UTN FACULTAD REGIONAL CORDOBA
INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Desarrollo de Software
Notas de Clase

Apunte 03 - Desarrollo Web y HTML

¿Qué es el desarrollo Web?

Desarrollo Web significa construir y mantener sitios web.
Al comienzo los sitios se componían de un conjunto de **páginas web** muy sencillas. Sin embargo, en la actualidad, y con el avance de la tecnología, en una web se puede hacer cualquier cosa. Es por eso que el desarrollo web se ha popularizado tanto que se ha convertido en un área bastante demandada.

Los términos sitio web y aplicación web suenan parecido y con frecuencia se utilizan sin distinción, hablando de ellos como si fueran lo mismo. Sin embargo, se trata de plataformas digitales con propósitos específicos que responden a necesidades muy diferentes. Cada una ofrece un conjunto distinto de funcionalidades a los usuarios, las que dependen directamente de los objetivos de un proyecto.

En pocas palabras, un **sitio web** es un conjunto de páginas estáticas que entregan información. Por su parte, las **aplicaciones web** son plataformas principalmente interactivas que se centran en que los usuarios realicen acciones y accedan a contenido dinámico recuperado generamente de diferentes fuentes de información. Una aplicación web puede ser parte de un sitio en un proyecto, pero no al revés.

Las webs que solo tienen frontend se les conoce como webs estáticas. Se les denomina así precisamente porque siempre muestran la misma información para todos los usuarios.
Ejemplo de eso son las páginas informativas o personales, puede ser perfectamente estáticas porque todos los usuarios que entren en la web la van a ver igual, es la misma información.

Pero si se piensa en una web como una red social, no puede ser estática porque necesita guardar usuarios en una base de datos y cada uno de estos usuarios ven cosas distintas en su timeline.

Características de las aplicaciones Web

- **Utilizan distintos lenguajes de programación:** A nivel técnico las aplicaciones web dinámicas son mucho más complejas que las estáticas y suelen utilizar los siguientes lenguajes de programación: React, Angular, PHP, JavaScript, Asp.NET entro otros, pues son los que permiten estructurar de mejor forma el contenido.
- **Se gestionan en un CMS:** A nivel técnico las aplicaciones web dinámicas son mucho más complejas que las estáticas y suelen utilizar los siguientes lenguajes de programación: React, Angular, JavaScript, PHP, Asp.NET, pues son los que permiten estructurar de mejor forma el contenido.

Ventajas

- Se puede ingresar a las aplicaciones web desde cualquier dispositivo tecnológico con conexión a internet y se pueden ejecutar en todos los sistemas operativos.
- Son mucho más fácil de actualizar que una aplicación estática o una página web, ya que toman los cambios de la información desde la base de datos (webservices del backend) y con código javascript actualiza dinámicamente la visualización del contenido html.
- No es necesario instalar aplicaciones, se ejecutan directamente desde el navegador, descargando los archivos HTML, CSS y Javascript necesarios.
- Se cargan mucho más rápido que una página web, ya que generalmente el código html y javascript quedan almacenados en el caché del navegador y sólo se realizan llamadas ajax en formato json para el intercambio de información con el servidor.
- Se desarrollan en muy poco tiempo utilizando librerías o frameworks modernos como React, Angular, Vue, pero requiere el dominio de html, javascript y eventualmente dependiendo de la tecnologia a utilizar de typescript.
- Permiten implementar distintas funcionalidades como acceso a bases de datos a través de webservices y foros, lo cual es muy usado para facilitar procesos de reglase de negocios de la organización en la intranet.
- Puede ser gestionada por un editor de contenido, no es necesario que la administre un webmaster.
- No exige el uso de servidor para el desarrollo, pero la publicación web en producción se realizará en un servidor web tipo IIS, Apache o NGINX.
- Facilita a los usuarios la localización del contenido que desean visitar y la manipulación del mismo es mucho más amigable.
- Es una excelente herramienta para recopilar los datos de los usuarios que han navegado en la app web, como tomar pedidos, cargar información de la lógica de negocios, y demas información para la gestión de una organización.

HTML (*)

Para construir un sitio Web es necesario diseñar un conjunto páginas WEB. Dichas páginas no son otra cosa que documentos de texto plano estructurados mediante un Lenguaje de Marcas de Hipertexto o **HTML**.

Hipertexto hace referencia a los enlaces que conectan páginas web entre sí, ya sea dentro de un único sitio web o entre sitios web. Los enlaces son un aspecto fundamental de la Web.

HTML es un estándar a cargo del World Wide Web Consortium (W3C), organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. En su última versión **HTML5** varias de sus anteriores etiquetas (HTML4) se han simplificado, por lo que es mucho más fácil y rápido escribir código.

