INSTITUTO SUPERIOR
TECNOLÓGICO PELILEO

INTERFACES MÓVILES

INTERFACES MÓVILES

Directorio editorial institucional

Dr. Rodrigo Mena Mg. *Rector*
Mg. Sandra Cando *Coordinadora Institucional*
Mg. Oscar Toapanta *Coordinador de I+D+i*
Ing. Johanna Iza *Líder de Publicaciones*

Programación Aplicaciones Web y Móviles
Mg. Diego Sánchez

Revisión técnica de pares académicos
Mg. Juan Carlos Pico
IST PELILEO
Correo: jcpico7@gmail.com
Mg. Darwin Fabricio Sánchez
IST PELILEO
Correo: fabricifabrici.05@gmail.com

ISBN: 978-9942-686-56-5
DOI:

Primera edición
Agosto 2024
https://istp.edu.ec

AUTORES

Ing. Diego Sánchez, Mg.

DOCENTE

Ingeniero en sistemas e Informática. profesional especializado en el diseño, desarrollo, implementación y mantenimiento de sistemas informáticos y tecnológicos que satisfacen las necesidades de una organización. Su trabajo abarca una amplia gama de actividades relacionadas con la tecnología de la información y la gestión de sistemas complejos, conocimiento en áreas como inteligencia artificial, análisis de datos, ciberseguridad, entre otras. Docente actualmente en el Instituto Superior Tecnológico Pelileo carrera de Desarrollo de Software.

Ing. Hernán Urquizo, Esp.

DOCENTE

Ingeniero en sistemas y Computación. Profesional especializado en Base de Datos desarrollo, implementación y mantenimiento de sistemas informáticos y tecnológicos que satisfacen las necesidades de una organización. Su trabajo abarca una amplia gama de actividades relacionadas con la tecnología de la información, conocimiento en áreas como Análisis y diseño de Sistemas, análisis de datos, entre otras. Docente actualmente en el Instituto Superior Tecnológico Pelileo carrera de Desarrollo de Software.

Ing. Fernando Beltran, Mg.

DOCENTE

Ingeniero en Sistemas, ex Docente del Instituto Tecnológico Superior Bolívar, Analista Provincial de Procesos Electorales Consejo Nacional Electoral.

Actualmente Docente del Instituto Superior Tecnológico Pelileo.

AUTORES

Ing. Freddy Morales T., Mg.

DOCENTE

Freddy Gustavo Morales Tubón es un destacado profesional del área de Desarrollo de Software en el Instituto Pelileo, con una sólida trayectoria en la enseñanza y práctica de la programación. Con 16 años de experiencia en el ámbito académico y profesional en diferentes Unidades de educación secundaria y superior, especializado en Programación Orientada a Objetos y metodologías modernas de desarrollo de software.

Ingeniero de Sistemas y Computación en Pontificia Universidad Católica del Ecuador, Magister en Educación Mención en Innovación y Liderazgo educativo por Universidad Tecnológica Indoamericana, Magister en Tecnologías de la Información Mención en Seguridad de Redes y Comunicaciones por universidad Técnica de Ambato. Freddy Morales ha dedicado su carrera a formar futuros profesionales en el campo de la tecnología, combinando una profunda comprensión teórica con una práctica constante en entornos reales. Su experiencia abarca la implementación de proyectos de software utilizando principios de diseño orientado a objetos, así como la aplicación de metodologías ágiles y otras técnicas de programación.

Ing. Fernando Pico, Mg.

DOCENTE

Destacado profesional por su capacidad de integrar soluciones tecnológicas en el ámbito empresarial y educativo, ha desarrollado materiales educativos innovadores en redes y seguridad informática. Actualmente, es docente en el Instituto Superior Tecnológico Pelileo, donde imparte materias relacionadas con la Ingeniería de Requerimientos, Integración de Sistemas y Pensamiento Computacional. Su experiencia en el sector privado y en la docencia superior lo consolidan como un experto en la implementación de tecnologías avanzadas y entornos virtuales de aprendizaje, contribuyendo al desarrollo de la próxima generación de profesionales en tecnología.

AUTORES

Ing. Javier Quinde, Mg.

DOCENTE

Ingeniero en sistemas e Informática. Profesional especializado en el diseño, desarrollo, implementación y mantenimiento de sistemas informáticos y tecnológicos que satisfacen las necesidades de una organización. Mi trabajo abarca una amplia gama de actividades relacionadas con la tecnología de la información y la gestión de sistemas complejos, conocimiento en áreas como inteligencia artificial, análisis de datos, ciberseguridad, entre otras. Docente actualmente en el Instituto Superior Tecnológico Pelileo carrera de Desarrollo de Software.

PRÓLOGO

En la actualidad, el desarrollo de aplicaciones Web ha surgido exponencialmente debido al impacto del internet en el mundo como medio de difusión de información y demás servicios. La complejidad de desarrollo de las aplicaciones Web se ha incrementado con los avances tecnológicos en el campo de la programación. En esta guía se desarrollará los contenidos relevantes correspondientes a la programación *front-end y back-end (full-stack)* que implica todo el conocimiento necesario para desarrollar aplicaciones web dinámicas e intuitivas. Es necesario indicar que el contenido base de esta materia otorga el conocimiento necesario y práctico que implica el desarrollode páginas web con las exigencias del mundo laboral.

Los sistemas operativos móviles son sistemas operativos diseñados específicamente para dispositivos móviles, como teléfonos inteligentes, tabletas y relojes inteligentes.

Android: Desarrollado por Google, Android es uno de los sistemas operativos móviles más utilizados en todo el mundo. Se utiliza en una amplia variedad de dispositivos de diferentes fabricantes y es conocido por su flexibilidad y personalización.

iOS: Desarrollado por Apple, iOS es exclusivo para dispositivos iPhone, iPad y iPod Touch. Es conocido por su diseño elegante y su ecosistema cerrado que incluye la App Store.

INSTITUTO SUPERIOR
TECNOLÓGICO PELILEO

TOMO 1:

Programación Web

Ing. Diego Sánchez, Mg.

CONTENIDOS

01

GENERALIDADES

CONCEPTOS GENERALES DE DESARROLLO WEB

Arquitectura de las Aplicaciones Web

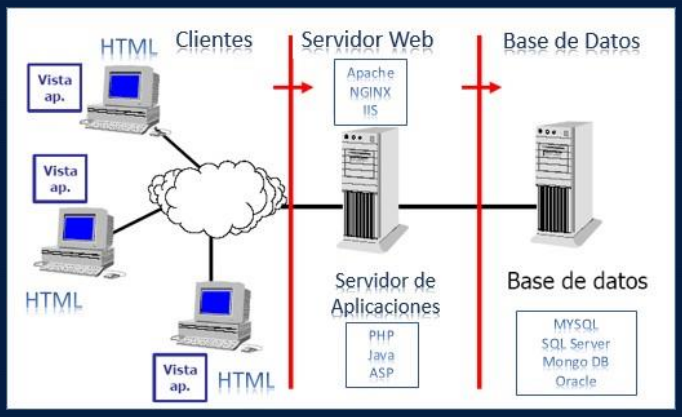
La Web

Los inicios del internet se remontan a los inicios de los años sesenta, perosu extendido no fue hasta los años noventa, gracias a la implementación y uso de la web; es así que en pocos años la web ha evolucionado enormemente, pasando de páginas sencillas con pocas imágenes y contenidos estáticos a páginas complejas con contenidos dinámicos y conexión a bases de datos loque permite la creación de las páginas web (Luján Mora, 2002).

Para tener acceso a la información que existe en internet es necesario identificar el proceso que se ejecutan para acceder a la misma. Los elementos principales que forman parte de la arquitectura de las aplicaciones web son: Clientes, Servidor Web y la Base de Datos. Cuando un cliente digita una dirección web en el navegador, automáticamente realiza una petición mediante una URL a un servidor; gracias al servicio DNS conocemos su IP, busca en el repositorio de páginas del servidory es devuelta al cliente (Lerma-Blasco, Raül V., Murcia Andrés, José Alfredo; Mifsud Talón, 2014).

Figura 1

Arquitectura de las Plataformas Web



Fuente: Elaboración propia

HTML

Hyper Text Markup Language más conocido simplemente como HTML, NO es un lenguaje de programación. Es un lenguaje de marcado, para maquetar tu página web y presentar la información, es decir crea la estructura básica. (Minera, 2011).

Servicio WWW

WWW son las iniciales que identifican a la expresión inglesa *World WideWeb*, el sistema de documentos de hipertexto que se encuentran enlazados entre sí y a los que se accede por medio de Internet. Según (Ferrer, García, & García, 2014) el servicio WWW es un sistema que contiene una cantidad de información casi infinita, y las páginas que lo contienen se caracterizan por contener texto, imágenes, animaciones e incluso sonido y video.

Dominio

El dominio es el nombre de su empresa o su página de internet, es la forma como se va a identificar su sitio o página web, es la dirección de acceso que conocerán sus visitantes. Además, las normas varían de país a país. A continuación, podemos observar la clasificación de los dominios que hace referencia (CC Bogotá, 2014).

Tipos de dominios

.com: Actividades empresariales en general – Ejm: www.google.com

.net: Actividades empresariales con la red – Ejm: www.idukay.net

.ec - .co: perteneciente a un páis – Ejm: www.gob.ec

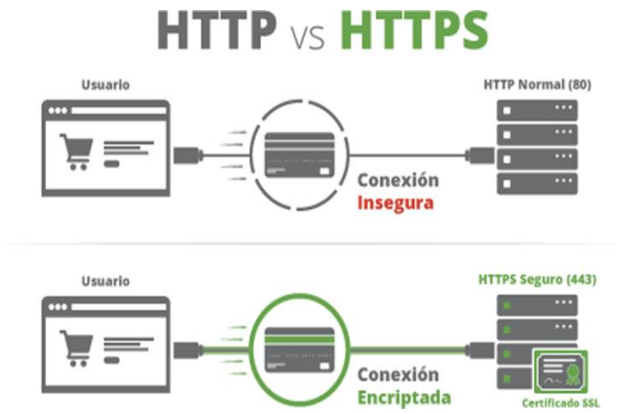
.edu: Instituciones educativas -

Introducción al Protocolo HTTP

Es el protocolo de transferencia de hipertexto, o sea, el protocolo que los servidores de *World Wide Web* utilizan para mandar documentos HTML a través de Internet. El Puerto que utiliza HTTP es el 80 (Guijarro, 2012). Al pasar el tiempo se ha visto necesario la utilización de certificados de seguridad que garanticen la seguridad en la navegación, por tanto, se ha agregado dos certificados: SSL (*Secure Sockets Layer*) y TLS (*Transport Layer Security*) que permiten una conexión segura, es así que dieron paso al protocolo HTTPS, el cual utiliza elpuerto 443 (Souza, Célia, & Freitas, 2009).

Figura 2

Comparación de conexión HTTP VS HTTPS



Fuente: (Goldberg et. al, 1998)

Tipos de páginas Web

Páginas Estáticas

Las páginas web estáticas son básicamente informativas y están enfocadas principalmente a mostrar una información permanente, donde el navegante se limita a obtener dicha información sin poder interactuar con la página visitada. En las páginas web estáticas no se utilizan bases de datos ni se requiere programación.

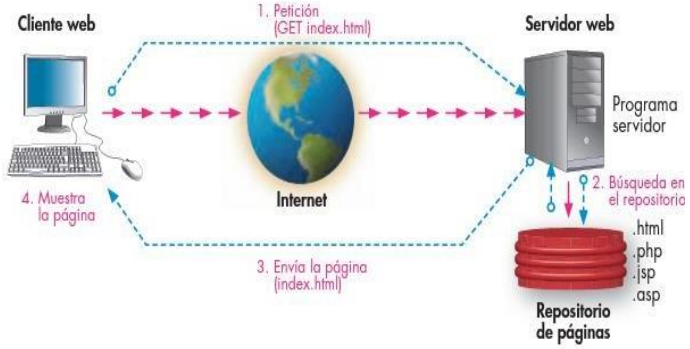
Páginas Dinámicas

Las páginas web dinámicas son aquellas en las que la información presentada se genera a partir de una petición del usuario de la página Los lenguajes utilizados para la generación de este tipo de páginas sonprincipalmente: Perl CGI, PHP, JSP y ASP.

Figura 3

Esquemas de funcionamiento de las páginas web estáticas y dinámicas.

PÁGINAS WEB ESTÁTICAS



PÁGINAS WEB DINÁMICAS

Fuente: A partir de (Lerma-Blasco et. al, 2014)

Evaluación Unidad I.

Explica las diferencias entre una arquitectura monolítica y una basada en microservicios. ¿En qué escenarios sería más conveniente utilizar cada una?

Describe el patrón de arquitectura MVC (Modelo-Vista-Controlador). ¿Cuáles son sus principales beneficios y posibles inconvenientes al aplicarlo en una aplicación web?

¿Qué es una arquitectura sin servidor (serverless) y cómo se diferencia de las arquitecturas tradicionales de servidor? Discute sus ventajas y desventajas.

En el contexto de una aplicación web, ¿qué es el escalado horizontal y vertical? Proporciona ejemplos de cuándo sería más apropiado utilizar cada tipo de escalado.

¿Cómo influye la arquitectura de una aplicación web en su capacidad de recuperación ante fallos (resiliencia)? Menciona prácticas arquitectónicas que mejoren la resiliencia.

