servidores Web, la Internet tal como la conocemos no existiría. Los servidores son el depositario de todo el contenido que existe en internet.

Entre estos servidores web tenemos:

- Apache. Es el servidor web de referencia, el más popular y extendido. ...
- Nginx. ...
- Microsoft IIS. ...
- Google GWS. ...
- Lite Speed. ...
- Lighttpd. ...
- Sun Java System Web Server

En resumen, cómo funciona un sitio web



Ilustración 6 Funcionamiento Sitio Web

2.4.2 TEMA 2 SERVICIOS WEB XML Y JSON

Para el envío de información entre el cliente y servidor se debe utilizar algún formato que ambos conozcan. En gran parte de los casos los más usados son **JSON** y **XML**. Usado para el transporte de información.

XML

XML o eXtensible Markup Language es un lenguaje basado en etiquetas, el cual se utiliza para llevar datos de forma organizada. Se conoce como extensible porque el conjunto de etiquetas no es finito, ya que se pueden crear según la necesidad.

JSON

JSON o JavaScript Object Notation es un objeto de JavaScript que consiste en un conjunto de atributos clave-valor, donde la clave corresponde al nombre del atributo y el valor a la información que el atributo lleva. Además, este tipo de objetos permiten identificar los tipos de dato cadena, booleano, número y arreglo. JSON son las siglas de J a v a S c r i p t O b j e c t N o t a t i o n, JSON es un formato de texto para almacenar y transportar datos, JSON es "autodescriptivo" y fácil de entender.

Los siguientes ejemplos de JSON y XML definen un objeto de empleados, con una matriz de 4 empleados:

EJEMPLO JSON

```
{ "employees": [
  { "firstName": "John", "lastName": "Doe" },
  { "firstName": "Anna", "lastName": "Smith" },
  { "firstName": "Peter", "lastName": "Jones" } { "firstName": "Gloria", "lastName": "Lopez" }
]}
```

EJEMPLO XML

```
<employee>
  <firstName>John</firstName> <lastName>Doe</lastName>
</employee>
<employee>
  <firstName>Anna</firstName> <lastName>Smith</lastName>
</employee>
<employee>
  <firstName>Peter</firstName> <lastName>Jones</lastName>
</employee>

<firstName>Gloria</firstName> <lastName>Lopez</lastName>
</employee>
</employees>
```

JSON es como XML porque

1. Tanto JSON como XML son "autodescriptivos" (legibles por humanos)
2. Tanto JSON como XML son jerárquicos (valores dentro de valores)
3. Tanto JSON como XML pueden ser analizados y utilizados por muchos lenguajes de programación.
4. Tanto JSON como XML se pueden recuperar con XMLHttpRequest

JSON es diferente a XML porque

1. JSON no usa etiqueta de cierre
2. JSON es más corto
3. JSON es más rápido de leer y escribir
4. JSON puede usar matrices

La mayor diferencia es:

XML debe analizarse con un analizador XML. JSON se puede analizar mediante una función estándar de JavaScript.

Por qué JSON es mejor que XML

XML es mucho más difícil de analizar que JSON, JSON se analiza en un objeto JavaScript listo para usar.

2.4.3 TEMA 3 ARQUITECTURAS WEB

2.4.3.1 ARQUITECTURA CLIENTE SERVIDOR

Tabla 2 Arquitectura Cliente Servidor

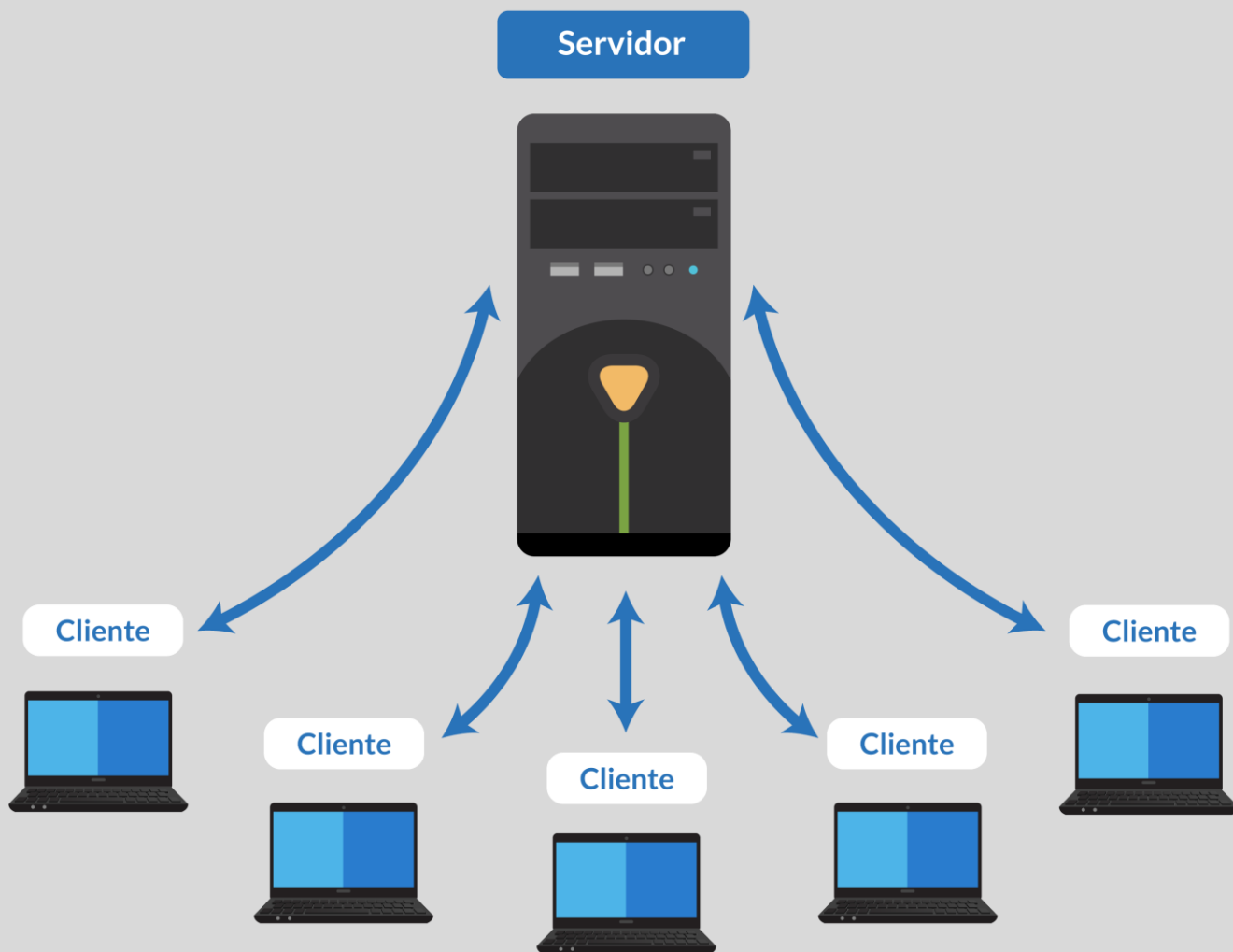
ARQUITECTURA CLIENTE SERVIDOR		
PARTES	Cliente	Estaciones de trabajo que solicitan servicios al servidor
	Servidor	Maquina potente con hardware y software específicos, actúa como depósito de datos
IMPORTANCIA	a. Mantener las comunicaciones entre diferentes entidades usando protocolos y almacenamiento establecidos. b. Conecta varios clientes a servicios que provee un servidor.	

	c. Un cliente también puede tener una función de servidor ya que el mismo puede almacenar datos en su disco duro para luego ser usados en vez de estar conectándose al servidor continuamente por una acción que quizás sea muy sencilla.
DIFERENCIA ENTRE CLIENTE Y SERVIDOR	<p>CLIENTE: Es un computador como el que tenemos en casa y oficina y el cual se conecta a un servidor o a los servicios de este a través de Internet o una red interna. Ejemplo: Es la forma en que trabaja una empresa con diferentes computadores donde cada uno de ellos se conectan a un servidor para poder obtener archivos de una base de datos o servicios ya sea correos electrónicos o aplicaciones</p> <p>SERVIDOR: El servidor al igual que el cliente, es un computador pero con diferencia de que tiene una gran capacidad que le permite almacenar gran cantidad de diversos de archivos, o correr varias aplicaciones en simultaneo para así nosotros los clientes poder acceder los servicios.</p>
TIPOS DE ARQUITECTURAS CLIENTE SERVIDOR	<p>Arquitectura de dos capas</p> <p>Se usa para describir los sistemas cliente servidor en donde el cliente solicita recursos y el servidor responde directamente a la solicitud con sus propios recursos. El servidor no requiere de una aplicación extra para proporcionar parte del servicio.</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> <p>Capa de Cliente</p> </div> <div style="text-align: center;"> <p>Capa de Servidor</p> </div> </div> <p>Arquitectura de 3 Capas o N capas</p> <p>En la arquitectura a tres capas existe un nivel intermedio, eso significa que la arquitectura generalmente está compartida por un cliente que solicita los recursos equipados con una interfaz de usuario o mediante un navegador web, la capa del medio es denominada software intermedio cuya función es proporcionar los recursos solicitados pero que requiere de otro servidor para hacerlo. La última capa es el servidor de datos que proporciona al servidor de aplicaciones los datos necesarios para poder procesar y generar el servicio que solicito el cliente en un principio.</p>

<p>VENTAJAS</p>	<ul style="list-style-type: none"> a. Facilita la integración entre diferentes sistemas y comparte información permitiendo de esta manera poder integrar varios computadores con sistemas medianos y grandes sin necesidad de que todos tengan que utilizar el mismo sistema operativo. b. Los sistemas construidos bajo este esquema tienen una mayor interacción con el usuario, puesto que favorece el uso de las interfaces gráficas interactivas, c. La estructura modular facilita de más la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional favoreciendo así la estabilidad de las soluciones, genera un orden de trabajo, en donde cada sector puede trabajar en su área, pero accediendo al mismo servidor e información que los demás sin generar conflictos
<p>DESVENTAJAS</p>	<ul style="list-style-type: none"> a. Requiere habilidad para que un servidor sea reparado. Es decir, si un problema ocurre en la red, se requiere de alguien con un amplio de esta para poder repararla en su totalidad para así dejar que la información y el correcto funcionamiento siga su flujo. b. Otro problema es la seguridad, el hecho que se comparte canales de información entre servidores y clientes requieren que estas pasen por procesos de validación, es decir protocolos de seguridad que pueden tener algún tipo de puerta abierta permitiendo que se generen daños físicos, amenazas o ataques de malware.
<p>COMPONENTES</p>	<ul style="list-style-type: none"> a. Red: Conjunto de clientes, servidores y base de datos unidos de una manera física o no, en el que existen protocolos de transmisión de información. b. Cliente: Hace referencia a un demandante de servicios, este cliente puede ser un ordenador como también una aplicación de informática, la cual requiere información proveniente de la red para funcionar. c. Servidor: Es un proveedor de servicios, puede ser un ordenador o una aplicación informática la cual envía información a los demás agentes de la red. d. Protocolo: Conjunto de normas o reglas concretas sobre el flujo de información en una red estructurada. e. Servicios: Conjunto de información que busca responder las necesidades de un cliente.

f. Base de datos: Bancos información ordenada, categorizada y clasificada que forman parte de la red.

