**FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

| **PROGRAMA DE FORMACIÓN** | Técnico en Procesamiento de Pruebas de Software |
| --- | --- |

| **COMPETENCIA** | 220591123 - Construir sitios web según técnicas de interoperabilidad y protocolos técnicos | **RESULTADOS DE APRENDIZAJE** | 220591123-01 - Caracterizar la estructura de una aplicación, teniendo en cuenta el entorno de desarrollo |
| --- | --- | --- | --- |

| **NÚMERO DEL COMPONENTE FORMATIVO** | CF01 |
| --- | --- |
| **NOMBRE DEL COMPONENTE FORMATIVO** | Estructura de una aplicación |
| **BREVE DESCRIPCIÓN** | En este módulo el aprendiz comprenderá los conceptos fundamentales del diseño y arquitectura de *software*, cuáles son los tipos de arquitectura de una aplicación web, identificar sus componentes, protocolos, herramientas, entornos y lenguajes de programación. Será capaz de identificar las diferencias entre los diversos tipos de proyectos en el desarrollo de *software* y tendrá la capacidad de interpretar diagramas de diseño y arquitectura de *software*. |
| **PALABRAS CLAVE** | Arquitectura web, Diseño, Navegación, CRM, herramientas, SEO |

| **ÁREA OCUPACIONAL** | 2 - CIENCIAS NATURALES, APLICADAS Y RELACIONADAS |
| --- | --- |
| **IDIOMA** | Español |

1. **TABLA DE CONTENIDOS:**

**Introducción**

1. **Arquitectura de *software***
   1. Tipos de arquitectura de aplicación
   2. Patrones de diseño
2. **Arquitectura web**
   1. Conceptos y características
   2. Tipos de arquitectura Web
   3. Mapas de navegación del sitio web
   4. Procedimientos para diseñar un sitio web
   5. Lenguajes de programación, frameworks, IDE’s y plugins usados en el desarrollo web
3. **Comunicación entre componentes de un sitio web**
   1. Técnicas y protocolos de comunicación
   2. Características del diseño web
4. **DESARROLLO DE CONTENIDOS**

**Introducción**

Le invitamos es a revisar el siguiente video para que usted comprenda algunas indicaciones iniciales del proceso de aprendizaje y los elementos de abordaje acerca de la caracterización de la estructura de una aplicación, teniendo en cuenta el entorno de desarrollo:

Interfaz de usuario gráfica, Aplicación, PowerPoint

Descripción generada automáticamente

1. **Arquitectura de *software***

Juega un papel muy importante en el proceso del desarrollo de *software*, ya que desde allí se definen la estructura, componentes, comportamiento e infraestructura requerida para la construcción del *software*, impactando la capacidad para satisfacer los atributos de calidad del sistema; entre estos están: el desempeño, tiempos de respuesta, usabilidad, accesibilidad, escalabilidad, mantenimiento y cambios de nuevas funcionalidades.

Los atributos de calidad son parte de los requisitos del sistema no funcionales y son características que se deben expresar de forma cuantitativa. No tienen sentido si se expresan de forma subjetiva, dado que esto no va a ser posteriormente evaluable; por ejemplo, decir que el sistema al recibir una petición “debe ser rápido” o que una página “debe ser ligera al cargar”, esto no se podría evaluar objetivamente. Por lo cual, podríamos decir que, si en un sistema estructurado la información debe pasar por muchos componentes antes de devolver el resultado, esto incurre en aumentar la latencia y su desempeño puede llegar a ser pobre. Otro ejemplo es que si los componentes de un *software* presentan un alto acoplamiento y una baja cohesión entre ellos se limita altamente la modificabilidad.

El definir la estructura de un sistema de información, representa un impacto menor ante los requisitos funcionales por el sistema. Un producto *software* que contiene una estructura en espagueti puede ser totalmente funcional, dado que da respuestas a las necesidades del cliente, pero puede ser un caos su mantenimiento dado que sus componentes no contendrán patrones fáciles de identificar para la realización de cambios o mejoras.

Las propiedades de calidad y la arquitectura son fundamentales para orientar el desarrollo de un producto *software*, pues a su vez permiten establecer un cronograma acertado de las tareas a implementar por el equipo de desarrollo, los diseños arquitectónicos que se definen para un proyecto en específico se pueden reutilizar, sin importar el *core* de negocio al que pertenezcan; en otras palabras no importa si el sistema es diferente, si inicialmente era un sistema bancario y el otro una plataforma de *streaming*, la arquitectura se podrá aprovechar en los diferentes modelos de negocio.

**Figura 1**

*Representación de la arquitectura por capas*



Nota. Adaptada de Programadorclic (s.f.)

En la anterior imagen se puede observar el ejemplo de una arquitectura con 5 capas, en la cual cada capa tiene una finalidad específica en la aplicación, las cuales se describen a continuación:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

**1.1 Tipos de arquitectura de aplicación**

Es necesaria al decidir la arquitectura que se debe usar en el desarrollo de un sistema de información, aplicación o sitio web, permitiendo evaluar y definir los objetivos estratégicos, esto ayudará que la arquitectura a diseñar respalde la finalidad del proyecto y que los componentes encajen adecuadamente.