HTML5 es mucho más dinámico e incluye elementos multimedia. Soporta de forma nativa el vídeo y el audio, e incluso es posible crear juegos o animaciones con él.

El estándar HTML ha evolucionado a lo largo del tiempo a través de varias versiones. Estas versiones incluyen:

- HTML 1.0: La primera versión lanzada en 1991, que estableció las bases para la web moderna con elementos básicos como texto y enlaces.
- HTML 2.0: Introducida en 1995, amplió las capacidades de HTML 1.0 al agregar soporte para formularios y tablas.
- HTML 3.2: Publicada en 1997, esta versión trajo consigo mejoras significativas en la presentación con la introducción de estilos y soporte para tablas más avanzadas.
- HTML 4.01: Lanzada en 1999, proporcionó una mayor claridad y especificación en la estructura del documento, además de incluir soporte para marcos y hojas de estilo en cascada (CSS).
- HTML5: Esta versión, que comenzó a desarrollarse en 2004 y se finalizó en 2014, marcó un cambio importante al introducir una serie de nuevas características, como elementos semánticos (por ejemplo, `<header>`, `<footer>`, `<nav>`), soporte nativo para multimedia, gráficos vectoriales (SVG) y capacidades de almacenamiento local. HTML5 también hizo énfasis en la accesibilidad y la compatibilidad con dispositivos móviles.

Si se piensa en la construcción de una casa, los cimientos, las vigas (columnas) y las paredes conforman la estructura del proyecto. De igual manera HTML define la estructura y el significado del contenido web.

Para editar una página HTML y posteriormente visualizarla, todo lo que se necesita es un editor de texto y un navegador Web. Para ver una página HTML no es necesario una conexión a Internet, cualquier explorador Web permite hacerlo de manera local.

A diferencia de los lenguajes de programación convencionales, HTML utiliza una serie de etiquetas (o marcas) intercaladas llamadas **elementos HTML**. Dichos elementos serán posteriormente interpretadas por los exploradores encargados de visualizar la página o el documento Web con el fin de establecer el formato.

Las partes principales del elemento son:

- La etiqueta de apertura: consiste en el nombre del elemento (en este caso, p), encerrado por paréntesis angulares (< >) de apertura y cierre. Establece dónde comienza o empieza a tener efecto el elemento —en este caso, dónde es el comienzo del párrafo—.
- La etiqueta de cierre: es igual que la etiqueta de apertura, excepto que incluye una barra de cierre (/) antes del nombre de la etiqueta. Establece dónde termina el elemento —en este caso dónde termina el párrafo—.
- El contenido: este es el contenido del elemento, que en este caso es sólo texto.

- El elemento: la etiqueta de apertura, más la etiqueta de cierre, más el contenido equivale al elemento.

Todo elemento HTML puede tener **atributos** en forma de clave=valor, tal como se muestra en la siguiente imagen:

Los atributos contienen información adicional acerca del elemento, la cual no quieres que aparezca en el contenido real del elemento. Aquí **class** es el nombre del atributo y **editor-note** el valor del atributo. En este caso, el atributo class permite darle al elemento un nombre identificativo, que se puede utilizar luego para apuntarle al elemento información de estilo y demás cosas.

Un atributo debe tener siempre:

- Un espacio entre este y el nombre del elemento (o del atributo previo, si el elemento ya posee uno o más atributos).
- El nombre del atributo, seguido por un signo de igual (=).
- Comillas de apertura y de cierre, encerrando el valor del atributo.

Los atributos siempre se incluyen en la etiqueta de apertura de un elemento, nunca en la de cierre.

Estructura básica de un documento HTML (*)

Tal como se muestra en la siguiente imagen, todo documento HTML se compone de un conjunto de elementos individuales que son combinados para formar una página HTML entera.