Describe el concepto de caching en una aplicación web. ¿Qué tipos de caché existen y cómo se pueden implementar eficazmente en diferentes capas de la arquitectura?

Explica cómo un Content Delivery Network (CDN) mejora la experiencia del usuario en una aplicación web. ¿Cuáles son las limitaciones de los CDNs?

¿Qué papel juega el balanceo de carga en una arquitectura web distribuida? Describe diferentes estrategias de balanceo de carga y sus aplicaciones.

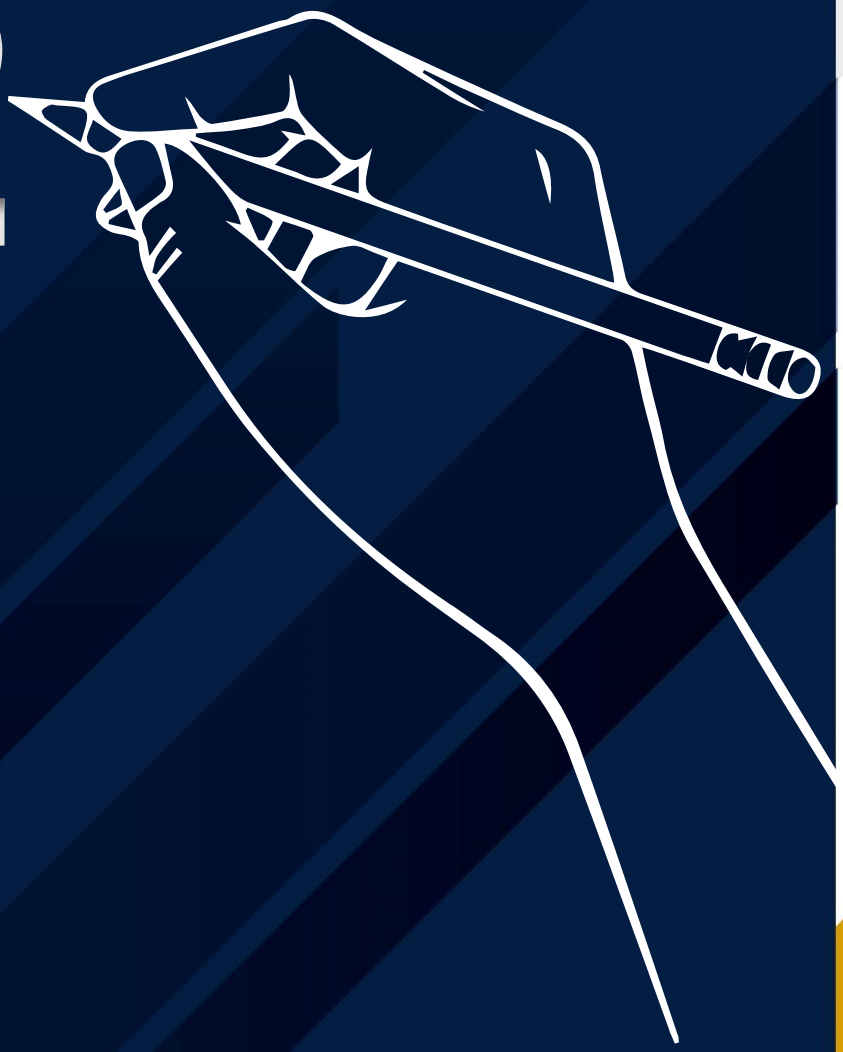
Analiza las consideraciones de seguridad que deben tenerse en cuenta al diseñar la arquitectura de una aplicación web. ¿Cómo puede una mala arquitectura comprometer la seguridad?

En una arquitectura de microservicios, ¿cómo se gestiona la comunicación entre servicios? Discute las diferentes técnicas de comunicación inter-servicios y sus implicaciones.

Explica cómo el uso de contenedores (por ejemplo, Docker) y orquestadores de contenedores (como Kubernetes) influye en el diseño y despliegue de aplicaciones web modernas.

¿Qué es la arquitectura orientada a eventos y cómo se implementa en aplicaciones web? Discute sus ventajas y desafíos en comparación con otras arquitecturas.

02



GESTORES DE CONTENIDOS

JOOMLA

Joomla es un sistema de gestión de contenidos (o CMS, por las siglas en inglés, Content Management System) que permite desarrollar sitios web dinámicos e interactivos. Permite crear, modificar o eliminar contenido de un sitio web de manera sencilla a través de un "panel de administración (Mazier,2015).

Configuración para el desarrollo práctico de las páginas Web en Joomlade manera local.

Software Necesario

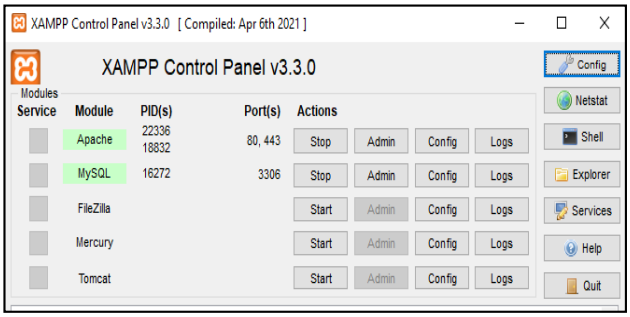
Para el funcionamiento correcto de Joomla de manera local en nuestro computador es necesario la instalación de su última reunión además del software Xampp.

Xampp

Dentro de Xampp necesitamos el gestor de base de datos de MySQL yel servidor de Apache. Una vez instalada la aplicación es necesario inicializar estos dos servicios:

Figura 4

Inicialización de los servicios MYSQL + APACHE

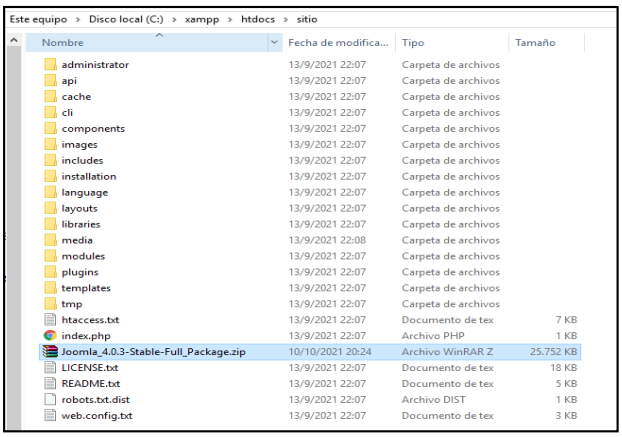


Fuente: elaboración propia

Una vez instalado xampp en Windows, dentro de la unidad C se crea una carpeta en el nombre "xampp", por tanto, para comenzar la instalación de Joomla, se debe copiar el instalador en la siguiente ruta: C:\xampp\htdocs; dentro de una carpeta creada por el usuario con el nombre a su elección, luegodescomprimir el .zip del instalador de Joomla.

Figura 5

Directorio de instalación de Joomla - Windows

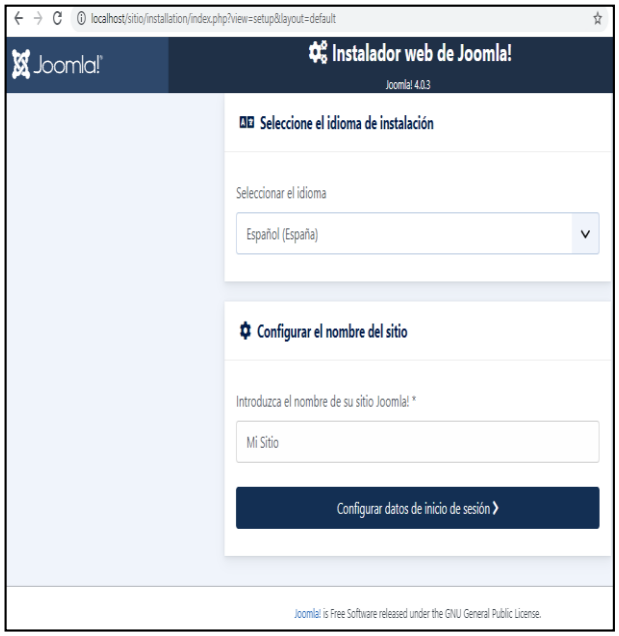


Fuente: elaboración propia

Una vez descomprimidos los archivos, en la carpeta indicada, se debeproceder a la instalación de xammp, para lo cual, se debe abrir su navegadorfavorito y digital la siguiente dirección: *localhost/"nombre_del_sitio"*. Ejem: localhost/sitio. Una vez dentro, en la ventana principal, seleccionar el idioma y el nombre del sitio.

Figura 6

Ventana de instalación de Joomla



Fuente: elaboración propia

Para completar la instalación se definen los usuarios: administrador, super administrador, el nombre de la base de datos y su contraseña, para luego comenzar con el desarrollo de una página web mediante con el gestor de contenidos de Joomla.

Ventajas

Amplio catálogo de módulos (el equivalente a plugins de WordPress), pero en menor medida. Más rápido de entender y con soporte en diferentes idiomas.

Desventajas

Se requieren más conocimientos a nivel técnico, tanto para su puesta en funcionamiento como para su mantenimiento. El SEO hay que trabajarlo máscon respecto a WordPress para conseguir resultados similares

Partes fundamentales para el desarrollo en Joomla

Artículos

Un Artículo es cierta información escrita que quiere mostrar en su sitioweb. Suele contener algo de texto y puede contener imágenes y otros tipos de contenido. Para muchos sitios web realizados con Joomla, los artículos constituyen la mayoría de la información mostrada en el sitio.

Categorías

Las Categorías en Joomla proporcionan un método opcional de organizar los artículos. He aquí como funciona. Una categoría contiene artículos y otras categorías. Un artículo puede estar sólo en una categoría. Si una categoría está dentro de otra categoría, se dice que es una subcategoría de dicha categoría.

Plantillas

Una plantilla es un tipo de extensión de Joomla, que cambia la apariencia del sitio.

Figura 7

Plantillas de Joomla

Fuente: A partir de (Alava, 2015)

Menús

Es un conjunto de elementos de menú que se utilizan para la navegación por el sitio web. Cada elemento de menú define una URL a una página del sitio, y tiene parámetros de configuración que controlan los contenidos (artículos, categoría o categorías, listas, elementos etiquetados, etc,) y estilo (módulo(s), diseño) de la página.

Plugins

Un plugin es un tipo de extensión de Joomla!. Los plugins proporcionan funciones que están asociadas con el lanzamiento de eventos o funciones adicionales que mejoran la apariencia del sitio Web.

WORDPRESS

WordPress es un sistema de gestión de contenidos, enfocado a la creación de cualquier tipo de página web. Originalmente alcanzó una gran popularidad en la creación de blogs, para luego convertirse, en una de las principales herramientas para la creación de páginas web comerciales. Está desarrollado en el lenguaje PHP para entornos que ejecuten MySQL y Apache, bajo licencia GPL y es software libre (Rodríguez, 2018).

Características de Wordpress

- Instalación, actualización y personalización fácil de realizar
- Separa el contenido del diseño
- Permite asignar diferentes estados a un artículo ("post) (Publicando, Borrador, Esperando Revisión y Privado.
- Ordena los artículos y páginas en Categorías subcategorías y etiquetas ("tags")
- Permite comentarios
- Utiliza diferentes plugins y Widgets para los temas.

Wordpress gratuito:
www.wordpress.org

Ventajas:

- Fácil de Instalar
- Su uso es gratuito con la utilización de un Subdominio de Wordpress.
- Excelente rendimiento
- Copias de seguridad automáticas
- Actualización automática

Desventajas:

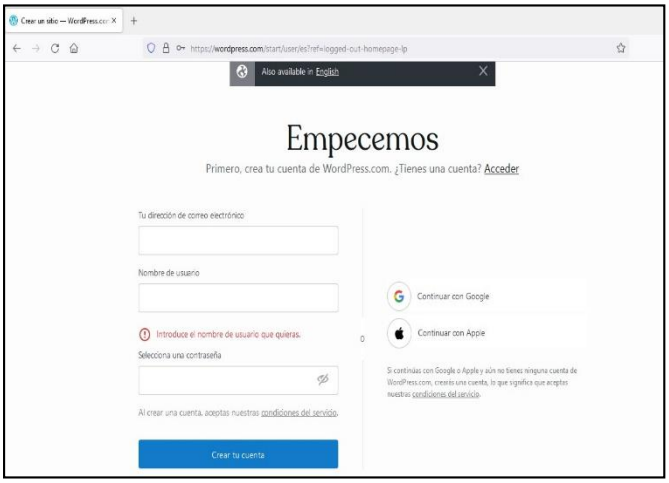
- No se puede instalar plugins
- Su uso es gratuito
- Excelente rendimiento
- Copias de seguridad automáticas
- Actualización automática

Partes fundamentales para el desarrollo en Wordpress

Para ingresar a wordpress se debe digitar en el navegador la página web: www.wordpress.com una vez dentro, si aún no se tiene una cuenta, hay que crearla, dando click en el botón de "empezar ahora" de la página de inicio. Existen tres maneras para crear una cuenta de Wordpress; 1. Nos registramos con nuestro correo personal, para lo cual ingresamos usuario y contraseña exclusivos para la página. 2. Iniciar sesión con una cuenta de Google o, 3. Iniciar sesión con una cuenta de Apple.