Modelos Cliente-Servidor

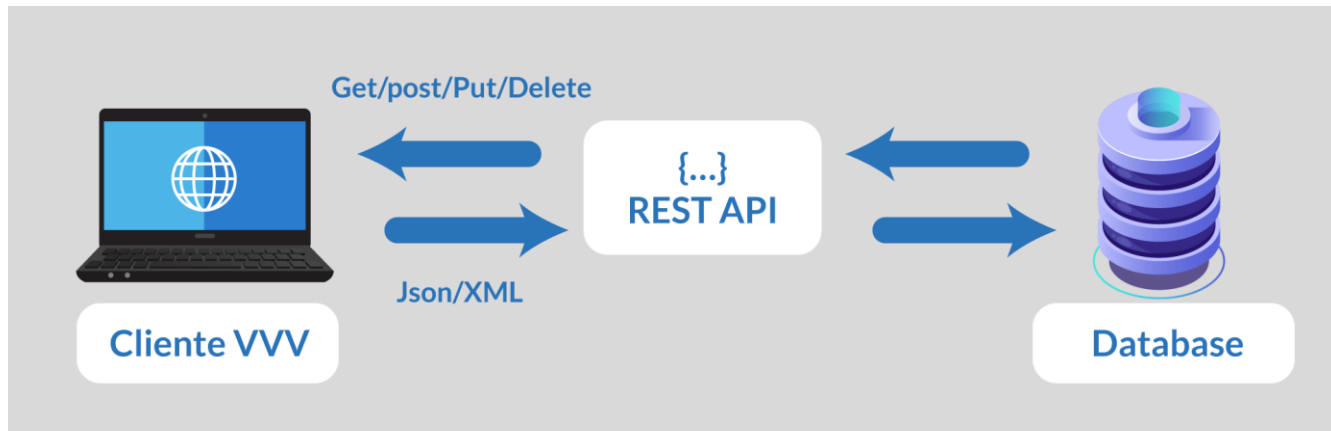


2.4.3.2 ARQUITECTURA REST API

Tabla 3 Arquitectura REST API

ARQUITECTURA REST API	
DEFINICIÓN	Arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP.
IMPORTANCIA	<p>La mayoría de las aplicaciones que se desarrollan para servicios profesionales disponen de una API REST para el intercambio de información entre el front end y el back end.</p> <p>Lo que la hace tan potente es precisamente el aislamiento que proporciona entre la lógica del back-end y cualquier cliente consumidor de éste.</p> <p>Esto le permite ser usada por cualquier tipo de cliente: web, móvil, entre otros. Así, cualquier dispositivo/cliente que entienda de HTTP puede hacer uso de su propia API REST de manera muy simple. Esto ha hecho que en los últimos años este tipo de arquitectura haya ganado peso frente a otras más complejas como SOAP, para el intercambio y manipulación de datos.</p>
MÉTODOS	<p>La arquitectura REST se basa en recursos, donde cada uno de ellos está identificado de forma única.</p> <p>Para poder interactuar con estos recursos utilizaremos, principalmente, los siguientes métodos HTML:</p> <ul style="list-style-type: none"> -GET: se utiliza para acceder a los distintos recursos. -POST: se utiliza para realizar acciones de creación de nuevos recursos. -PUT: se utiliza para modificar los recursos existentes. -DELETE: se utiliza para eliminar los recursos.
VENTAJAS	<p>Ventajas que ofrece REST para el desarrollo</p> <ol style="list-style-type: none"> Separación entre el cliente y el servidor: el protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Visibilidad, fiabilidad y escalabilidad. La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Esta separación facilita tener en servidores distintos el front end y el back end y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar. La API REST siempre es independiente del tipo de plataformas o lenguajes: la API REST siempre se adapta al tipo de sintaxis o

plataformas con las que se estén trabajando, lo que ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo. Con una API REST se pueden tener servidores PHP, Java, Python o Node.js.



2.4.4 ARQUITECTURA DE MICROSERVICIOS

Tabla 4 Arquitectura Microservicios

ARQUITECTURA MICROSERVICIOS	
DEFINICION	Los microservicios son tanto un estilo de arquitectura como un modo de programar software. Con los microservicios, las aplicaciones se dividen en sus elementos más pequeños e independientes entre sí.
IMPORTANCIA	Proporciona a los equipos de desarrollo un enfoque más descentralizado para construir software. Los microservicios permiten que cada servicio sea aislado, reconstruido, redesplegado y administrado de forma independiente.
VENTAJAS	<ul style="list-style-type: none"> Despliegue fácil. Requieren menos tiempo de desarrollo. Puede escalar rápidamente Se puede reutilizar en diferentes proyectos. Contienen mejor aislamiento de fallas. Puede ser desplegado en equipos relativamente pequeños. Trabaja bien con los contenedores. Aplicaciones Abiertas
DESVENTAJAS	<ul style="list-style-type: none"> Demasiada Granularidad. Esfuerzo extra de diseño para la comunicación entre servicios. Latencia durante el uso pesado. Pruebas complejas.

**ARQUITECTURA
MONOLÍTICA**



ARQUITECTURA DE MICROSERVICIOS



<https://i.imgur.com/YiVmDyL.png>

Fecha Recuperación: 15/Octubre/2021.

2.5 RECURSOS

2.5.1 Documentación Referenciada

Tema	Enlace URL	Especificación de páginas o instrucciones de lectura
Desarrollo Web HTML-CSS	https://elibro.net/es/ereader/remington/56045	9-11 19-57
HTTP	https://elibro.net/es/ereader/remington/50305	Página 80

2.5.2 Material de profundización

Tema	Enlace URL	Especificación de páginas o instrucciones de lectura
HTML JAVASCRIPT APLICACIÓN WEB	https://rua.ua.es/dspace/bitstream/10045/16995/1/sergio_lujan-programacion_de_aplicaciones_web.pdf	91 y siguiente 181 y 299 47

CLIENTE-SERVIDOR		39
XML	http://sedici.unlp.edu.ar/bitstream/handle/10915/4071/Documento_completo_.pdf?sequence=39&isAllowed=y	84-90
ARQUITECTURA MULTICAPA	https://www.researchgate.net/profile/Tiago-Fernandez-Carames/publication/235966532_Arquitectura_multicapa_distribuida_para_demostradores_MIMO/links/5881b2f34585150dde3f4a49/Arquitectura-multicapa-distribuida-para-demostradores-MIMO.pdf	1-4

2.6 EJERCICIOS DE PROFUNDIZACIÓN

- A. Determine los 20 conceptos más importantes para el desarrollo de sitios web, defínalos con sus palabras y construya un mapa mental.
- B. Desarrolle un mapa conceptual con los aspectos más importantes de cada una de las arquitecturas Web.
- C. Convierta los siguientes objetos a JSON y XML:
 1. Estudiante
 - a) Cedula= 43151922
 - b) Nombre = Gloria Pérez
 - c) Salario por hora = 50000
 - d) Nivel Estudios = [Pregrado, Maestría, Doctorado]
 2. Mascota
 - a) Código Chip = 234789

- b) Nombre = Copito
- c) Raza = [persa, Azul ruso, Siamés, Angora turco.]
- d) Nombre Propietario = Mary Perdomo

2.7 PISTAS DE APRENDIZAJE



Recuerde que:

- Desarrollo Web
- Navegador Web
- Servidor Web
- Dominio
- XML
- JSON
- Arquitecturas

3 UNIDAD 2 INTRODUCCIÓN A LAS INTERFACES DE USUARIO WEB

3.1 OBJETIVOS

3.1.1 General

Conocer las características de los lenguajes de programación para la creación de sitios web.

3.1.2 Específicos

- Conocer los elementos básicos que compone una interfaz de usuario.
- Identificar las principales características del lenguaje de maquetado HTML, CSS
- Reconocer los aspectos principales de JavaScript.

3.2 RELACIÓN DE CONCEPTOS

3.2.1 Mapa Mental o Conceptual

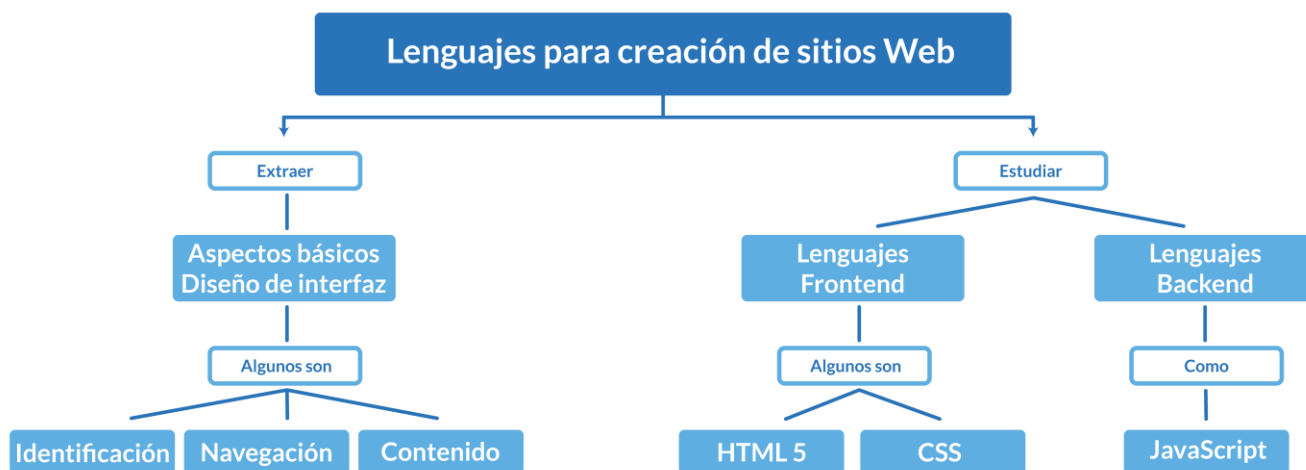


Ilustración 7 Relación conceptos Unidad 2

3.2.2 Lista de Conceptos

HTML5: Versión del estándar HTML que se utiliza para crear las páginas web e incorpora posibilidades como la reproducción de contenido multimedia.

CSS: Siglas en inglés para «hojas de estilo en cascada» (Cascading Style Sheets). Es un lenguaje que maneja el diseño y presentación de las páginas web. Funciona junto con el lenguaje HTML que se encarga del contenido básico de las páginas.

JavaScript: Lenguaje de secuencias de comandos o de programación que permite implementar funciones complejas en páginas web. Junto con HTML y CSS configuran las tecnologías estándar para desarrollar páginas web.

3.3 PRESENTACIÓN DE LA UNIDAD

Esta unidad va a permitir al estudiante conocer las principales herramientas que debe de identificar y conocer los conceptos para el desarrollo de una interfaz de usuario de acuerdo con los estándares, como también conocer las principales características de los lenguajes de maquetado HTML y CSS y como estos permiten ser la base para el diseño de interfaces, al igual la tendencia de lenguajes de programación orientadas a la WEB.

3.4 DESARROLLO TEMÁTICO

3.4.1 TEMA 1 ELEMENTOS BÁSICOS QUE COMPONEN UNA INTERFAZ DE USUARIO

a) Definición

Es la forma de exponer o de presentar la información hacia los usuarios a través la página web. Medio por el cual una persona controla una aplicación de software o dispositivo de hardware. Esto quiere decir que el programa incluye controles gráficos que optimizan la experiencia de usuario usando un mouse o teclado.

Un ejemplo muy común de esto son los sistemas operativos Macintosh y Windows tienen diferentes interfaces de usuario y comparten muchos elementos, como un escritorio, ventanas o iconos. Estos elementos comunes hacen posible que las personas utilicen cualquiera de estos sistemas operativos sin tener que aprender a manejar una nueva interfaz.

Al igual los programas como los procesadores de texto y los navegadores web tienen interfaces bastante similares, lo que proporciona una experiencia de usuario consistente entre varios programas.

Escritorio de Windows:



Ilustración 12 Interfaz Escritorio Windows

Escritorio MAC:



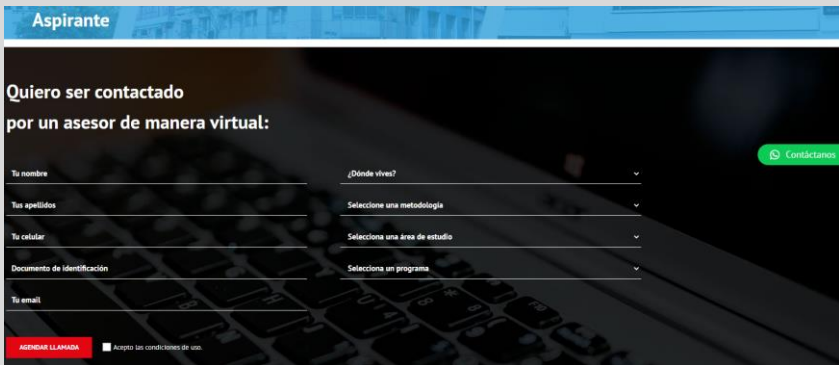
Ilustración 13 Interfaz Escritorio Mac

b) Elementos básicos de la interfaz.

