**Datos de identificación del programa de formación**

| PROGRAMA DE FORMACIÓN | Desarrollo de aplicaciones web *full stack* |
| --- | --- |

| COMPETENCIA | 220501096 - Desarrollar la solución de *softwar*e de acuerdo con el diseño y metodologías de desarrollo. | RESULTADOS DE APRENDIZAJE | 220501096-02 - Construir la interfaz de usuario del *software* según el diseño determinado. |
| --- | --- | --- | --- |

| NÚMERO DEL COMPONENTE FORMATIVO | 12 |
| --- | --- |
| NOMBRE DEL COMPONENTE FORMATIVO | Construcción de la interfaz de usuario del *software* |
| BREVE DESCRIPCIÓN | En este componente formativo se trabajarán los temas de interfaces gráficas de usuario, con todos sus conceptos, buenas prácticas y como entorno visual de imágenes y objetos donde se da una interacción productiva, así como también el tema de experiencia de usuario que se enfoca, sobre todo, en páginas web y aplicaciones móviles, como un concepto clave dentro del entorno digital. |
| PALABRAS CLAVE | Framework, librería, HTML, CSS, JavaScript. |

| ÁREA OCUPACIONAL | 6 - VENTAS Y SERVICIOS |
| --- | --- |
| IDIOMA | Español |

**TABLA DE CONTENIDOS**

1. **Lenguajes de programación**
   1. HTML
   2. CSS

1.3 JavaScript

**2. Tecnologías de programación, *frameworks* y librerías**

**3. Codificación en lenguajes de programación**

**4. Estilo de codificación**

**5. Experiencia de usuario**

**INTRODUCCIÓN**

| Cuadro de texto |
| --- |
| Apreciado aprendiz, bienvenido a este componente formativo donde aprenderá a construir la interfaz gráfica de usuario, con el objetivo de establecer una buena experiencia de usuario al momento de usar la aplicación.  En el siguiente video conocerá, de forma general, la temática que se estudiará a lo largo del componente formativo. |

**GUION DE VIDEO INTRODUCTORIO**

| **Tipo de recurso** | Video *motion* | | | |
| --- | --- | --- | --- | --- |
| **NOTA** |  | | | |
| **Título** | Introducción | | | |
| **Escena** | **Imagen** | **Sonido** | **Narración** | **Texto** |
| **1** |  |  | Bienvenido a este componente formativo donde podrá actualizar sus conocimientos sobre la construcción de la interfaz de usuario del *software*. Se pretende brindar los conocimientos fundamentales sobre los diferentes lenguajes de programación que se utilizan para la creación de la GUI en un *software*; en este caso, se involucran HTML, CSS y JS como el conjunto de lenguajes predilectos para la creación de esta interfaz. Aparte de lo anterior, se revisarán las diferentes tecnologías web usadas en la programación, las cuales existen para tener ese conocimiento clave al momento de tomar una decisión durante el desarrollo del proyecto. | *Software*  Lenguajes de programación  Tecnologías web  Conocimiento  Proyecto |
| **2** |  |  | En la industria del *software*, los diseños de la interfaz de usuario son importantes porque son la cara visible al cliente, a la interacción de ellos con el *software* y definen la experiencia positiva o negativa del cliente. Por eso, es importante tener claros los conceptos de los lenguajes de programación y entender qué son los *framework*, los cuales son aplicados de manera fluida en la industria y en el desarrollo de todos los proyectos web, saber cuáles son los tipos que existen y cuáles son sus características principales. | Diseños  Interfaz  *Software*  Programación  *Framework*  Industria |
| **3** |  |  | La industria también provee a las personas encargadas de la construcción de esta interfaz las librerías, que han sido creadas por los diferentes proyectos evolucionados y por las diferentes empresas; en este componente, también se aborda qué es una librería, para tener conocimiento de cuándo se puede utilizar.  Al ser distinta a un *framework*, una librería no proporciona la fundamentación sobre cómo llevar a cabo el desarrollo, sino que genera funcionalidades comunes, las cuales han sido resueltas con anticipación por otros programadores y evitan la duplicidad de código. Igualmente, reducen el tiempo de desarrollo y aumentan la calidad del mismo. | Librerías  Interfaz  Empresas  Desarrollo  Programadores |
| **4** |  |  | En un proceso de construcción de interfaces, aparte de identificar los objetos que la componen, es importante conocer la nomenclatura del lenguaje de programación, la forma como se declaran estos objetos, variables o constantes; entender qué es un paquete, cómo se crean las clases, cómo se declaran las variables y sus funciones.  Igualmente, se detallan los diferentes estilos de codificación, cómo se usan las sentencias de control, cómo se declara una variable, cómo se aplican comentarios en el código y las buenas prácticas de desarrollo. | Interfaces  Lenguaje  Objetos  Variables  Codificación |
| **5** |  |  | Este aprendizaje tendrá un carácter teórico-práctico, o sea, se llevará una secuencia de teorías muy importantes para la conceptualización de los conceptos sobre la construcción de las interfaces gráficas de usuario, sus elementos, sus objetos y los diferentes patrones de diseño y navegación. Además, al tener los conceptos claros, se irán aplicando de forma práctica, para construir la interfaz completa. | Aprendizaje  Gráficas  Usuario  Elementos |
| **6** |  |  | Aparte de lo anterior, se tendrá claridad en cuanto a las diferentes herramientas para la construcción de interfaces, de acuerdo con el lenguaje de programación a utilizar, el *framework* o las librerías a usar. | Herramientas  Interfaces  *Framework* |
| **7** |  |  | Así mismo, se busca que logremos desarrollar **conocimientos** basados en la experiencia, qué son y para qué sirven, señalar algunos e indicar su objetivo y cuáles son las características principales que impactan al usuario, para experimentar con la aplicación. | Conocimientos  Objetivo  Aplicación |
| **Nombre del archivo** | 228125\_v1 | | |  |

**DESARROLLO DE CONTENIDO**

1. **Lenguajes de programación**

| Cuadro de texto |
| --- |
| Comencemos diciendo que los lenguajes de programación son un conjunto de instrucciones a través de las cuales se pueden programar funcionalidades que serán ejecutadas a través de una computadora. Conozcamos los diferentes lenguajes. |

**1.1 HTML**

| Cuadro de texto |
| --- |
| Antes de entrar a revisar el lenguaje HTML5, se deben tener presentes un conocimiento base y unos conceptos fundamentales.  Toda la tecnología web nace con la aparición del internet; en realidad, internet es la unión de dos palabras ***interconnected*** y ***network***, lo que quiere decir que es una red global de computadores interconectados entre sí, que permiten compartir información como texto, sonido, imágenes, etc. |