Figura 8

Registro e inicio de sesión en Wordpress



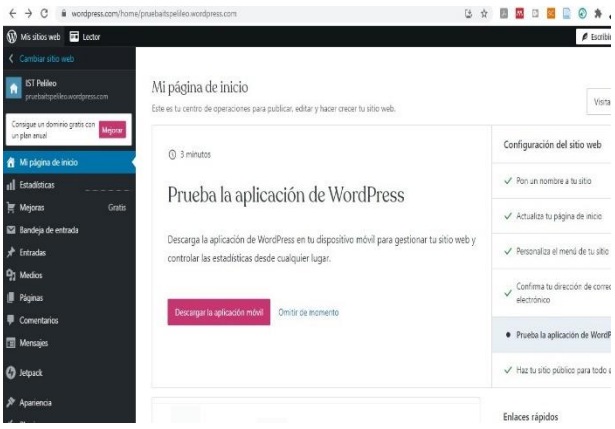
Fuente: elaboración propia

Panel de Configuración de Wordpress

Este panel de administración permite al administrador configurar la información que se va a mostrar. Aquí podemos encontrar las siguientes opciones de configuración: entradas, medios, páginas, apariencia, plugins, widgets, menús, entre otras.

Figura 9

Panel de configuración de Wordpress



Fuente: elaboración propia

Entrada

La Entrada o Post en WordPress es el contenido que diferencia a un blog de una página web, ya que mientras las páginas son estáticas, las entradas generalmente tienen opción de agregar opiniones o comentarios y de publicarse de acuerdo a un orden cronológico.

Medios

En WordPress se entiende por elemento multimedia, que se almacena en la biblioteca de medios, cualquier archivo que se carga en la web o que se puede descargar desde un enlace.

Páginas

Las páginas de WordPress se han desarrollado para albergar aquellos contenidos que hay en tu web que normalmente no recibirán actualizaciones. Los escribes una vez y permanecerán invariables en el tiempo. Serían las típicas páginas de quiénes somos, contacto, servicios, etc

Apariencia

En el apartado de apariencia, se personaliza el diseño del sitio mediante la instalación los temas disponibles en la web además permite agregar los menús que se van a utilizar en el sitio. Otra herramienta interesante en este apartado es la utilización de los *Widgets*, los cuales nos permiten agregar la funcionalidad del sitio. Entre los principales *widgets* disponibles tenemos: Sliders: que dan la funcionalidad del sitio. Entre los principales *widgets* disponibles tenemos: Sliders: que dan la funcionalidad al sitio con imágenes aleatorias, *WooCommerce*: agrega la funcionalidad de comprar y vender mediante el carrito de comprasen línea, *Worfence*: seguridad al sitio web, *Elementor*: aplicación para distribuir de mejor manera el contenido web, *UpdraftPlus*: permite sacar respaldos periódicos al sitio web (Hayder, 2006).

Plugin

Un *plugin* es una pequeña aplicación que se añade al sistema, generalmente aportan funciones muy específicas. Podemos incluir *plugins* en navegadores *web*, reproductores de audio o vídeo, o en gestores de contenidos (Rodríguez, 2018).

Evaluación Unidad II.

Explica qué es Joomla y cómo se diferencia de otros sistemas de gestión de contenido como WordPress o Drupal.

Describe la evolución de Joomla desde su creación. ¿Cuáles han sido los cambios más significativos en sus versiones principales?

¿Qué ventajas ofrece Joomla para la creación de sitios web en comparación con otras plataformas?

Comenta cómo Joomla se organiza en términos de componentes, módulos y plugins. ¿Cómo interactúan entre sí?

Describe el proceso de instalación de Joomla en un servidor web. ¿Qué pasos son esenciales y qué configuraciones adicionales se pueden realizar después de la instalación?

Explica cómo se gestiona la base de datos en Joomla. ¿Qué archivo es crucial para esta configuración y cómo se edita?

¿Cómo se pueden implementar URLs amigables en Joomla? Describe los pasos necesarios para activarlas y configurarlas correctamente.

Explica el concepto de "override" en Joomla. ¿Cómo se realiza y en qué casos es recomendable utilizarlo?

¿Cómo se realiza la actualización de Joomla a una nueva versión? ¿Qué precauciones se deben tomar antes y durante el proceso de actualización?

Describe cómo se puede optimizar el rendimiento de un sitio Joomla. ¿Qué configuraciones o extensiones pueden ayudar en este proceso?

¿Cómo se gestionan los permisos y niveles de acceso de los usuarios en Joomla? Da ejemplos de situaciones en las que se necesiten configuraciones específicas.

Explica cómo se puede personalizar una plantilla en Joomla. ¿Qué herramientas y técnicas se pueden utilizar para adaptar una plantilla a las necesidades específicas de un sitio web?

Describe el proceso para crear y asignar un nuevo módulo en Joomla. ¿Qué opciones están disponibles para personalizar la posición y visibilidad del módulo en el sitio?

¿Qué consideraciones se deben tener en cuenta al elegir y gestionar extensiones en Joomla? ¿Cómo se evalúa la seguridad y compatibilidad de una extensión?

¿Cómo se puede mejorar la seguridad de un sitio Joomla? Menciona prácticas recomendadas y extensiones que ayudan a proteger el sitio contra amenazas comunes.

Explica cómo se puede integrar Joomla con otras plataformas o servicios (por ejemplo, CRM, sistemas de pago, redes sociales). ¿Qué extensiones o configuraciones son necesarias para esta integración?

¿Qué es y cómo funciona el sistema de caché en Joomla? Explica cómo configurarlo para mejorar la velocidad de carga del sitio sin afectar su funcionalidad.

03

LENGUAJES DE MARCADO

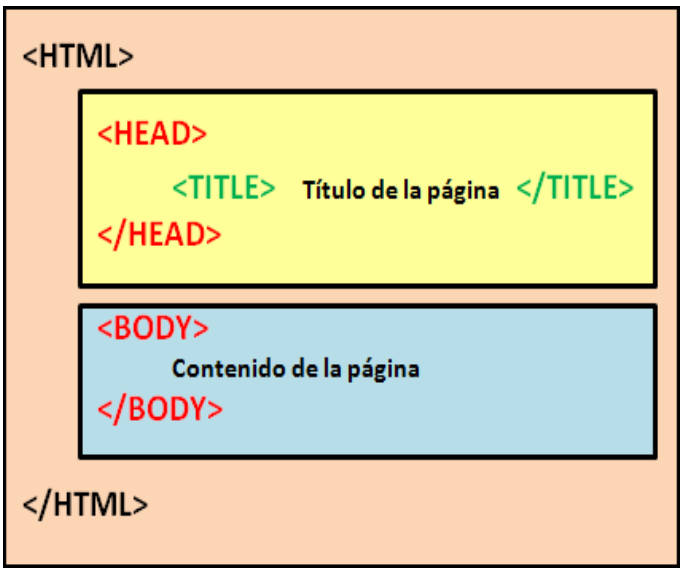
HTML Básico (HTML5)

Hyper Text Markup Language más conocido simplemente como HTML, NO es un lenguaje de programación, no se programa. Es un lenguaje de marcado, para maquetar tu página web y presentar la información, crea la estructura básica (Equipo Vértice, 2019)

Estructura básica de un documento HTML

Figura 10

Estructura de los documentos HTML (Etiquetas – Atributos)



Fuente: Elaboración propia

Las etiquetas HTML son pequeños bloques de código, que indican al navegador como debe interpretar el contenido recogido entre dichas etiquetas. La etiqueta presenta frecuentemente dos partes: Una apertura de forma general <etiqueta> Un cierre de tipo </ etiqueta>.

Todo lo incluido en el interior de esa etiqueta sufrirá las modificaciones que caracterizan a esta etiqueta.

Así, por ejemplo:

Las etiquetas y definen un texto en negrita. Si en nuestro documento HTML escribimos una frase con el siguiente código: Esto está en negrita

El resultado Será: Esto está en negrita

Tabla 11

Detalle de las etiquetas básicas para la estructura de un documento HTML

Etiqueta	Función
<html>	Empieza un documento HTML
<head>	Zona de cabecera
<tittle>	Zona de título
</tittle>	Termina zona de título
</head>	Termina zona de cabecera
<body>	Zona del cuerpo del documento
</body>	Termina zona del cuerpo del documento
</html>	Termina documento HTML

Fuente: Elaboración propia

Tabla 12

Detalle de las etiquetas generales de un documento HTML

Detalle de las etiquetas generales de un documento HTML

Etiqueta	Función
<h1> …. <h5>	Son los tamaños de fuente para los títulos o encabezados
<a>	Definen los enlaces
<p>	Etiqueta para el párrafo
<b>	Se utiliza para resaltar el texto
<u>	Texto subrayado
<i>	Texto en cursiva
<table>	Etiqueta para dibujar una tabla, dentro de esta tenemos filas <tr> y celdas <td>
</html>	Termina documento HTML

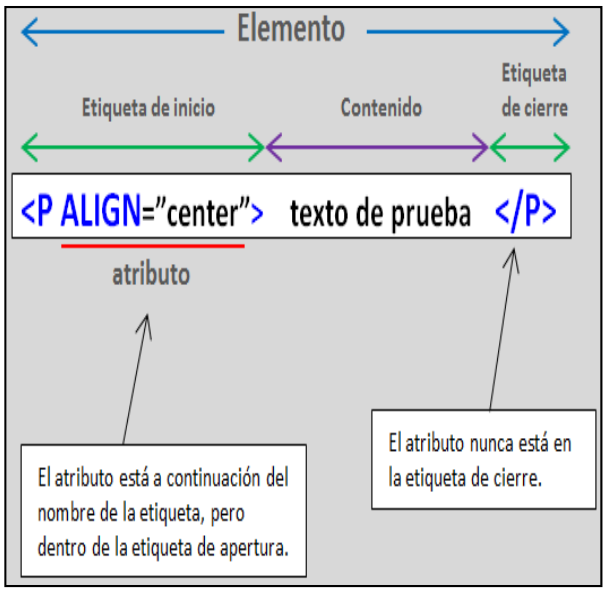
Fuente: Elaboración propia

Atributos

Las etiquetas tienen propiedades, las cuales son descritas por medio delos Atributos. Un atributo es una opción que permite proporcionar detalles acerca de cómo una etiqueta afectará el contenido.

Tabla 13

Estructura básica de un documento HTML



Fuente: Elaboración propia

Ejemplos del uso de los atributos con etiquetas

Tamaño del texto

 Texto </size>

Color del Texto

 Texto

Tipo de Letra

<font face=" Century Gothic">

Color de Fondo del Documento

<body bgcolor= "#A9DFBF">

Color de Texto de todo el documento

<body text="red">.

Manejo de Texto, Enlaces, Imágenes, Tablas

Para dar formato a un texto, se debe realizar tres tareas tan tales como: definir los párrafos, justificarlos, introducir viñetas, numeraciones o bien poner en negrita, itálica. Para definir los párrafos utilizamos la etiqueta <p> que introduce un salto y deja una línea en blanco antes de continuar con el resto del documento.

La etiqueta
, no tiene su cierre correspondiente (</br>), pero nos sirve para dar un salto de línea.

Listas

Las listas en HTML proporcional una forma de clasificar la información.

Lista numerada

 ...

Ejemplo:

 uno

 dos

 tres

Lista no numerada

 ...

Ejemplo:

 uno

 dos

 tres

Elementos de una lista.

 ... (No es necesario cerra la etiqueta)

Enlaces

Sirven para acceder desde una página a otra página o a otro recursodisponible.

Enlace absoluto a una página

texto del enlace

Enlace relativo a una página

texto del enlace.

Marcadores

(enlace interno) dentro de una página

 ... Ejemplo:

Enlace a un marcador de la misma página

texto del enlaceEjemplo:

Ir a inicio de la página

Enlace a un marcador de otra página (que puede darse con dirección absoluta o relativa)

Ejemplo:

texto del enlace

Enlace a otra página (absoluta o relativa, con o sin marcador) que se abra en otra ventana.

texto del enlace

Ejemplo:

 Ir a Google

Imágenes

<img src="nombre.gif">

Imágenes con texto Alternativo:

<mg src="ruta_imagen" alt="Texto Alternativo"> Ejemplo:

<img src="Logo.png" alt="Imágen_de_Pelileo">

Tamaño de imágenes

<img src="ruta_imagen" width="111px" height="222px" alt="Texto Alternativo">

Ejemplo:

<img src=" Logo.png " width="111px" height="222px" alt=" Imagen de Empresa">

Imagen como fondo de una página Web

<body background="fondo.gif">
Ejemplo:

<body background=" Logo.png ">

Alineación y dimensionado de imágenes

(Top, middle, buttom, align, clear, width, height).

La alineación de las imágenes fue, en su día, el primer golpe de efecto del programa Navigator de Netscape. Permitió alinear una imagen a la izquierda o a la derecha de la página y hacer que el texto la rodee completamente, consiguiéndose así una apariencia similar a la de una revista.

Es el caso de este párrafo con respecto a la imagen de la derecha. Obsérvese cómo las líneas, cuando rebasan su parte inferior, continúan normalmente hasta el margen derecho de la página. De manera análoga, sepuede alinear la imagen a la izquierda, comportándose el texto de una forma equivalente.

<br clear="left"> Busca el primer margen libre (clear) a la izquierda.

