«Talento Tech»

Front-End JS

Clase 01

Buenos Aires Ciudad

Clase N° 1 Conceptos Básicos de HTML

Temario:

1. **Conceptos básicos sobre Full Stack**
    - Explicación general de Full Stack y las áreas que abarca.
2. **Diferencias entre Front-End y Back-End**
    - Comparación entre las responsabilidades y tecnologías de Front-End y Back-End.
3. **Herramientas y tecnologías utilizadas en cada área**
    - Tecnologías clave en Front-End (HTML, CSS, JS, etc.) y Back-End (Bases de Datos, Servidores, APIs, etc.).
4. **Instalación y configuración de Visual Studio Code**
    - Introducción a Visual Studio Code como herramienta principal de desarrollo.
    - Paso a paso para la instalación y configuración básica de Visual Studio Code.
5. **Introducción a HTML**
    - ¿Qué es HTML y cuál es su función en el desarrollo web?
    - Diferencias entre una página web y un sitio web.
    - Importancia del estándar HTML5.
6. **Etiquetas Semánticas en HTML**
    - ¿Qué son las etiquetas semánticas?
    - ¿Para qué sirven?
    - Ejemplos y uso de etiquetas semánticas (`<header>`, `<nav>`, `<main>`, `<footer>`, `<section>`, `<article>`, `<aside>`).
    - Encabezados (`<h1>`, `<h2>`, `<h3>`, …)
    - Párrafos (`<p>`)

Objetivo de la clase:

Hoy vamos a dar nuestros primeros pasos en el mundo del desarrollo web. Vamos a descubrir qué significa ser Full Stack y conocer las dos grandes partes que lo componen: el Front-End, que es todo lo que vemos y usamos en una página, y el Back-End, que es lo que pasa detrás para que todo funcione. También vamos a instalar una herramienta clave: Visual Studio Code, que va a ser nuestra “caja de herramientas” para programar. Después nos introduciremos con HTML, el lenguaje que le da estructura a las páginas web, aprendiendo para qué sirve y cómo se usa. Y para cerrar, vamos a practicar con etiquetas básicas y semánticas que nos van a permitir crear nuestras primeras estructuras web de forma ordenada y con sentido. ¡A programar se ha dicho!

Conceptos Básicos sobre Full Stack

¿Qué es Full Stack?

El término "Full Stack" se refiere a la capacidad de un desarrollador para trabajar en todas las capas de una aplicación web, desde la interfaz de usuario (Front-End) hasta la lógica del servidor y la base de datos (Back-End).

Un desarrollador o desarrolladora Full Stack tiene conocimientos en varias tecnologías y herramientas que le permiten construir aplicaciones completas, abarcando desde la experiencia del usuario hasta la gestión de datos y la lógica del servidor. Comprende el flujo completo de desarrollo de una aplicación web, lo que le permite tener una visión global del proyecto y la capacidad de solucionar problemas que puedan surgir en cualquier parte del stack.

Diferencias entre Front-End y Back-End

Front-End (Lo que ves):

Es como la “cara” de un sitio web o una app, todo lo que se muestra en la pantalla.. El desarrollo Front-End involucra la creación de interfaces de usuario utilizando tecnologías como HTML, CSS y JavaScript.

Tecnologías principales:

- **HTML**: Es el esqueleto de la página. Si una web fuera una casa, el HTML serían las paredes, el techo, las puertas… la estructura básica.
- **CSS**: Se encarga del look. Siguiendo el ejemplo de la casa, CSS sería la pintura de las paredes, la decoración, los muebles, las cortinas... lo que hace que se vea linda y ordenada.
- **JavaScript**: Le da vida a la casa. Es como cuando tocás un botón y se prende la luz, o abrís una ventana automáticamente. En la web, eso sería hacer clic y que se abra un menú, deslizar imágenes, completar formularios, etc.