**Infografía 1**

| **Tipo de recurso** | Infografía interactiva Punto caliente | |
| --- | --- | --- |
| **Texto introductorio** | Tim Berners Lee, inventor de la *World Wide Web* (WWW), fundó el consorcio W3C para estandarizar, supervisar y estar al tanto de las tecnologías basadas en el internet; debido a este procedimiento, se originan los siguientes conceptos: | |
| Los textos y enlaces se encuentran en la carpeta Anexos con el nombre Infografia1.pptx | | |
| **Código de la imagen** | 225125\_i1 | |
| **Punto caliente 1** | Es el protocolo de transferencia de hipertexto y es el apoyo que permite la operación de la comunicación entre los dispositivos conectados en la red; por ejemplo, entre computadores de usuarios y los servidores. | HTTP  *Hypertext Transfer Protocol* (Protocolo de Transferencia de Hipertextos) |
| **Punto caliente 2** | Es el localizador de recursos únicos. Es más conocida como la dirección de un sitio web y es la forma como buscamos un sitio en algún punto de la red, a través de su nombre. | URL  *Uniform Resource Locator* (Localizador de Recursos Uniforme) |
| **Punto caliente 3** | El lenguaje de marcado de hipertexto es el que permite escribir la estructura de una página web, dando recursos de maquetación como listas, párrafos, viñetas, etc. | HTML  *Hypertext Markup Language*  (Lenguaje de marcas de hipertexto) |
| **Punto caliente 4** | Es el programa que se encarga de gestionar las peticiones con los navegadores: tanto de recibirlas, como de responderlas. | Servidor web HTTP |
| **Punto caliente 5** | Es el *hosting*, el cual corresponde a un equipo conectado a la red, que tiene una configuración específica y una serie de programas instalados, entre ellos un servidor web. | Alojamiento web |

| Tipo de recurso | Cajón de texto de color |
| --- | --- |
| Otro de los conceptos importantes que debemos tener en cuenta es el de navegador. Los navegadores son los programas en donde se van a ejecutar las aplicaciones web.  Un navegador es un programa encargado de interpretar el código que se recibe desde el servidor y, luego, exponerlo en un término entendible para que el usuario lo visualice, a través de la página web, con texto, imágenes, etc. | |

| **Tipo de recurso** | Tarjetas Avatar | |
| --- | --- | --- |
| **Introducción** | Aparte de interpretar y mostrar al usuario la página, algunas de las responsabilidades del navegador son: | |
| **Integridad**  Para la transmisión de las páginas, solo se usa el protocolo HTTP como lenguaje de comunicación entre servidor y navegador. | | **Imagen** 225125\_i2 |
| **Seguridad**  Como el navegador interpreta diferentes códigos, desde diversas fuentes, utiliza seguridad en la conexión a través del protocolo HTTPS y se ejecuta a través de CORS. | | **Imagen:** 225125\_i3 |
| **Optimización**  Para optimizar la carga de las páginas, el navegador utiliza la memoria caché para guardar una copia y, de esta manera, garantizar que la próxima carga sea ligera. | | Icono de memoria caché vector editable  **Imagen:** 225125\_i4 |

**Peticiones HTTP**

| Cuadro de texto |
| --- |
| Cuando los navegadores acceden a una página web, lo hacen a través de **http,** como se había dicho anteriormente; en esa comunicación con el servidor, se descompone la URL en diferentes partes: una parte es el *host* que es la dirección de la máquina y, la otra, el *path* o la ruta del recurso al que se quiere acceder. |

| **Tipo de recurso** | | Pestañas o tabs Verticales |
| --- | --- | --- |
| **Introducción** | | Existen técnicas eficientes que pueden enviar los datos a un servidor o navegador, cuando se crea una petición por parte del usuario; por ahora, vamos a explicar dos de ellas: |
| solicitud icono gratis  **Imagen:** 225125\_i5 | | |
| **GET** | Es cuando se hace una petición o solicitud al servidor y se envía la información solicitada, a través de la URL, como lo hace Google. | |
| **POST** | Es una petición donde se envían los datos ocultos, a través de un objeto que será descifrado por el servidor y, de esta manera, se tiene seguridad de la información. | |

| **Tipo de recurso** | Tarjetas Avatar | |
| --- | --- | --- |
| **Introducción** | Cuando se hace una petición, se pueden obtener varios mensajes como respuesta; puede ser desde una respuesta exitosa, hasta una de error. | |
| **200**  La petición solicitada se encuentra con éxito. | |  |
| **403**  No se tiene permiso para acceder al recurso solicitado | |  |
| **404**  El recurso solicitado no está disponible o no se pudo encontrar. | |  |

| Tipo de recurso | Cajón de texto de color |
| --- | --- |
| Las primeras páginas web fueron estáticas y rígidas, no permitían la fluidez del contenido, la organización dinámica del mismo o la aplicación de estilos. Debido a esa problemática, en 1994, nace el CSS (hoja de estilos en cascada), lenguaje que se compone de una serie de reglas que permiten dar apariencia a la página web e incluir elementos como: color de la letra, márgenes, el tamaño de los elementos y, también, posibilita la adaptación de las páginas a las diferentes pantallas de los dispositivos, lo que se llama ***responsive design***. | |

| **Tipo de recurso** | Pestañas o tabs horizontales | |
| --- | --- | --- |
| **Introducción** | HTML5 mejora la versión de HTML, de acuerdo con el avance en las API y en el conjunto de herramientas que conforman la tecnología web; pero, aparte de esto, presenta nuevas características en el lenguaje HTML, las cuales son: | |
| **Semántica** | La semántica se define como el lenguaje o la forma de expresarse y, en este caso, es la forma de escribir el lenguaje.  En este apartado se crearon nuevas etiquetas que permiten una funcionalidad más amplia del lenguaje y ventajas en el manejo de páginas.  Se crearon las nuevas etiquetas que seccionan la página, como: <section>, <article>, <nav>, <header>, <footer> y <aside>. Estas etiquetas se profundizarán más adelante y, por ahora, sabemos que permiten dar estructura a la página.  Las nuevas etiquetas de audio y video (<audio>, <video>) permiten, al desarrollador, manipular audio y video de forma sencilla.  La etiqueta <video> permite incrustar en la página, un video de forma fácil; a través de sus atributos ***src*** y ***controls***, se agregan el video y los botones de control para su reproducción.  El elemento <audio> funciona de forma similar a la etiqueta <video>, con la diferencia que ocupa menos espacio en la página y no soporta los atributos de *width* y *height*.  Otra nueva característica en semántica, se refiere a formularios en HTML5; se incluye nueva API de validación; aparte, atributos nuevos, nuevos valores para la etiqueta ***input*** con el atributo ***type*** y ***output***.  Además de lo anterior, también se incluyen nuevos elementos semánticos que permiten incluir directamente fórmulas matemáticas (MathML), las etiquetas <mark>, <figure>, <data>, <time>, entre otras, y una mejora en el funcionamiento de la etiqueta <iframe>. | etiqueta html icono gratis  Imagen: 225125\_i6 |
| **Conectividad** | En la parte de conectividad, se tienen unas características nuevas que se muestran a continuación:   * ***Web Sockets*:** es una API que permite abrir una sesión de manera bidireccional entre usuario y servidor, permitiendo el envío de peticiones por eventos, sin tener que estar testeando el servidor en cada llamada. * ***Server-send events*:** esta función se agrega cuando se necesita que la aplicación web transmita eventos desde el servidor al *front-end* del cliente; se hace una conexión unidireccional, es decir, el cliente no podrá responder al servidor. * ***WebRTC***(*Web Real-Time Communication*): es una tecnología que permite a las aplicaciones y sitios web, capturar y, opcionalmente, transmitir audio y/o video, así como intercambiar datos arbitrarios entre navegadores, sin necesidad de un intermediario. El conjunto de estándares que componen WebRTC hace posible compartir datos y realizar teleconferencias *peer-to-peer*, sin requerir que el usuario instale complementos o cualquier otro *software* de terceros. | Conectividad icono gratis  Imagen: 225125\_i7 |