<br clear="right"> Busca el primer margen libre a la derecha.

<br clear="all"> Busca el primer margen libre a ambos lados.

Tablas

Etiqueta: <table>

<tr> : (representando a las filas contenedoras de las celdas)

<td>: (representando a las celdas)

Ejemplo:

```
<table>
<tr>
  <td>Celda 1</td>
  <td>Celda 2</td>
  <td>Celda 3</td>
</tr>
<tr>
  <td>Celda 4</td>
  <td>Celda 5</td>
  <td>Celda 6</td>
</tr>
</table>
```

Para agregar un título a la tabla

<caption> Tabla 1 </caption>

Atributos de las Tablas

Borde:

<table border="2">

Es el caso de este párrafo con respecto a la imagen de la derecha. Obsérvese cómo las líneas, cuando rebasan su parte inferior, continúan normalmente hasta el margen derecho de la página. De manera análoga, sepuede alinear la imagen a la izquierda, comportándose el texto de una forma equivalente.

Esto se consigue con las extensiones de la etiqueta align, con los comandos top, middle y bottom).

<br clear="left"> Busca el primer margen libre (clear) a la izquierda.

<br clear="right"> Busca el primer margen libre a la derecha.

<br clear="all"> Busca el primer margen libre a ambos lados.

Tablas

Etiqueta: <table>

<tr> : (representando a las filas contenedoras de las celdas

<td>: (representando a las celdas)

Ejemplo:

```
<table>
<tr>
<td>Celda 1</td>
<td>Celda 2</td>
<td>Celda 3</td>
</tr>
<tr>
<td>Celda 4</td>
<td>Celda 5</td>
<td>Celda 6</td>
</tr>
</table>
```

Para agregar un título a la tabla

<caption> Tabla 1 </caption>

Atributos de las Tablas

Borde:

<table border="2">

Tamaño

width="550"

Color

bgcolor=FFCECB

Alineación

align=center

Borde

border="1"

Ejemplo 1:

<table width="200" bgcolor=FFCECB align="center" border="1">

Ejemplo 2:

```
<table class="egt">
<tr>
<td>Celda 1</td>
<td>Celda 2</td>
<td>Celda 3</td>
</tr>
<tr>
<td>Celda 4</td>
<td>Celda 5</td>
<td>Celda 6</td>
</tr>
</table>
```

Estilos HTML (Style)

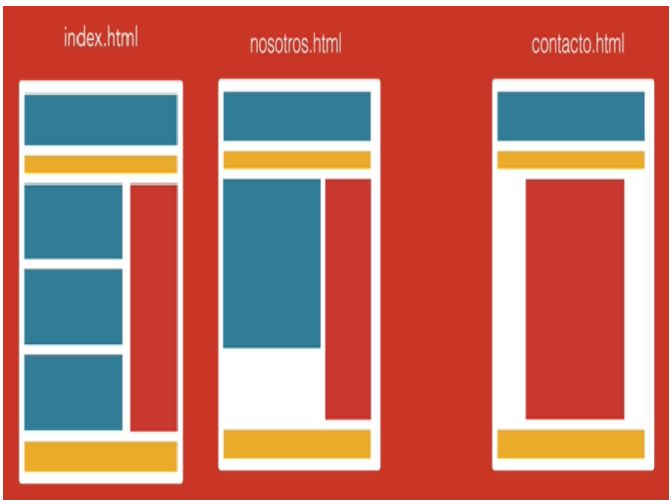
<DIV STYLE="font-size: 16pt; color: red">

</DIV>

Diseño y Estructura HTML

En un sitio web básico cada página puede tener un contenido diferente, a continuación, se presentan los diseños dispuestos para cada página. Todo depende del tipo de contenido que se requiera presentar, tomando en consideración las etiquetas necesarias para estructurar cada sitio.

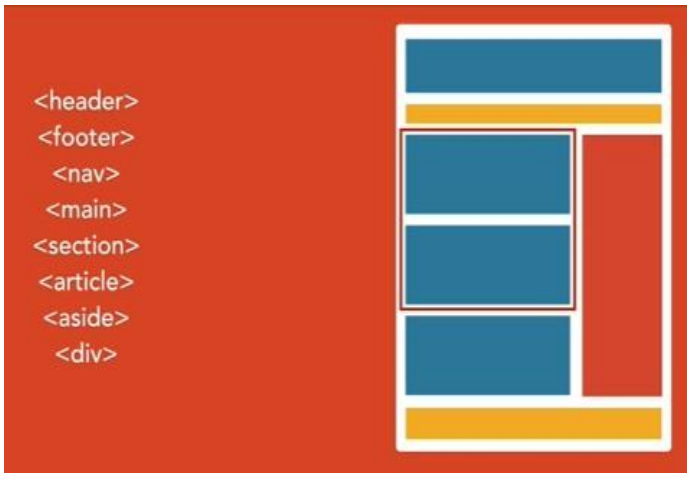
Figura 14
Estructuras básicas de los sitios web



Fuente: Elaboración propia

Figura 15

Etiquetas de estructura HTML



Fuente: Elaboración propia

Formularios y Hojas de Estilo (CSS)

Son hojas de estilo en cascada, que permean darle al código HTML un diseño único y especial en pocas líneas. Por tanto, CSS se encarga de:

Tamaños y Tipos de Fuente

- Colores
- Espacios
- Márgenes
- Adaptar diseños a distintos dispositivos

Animaciones (Eguíluz, 2008).

Programación en cascada:

La cascada es el mecanismo que controla el resultado final de como visualizaremos una etiqueta html cuando a esta se le apliquen distintas instrucciones CSS.

Ejemplo: se aplica solo la última instrucción.

```
p {
   Color: blue;
}
p {
  Color: red;
}
```

Formato CSS

A continuación de detalla el formato recomendado para la programación de una hoja de estilos css:

```
p {
   color: blue;
}
```

Inclusión global del estilo en un documento HTML

Se hace poniendo un bloque de instrucciones dentro de las etiquetas

<style> </style>, que deberá estar colocado dentro de la cabecera del documento, después del título, entre las etiquetas </title> y </head> (de igual manera que se hace con los scripts de Javascript).

Esta etiqueta <style> tiene un atributo, TYPE, que especifica el tipo de medio en que va a ser publicado en Internet, en nuestro caso será "text/css" (que permitirá a los navegadores que no soporten este tipo el ignorar la hoja de estilo). Es decir, la etiqueta queda de esta manera: <style type="text/css"> (Schulz, 2008).

Por tanto, la estructura será la siguiente:

<html>

<head>

<title>[título] </title>

<style type="text/css">

[bloque de instruciones de estilo]

</style>

</head>

<body>

[conjunto de todas las etiquetas que componen la página]

</body>

</html>

Veamos ahora con un ejemplo, cómo se escribe el bloque de instrucciones del estilo, incluido dentro de la etiqueta <style> (más tarde se verán con detalle cada uno de los atributos):

<style type="text/css">

body {background: yellow ;font-size: 10pt; font-family: arial; margin-left: 0.5in; margin-right: 0.5in}

h1 {background: blue; font-size: 14pt;font-weight: bold; color: red} h2 {font-size: 12pt;font-weight: bold; color: red}

div {background: url(nubes.jpg)}

</style>

CSS en documento externo

Nombre del archivo: miestilo.css

BODY {background: yellow ;font-size: 10pt; font-family: Arial; margin-left: 0.5in; margin-right: 0.5in}

H1 {background: blue; font-size: 14pt;font-weight: bold; color: red} H2 {font-size: 12pt;font-weight: bold; color: red}

DIV {background: URL(fondo.jpg)}

Obsérvese que no tiene ninguna etiqueta, pues no es un documento normal HTML, sino que es un fichero de texto que sólo contiene el bloque de definición del estilo. Se guarda con el nombre que se quiera, pero tiene que tener necesariamente la extensión .css. Supongamos que lo guardamos con el nombre de miestilo.css

En todas las páginas que queremos que tengan este estilo concreto, solamente deberemos de añadir (en el mismo sitio de la cabecera que para el caso anterior, es decir entre </title> y </head>, la siguiente etiqueta:

<link rel=stylesheet href="miestilo.css" type="text/css">

Atributos de las hojas de estilo.

A continuación, se muestra en una tabla el resumen de los atributos que se pueden incluir en las hojas de estilo.

Tabla 16

Detalle de las etiquetas generales de un documento HTML.

Atributo	Descripción	Valores	Ejemplo
font-size	Establece el tamaño del texto	Puntos(pt) Pulgadas(in) Centímetros(cm) Pixeles(px)	{font-size: 12pt}
font-family	Establece la fuente	Nombre de la fuente Nombre de la familia de la fuente	{font-family: century gothic}
font-weight	Establece el espesor de la fuente	Extra-light Light Demi-light Medium Demi-bold Bold Extra-bold	{font-weigth: bold}
font-style	Convierte el texto a cursiva	Normal italic	{font-style: italic}
line-height	Establece la distancia entre las líneas	Puntos(pt) Pulgadas(in) Centímetros(cm) Pixeles(px) Porcentaje(%)	{line-height: 24pt}
color	Establece el color del texto	Nombre del color valores: RGB	{color_blue}
text-decoration	Subraya o demarca el texto	None Underline Italic Line-through	{text-decoratio: underline}
margin-left	Establece el margen izquierdo de la página	Puntos(pt) Pulgadas(in) Centímetros(cm) Pixeles(px)	{margin-left: 1 in}
margin-rigth	Establece el margen derecho de la página	Puntos(pt) Pulgadas(in) Centímetros(cm) Pixeles(px)	{margin-right: 1 in}
margin-top	Establece el margen superior de la página	Puntos(pt) Pulgadas(in) Centímetros(cm)	{margin-top: 20 px}

Fuente: Elaboración propia

Explicación de los atributos

font-size

El atributo font-size establece el tamaño del texto en puntos (pt), pulgadas (in), centímetros (cm), o pixeles (px).

Ejemplos:

{font-size: 12pt}

{font-size: 1in}

{font-size: 5cm}

{font-size: 24px}

Explicación de los atributos

font-family

el atributo font-family establece la fuente del texto. Se puede especificar una única fuente, **ejemplo:**

{font-family: Arial}

u otras fuentes alternativas, separadas por una coma, **ejemplo:**

{font-family: Arial, Helvetica}

En el ejemplo anterior, nos aseguramos que los sistemas que no soporten la fuente Arial, utilicen la fuente Helvética. Es muy aconsejable especificar, como último recurso, un nombre genérico de familia de fuentes.

Ejemplo:

{font-family: Arial, Helvetica, sans-serif}

Estos nombres genéricos de familia de fuentes (serif, sans-serif, cursive, fantasy, o monospace) tienen la ventaja de que son representados como las fuentes que tenga instaladas el usuario. Si se hace referencia a una fuente cuyo nombre consiste en varias palabras (separadas por espacios en blanco), hay que englobarla entre comillas.

Ejemplo:

{font-family: "Courier New"}

font-weight

El atributo font-weight establece el espesor de la fuente:

{font-weight: medium}

{font-weight: bold}

Los valores aceptados (extra-light, light, demi-light, medium, demi-bold, bold, y extrabold) dependen en las fuentes que tenga instaladas el usuario. (Porejemplo, el sistema del usuario puede que sólo permita medium y bold para una determinada fuente).

font-style

El atributo font-style establece la fuente como cursiva:
{font-style: italic}

line-height

Este atributo establece la separación entre líneas, que se puede expresar en puntos (pt), pulgadas (in), centímetros (cm), pixels (px) o porcentaje (%).

Ejemplo:
{line-height: 20pt}

Color

Este atributo establece el color del texto de acuerdo con su valor

hexadecimal.

Ejemplo:
{color: #33CC00}
{color: red}
Los nombres de los colores son los siguientes:

Figura 11

Etiquetas de estructura HTML

black	silver	gray	white
maroon	red	purple	fuchsia
green	lime	olive	yellow
navy	blue	teal	aqua

Fuente: Elaboración propia

Background

Se utiliza este atributo para destacar secciones de una página, estableciendo un color de fondo o una imagen de fondo. Para establecer un color de fondo, se especifica su valor hexadecimal, o un nombre de color (ver el atributo color visto anteriormente).

Ejemplos:
{background: red}
{background: #6633FF}

También se puede colocar una imagen de fondo en el ámbito de la etiqueta. Es decir, se puede poner, por ejemplo, una imagen de fondo en un párrafo determinado. Para colocar una imagen, se especifica el URL entre paréntesis (no entre comillas, como es lo habitual).

Ejemplo:
{background: URL(http://nubes.jpg)}

Evaluación Unidad III.

Explica qué es un lenguaje de marcado y da ejemplos de lenguajes de marcado comunes.

Describe la función de las etiquetas en HTML y proporciona ejemplos de cómo se utilizan.

¿Cómo se diferencian HTML y XML en cuanto a su propósito y estructura?

Explica la importancia de la etiqueta <head> en un documento HTML y qué elementos suele contener.

Describe el proceso de anidamiento de etiquetas en HTML y por qué es importante seguir una estructura correcta.

¿Qué es el DOM (Document Object Model) y cómo se relaciona con HTML?

Explica qué es un atributo en HTML y proporciona ejemplos de atributos comunes que se utilizan en diferentes etiquetas.

Describe cómo se puede utilizar CSS junto con HTML para mejorar la presentación de una página web.