- `<html></html>` . Este elemento encierra todo el contenido de la página entera y, a veces, se le conoce como el elemento raíz (root element).
- `<head></head>` . Este elemento actúa como un contenedor de todo aquello que quieres incluir en la página HTML que no es contenido visible por los visitantes de la página. Incluye palabras clave (keywords), una descripción de la página que quieres que aparezca en resultados de búsquedas, código CSS para dar estilo al contenido, declaraciones del juego de caracteres, etc. Dentro del `<head>` comunmente se incluyen:
  - `<meta>` Se utiliza para especificar metadatos, como el conjunto de caracteres utilizado, la descripción de la página para los motores de búsqueda y otras etiquetas útiles para la optimización de motores de búsqueda (SEO) y accesibilidad. Por ejemplo si se quiere indicar el conjunto de caracteres que se mostrarán en el página se utiliza:
    `<meta charset="utf-8">` . Este elemento establece el juego de caracteres que tu documento usará en utf-8, que incluye casi todos los caracteres de todos los idiomas humanos. Básicamente, puede manejar cualquier contenido de texto que puedas incluir.
  - `<title></title` .Establece el título de tu página, que es el título que aparece en la pestaña o en la barra de título del navegador cuando la página es cargada, y se usa para describir la página cuando es añadida a los marcadores o como favorita.
  - `<link>` : Se utiliza para vincular la página con hojas de estilo externas (CSS), fuentes tipográficas, iconos y otros recursos externos.
  - `<script>` : Permite incluir scripts JavaScript en la página, que pueden proporcionar funcionalidades dinámicas y mejoras de interactividad. También se puede colocar en otras partes del documento HTML, como dentro del elemento `<body>` . La decisión de dónde ubicar el elemento `<script>` depende de su función y del momento en que se necesita cargar.
- `<body></body>` . Encierra todo el contenido que deseas mostrar a los usuarios web que visiten tu página, ya sea texto, imágenes, videos, juegos, pistas de audio reproducibles, y demás. Dentro de `<body>` , se pueden incluir varios elementos HTML, como encabezados (`<h1>` , `<h2>` , etc.), párrafos (`<p>`), listas (`<ul>` , `<ol>`), enlaces (`<a>`), imágenes (`<img>`), formularios (`<form>`), y muchos más. Además del contenido estático, también es común incluir scripts JavaScript y otros elementos dinámicos dentro de `<body>` para agregar interactividad y funcionalidad a la página.
  Es importante tener en cuenta que `<body>` es uno de los elementos más importantes en la estructura de un documento HTML, ya que define la experiencia de usuario y el contenido visible de la página.

> Al comienzo del documento, antes del elemento raíz, suele indicarse el elemento `<!DOCTYPE html>` . Este elemento no es una etiqueta pero sirve para identificar la versión de HTML en la que está escrita el documento, así de sencillo. `<!DOCTYPE html>` indica que el documento es HTML 5.

El propósito principal es asegurarse de que el navegador interprete correctamente el código HTML y aplique las reglas de renderizado correctas. Al proporcionar esta declaración al principio del documento, se indica al navegador que debe interpretar el contenido como HTML5, lo que ayuda a garantizar una visualización coherente y precisa en diferentes navegadores y dispositivos.

Etiquetas basicas HTML (*)

Las etiquetas indican a los exploradores Web cómo tienen que mostrar el texto y los gráficos. La siguiente tabla muestra las etiquetas básicas como encabezados, enlaces, imágenes, listas y formularios.

Etiqueta	Significado	Detalle
`<h1><h2><h3><h4><h5><h6>`	Cabeceras	Los documentos HTML, al igual que los documentos Word pueden tener cabeceras para indica subtítulos. El tamaño del texto es mayor cuanto mayor es el nivel (siendo 1 el nivel más alto).
`<hr>`	Línea horizontal	Permite dibujar un separador con forma de línea horizontal
`<p>`	Párrafos	Se utiliza para escribir un párrafo de texto. Todo párrafo va predecida automáticamente por una blanco. Junto con el texto es posible incluir etiquetas en línea para resaltar partes específicas c `<span><strong><small><del>` .
` `	Retorno de carro	Los retornos de carro en el texto son ignorados; por lo tanto, para introducir uno se utiliza esta
`<a>`	Enlaces	Permite definir un enlace. Esta etiqueta delimita el texto que se quiere utilizar para enlazar con o parte de la misma. Por ejemplo: `<a href='#identificador'>` Sección dentro del documento referenciar una sección dentro del documento, o: `<a href='/path de la pagina'>` Otra página `</a>` para referenciar a otra página dentro del sitio, o bien: `<a href='URL externa+protocolo' target='blank' rel="noopener noreferrer" >Otra página en` para enlazar a otro documento fuera del sitio.
`<img>`	Imágenes	Mediante el atributo **src** permite especificar el nombre del fichero que contiene la imagen. "<im del recurso' alt='Texto alternatico'>". El atributo **alt** permite indicar un texto alternativo en caso imagen no pueda ser cargada exitosamente.
`<ol>`	Listas ordenadas	`<ol><li>item</li></ol>` . El elemento interno `<li>` (list item) permite indicar cada elemento de enumerada. Salida: 1. item
`<ul>`	Listas desordenas	`<ul><li>item</li></ul>` . Idem anterior con viñetas
`<form>`	Formularios	Son uno de los principales puntos de interacción entre un usuario y un sitio web. El atributo má es **action**='recurso' que permite definir la acción por defecto del formulario cuando se envían d servidor. Este atributo va acompañado de **method**='GET/POST' para indicar el tipo de método para generar la petición HTTP.
`<input>`	Campos de entrada	Existe una amplia variedad de controles de entrada de datos. Para crearlo se utiliza la etiqueta los atributos **type** y **name**. Para ver todas las posibilidades acceder a: MDN Reference - Input
`<span>, <strong>, <small>, <del>, <i>`	Etiquetas en línea	Estas etiquetas se utilizan para afectar elementos de texto específicos dentro de un párrafo u c contenedor de texto sin afectar a otros elementos circundantes. Las etiquetas en línea son útile aplicar estilos y formatos específicos a partes del texto, como cambiar el color, la fuente, el tam estilo del texto, agregar efectos de texto como cursiva o subrayado, o incluso insertar elemento multimedia, como imágenes o videos, dentro del texto.