Tabla 8 Elementos Básicos de la interfaz

ARQUITECTURA MICROSERVICIOS		
Identificación	Definición	Identifica plenamente el sitio web. Al usuario, ver estos elementos, debe saber a quién pertenece la web.
	Compuesto por	1.El nombre de la página web 2.El logo 3.La imagen de cabecera
	Ejemplo	
Navegación	Definición	Permiten al usuario moverse por las diferentes secciones del sitio y volver de nuevo a la página inicial, lo que se llama la Home. Esta presentes en cada una de las páginas de la web y ser lo suficientemente intuitivos para que el usuario sepa adónde lleva cada elemento
	Compuesto por	1.Menú principal. 2.Información sobre la ubicación del usuario dentro del sitio web. 3.Elemento de regreso al home 4. Widgets.
	Ejemplo	

Contenido	Definición	Son las zonas en las que se muestra la información relevante de la página web que componen el sitio.
	Compuesto por	<ol style="list-style-type: none"> 1. Áreas de texto o imágenes. 2. Vídeos 3. Archivos adjuntos.
	Ejemplo	

Interacción	Definición	Son los elementos de las diferentes zonas del Sitio Web en las que el usuario puede interactuar con la página.
	Compuesto por	<ol style="list-style-type: none"> 1. búsqueda en el sitio 2. Consultar 3. Carrito de compras <p>En la siguiente imagen esta la interacción de los futuros estudiantes Uniremington ya sea vía formulario o vía WhatsApp</p>
	Ejemplo	

3.4.2 TEMA 2 LENGUAJES DE MAQUETADO HTML, CSS, JAVASCRIPT

3.4.2.1 HTML

3.4.2.1.1 Introducción

Divida el texto en temas, cada uno con un título apropiado y con estilo Título 3. Importante: Imágenes de buena calidad y su respectiva referencia bibliográfica. HTML es el lenguaje de marcado estándar para crear páginas web, Son las siglas de Hyper Text Markup Language, describe la estructura de una página web, consta de una serie de elementos, los elementos HTML le dicen al navegador cómo mostrar el contenido, los elementos HTML etiquetan partes de contenido como "esto es un encabezado", "esto es un párrafo", "esto es un enlace", etc.

En este curso se trabajarán todas las etiquetas del estándar HTML5.

3.4.2.1.2 Editores

Es todo lo que necesita para aprender HTML, entre estos tenemos los que se presentan a continuación con su respectivo link de descarga.

- Atom.

Link: <https://atom.io/>

- Notepad++

Link: <https://notepad-plus-plus.org/>

- Sublime

Link: <https://www.sublimetext.com/>

- Visual Studio Code

Link: <https://code.visualstudio.com/#meet-intellisense>

3.4.2.1.3 HTML básico

Todos los documentos HTML deben comenzar con una declaración de tipo de documento: `<!DOCTYPE html>`.

El documento HTML en sí comienza con `<html>` y termina con `</html>`.

La parte visible del documento HTML está entre `<body>` y `</body>`.

Muchas de las etiquetas que se explican en el siguiente ejercicio no se han visto pero a medida que vamos avanzando en el contenido del módulo.

- **Documento HTML básico**

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1>Bienvenidos a mi primera Página</h1>
```

```
<p>Ensayo, Aprendiendo HTML.</p>
```

```
</body>
```

```
</html>
```

RESULTADO

Bienvenidos a mi primera Página

Ensayo, Aprendiendo HTML.

Ilustración 14 Pruebas de a código HTML

EXPLICACIÓN

- La `<!DOCTYPE html>` declaración define que este documento es un documento HTML5
- El `<html>` elemento es el elemento raíz de una página HTML.
- El `<head>` elemento contiene metainformación sobre la página HTML.
- El `<title>` elemento especifica un título para la página HTML (que se muestra en la barra de título del navegador o en la pestaña de la página)
- El `<body>` elemento define el cuerpo del documento y es un contenedor para todos los contenidos visibles, como encabezados, párrafos, imágenes, hipervínculos, tablas, listas, etc.
- El `<h1>` elemento define un encabezado grande
- El `<p>` elemento define un párrafo

La declaración `<!DOCTYPE>`

La `<!DOCTYPE>` declaración representa el tipo de documento y ayuda a los navegadores a mostrar las páginas web correctamente.

Solo debe aparecer una vez, en la parte superior de la página (antes de cualquier etiqueta HTML).

La `<!DOCTYPE>` declaración no distingue entre mayúsculas y minúsculas.

La `<!DOCTYPE>` declaración para HTML5 es:

```
<!DOCTYPE html>
```


Encabezados HTML

Los encabezados HTML se definen con las etiquetas <h1> to <h6>.

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1>Parte 1</h1>
```

```
<h2>Parte 1</h2>
```

```
<h3>Parte 1</h3>
```

```
<h4>Parte 1</h4>
```

```
<h5>Parte 1</h5>
```

```
<h6>Parte 1</h6>
```

```
</body>
```

```
</html>
```

RESULTADO

Parte 1

Parte 1

Parte 1

Parte 1

Parte 1

Parte 1

Ilustración 15 Pruebas de a código HTML

Párrafos HTML

Los párrafos HTML se definen con la <p>etiqueta:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p>Gloria Lora.</p>
```

```
<p>Gloria Lora.</p>
```

```
</body>
```

```
</html>
```

RESULTADO

Gloria Lora.

Gloria Lora.

Ilustración 16 Pruebas de a código HTML

Enlaces HTML

Los enlaces HTML se definen con la <a>etiqueta:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>Link Ingreso Universidad</h2>
```

```
<p>Link Ingreso</p>
```

```
<a href="https://www.uniremington.edu.co">Link</a>
```


</body>

</html>

RESULTADO

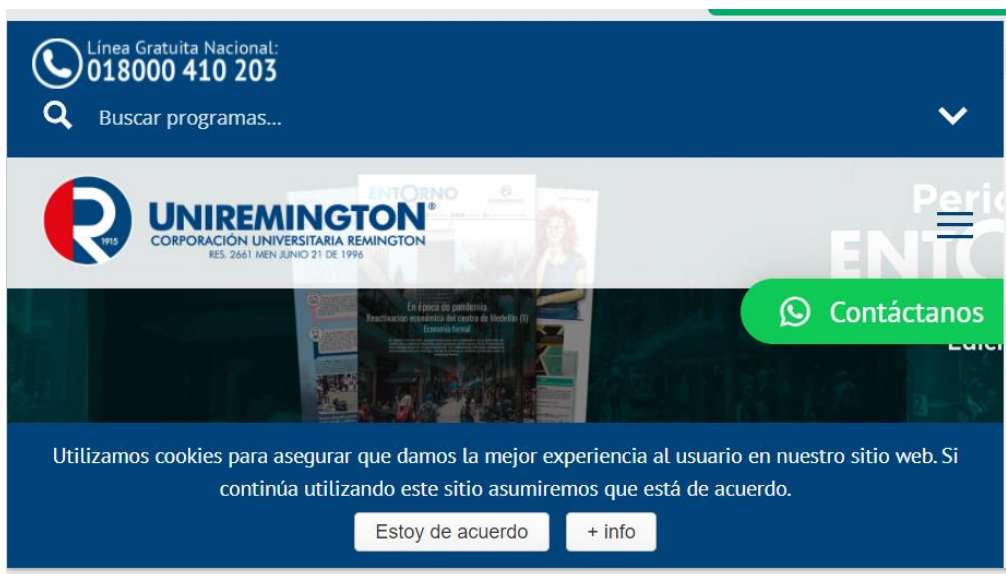


Ilustración 17 Pruebas de a código HTML

Imágenes HTML

Las imágenes HTML se definen con la etiqueta.

El archivo de origen (src), el texto alternativo (alt) widthy heightse proporcionan como atributos:

<!DOCTYPE html>

<html>

<body>

<h2>Manejo de Imagenes</h2>

<p>Manejo de Imagenes en HTML:</p>

</body>

</html>

RESULTADO

Manejo de Imagenes

Ejemplo:



Ilustración 18 Pruebas de a código HTML

3.4.2.1.4 Atributos HTML

Todos los elementos HTML pueden tener atributos

- Los atributos proporcionan información adicional sobre los elementos.
- Los atributos siempre se especifican en la etiqueta de inicio
- Los atributos generalmente vienen en pares de nombre / valor como: nombre = "valor"

A. El atributo href

La <a> etiqueta define un hipervínculo. El href atributo especifica la URL de la página a la que va el enlace:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
    <h2>Atributo href</h2>
```

<p>Los enlaces HTML se definen con la etiqueta a. La dirección del enlace se especifica en el atributo href :</p>

Visita Uniremington

</body>

</html>

RESULTADO

Atributo href

Los enlaces HTML se definen con la etiqueta a. La dirección del enlace se especifica en el atributo href :

[Visita Uniremington](https://www.uniremington.edu.co/)

Ilustración 8 Pruebas de a código HTML

Al dar clic Visita Uniremington abre la página WEB institucional.

B. El atributo src

La etiqueta se utiliza para incrustar una imagen en una página HTML. El src atributo especifica la ruta a la imagen que se mostrará:

<!DOCTYPE html>

<html>

<body>

<h2>Atributo Src</h2>

<p>Las imágenes HTML se definen con la etiqueta img y el nombre de archivo de la fuente de la imagen se especifica en el atributo src:</p>

</body>

</html>

RESULTADO

Atributo Src

Las imágenes HTML se definen con la etiqueta `img` y el nombre de archivo de la fuente de la imagen se especifica en el atributo `src`:



Ilustración 9 Pruebas de a código HTML

Hay dos formas de especificar la URL en el `src` atributo:

1. URL absoluta : enlaces a una imagen externa alojada en otro sitio web. Ejemplo: `src = "https://www.ejemplo.com/images/img_girl.jpg"`.
2. URL relativa : enlaces a una imagen alojada en el sitio web. Aquí, la URL no incluye el nombre de dominio. Si la URL comienza sin una barra, será relativa a la página actual. Ejemplo: `src = "img_girl.jpg"`. Si la URL comienza con una barra, será relativa al dominio. Ejemplo: `src = "/images/img_girl.jpg"`.

Sugerencia: casi siempre es mejor usar URL relativas. No se romperán si cambia de dominio.

C. El atributo lang

Siempre debe incluir el `lang` atributo dentro de la `<html>` etiqueta, para declarar el idioma de la página web. Esto está destinado a ayudar a los motores de búsqueda y navegadores.

El siguiente ejemplo especifica el inglés como idioma:

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<body>
```

```
...
```

```
</body>
```

```
</html>
```

D. El atributo del título

El title atributo define información adicional sobre un elemento. El valor del atributo de título se mostrará como información sobre herramientas cuando pase el mouse sobre el elemento:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2 title="Yo soy un título">Atributo título</h2>
```

```
<p title="Yo soy un título"> Coloca el mouse sobre este párrafo para mostrar el atributo de título como descripción emergente.</p>
```

```
</body>
```

```
</html>
```

RESULTADO

Atributo título

Coloca el mouse sobre este párrafo para mostrar el atributo de título como descripción emergente..

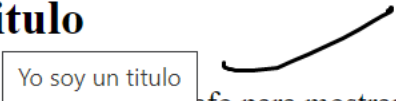


Ilustración 10 Pruebas de a código HTML

3.4.2.1.5 Estilos

A. Estilos HTML

El style atributo HTML se usa para agregar estilos a un elemento, como color, fuente, tamaño y más.

- Utilice el style atributo para diseñar elementos HTML

- Usar background-color para color de fondo
- Usar color para colores de texto
- Usar font-family para fuentes de texto
- Usar font-size para tamaños de texto
- Usar text-align para alineación de texto

Ejemplo:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body style="background-color:powderblue;">
```

```
<p>LETRA NORMAL</p>
```

```
<p style="color:red;">ROJO</p>
```

```
<p style="color:blue;">AZUL</p>
```

```
<p style="font-size:50px;">TAMAÑO GRANDE</p>
```

```
<h1 style="font-family:verdana;">TIPO LETRA VERDANA</h1>
```

```
<h1 style="font-size:300%;">AUMENTO AL 300</h1>
```

```
<h1 style="text-align:center;">TEXTO CENTRADO</h1>
```

```
</body>
```

```
</html>
```

RESULTADO

AUMENTO AL 300

TEXTO CENTRADO

LETRA NORMAL

ROJO

AZUL

TAMAÑO GRANDE

TIPO LETRA VERDANA

Ilustración 11 Pruebas de a código HTML

3.4.2.1.6 Formatos

Los elementos de formato HTML se diseñaron para mostrar tipos especiales de texto:

 - Texto en negrita

 - texto importante

<i> - Texto en cursiva

 - Texto enfatizado

<mark> - Texto marcado

<small> - Texto más pequeño

 - texto eliminado

<ins> - texto insertado

<sub> - Texto de subíndice

<sup> - Texto en superíndice

Ejemplo con algunos formatos:

<!DOCTYPE html>

<html>

<body>

<p>TEXTO NORMAL.</p>

<p>TEXTO MUY IMPORTANTE!</p>

<i>CURSIVA</i>

ENFATIZA

<p>NO OLVIDES COMPAR <mark>MILO</mark> HOY.</p>

<p>MI COLOR FAVORITO ESAZUL ROJO.</p>

<p>TEXTO _{SUBINDICE} TEXTO.</p>

<p>TEXTO ^{SUPERINDICE} TEXTO.</p>

</body>

</html>

RESULTADO

TEXTO NORMAL.