Entre las arquitecturas más conocidas y empleadas para el desarrollo de sitios, aplicaciones o sistemas de información web están las siguientes:

**Arquitectura en capas y N-tier:**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

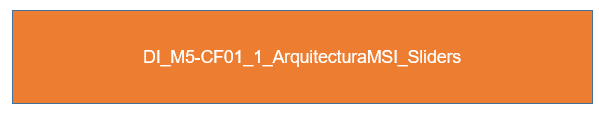
**Arquitectura monolítica:**



**Arquitectura orientada a servicios (SOA):**



**Arquitectura de microservicios (MSI):**



El objetivo de implementar este tipo de arquitecturas es distribuir el *software* de calidad con mayor rapidez, dado que se facilita el proceso de crear diversos servicios al mismo tiempo y desplegarlos, sin la necesidad de tener que construir una sola aplicación monolítica inmensa que al final puede llegar a ser insostenible. Gracias a esto cada desarrollador puede trabajar en sus servicios o funcionalidades al mismo tiempo sin afectar toda la aplicación y permitiendo de esta manera lanzar nuevas características de forma frecuente.

Es importante destacar la diferencia entre las arquitecturas tradicionales y las modernas, de las cuales nos encontramos con la monolítica y la orientada a microservicios. En la siguiente figura, se puede observar que mientras el primer apartado gráfico indica que todos los componentes de *software* están inmersos en un solo componente y realizar un cambio suele ser un proceso crítico que se realiza 1 a 2 veces por año, la segunda parte presenta la segregación y distribución de funcionalidades de tal manera que corregir o crear nuevas funcionalidades es sencillo sin repercutir en generar caos al desplegar una nueva funcionalidad o mejora, esto permite que haya un despliegue continuo. Para describir un poco mejor las diferencias entre las dos vertientes se presenta la siguiente figura:

**Figura 2**

*Arquitectura monolítica vs arquitectura orientada a microservicios*

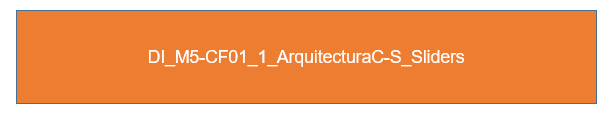
****

Nota. Adaptado de Decide Soluciones (2019).

**Arquitectura orientada a eventos (EDA):**



**Arquitectura cliente-servidor:**



**1.2 Patrones de diseño**

Los patrones son procesos, métodos y técnicas comunes que se utilizan en el proceso de desarrollo de *software* entre otros elementos que están directamente relacionados con el diseño de interfaces y soluciones desde el diseño. Para que una solución sea considerada como patrón deben cumplir diferentes particularidades, una de ellas es poderse aplicar afectivamente en otras soluciones, en pocas palabras debe ser reutilizable.

Los patrones de diseño buscan lo siguiente:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Los patrones de diseño no pretenden:

* Imponer una alternativa de solución ante otra.
* Eliminar o limitar la creatividad inherente ante el proceso de diseño de soluciones.

El usar patrones de diseño no es obligatorio, pero es aconsejable, dado ayuda diseños cumplieron con el objetivo de solucionar algún problema similar; sin embargo, en casos particulares los patrones no pueden ser aplicables y forzar o abusar del uso de patrones puede ser un error.

Según la escala y el nivel de abstracción podemos encontrar los siguientes tipos:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. **Arquitectura web**

Es la estructura funcional de un sitio web, creando un mapa jerárquico de todas las páginas y la relación entre ellas, de una forma lógica y coherente.

La estructura de un sitio webcuenta con varias páginas y se distribuyen en forma de mapa, agrupando de forma lógica y jerárquica las páginas que comparten un mismo criterio para facilidad de navegación de los usuarios o clientes.

En el siguiente recurso de aprendizaje se presentan los conceptos y representación de la estructura de navegación:

Interfaz de usuario gráfica, Aplicación, PowerPoint

Descripción generada automáticamente

**2.1. Conceptos y características**

La arquitectura web es el ordenamiento y jerarquización del contenido de un sitio web, en el cual se trabaja bajo los estándares de los sitios web y las categorías que conforman la optimización y cara al posicionamiento en los buscadores, interacción y experiencia de usuarios (UX). El objetivo es resolver las necesidades del cliente, que posiblemente necesite o busque un producto *software* *online* y se dé respuesta por medio de una implementación web.

| Algunos de los conceptos base para desglosar la arquitectura web son la taxonomía web y la ontología web, donde la primera hace referencia a los diferentes términos que dividen un sitio web y la segunda nos habla de las relaciones entre los diferentes ítems de navegación. | SEO: Qué es la taxonomía de sitios web o taxonomia de Url – SpoqTech News |
| --- | --- |

En la arquitectura web también se incluye el diseño de interfaz de usuario, el diseño de la aplicación y la estructura de la información, con lo cual hay un enfoque del aspecto visual, colores, tamaños de letra, estructura de las diferentes secciones, las funcionalidades segregadas en cada una de las vistas del sitio web y por último, se determina la información que debe existir en cada una de las secciones del sitio web y que se verá reflejado en un ambiente productivo.