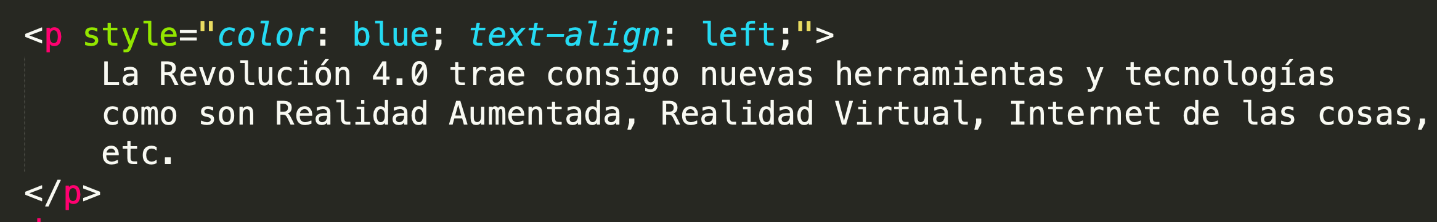
| Cuadro de texto |
| --- |
| Lo invitamos a leer los siguientes documentos para ampliar la información sobre la temática que venimos estudiando:   * Estructura del lenguaje HTML5. * Etiquetas básicas. * Formularios. |

**1.2 CSS**

| Cuadro de texto |
| --- |
| CSS – Hoja de estilos en cascada, (*Cascading Style Sheets* por sus siglas en inglés), es el encargado de darle apariencia a la página web. Esta hoja de estilos sirve para especificarle al navegador cómo se deben mostrar los elementos de la página (documentos, información, imágenes, etc.) y cómo estarán organizados o diseñados.  Cuando se habla de CSS se habla de reglas que se aplican a los elementos de la página o a un grupo de ellos.  Una regla es un código que le indica a un elemento, o a un grupo de ellos, cómo debe ser su diseño; es decir, qué color tendrá, si tiene fondo o color de fondo, tipografía, organización, etc.  Las reglas de CSS se pueden anidar, es decir, se pueden agregar reglas a etiquetas que estén dentro de otras, de acuerdo con las especificaciones dadas en el selector.  Para la aplicación de dichas reglas, anteriormente se hacía en la misma línea de donde se declaraba la etiqueta HTML, por ejemplo: |

**Figura 1**

*Estilos embebidos*



| Cuadro de texto |
| --- |
| Como se observa en la imagen anterior, cuando se colocan los estilos dentro de la misma etiqueta, se llaman **estilos embebidos**. Esto se hacía con las páginas hace varios años atrás, pero, a medida que la información fue creciendo, la tecnología fue avanzando y se requirieron páginas dinámicas, lo que necesita una gran cantidad de estilos y tenerlos embebidos se vuelve una mala praxis por la confusión que puede generar al programador y por el peso dado a cada página, con códigos repetitivos.  Por lo anterior, la hoja de estilos en cascada es la mejor y más eficiente práctica para la aplicación de estilos en una página web y se hace a través de un archivo externo, el cual se relaciona en la cabecera de la página.  Para relacionar un archivo externo, se hace de la siguiente manera: |

**Figura 2**

*CSS externo*

Imagen que contiene dibujo, alimentos

Descripción generada automáticamente

| Cuadro de texto |
| --- |
| Al observar la imagen anterior se ve la etiqueta enlace de HTML que es la que permite relacionar el archivo externo de CSS a través de su propiedad **href**.  **href** solicita la URL donde se encuentra el archivo externo a relacionar; se debe tener presente que esa dirección debe estar dentro de la carpeta de proyecto, para que no se presente ningún error al momento de subir el proyecto al *hosting*.  Con los conceptos anteriores ya se puede empezar a observar los estilos CSS y hacer un ejemplo de maquetación de páginas, para lo cual lo invitamos a leer los siguientes documentos:   * **Hojas de estilo CSS.** * **Diseño web *responsive.*** |

**1.3 JavaScript**

| Cuadro de texto |
| --- |
| JavaScript es un lenguaje de programación interpretado y ligero, se compila en tiempo real (*just-in-time*) con diferentes tipos de funcionalidades. Su uso más conocido es la programación de *script* aplicados en entornos web, pero es usado en muchos más entornos como AngularJS, NodeJS, Adobe Acrobat, entre otros.  Una de las evoluciones más favorables fue la creación de motores de interpretación que aceleran el procesamiento del código. El éxito de dichos motores es que transforman el código JavaScript a código máquina, para tener velocidades similares a las de aplicaciones de escritorio; con lo anterior, JavaScript se confirmó como la mejor opción para aplicación en la web.  Para aprovechar todas las ventajas de los nuevos motores, JavaScript se expandió en portabilidad e integración. También se agrega APIs por defecto a los navegadores, para asistir el lenguaje en funciones básicas. Algunas de las APIs son Canvas y también Web Storage, entre otras más.  Para ampliar la información sobre Canvas, lo invitamos a leer el documento **Canvas**. |

**2. Tecnologías de programación, *frameworks* y librerías**

| Cuadro de texto |
| --- |
| Cuando se habla de tecnologías de programación, se refiere a los lenguajes de programación que se pueden utilizar para el desarrollo de *software*, así como las diferentes herramientas y *frameworks* que se pueden aplicar. |

**Infografía 2**

| **Tipo de recurso** | Infografía interactiva Punto caliente | |
| --- | --- | --- |
| **Texto introductorio** | Entre estas tecnologías podemos encontrar las siguientes: | |
| Los enlaces y editables se encuentran en la carpeta Anexos con el nombre Infografia2.pptx | | |
| **Código de la imagen** | 225125\_i10 | |
| **Punto caliente 1** | Es una de las tecnologías web más usadas. Con este lenguaje de programación multiplataforma es posible dar una mayor interactividad y dinamismo a los sitios web. Entre sus distintas funcionalidades, JavaScript permite crear tanto animaciones como objetos, localizar errores en formularios, cambiar elementos web de manera intuitiva y crear *cookies*.  Además, también permite desarrollar aplicaciones tan potentes como Facebook o Twitter. Por esto, JavaScript es una de las mejores tecnologías para desarrollo web de la actualidad. | **JavaScript** |
| **Punto caliente 2** | Es el lenguaje de marca o etiquetas utilizado para la creación de sitios web. Con HTML se realiza la estructura del sitio, su maquetación a través de etiquetas que permiten la organización de contenido. Estas etiquetas son interpretadas por los navegadores para mostrar la estructura, tal cual como fue organizada.  En conclusión, HTML permite a los desarrolladores FrontEnd describir, estructura o maquetar el contenido del sitio web, como es su información a través de párrafos, imágenes, listas, tablas, barras de menú, etc. | **HTML** |
| **Punto caliente 3** | CSS es la hoja de estilos en cascada que, como su nombre lo indica, permite agregar estilos o diseño a las páginas web o desarrollo de aplicaciones web. Haciendo una analogía el HTML es como hacer la estructura de la casa y el CC ya permite estucar, colocar baldosa, etc.  Este lenguaje da diseño a la maquetación o representación visual que se hizo en HTML, dándole color, estilos de borde, fondo, animaciones, etc. | **CSS** |