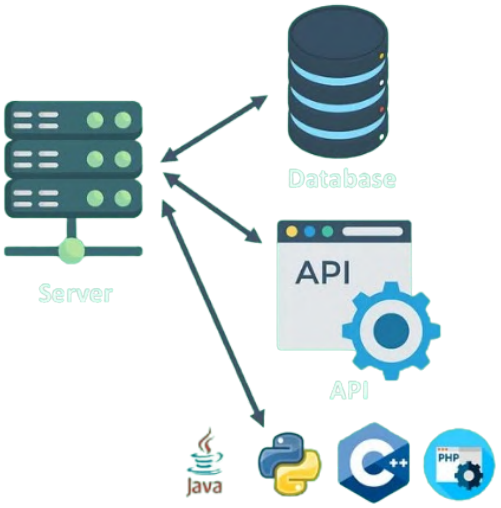
Ejemplo: Cuando entrás a una tienda online y ves los productos, los botones para agregar al carrito, o el menú que se despliega… todo eso es Front-End.

Back-End (Lo que no ves):

El **Back-End** es la parte que trabaja detrás de escena. Es el “cerebro” que procesa todo lo que pasa cuando hacés clic en algún lugar. Aunque no lo ves, es fundamental para que todo funcione bien. Se encarga de procesar solicitudes, ejecutar operaciones en la base de datos y devolver respuestas al Front-End.

Elementos clave:

- **Servidor Web**: Es como un mozo en un restaurante. Recibe lo que pedís (como ver tus compras anteriores) y va a buscar esa información a la cocina (la base de datos).
- **Base de Datos**: Es el lugar donde se guarda toda la info, como usuarios/as, productos, pedidos, etc. Sería como la despensa del restaurante: ahí está todo lo necesario para preparar lo que pediste.
- **Lenguajes de Programación**: El Back-End se programa usando lenguajes como **Python**, **Java**, **PHP** o **Node.js**. Son las herramientas que permiten que el mozo (servidor) y la cocina (base de datos) se entiendan.



Ejemplo: Cuando hacés clic en “Comprar”, el Front-End le avisa al Back-End. Este se fija si hay stock, guarda tu pedido en la base de datos, y luego le devuelve al Front-End un mensaje que diga: “¡Compra realizada con éxito!”.

¿Cómo trabajan juntos?

Podemos imaginar una tienda online como un restaurante:

- **Front-End**: Es el salón donde te sentás, el menú que ves, y el mozo que te atiende. Todo lo que ves y tocás.
- **Back-End**: Es la cocina, donde se preparan los platos. Vos no la ves, pero sin ella, no hay comida.

Ambos trabajan en equipo para que tengas una buena experiencia.

Herramientas y Tecnologías en Front-End:

- **HTML, CSS, y JavaScript:** las tecnologías fundamentales para la creación de la interfaz de usuario.
- **Frameworks y Librerías:** herramientas como React, Angular o Vue.js ayudan a construir aplicaciones más complejas y dinámicas.
- **Preprocesadores de CSS:** Sass o Less permiten escribir CSS de manera más eficiente y organizada.
- **Control de Versiones:** Git es esencial para el control de versiones, permitiendo a los desarrolladores realizar un seguimiento de los cambios en su código.

Herramientas y Tecnologías en Back-End:

- **Lenguajes de Programación:** Python, Java, PHP, Ruby, Node.js.
- **Bases de Datos:** MySQL, PostgreSQL, MongoDB, SQL Server.
- **Frameworks:** Django, Ruby on Rails, Express.js.
- **Servidores:** Apache, Nginx, o servidores en la nube como AWS y Azure.
- **APIs y RESTful Services:** Creación de APIs para interactuar con el Front-End y otras aplicaciones.

Instalación y Configuración de Visual Studio Code

Paso 1: Descarga e Instalación

- Visitá el sitio oficial de Visual Studio Code: https://code.visualstudio.com/
- Descargá la versión correspondiente a tu sistema operativo (Windows, macOS, Linux).
- Seguí las instrucciones de instalación en pantalla. El proceso es sencillo y no requiere configuración especial.

Paso 2: Configuración Básica

- **Abrir Visual Studio Code:** Una vez instalado, abrí el editor.
- **Configurar Preferencias:**
  - Ve a `File > Preferences > Settings`.
  - Desde aquí, podés personalizar el tema, fuentes, y otros aspectos visuales.