* **Arquitectura de aplicación o arquitectura de *software*:** en la arquitectura de aplicación se pueden encontrar métodos, técnicas, diseños y patrones para desarrollar un producto *software*, la cual provee una guía y una serie de prácticas recomendadas para construir un producto *software*, de tal modo que se obtenga una solución bien estructurada y siguiendo estándares.

| En un producto *software* se puede encontrar diferentes tipos de soluciones, como por ejemplo una solución front-end o una solución back-end. | Qué es el Desarrollo Front end, Back end y Fullstack? | by Ken Ruiz Inoue |  Deuk | Medium |
| --- | --- |

Interfaz de usuario gráfica, Aplicación, PowerPoint

Descripción generada automáticamente

La arquitectura es un punto de partida que permite tomar decisiones de desarrollo, como por ejemplo el tipo de estructura, el lenguaje de programación y la infraestructura que se deberá tener en cuenta para su despliegue. En la actualidad se cuenta con diferentes lenguajes de programación, entre los cuales encontramos Swift, Kotlin, JavaScript, Java, Python, Php, entre otros, los cuales son usados para un tipo de programación en específico.

* **Desarrollo Frontend:** se puede traducir como frente o fachada final, esta vertiente del desarrollo de *software* es una especialidad para el desarrollo web y concentra lo que se define como interfaz de usuario (GUI - Graphical User Interface), esta es la parte del sitio web con la cual los usuarios interactúan y este se despliega del lado del cliente; está orientado al lenguaje de marcas o etiquetas, todo lo visual de un sitio web, tamaños, colores, formas, formatos y renderizado de información al usuario.

Entre las tecnologías que son usadas del lado Front-end podemos indicar las siguientes:

* Html5 / Html / XHTML
* XML, SVG
* JavaScript, Json, Canvas
* Css3
* **Desarrollo Backend:** está enfocado a lenguajes de programación y hace referencia a la lógica interna del sistema, se le conoce también como el desarrollo del lado del servidor, dado que los componentes están en un espacio servidor que normalmente está conectado al internet (nube) o algún tipo de red que permita acceder a sus funcionalidades.

Entre las tecnologías que son usadas del lado Back-end podemos indicar las siguientes:

* Servidores de aplicaciones como Apache Friends, Apache Tomcat, Glassfish, Spark, Heroku.
* Lenguajes como PHP, Java, JavaScript, Python.

El desarrollo Front-end y Back-end se complementan y convergen, un Front-end sin Back-end no tendría una utilidad significativa en el ámbito del desarrollo de aplicaciones, y un Back-end sin un Front-end estaría incompleto dado que no tendríamos un interfaz de comunicación que permita ejecutar o desencadenar las acciones relacionadas a la lógica de negocio. En la siguiente figura se presenta la relación entre estas.

**Figura 3**

*Sinergia entre Front-end y Back-end*



Nota. Adaptación propia

**2.2. Tipos de arquitectura web**

Los diferentes tipos de arquitecturas web se pueden catalogar dada su profundidad, cantidad de interacciones con el usuario al dar clic, como las siguientes:

* **Arquitectura vertical:** son aquellos sitios web que emplean una gran cantidad de categorías, para organizar la información, dejando una gran distancia entre el home y la última página posible, este distanciamiento se mide en clics realizados por el usuario al momento de efectuar la navegación en el sitio. Un ejemplo claro de este tipo de arquitectura se puede presentar las páginas web de las tiendas *online*, en la siguiente grafica puede visualizar este tipo de arquitectura.

**Figura 4**

*Arquitectura vertical*



Nota. Adaptado de Idital (2020)

* **Arquitectura horizontal:** la arquitectura web horizontal o plana, es aquella que tiene la posibilidad de acceder a todo el contenido del sitio web, con uno o dos clics nada más; se trata del método de distribución de información ideal, además los bots (programa informático que actúa como un robot) podrán leer los patrones presentes el sitio web más rápido y esto resulta muy positivo. En la siguiente gráfica se puede visualizar este tipo de arquitectura:

**Figura 5**

*Estructura horizontal*



Nota. Adaptado de Idital (2020).

Para este tipo de arquitectura se debe aplicar la regla de los tres clics, basada en la forma que los usuarios utilizan las páginas web, lo que permite ayudar al desarrollador sitios webs más intuitivos y con estructuras lógicas jerárquicas; aplicar esta regla se considera cómo una buena práctica en el diseño web.