**Tecnologías del lado del servidor**

| Cuadro de texto |
| --- |
| Las tecnologías del lado del servidor o también llamadas *backend* son las que tienen como objetivo implementar los comportamientos que tendrá la web en el servidor; es decir, sus peticiones, cómo se tratan, cómo se consume la capa de datos, etc. |

| **Tipo de recurso** | | Pestañas o tabs Verticales |
| --- | --- | --- |
| **Introducción** | | Entre ellas, algunas de las tecnologías para desarrollo web más populares son: |
| Concepto de diseño web de cursos de programación con personas. Estudiantes estudiando con la escena de las computadoras. Composición en línea de cursos de TI en estilo plano. Ilustración vectorial para materiales promocionales de medios sociales.  **Imagen:** 225125\_i11 | | |
| **PHP** | PHP es un lenguaje de programación del lado del servidor, que permite la comunicación de la web con servidores de datos o persistencia. Esta funcionalidad permite crear la capa de datos en los desarrollos web y comunicarse con diferentes motores de bases de datos como MYSQL, PostgreSQL, etc.; además, con este lenguaje de programación se gestiona todo el comportamiento del servidor, el tratamiento de archivos, imágenes, la recopilación de información desde los formularios del sitio, entre otras. | |
| **Python** | La popularidad de Python se debe a que, además de tener una gran cantidad de librerías, es totalmente gratis. Asimismo, destaca por su sencillez y potencia, porque usa menos líneas de código que cualquier otro lenguaje de programación, para cualquier aplicación que se ejecute. Esto supone una gran ventaja con respecto a otros tipos de lenguajes de programación y lo convierte, sin duda, en una de las tecnologías web más usadas. | |
| ***Frameworks* y librerías** | Los lenguajes de programación son importantes, pero, aparte de ellos, existen *frameworks* que permiten dar una forma o buenas prácticas sobre cómo trabajar el desarrollo. Son una serie de herramientas y librerías de código ya escrito, que facilitan a los desarrolladores las tareas cotidianas o de uso común, para evitar ejecutarlas desde cero.  Los *frameworks* dan un marco de trabajo para programar en un lenguaje y las librerías se encargan de solucionar problemas, haciendo el código más legible. | |
| **Angular** | Entre los *frameworks* más usados se encuentra Angular, perfecto para aplicaciones web. Además, utiliza HTML para realizar UI y es compatible con las últimas versiones de Chrome, Firefox, Edge, iOS y Android. | |
| **React** | React está escrito en código abierto en JavaScript, muy útil para construir las interfaces de usuario. Es una biblioteca, aunque muchos programadores la consideran un *framework* por la excelencia de sus componentes. Asimismo, admite y combina diferentes idiomas y tecnologías. | |

***Frameworks***

| Cuadro de texto |
| --- |
| Un *framework,* como se dijo en el ítem anterior, es un marco de trabajo para programar en diferentes lenguajes, usando librerías, buenas prácticas y una forma más sencilla de ejecutar funcionalidades, al reutilizar código.  El concepto de *framework* no es de uso exclusivo de la tecnología web, del desarrollo de sus aplicaciones; también se puede encontrar en aplicaciones de realidad aumentada, de inteligencia artificial y de escritorio. Es decir, es una estructura compuesta por componentes reutilizables e intercambiables, durante el proceso de desarrollo de una aplicación.  Existen varios tipos de *frameworks* web: orientados a la interfaz de usuario, como Java Server Faces; orientados a aplicaciones de publicación de documentos, como Cocoon; orientados a la parte de control de eventos, como Struts, y algunos que incluyen varios elementos, como Tapestry.  La mayoría de *frameworks* web se encargan de ofrecer una capa de controladores, de acuerdo con el patrón MVC o con el modelo 2 de Servlets y JSP, ofreciendo mecanismos para facilitar la integración con otras herramientas para la implementación de las capas de negocio y presentación. |

| **Tipo de recurso** | Acordeón tipo 2 |
| --- | --- |
| **Introducción** | A continuación, algunas características que pueden tener los diferentes *frameworks*: |
| Frameworks outline icon. Thin line style from big data icons collection. Pixel perfect simple element frameworks icon for web design, apps, software, print usage  **Imagen:** 225125\_i12 | |
| **Acceso a datos**  Se incluyen clases, herramientas, interfaces, con las cuales se puede integrar la manipulación de bases de datos de una forma más directa y sencilla. | |
| **Controladores**  Todos los *framework*s tienen implementada, una serie de controladores con los cuales se pueden gestionar eventos, como las operaciones CRUD, formularios de captura de datos, etc. | |
| **Autenticador**  Incluyen mecanismos con los cuales identificar usuarios a través del inicio de sesión. | |
| **Abstracción**  Cuando se habla de abstracción de URL y sesiones, no se necesita manipular directamente la URL ni las variables de sesión; el *framework* se encarga de eso. | |

**Librerías**

| Cuadro de texto |
| --- |
| Para conocer qué es una librería, sus tipos y características, lo invitamos a ver el siguiente video. |

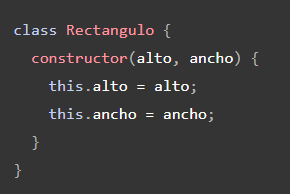
| **Tipo de recurso** | Video motion | | | |
| --- | --- | --- | --- | --- |
| **NOTA** |  | | | |
| **Título** | **Librerías en programación** | | | |
| **Escena** | **Imagen** | **Sonido** | **Narración** | **Texto** |
| **1** |  |  | En programación, una librería es un archivo o conjunto de archivos de código, que se utiliza para facilitar la programación y desarrollar *software*. Se fundamenta en archivos de código, los que llamamos al principio de la página como, por ejemplo, una librería JavaScript.  Podemos insertar cualquier archivo al principio de la página; sin embargo, no todos los archivos pueden considerarse librerías. Para que un archivo sea una librería, puede usarse en distintas páginas, no solo del mismo sitio, sino de varios sitios. Además, debe modificar el código, de tal manera que nos facilite la inserción de un nuevo código en la página. | Librería  Archivo  *Software*  Código |
| **2** |  |  | Un ejemplo de librería es la inclusión de un archivo reset.css que resetea la página para que esta se vea igual en todos los navegadores. El archivo es útil para páginas de distintos sitios y asegura que el estilo de la página se verá igual en todos los navegadores.  Resumiendo, una librería es un conjunto de instrucciones que pueden ser utilizadas por diferentes programas. La misma librería se utiliza en varios programas, dándole un uso igual o uno diferente. Es una herramienta que permite optimizar tiempo y recursos. | Archivo  Navegador  Página  Recursos |
| **3** |  |  | Las bibliotecas de programación las podemos definir como espacios donde se aseguran archivos que más adelante van a utilizar los programadores, para ejecutar un programa. Existen diferentes tipos de bibliotecas y estas se usan según el programa o aplicación que se desea desarrollar. | Biblioteca  Programación  Programador |
| **4** |  |  | **Tipos de librerías**  **Librerías propias.** Si se entiende el concepto de librería, no será difícil explicar qué es una librería propia. Cuando se habla de propia es porque el mismo desarrollador del proyecto la crea, porque tiene un *set* de operaciones que se repiten a lo largo de la funcionalidad del sistema y, por eso, crea la librería con sus mejores prácticas, englobando dichas tareas, para no escribirlas repetidamente, sino que solamente se usan las de la librería y así se optimiza su desarrollo.  Las librerías propias son, por lo tanto, archivos que fabrica el propio programador para facilitar su propio trabajo, los cuales puede incluir en cualquier página cuando lo necesite. | Librería propia  Desarrollador  Sistema |
| **5** |  |  | **Librerías externas.** El concepto de librería externa hace referencia a una librería propia, pero ampliada por una comunidad, que existe detrás de su creación y mantenimiento. También se pueden dar librerías desarrolladas por expertos de diferentes empresas como Google, Facebook, entre otras.  Con su uso, se pueden ejecutar funciones ya escritas, consultar datos, validar sesiones, ejecutar pruebas de código, entre otras funciones. Por eso, es de vital importancia que se aprenda a programar usando librerías y no volver a inventar funciones que ya están creadas por expertos; utilizar patrones, buenas prácticas y, sobre todo, que estén testeadas para validar su calidad. | Librería externa  Experto  Funciones  Programar |
| **6** |  |  | **Características**  A continuación, se nombran algunas de las características más importantes de las librerías para lograr obtener una mejor comprensión sobre estos archivos: | Librerías  Archivos |
| **7** |  |  | * Aportan comandos e instrucciones. * Pueden ser manejadas en distintas capas y niveles en el programa. * Reducen el tiempo para escribir el código. * Reutilización de código. * Algunas permiten realizar pruebas unitarias de código. | Comandos  Capas  Programa  Código |
| **Nombre del archivo** | 225125\_v2 | | |  |