TEXTO MUY IMPORTANTE!

CURSIVA ENFATIZA

NO OLVIDES COMPAR **MILO** HOY.

MI COLOR FAVORITO ES ~~AZUL~~ ROJO.

TEXTO_{SUBINDICE} TEXTO.

TEXTO^{SUPERINDICE} TEXTO.

Ilustración 12 Pruebas de a código HTML

3.4.2.1.7 Tablas

Las tablas HTML permiten a los desarrolladores web organizar los datos en filas y columnas

Algunas etiquetas usadas son:

- <tabla> Define una tabla
- <th> Define una celda de encabezado en una tabla
- <tr> Define una fila en una tabla
- <td> Define una celda en una tabla
- <caption> Define un título de tabla

- `<colgroup>` Especifica un grupo de una o más columnas en una tabla para formatear
- `<col>` Especifica las propiedades de columna para cada columna dentro de un elemento `<colgroup>`
- `<thead>` Agrupa el contenido del encabezado en una tabla
- `<tbody>` Agrupa el contenido del cuerpo en una tabla
- `<tfoot>` Agrupa el contenido del pie de página en una tabla

Ejemplo usando algunas etiquetas:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
table, th, td {
```

```
border: 1px solid black;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1>TABLAS EN HTML </h1>
```

```
<table>
```

```
<caption>Libros Adquiridos</caption>
```

```
<colgroup>
```

```
<col span="2" style="background-color:red">
```

```
<col style="background-color:yellow">
```



```
</colgroup>

<tr>

<th>ISBN</th>

<th>Title</th>

<th>Price</th>

</tr>

<tr>

<td>3476896</td>

<td>My first HTML</td>

<td>$53</td>

</tr>

<tr>

<td>5869207</td>

<td>My first CSS</td>

<td>$49</td>

</tr>

</table>

</body>

</html>
```

RESULTADO

TABLAS EN HTML

Libros Adquiridos

ISBN	Title	Price
3476896	My first HTML	\$53
5869207	My first CSS	\$49

Ilustración 13 Pruebas de a código HTML

3.4.2.1.8 Listas

Las listas HTML permiten a los desarrolladores web agrupar un conjunto de elementos relacionados en listas.

Ejemplo

Una lista HTML desordenada:

- Artículo
- Artículo
- Artículo
- Artículo

Una lista HTML ordenada:

1. Primer elemento
2. Segundo artículo
3. Tercer artículo
4. Cuarto elemento

Etiquetas:

- `` Define una lista desordenada
- `` Define una lista ordenada
- `` Define un elemento de lista
- `<dl>` Define una lista de descripciones
- `<dt>` Define un término en una lista de descripción
- `<dd>` Describe el término en una lista de descripción

Ejemplo usando algunas etiquetas:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>EJEMPLO DE LISTAS ORDENADAS DESORNEDAS Y DE DESCRIPCION</h2>
```

```
<ul>
```

```
<li>Coffee</li>
```

```
<li>Tea</li>
```

```
<li>Milk</li>
```

```
<ol>
```

```
<li>Coffee</li>
```

```
<li>Tea</li>
```

```
<li>Milk</li>
```

```
</ol>
```

```
</ul>
```

```
<dl>
```

```
<dt>Coffee</dt>

<dd>- black hot drink</dd>

<dt>Milk</dt>

<dd>- white cold drink</dd>

</dl>

</body>

</html>
```

RESULTADO

EJEMPLO DE LISTAS ORDENADAS DESORNEDAS Y DE DESCRIPCION

- Coffee
 - Tea
 - Milk
1. Coffee
 2. Tea
 3. Milk

Coffee

- black hot drink

Milk

- white cold drink

Ilustración 14 Pruebas de a código HTML

3.4.2.1.9 Formularios en HTML

Se utiliza un formulario HTML para recopilar la entrada del usuario. La entrada del usuario se envía con mayor frecuencia a un servidor para su procesamiento.

Elementos:

El <input>elemento HTML es el elemento de formulario más utilizado.

El `<input type="text">` define un campo de entrada de una sola línea para la entrada de texto.

El `<label>` etiqueta define una etiqueta para muchos elementos de formulario.

El `<input type="radio">` define un botón de radio. Los botones de radio permiten al usuario seleccionar UNA de un número limitado de opciones.

El `<input type="checkbox">` define una casilla de verificación.

Ejemplo usando algunas etiquetas

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1>Ejemplo usando algunas opciones del Formulario</h1>
```

```
<p>Se especifican algunas etiquetas:</p>
```

```
<form action="/action_page.php">
```

```
<label for="fname">Primer Nombre:</label><br>
```

```
<input type="text" id="fname" name="fname" value="John"><br>
```

```
<label for="lname">Apellido:</label><br>
```

```
<input type="text" id="lname" name="lname" value="Doe"><br><br>
```

```
<input type="submit" value="Ingresar">
```

```
<p>Radio Button:</p>
```

```
<input type="radio" id="html" name="fav_language" value="HTML">
```

```
<label for="html">HTML</label><br>
```

```
<input type="radio" id="css" name="fav_language" value="CSS">
```

```
<label for="css">CSS</label><br>
```

```
<input type="radio" id="javascript" name="fav_language" value="JavaScript">
```

```
<label for="javascript">JavaScript</label>
```

<p>Casilla de verificacion:</p>

<input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">

<label for="vehicle1"> I have a bike</label>

<input type="checkbox" id="vehicle2" name="vehicle2" value="Car">

<label for="vehicle2"> I have a car</label>

<input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">

<label for="vehicle3"> I have a boat</label>

</form>

</body>

</html>

RESULTADO

Ejemplo usando algunas opciones del Formulario

Se especifican algunas etiquetas:

Primer Nombre:

Apellido:

Radio Button:

- ☐ HTML
- ☐ CSS
- ☐ JavaScript

Casilla de verificación:

- ☐ I have a bike
- ☐ I have a car
- ☐ I have a boat

Ilustración 26 Pruebas de a código HTML

3.4.2.2 CSS

A las páginas web cada vez se le exigía más potencialidad y más diseño. Ya hemos visto que la aparición de HTML 5 complementa HTML con nuevas etiquetas, pero las propiedades HTML resultaban escasas para mejorar el diseño y surgió CSS y CSS3 que lograban mejorar el aspecto de las Webs.

Hoy en día todas las páginas hacen uso de CSS, bien aplicando características a las etiquetas o haciendo uso de plantillas. Las utilidades que te permiten crear páginas hacen uso de CSS para cambiar colores, tamaños, fondos, por tanto para complementar lo aprendido hasta ahora necesitamos aprender CSS.

Ejemplo usando etiquetas CSS:

```
<html>
```

```
<head>
```

```
<title>PAGINA CON ESTILO EN EL TITULO DE TIPO H1</title>
```



```
<style type='text/css'>
```

```
h1{ color:blue;
```

```
text-decoration:underline;
```

```
text-align:center;
```

```
}
```

```
p{
```

```
font-family:verdana;
```

```
background-color:pink;
```

```
color:green
```

```
}
```

```
body{
```

```
color:red;
```

```
background-color:#cccccc;
```

```
text-indent:1cm
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1>PAGINA CON ESTILO EN EL TITULO DE TIPO H1</h1>
```

hola en el body directamente

```
<p>Muestro el estilo de un parrafo</p>
```

```
</body>
```

```
</html>
```

El código color verde es HTML, con la etiqueta style indicamos que el código que se encuentra dentro antes del cierre `</style>` escrito en color azul es CSS.

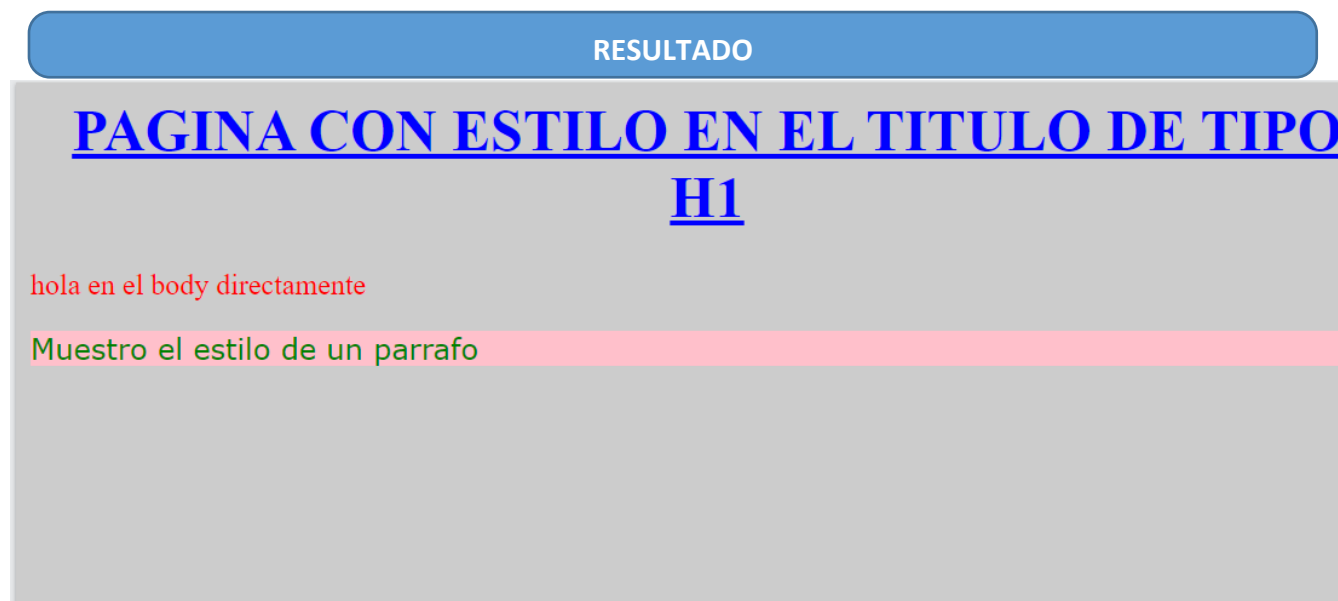


Ilustración 15 Pruebas de a código CSS

3.4.2.2.1 Sintaxis

Un archivo CSS puede estar compuesto por varias reglas de estilo, y cada una de esas reglas debe tener 2 partes importantes, el selector y las propiedades. A continuación, se muestra un ejemplo de una regla en un archivo CSS:

```
h1{  
  color: blue.  
  font size:12px.  
}
```

Del ejemplo anterior se puede ver que el selector corresponde a `h1` y todo lo demás corresponde al bloque de propiedades. Además, también se puede deducir que a las etiquetas `h1` de HTML se les va a aplicar un tamaño de fuente de 12px y un color azul.

3.4.2.2.2 Selectores

Tabla 9 Selectores en CSS

TIPO	EJEMPLO
Selector universal: Se utiliza para seleccionar todos los elementos de la página	<pre>* { margin: 0; padding: 0; }</pre>
Selector de tipo o etiqueta: Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector. Por ejemplo, a todos los párrafos (etiqueta P), a todas las listas desordenadas (etiqueta UL) o a todos los enlaces (etiqueta A).	<pre>p { ... }</pre>
Selector descendente: Selecciona los elementos que se encuentran dentro de otros elementos. Un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento	<p>selector1 selector2 selector3 ... selectorN.</p> <p>En este ejemplo El selector descendente se compone de cuatro selectores, miremos este ejemplo:</p> <pre>p a span em { text-decoration: underline; }</pre>
Selectores de clase: Los selectores de clase son muy importantes, porque nos permiten aplicar estilos a conjuntos de elementos de manera muy versátil.	<p>Sintaxis: .classname</p> <p>Ejemplo:.index eleccionará cualquier elemento que tenga la clase "index".</p>
Selector de atributo: Selecciona elementos basándose en el valor de un determinado atributo.	<pre>[attr] [attr=value] [attr~=value] [attr =value] [attr^=value] [attr\$=value] [attr*=value]</pre>

3.4.2.2.3 Propiedades CSS

Tabla 10 Propiedades CSS

TIPO	EJEMPLO
width: Define el ancho de un elemento	<code>width: 20px;</code>
height: Define el alto de un elemento.	<code>height: 20px;</code>
background: Puede definir el color de fondo o la url de fondo de un elemento.	<code>background: url('puppy.png');</code>
border: Define el tamaño, estilo y color del borde de un elemento. Por ejemplo:	<code>border: 2px solid yellow;</code>
margin: Define la margen de un elemento.	<code>margin: 2px</code>
font-size: Define el tamaño de la fuente.	<code>font-size: 20px</code>
font-family: Define la familia tipográfica de la fuente	<code>font-family: 'Roboto', sans-serif;</code>

3.4.3 TEMA 3 LENGUAJES DE MAQUETADO HTML, CSS, JAVASCRIPT

3.4.3.1 INTRODUCCIÓN A JAVASCRIPT

JavaScript o JS es uno de los lenguajes de programación más populares hoy en día y es utilizado para añadir interactividad a páginas web, aplicaciones móviles, juegos y más. Además, este lenguaje de programación no solo se utiliza del lado del cliente, también se ha venido trabajando en frameworks como NodeJS para desarrollar servicios web.

3.4.3.1.1 IDE

Para poder escribir cualquier lenguaje de programación es necesario un editor de texto cualquiera, pero en la actualidad **Visual Studio Code** es el IDE de preferencia de los programadores de JavaScript. Para obtenerlo solo basta ir al enlace <https://code.visualstudio.com/>, descargarlo e instalarlo.

3.4.3.1.2 Variables

Las variables son contenedores para guardar valores y normalmente estos valores pueden ser modificados a medida que se avanza en el programa.

La creación de una variable se puede hacer a través de la palabra reservada `var` tal y como se muestra a continuación:

```
var number = 10;
```

En el ejemplo anterior, el valor de 10 es asignado a la variable ***number***.

Hay tres tipos de variables en Javascript:

- **Var:** Declara una variable, iniciándola opcionalmente a un valor. Podrá cambiar su valor y su Scope es local
- **Let:** Declara una variable local en un bloque de ámbito iniciándola opcionalmente a un valor. Podrá cambiar su valor.
- **Const:** Declara una variable de solo lectura en un bloque de ámbito. No será posible cambiar su valor mediante la asignación.

3.4.3.1.3 Comentarios

Al igual que en otros lenguajes de programación, las líneas que son comentadas no son ejecutadas. Para comentar una sola línea se utilizan dos slash (`//`) y para comentar un bloque de código se utiliza `/*` para indicar el comienzo y `*/` para indicar el final.

3.4.3.1.4 Tipos de datos

Tabla 11 Tipos Dato JavaScript

TIPO DE DATO	EJEMPLO
NUMBER JavaScript tiene un solo tipo de números, es decir que los números decimales y los enteros pertenecen al mismo tipo de dato.	<pre>var valor1 =45.00; var valor2 = 34;</pre>
BOOLEAN Solo Tiene dos valores posibles	Verdadero Falso
STRINGS Es un conjunto de caracteres. Se pueden reconocer porque	<pre>var nombre1 = "GLORIA"; var nombre2 = 'GLORIA';</pre>

están encerrados entre comilla simple o doble comilla.

OBJECTS

Los objetos originalmente en JavaScript son un conjunto de propiedades clave-valor separados por coma. Se dice que originalmente porque gracias a EcmaScript los objetos tienen más componentes de la programación orientada a objetos

```
var persona = {nombre: "GLORIA", apellido: "LORA",  
edad: "100"}
```

ARRAYS

Los arreglos son un conjunto de elementos que están separados por coma y encerrados entre corchetes.

```
var carros = ["TOYOTA", "Volvo", "BMW"];
```

UNDEFINED

Una variable que no tiene ningún valor asignado por defecto está como undefined.

```
var carros;
```

Este valor también se puede asignar a propósito si así se requiere.

```
var carros = undefined;
```

3.4.3.1.5 Tipos de operadores

A continuación, se describen los diferentes operadores que tiene el lenguaje de programación JavaScript:

Tabla 12 Tipos Operadores

ARITMÉTICOS	+	Adición	let x = 5; let y = 9; let z = x + y;
	-	Resta	let x = 5; let y = 9; let z = x - y;
	*	Multiplicación	let x = 5; let y = 9; let z = x * y;
	/	División	let x = 5; let y = 9;

			let z = x / y;
	%	Módulo	let x = 5; let y = 9; let z = x% y;
	++	Incremento	let x = 5; x++; let z = x;
	--	Decremento	let x = 5; x--; let z = x;
	==	Igual a...	x == y
COMPARACIÓN	===	Idéntico (igual y del mismo tipo)	4 == 10 (false)
	!=	No igual a...	8 === 10 (false)
	!==	No idéntico	6 != 10 (true)
	>	Mayor que...	12 !== 10 (true)
	>=	Mayor o igual a...	9 > 5 (true)
	<	Menor que...	10 >= 7 (true)
	<=	Menor o igual a...	10 < 9 (false)
	>=	Mayor o igual que	10 <= 6 (false)
LÓGICOS	&&	Retorna true si ambos operandos son true .	
		Retorna true si al menos un operando es true .	
	!	Retorna true si el operando es false , y false si el operando es true .	
ASIGNACIÓN	=	x = y	
	+=	x += y	
	-=	x -= y	
	*=	x *= y	
	/=	x /= y	
	%=	x %= y	
	<<=	x <<= y	

>>=

>>>=

&=

^=

|=

**=

x >>= y

x >>>= y

x &= y

x ^= y

x |= y

x **= y

3.4.3.1.6 Estructuras de control

En JavaScript tenemos las siguientes declaraciones condicionales:

- Úselo **if** para especificar un bloque de código que se ejecutará, si una condición especificada es verdadera.
- Úselo **else** para especificar un bloque de código que se ejecutará, si la misma condición es falsa.
- Úselo **else if** para especificar una nueva condición para probar, si la primera condición es falsa.
- Úselo **switch** para especificar muchos bloques alternativos de código que se ejecutarán.

3.4.3.1.6.1 IF...ELSE

Se evalúa una condición, si es **true** entonces se ejecuta lo que hay dentro del **if**, pero si es **false** entonces se ejecuta lo que hay dentro del **else**. En caso de que no haya sentencia **else** y la condición es **false** entonces no se ejecutará ninguna instrucción.

```
if (condición) {  
  Instrucciones que serán ejecutadas si la condición es verdadera  
} else {  
  Instrucciones que serán ejecutadas si la condición es falsa  
}
```

Ejemplo:

Si el tiempo es inferior a las 10:00, cree un saludo de "Buenos días", si no es así, pero el tiempo es inferior a las 20:00, cree un saludo de "Buen día", de lo contrario un "Buenas noches":

```
if (time < 10) {  
  greeting = "Buenos dias";  
} else if (time < 20) {
```

```
greeting = "Buen dia";  
} else {  
greeting = "Buenas noches";  
}
```

3.4.3.1.6.2 SWITCH

Se utiliza para elegir uno de muchos bloques de código, dependiendo del valor de la variable que se manda como argumento.

```
switch (variable) {  
  case 1:  
    Instrucciones  
    break.  
  
  case 2:  
    Instrucciones  
    break.  
  ...  
  
  case n:  
    Instrucciones  
    break;  
  
  default:  
    Instrucciones  
}
```

Si el valor no se encuentra en ninguna de las posibilidades entonces se ejecuta lo que esté en **default**. Pero si por el contrario logra entrar a alguna de las posibilidades ejecutará todo lo que esté a su paso hasta que encuentre una sentencia **break**.

Ejemplo:

El `getDay()` método devuelve el día de la semana como un número entre 0 y 6.

(Domingo = 0, lunes = 1, Martes = 2 ...)

Este ejemplo utiliza el número del día de la semana para calcular el nombre del día de la semana:

```
switch (new Date().getDay()) {  
  
  case 0:  
  
    day = "Sunday";  
  
    break;
```

case 1:

day = "Monday";

break;

case 2:

day = "Tuesday";

break;

case 3:

day = "Wednesday";

break;

case 4:

day = "Thursday";

break;

case 5:

day = "Friday";

break;

case 6:

day = "Saturday";

}

3.4.3.1.7 Ciclos

Los ciclos permiten ejecutar bloques de código varias ocasiones. A continuación, se describirán los ciclos que JavaScript permite.

3.4.3.1.7.1 Ciclo WHILE

Se ejecutará mientras la condición sea **true**.

```
while (condición) {
```

```
Instrucciones que serán ejecutadas si la condición es verdadera
}
```

EJEMPLO:

```
while (i < 10) {
  text += "The number is " + i;
  i++;
}
```

3.4.3.1.7.2 Ciclo FOR

A diferencia del ciclo **while**, en el ciclo **for** se puede especificar el número finito de veces que se ejecutará un bloque de instrucciones; esto se logra tomando en cuenta el valor inicial, una condición y un incremento o decremento según el caso.

```
for (inicialización; condición; incremento o decremento) {
  Instrucciones que serán ejecutadas siempre y cuando la condición sea verdadera
}
```

EJEMPLO:

```
for (let i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}
```

3.4.3.1.7.3 Ciclo FOR _ IN

Se utiliza para recorrer las propiedades de un objeto.

```
var persona = {nombre:"GLORIA", apellido:"LORA", edad:100};

var texto = "";
var clave;
for (clave in persona) {
  texto += persona[clave];
}
```

En cada una de las iteraciones del código anterior la variable **clave** tomará una de las claves del objeto, lo que quiere decir que al final del ciclo la variable texto quedará con el string "GLORIA LORA 100", dado que a través de las claves se están accediendo a los respectivos valores.

Ejemplo:

```
const person = {fname:"John", lname:"Doe", age:25};

let text = "";
```

```
for (let x in person) {  
  text += person[x];  
}
```

3.4.3.1.7.4 Sentencia BREAK

Esta sentencia rompe abruptamente un ciclo o una sentencia switch, y permite que se sigan ejecutando las instrucciones que están después (si es que las hay).

EJEMPLO:

```
for (let i = 0; i < 10; i++) {  
  if (i === 3) { break; }  
  text += "The number is " + i + "<br>";  
}
```

3.4.3.1.7.5 Sentencia CONTINUE

Esta sentencia rompe abruptamente solamente una iteración del ciclo, es decir que en el momento que la ejecución se encuentre con un **continue** inmediatamente seguirá con la siguiente iteración.

Ejemplo:

```
for (let i = 0; i < 10; i++) {  
  if (i === 3) { continue; }  
  text += "The number is " + i + "<br>";  
}
```

3.4.3.1.8 Funciones

Una función es un bloque de código designado para realizar una tarea particular.

Es uno de los pilares fundamentales de JavaScript

```
function nombreDeLaFuncion(lista-de-parametros) {  
  instrucciones  
}
```

La ventaja de las funciones es que es código que se puede reutilizar, y así se evita tener que escribir lo mismo varias veces.

3.4.3.1.9 ECMAScript 2015

Es una actualización del lenguaje de programación JavaScript que actualmente todos los navegadores ya implementan en 2021. Todos los agregados a JavaScript definidos en este nuevo estándar ya se los está aplicando desde hace un tiempo con Node.js en el servidor, y en framework de cliente como React, Angular y Vue. Estos agregados al lenguaje JavaScript nos facilitan la programación.

En el siguiente link <https://caniuse.com/?search=const>, nos sirve para saber qué porcentaje de navegadores soportan la palabra clave 'const' para definir constantes

En ES6 hay 3 formas de escribir una variable:

```
var número = 10;
const saludo = 'hello';
let variable = true;
```

El tipo de declaración se usa dependiendo del alcance. El alcance es un concepto fundamental en los lenguajes de programación que define la visibilidad de una variable. A diferencia de la palabra reservada **var**, quien define una variable global o localmente para una función, **let** permite declarar variables que son limitadas por el alcance de un bloque, sentencia o expresión en el cual está siendo usado.

```
if (true) {
  let nombre = 'Gloria Lora'.
}
alert(nombre); //genera un error
```

En este caso, la variable **nombre** es accedida únicamente en el alcance de la sentencia **if** porque fue declarada con **let**, lo que quiere decir que esta variable no existe por fuera del **if**.

El **const** tiene el mismo alcance que el **let**. La diferencia es que las variables de tipo const son inmutables, es decir, no se permite que sean reasignadas.

```
const saludo = 'Hello';
saludo = 'Bye'; //genera un error
```

3.4.3.1.10 Clases

ECMAScript 2015, también conocido como ES6, introdujo las clases de JavaScript.

Las clases de JavaScript son plantillas para objetos de JavaScript.

Se puede crear una clase para crear múltiples objetos de la misma estructura.