¿Qué es un archivo XML bien formado? Explica las reglas básicas que deben seguirse para asegurar que un archivo XML esté bien formado.

Discute las ventajas y desventajas de utilizar HTML5 en comparación con versiones anteriores de HTML.

04

INTERACCIÓN CON LA BASE DE DATOS

Php (Php + MySQL)

Para trabajar en PHP necesitamos tener un servidor local, para ello instalamos los siguientes programas:

Xampp (Servidor SQL)

Sublime Text o Visual Studio Code (Editor html)

Una vez instalados los programas, hay que ejecutarlos.

Creación de Directorio:

Crear la carpeta **phpejm** dentro del siguiente directorio: C:/xampp/htdocs/. Abrimos el programa SublimeText y ubicamos la carpeta creada desde el programa de la siguiente manera: File Open Folder y seleccionamos la carpeta phpejm dentro de la ruta detallada anteriormente; dentro de la misma creamos el archivo **hola.php**, el cuál va a ser nuestro primerejemplo de PHP.

hola.php:

```
<?php
echo ("hola mundo desde PHP") ;
?>
```

Para visualizar el resultado, dentro de la barra de navegación del explorador de internet se debe digitar localhost/phpejm y seleccionamos nuestro archivo.

Variables en PHP

Las variables en php se definen de la siguiente manera**: $variable,** con el signo **$.**

Tipo Numérico

```
<?php
$edad = 29; echo $edad;
?>
```

En este ejemplo se imprime en pantalla el número 29.

Tipo Texto:

Si agregamos en siguiente código:

```
echo 'Mi edad es: '.$edad;
```

En pantalla de muestra **Mi edad es: 29,** tomando en consideración queél **"."** en php es concatenar el texto con el número.

Para definir una variable *String*, la hacemos de la siguiente manera:

```
$nombre = 'Pablo';
```

Automáticamente la variable le reconoce como una variable de tipo *String*.

Tipo Verdadero/Falso (Boleano)

Condicionales:SI

```
if (condición) {
        Código Respuesta
}

if (condición) {
        Código Respuesta
}
else {
Código Respuesta 2
}
```

Operadores Aritméticos:

Tabla 12

Detalle del uso de los operadores aritméticos.

Operadores Aritméticos		
Ejemplo	**Nombre**	**Resultado**
+$a	Identidad	Conversión de *$a* a int o float según el caso.
-$a	Negación	Opuesto de *$a*.
$a + $b	Adición	Suma de *$a* y *$b*.
$a - $b	Sustracción	Diferencia de *$a* y *$b*.
$a * $b	Multiplicación	Producto de *$a* y *$b*.
$a / $b	División	Cociente de *$a* y *$b*.
$a % $b	Módulo	Resto de *$a* dividido por *$b*.
$a ** $b	Exponenciación	Resultado de elevar *$a* a la potencia *$b*ésima. Introducido en PHP 5.6.

Fuente: A partir de (Arias, 2013).

Operadores Lógicos:

Tabla 13

Detalle del uso de los operadores lógicos

Operadores Lógicos		
Ejemplo	**Nombre**	**Resultado**
$a and $b	And (y)	**TRUE** si tanto *$a* como *$b* son **TRUE**.
$a or $b	Or (o inclusivo)	**TRUE** si cualquiera de *$a* o *$b* es **TRUE**.
$a xor $b	Xor (o exclusivo)	**TRUE** si *$a* o *$b* es **TRUE**, pero no ambos.
! $a	Not (no)	**TRUE** si *$a* no es **TRUE**.
$a && $b	And (y)	**TRUE** si tanto *$a* como *$b* son **TRUE**. (Java – JavaScript)
$a || $b	Or (o inclusivo)	**TRUE** si cualquiera de *$a* o *$b* es **TRUE**. (Java – JavaScript)

Fuente: A partir de (Arias, 2013).

Operadores de Comparación:

Tabla 14

Detalle del uso de los operadores de comparación

Operadores de Comparación		
Ejemplo	**Nombre**	**Resultado**
$a == $b	Igual	**TRUE** si *$a* es igual a *$b* después de la manipulación de tipos.
$a === $b	Idéntico	**TRUE** si *$a* es igual a *$b*, y son del mismo tipo.
$a != $b	Diferente	**TRUE** si *$a* no es igual a *$b* después de la manipulación de tipos.
$a <> $b	Diferente	**TRUE** si *$a* no es igual a *$b* después de la manipulación de tipos.
$a !== $b	No idéntico	**TRUE** si *$a* no es igual a *$b*, o si no son del mismo tipo.
$a < $b	Menor que	**TRUE** si *$a* es estrictamente menor que *$b*.
$a > $b	Mayor que	**TRUE** si *$a* es estrictamente mayor que *$b*.
$a <= $b	Menor o igual que	**TRUE** si *$a* es menor o igual que *$b*.
$a >= $b	Mayor o igual que	**TRUE** si *$a* es mayor o igual que *$b*.
$a <=> $b	Nave espacial	Un integer menor que, igual a, o mayor que cero cuando *$a* es respectivamente menor que, igual a, o mayor que *$b*. Disponible a partir de PHP 7.
$a ?? $b ?? $c	Fusión de null	El primer operando de izquierda a derecha que exista y no sea **NULL**. **NULL** si no hay valores definidos y no son **NULL**. Disponible a partir de PHP 7.

Fuente: A partir de (Arias, 2013).

Operadores de Incremento/Decremento:

Tabla 15

Detalle del uso de los operadores incremento/decremento.

Operadores Incremento/Decremento		
Ejemplo	**Nombre**	**Resultado**
++$a	Pre-incremento	Incrementa *$a* en uno, y luego retorna *$a*.
$a++	Post-incremento	Retorna *$a*, y luego incrementa *$a* en uno.
--$a	Pre-decremento	Decrementa *$a* en uno, luego retorna *$a*.
$a--	Post-decremento	Retorna *$a*, luego decrementa *$a* en uno.

Fuente: A partir de (Arias, 2013).

While:

Ejemplo:

```
<?php
$i = 1;
        while ( $i<= 10) {
                echo " $i <br> ";
                $i++;
                }
?>
```

While:

Ejemplo:

```
<?php
$i = 1;
        do {
                echo $i.'<br>';
                $i++;
}
;
                echo "fin de ciclo";
?>
```

HTML + PHP:

Para trabajar con html con php se debe realizar el siguiente procedimiento, tomar en consideración que el archivo se debe guardar con formato .php:

Inicializar el archivo con el siguiente código:

```
<?php
$nombre = 'Pablo';
?>
```

Seguido trabajamos con la estructura básica de un archivo en html:

```
<!DOCTYPE html>
<html>
<head>
<title></title>
</head>
<body>
<h2> Hola: <?php echo $nombre
?></h2>
</body>
</html>
```

Inicializamos en código php en el lugar en donde queremos aparezca la información.

Arreglos

Para trabajar con arreglos en html y php, lo primero que debemos hacer es crear un archivo con el nombre: arreglos.php.

Existen 2 maneras de declarar las variables de un array

$variable = [‘valor1’, ‘valor2’, ‘valor3’,…]

$variable =array (‘valor1’, ‘valor2’, ‘valor3’,…)

Ejemplo:

```
<?php
$animales = ['Gato', 'Perro', 'Elefante', 'León'];
?>
<!DOCTYPE html>
<html>
<head>
        <title></title>
</head>
<body>
<h2> Hola: <?php echo "Mi animal Favorito es el $animales[3]"  ?></h2>
</body>
</html>
```

```
<?php
$animales = ['Gato', 'Perro', 'Elefante', 'León'];
?>
<!DOCTYPE html>
<html>
<head>
        <title></title>
</head>
<body>
<h2> Hola: <?php echo "Mi animal Favorito es el $animales[3]"  ?></h2>
</body>
</html>
```

Switch PHP

Las instrucciones if...else if...else if permiten resolver prácticamente todas las decisiones que haya que tomar en programación, pero en ocasiones nos encontraremos con casos en que queremos evaluar condiciones con una estructura que hacen más cómodo usar una instrucción alternativa: switch.

Ejemplo:

```
Switch PHP
<?php
$animales = 'gato';
switch ($animales) {
case 'leon':
echo "El animal es leon"; break;
case 'perro':
echo "El animal es perro";break;
                    case 'gato':
echo "El animal
es gato";break;
default:
echo "Código inválido";break;
```

Include

Include en php nos permite llamar con código php otro archivo php externo:

Ejemplo:

Index.php

```
<body>
        <?php include("header.php") ?>
```

En este ejemplo, se trabaja con layouts, y se puede invocar el código insertado desde otro archivo.

En este ejemplo, se trabaja con layouts, y se puede invocar el código insertado desde otro archivo.

Funciones en PHP:

Una de las herramientas más importantes en cualquier lenguaje de programación son las funciones. Una función es un conjunto de instrucciones que a lo largo del programa van a ser ejecutadas multitud de veces. Es por ello, que estos conjuntos de instrucciones se agrupan en una función. Las funciones pueden ser llamadas y ejecutadas desde cualquier punto del programa (Heurtel, 2016).

Ejemplo:

Vamos a realizar una función que me dé el promedio de 2 número insertados

```
fuction promedio($n1,$n2){ return ($n1+$n2)/2;
}
```

Imprimir en html o en el lugar que invoquemos la función:

Para imprimir en cualquier parte del documento:

```
<?php echo operaciones(5,4,"sumar"); ?>
```

Métodos GET y POST

Los métodos HTTP GET y HTTP POST permiten enviar información al servidor, en PHP se administra mediante los arrays $_GET y $_POST.

Características GET

La Información se envía de forma visible. El método GET envía la información codificada del usuario en el header del HTTP request, directamente en la URL. La página web y la información codificada se separan por un interrogante "?":
Ejemplo:

Características POST

La Información se envía de forma visible. El método GET envía la información codificada del usuario en el header del HTTP request, directamente en la URL. La página web y la información codificada se separan por un interrogante "?":
Ejemplo:

```
www.ejemplo.com/index.htm?key1=value1&key2=value2&key3=value3...
```

Características POST

Con el método HTTP POST también se codifica la información, pero ésta se envía a través del body del HTTP Request, por lo que no aparece en la URL.

```
<form method="post">
```

JAVASCRIPT

Como cualquier otro lenguaje de programación, JavaScript tiene algunas características especiales: sintaxis, modelo de objetos, etc. Claramente, cualquier cosa que diferencia un lenguaje de otro. Además, descubrirás rápidamente que JavaScript es un lenguaje relativamente especial en su acercamiento a las cosas.

Introducción a JavaScript

JavaScript es un lenguaje de programación de scripts (secuencia de comandos) orientado a objetos. Esta descripción es un poco rudimentaria, hay varios elementos que vamos a diseccionar.

Características JavaScript

Es un lenguaje de programación Web

Añade interactividad a los sitios web

- Desarrollo Front End
- Desarrollo Back End
- Desarrollo de Aplicaciones Móviles y escritorio.

Scripts de programación

JavaScript te permite programar scripts. Como se mencionó anteriormente, un lenguaje de programación es utilizado para escribir código fuente a ser analizada por un ordenador. Hay tres formas de usar el código fuente. Además, permite reaccionar a eventos del usuario en las páginas web, validar formularios y procesar información (pagos, mostrar mapas, etc.)

Los scripts son en su mayoría interpretados. Y cuando decimos que JavaScript es un lenguaje interpretado, lo que significa que es un lenguaje interpretado. Por tanto, es necesario contar con un intérprete para ejecutar.

Primeros pasos en JavaScript

JavaScript es un lenguaje utilizado principalmente con el lenguaje HTML, además es necesario aprender cómo integrar este lenguaje en tus páginas web, descubrir su sintaxis básica y mostrar un mensaje en la pantalla del usuario.

Cuadro de diálogo JavaScript

¡No se deroga la regla tradicional de que todos los tutoriales de programación comenzarán mostrando el texto "Hello World!", ("¡Hola Mundo!" en español) al usuario. A continuación, se muestra un programa HTML simple que contiene la instrucción JavaScript, situada dentro de un elemento <script>:

```
<!DOCTYPE html>
<html>
        <head>
        <title>Hello World!</title>
        </head>
<body>
        <script>
        alert('Hello world!');
        </script>
</body>
</html>
```

Apareciendo al ejecutarlo la siguiente caja de diálogo:

El cuadro de diálogo de alert ()

Alert () es una declaración simple, llamada función, que muestra un cuadro de diálogo que contiene un mensaje. este mensaje se coloca entre comillas, entre los paréntesis de la function alert ().

Sintaxis de JavaScript

Instrucción:

La sintaxis de Javascript no es complicada. Generalmente, las instrucciones deben estar separadas por un punto y coma que se coloca al final de cada instrucción:

Código: JavaScript

```
sentencia_1; sentencia_2; sentencia_3;
```

Algunas secuencias de comandos están disponibles en un formato comprimido, es decir, todo el código se escribe como secuencia, no hay retornos de línea. Esto reduce considerablemente el tamaño de una secuencia de comandos y sirve para asegurar que la página se carga más rápidamente. Existen programas para "comprimir" el código JavaScript. Pero si has olvidado un solo punto y coma el código comprimido no va a funcionar porque las instrucciones no están debidamente separadas. También es una de las las razones por las que siempre se pone el punto y coma al final de la instrucción.

Espacios

JavaScript no es sensible a los espacios. Esto significa que puedes alinear las instrucciones que quieras, siempre que no interfiera con la secuencia de comandos. Por ejemplo, esto es correcto.

