DESARROLLO DE SOFTWARE

UNIDAD N° 4

Contenidos

- Fundamentos del Desarrollo Web
- Herramientas de Desarrollo del Navegador (DevTools)
- HTML (HyperText Markup Language)
- CSS (Cascading Style Sheets)
- Diseño Web Adaptativo (Responsive Design)
- JavaScript
- JavaScript: Manipulación del DOM (Document Object Model)
- JavaScript: Manejo de Controles y Arrays
- JavaScript: Manejo de Errores y Eventos
- Tablas Dinámicas y Paginación
- Formularios y Validación
- Fetch API
- Almacenamiento del Lado del Cliente

Recursos de lectura

- https://developer.mozilla.org/es/
- https://es.react.dev/
- https://www.reactjs.wiki/
- https://vite.dev/guide/
- https://react-hook-form.com/get-started
- https://tailwindcss.com/docs/installation/using-vite

Material de Estudio

Primera Parte

Obligatorio:

- ¿Cómo funciona internet?
- ¿Qué es un nombre de dominio?
- ¿Qué es una URL?
- ¿Qué es un servidor WEB?
- HTML
- CSS

Complementario:

- DevTools
- CSS - Cascada y Herencia

Fundamentos del Desarrollo Web

Fundamentos del Desarrollo Web

Frontend (Cliente)	Backend (Servidor)	Conocimientos generales	Herramientas y flujos de trabajo modernos
Lenguajes esenciales: • **HTML** (Lenguaje de etiquetas de hipertexto) • **CSS** (Hojas de estilo en cascada) • **JavaScript** **Tecnologías y herramientas comunes:** • **Frameworks y bibliotecas:** React, Vue.js, Angular. • **Preprocesadores CSS:** SASS, LESS. • **Gestores de paquetes:** npm, yarn. • **Bundlers y transpiladores:** Webpack, Babel, Vite.	**Lenguajes y entornos comunes:** • **JavaScript** (Node.js), **Python** (Django/Flask), **PHP**, **Java** (Spring), **C#** (.NET), **Ruby** (Rails). **Componentes clave:** • **Servidor web:** Nginx, Apache. • **Frameworks de backend** • **Base de datos** **Conceptos importantes:** • **Rutas (Routing):** define cómo responder a solicitudes HTTP. • **APIs (REST, GraphQL):** permiten que frontend y backend se comuniquen. • **Autenticación/autorización:** control de acceso con JWT, OAuth, etc.	**Protocolos y arquitectura:** • **HTTP/HTTPS** • **DNS, IP, URL, dominios.** • **Modelo cliente-servidor.** • **Ciclo de vida de una petición web.** **Buenas prácticas:** • **Seguridad** • **Performance** • **Accesibilidad (a11y)** • **Responsive design** • **Control de versiones** **Testing y mantenimiento:** • **Testing:** unitario, integración, e2e. • **CI/CD:** integración y despliegue continuo (ej: GitHub Actions, Jenkins). • **Documentación del código y APIs.**	**Editores de código:** VS Code, WebStorm, Visual Studio. **DevTools del navegador.** **Contenedores y entornos virtuales:** Docker, Vagrant. **Despliegue en la nube:** Vercel, Netlify, Heroku, AWS, etc.

Fundamentos del Desarrollo Web

- **IP:** Dirección numérica única que identifica a cada dispositivo a una red.
- **Dominio:** Nombre fácil de recordar que se usa para acceder a una dirección IP.
- **DNS:** Sistema que traduce un Dominio a su dirección IP real.
- **URL:** Una dirección completa que se escribe en el navegador