```
class Rectangulo {
  constructor(altura, ancho) {
    this.altura = altura;
    this.ancho = ancho;
  }
}
```

```
}  
}
```

```
const cuadrado = new Rectangulo(5, 5);  
const poster = new Rectangulo(2, 3);
```

3.4.3.1.11 Objetos

Los objetos son creados gracias al operador **new**, y se debe tener en cuenta que en la clase solo se puede tener un constructor. En la vida real, un automóvil es un objeto. Un automóvil tiene propiedades como el peso y el color, y métodos como arrancar y parar.

Hay dos formas de crear un objeto:

```
Var objeto=new Object();
```

```
Var objeto=[] objeto literal.
```

```
Var persona{  
  
    Nombre='Maria',  
  
    edad:28;  
  
    profesión:'Astronauta'  
  
}
```

Podemos acceder a sus propiedades de un objeto a través de

- Puntos: `persona.nombre`
- Corchetes: `persona['nombre']`

Podemos crear miembros completamente nuevos:

```
Persona.pais='Canada'
```

3.5 RECURSOS

3.5.1 Documentación Referenciada

Tema	Enlace URL	Especificación de páginas o instrucciones de lectura
Interfaces de usuario	https://elibro.net/es/ereader/remington/62487	Todo el libro.
HTML CSS JAVASCRIPT	https://elibro.net/es/ereader/remington/183103	Todo el libro.
HTML CSS JAVASCRIPT	https://elibro.net/es/ereader/remington/106414	Todo el libro.
Interfaces de usuario	https://elibro.net/es/ereader/remington/56294	217-236

3.5.2 Material de profundización

Tema	Enlace URL	Especificación de páginas o instrucciones de lectura
HTML, CSS y JAVASCRIPT	https://core.ac.uk/download/pdf/235988523.pdf	Leer todo el artículo.
Interfaces de usuario	http://sedici.unlp.edu.ar/bitstream/handle/10915/41578/Documento_completo.pdf?sequence=1&isAllowed=y	Leer todo el artículo.

3.6 EJERCICIOS DE PROFUNDIZACIÓN

- Desarrolle una calculadora usando HTML, CSS, JAVASCRIPT donde se puedan realizar las operaciones básicas de suma, resta, multiplicación, división además del cálculo de porcentajes. Debe de ser totalmente funcional.
- Cree una aplicación web tipo CRUD que permita administrar la información de los estudiantes de una institución Técnica. Para ello permita que se pueda modificar una lista de objetos de JavaScript a través de una interfaz de usuario, alojada en un servidor WEB.

Cada Docente debe contener la siguiente información:

- Tipo Documento (C.C, C.E, TI, NIUP)
- Nombre
- Apellido
- Fecha Nacimiento (DD/MM/AAAA)
- Nivel de estudios (Pregrado, maestría, Doctorado)
- Área a la que pertenece (Matemáticas, Ciencias, Sistemas)

- Grado (Primer Semestre, Segundo Semestre, Tercer Semestre, Cuarto Semestre)
- EPS
- Salario

3.7 PISTAS DE APRENDIZAJE



Recuerde que:

- Diseño de interfaz

- HTML5

- CSS

- JavaScript

4 UNIDAD 3 FRAMEWORK PARA EL DESARROLLO WEB

4.1 OBJETIVOS

4.1.1 General

Implementar el uso de framework para el desarrollo de sitios web.

4.1.2 Específicos

- Implementar framework de desarrollo Front End.
- Mostrar cómo se consumen API a través del Framework que se esté usando.

4.2 RELACIÓN DE CONCEPTOS

4.2.1 Mapa Mental o Conceptual

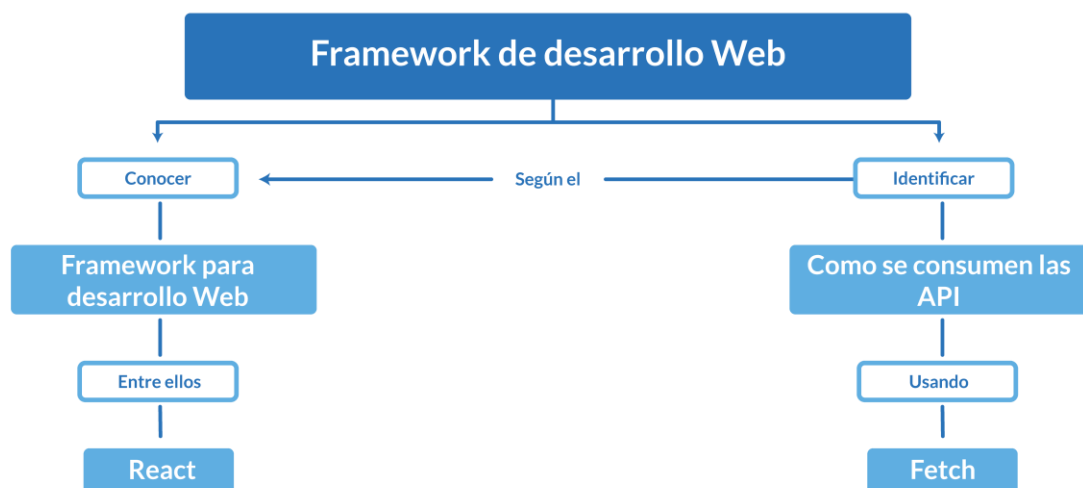


Ilustración 16 Relación Conceptos Unidad 3

4.2.2 Lista de Conceptos

Framework: Esquema o patrón que ofrece un entorno genérico para escribir código en un lenguaje concreto.



Fetch: La API Fetch proporciona una interfaz JavaScript para acceder y manipular partes del canal HTTP, tales como peticiones y respuestas. También provee un método global `fetch()` que proporciona una forma fácil y lógica de obtener recursos de forma asíncrona por la red.

API: La expresión Application Programming Interface o en español, Interfaz de Programación de Aplicaciones, originó el acrónimo API. Las API son «traductores» cuya función es conectar sistemas, software y aplicaciones.

React: Biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página.

4.3 PRESENTACIÓN DE LA UNIDAD

El término *framework*, es una estructura de software compuesta de componentes personalizables y comunes para el desarrollo de una aplicación, un *framework* se puede considerar como una aplicación genérica incompleta y configurable a la que se le añade las últimas piezas para construir una aplicación concreta. Su objetivo es acelerar el proceso de desarrollo de aplicativos de software, reutilizando código ya existente y promover buenas prácticas, también es importante conocer el proceso de consumo de API a través de estos.

4.4 DESARROLLO TEMÁTICO

4.4.1 TEMA 1 FRAMEWORK JAVASCRIPT PARA FRONTEND

Adicionalmente de las virtudes mencionadas respecto a los beneficios de un *framework*, los marcos de trabajo de JavaScript surgen como una respuesta a la necesidad del correcto funcionamiento de las aplicaciones web que conviven en los ambientes de los navegadores, tales como: Chrome, Firefox, Opera, Safari e Internet Explorer.

Algunas ventajas son:

Ahorro de tiempo: Una de las mayores problemáticas que ha tenido el desarrollo de aplicaciones web es su alto tiempo de implementación.

Mayor velocidad: Contar con estas herramientas de trabajo ha permitido a los equipos de desarrollo aumentar su velocidad de implementación de las soluciones de software

Mejor entendimiento de código: Usar las mejores prácticas y estándares de programación ayudan que los productos codificados por los desarrolladores sean más entendibles y claros para los demás.

Algunas desventajas:

Dependencia de una librería: la expansión de las aplicaciones es sin lugar a duda una realidad que se debe tener en cuenta en los proyectos a corto y largo plazo. Esto implica que los marcos de trabajo en JavaScript deben contar con mayores prestaciones que les permitan a las aplicaciones crecer de una manera más rápida y sin mayores traumatismos.

Demanda de recursos: implementar cualquier librería o marco de trabajo, representa un consumo considerable en cualquier aplicación de software, debido a que internamente estas cuentan con procesos y rutinas que demandan recursos a los clientes que están usando estas aplicaciones.

Los principales framework JavaScript FrontEnd

- **React:** Uno de los frameworks web más usados por parte de los desarrolladores. Es una librería de Facebook enfocada en la creación de vistas con particularidades como los patrones de eventos que permiten actualizar las mencionadas vistas cuando los datos hayan sido modificados, aportando una carga en tiempo real a nuestro desarrollo.
- **Angular:** Este framework, y el resto de sus versiones fue creado inicialmente y es promovido por Google, entre sus características encontramos que modifica el DOM directamente y alguna que otra característica revolucionaria para su época que hoy en día nos sorprenden poco, habiendo visto el panorama que hay de frameworks y librerías.
- **VueJs:** Vue es una iniciativa de un ex-empleado de Google, que había estado relacionado con AngularJS y que pretendía coger sus partes favoritas de este framework y crear algo muy ligero. En 2014 fue liberado a la comunidad y se encuentra disponible de manera Open Source.

Este módulo solo se centrará en el framework React

4.4.1.1.1 React

React es una librería JavaScript desarrollada principalmente por el grande las redes sociales Facebook. Se fundamente principalmente como software libre y desde su liberación al público ha acaparado una creciente comunidad de desarrolladores que buscan ayudar al crecimiento de esta fantástica tecnología. La creación de este framework se derivada de las diferentes necesidades presentes en la popular red social, en su carrera por brindar unaexcelente experiencia de usuario a su público.

¿Cómo instalar React?

Antes de iniciar la instalación de cualquier entorno o framework, es importante tener en cuenta la página oficial de este, con el fin de referenciar la documentación que nos permita realizar todas las

configuraciones iniciales de forma adecuada y estar más actualizado en torno a su manera de uso. En el caso de REACT su sitio oficial es <https://reactjs.org/> y su sección de documentación es <https://reactjs.org/docs>

Para instalar REACT en nuestras aplicaciones se puede realizar de la siguiente manera:

Lo primero que se debe tener en cuenta es instalar los siguientes programas que se listan a continuación para facilitar la creación de una aplicación en REACT.

Node.js: Es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Gracias a node se podrá gestionar los paquetes necesarios que se necesiten instalar en el proyecto de REACT.

Para instalar node.js, se debe descargar desde el siguiente enlace

<https://nodejs.org/es/download/>.

Dependiendo de la versión de su sistema operativo, instale la que corresponda según el sistema operativo que tenga.

Una vez se tenga instalado node, se puede verificar la versión que quedo en la máquina. Por lo cual debemos abrir una terminal (la terminal, CMD o símbolo del sistema en Windows, es desde donde se puede ejecutar los comandos que se reconozcan desde el sistema operativo, tales como dir, mkdir, cd, etc) y procedemos a ejecutar el siguiente comando: node -v

Node.js command prompt

```
Your environment has been set up for using Node.js 14.18.1 (x64) and npm.

C:\Users\Usuario>node -v
v14.18.1

C:\Users\Usuario>
```

Ilustración 17 Versión Node.js

Node Package Manager o simplemente npm Es un gestor de paquetes, el cual hará más fáciles nuestras vidas al momento de trabajar con Node, ya que gracias a él podremos tener cualquier librería disponible con solo una línea de código, npm nos ayudará a administrar nuestros módulos, distribuir paquetes y agregar dependencias de una manera sencilla.

Con la instalación de node.js, se instala NPM, por ello se puede verificar la versión instalada desde la terminal, ejecutando el siguiente comando: `npm -v`.

Esta puede variar, dependiendo del equipo no necesariamente tiene que aparecer la de la imagen.

Yarn: YARN es un gestor de dependencias de JavaScript, que está enfocado en la velocidad y la seguridad, y a diferencia de otros gestores como NPM, YARN es muy rápido y fácil de usar.

A continuación, se creará la primera aplicación REACT, para ello se empleará Create React App, esta es un ambiente cómodo para aprender React, y es la mejor manera de comenzar a construir una nueva aplicación de página única usando React.

Create React App configura tu ambiente de desarrollo de forma que puedas usar las últimas características de Javascript, brindando una buena experiencia de desarrollo, y optimizando tu aplicación para producción. Necesitarás tener Node ≥ 8.10 y npm ≥ 5.6 instalados en tu máquina. Para crear un proyecto ejecuta:

`npx create-react-app mi-aplicación`

```
Node.js command prompt

Creating a new React app in C:\Users\Usuario\mi-aplicacion.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

core-js@2.6.12 postinstall C:\Users\Usuario\mi-aplicacion\node_modules\babel-runtime\node_modules\core-js
node -e "try{require('./postinstall')}catch(e){}"

core-js@3.18.3 postinstall C:\Users\Usuario\mi-aplicacion\node_modules\core-js
node -e "try{require('./postinstall')}catch(e){}"

core-js-pure@3.18.3 postinstall C:\Users\Usuario\mi-aplicacion\node_modules\core-js-pure
node -e "try{require('./postinstall')}catch(e){}"

ejs@2.7.4 postinstall C:\Users\Usuario\mi-aplicacion\node_modules\ejs
node ./postinstall.js

cra-template@1.1.2
react@17.0.2
react-dom@17.0.2
react-scripts@4.0.3
added 1867 packages from 708 contributors and audited 1870 packages in 166.563s

52 packages are looking for funding
run 'npm fund' for details
```

Ilustración 18 Creación proyecto en React

Una vez se completa la ejecución de este comando ya se tendrá creada el scaffolding de la aplicación.

Al finalizar, se ingresa a la carpeta y se abre el proyecto con algún editor para este caso se usa Visual Studio Code, como se visualiza en la imagen. También se pueden visualizar algunos comandos que se pueden ejecutar para la ejecución de la aplicación.