**2.3. Mapas de navegación del sitio web**

El mapa de navegación es un recurso, esquema o elemento gráfico por lo general en forma de árbol, que representa la estructura o arquitectura del sistema. Este tipo de recursos es usado para poder optimizar los diseños y tener una adecuada estructuración de los contenidos, previamente a desarrollar los contenidos de la aplicación web buscando que el usuario siempre acceda de forma sencilla a lo que está buscando. Sabiendo esto, ahora podrás conocer información relevante sobre los mapas de navegación:

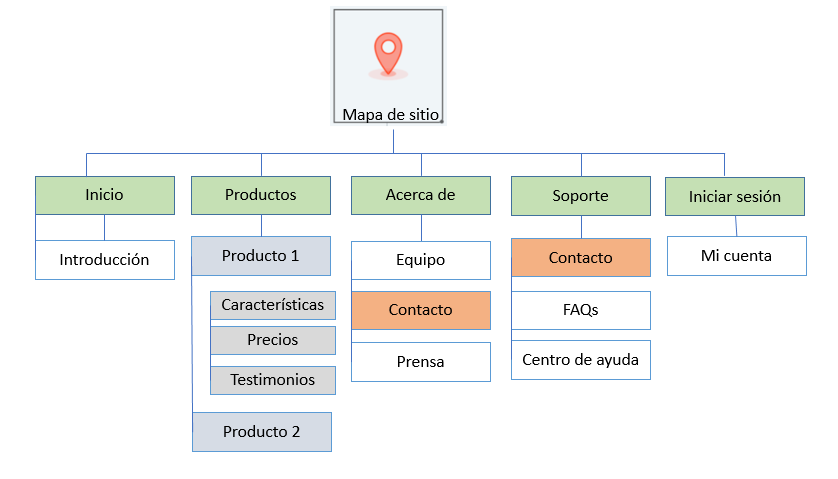
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

A continuación, se presenta un ejemplo grafico de un mapa de navegación con la herramienta Mindmeister:

**Figura 6**

*Mapa de navegación*



Nota. Adaptado de Eniun (s.f)

**2.4. Procedimientos para diseñar un sitio web**

En el proceso del diseño y desarrollo de un sitio web, es importante tener presente los siguientes aspectos:

* **Realizar una planificación adecuada:** es importante tener claro cuál es el objetivo y que se quiere construir antes de iniciar. Un producto software puede tener muchas utilidades y según las especificaciones de los requisitos de *software* habrá una u otra estructura que se acople mejor a la necesidad. Teniendo en cuenta características como el objetivo (target), la información a presentar, las palabras clave, formulario de acceso, recursos multimedia, gama de colores, tipografía, entre otros.

Algunas categorizaciones:

* Aplicaciones web
* Landing page
* E-Commerce
* Página web de servicios
* Porfolios / Pagina web personal
* **Establece el dominio y selecciona el hosting:** sin estos dos elementos es muy poco probable que tu producto *software* web pueda estar en línea, dado que el hosting es el espacio donde se aloja el producto y todos los recursos que lo conforman, en pocas palabras es el espacio en el cual está desplegado el producto. Dependiendo de la complejidad del proyecto se podrá escoger un hosting básico u otro más complejo que dé respuesta a las necesidades de infraestructura.

Mientras, el dominio es el nombre que asignamos a la web este también es conocido como DNS (*Domain Name Server*) por sus siglas en inglés y su función consiste en traducir la dirección IP para encontrarla fácilmente en internet. Es importante que el nombre del dominio sea una insignia del sitio web, dado que será un identificador para que los usuarios tengan mayor recordación. Algunos ejemplos de dominio son ‘google.com’, ‘facebook.com’, ‘gmail.com’, ‘instagram.com’, ‘gov.com’, entre otros.

* **Elige el lenguaje de programación para el desarrollo de tu sitio web:** si tu sitio web es un sistema de información a la medida, lo más probable es que el desarrollo se realice desde cero, por eso es importante identificar cual es la arquitectura de aplicación más apropiada para la solución, identificar los pro y contra del uso de uno u otro lenguaje y de ser necesario explorar un framework que se adapte a la necesidad.

Después de realizar la identificación de la arquitectura de aplicación, se debe seleccionar un lenguaje de programación, explorar los posibles frameworks, comenzar la instalación del entorno de desarrollo, lo que incluye instalar los sdks (Software Development Kit) necesarios, instalar un servidor de aplicaciones, seleccionar un motor de base de datos para almacenar la información requerida, instalar el IDE y *plugins*, configurar una herramienta el control de las versiones de código y comenzar a codificar la solución.

Si se identifica que la necesidad es genérica y que en el mercado se cuenta con herramientas que permitan construir un sitio web por medio de un CMS (Content Management System), los pasos descritos previamente no serán necesarios y el enfoque estará en escoger bien qué componentes usar para iniciar el sitio web.

* **Elige el CMS idóneo para tu página web y realiza la instalación:** estas herramientas de diseño “en tres clics”, se encuentran una serie de gestores de contenidos que permiten crear páginas webs desde cero con buenos resultados, entre ellos se puede destacar WordPress como una de las mejores opciones para crear una página web.

Drupal y magneto son CMS, reconocidos y potentes con los cuales se pueden crear sitios web. Otro ejemplo son Prestashop o Shopify que están enfocados en la creación de tiendas *online*. Un CMS bastante usado es WordPress, ya que es un gestor sencillo al momento de implementar los componentes web con una amplia documentación y comunidad.

Un valor agregado de los CMS, es que cuentan con una comunidad de desarrolladores que construyen plugins y componentes para diferentes necesidades, con lo cual el propietario del sitio solo es ir al panel de control y añadir el componente que desea para comenzar a agregar nuevas funcionalidades a su sitio web.