Código JavaScript

```
instruccion_1;
instruccion_1_1;
instruccion_1_2;
instruccion_2;
instruccion_3;
```

Sangría y presentación

La sangría en la programación informática, es una manera de estructurar el código para hacerlo más legible. Las instrucciones son priorizadas en varios niveles y espacios de usos o lengüetas para desplazar a la derecha y crear una jerarquía. Un ejemplo de código sangrado (Navarrete, 2006):

Código: JavaScript

```
function interruptor(elemID) {
var elem =
document.getElementById(elemID);

if
    (elem.style.displ
    ay =='block') {
    elem.style.displ
    ay = 'none';
} else {
elem.style.display = 'block';

  }
}
```

Variables

Para declarar una variable, simplemente hay que escribir la siguiente.

línea:

Ejemplo:

```
var miVariable;
```

JavaScript es un lenguaje sensible en las denominaciones, ten cuidado de no confundir las mayúsculas y minúsculas. En el siguiente ejemplo, tenemos tres variables diferentes declaradas:

Ejemplo

```
var miVariable;
var mivariable;
var MIVARIABLE;
```

La palabra clave var está presente para indicar que se declara una variable. Una vez que se declara, se puede almacenar lo que quieras:

Ejemplo:

```
var miVariable;
miVariable = 2;
```

El signo = se utiliza para asignar un valor a la variable, aquí le hemos asignado el número 2. Cuando das un valor a una variable, decimos que se trata de una asignación, ya que asigna un valor a la variable.

Frameworks Java

Un Framework es un esquema o patrón que nos ofrece un entorno genérico para escribir código en un lenguaje concreto dentro de una página web, además actúa como una plantilla o esqueleto, que un desarrollador puede usar y reutilizar para crear una aplicación completando su código según sea necesario para que la aplicación funcione como se pretende que se denomine Framework. La reutilización de frameworks permite a los desarrolladores programar su aplicación sin la sobrecarga manual de crear cada línea de código desde cero (Aponte, 2014).

Un framework de trabajo en Java es específico del lenguaje de programación Java, utilizado como plataforma para desarrollar aplicaciones de software y programas Java. Además, los frameworks de trabajo de Java pueden incluir clases y funciones predefinidas utilizadas para procesar, ingresar y administrar dispositivos de hardware, así como interactuar con el software del sistema. Depende del tipo de framework, el nivel de habilidad del programador, lo que está tratando de lograr y sus preferencias.

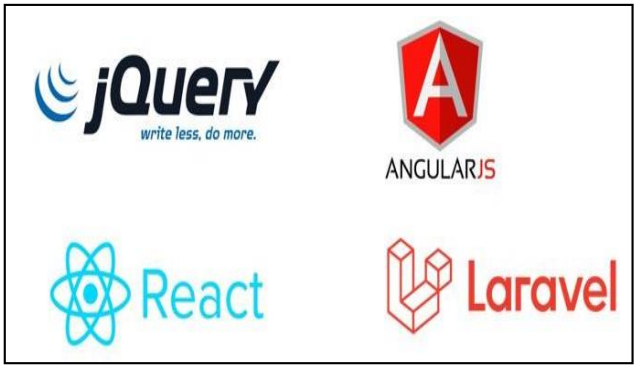
Ventajas

Permite automatizar procesos, que en ocasiones pueden ser repetitivos:

Conexión a las bases de datosLlamado a servicios Web. Permite crear Software de mayor complejidad.

Figura 15

Frameworks Front End



Fuente: Elaboración propia

Evaluación Unidad IV.

Explica la diferencia entre mysql_connect(), mysqli_connect(), y PDO en PHP. ¿Cuál es la mejor opción y por qué?

¿Qué son las consultas preparadas en MySQL y cómo ayudan a prevenir inyecciones SQL en PHP? Proporciona un ejemplo de código.

Describe el proceso de conectar a una base de datos MySQL usando mysqli en PHP. ¿Cuáles son los pasos principales y qué funciones se utilizan?

¿Cómo puedes manejar errores al ejecutar una consulta MySQL en PHP? Proporciona un ejemplo donde se verifique y maneje un error al realizar una consulta.

¿Qué es PDO en PHP y cuáles son sus principales ventajas sobre otras extensiones de base de datos como mysqli?

¿Explica cómo puedes utilizar transacciones en MySQL con PHP? ¿Por qué son útiles las transacciones, y en qué casos las utilizarías?

¿Cómo puedes proteger tu aplicación PHP contra inyecciones SQL al trabajar con formularios que envían datos a una base de datos MySQL?

Describe cómo se puede realizar una operación de búsqueda en una base de datos MySQL utilizando PHP. ¿Cómo manejas los resultados y los errores que puedan surgir?

Explica el concepto de "hoisting" en JavaScript. ¿Cómo afecta la declaración y inicialización de variables y funciones?

¿Qué es el "scope" en JavaScript? Diferencia entre el "scope" global y el "scope" local, y proporciona ejemplos de cómo se comportan.

¿Cuál es la diferencia entre == y === en JavaScript? Explica con ejemplos cuándo es preferible usar cada uno.

¿Qué son las "promesas" en JavaScript y cómo se utilizan? Proporciona un ejemplo de cómo manejar operaciones asincrónicas utilizando promesas.

Describe el funcionamiento del modelo de "event loop" en JavaScript. ¿Cómo gestiona JavaScript las operaciones asincrónicas?

¿Qué es el "closure" en JavaScript? Explica su importancia con un ejemplo práctico.

¿Cómo funcionan los módulos en JavaScript? Explica cómo se pueden importar y exportar funciones o variables entre archivos.

¿Qué son las funciones de flecha (arrow functions) en JavaScript y cómo difieren de las funciones tradicionales? Proporciona ejemplos de su uso.

05

SEGURIDAD WEB

Generalidades

La seguridad web a raíz de la pandemia del COVID-19 se ha visto vulnerada, debido al incremento en las intrusiones en las aplicaciones web según el Reporte de Ciberseguridad (BID-OEA, 2020, p. 16). Los organismos European Union Agency for Cybersecurity (ENISA), ESET Security y EcuCert del Ecuador realizaron estudio actualizado sobre los riesgos latentes que pueden afectar a las infraestructuras tecnológicas y aplicaciones web a nivel mundial.

A continuación, se hace un análisis e identificación de las amenazas que afectan a las aplicaciones Web:

Tabla 16
Clasificación de las ciber amenazas de las aplicaciones web

Etapas de Seguridad en aplicaciones web	Ciber amenaza
Autenticación	Phishing Comunicación insegura
Disponibilidad	DDoS (denegación de servicios)
Confidencialidad	Manejo incorrecto de errores Fuga de Información
Integridad	SQLi (Inyección SQL) XSS (secuencia de comandos en sitios cruzados) PHPi (Inyección PHP) RFI (Remote File Inclusion) CMDi (Inyección de Comandos)

Fuente: (Morales & Chicaiza, 2021)

A continuación, se detallan las herramientas y técnicas más utilizadas actualmente para detección de vulnerabilidades en base al estudio de (Hernandez & Mejia, 2015):

A continuación, se detallan las herramientas y técnicas más utilizadas actualmente para detección de vulnerabilidades en base al estudio de (Hernandez & Mejia, 2015):

Black-box

Es una técnica basada para descubrir vulnerabilidades en aplicaciones web, probando la aplicación desde el punto de vista del atacante.

White-box

Está del lado del servidor. En este tipo de enfoque se tiene acceso a información relevante de la organización.

Análisis estático de código (auditoria de código fuente)

Es un método en el que no se requiere ejecutar el programa, este realiza un análisis de código fuente directo para determinar huecos enla seguridad.

Análisis dinámico de código

Se comunica con la aplicación *web* a través de *front-end* de la aplicación en orden de identificar vulnerabilidades de seguridad potenciales y debilidades en la arquitectura de la aplicación web.

Pruebas de penetración

Consiste en la simulación de un ataque de los maliciosos *outssiders* (que no tienen un medio autorizado de acceder a los sistemas de la organización) y de maliciosos *insiders* (que tienen algún nivel de acceso autorizado). El proceso implica un análisis activo del sistema en busca de posibles vulnerabilidades.

Pruebas pasivas

Las pruebas pasivas están diseñadas para el análisis del tráfico de telecomunicaciones. Permite detectar fallas y defectos de seguridad mediante el examen de los paquetes capturados (livetrafficor log files).

Pruebas activas

Utiliza un programador de subprocesos asignados al azar para verificar si las advertencias comunicadas por un análisis predictivo de programa son errores reales.

Fuzz testing (pruebas de caja negra): Consiste en estimular el sistema bajo prueba, utilizando datos aleatorios o mutados queridos, con el fin de detectar comportamientos no deseados como violación de confidencialidad.

Publicación de sitios Web

Hosting y Alojamiento

El hosting sirve para alojar páginas web, aplicaciones de software y administradores de Email (CC Bogotá, 2014).

Dedicado: Aquí la empresa que le presta el servicio, le asigna y alquila un equipo o servidor de uso exclusivo de su empresa. Tipos de Hosting:

Colocación: Este es un servicio donde la empresa de hosting no proveeel servidor, por lo cual, sólo se encarga de la administración del servidory asignación del canal de comunicación.

Compartido: Es el servicio mediante el cual la empresa de hosting alquilael uso de una parte de su servidor.

Existen empresas que bridan el servicio de hosting para el alojamiento delas páginas web, además otorgan una dirección web (dominio). A continuación, se detallan las páginas web que brindan estos servicios:

Servicios de Dominio y hosting

Gratuitas

www.tonohost.com

www.000webhost.com

Pagadas .

www.hostinger.es

www.ecuahosting.net

www.nic.ec

www.godaddy.com

Certificados SSL

SSL (Secure Sockets Layer) es un protocolo de propósito general para establecer comunicaciones seguras, propuesto en 1994 por Netscape Communications Corporation junto con su primera versión del *Navigator*. Sinembargo, no fue hasta su tercera versión, conocida como SSL v3.0 que alcanzó su madurez, superando los problemas de seguridad y limitaciones de sus predecesores. No es exclusivo del comercio electrónico, sino que sirve para cualquier comunicación vía Internet para transacciones económicas. Hoy constituye la solución de seguridad implantada en la mayoría de los servidores web que ofrecen servicios de comercio electrónico (Ortega Martorell & CaninoGutiérrez, 2006).

Gráfico 17

Certificados SSL

Fuente: Fuego Yámana 2022 — España

Funcionamiento SLL

En el modelo de referencia TCP/IP, SSL se introduce como una especie de nivel o capa adicional, situada entre la capa de aplicación y la capa de transporte (figura 1). Lo anterior hace que sea independiente de la aplicaciónque lo utilice, es decir, que no solo puede ser utilizado para encriptar la comunicación entre un navegador y un servidor Web, sino también en cualquier aplicación como IMAP, FTP, Telnet, etc. También puede aplicar algoritmos de compresión a los datos a enviar y fragmentar los bloques de tamaño mayor a 214 bytes, volviendo a reensamblarlos en el receptor.1 Además, SSL establece una comunicación segura a nivel de socket(nombre de máquina más puerto), de forma transparente al usuario y a las aplicaciones que lo usan.

Para utilizar un certificado SSL en tu sitio web, es de vital importancia que el servidor de Internet que contrates, soporte SSL. Fuego Yámana, en alianza con Bee Web Hosting, lo ofrece sin cargo en todos los planes disponibles. Si tu empresa quiere conseguir este certificado, nuestros expertos se encargarán de gestionar con la Autoridad de Certificación Correspondiente (CA o Certification Authority) para validar dicha garantía de seguridad según los distintos tipos de Certificados SSL existentes.

Gráfico 18

Función SLL

Fuente: Fuego Yámana 2022 — España

Evaluación Unidad V.

¿Qué es el HTTPS y por qué es importante utilizarlo en sitios web?

Describe cómo funciona un ataque de Cross-Site Scripting (XSS) y qué medidas se pueden tomar para prevenirlo.

Explica qué es un ataque de SQL Injection y cómo se puede mitigar.

¿Cuáles son las diferencias entre un firewall y un Web Application Firewall (WAF)? ¿En qué situaciones usarías cada uno?

¿Qué es la autenticación multifactor (MFA) y cómo contribuye a la seguridad de una aplicación web?

Describe los riesgos asociados con el uso de contraseñas débiles y cómo pueden mitigarse.

Explica la importancia de la gestión de parches en la seguridad web y qué consecuencias puede tener no aplicarlos a tiempo.

¿Qué es la seguridad en la capa de transporte (TLS) y cómo protege la comunicación en una red?

¿Cómo funciona el Content Security Policy (CSP) y cómo ayuda a mejorar la seguridad de una aplicación web?

Describe cómo funciona un ataque de Denegación de Servicio (DoS) y qué estrategias pueden emplearse para mitigarlo.

¿Qué es la gestión de vulnerabilidades y por qué es crucial en la seguridad de aplicaciones web?

Explica la importancia del principio de mínimo privilegio en la gestión de usuarios y permisos en sistemas web.

¿Qué son los tokens en el contexto de la autenticación web, y cómo contribuyen a la seguridad?

¿Cómo puedes asegurar que las cookies utilizadas en una aplicación web no representen un riesgo de seguridad?