**3. Codificación en lenguajes de programación**

| Cuadro de texto |
| --- |
| La nomenclatura de JavaScript indica la forma como se escriben sus variables, es decir, su sintaxis, gramática, cómo se puede aplicar. Este tema, en concreto, será detallado más adelante, cuando se estén estudiando variables, constantes y funciones.  En JavaScript existen los gestores de paquetes, los cuales se encargan de gestionar las dependencias entre código y liberar así de esta ardua tarea, al programador. Estas herramientas son gestores como Gradle, Maven Ant y su función es encargarse de la instalación, actualización, configuración y eliminación de los diferentes paquetes que necesite el *software* para su funcionamiento.  Las clases en JavaScript permiten la programación orientada a objetos, con sus atributos, sus propiedades, sus constructores y permitiendo de una forma sencilla, la creación de objetos y la implementación de la herencia. |

**Figura 3**

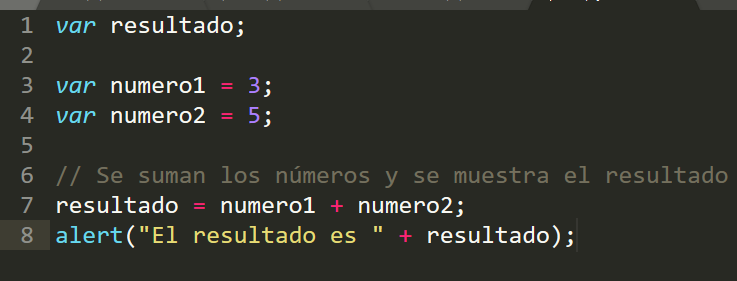
*Declaración de clase*



| Cuadro de texto |
| --- |
| Las funciones son útiles en el proceso de desarrollo de aplicaciones web, donde una y otra vez se repiten las mismas instrucciones en varias partes del desarrollo. Por ejemplo, si se está haciendo una tienda virtual, siempre se debe estar calculando el precio de los productos, añadiendo los impuestos necesarios y los gastos de envío; para este proceso es posible utilizar un ***script****.*  Un ejemplo sencillo para determinar cómo se escribe una función, es la suma de dos números; en JavaScript, si yo quiero sumar dos números, lo hago como se observa en la imagen. |

**Figura 4**

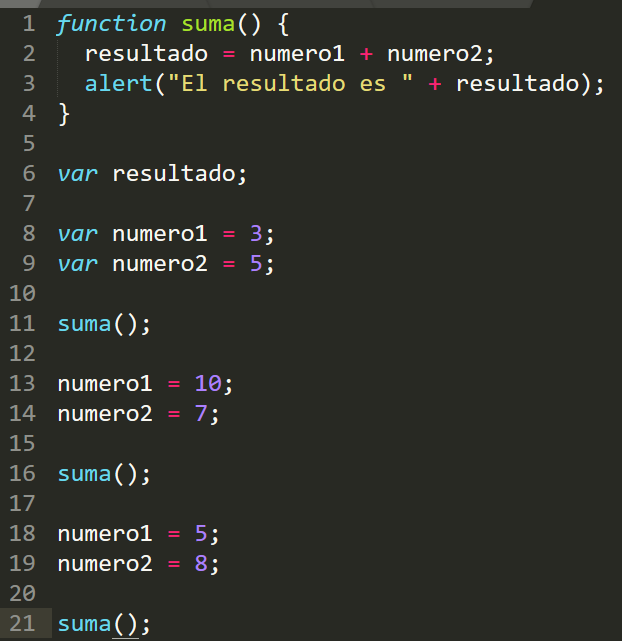
*Ejemplo suma de dos números*



| Cuadro de texto |
| --- |
| La imagen anterior muestra cómo se agrega código que declara tres variables, la de número1 y número2, se les asigna un valor y después se hace la operación para guardar el resultado en la variable resultante.  Si se supone que esta acción se debe hacer varias veces en diferentes partes del código, entonces, se estaría siempre agregando 8 líneas de código y no sería muy óptimo dejar tal cantidad de código, cuando se puede optimizar, usando una función. Se deberá crear una función que haga la respectiva suma y muestre el resultado. |

**Figura 5**

*Usando función suma*



| Cuadro de texto |
| --- |
| La imagen anterior muestra el funcionamiento de una función, pero se debe explicar su sintaxis.  **funtion nombre\_funcion ( argumentos ) {**  **}**  La sintaxis de una función es la que se mostró anteriormente, es decir, se declara con la palabra clave **function,** se le asigna un nombre y, entre los paréntesis, van los argumentos o los parámetros de entrada que recibe; recuerde que una función puede necesitar parámetros de entrada o puede funcionar sin ellos, de acuerdo con la necesidad y el contexto.  Después de explicar la sintaxis, todas las funciones se crean de la misma forma y, por eso, en el ejemplo de la imagen, se creó la función suma, la cual suma los dos números, los guarda en la variable resultado y genera un mensaje emergente con una ventana de diálogo para mostrar su resultado a través de la palabra clave ***alert*.** |

**Variables**

| Cuadro de texto |
| --- |
| Las variables, en programación, son espacios en memoria que se identifican a través de un nombre dado y permiten guardar un valor de un tipo de dato dado.  En JavaScript se manejan de igual forma; lo único es que es un lenguaje no tipado y, por eso, no se le debe dar el tipo de dato a la variable, solo utilizar la palabra clave **var** para definirla. Por ejemplo:  **var nombre;**  La línea anterior crea una variable en JavaScript llamada nombre, la cual podrá almacenar cualquier tipo de dato, pero como las buenas prácticas indican que se deben dar nombres coherentes a las variables, lo ideal es que, si se llama así, guarde un valor de un nombre de una persona.  La palabra clave **var,** antes de 2015, era la única forma de declarar una variable para JavaScript; pero la nueva versión, después de 2015, permite al lenguaje definir constantes, es decir, variables cuyo valor nunca va a cambiar, dentro de la ejecución de todo el programa y para esto utiliza la palabra clave **const.** |

| Cuadro de texto |
| --- |
| La otra forma que salió es la palabra clave **let** que define una variable con alcance restringido.  Con la forma de definir las variables, se debe expresar que se usan, no solo para mantener valores, sino para realizar expresiones algorítmicas con diferentes operadores.  Las variables declaradas en cada lenguaje de programación, al igual que en JavaScript, deben identificarse con nombres únicos denominados identificadores.  Los identificadores pueden ser nombres cortos (como **x** o **y**) o nombres más descriptivos (edad, suma, volumen total). |

| **Tipo de recurso** | Carrusel de tarjetas | |
| --- | --- | --- |
| **Introducción** | Existen reglas generales para construir nombres de variables (identificadores únicos) que son: | |
| Los nombres pueden contener letras, dígitos, guiones bajos y signos de dólar. | |  |
| Los nombres deben comenzar con una letra. | |  |
| Los nombres también pueden comenzar con **$** y **\_** (pero no son usados comúnmente y acá tampoco se usarán). | |  |
| Los nombres distinguen entre mayúsculas y minúsculas (z y Z son variables distintas). | |  |
| Las palabras reservadas del sistema o lenguaje no se pueden usar como nombres. | |  |

| Cuadro de texto |
| --- |
| Con lo anterior un ejemplo de declaración de variable es:  **var x;**  y si se le asigna un valor a través del operador de asignación que es el igual, sería de la siguiente manera:  **var x= 5;**  Se empezará observando que las variables pueden tener tipos de datos numéricos como **x** que se le asigna el valor de 5 o valores de texto, que se asignan entre comillas.  **var nombre = “Jose Aguilar”;**  Ahora, también se puede interactuar con etiquetas HTML; por ejemplo, asignarle el valor de una variable a una etiqueta de párrafo HTML.  **<p id="ejemplo"></p>**  **<script>**  **var carName = "Audi";**  **document.getElementById("ejemplo").innerHTML = carName;**  **</script>**  Lo anterior, en tiempo de ejecución, ejecuta el Script y llama al elemento con el **id** ejemplo y le asigna el valor de la variable carName.  Ahora vamos a revisar los **operadores**; estos, al igual que en cualquier otro lenguaje, son: |