* **Establece la estructura web:** en este apartado se hace referencia a la estructuración, distribución y organización de los elementos que compondrán la estructura visual del componente web, de la cual se crea la diagramación del sitio y se establece como verá la información el usuario, también se incluye la relación de las diferentes páginas entre sí, cuáles serán las superiores, cuales tendrán dependencias de otras, cuáles serán públicas, cuáles requerirán de un inicio de sesión, cuáles tendrán su representación en la parte superior (header) o en la parte inferior (footer). Al hablar de diseño web, no solo se hace referencia a la parte visual, dado que la arquitectura interna también es importante y dentro de esa arquitectura se puede tener una arquitectura específica por el front-end, por el back-end y una genérica de sistema.
* **Define un diseño para el sitio web:** en este punto es requerido definir varios elementos que están relacionados con las plantillas web, es importante tener como referencia el tipo de información que se refleja y el tipo de usuario que tendrá acceso a la información. Imagina que estás diseñando un producto *software* para una escuela donde se orientan clases a niños de primero hasta quinto de primaria, donde las edades oscilan entre los 7 a 12 años, el requisito es que se construya una aplicación web que por medio de juegos y se pueda tener un recurso didáctico adicional.

Por lo cual jugar con colores llamativos, letra grande, tipografía animada, muchas imágenes hará que los niños olviden que están estudiando y se enfoquen en aprender jugando, pero si al contrario viras ese diseño a sitios empresariales orientado a personas mayores, posiblemente no se le sacará el mayor provecho.

**2.5. Lenguajes de programación, frameworks, IDE’s y plugins usados en el desarrollo web**

Un lenguaje de programación es un lenguaje formal, en pocas palabras es un lenguaje con reglas gramaticales bien definidas, que proporciona a una persona la capacidad de escribir una serie de instrucciones (programar) con el fin de controlar el comportamiento de un componente informático, tales como: los computadores, smartphones, microcontroladores, Smart Tv, entre otros.

En la actualidad, hay una serie de lenguajes que se usan en diferentes frentes o áreas de la programación; por ejemplo, para desarrollar front-end existe JavaScript en conjunto con HTML5 y Css3, para el desarrollo Mobile dependiendo si es iOS se usa Swift, y en su contraparte Android se usa Java o Kotlin, en el desarrollo back-end podemos encontrar lenguajes como Java, PHP, Python, C#, C++.

Por otro lado, los Frameworks (Marcos de trabajo) son una serie de librerías, paquetes y clases, que buscan optimizar el proceso de desarrollo, por ejemplo, en JavaScript podemos encontrar a Angular, VueJs, ReactJs, en PHP encontramos a Yii, Cake, Laravel, Java a Spring MVC, Spring Boot, PrimeFaces. Estos fueron diseñados con la finalidad de implementar patrones identificados y maximizar el proceso de configuración y desarrollo inicial de los productos software, de tal manera que los equipos de trabajo se enfocarán explícitamente en la lógica de negocio relacionada al problema.

A continuación, se presentan los lenguajes de programación más reconocidos con sus respectivos Frameworks:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Por su parte,un IDE (Integrated Developer Environment) o entorno de desarrollo integrado, es una aplicación especial diseñada para programar, la cual cuenta con una interfaz gráfica (GUI), con un editor de texto que permite escribir las sentencias u órdenes en el lenguaje de programación seleccionado para construir el producto software. Este tipo de herramientas vienen con una serie de pasos de configuración para que al realizar el proceso de programación, compilación y ejecución todo esté acorde y se pueda probar y desplegar el producto de manera local (en la máquina del desarrollador). Sabiendo esto, las siguientes son algunas de las características de los IDE:

* **Editor de código fuente:** es la herramienta que permite al desarrollador escribir el código fuente del producto *software*, esta herramienta por lo general contiene indicadores visuales, que ayudan a identificar los errores de sintaxis, autocompletado, comprobación de errores a la medida que se van escribiendo las sentencias. En esta herramienta se hace plena validación de las reglas semánticas del lenguaje en el cual se está construyendo el producto *software*.
* **Automatización de compilaciones locales:** herramienta que sirve para automatizar tareas sencillas y repetitivas como parte de la comisión local, esto permite crear el código binario, ejecución de prueba automatizadas, verificación de cobertura, verificación de estilos en la programación para el código fuente.
* **Depurador:** es la herramienta que sirve para probar los programas, en el cual se realiza una inspección y se ubican los errores en el código de forma visual o en un reporte en texto plano, esto es de mucha utilidad para los desarrolladores ya que permite encontrar problemas rápidamente y dar solución.