Describe un ataque de phishing y cómo se pueden entrenar a los usuarios para evitar ser víctimas de estos ataques.

BIBLIOGRAFÍA.

Alava Santana, B. L. (2015). Diseño, Desarrollo E Implementación De Un Sitio Web Dinámico Utilizando El Cms Joomla Y Google Analytics Para La Maestría en Seguridad Informatica Aplicada Msia-Espol.

Aponte, Á. M. V. (2014). Guía comparativa de Frameworks para los lenguajes HTML 5, CSS y JavaScript para el desarrollo de aplicaciones Web (Doctoral dissertation, Universidad Tecnológica de Pereira. Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación. Ingeniería de Sistemas y Computación).

Arias, M. A. (2013). Introducción a PHP. IT Campus Academy.

CC Bogotá, C. (2014). Aplicaciones de internet útiles para la expansión de su negocio. Camara de Comercio Bogotá, 67. Retrieved from http://bibliotecadigital.ccb.org.co/bitstream/handle/11520/3071/2958_aplicaciones_de_internet_utiles_para_la_expansion_de_su_negocio.pdf?sequence=1

Eguíluz, J. (2009). Introducción a CSS.[En línea]. LIBROSWEB.[Consulta: jul 2009]. Disponible en Internet:<http://www. librosweb. es/css/index. html.

Equipo Vértice. (2009). Diseño básico de páginas web en HTML. Editorial Vértice.

Ferrer, J., García, V., & García, R. (2014). Curso completo de HTML. 9 Mar. 1998, 447. Retrieved from http://es.tldp.org/Manuales-LuCAS/doc-curso-html/doc-curso-html.pdf.

Guijarro, Á. P. (2012). Protocolo HTTP. Teoría de SRI, (66), 54. Retrieved from http://bibing.us.es/proyectos/abreproy/11372/fichero/Memoria%252F05+-+Protocolo+HTTP.pdf%0Ahttp://bibing.us.es/proyectos/abreproy/11372/fichero/Memoria%7B%25%7D2F05+-+Protocolo+HTTP.pdf

Hayder, H. (2006). WordPress Complete. In *Image Rochester NY*. Retrieved from http://www.packtpub.com/wordpress/book

Hernandez, A., & Mejia, J. (2015). Guía de ataques, vulnerabilidades, técnicas y herramientas para aplicaciones web. *Revista Electrónicade Computación, Informática Biomédica y Electrónica*, *4*(1). Retrieved from https://bit.ly/2RMLfFD.

Heurtel, O. (2016). PHP 7: Desarrollar un sitio web dinámico e interactivo.Ediciones ENI.

Lerma-Blasco, Raül V., Murcia Andrés, José Alfredo; Mifsud Talón, E. (2014).Aplicaciones web. In *Aplicaciones web*.

Luján Mora, S. (2002). *Programación de aplicaciones web: historia, principios básicos y clientes web*. Retrieved from http://rua.ua.es/dspace/handle/10045/16995

Mazier, D. (2015). Joomla! 3.3: cree y administre sus sitios Web. EdicionesEni.

Minera, F. (2011). *Desarrollador Web* (p. 400). p. 400. RedUSERS. Navarrete, T. (2006). El lenguaje JavaScript. PROPUESTAS Macroproceso:

Gestión Académica Versión, 2, 1. Ortega Martorell, S., & Canino utiérrez, L. (2006). Protocolo de seguridad SSL. *Ingeniería Industrial*, *XXVII*(2–3), 57–62.

Pérez, J. E. (2019). introduccion a JavaScript. Rodríguez, J. M. S. (2018). Empezando con WordPress. *FSI - Formación de Seguridad Informática*, 1–20.

Schulz, R. G. (2008). Diseño web con CSS. Marcombo Souza, A. L. De, Célia, R., & Freitas, M. (2009). *HTTPxHTTPS*. 85–92.

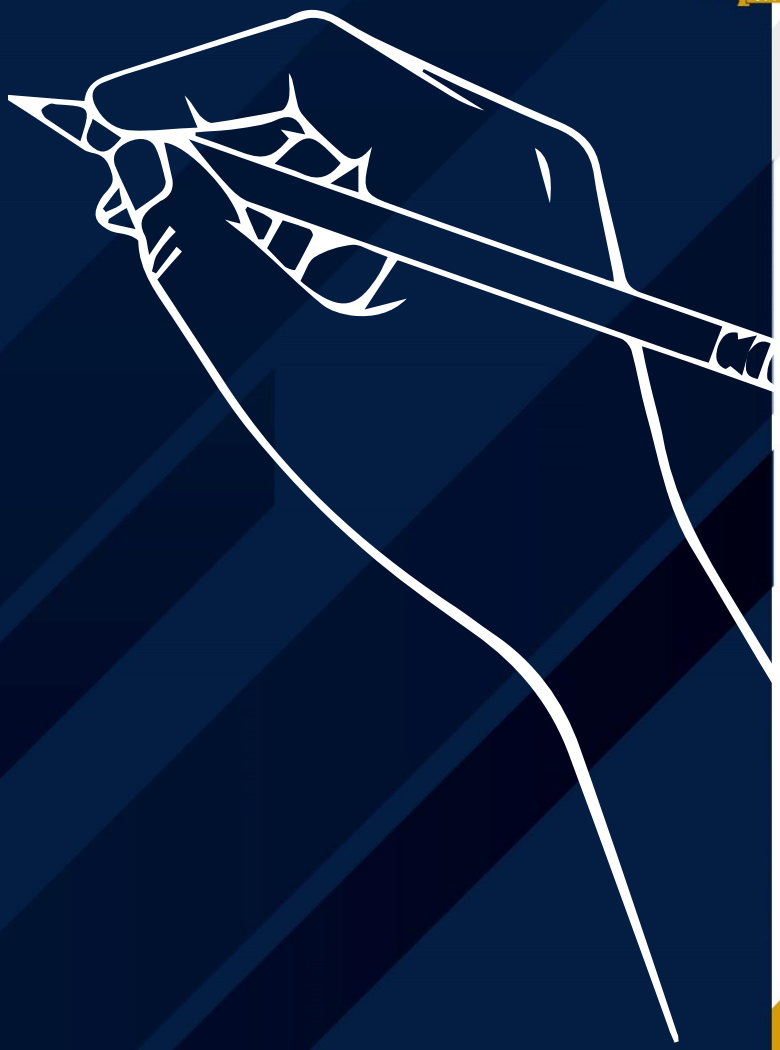
INSTITUTO SUPERIOR
TECNOLÓGICO PELILEO

TOMO 2:

Desarrollo Móvil

CONTENIDOS

01



INTRODUCCIÓN AL DESARROLLO DE APLICACIONES

SISTEMAS OPERATIVOS MÓVILES

Figura 1
Dispositivos móviles

Los sistemas operativos móviles son sistemas operativos diseñados específicamente para dispositivos móviles, como teléfonos inteligentes, tabletas y relojes inteligentes. Estos sistemas operativos están optimizados para la movilidad y la interacción táctil, y proporcionan una plataforma para que las aplicaciones móviles se ejecuten de manera eficiente. Algunos de los sistemas operativos móviles más populares hasta mi última actualización en septiembre de 2021 incluyen:

Android: Desarrollado por Google, Android es uno de los sistemas operativos móviles más utilizados en todo el mundo. Se utiliza en una amplia variedad de dispositivos de diferentes fabricantes y es conocido por su flexibilidad y personalización.

iOS: Desarrollado por Apple, iOS es exclusivo para dispositivos iPhone, iPad y iPod Touch. Es conocido por su diseño elegante y su ecosistema cerrado que incluye la App Store.

iOS/iPadOS: A partir de iOS 13, Apple introdujo una versión específica para iPad llamada iPadOS, que ofrece características adicionales y una experiencia más orientada a la productividad en tabletas.

Windows 10 Mobile: Microsoft tenía un sistema operativo móvil llamado Windows 10 Mobile, pero discontinuó su desarrollo en 2017 y ya no se encuentra en uso.

KaiOS: KaiOS es un sistema operativo ligero diseñado para teléfonos móviles básicos y está ganando popularidad en mercados emergentes.

Tizen: Desarrollado principalmente por Samsung y respaldado por otros fabricantes, Tizen se utiliza en algunos de los dispositivos de la marca, como relojes inteligentes y televisores.

HarmonyOS: Desarrollado por Huawei, HarmonyOS es un sistema operativo multiplataforma diseñado para una variedad de dispositivos, incluidos teléfonos inteligentes, tabletas, televisores y otros dispositivos inteligentes.

Sailfish OS: Este sistema operativo móvil de código abierto se utiliza en algunos dispositivos de Jolla y otros fabricantes. Ofrece una experiencia de usuario única y es conocido por su enfoque en la privacidad.

Android.

Android es un sistema operativo móvil desarrollado por Google. Es uno de los sistemas operativos móviles más populares y ampliamente utilizados en el mundo. Android se utiliza en una variedad de dispositivos, incluidos teléfonos inteligentes, tabletas, relojes inteligentes, televisores y dispositivos de automóviles.

Aquí hay algunas características y aspectos destacados de Android:

Código abierto: Android se basa en el núcleo de Linux y es un sistema operativo de código abierto. Esto significa que su código fuente está disponible para que los desarrolladores lo modifiquen y personalicen según sus necesidades.

Diversidad de dispositivos: Android se ejecuta en una amplia gama de dispositivos fabricados por diferentes compañías. Esto da lugar a una gran diversidad de modelos y opciones para los consumidores.

Google Play Store: La tienda de aplicaciones de Android, llamada Google Play Store, ofrece una vasta selección de aplicaciones y juegos para descargar. Los usuarios pueden acceder a miles de aplicaciones para diversas necesidades y gustos.

-

Personalización: Android permite una gran personalización de la interfaz de usuario y la experiencia del usuario. Los usuarios pueden cambiar los fondos de pantalla, agregar widgets, usar lanzadores de terceros y realizar otras modificaciones para adaptar su dispositivo a sus preferencias.

Notificaciones: Android tiene un sistema de notificaciones rico y altamente configurable que permite a los usuarios estar al tanto de las actualizaciones, mensajes y eventos importantes.

Integración de servicios de Google: Android está estrechamente integrado con los servicios de Google, como Gmail, Google Maps, Google Drive y otros. Esto facilita la sincronización de datos y la colaboración en la nube.

Actualizaciones: Google lanza regularmente nuevas versiones de Android con mejoras en la seguridad, el rendimiento y las características. Sin embargo, la disponibilidad de estas actualizaciones puede variar según el fabricante y el modelo del dispositivo.

Seguridad: Android ofrece características de seguridad como bloqueo de pantalla, cifrado de datos y protección contra malware.

iOS.

iOS es un sistema operativo móvil desarrollado por Apple Inc. Diseñado exclusivamente para los dispositivos de la marca, como el iPhone, el iPad y el iPod Touch. Aquí te proporciono una descripción general de iOS y sus principales características:

Ecosistema cerrado: iOS es conocido por su ecosistema cerrado y controlado por Apple. La compañía controla tanto el hardware como el software, lo que permite una integración más estrecha y un alto nivel de optimización.

Interfaz de usuario elegante: iOS se caracteriza por su diseño elegante y minimalista, con iconos y una interfaz de usuario limpia y fácil de usar. También incluye gestos intuitivos, como deslizar y pellizcar, para la navegación táctil.

App Store: La App Store de Apple es la tienda de aplicaciones oficial de iOS, que ofrece una amplia gama de aplicaciones y juegos de alta calidad. Apple revisa y aprueba todas las aplicaciones antes de que estén disponibles en la App Store, lo que ayuda a mantener un alto nivel de calidad y seguridad.

Actualizaciones regulares: Apple suele lanzar actualizaciones periódicas de iOS que incluyen mejoras de seguridad, correcciones de errores y nuevas funciones. Los usuarios de dispositivos iOS suelen recibir estas actualizaciones de manera oportuna.

Seguridad y privacidad: Apple pone un gran énfasis en la seguridad y la privacidad de los datos de sus usuarios. Características como Touch ID y Face ID ofrecen métodos de autenticación seguros. Además, Apple ha implementado medidas de privacidad, como la opción de limitar el seguimiento de aplicaciones.

Sincronización con iCloud: iCloud es el servicio de almacenamiento en la nube de Apple que permite a los usuarios almacenar datos como fotos, contactos, calendarios y documentos, y sincronizarlos automáticamente entre dispositivos iOS y otros dispositivos Apple.

Siri: Siri es el asistente virtual de Apple que permite a los usuarios realizar tareas mediante comandos de voz. Siri puede responder preguntas, controlar dispositivos domésticos inteligentes y ejecutar tareas diversas.

Multitarea: iOS admite la multitarea en iPads y iPhones más recientes, lo que permite a los usuarios ejecutar múltiples aplicaciones al mismo tiempo y cambiar entre ellas de manera eficiente.

Desarrollo de aplicaciones: Para desarrollar aplicaciones para iOS, los desarrolladores utilizan el lenguaje de programación Swift o Objective-C, junto con el kit de desarrollo de software (SDK) de iOS proporcionado por Apple.

Universal Windows Platform

La Plataforma Universal de Windows (Universal Windows Platform o UWP) es una plataforma de desarrollo de software creada por Microsoft. Está diseñada para permitir a los desarrolladores crear aplicaciones que funcionen en una variedad de dispositivos con Windows 10 y versiones posteriores del sistema operativo Windows. La idea detrás de UWP es proporcionar una única base de código que pueda adaptarse y ejecutarse en diferentes tipos de dispositivos Windows, como PC, tabletas, teléfonos, consolas Xbox, HoloLens y más.