> Para ver el detalle completo de etiquetas de HTML acceder al siguiente pdf: Etiquetas HTML

Secciones en HTML5

Las **secciones en HTML5** son elementos que se utilizan para estructurar y organizar el contenido de una página web de manera semántica y significativa. Estos elementos proporcionan una forma clara de dividir el contenido en partes lógicas, lo que hace que el código HTML sea más legible y comprensible tanto para los desarrolladores como para los motores de búsqueda.

HTML5 incorpora las secciones para mejorar la claridad y la semántica del código HTML, lo que facilita la comprensión del propósito y la relación de diferentes partes del contenido. Además, el uso adecuado de las secciones también mejora la accesibilidad de la página web para usuarios con tecnologías de asistencia, como lectores de pantalla, al proporcionar una estructura lógica y bien definida.

Aquí hay algunas razones por las que HTML5 incorpora las secciones:

1. Claridad estructural: Las secciones como `<header>`, `<footer>`, `<main>`, `<article>`, `<nav>`, entre otros, proporcionan nombres descriptivos que indican claramente la función y el propósito de cada parte del contenido.
2. Facilita el mantenimiento: La estructura clara y organizada de las secciones facilita la comprensión y el mantenimiento del código HTML, lo que hace que sea más fácil realizar cambios y actualizaciones en la página web.
3. Mejora la accesibilidad: Al utilizar elementos semánticos como `<header>`, `<footer>`, etc., se mejora la accesibilidad de la página web para usuarios con discapacidades y tecnologías de asistencia al proporcionar una estructura clara y significativa.
4. SEO: Los motores de búsqueda pueden interpretar mejor el contenido de una página web cuando está estructurado de manera semántica utilizando elementos de sección, lo que puede mejorar el posicionamiento en los resultados de búsqueda.

En resumen, HTML5 incorpora las secciones para mejorar la estructura, la claridad y la semántica del código HTML, lo que facilita la comprensión, el mantenimiento y la accesibilidad de las páginas web.

Sección	Significado
`<header>`	Define el encabezado de una sección o del documento entero.
`<nav>`	Define una sección de navegación.
`<main>`	Define el contenido principal de un documento.
`<article>`	Define un contenido independiente y autocontenido, como un artículo o una publicación de blog.
`<section>`	Define una sección genérica en un documento.
`<aside>`	Define contenido secundario, como una barra lateral, que no está directamente relacionado con el contenido principal.
`<footer>`	Define el pie de una sección o del documento entero.

El siguiente código HTML ejemplifica el uso de las secciones HTML utilizadas con mayor frecuencia:

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo de Secciones HTML5</title>
</head>
<body>

    <header>
        <h1>Encabezado de mi Página</h1>
        <nav>
            <ul>
                <li><a href="#">Inicio</a></li>
                <li><a href="#">Acerca de</a></li>
                <li><a href="#">Contacto</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <section>
            <h2>Artículo Principal</h2>
            <article>
                <h3>Título del Artículo</h3>
                <p>Contenido del artículo...</p>
            </article>
            <article>
                <h3>Otro Artículo</h3>
                <p>Contenido del otro artículo...</p>
            </article>
        </section>
    </main>

    <aside>
        <h2>Barra lateral</h2>
        <p>Contenido de la barra lateral...</p>
    </aside>