Estos son algunos de los IDE’s más empleados en el ámbito del desarrollo web:

* IntelliJ Idea
* Visual Studio Code / Visual Studio
* Sublime Text 3
* Atom
* Netbeans
* Notepad++
* Eclipse

A su vez, los *plugins* son componentes que permiten habilitar nuevas características a los IDE y entornos de desarrollo. Utilizar *plugins* implica asumir cierto riesgo en la seguridad, ya que los hackers pueden usar estas ampliaciones como puerto de entrada para un ataque en el computador del desarrollador, esto se puede dar, dado que los *plugins* son parches de *software* desarrollados por agentes externos al propietario del IDE, por lo cual podría contener en su código fuente un malware que se use con fines adversos al inicialmente definido por el entorno de desarrollo. Tener en cuenta que cada IDE tiene su propia línea de plugins, con lo cual el desarrollador dispondrá de una amplia gama de opciones para definir cuál será la mejor herramienta para poder desempeñar sus funciones.

Estos son algunos plugins:

| **PhpStorm**, **Plugins:** Material Theme UI for Jetbrains, IntelliJ IDEA / PhpStorm Laravel Plugin, Php Inspections (EA Extended), IntelliJ Bash Plugin. |  |
| --- | --- |
| **Sublime Text 3,** **Plugins:** Seti UI, Emmet, Auto Filename, SFTP, LiveReload, ColorPicker, Alignment, BracketHighlighter, Advanced New File. | SUBLIME TEXT 3 COMPLETO – My Web Hobby |
| **Atom, plugins:** file-icons, minimap, pigments, color-picker, highlight-selected, goto-definition, git-plus, git-time-machine, merge-conflicts, todo-show. | Atom y sus mejores plugins | Rincón de la Tecnología |
| **Coda 2, plugins:** Importador, Kara, Ficha desplegable, Microformatos, WakaTime, Blanqueamiento. | CodeKit - Coda 2 Plugin |
| **NetBeans, plugins:** easyUML, EasyPmd, JSLint, Complemento Gluon, Darcula LAF para netbeans, Barra de herramientas de Git, Sugerencias adicionales de Java. | Download NetBeans Logo in SVG Vector or PNG File Format - Logo.wine |

1. **Comunicación entre componentes de un sitio web**

En un producto *software* es posible que muchos de los componentes se desarrollen de forma independiente, de tal manera que no tengan comunicación e interacción con otros elementos del sistema de información. Por lo cual se presentan algunas técnicas y protocolos para establecer una correcta comunicación entre los diferentes componentes web y las características del diseño web.

**3.1. Técnicas y protocolos de comunicación**

A continuación, se listan los conceptos claves relacionados con técnicas y protocolos de comunicación en los sitios web:

* **Protocolos de comunicación de red:** TCP/IP (tcp, que indica protocolo de transmisión de datos e ip, protocolo de internet) y HTTP, son los protocolos encargados de establecer la comunicación y llevar a de un punto a otro paquete básicos de datos. Se consideran los más importantes dado que estos permiten establecer una conexión entre los equipos que están inmersos en la red.

HTTP, POP o SMTP, son otros protocolos usados en internet, los cuales se usan a diario, los cuales permiten navegar por la red, escuchar en línea videos o música o enviar correos electrónicos.

* **Número de participantes en la comunicación:** podemos diferenciar los protocolos por el número de ordenadores que participan de la comunicación. Por ejemplo, se denomina *unicast* si solo se tiene un destinatario en la comunicación, si se habla de dos o más se considera un *multicast*, si el paquete de datos implica a todos los participantes se considera un *broadcasting*, este último es una conexión se asemeja a la de radio o televisión.
* **Modo de transmisión de los datos:** la dirección en cómo se transmiten los datos también nos brinda criterios para poder diferenciar entre los diferentes protocolos de red, los protocolos de comunicación unilateral, o sea son del tipo tal que, solo se envía el mensaje desde un emisor y no se recibe interacción por el receptor este se denomina simplex (sx), por otro lado podemos tener el escenario donde el rol del emisor y receptor cambia, y existe una interacción por decirlo así, A envía mensaje a B, y luego B envía mensaje a A, a este tipo de interacción se le denomina semidúplex (half-duplex, hdx), teniendo en cuenta que este cambio no es en simultáneo y por último el modo dúplex completo (full-duplex, fdx), este permite el envío de datos en ambas direcciones en simultáneo.
* **Jerarquía de los participantes:** el modelo cliente-servidor es un tipo de conexión que se basa en una estructura jerárquica. Por decirlo de esta manera muchos clientes (ordenadores/aplicaciones) se conectan a un servidor, el cual recepciona, gestiona y procesa las peticiones. Este es un tipo de comunicación asimétrica, pero en su contraparte tenemos en cuenta la comunicación simétrica, la cual consiste en la comunicación de red por iguales peer to peer, en este caso todos los ordenadores están en igual disposición de proporcionar servicios y ser usados.
* **Sincronización de la comunicación:** teniendo en cuenta la sincronización de los datos entre emisor y receptor, también se puede denotar el término de comunicación asíncrona o síncrona. Todo esto en función de la conexión con los bits transmitidos entre los puntos.
* **Tipo de conexión:** también podemos encontrar una clasificación orientada a conexión o aquellos que no, por decirlo así, hay una conexión entre emisor y receptor, que durante la comunicación se intenta asegurar que los paquetes de datos lleguen al destino, verificando que los paquetes lleguen en el mismo orden emitidos, dado el caso que esto no sea así, la transmisión fallará y se intentará enviar nuevamente. Por otro lado, los segundos no establecen un orden de envío ni se interrumpe la conexión, en este caso los paquetes de datos tienen mucha menos metadata y su llegada puede ser en desorden, dado el caso no se vuelven a enviar ante una conexión errónea.