```
Node.js command prompt

Success! Created mi-aplicacion at C:\Users\Usuario\mi-aplicacion
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd mi-aplicacion
  npm start

Happy hacking!
C:\Users\Usuario>
```

Scripts

Estos scripts, serán los comandos que más se usen para la ejecución del proyecto en REACT:

npm start o yarn start: inicia el servidor de desarrollo y abre un navegador con la aplicación.

npm test o yarn test: ejecuta las pruebas.

npm run build o yarn build: empaqueta la aplicación para producción en la carpeta build.

npm run eject o yarn eject: permite cambiar manualmente las librerías y configuración que utiliza create-react-app por defecto. Ten cuidado con este comando, una vez lo ejecutas no hay marcha atrás.

Para mi caso voy y busco el archivo en la carpeta User del disco local c:

Al abrir este archivo usando el editor nos muestra lo siguiente:


```
1 {
2   "name": "mi-aplicacion",
3   "version": "0.1.0",
4   "private": true,
5   "dependencies": {
6     "@testing-library/jest-dom": "^5.14.1",
7     "@testing-library/react": "^11.2.7",
8     "@testing-library/user-event": "^12.8.3",
9     "react": "^17.0.2",
10    "react-dom": "^17.0.2",
11    "react-scripts": "4.0.3",
12    "web-vitals": "^1.1.2"
13  },
14  "scripts": {
15    "start": "react-scripts start",
16    "build": "react-scripts build",
17    "test": "react-scripts test",
18    "eject": "react-scripts eject"
19  },
20  "eslintConfig": {
21    "extends": [
22      "react-app",
23      "react-app/jest"
24    ]
25  },
26  "browserslist": {
27    "production": [
28      ">0.2%",
29      "not dead",
30      "not op_mini all"
31    ]
32  }
33 }
```

Ilustración 19 Archivo React

Edite el archivo `src/App.js`, para adicionar una etiqueta `p` con el mensaje: Hola Estudiante Uniremington

```
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>
10           Edit <code>src/App.js</code> and save to reload.
11           <p>Hola Estudiante Uniremington </p>
12         </p>
13       </div>
14     <a
15       className="App-link"
16       href="https://reactjs.org"
17       target="_blank"
18       rel="noopener noreferrer"
19     >
20       Learn React
21     </a>
22   );
23 }
```

Ilustración 20 Archivo editado en React

Por último, ejecute la aplicación con el comando: `yarn start` o `npm start`. Para lo cual abra una terminal que se encuentra en la ruta: Terminal -> New Terminal.

```
C:\Users\Usuario\mi-aplicacion>npm start

> mi-aplicacion@0.1.0 start C:\Users\Usuario\mi-aplicacion
> react-scripts start

i @wds: Project is running at http://192.168.1.8/
i @wds: webpack output is served from
i @wds: Content not from webpack is served from C:\Users\Usuario\mi-aplicacion\public
i @wds: 404s will fallback to /
Starting the development server...
Compiled successfully!

You can now view mi-aplicacion in the browser.

Local:      http://localhost:3000
On Your Network:  http://192.168.1.8:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

Luego se abre el navegador.

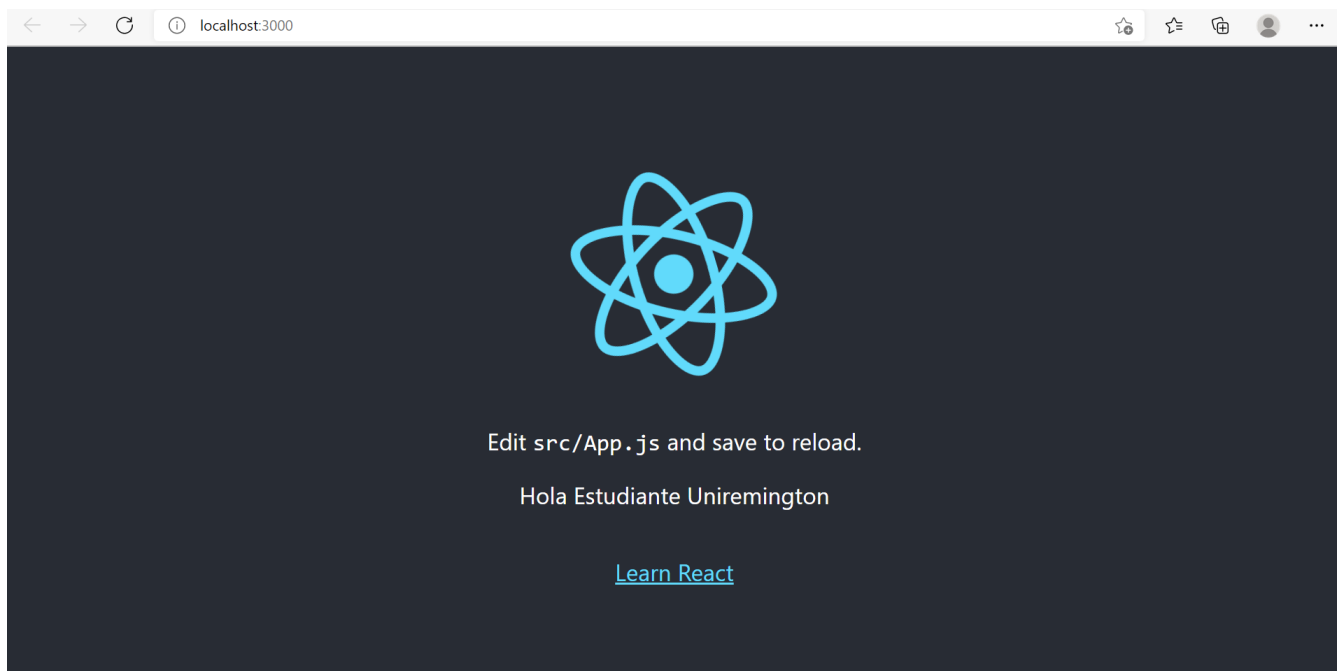


Ilustración 33 Ejecución React

Estructura de un proyecto en React

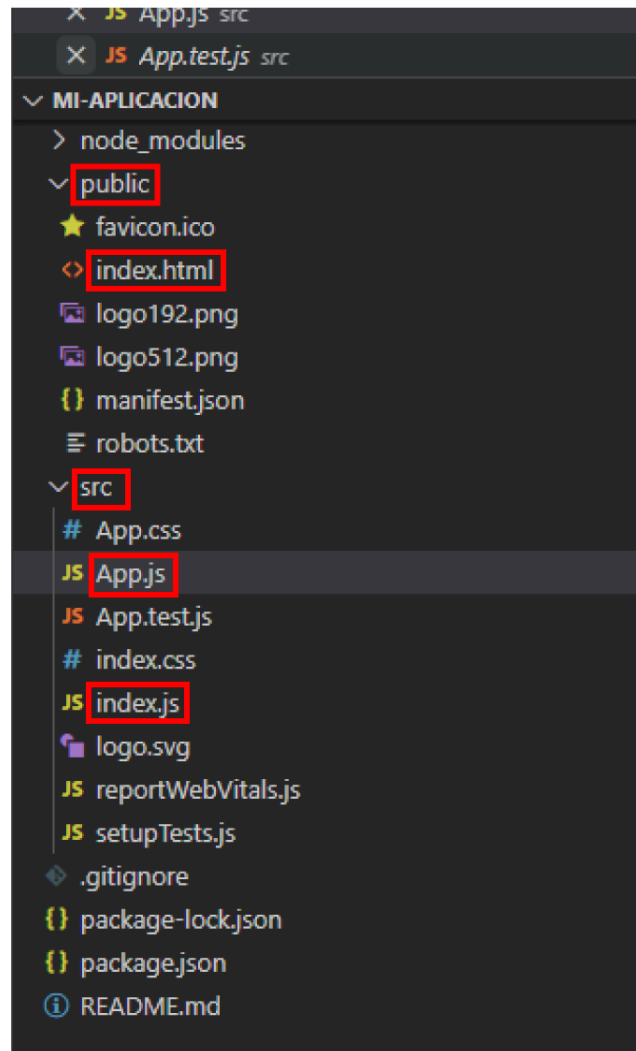


Ilustración 21 Estructura de un proyecto en React

Vamos a realizar un pequeño ejemplo donde se sumen dos números y para que repasemos algunas de las etiquetas que ya hemos visto hasta el momento en este módulo:

```
import logo from './logo.svg';
import './App.css';
function App() {
  return (
    <div>
      <form onSubmit={ejemplo}>
        <p>Ingrese el primer valor a sumar:
        <input type="number" name="valor1" />
        </p>
        <p>Ingrese segundo valor a sumar:
        <input type="number" name="valor2" />
        </p>
      </div>
    </div>
  );
}
```

```
<input type="submit" value="Realizar Suma" />
</p>
</form>
</div>
);
}
function ejemplo(e) {
e.preventDefault();
const v1=parseInt(e.target.valor1.value, 10);
const v2=parseInt(e.target.valor2.value, 10);
const suma=v1+v2;
alert('El valor de la suma es:'+suma);
}

export default App;
```

RESULTADO

4.4.1.2 COMPROBANTES BASADOS EN CLASES

Estos si deben ser conscientes de lo que está haciendo el usuario. Por ejemplo, si un usuario hace clic o escribe en la página, el componente debe poder responder de alguna manera dependiendo de lo que el usuario esté haciendo.

Este tipo de componentes tienen varios métodos de ciclo de vida que se activan en diferentes puntos durante la instanciación de renderizado, incluyendo constructor, render y componentDidMount. El único requisito de un componente basado en clases es el método render, ya que a través de este método se retorna el JSX, aunque el estado a menudo se establece dentro del método constructor. Finalmente, si desea obtener datos de algún lugar externo (por ejemplo, una API externa), podría hacerlo dentro del método componentDidMount, el cual se activa luego de renderizar.



4.4.1.3 JSX

Es una extensión de sintaxis para JavaScript. Se parece mucho a HTML, pero con algunas diferencias clave. Por ejemplo, puede incluir una expresión de JavaScript entre llaves:

```
<div className="App">
  <header className="App-header">
    <img src={logo} className="App-logo" alt="logo" />
    <h1 className="App-title">Hola mundo</h1>
  </header>
</div>
```

Adicionalmente, los elementos de HTML tienen un atributo llamado `class`, mientras que los elementos creados a través de JSX tienen un elemento llamado `className`. Ambos se refieren a lo mismo.

4.4.1.4 PROPS

Es un objeto de solo lectura que guarda los valores de los atributos de una etiqueta y trabaja de forma similar a los atributos de HTML. También es una forma de pasar datos de un componente a otro, de forma similar a como se envían argumentos a una función.

Veamos el siguiente ejemplo, donde se implementa el componente `Usuario` con dos props: nombre y apellido y serán renderizados mediante en un `ul`:

```
import React, { Component } from 'react';
export default class Usuario extends Component {
  render () {
    return (
      <ul>
        <li>Nombre: {this.props.nombre}</li>
        <li>Apellido: {this.props.apellido}</li>
      </ul>
    );
  }
}
```

4.4.2 TEMA 2 CONSUMO API USANDO UN FRAMEWORK

Divida el texto en temas, cada uno con un título apropiado y con estilo Título 3. Importante: Imágenes de buena calidad y su respectiva referencia bibliográfica. La API Fetch proporciona una interfaz JavaScript para acceder y manipular partes del canal HTTP, tales como peticiones y respuestas. También provee un método global `fetch()` (en-US) que proporciona una forma fácil y lógica de obtener recursos de forma asíncrona por la red.