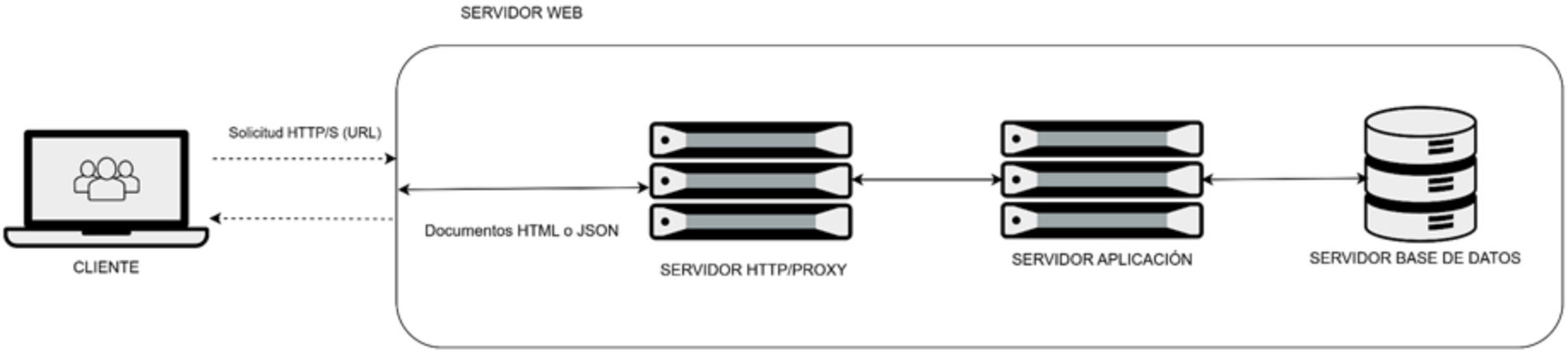
Ejemplo:

- URL: https://www.google.com/search?q=chatgpt
  - Protocolo: https://
  - Dominio: www.google.com
  - Ruta del recurso: /search
  - Parámetros: ?q=chatgpt

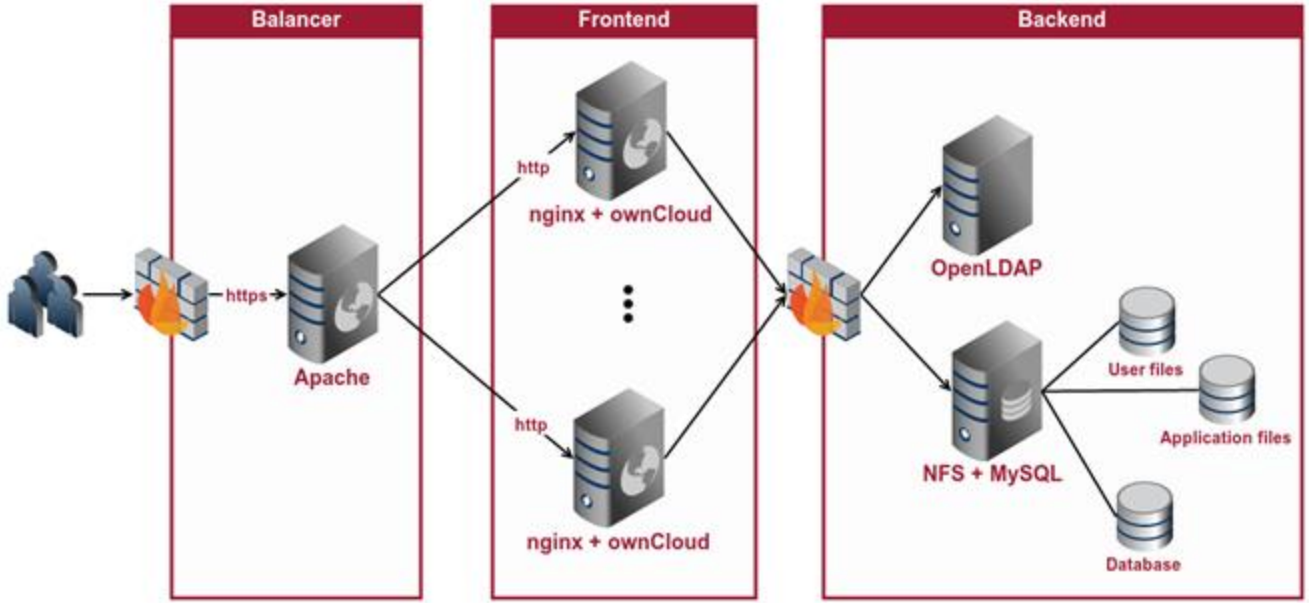
- DNS convierte www.google.com a la IP 142.250.217.46.
- Tu navegador usa esa IP para conectarse al servidor de Google.