    <footer>
        <p>Pie de página &copy; 2024</p>
    </footer>

</body>
</html>
```

Este ejemplo contiene un encabezado (`<header>`), una barra de navegación dentro del encabezado (`<nav>`), el contenido principal de la página (`<main>`) que incluye un par de artículos (`<article>`), una barra lateral (`<aside>`) y un pie de página (`<footer>`). Estas etiquetas estructuran el contenido de la página de manera semántica y significativa, lo que facilita su comprensión tanto para los desarrolladores como para los motores de búsqueda.

Atributos comunes de etiquetas HTML

Algunas propiedades comunes de elementos HTML:

- **id**: Permite asignar un identificador único a un elemento, que luego puede ser utilizado para referenciarlo en CSS o JavaScript.
- **class**: Se utiliza para aplicar estilos a un grupo de elementos relacionados mediante CSS. Puede asignarse a múltiples elementos y se puede utilizar para aplicar estilos o realizar acciones en JavaScript.
- **src**: Especifica la ruta de origen (URL) para recursos como imágenes, scripts, archivos de audio o video, que se utilizarán en la página.
- **href**: Se utiliza en elementos de enlace `<a>` para especificar la URL de destino a la que se enlaza el elemento.
- **alt**: Proporciona un texto alternativo que se muestra si el elemento no se puede cargar, como en el caso de una imagen que falte.
- **title**: Proporciona un texto que se muestra como información sobre herramientas cuando el usuario pasa el cursor sobre el elemento.
- **type**: Define el tipo de contenido para elementos como `<script>` o `<input>` , por ejemplo, type="text/javascript" para scripts o type="text/css" para hojas de estilo.
- **value**: Utilizado en elementos de formulario como `<input>` para especificar el valor inicial del control.

Estas son solo algunas de las propiedades más comunes que se encuentran en los elementos HTML. La variedad y uso de las propiedades varían dependiendo del tipo de elemento y su función específica.

Atributos HTML 5

La especificación HTML5 trajo consigo muchas más características nuevas y mejoras que permiten la creación de experiencias web más ricas y dinámicas. Las más comunes se enumeran a continuación:

- **placeholder**: Se utiliza en elementos de formulario para proporcionar un texto de ejemplo dentro del campo de entrada, que desaparece cuando se comienza a escribir.
- **required**: Especifica que un campo de entrada en un formulario es obligatorio de completar antes de enviarlo.
- **autofocus**: Hace que un campo de entrada reciba automáticamente el enfoque cuando se carga la página, lo que permite al usuario comenzar a escribir inmediatamente.
- **autocomplete**: Controla si un campo de entrada de formulario debe tener activada la función de autocompletado del navegador.
- **download**: Utilizado en enlaces `<a>` para indicar que el destino del enlace es para descargar un archivo en lugar de navegar a él.
- **media**: Utilizado en elementos `<source>` dentro de un elemento multimedia `<video>` o `<audio>` para especificar la URL de origen de los archivos multimedia alternativos.
- **draggable**: Permite a los usuarios arrastrar y soltar elementos HTML utilizando eventos de arrastrar y soltar en JavaScript.
- **contenteditable**: Permite que el contenido de un elemento sea editable por el usuario, convirtiendo un elemento HTML en un área de texto editable.
- **spellcheck**: Controla si el navegador debe verificar la ortografía del texto en un elemento editable.
- **placeholder**: Este atributo se utiliza en los elementos de formulario `<input>` y `<textarea>` para proporcionar un texto de marcador de posición que se muestra cuando el campo está vacío.
- **for**: Se utiliza en las etiquetas para asociarlas con elementos de formulario específicos mediante el uso del atributo "id". Esto ayuda a mejorar la accesibilidad y la usabilidad del formulario al hacer clic en la etiqueta para activar el campo asociado.

Formularios HTML (*)

Un formulario HTML es una sección de una página web que permite al usuario ingresar y enviar datos a un servidor web. Los formularios HTML están compuestos por una serie de elementos, como campos de entrada de texto, botones de opción, casillas de verificación, menús desplegables, etc.

Cuando un usuario completa y envía un formulario HTML, los datos ingresados en el formulario se envían al servidor web que está detrás del sitio web. El servidor web luego procesa los datos y realiza alguna acción basada en la información que recibió. Por ejemplo, si el formulario es un formulario de contacto, el servidor podría enviar un correo electrónico al propietario del sitio web con los detalles de la consulta del usuario.

Los formularios HTML son una parte importante del diseño web y se utilizan comúnmente en una amplia variedad de aplicaciones, como encuestas, registros de usuarios, compras en línea, entre otros. Los desarrolladores web utilizan una combinación de lenguaje HTML, CSS y JavaScript para crear formularios interactivos y atractivos para los usuarios.

A continuación, se enumeran algunas de las propiedades principales de la etiqueta HTML `<form>` :

- **action**: es la URL a la que se enviarán los datos del formulario cuando se envíe. Es importante que se especifique una URL válida para que los datos del formulario se puedan procesar correctamente.
- **method**: especifica el método HTTP que se utilizará para enviar los datos del formulario. Los dos métodos más comunes son GET y POST. GET se utiliza para obtener información de la URL, mientras que POST se utiliza para enviar información a un servidor.
- **enctype**: especifica cómo se codificarán los datos del formulario antes de enviarlos. El valor predeterminado es application/x-www-form-urlencoded, pero también se pueden usar otros valores, como multipart/form-data para enviar archivos.
- **target**: especifica el destino donde se abrirá la respuesta del servidor después de enviar el formulario. Si se omite, el valor predeterminado es _self, lo que significa que la respuesta se mostrará en la misma ventana o pestaña que el formulario.
- **name**: especifica un nombre para el formulario que se puede utilizar para referirse al formulario en JavaScript y CSS.
- **autocomplete**: especifica si el navegador debe completar automáticamente los campos del formulario. Los valores posibles son on y off.
- **novalidate**: especifica que el formulario no debe ser validado cuando se envíe. Esto se usa comúnmente en pruebas y en formularios que no requieren validación.
- **class y id**: especifican clases y un identificador únicos para el formulario, que se pueden utilizar para aplicar estilos y scripts personalizados al formulario.

A continuación, se presentan los elementos clave de los formularios en HTML:

1. **Elemento `<form>`** : Define el inicio y el final de un formulario. Todos los elementos del formulario deben estar contenidos dentro de esta etiqueta. Puede especificar el método de envío (GET o POST) y la URL de destino para procesar los datos del formulario.
2. **Elementos de entrada (`<input>`)**: Son utilizados para recopilar información del usuario. Pueden ser de varios tipos, como texto, contraseña, casillas de verificación, botones de radio, botones de envío, botones de restablecimiento, etc.
3. **Elemento `<select>` y `<option>`** : Permite al usuario seleccionar una opción de una lista desplegable. El elemento `<select>` define la lista, mientras que `<option>` define cada opción individual dentro de esa lista.
4. **Elemento `<textarea>`** : Proporciona un área de texto de múltiples líneas donde los usuarios pueden ingresar texto largo, como comentarios o mensajes.
5. **Etiqueta `<label>`** : Se utiliza para asociar texto descriptivo con elementos de entrada del formulario. Mejora la accesibilidad y la usabilidad, ya que permite hacer clic en el texto asociado para activar el elemento de entrada correspondiente.

6. **Validación de Formularios**: HTML5 introduce atributos de validación para verificar los datos del usuario antes de enviarlos al servidor. Algunos atributos comunes incluyen `required`, `pattern`, `min`, `max`, `maxlength`, `minlength`, etc.
7. **Envío de Formularios**: Cuando un usuario envía un formulario, los datos ingresados se envían al servidor para su procesamiento. El método de envío puede ser GET o POST, y se especifica en el atributo `method` del elemento `<form>`.

En resumen, los formularios en HTML nos permite recopilar datos de los usuarios en la web, con este tipo de elementos de HTML junto con React en las unidades posteriores vamos a procesar los datos que ingrese el usuario.

En el siguiente ejemplo de "Afiliación a un club deportivo" se detalla el código HTML para solicitar la información:

```
    <form id="formularioAfiliacion" action="#" method="post">
        <!-- Nombre -->
        <label for="nombre">Nombre:</label>
        <input type="text" id="nombre" name="nombre" required><br><br>

        <!-- Contraseña -->
        <label for="contrasena">Contraseña:</label>
        <input type="password" id="contrasena" name="contrasena" minlength="8" required><br><br>

        <!-- Edad -->
        <label for="edad">Edad:</label>
        <input type="number" id="edad" name="edad" min="18" max="100" required><br><br>

        <!-- Fecha de Nacimiento -->
        <label for="fecha_nacimiento">Fecha de Nacimiento:</label>
        <input type="date" id="fecha_nacimiento" name="fecha_nacimiento" required><br><br>

        <!-- Género -->
        <label for="genero">Género:</label>
        <select id="genero" name="genero" required>
            <option value="">Seleccione</option>
            <option value="masculino">Masculino</option>
            <option value="femenino">Femenino</option>
            <option value="otro">Otro</option>
        </select><br><br>

        <!-- Deportes -->
        <label>Deportes:</label><br>
        <input type="checkbox" id="futbol" name="deporte" value="futbol">
        <label for="futbol">Fútbol</label><br>
        <input type="checkbox" id="basket" name="deporte" value="basket">
        <label for="basket">basket</label><br>
        <input type="checkbox" id="tenis" name="deporte" value="tenis">
        <label for="tenis">Tenis</label><br><br>