Un api en React es el punto de entrada a la biblioteca de React. Si se carga React desde una etiqueta `<script>`, estas API de alto nivel estarán disponibles en el React global. Si se usa ES6 con npm se puede escribir `import React from 'react'`

Está soportado de forma nativa en navegadores como: Edge 14, Firefox 39, Chrome 42 y Opera 29. Para Internet Explorer 10 o Safari 6.1 o superiores o versiones anteriores de Firefox o Chrome. Para Node existe la librería `node-fetch`

<https://www.npmjs.com/package/node-fetch>

Los pasos para su instalación son muy sencillos abrimos una ventana de comandos y ejecutamos el siguiente código:

```
Node.js command prompt
C:\Users\Usuario\mi-aplicacion>cd..
C:\Users\Usuario>npm install node-fetch
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\Usuario\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\Usuario\package.json'
npm WARN Usuario No description
npm WARN Usuario No repository field.
npm WARN Usuario No README data
npm WARN Usuario No license field.

+ node-fetch@3.0.0
added 4 packages from 5 contributors and audited 4 packages in 2.36s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Ilustración 22 Instalación API Fetch

Flujo básico de fetch

Cuando se emplea el API de fetch se emplea el siguiente flujo:

- El método `fetch()` realiza una petición del recurso que necesita sobre el servidor que lo aloja.
- El propio objeto devuelve una promesa con el objeto `Response` de la petición, tanto si tiene éxito como si no.
- Una vez obtenida la respuesta, ésta proporciona una serie de métodos que permiten comprobar su contenido y manejarlo.

Su sintaxis básica es:

```
fetch( url )  
.then( response => response.json() )  
.then( data => console.info( data ) )  
  
.catch( e => console.error( 'Que paso viejo!!' ) );
```

Consumo de una API

```
/import { useState, useEffect } from 'react';  
import React from 'react'  
function App() {  
  
function miPromesa(){  
  setTimeout(function() {  
    let x = 9, y = 8  
    console.log("Se ejecuto a los 3 segundos")  
    console.log(`La suma de ${x} y ${y} es igual a ${x+y}`)  
  },1000);  
  
  let promesa = new Promise((resolve, reject) => {  
    setTimeout(() => {  
      let N_Compra = 11  
      if(N_Compra == 10)  
        resolve("La compra se puedo realizar exitosamente con "+N_Compra+" articulos.")  
      else  
        reject("No se pudo realizar la compra de los 10 art.")  
    },2000)  
  })  
  
  console.log("Se está realizando la compra")  
  promesa  
  .then(function(r){  
    console.log(r)  
  })  
  .catch(r => {  
    console.log("=====")  
    console.log(r)  
    console.log("=====")  
  })  
  .finally(() => console.log("Siempre se ejecuta"))  
}  
  
function consumirAPI(){
```

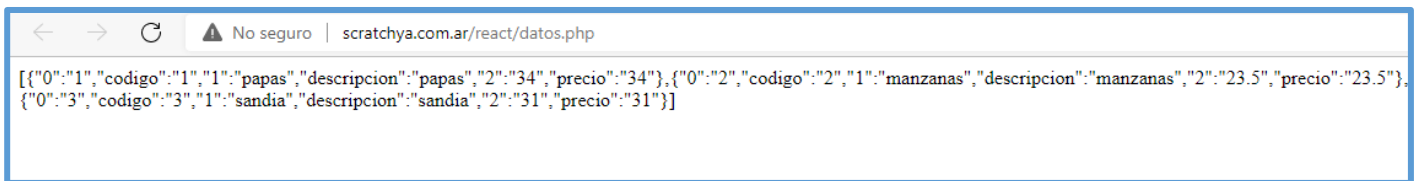
```
let respuesta = fetch('http://scratchya.com.ar/react/datos.php')

respuesta
  .then(r => {
    return r.json()
  })
  .then(data => console.log(data))
  .catch()
  .finally()
}

//miPromesa()
consumirAPI()

}
export default App
```

Al dar clic 'http://scratchya.com.ar/react/datos.php'



Se está consumiendo un archivo que se creó en PHP cuya función es guardar unos archivos llamado datos.php.

4.5 RECURSOS

4.5.1 Documentación Referenciada

Tema	Enlace URL	Especificación de páginas o instrucciones de lectura
Framework React	https://link.springer.com/content/pdf/bfm%253A978-1-4842-1245-5%252F1.pdf	Capítulo I Capitulo IV

4.5.2 Material de profundización

Tema	Enlace URL	Especificación de páginas o instrucciones de lectura
Framework React	https://www.theseus.fi/bitstream/handle/10024/261970/Thesis-Elar-Saks.pdf?sequence=2	Página 11-12, 24-27
Fetch	http://edshare.soton.ac.uk/20589/2/03c%20-%20Web%20APIs.pdf	Página 19-22

4.6 EJERCICIOS DE PROFUNDIZACIÓN

- Crear otro proyecto HTML básico, que cuente con una caja de texto, a esta caja de texto se le deben capturar todos los eventos relacionados con el teclado y determine en que evento se puede determinar si la palabra ingresada en la caja de texto es palíndroma, si lo es mostrar un alert con el mensaje “Es palíndromo”.
- Crear un proyecto REACT con el nombre suyo, teniendo en cuenta de usar Create React App.
 - Compile el proyecto y evidencie la creación de la carpeta build (Toma de pantalla).
 - Modifique la aplicación agregando una tabla, donde este los siguientes datos personales suyos con las siguientes columnas: Nombres, Apellidos, Correo y Celular.
 - Ejecute el proyecto y muestre la evidencia con una toma de pantalla.
- ¿Qué es Babel y WebPack y para que se usa en REACT?
- Implemente una función por el item (cada uno muestra la documentación de un servicio), donde se evidencie el uso del API de fetch (se extrae este servicio de la siguiente fuente <https://petstore.swagger.io/>).

NOMBRE	Crear Categoría		
OBJETIVO	El servicio recibe los datos de la categoría y la crea.		
MÉTODO	URL	Parámetros	Datos
POST	https://petstore.swagger.io/v2/cat	<pre>{"Cod": int, "category": { "id": int, "name": "string" }, "name": "string", "photoUrls": ["string"], "tags": [{ "id": int, "name": "string" }], "status": "string" }</pre>	<pre>{"id": int, "category": { "id": int, "name": "string" }, "name": "string", "photoUrls": ["string"], "tags": [{ "id": int, "name": "string" }], "status": "string" }</pre>

4.7 PISTAS DE APRENDIZAJE



Recuerde que:

- Framework

- API

- React

- Fetch

5 VIDEO LECCIÓN



https://remingtonedu-my.sharepoint.com/:v:/r/personal/falvarez_uniremington_edu_co/Documents/AAAModulosCompartidos/Ingenierias/Lenguaje%202/Presentacion%20por%20Unidad%20Lenguajes%20II%20Gloria%20Lora.mp4?csf=1&web=1&e=Q6OhGm

6 BIBLIOGRAFÍA

6.1 BASES DE DATOS INSTITUCIONALES

Atchison, L. (2021, October 25). Why you should use a microservice architecture. InfoWorld.com, NA. https://link.gale.com/apps/doc/A680133020/SPJ.SP01?u=cur_co&sid=bookmark-SPJ.SP01&xid=5b95f0e8

Appleton, B. (2015, August 11). Build a reusable REST API back end. InfoWorld.com. https://link.gale.com/apps/doc/A481609199/SPJ.SP01?u=cur_co&sid=bookmark-SPJ.SP01&xid=c36c3c6

Celaya Luna, A. (2014). Creación de páginas web: HTML 5. Editorial ICB. <https://elibro.net/es/lc/remington/titulos/56045>

Campderrich Falgueras, B. (2013). Ingeniería del software. Editorial UOC. <https://elibro.net/es/lc/remington/titulos/56294>

Garland, J., & Anthony, R. (2003, February). What is software architecture? (Database and Network Intelligence). Database and Network Journal, 33(1), 3+. https://link.gale.com/apps/doc/A98254069/SPJ.SP01?u=cur_co&sid=bookmark-SPJ.SP01&xid=89ebaa7b

Larsen, R. (2013). Beginning HTML and CSS. Wiley. <https://elibro.net/es/lc/remington/titulos/183103>

Orós Cabello, J. C. (2014). Diseño de páginas Web con XHTML, JavaScript y CSS. RA-MA Editorial. <https://elibro.net/es/lc/remington/titulos/106414>

Muhr, J. (Trad.) y Karlins, D. (2014). HTML5 & CSS3 For Dummies. Wiley. <https://elibro.net/es/lc/remington/titulos/187637>

Prieto Espinosa, A. (2005). Conceptos de informática. McGraw-Hill España. <https://elibro.net/es/lc/remington/titulos/50305>

Jones, K. (2012, August 23). How-to: REST Web services demystified; REST API integrations can be daunting if you haven't worked with Web services before. Here's how to get started. InfoWorld.com. https://link.gale.com/apps/doc/A300563886/SPJ.SP01?u=cur_co&sid=bookmark-SPJ.SP01&xid=a85b4e0b

Tyson, M. (2021, January 14). How to use React functional components. InfoWorld.com, NA. https://link.gale.com/apps/doc/A648335715/SPJ.SP01?u=cur_co&sid=bookmark-SPJ.SP01&xid=d4d1bec2

Tyson, M. (2021, February 11). Angular, React, Vue: JavaScript frameworks compared. InfoWorld.com. https://link.gale.com/apps/doc/A651462930/SPJ.SP01?u=cur_co&sid=bookmark-SPJ.SP01&xid=4e5bd418

Krill, P. (2018, August 9). React Native JavaScript framework stumbles. InfoWorld.com. https://link.gale.com/apps/doc/A549534081/SPJ.SP01?u=cur_co&sid=bookmark-SPJ.SP01&xid=c9e6040b

6.2 CIBERGRAFÍA

css. (1 de Noviembre de 2021). Obtenido de css: <https://egghead.io/q/css>

Daniel, L. (5 de Noviembre de 2021). Arquitectura de Software basada en Microservicios para servicios Web. Obtenido de <https://documentos.redclara.net/bitstream/10786/1277/1/93%20Arquitectura%20de%20Software%20basada%20en%20Microservicios%20para%20Desarrollo%20de%20Aplicaciones%20Web.pdf>

Emiliano, M. (7 de Noviembre de 2021). El Modelo Cliente/Servidor. Obtenido de El Modelo Cliente/Servidor: <https://www.linuxito.com/docs/el-modelo-cliente-servidor.pdf>

geeksforgeeks. (7 de Noviembre de 2021). Obtenido de geeksforgeeks: <https://www.geeksforgeeks.org/javascript-tutorial/?ref=ghm>

geeksforgeeks. (5 de Noviembre de 2021). Obtenido de geeksforgeeks: <https://www.geeksforgeeks.org/html-tutorials/?ref=ghm>

geeksforgeeks. (1 de Noviembre de 2021). Obtenido de geeksforgeeks: <https://www.geeksforgeeks.org/react-js-introduction-working/>

Javier, C. F. (2008). Java 2 Interfaces graficas y aplicaciones para la web. Madrid: AlfaOmega.

Learn the best JavaScript tools and frameworks from industry professionals. (1 de Noviembre de 2021). Obtenido de Learn the best JavaScript tools and frameworks from industry professionals: <https://egghead.io/lessons/javascript-test-javascript-with-jest>

React. (1 de Noviembre de 2021). Obtenido de React: <https://egghead.io/q/react>

Red Hat. (7 de Noviembre de 2021). Obtenido de <https://www.redhat.com/es/topics/microservices>

sololearn. (1 de Noviembre de 2021). Obtenido de sololearn: <https://www.sololearn.com/learning/1024>

sololearn. (1 de Noviembre de 2021). Obtenido de sololearn: <https://www.sololearn.com/learning/1014>



Tutorialesya. (1 de Noviembre de 2021). Obtenido de Tutorialesya:
<https://www.tutorialesprogramacionya.com/htmla/>

Tutorialesya. (7 de Noviembre de 2021). Obtenido de Tutorialesya:
<https://www.tutorialesprogramacionya.com/cssa/>

Tutorialesya. (1 de Noviembre de 2021). Obtenido de Tutorialesya:
<https://www.tutorialesprogramacionya.com/reactya/>

TutorialesYa. (1 de Noviembre de 2021). Obtenido de TutorialesYa:
<https://www.tutorialesprogramacionya.com/javascriptya/>

Universidad de Salamanca. (2 de Noviembre de 2021). Obtenido de
<http://vis.usal.es/rodrigo/documentos/soa/REST%20introduccion.pdf>

Universidad Tecnica Federico Santamaria. (7 de Noviembre de 2021). Obtenido de Universidad Tecnica
Federico Santamaria:
<http://profesores.elo.utfsm.cl/~agv/elo322/1s16/projects/reports/Proyecto%20Cliente%20-%20Servidor.pdf>

w3schools. (1 de Noviembre de 2021). w3schools. Obtenido de w3schools:
<https://www.w3schools.com/>