Fundamentos del Desarrollo Web (Visión simplificada)

Fundamentos del Desarrollo Web (En nuestra local)



Fundamentos del Desarrollo Web (Caso de uso real)



https://caminstech.upc.edu/es/blog/caminsdrive

Frontend

Parte visible de un sitio o aplicación web, es decir, todo lo que el usuario ve e interactúa directamente desde el navegador.
Contiene:

- HTML > Estructura
- CSS > Estilos visuales
- Javascript > Lógica

Frontend - Qué contiene?

- Menús, formularios, botones, imágenes, etc.
- Animaciones, validaciones, navegación entre páginas.
- Responsive.
- Accesibilidad para personas con discapacidad.

Frontend - Tipos basado en renderizado

1 - Renderizado en el servidor (Server-Side Rendering - SSR):

El servidor genera un HTML completo en cada solicitud y lo envía al navegador. Contenido cambia con frecuencia y necesito buen SEO.

Ejemplo: Blogs, ecommerce, portales de noticias.

Pros:
- Buena performance ideal.
- SEO optimizado.

Contras:
- Carga más lenta entre páginas.
- Requiere más recursos del servidor.

Frameworks: Next.js (modo SSR), Laravel, RoR, ASP.NET MVC

Frontend - Tipos basado en renderizado

2 - Renderizado en el client (Client-Side Rendering - CSR):

Se entrega al navegador un HTML vacío, el cual es completado con Javascript. El navegador construye la interfaz dinámicamente.

Ejemplo: Apps tipo dashboard, panel de usuario, SPA.

Pros:
- Navegación fluida.
- Separación clara entre frontend y backend.

Contras:
- Peor SEO (inicialmente el contenido no está en el HTML).
- Primera carga más lenta.

Frameworks: React (con Vite o CRA), Vue, Angular

Frontend - Tipos basado en renderizado

3 - Aplicación de una sola página (Single Page Application - SPA):

Es un tipo particular de CSR. Toda la app corre en una sola página index.html, y la navegación no recarga la página.

Ejemplo: Gmail, Trello, Google Drive.

Pros:
- UX fluida.
- Ideal para apps con muchas interacciones.

Contras:
- Manejo más complejo del estado y las rutas.
- SEO limitado (uso de técnicas de prerendering).

Frontend - Tipos basado en renderizado

4 - Aplicación web progresiva (Progressive Web App - PWA):

No es un método de renderizado en sí, es un enfoque que permite:
- funcionar offline
- se vea como una app móvil
- se instala en dispositivo

Ejemplo: Spotify web app.

Pros:
- Carga rápida.
- Experiencia tipo app.
- Funciona offline

Contras:
- Compleja de configurar.

Requiere: Service workers.

Frontend - Tipos basado en renderizado

5 - MPA (Multi-Page Application):

Ejemplo: Sitios tradicionales en php, WordPress, ASP.

6 - Static Site Generation (SSG):

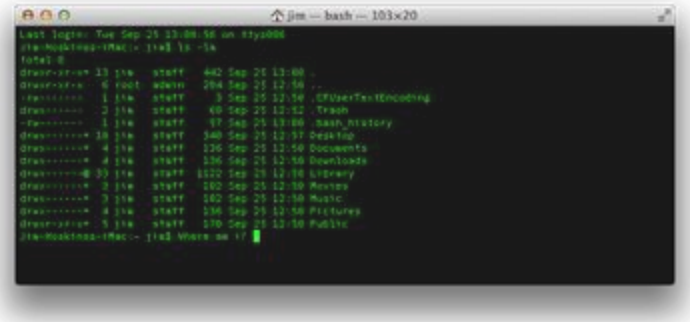
Ejemplo: Blogs, landing pages, portfolios.