        <!-- Nivel de Experiencia -->
        <label for="experiencia">Nivel de Experiencia:</label><br>
        <input type="radio" id="novato" name="experiencia" value="principiante" checked>
        <label for="principiante">Principiante</label><br>
        <input type="radio" id="intermedio" name="experiencia" value="intermedio">
        <label for="intermedio">Intermedio</label><br>
        <input type="radio" id="experto" name="experiencia" value="experto">
        <label for="experto">Experto</label><br><br>

        <!-- Comentario -->
        <label for="comentario">Comentario:</label><br>
        <textarea id="comentario" name="comentario" rows="4" cols="50" required></textarea><br><br>

        <!-- Botón de Enviar -->
        <button type="submit">Enviar</button>

        <!-- Botón de Mostrar Datos -->
        <button type="button" onclick="mostrarDatos()">Mostrar Datos</button>
    </form>
<script>
function mostrarDatos() {
    var nombre = document.getElementById("nombre").value;
    var contrasena = document.getElementById("contrasena").value;
    var edad = document.getElementById("edad").value;
    var fechaNacimiento = document.getElementById("fecha_nacimiento").value;
    var genero = document.getElementById("genero").value;
    var deportes = document.querySelectorAll('input[name="deporte"]:checked');
    var experiencia = document.querySelector('input[name="experiencia"]:checked').value;
    var comentario = document.getElementById("comentario").value;

    var mensaje = "Nombre: " + nombre + "\n";
    mensaje += "Contraseña: " + contrasena + "\n";
    mensaje += "Edad: " + edad + "\n";
    mensaje += "Fecha de Nacimiento: " + fechaNacimiento + "\n";
    mensaje += "Género: " + genero + "\n";
    mensaje += "Deportes: ";
```

```
    deportes.forEach(function(deporte) {
        mensaje += deporte.value + ", ";
    });
    mensaje = mensaje.slice(0, -2); // Eliminar la última coma y espacio
    mensaje += "\nExperiencia: " + experiencia + "\n";
    mensaje += "Comentario: " + comentario;