Paso 3: Idioma

- **Abrir Visual Studio Code:** Una vez instalado, abrí el editor.
- Hacé clic en el ícono de **Extensiones** (o apretá `Ctrl + Shift + X`).
- En la barra de búsqueda escribí:
  `Spanish Language Pack for Visual Studio Code`
- Aparecerá una extensión oficial publicada por **Microsoft** con este nombre.
  Hacé clic en **"Instalar"**.
- Una vez instalada, VS Code te va a sugerir reiniciar para aplicar el idioma.
  Hacé clic en **"Restart"** o cerrá y abrí de nuevo VS Code.
- ¡Listo! Tu VS Code ahora está en español.

Paso 4: Live Server

- **Abrir Visual Studio Code:** Una vez instalado, abrí el editor.
- Hacé clic en el ícono de **Extensiones** (o apretá `Ctrl + Shift + X`).
- En la barra de búsqueda escribí:
  `Live Server`
- Buscá la extensión llamada **Live Server** creada por **Ritwick Dey** (es la más popular).
- Hacé clic en **"Instalar"**.

- Una vez instalada, vas a ver un botón abajo a la derecha que dice **"Go Live"** cuando estés editando un archivo HTML.
- Hacé clic en **"Go Live"** y se abrirá tu página web en el navegador.
  Cada vez que guardes (`Ctrl + S`), el navegador se actualiza solo.

🛑Live Server **puede mostrar errores cuando no encuentra bien las rutas** a tus archivos (por ejemplo, imágenes o CSS).
Esto puede pasar si no estás trabajando con una estructura de carpetas organizada o si escribiste mal el camino.
Acordate que las rutas siempre se escriben desde el archivo HTML hacia donde está el archivo que querés vincular.

Introducción a HTML

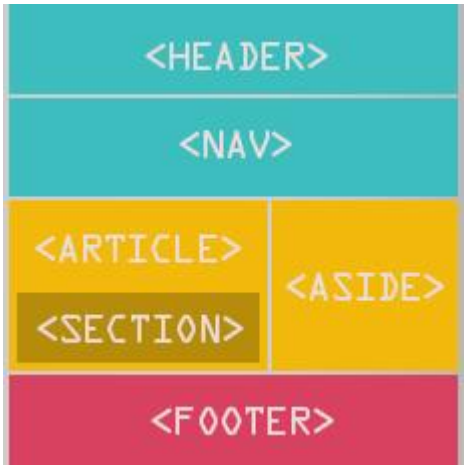
¿Qué es HTML?

HTML (HyperText Markup Language) es el lenguaje de marcado estándar utilizado para crear y estructurar contenido en la web. Define cómo se organizan los elementos en una página web y cómo deben mostrarse estos elementos en los navegadores.

¿Cómo funciona HTML?

HTML permite a los y las desarrolladores crear la estructura básica de una página web. El navegador interpreta el código HTML y lo renderiza en pantalla para que las personas puedan ver y navegar por la página.

Etiquetas Semánticas en HTML



¿Qué son las Etiquetas Semánticas?

Las etiquetas semánticas en HTML5 son aquellas que describen el propósito de diferentes partes de una página web, tanto para los navegadores como para los motores de búsqueda. Estas etiquetas mejoran la accesibilidad y la optimización de la página, facilitando la comprensión del contenido.

¿Para qué sirven las Etiquetas Semánticas?

Las etiquetas semánticas ayudan a estructurar mejor el contenido y permiten que los navegadores y motores de búsqueda interpreten más fácilmente la jerarquía y la importancia de la información en la página.

Ejemplos y Uso de Etiquetas Semánticas:

- `<header>`: define el encabezado de una página o sección. Contiene elementos como logotipos, títulos y menús de navegación.
- `<nav>`: representa una sección de navegación con enlaces a otras partes de la página o a diferentes páginas.
- `<main>`: contiene el contenido principal del documento, que es único y esencial por cada página.
- `<footer>`: define el pie de página, donde generalmente se encuentran los derechos de autor, enlaces adicionales y contactos.
- `<section>`: define una sección temática en el contenido que agrupa elementos relacionados.
- `<article>`: representa una pieza autónoma de contenido, como un artículo de un blog o una noticia.
- `<aside>`: contiene información adicional que puede estar relacionada con el contenido principal, como anuncios o enlaces secundarios.