Herramientas: Astro, entre otros

Herramientas a utilizar

Visual Studio Code

https://code.visualstudio.com/



Línea de comandos

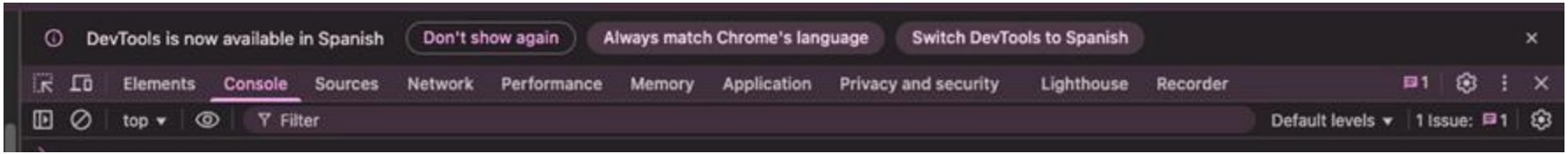
git

Git

DevTools de navegadores

Herramientas integradas en todos los navegadores.
Para acceder, dentro del navegador:

- F12
- Click derecho -> Inspeccionar
- Configuración -> Más Herramientas -> Herramientas del Desarrollador (Chrome)



DevTools de navegadores

Pestaña	Función principal	¿Qué permite hacer?
Elements	Inspección de HTML y CSS	Ver, editar estructura y estilos en vivo
Console	Visualización de mensajes y errores de JavaScript	Ejecutar comandos JS, ver logs y advertencias
Sources	Depuración de código JavaScript	Colocar breakpoints, revisar archivos, debugging paso a paso
Network	Análisis de solicitudes y recursos de red	Ver peticiones HTTP, tiempos de carga, cabeceras

DevTools de navegadores

Performance	Análisis del rendimiento de la página	Medir FPS, repaints, scripting, tiempos de renderizado
Application	Gestión de almacenamiento local y recursos	Revisar cookies, localStorage, sessionStorage, cachés
Security	Revisión de certificados y políticas de seguridad	Verificar HTTPS, problemas de contenido mixto
Lighthouse	Auditoría de calidad (solo en Chrome)	Evaluar performance, accesibilidad, SEO, PWA
Memory	Análisis del uso de memoria	Detectar fugas de memoria, visualizar heap

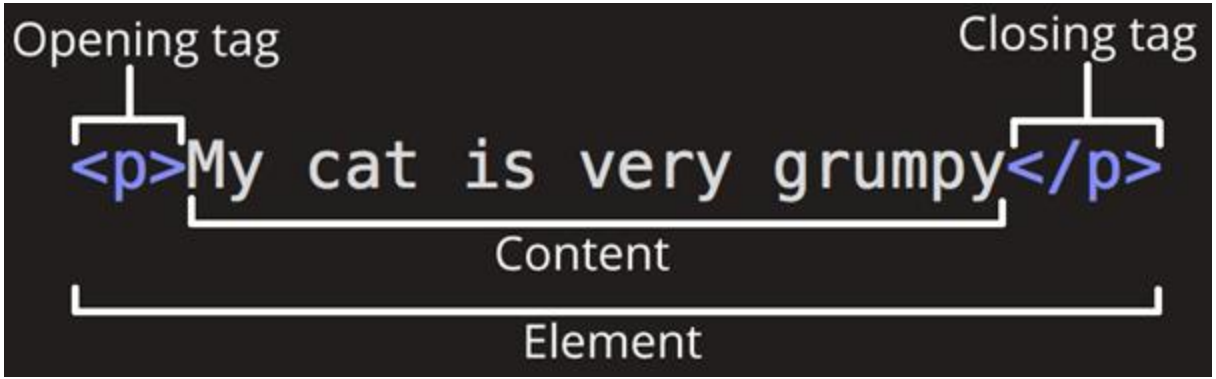
DevTools de navegadores (Extensiones)

- React Developer Tools
- React Context Devtool

HTML (HyperText Markup Language)

- Lenguaje para estructurar páginas web. Donde se define una estructura de un documento usando **etiquetas** (<h1>, <p>, <img>, etc) y es interpretado por navegadores para mostrar contenido visual al usuario.
- Es el **esqueleto de la página**: HTML con CSS (estilos) y JS (comportamiento) forman la base del desarrollo web.