**3.2. Características del diseño web**

Los elementos de un sitio web pueden cambiar, esto de acuerdo con las necesidades de comunicación que se hayan especificado, aunque siempre contendrá unos componentes característicos que por lo general siempre están presentes en los sitios web, esto con la finalidad de tener referente de navegación y facilitar la accesibilidad y usabilidad del usuario. Sabiendo lo anterior, las siguientes son las características que debe tener un sitio web:

* **Contenidos de utilidad:** el tema y elementos establecidos en el sitio web deben estar acordes al usuario final, dado que de no ser la información coherente con el objetivo (*target*) este abandonara rápidamente el sitio.
* **Debe ser intuitivo:** debe ser intuitivo, de fácil acceso, fácil navegación, que los elementos definidos cumplan el objetivo, sin necesidad de dar mucha vuelta, además de fácil recordación, esto con la finalidad de evitar abandonos del sitio rápidamente.
* **Diseño atractivo:** la calidad de imágenes y contenido, impactaran de mejor manera que el texto. Por lo cual es de suma importancia que las imágenes, color de texto, contenido multimedia, paleta de colores, concuerden con el negocio o finalidad que se pretende plasmar, esto permitirá generar un vínculo con el usuario y se quedara por más tiempo, además de generar confianza de marca con la empresa.
* **Contenido bien estructurado:** el contenido establecido debe ser lo más simple, sencillo y claro posible, donde se haya realizado un proceso de explicación con miras al beneficio del usuario, la importancia de una estructura clara y objetiva es lo que permitirá que el usuario esté conectado y que el producto sea exitoso.
* **Importancia de la velocidad de la web:** uno de los factores más relevantes es la carga de contenido en la web, dado que las páginas que tardan mucho en renderizar tienen un alto porcentaje de abandono y los usuarios pierden la confianza en este tipo de productos. Muchas veces este tipo de problemas no dependen solamente de la red o el ancho de banda del servidor, si no, que se ven implícitamente afectados por los recursos usados para la construcción de la misma, donde se establecen recursos como imágenes, vídeos, fuentes, entre otros que no están optimizados para la web, y representan una sobrecarga en la renderización; en otros casos puede ser tema de programación, dado que puede haber algún tipo de funcionalidad o programación que impida que esta renderice rápidamente.
* **Información de contacto:** este espacio en el sitio web es muy valioso y debe estar siempre disponible, en un lugar visible para los usuarios, es en esta zona donde se deberá ubicar la información de contacto, dirección, teléfono y cualquier tipo de dato que permita establecer comunicación con el propietario o los administradores del sitio. Un ejemplo de ello es invitarlos a la acción, como por ejemplo poner *links* de **contacto con nosotros**, **solicitar más información**, etc. Si el contenido le parece interesante y le genera valor, el usuario no dudará en ponerse en contacto.

1. **ACTIVIDADES DIDÁCTICAS (OPCIONALES SI SON SUGERIDAS)**

| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA | |
| --- | --- |
| Nombre de la Actividad | Autoevaluación de conceptos básicos. |
| Objetivo de la actividad | Evaluar la comprensión de los conceptos, permitiendo que el aprendiz pueda establecer la mejor estructura para crear una página web, teniendo en cuenta su arquitectura. |
| Tipo de actividad sugerida | Interfaz de usuario gráfica  Descripción generada automáticamente |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | Anexo\_CF01\_ActividadDidactica |

1. **MATERIAL COMPLEMENTARIO:**

| Tema | Referencia APA del Material | Tipo de material  (Video, capítulo de libro, artículo, otro) | Enlace del Recurso o  Archivo del documento o material |
| --- | --- | --- | --- |
| 1. Arquitectura de software | IBM. (2006). Concepto: *Arquitectura de software.* Cgrw. | Documento | <https://cgrw01.cgr.go.cr/rup/RUP.es/SmallProjects/core.base_rup/guidances/concepts/software_architecture_4269A354.html> |
| 2. Arquitectura web | Luque, R. (2020). *Cómo crear una arquitectura web óptima para SEO*. Seoalive.com. | Artículo | <https://seoalive.com/arquitectura-web/> |
| 2.1. Conceptos y características | Publisuites. (2020). *Arquitectura web: Cómo organizar bien los contenidos de tu web*. [Video]. YouTube. | Video | <https://youtu.be/VSXdCHSFHXM> |
| 2.2. Tipos de arquitectura Web | Cuesta, D. (2019). *Arquitectura Web, Estructura vertical, horizontal y mixta*. [Video]. YouTube. | Video | <https://youtu.be/BxZFa9EK028> |
| 2.2. Tipos de arquitectura Web | Fernández, A. (2021). *Tipos de arquitectura web y eficiencia para SEO*. Blog. | Documento | <https://albertofdez.com/blog/seo-on-page/tipos-arquitectura-web> |
| 2.6. Lenguajes de programación, frameworks, IDE’s y plugins usados en el desarrollo web | Rodríguez, A. (2020). *¿Cuáles son lenguajes de programación web más usados en Internet?* Godaddy. | Documento | <https://es.godaddy.com/blog/cuales-son-lenguajes-de-programacion-web-mas-usados-en-internet> |