**Tabla 1**

*Operadores JavaScript*

| **Operador** | **Descripción** |
| --- | --- |
| + | Suma |
| - | Resta |
| \* | Multiplicación |
| / | División |
| \*\* | Exponente |
| ++ | Incremento |
| -- | Decremento |

| Cuadro de texto |
| --- |
| Estos operadores también se pueden conjugar con el de asignación y escribirse de la siguiente manera: |

**Tabla 2**

*Operadores de asignación*

| Operador | Ejemplo | Igual a |
| --- | --- | --- |
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x – y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |
| \*\*= | x \*\*= y | x = x \*\* y |

| Cuadro de texto |
| --- |
| Aparte de los operadores aritméticos revisados en los cuadros anteriores, también se cuenta con operadores de comparación: |

**Tabla 2**

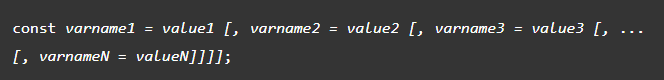
*Operadores de comparación*

| **Operador** | **Descripción** |
| --- | --- |
| == | Igual a |
| === | Igual valor e igual tipo de dato |
| != | Diferente |
| !== | Diferente valor o diferente tipo de dato |
| > | Mayor que |
| < | Menor que |
| >= | Mayor o igual que |
| <= | Menor o igual que |
| ? | Operador ternario |

| Cuadro de texto |
| --- |
| Con todos los operadores anteriores, se pueden combinar las variables y hacer expresiones aritméticas o usarse en los condicionales como pruebas lógicas para determinar por cual camino seguir, según se cumpla o no, la condición a través del condicional **if** y los operadores de comparación.  Ahora, las constantes son nombres que se asignan a espacio en memoria, para guardar un único valor, el cuál no podrá cambiar en ningún momento de la ejecución del *software*. |

**Figura 6**

*Sintaxis de constantes*

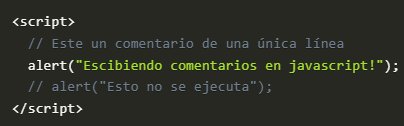
****

**4. Estilo de codificación**

| Cuadro de texto |
| --- |
| Los comentarios son parte del desarrollo que están en el código fuente, pero que el sistema no los ejecuta, los pasa por alto; sirven para documentar el código e identificar ciertas partes importantes en su mantenimiento a futuro.  Los comentarios pueden ser de una línea o pueden ser comentarios de varias líneas, como se presenta a continuación. |

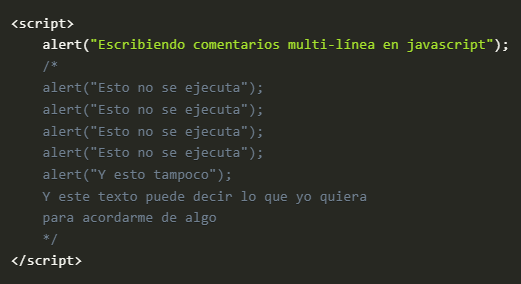
**Figura 7**

*Comentario de una sola línea*



**Figura 8**

*Comentario de varias líneas*

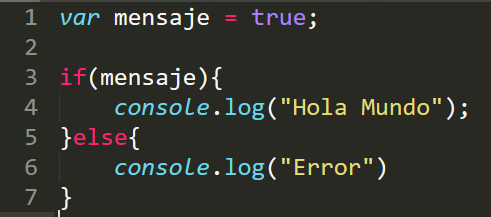


| **Tipo de recurso** | Tarjetas Animadas |
| --- | --- |
| **Introducción** | En JavaScript, existen tres formas de declarar variables; estas declaraciones son sensibles a mayúsculas y minúsculas. |
| **var** | Declara una variable, opcionalmente, la inicia a un valor. |
| **let** | Declara una variable local con ámbito de bloque, opcionalmente, la inicia a un valor. |
| **const** | Declara un nombre de constante de solo lectura y ámbito de bloque. |

| Cuadro de texto |
| --- |
| Para dar inicio a las estructuras de control, se va a iniciar con los condicionales, estructura que permite controlar el flujo del programa o de la función que se esté realizando.  **If (condición) {**  **} else {**  **}**  La sintaxis es la anterior, donde se puede ver que se aplica igual que en los lenguajes como **php** o Java. A continuación, un ejemplo de su uso. |

**Figura 9**

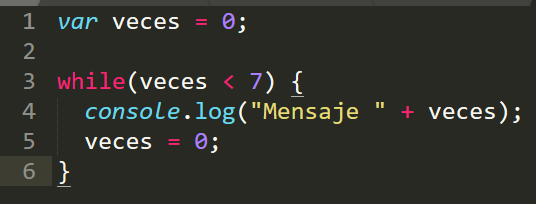
*Estructura de control if*



| Cuadro de texto |
| --- |
| La imagen muestra cómo se aplica un ***if*** de acuerdo con el valor de la variable mensaje; si su valor es ***true*** se muestra el mensaje “Hola Mundo” y de lo contrario se muestra el mensaje “Error”.  Para continuar, en JavaScript existen diferentes tipos de bucles que se pueden programar, pero, en esta ocasión, se abordarán solo dos, el ciclo mientras y el ciclo para.  La estructura ***while*** o mientras, es un ciclo que permite repetir las acciones varias veces, hasta que una condición se cumpla; si la condición se deja de cumplir, el ciclo termina.  Se debe tener muy claro que los dos ciclos el ***for*** y el ***while*** son repetitivos, pero el ***for***, por ejemplo, se usa cuando se conoce cuántas veces se va a repetir las acciones que estarán dentro de él, mientras que, en el ***while***, no se conoce cuántas veces se va a repetir, sino que se repetirá mientras una condición se siga cumpliendo. |

**Figura 10**

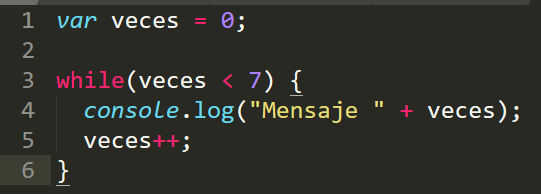
*Estructura while infinita*



| Cuadro de texto |
| --- |
| Cuando se trabaja el ciclo *while*, se debe tener mucho cuidado que no se haga un ciclo infinito; la imagen anterior muestra cómo el ciclo *while* nunca termina, porque no tiene nada dentro de él que haga que, en algún momento, se rompa la condición (se deje de cumplir).  Ahora, la forma correcta de hacer el ciclo anterior es la siguiente: |