Anatomía de un elemento



- **Etiqueta de apertura**: nombre del elemento (ej. *p* para el párrafo), envuelto en corchetes. Esta etiqueta de apertura indica donde inicia el elemento.
- **Contenido**: es el contenido del elemento. En el ej, es el texto del párrafo.
- **Etiqueta de cierre**: es la misma etiqueta de apertura, pero incluye barra diagonal del nombre de la etiqueta. Esto marca donde termina. No incluir puede generar una mala visualización, aunque algunos navegadores logran interpretar donde puede o no terminar un elemento, completando la estructura automáticamente (**NO SE DEBE CONFIAR EN ESTO NUNCA, Y ASEGURARSE DE CERRAR EL ELEMENTO).**

Anidamiento

```HTML
<p>Mi gato es <strong>muy</strong> gruñón.</p>
```

Mi gato es **muy** gruñón.

```HTML
<p>Mi gato es <strong>muy gruñón.</p></strong>
```

Mi gato es **muy gruñón.**

- Casos especiales

```HTML
<img
  src="https://raw.githubusercontent.com/mdn/beginner-html-site/gh-pages/images/firefox-icon.png"
  alt="Icono de Firefox" />
```

Atributos

Un atributo debe tener:

- Espacio entre el atributo y el elemento.
- Nombre de atributo seguido de un signo igual.
- Un valor de atributo, envuelto con comillas de apertura y cierre. Hay casos particulares.

Anatomía de un documento HTML

```html
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Título de la página</title>
  </head>
  <body>
    <h1>Encabezado principal</h1>
    <p>Párrafo de ejemplo en una página HTML.</p>
  </body>
</html>
```

- **<!DOCTYPE html>:** indica que es HTML5.
- **<html>:** elemento raíz del documento.
- **<head>:** metadatos, título, enlaces a estilos o scripts.
- **<body>:** contenido visible (texto, imágenes, enlaces, etc.)

Etiquetas comunes

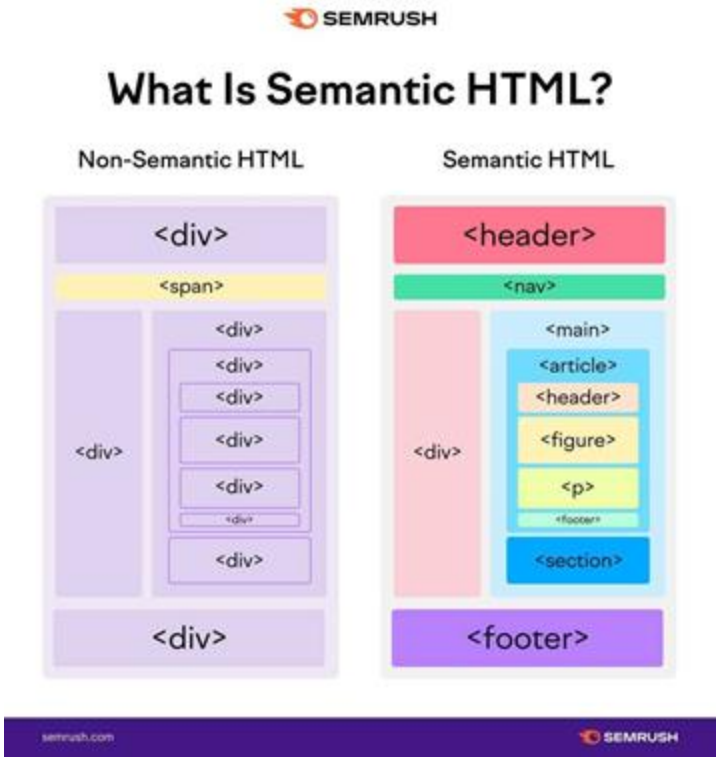
- **Encabezados:** <h1>–<h6>
- **Párrafos:** <p>
- **Enlaces:** Texto
- **Imágenes:**
- **Listas:**
    a. Ordenadas ...
    b. Desordenadas ...
- **Divisiones genéricas:** , (sin semántica) ().