1. **GLOSARIO:**

| TÉRMINO | SIGNIFICADO |
| --- | --- |
| **Aplicación** | Producto *software* creado para ejecutar una acción en específico, tales como cálculo de notas, recepción de pagos en línea, gestión de información, entre otros. |
| **Arquitectura** | Estructura lógica y física de los componentes de una computadora. |
| **SEO** | De sus iniciales en inglés: Search Engine Optimization, que traduce Optimización en el motor de búsquedas, y su objetivo es adecuar el sitio web a desarrollar para ser detectado rápidamente dentro de las búsquedas realizadas, mejorando el orden de aparición. |
| ***Software*** | Conjunto de algoritmos, reglas, componentes e instrucciones informáticas, para ejecutar una tarea computacional. |
| **Patrón** | Modelo que sirve de muestra para sacar otra cosa igual. |
| **UX** | De sus siglas en inglés: User eXperience, que traducido al español sería Experiencia de Usuario, abarca todos los aspectos de la interacción de un usuario final con un producto o servicio. |
| ***Web*** | Red informática. |

1. **REFERENCIAS BIBLIOGRÁFICAS:**

Alarcón, J. M. (2019). *Los conceptos fundamentales sobre Programación Orientada Objetos explicados de manera simple*. CampusMVP. <https://www.campusmvp.es/recursos/post/los-conceptos-fundamentales-sobre-programacion-orientada-objetos-explicados-de-manera-simple.aspx>

Blancarte, O. (2018). Arquitectura Orientada a Microservicios (MSI). Blog. <https://www.oscarblancarteblog.com/2018/05/22/que-son-los-microservicios/>

Blancarte, O. (2020). *Arquitectura Monolítica*. Programming. <https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/monolitico>

CursosGratis (2021). *Arquitectura cliente-servidor: definición, tipos, ejemplos, ventajas y desventajas.* Aprendiendoaprogramar.es. <https://aprendiendoaprogramar.es/blog/arquitectura-cliente-servidor-definicion-tipos-ejemplos-ventajas-y-desventajas/>

Desarrollos Creativos. (2020). *Importancia de la Arquitectura de una página web en el SEO*. Desarrolloscreativos.net. <http://desarrolloscreativos.net/posicionamiento-web-seo/importancia-de-la-arquitectura-de-una-pagina-web-en-el-seo/>

Eniun. (s.f.). *Mapas conceptuales, mapas de navegación y sitemaps*. Eniun. <https://www.eniun.com/mapas-conceptuales-navegacion-sitemaps/>

Idital. (2020). *Arquitectura página web*. Idital. <https://idital.com/diccionario-seo/arquitectura-pagina-web/>

Programadorclic. (s.f.). *Arquitectura en capas del patrón de arquitectura de software*. Programadorclic. <https://programmerclick.com/article/338572330/>

Reynoso, C. B. (2004). Introducción a la Arquitectura de Software. *Universidad de Buenos Aires*, *33*. <http://carlosreynoso.com.ar/archivos/arquitectura/Arquitectura-software.pdf>

Tovar Casallas, A. (2016). *Arquitectura de software para la gestión del proceso de validación de núcleos familiares*. udistrital. <https://repository.udistrital.edu.co/bitstream/handle/11349/5184/TovarCasallasAlvaro2016.pdf;jsessionid=3AF46842DF4FDE64884BA906E3AFA903?sequence=1>

1. **CONTROL DEL DOCUMENTO**

|  | Nombre | Cargo | Dependencia  *(Para el SENA indicar Regional y Centro de Formación)* | Fecha |
| --- | --- | --- | --- | --- |
| **Autor (es)** | Julián Estrada Guzmán | Experto Temático | SENA Colombo Alemán | Septiembre de 2021 |
| Gustavo Rodríguez | Experto Temático | SENA Colombo Alemán | Septiembre de 2021 |
| Giovanna Andrea Escobar Ospina | Diseñadora Instruccional | Regiona Antioquia - Centro de Servicios de Salud | Octubre 2021 |
| Andrés Felipe Velandia Espitia | Revisor Metodológico y Pedagógico | Regional Distrito Capital – Centro de Diseño y Metrología | Octubre de 2021 |
| Rafael Neftalí Lizcano Reyes | Asesor Pedagógico | Regional Santander - Centro Industrial del Diseño y la Manufactura | Octubre de 2021 |
| Sandra Patricia Hoyos Sepúlveda | Revisión y corrección de estilo | Centro para la Industria de la Comunicación Gráfica - Regional Distrito Capital | Octubre de 2021 |

1. **CONTROL DE CAMBIOS**

|  | Nombre | Cargo | Dependencia | Fecha | Razón del Cambio |
| --- | --- | --- | --- | --- | --- |
| Autor (es) |  |  |  |  |  |