Paso 2: Creá la estructura principal del documento:

Creá la siguiente estructura dentro de tu documento index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
</body>
</html>
```

Paso 2: Agregá etiquetas semánticas estructurales

Dentro de la etiqueta `<body>`, agregá las siguientes etiquetas semánticas:

```
    <header>
        <!-- Acá estamos dentro del header-->
    </header>
    <nav>
        <!-- Acá estamos dentro del nav-->
    </nav>
    <main>
        <!-- Acá estamos dentro del main-->
    </main>
    <footer>
        <!-- Acá estamos dentro del footer-->
    </footer>
```

Paso 3: Resultado Final

Siguiendo los dos pasos anteriores tendrás como resultado la estructura básica principal para seguir construyendo tu proyecto HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <header>
        <!-- Acá estamos dentro del header-->
    </header>
    <nav>
        <!-- Acá estamos dentro del nav-->
    </nav>
    <main>
        <!-- Acá estamos dentro del main-->
    </main>
    <footer>
        <!-- Acá estamos dentro del footer-->
    </footer>
</body>
</html>
```

Etiquetas Básicas más comunes en HTML

Encabezados (<h1> - <h6>)

Los encabezados son utilizados para definir títulos y subtítulos en una página web. HTML soporta seis niveles de encabezados, donde <h1> es el más importante y <h6> el menos relevante.

Ejemplo:

```
<h1>Este es el encabezado principal</h1>

<h2>Este es un subtítulo</h2>

<h3>Este es otro subtítulo</h3>
```

Párrafos (<p>)

La etiqueta <p> se utiliza para definir párrafos de texto. Es una de las etiquetas más utilizadas en HTML para organizar contenido textual.

Ejemplo:

```
<p>Esto es un párrafo. Aquí puedes escribir cualquier texto que desees mostrar en tu página.</p>
```

Negrita ()

La etiqueta se utiliza para resaltar texto con importancia, mostrándolo en negrita.

Ejemplo:

```
<p>Este es un <strong>texto en negrita</strong>.</p>
```

Itálica ()

La etiqueta se utiliza para enfatizar texto, mostrándolo en cursiva.

Ejemplo:

```
<p>Este es un <em>texto en cursiva</em>.</p>
```

Tachado (<s>)

La etiqueta <s> se utiliza para mostrar texto tachado, generalmente para indicar que algo ha sido eliminado o corregido.

Ejemplo:

```
<p>Este es un <s>texto tachado</s>.</p>
```

Integración de Etiquetas Básicas en el Proyecto:

- En el <header>: agrega un título principal usando <h1>, seguido de un subtítulo con <h2>.
- En el <main>: incluye un párrafo de introducción con <p>, y utiliza , , y <s> dentro de los textos según sea necesario.
- En el <footer>: podés repetir algunos estilos de texto o añadir un párrafo con contacto.

Ejemplo Integrado:

```
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mi Primer Proyecto</title>
</head>
<body>
<header>
    <h1>Bienvenidos a Mi Proyecto</h1>
    <h2>Este es un subtítulo en el header</h2>
</header>
<nav>

</nav>
<main>
      <p>Este es un párrafo de introducción sobre el proyecto. Aquí puedes usar
<strong>negrita</strong>, <em>cursiva</em>, y <s>tachado</s>.</p>
</main>
<footer>
    <p>Este es el pie de página con la información de contacto.</p>
</footer>
</body>
</html>
```

💭**Storytelling**

Bienvenida al equipo de desarrollo Talento Lab

¡Bienvenidos a Talento Lab!

Estás por empezar una nueva etapa en tu camino profesional. **Te acabas de unir al equipo de desarrollo de Talento Lab**, una iniciativa que busca formar a nuevos talentos digitales a través de proyectos reales, guiados por profesionales del rubro.