**Figura 11**

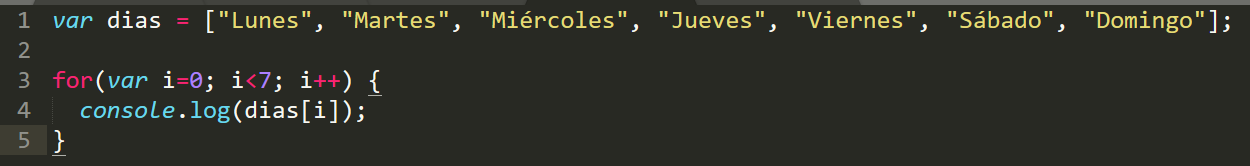
*Estructura while correcta*



| Cuadro de texto |
| --- |
| Como se puede observar, el código anterior en la línea 5 hace que la variable veces aumente de uno en uno, cada vez que da un giro el ciclo y, así, en algún momento, llegará a no cumplir la condición de veces < 7 y, en ese momento, el ciclo termina.  El siguiente ciclo es el *for*. Este ciclo es repetitivo, pero se conoce cuántas veces se va a repetir las acciones que están dentro de él, 5 veces, 6 veces, 15 veces o N veces si el usuario es el encargado de determinar las veces que se repetirá.  **for(inicializacion; condicion; actualizacion) {**  **...**  **}**  La sintaxis del *for* es la anterior; en el paréntesis se configura y tiene tres parámetros, la inicialización que indica la variable de control en qué valor inicia; normalmente, la variable de control es **i** o **j**, e inicia en cero. El segundo parámetro es la condición, es decir, hasta donde va ir la variable y el tercer parámetro es la actualización o incremento, cada giro del ciclo cuánto va a incrementar la variable de control.  **for (var i = 0; i < 5; i++) {**  **console.log(mensaje);**  **}**  La sintaxis completa es la anterior, donde se ve que se declara una variable i = 0 como variable inicializadora; el segundo parámetro indica la condición, que la **i** sea menor a 5, apenas se incumpla esa condición. |

**Figura 12**

*Estructura ciclo for*



| Cuadro de texto |
| --- |
| La imagen anterior muestra primero, cómo se declara y se llena un ***array*** unidimensional con los días de la semana y, en seguida, se crea un ciclo ***for*** para recorrer ese ***array*** e ir mostrando en consola, el día de la semana correspondiente a cada posición del vector.  El *for* termina cuando se deje de cumplir la condición de i menor a 7 y se puede ver que, en cada giro, el *for* aumenta la i en 1. |

**5. Experiencia de usuario**

| Cuadro de texto |
| --- |
| La experiencia de usuario es un tema muy importante en el desarrollo de *software*, porque permite definir cómo se sentirá el usuario mientras navega en el sistema o aplicación web. Se trata de hacerle vivir una experiencia agradable para que quiera seguir usando la aplicación, que la pueda usar de una forma sencilla y, sobre todo, que cumpla con las funciones pactadas o por las cuales se creó. |

| **Tipo de recurso** | | Pestañas o tabs Verticales |
| --- | --- | --- |
| **Introducción** | | Según la norma ISO 25000, se conoce por usabilidad, la capacidad del producto *software* para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones. Esta característica se subdivide, a su vez, en las siguientes subcaracterísticas: |
| **Imagen:** 225125\_i23 | | |
| **Capacidad para reconocer su adecuación** | Capacidad del producto que permite al usuario entender si el *software* es adecuado para sus necesidades. | |
| **Capacidad de aprendizaje** | Capacidad del producto que permite al usuario aprender su aplicación. | |
| **Capacidad para ser usado** | Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad. | |
| **Protección contra errores de usuario** | Capacidad del sistema para proteger a los usuarios de cometer errores. | |
| **Estética de la interfaz de usuario** | Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario. | |
| **Accesibilidad** | Capacidad del producto que permite que sea utilizado por usuarios, con determinadas características y discapacidades. | |

| Cuadro de texto |
| --- |
| La accesibilidad hace que los sitios web puedan ser utilizados por la mayor cantidad de personas posible. Tradicionalmente, se ideó para las personas con discapacidad, pero la creación de sitios accesibles también beneficia a otros grupos, como los que utilizan los dispositivos móviles o los que tienen conexiones de red lentas.  También se podría pensar en la accesibilidad como una forma de tratar a todos por igual y darles las mismas oportunidades, sin importar su capacidad o circunstancias. Tal y como es injusto excluir a alguien de un edificio, porque ande en silla de ruedas (generalmente, los edificios públicos modernos tienen rampas o ascensores), tampoco es correcto excluir a alguien de un sitio web porque tenga una discapacidad visual. Todos somos diferentes, pero todos somos humanos y, por lo tanto, tenemos los mismos derechos.  Por eso, debemos hacer las cosas accesibles. En algunos países, es obligado por la ley proporcionar sitios web accesibles, lo que puede abrir algunos mercados importantes que, de otra manera, no serían capaces de utilizar los servicios o comprar los productos.  Crear sitios web accesibles debe ser lo ideal y beneficia con lo siguiente:   * El HTML semántico, aparte de mejorar la accesibilidad, también mejora la optimización en motores de búsqueda (SEO), y la gente encuentra un sitio web con mayor facilidad. * Preocuparse por la accesibilidad demuestra buenos criterios éticos y morales, lo que mejora la imagen pública. * Mejorar la accesibilidad, también hace que el sitio sea más fácil de usar por otros grupos, como los usuarios de dispositivos móviles o los que tienen internet de baja velocidad. De hecho, todo el mundo puede beneficiarse de estas mejoras. * Además, hemos mencionado también que en algunos países está estipulado por ley. |

| Cuadro de texto |
| --- |
| Recuerde explorar los demás recursos que se encuentran disponibles en este componente formativo; para ello, diríjase al menú principal, donde encontrará la síntesis, una actividad didáctica para reforzar los conceptos estudiados, material complementario, entre otros. |

**SÍNTESIS**

| **Tipo de recurso** | Síntesis |
| --- | --- |
| Desarrollo de aplicaciones web *full stack*  Síntesis: construcción de la interfaz de usuario del *software* | |
| **Introducción** | El siguiente mapa integra los criterios y especificidades de los conocimientos expuestos en el presente componente formativo. |
|  | |