    alert(mensaje);
}
</script>
```

El ejemplo funcionando está aquí: https://stackblitz.com/edit/dds-formulario-html?file=index.html

Elementos utilizados en el ejemplo anterior para el ingreso de datos

- `<form id="formularioAfiliacion" action="#" method="post">` :
  - `<form>` es un elemento HTML que define un formulario para recopilar datos del usuario.
  - `id="formularioAfiliacion"` asigna un identificador único al formulario, lo que permite referirse a él desde otros elementos o scripts.
  - `action="#"` especifica la URL a la que se enviarán los datos del formulario cuando se envíe. En este caso, `#` indica que los datos se enviarán a la misma página, lo que es común cuando se manejan formularios de forma dinámica o mediante scripts.
  - `method="post"` indica que los datos del formulario se enviarán mediante el método HTTP POST cuando se envíe el formulario. Esto significa que los datos del formulario se incluirán en el cuerpo de la solicitud HTTP cuando se envíe.
- **Nombre (`<input type="text">`)**:
  - Permite al usuario ingresar su nombre.
  - Requerido mediante el atributo `required` .
- **Contraseña (`<input type="password">`)**:
  - Permite al usuario ingresar una contraseña.
  - Se especifica una longitud mínima de 8 caracteres mediante el atributo `minlength="8"` .
  - Requerido mediante el atributo `required` .
- **Edad (`<input type="number">`)**:
  - Permite al usuario ingresar su edad.
  - Se especifica un rango mínimo y máximo de 18 a 100 años mediante los atributos `min="18"` y `max="100"` .
  - Requerido mediante el atributo `required` .
- **Fecha de Nacimiento (`<input type="date">`)**:
  - Permite al usuario seleccionar su fecha de nacimiento.
  - Requerido mediante el atributo `required` .
- **Género (`<select>`)**:
  - Permite al usuario seleccionar su género de una lista desplegable.
  - Requerido mediante el atributo `required` .
- **Deportes (`<input type="checkbox">`)**:
  - Permite al usuario seleccionar uno o varios deportes de una lista.
  - Cada deporte tiene su propia casilla de verificación.
- **Nivel de Experiencia (`<input type="radio">`)**:
  - Permite al usuario seleccionar su nivel de experiencia.
  - Cada opción de nivel tiene su propio botón de selección.
  - La agrupación de conjunto de elementos se puede realizar estableciendo el atributo name con el mismo valor para cada grupo que se desee crear.
- **Comentario (`<textarea>`)**:
  - Permite al usuario ingresar un comentario.
  - Es un campo de texto multilínea.
  - Requerido mediante el atributo `required` .

El código Javascript (que veremos con más detalle en las unidades siguientes) dentro del elemento **script** realiza lo siguiente:

- **function mostrarDatos() { ... }**: Se define una función llamada mostrarDatos que se ejecuta cuando se hace clic en el botón "Mostrar Datos".
- **document.getElementById("nombre").value**: Obtiene el valor del campo de texto de nombre.
- **document.getElementById("contrasena").value**: Obtiene el valor del campo de contraseña.
- **document.getElementById("edad").value**: Obtiene el valor del campo de edad.
- **document.getElementById("fecha_nacimiento").value**: Obtiene el valor del campo de fecha de nacimiento.
- **document.getElementById("genero").value**: Obtiene el valor seleccionado en el menú desplegable de género.
- **document.querySelectorAll('input[name="deporte"]:checked')**: Obtiene todos input que contienen el atributo **name="deporte"**, y extraer los deportes seleccionados mediante casillas de verificación (sólo los que están checkeados **:checked**).
- **document.querySelector('input[name="experiencia"]:checked').value**: Obtiene el valor del nivel de experiencia seleccionado mediante botones de radio.
- **document.getElementById("comentario").value**: Obtiene el valor del área de texto de comentario.

- **Construcción del mensaje**: Se concatena cada valor obtenido en un mensaje de texto, formateado de manera legible para su visualización.

  `deportes.forEach(function(deporte) { ... });` :
  - Este bloque de código itera sobre cada elemento seleccionado de la lista de deportes.
  - `deportes` es una colección de elementos que representan las casillas de verificación de los deportes seleccionados por el usuario.
  - `forEach()` es un método que ejecuta una función proporcionada una vez por cada elemento en la colección `deportes` .
  - La función proporcionada recibe un parámetro `deporte` , que representa cada elemento de la colección durante la iteración.
  - Dentro de la función, se agrega el valor (`value`) del deporte seleccionado al mensaje, seguido de una coma y un espacio para separar los valores.
- **alert(mensaje)**: Finalmente, se muestra el mensaje de datos recopilados en un cuadro de alerta para que el usuario lo vea.

Ejemplo de aplicación

Se propone como actividad paso a paso diseñar un página con un formulario de contacto que permita ingresar los datos: nombre completo, correo electrónico, teléfono, un mensaje y una opción desde una lista de opciones.
Los datos del formulario serán enviados a: https://labsys.frc.utn.edu.ar/dds-express/eco quien recibe los datos del formulario y devuelve una tabla HTML con los campos/valores enviados.

> Nota: Todos los campos serán obligatorios y los tipos de entrada deben ser los más adecuados según el valor del campo del formulario.

Ver código del ejemplo aquí (Se sugiere descargar el archivo index.html,y probar local en su equipo instalando la extensión **Live Server** de Visual Studio Code)

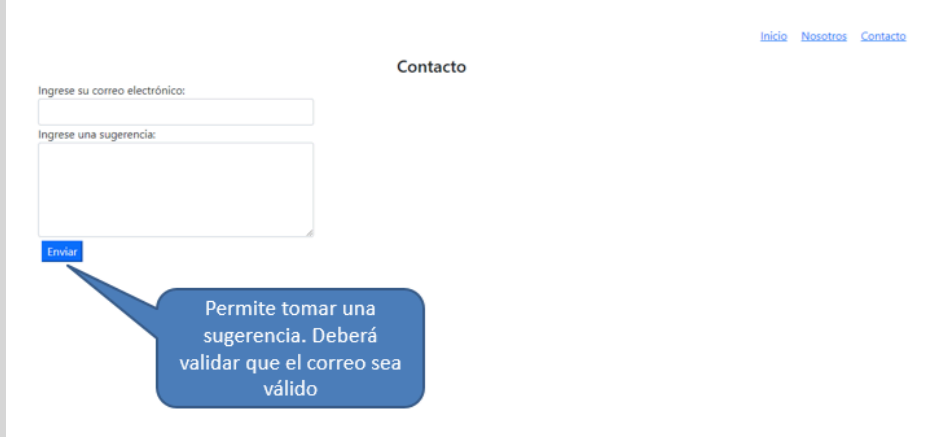
Ejemplo de aplicación 2

Desarrollar un primer sitio Web tal como se indica en las siguientes imágenes:

> Página de inicio

> Página Nosotros

> Página Contacto



Ver código del ejemplo aquí

Bibliografía o Referencia

- Aprende sobre desarrollo web - Conceptos básicos de HTML
- Ceballos Fco. J. (2006). *Interfaces gráficas y aplicaciones para Internet* (2° Edición). Editorial RA-MA.