Acá no venís solamente a estudiar: **venís a ser parte de un equipo real**, que trabaja como lo hacen en las empresas tecnológicas del mundo actual.

¿Cuál es tu rol?

Sos un/a Desarrollador/a Front-End Trainee. Acabás de entrar al equipo como parte de un grupo de nuevos talentos que se están formando para dar sus primeros pasos en el desarrollo web. Tu misión es clara: **construir la base de un sitio web**, entender cómo se organiza un proyecto desde cero y aprender a trabajar en equipo aplicando buenas prácticas reales.

No estás solo/a. Te acompañan personas con experiencia, que serán tus referentes, y una asistente virtual que estará disponible para ayudarte siempre que lo necesites.

La historia comienza así…

Recién llegaste a la oficina de Talento Lab (virtual, pero muy real). Te asignaron una computadora, abriste tu entorno de desarrollo, y recibiste tu primera notificación.
Se abre un mensaje de bienvenida en pantalla, y ahí están:

Lucía – Product Owner

“Hola, bienvenido/a al equipo. Yo soy Lucía. En Talento Lab me encargo de asegurarme de que todo lo que construyamos sea útil, intuitivo y pensado para los usuarios. En este primer desafío, vamos a crear la estructura base del sitio web de Talento Lab. Tu trabajo va a ser muy importante: lo que hagas hoy va a ser el punto de partida para todo lo que venga después.”

Tomás – Desarrollador Senior

"¡Hola crack! Soy Tomás. No te asustes: todos empezamos por acá. Mi rol es ayudarte a escribir código limpio y entender cómo pensamos los desarrolladores cuando armamos una página web.
No todo es teoría: acá vas a practicar con proyectos reales, usando herramientas que usamos en el trabajo todos los días."

Tus primeras misiones (guiadas por el equipo):

- **Lucía**: "Queremos que armes la **estructura base del sitio en HTML**, usando etiquetas semánticas como `<header>`, `<nav>`, `<main>` y `<footer>`. Imaginá que estás diseñando el esqueleto de una casa."

- **Tomás**: "Usá buenas prácticas desde el comienzo. No te preocupes por los estilos todavía, lo importante es que el código sea claro, ordenado y entendible."

Ejercicio Práctico #1:

Crear la estructura básica del proyecto, incluyendo las etiquetas `<header>`, `<main>`, `<nav>` y `<footer>`. Dentro de `<header>`, incluye un título `<h1>` con el nombre del proyecto.

Para resolver este ejercicio, debés seguir estos pasos:

1. **Abrir Visual Studio Code:**
    - Abre Visual Studio Code y creá un nuevo archivo.
    - Guarda el archivo con el nombre index.html.
2. **Escribir la estructura básica del documento HTML:**
    - A continuación, debés escribir el código HTML que conformará la estructura básica de tu proyecto. El código debe incluir las etiquetas <header>, <nav> <main>, y <footer>.
3. **Guardar y visualizar en el navegador:**
    - Guardá los cambios y abre el archivo index.html en tu navegador para ver la estructura básica de la página web.

Ejercicio Práctico #2:

Agregar un archivo `README.md` en el proyecto, explicando brevemente de qué tratará la página que se va a desarrollar.

Para resolver este ejercicio, debés crear un archivo `README.md` en tu proyecto con una breve descripción del mismo.

1. **Crear el archivo `README.md`:**
    - Dentro del mismo directorio donde guardaste `index.html`, creá un nuevo archivo llamado `README.md`.
2. **Escribir la descripción del proyecto:**
    - Abrí el archivo `README.md` en Visual Studio Code y escribí una breve descripción del proyecto.

Ejemplo del contendio del `README.md`:

Nombre del Proyecto
Descripción:
Este proyecto es una página web básica desarrollada como parte de un curso de Front-End. La página está estructurada con HTML semántico y utiliza las etiquetas `<header>`, `<main>`, y `<footer>` para organizar el contenido. El objetivo es aprender a crear la estructura básica de una página web y prepararla para futuras mejoras con CSS y JavaScript.

Buenos Aires aprende

Buenos Aires Ciudad