**ACTIVIDAD DIDÁCTICA**

| **Tipo de recurso** | Actividad didáctica. Opción múltiple | | |
| --- | --- | --- | --- |
| Apreciado aprendiz, a continuación encontrará una serie de preguntas que deberá resolver, con el objetivo de evaluar la aprehensión de los conocimientos expuestos en este componente formativo. | | | Concepto de diseño web, desarrollo de páginas web. El proceso de trabajo. Página de inicio de plantilla para el sitio web. Desarrollo web, optimización, experiencia de usuario. Elementos de diseño del sitio web, foto, transferencia de datos.  **Imagen 1:** 225125\_i25 |
| 1. Cuando se está creando una página web, se debe identificar en su código, cuál es la etiqueta correcta que se usaría para un documento HTML5. | | |  |
| <!doctype html5> | | | <html5> |
| <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 5.0 Transitional//EN"> | | | <!doctype html> |
| Retroalimentación positiva:  ¡Muy bien! Ha elegido la respuesta correcta.  Retroalimentación negativa:  ¡Incorrecto! Es necesario revisar nuevamente el tema sobre construcción de interfaces. | | | |
| 2 En la estructuración de una página web, las etiquetas HTML5 definen varios sectores; entre esos, donde están los enlaces de navegación; cómo se llama esa etiqueta: | | |  |
| <section> | | | <menu> |
| <header> | | | <nav> |
| Retroalimentación positiva:  ¡Muy bien! Ha elegido la respuesta correcta.  Retroalimentación negativa:  ¡Incorrecto! Es necesario revisar nuevamente el tema sobre construcción de interfaces. | | | |
| 3 El código JavaScript son pedazos de código que ejecutan ciertas acciones y permiten mejor interacción con el usuario. ¿Cuál de los siguientes usos del elemento <script> le parece el más apropiado? | | |  |
| <script src="archivo.js"></script> | | | <script src="archivo.js" /> |
| <script type="text/javascript" src="archivo.js"></script> | | | <sc></sc> |
| Retroalimentación positiva:  ¡Muy bien! Ha elegido la respuesta correcta.  Retroalimentación negativa:  ¡Incorrecto! Es necesario revisar nuevamente el tema sobre construcción de interfaces. | | | |
| 4 Es importante conocer los elementos que componen la estructura de una página en HTML5; por eso se debe indicar, de los siguientes, cuál es un elemento inválido: | | |  |
| <meter> | | | <post> |
| <hgroup> | | | <progress> |
| Retroalimentación positiva:  ¡Muy bien! Ha elegido la respuesta correcta.  Retroalimentación negativa:  ¡Incorrecto! Es necesario revisar nuevamente el tema sobre construcción de interfaces. | | | |
| 5 El código JavaScript son pedazos de código que ejecutan ciertas acciones y permiten mejor interacción con el usuario. ¿Cuál de los siguientes scripts se utiliza generalmente para detectar capacidades de JS, HTML5 y CSS3 en el navegador? | |  | |
| HTML5 Shim | | Google Detector | |
| Modernizr | | Selectivizr | |
| Retroalimentación positiva:  ¡Muy bien! Ha elegido la respuesta correcta.  Retroalimentación negativa:  ¡Incorrecto! Es necesario revisar nuevamente el tema sobre construcción de interfaces. | | | |
| 6 Los *input* son elementos que permiten el ingreso de datos por parte del usuario y también conforman los elementos de un formulario. Seleccione el tipo de *input* válido. | |  | |
| type=»boolean» | | type=»textarea» | |
| type=»alphanumeric» | | type=»range» | |
| Retroalimentación positiva:  ¡Muy bien! Ha elegido la respuesta correcta.  Retroalimentación negativa:  ¡Incorrecto! Es necesario revisar nuevamente el tema sobre construcción de interfaces. | | | |
| 7 Si tuviera que crear un campo de búsqueda, ¿qué sería lo más apropiado? | |  | |
| <input type="find" /> | | <input type="text" search /> | |
| <input type="search" /> | | <input type=”Select”/> | |
| Retroalimentación positiva:  ¡Muy bien! Ha elegido la respuesta correcta.  Retroalimentación negativa:  ¡Incorrecto! Es necesario revisar nuevamente el tema sobre construcción de interfaces. | | | |
| 8. ¿Cuál de los siguientes atributos permite mostrar un valor, por defecto, en un elemento ***input***, pero que se borra cuándo hacemos foco en el mismo? | |  | |
| Content | | Placeholder | |
| Source | | value | |
| Retroalimentación positiva:  ¡Muy bien! Ha elegido la respuesta correcta.  Retroalimentación negativa:  ¡Incorrecto! Es necesario revisar nuevamente el tema sobre construcción de interfaces. | | | |
| 9. Si estuviera maquetando un blog, ¿en cuál de los siguientes elementos pondría los enlaces a archivo, categorías, íconos de redes sociales, enlaces a artículos más populares, etc.? | |  | |
| <aside> | | <section> | |
| <summary> | | <footer> | |
| Retroalimentación positiva:  ¡Muy bien! Ha elegido la respuesta correcta.  Retroalimentación negativa:  ¡Incorrecto! Es necesario revisar nuevamente el tema sobre construcción de interfaces. | | | |
| 10 Para que un ***input*** tenga el foco apenas se termine de cargar el documento, ¿cuál de las siguientes soluciones aplicaría? | |  | |
| Utilizar el atributo booleano autofocus | | Utilizar el atributo autofocus=»true» | |
| Utilizar el atributo placeholder | | Utilizar el atributo required | |
| Retroalimentación positiva:  ¡Muy bien! Ha elegido la respuesta correcta.  Retroalimentación negativa:  ¡Incorrecto! Es necesario revisar nuevamente el tema sobre construcción de interfaces. | | | |

**MATERIAL COMPLEMENTARIO**

| Tipo de recurso | Material complementario | | |
| --- | --- | --- | --- |
| Tema | Referencia APA del material | tipo | Enlace |
| Lenguajes de programación | EDteam. (2020). *Tipos de lenguajes de programación* [video]. YouTube. | Video | <https://www.youtube.com/watch?v=gFMMmi-EYEM> |
| HTML | Fazt. (2018). *Curso HTML para principiantes* [video]. YouTube. | Video | <https://www.youtube.com/watch?v=rbuYtrNUxg4> |
| CSS | HolaMundo. (2021). *¡Aprende CSS ahora! Curso completo GRATIS desde cero* [video]. YouTube. | Video | <https://www.youtube.com/watch?v=wZniZEbPAzk> |
| JavaScript | Fazt. (2018). *Curso Javascript para Principiantes* [video]. YouTube. | Video | <https://www.youtube.com/watch?v=RqQ1d1qEWlE> |

**GLOSARIO**

| **Tipo de recurso** | Glosario |
| --- | --- |
| Block: | indica que los elementos estarán organizados en una línea vertical uno abajo del otro, es decir, de esa forma los interpreta el navegador. |
| CSS: | son las siglas que, en español, significa hoja de estilos en cascada y complementa las páginas web, dando estilo y forma. |
| DOCTYPE: | es la etiqueta que se coloca al inicio de una página HTML para decirle al navegador cómo va a interpretar la página siendo HTML5. |
| Inline: | indica que los elementos estarán organizados en una línea horizontal uno al lado del otro, es decir, de esa forma los interpreta el navegador. |
| Parámetro: | especifica el tipo de dato que recibirá un método. |
| PHP: | es un lenguaje de programación del lado del servidor. que permite la comunicación de la web con servidores de datos o persistencia. |
| React: | está escrito en código abierto en JavaScript, muy útil para construir las interfaces de usuario. |
| Responsive: | propiedad que indica que la página se adapta a todas las pantallas de los diferentes dispositivos donde se revise. |
| Script: | es un pedazo de código JavaScript creado para ejecutar cierta función o procedimiento. |
| Selector: | es la forma como se selecciona o indica a cuál elemento HTML se le va a aplicar las reglas CSS escritas. |

**REFERENCIAS BIBLIOGRÁFICAS:**

| **Tipo de recurso** | Bibliografía |
| --- | --- |
| Campaña Haro, M. N., y Martillo Vásquez, E. L. (2020). *Análisis, estudio y prototipo de la aplicación de experiencia de usuario (UX) en el desarrollo de la interfaz gráfica de usuario de la plataforma e integración con componentes del sistema del proyecto FCI temonet fase II de la Universidad de Guayaquil* (Proyecto de titulación, Universidad de Guayaquil, Guayaquil). Repositorio UG. <http://repositorio.ug.edu.ec/bitstream/redug/52257/1/B-CINT-PTG-N.%20612%20Campa%C3%B1a%20Haro%20Michelle%20Nydia%20.%20Martillo%20V%C3%A1squez%20Estefania%20%20Lisset.pdf> | |
| Guapi Auquilla, M. J. (2018). *Diseño metodológico para el desarrollo de interfaces gráficas en páginas web utilizando los lenguajes HTML5 y CSS3.* <http://dspace.unach.edu.ec/bitstream/51000/5163/1/UNACH-FCEHT-DS%c3%91-GRF-2018-000018.pdf> | |
| Llerena Ocaña, L. A., Fernández Villacres, G. E., Viscaino Naranjo, F. A., y Baño Naranjo, F. P. (2021). Frameworks basados en typescript para el desarrollo de aplicaciones web interactivas. *Dilemas contemporáneos: educación, política y valores*, *8*(3). <https://dilemascontemporaneoseducacionpoliticayvalores.com/index.php/dilemas/article/view/2644/2675> | |