Etiquetas semánticas

Son etiquetas que transmiten el **significado del contenido**, mejorando accesibilidad y SEO .

- **<header>:** cabecera del sitio o sección
- **<nav>:** bloque de navegación
- **<main>:** contenido principal de la página
- **<section>:** sección genérica de contenido
- **<article>:** contenido independiente (artículos, entradas)
- **<aside>:** contenido secundario (sidebar)
- **<footer>:** pie de página o sección
- **<figure>** y **<figcaption>:** imágenes con título/descripción ()
- Otras: **<mark>**, **<time>**, **<progress>**, **<details>**, **<summary>**

Etiquetas semánticas



Ejercicio #16

- Crear una carpeta con el nombre “**Ejercicio_16**”
- Abrir la Visual Studio y cargar la carpeta “**Ejercicio_16**”
- Crear un archivo “**index.html**”
- Agregar los siguientes elementos, en el orden dado:
    - Elemento “**header**”, contenido “**Soy una página**”
    - Elemento “**nav**”, dentro incluir lo siguiente:
        - Elemento “**a**”, contenido “**Inicio**”
        - Elemento “**a**”, contenido “**Productos**”
    - Elemento “**main**”, dentro incluir lo siguiente:
        - Elemento “**div**”, contenido “**Hola a todos!**”
    - Elemento “**footer**”, contenido “**Información de mi empresa**”

- Abrir la carpeta “**Ejercicio_16**” y hacer doble click en “**index.html**”

Resultado: Debe abrirse un navegador y se debe mostrar el contenido de “**index.html**”

Ejercicio #16.2

Tenemos que relacionar dos páginas.

- Crear un segundo archivo en Visual Studio, llamado “**products.html**”.
- Copiar el contenido desde “**index.html**”
- Cambiar el contenido del elemento “**div**” y poner “**Soy un listado de Productos**”
- En los archivos “**index.html**” y “**products.html**” deben modificar las etiquetas “**a**”, agregando el atributo “**href**”
- Para las etiqueta que es “**Inicio**”, el atributo “**href**” debe apuntar al archivo “**index.html**”
- Para las etiqueta que es “**Productos**”, el atributo “**href**” debe apuntar al archivo “**products.html**”

Resultado: Desde el navegador, si haces click en los enlaces, tienes que navegar desde una página a otra.

Forms

Un formulario HTML permite al usuario ingresar y enviar datos a un servidor o procesarlo en el cliente.

```
<form action="/procesar" method="POST">
  <!-- elementos del formulario -->
</form>
```

Elementos de un formulario

`<input type="text">`	Campo de texto de una sola línea	`Nombre:` <input type="text">
`<input type="password">`	Campo para contraseñas (oculta el texto)	`Contraseña:` <input type="password">
`<input type="email">`	Valida que el contenido tenga formato de email	<input type="email">
`<input type="number">`	Permite solo números	<input type="number">
`<input type="checkbox">`	Casilla de verificación	<input type="checkbox"> Acepto términos
`<input type="radio">`	Opción única dentro de un grupo	<input type="radio" name="color"> Rojo

Elementos de un formulario

`<input type="submit">`	Botón para enviar el formulario	<input type="submit" value="Enviar">
`<input type="reset">`	Botón para limpiar los campos	<input type="reset" value="Borrar">
`<input type="date">`	Selector de fecha	<input type="date">
`<input type="file">`	Selector de archivos para subir	<input type="file">
`<textarea>`	Área de texto multilínea	`<textarea rows="4" cols="40"></textarea>`
`<select>` + `<option>`	Menú desplegable	`<select><option>Argentina</option></select>`
`<label>`	Etiqueta para asociar texto con un control	`<label for="email">Correo</label>`

Ejemplo

```
<form action="/registrar" method="POST">
  <label for="nombre">Nombre:</label>
  <input type="text" id="nombre" name="nombre" required><br><br>

  <label for="email">Correo:</label>
  <input type="email" id="email" name="email" required><br><br>

  <label for="pais">País:</label>
  <select id="pais" name="pais">
    <option value="ar">Argentina</option>
    <option value="br">Brasil</option>
    <option value="uy">Uruguay</option>
  </select><br><br>

  <label><input type="checkbox" name="terminos" required> Acepto los términos</label><br><br>

  <input type="submit" value="Enviar">
</form>
```

Ejercicio #17

- Crear una carpeta con el nombre “**Ejercicio_17**”, abrir Visual Studio y cargar la carpeta creada
- Crear un archivo “**index.html**”
- Agregar formulario para realizar registro de usuario:
    - Elemento “**form**”, dentro incluir lo siguiente:
        - Elemento “**label**”, contenido “**Usuario:**”
        - Elemento “**input**” tipo texto, requerido.
        - Elemento “**label**”, contenido “**Email:**”
        - Elemento “**input**” tipo email, requerido.
        - Elemento “**label**”, contenido “**Contraseña:**”
        - Elemento “**input**”, tipo password, requerido.
        - Elemento “**input**” tipo submit

Resultado: Debe visualizar el formulario de registro con las validaciones en cada campo.

CSS (Cascading Style Sheets)

Hojas de Estilo en Cascada o **CSS** es un lenguaje de estilos que describe cómo debe renderizar un elemento en la pantalla.

```
1  h1 {
2    color: red;
3    font-size: 5em;
4  }
```

Aplicar CSS

Forma	Cómo se usa	Ventajas	Desventajas
Inline	En el mismo elemento, usando el atributo “style”: Hola	Rápido para pruebas	No reutilizable, difícil de mantener
Interno (style)	Dentro de una etiqueta “style” en el “head”: <head><style> h1 { color: red; } </style></head>	Útil para archivos simples o pruebas	No separa contenido de estilo
Externo	En un archivo “.css” referenciado: <link rel="stylesheet" href="estilos.css">	Reutilizable, limpio y organizado	Requiere carga adicional

Aplicar CSS

Tipo de Selector	Ejemplo	Qué selecciona
Universal	* { }	Todos los elementos
Elemento	p { }	Todos los <p>
Clase	.rojo { }	Elementos con class="rojo"

Aplicar CSS

Tipo de Selector	Ejemplo	Qué selecciona
ID	#principal { }	Elemento con id="principal"
Agrupado	h1, h2, p { }	Todos los h1, h2, y p
Descendiente	article p { }	Todos los p dentro de article

Aplicar CSS

Tipo de Selector	Ejemplo	Qué selecciona
Hijo directo	ul > li { }	li hijos directos de ul
Hermano adyacente	h1 + p { }	Primer p que sigue a un h1
Atributo	input[type="text"]	Inputs de tipo texto

Aplicar CSS

Tipo de Selector	Ejemplo	Qué selecciona
Pseudo-clase	a:hover	Estado al pasar el mouse
Pseudo-elemento	p::first-line	La primera línea de cada párrafo

Ejercicio #18

- Crear una carpeta con el nombre “**Ejercicio_18**” y abrir la carpeta en Visual Studio
- Copiar los archivos “**index.html**” y “**products.html**” del ejercicio 16.
- Elemento “**header**”:
  - style=”border: 1px solid gray;”
- Elemento “**nav**”:
  - style=”border: 1px solid red; height: 30px;”
- Elemento “**main**”:
  - style=”border: 1px solid blue; height: 100px;”
- Elemento “**footer**”:
  - style=”border: 1px solid green; height: 30px;”
- Aplicar los mismos en archivo externo

Resultado: Al abrir el navegador y cargar “**index.html**” debe mostrarse los bordes en cada sección.