Un solo código fuente: Con UWP, los desarrolladores pueden escribir una sola aplicación que se adapta automáticamente a diferentes tamaños de pantalla y tipos de dispositivos. Esto simplifica el proceso de desarrollo y reduce la necesidad de mantener múltiples versiones de una aplicación para dispositivos específicos.

Adaptabilidad: Las aplicaciones UWP pueden adaptarse a la pantalla, la resolución y la orientación del dispositivo en el que se ejecutan. Esto garantiza una experiencia de usuario coherente en una amplia variedad de dispositivos.

Plataforma de desarrollo común: UWP utiliza tecnologías de desarrollo familiares, como XAML (Lenguaje Extensible de Aplicaciones de Marcado) y C#, lo que facilita que los desarrolladores con experiencia en el ecosistema de Microsoft creen aplicaciones.

Distribución a través de la Microsoft Store: Las aplicaciones UWP se pueden distribuir a través de la Microsoft Store, lo que facilita su descubrimiento y descarga por parte de los usuarios. También se pueden actualizar automáticamente a través de la tienda.

Acceso a características de Windows: UWP permite a las aplicaciones acceder a las características y funcionalidades de Windows, como notificaciones, Live Tiles, integración con Cortana (el asistente virtual de Microsoft), acceso a hardware específico y más.

Seguridad: UWP se ejecuta en un entorno de seguridad aislado, lo que significa que las aplicaciones tienen un acceso limitado al sistema y los datos del usuario. Esto ayuda a proteger la seguridad y la privacidad de los usuarios.

Soporte para múltiples idiomas: UWP facilita la localización y la internacionalización de aplicaciones, lo que permite a los desarrolladores ofrecer sus aplicaciones en varios idiomas y regiones.

- **Desarrollo para Realidad Mixta:** UWP también es compatible con dispositivos de realidad mixta,

Tiendas virtuales

Las tiendas virtuales en aplicaciones móviles son una extensión de las tiendas en línea que ofrecen una experiencia de compra optimizada y personalizada para los usuarios de dispositivos móviles, como teléfonos inteligentes y tabletas. Estas aplicaciones permiten a los clientes buscar, ver y comprar productos o servicios de una manera más conveniente y adaptada a sus dispositivos móviles. A continuación, se mencionan algunos aspectos clave de las tiendas virtuales en aplicaciones móviles:

Interfaz de usuario adaptada a dispositivos móviles: Las aplicaciones móviles para tiendas virtuales están diseñadas específicamente para pantallas táctiles y tamaños de pantalla más pequeños. Esto permite una experiencia de usuario más fluida y fácil de usar en comparación con los sitios web móviles.

Navegación intuitiva: Las aplicaciones móviles suelen presentar una navegación simplificada y una estructura de menú intuitiva para facilitar a los usuarios la búsqueda de productos, la exploración de categorías y la realización de compras.

Notificaciones y actualizaciones en tiempo real: Las aplicaciones móviles pueden enviar notificaciones push a los usuarios para informarles sobre promociones, descuentos, ventas flash o actualizaciones de productos en tiempo real. Esto puede aumentar la participación de los clientes.

Autenticación y pago simplificados: Las aplicaciones móviles a menudo permiten a los usuarios iniciar sesión de forma rápida y segura utilizando métodos como la huella dactilar o el reconocimiento facial. También pueden almacenar información de pago de manera segura para acelerar el proceso de pago.

Funciones de geolocalización: Al aprovechar la geolocalización, las aplicaciones móviles pueden proporcionar a los usuarios información relevante basada en su ubicación actual, como la disponibilidad de productos en tiendas físicas cercanas.

Integración con funciones del dispositivo: Las aplicaciones móviles pueden utilizar funciones del dispositivo, como la cámara para escanear códigos de barras o QR, el micrófono para realizar búsquedas por voz y la capacidad de notificación de Bluetooth para mejorar la experiencia del usuario.

Capacidad para comprar en cualquier momento y lugar: Las aplicaciones móviles permiten a los usuarios comprar productos desde cualquier lugar, lo que aumenta la conveniencia y la accesibilidad de la tienda virtual.

Análisis de datos y personalización: Las tiendas virtuales en aplicaciones móviles pueden recopilar datos sobre el comportamiento del usuario y utilizar esta información para personalizar las recomendaciones de productos y mejorar la experiencia del cliente.

Compatibilidad multiplataforma: Para llegar a la mayor audiencia posible, las tiendas virtuales a menudo desarrollan aplicaciones tanto para dispositivos iOS como Android, ya que estos son los dos sistemas operativos móviles más populares.

Actualizaciones y soporte continuos: Mantener la aplicación móvil actualizada con nuevas características, correcciones de errores y mejoras de seguridad es esencial para garantizar una experiencia de usuario óptima.

Evaluación Unidad I.

Desarrollo Frontend y Backend:

Explica las responsabilidades del desarrollo frontend y backend en el contexto del desarrollo de aplicaciones web.

¿Cuál es la diferencia entre HTML, CSS y JavaScript? Proporciona ejemplos de su uso en el desarrollo web.

Define qué es el desarrollo de aplicaciones y explica su importancia en la industria de la tecnología.

Explica las etapas del ciclo de vida del desarrollo de software y proporciona un ejemplo para cada etapa.

Lenguajes de Programación y Plataformas:

Enumera y describe brevemente al menos cinco lenguajes de programación comúnmente utilizados en el desarrollo de aplicaciones.

¿Cuál es la diferencia entre una aplicación de escritorio y una aplicación web? Proporciona un ejemplo de cada una.

Bases de Datos:

Define qué es una base de datos y enumera al menos tres tipos diferentes de bases de datos.

¿Qué es SQL y para qué se utiliza en el desarrollo de aplicaciones? Proporciona un ejemplo de una consulta SQL básica.

Pruebas y Depuración:

Explica la importancia de las pruebas en el desarrollo de aplicaciones y enumera al menos tres tipos diferentes de pruebas que se pueden realizar.

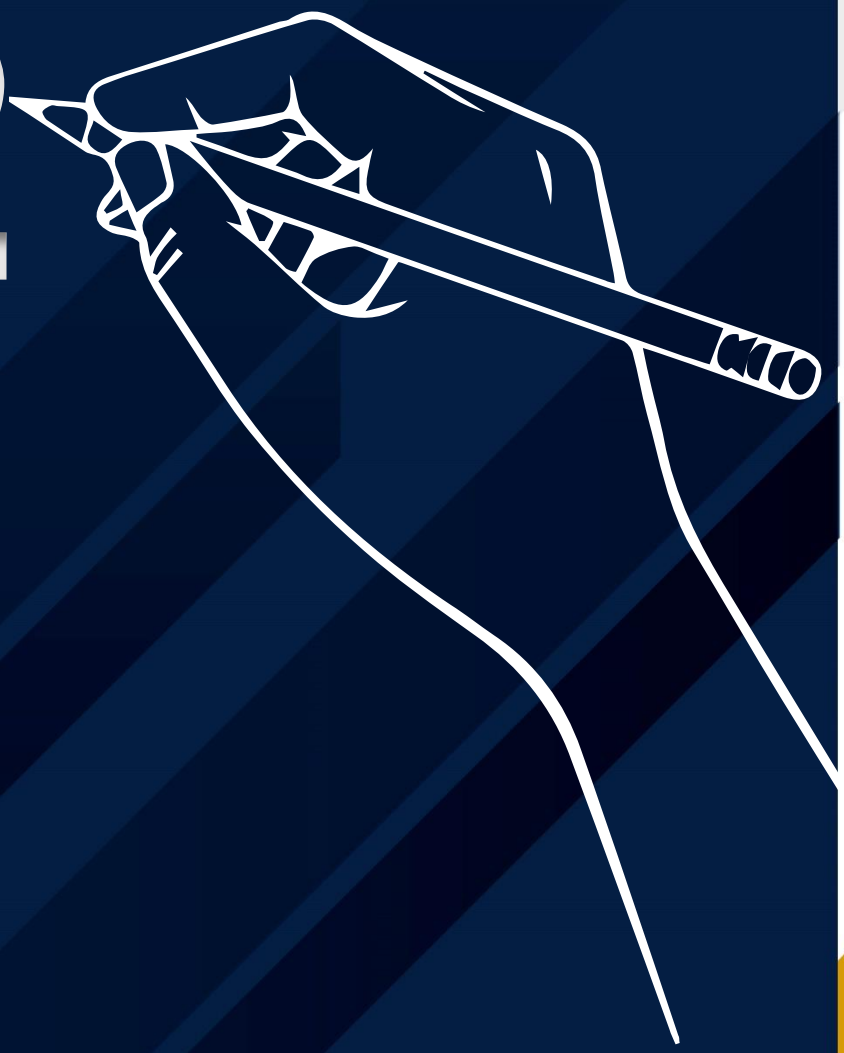
Describe el proceso de depuración de software y menciona algunas herramientas comunes utilizadas para depurar aplicaciones.

Despliegue y Mantenimiento:

¿Qué es el despliegue de software y por qué es importante en el desarrollo de aplicaciones?

Explica la importancia del mantenimiento de software y menciona algunas prácticas comunes utilizadas para mantener aplicaciones después de su lanzamiento inicial.

02



EMULADORES

Los emuladores son programas o dispositivos que permiten ejecutar software diseñado para una plataforma en un entorno diferente. En esencia, emulan el comportamiento del hardware y el sistema operativo de una plataforma específica para que un sistema diferente pueda ejecutar software diseñado originalmente para el primero. Los emuladores son utilizados en una variedad de contextos, desde el desarrollo de software hasta el juego retro. Aquí hay algunos tipos comunes de emuladores:

Emuladores de consola de videojuegos: Estos emuladores permiten a los usuarios ejecutar juegos de consola en una computadora u otro dispositivo que no sea la consola original. Algunos ejemplos populares incluyen emuladores de Super Nintendo (SNES), Sega Genesis y PlayStation.

Emuladores de teléfonos móviles: Estos emuladores permiten a los desarrolladores probar aplicaciones y juegos diseñados para dispositivos móviles en una computadora. Android Studio, por ejemplo, incluye un emulador de Android para este propósito.

Emuladores de sistemas operativos: Algunos emuladores permiten ejecutar sistemas operativos completos en una máquina virtual dentro de otro sistema operativo. VirtualBox y VMware son ejemplos de programas que permiten emular sistemas operativos, lo que es útil para probar diferentes versiones de sistemas operativos o ejecutar software que no es compatible con el sistema anfitrión.

Emuladores de hardware antiguo: Estos emuladores permiten a los usuarios ejecutar software diseñado para hardware antiguo en hardware moderno. Por ejemplo, DOSBox emula el entorno DOS para ejecutar programas y juegos de MS-DOS en sistemas modernos.

Emuladores de arcade: Estos emuladores permiten ejecutar juegos de arcade en una computadora o consola. MAME (Multiple Arcade Machine Emulator) es un ejemplo destacado que emula una amplia variedad de juegos de arcade.

Emuladores de computadoras vintage: Algunos emuladores permiten a los usuarios ejecutar sistemas informáticos antiguos, como el Commodore 64 o el Amiga, en hardware moderno.

Emuladores de red: Estos emuladores recrean las condiciones de red para probar cómo se comportan las aplicaciones y servicios en diversas situaciones de red, como conexiones lentas o caídas de red.

Emuladores de hardware específicos: Algunos emuladores se crean específicamente para emular hardware especializado, como chips de procesador específicos o tarjetas gráficas, para fines de desarrollo y pruebas.

Emuladores de sistemas operativos alternativos: Algunos emuladores permiten ejecutar sistemas operativos diferentes en un dispositivo que originalmente no es compatible con ellos. Por ejemplo, Wine es un emulador de Windows que permite ejecutar aplicaciones de Windows en sistemas Linux.

Es importante destacar que el uso de emuladores puede estar sujeto a restricciones legales, especialmente cuando se trata de emuladores de software o hardware con derechos de autor. Por lo tanto, es fundamental respetar las leyes de propiedad intelectual y licencias de software al utilizar emuladores.

Emuladores Android en Visual Studio.

Visual Studio, el entorno de desarrollo integrado (IDE) de Microsoft, ofrece herramientas y extensiones que permiten a los desarrolladores crear y probar aplicaciones de Android mediante emuladores. Esto es útil para el desarrollo de aplicaciones móviles en el entorno de Visual Studio. A continuación, te proporciono información sobre cómo usar emuladores de Android en Visual Studio:

Instala Visual Studio: Asegúrate de tener Visual Studio instalado en tu computadora. Puedes descargar la última versión de Visual Studio desde el sitio web oficial de Microsoft.

Instala las herramientas de desarrollo para Android: Para desarrollar aplicaciones de Android en Visual Studio, debes instalar las herramientas de desarrollo adecuadas. Esto incluye el Android SDK (Software Development Kit) y el Android Emulator. Puedes instalar estas herramientas utilizando el Administrador de Instalación de Visual Studio.

Crea un proyecto de Android: Abre Visual Studio y crea un nuevo proyecto de Android utilizando las plantillas proporcionadas. Puedes elegir entre diferentes tipos de proyectos, como aplicaciones nativas de Android o aplicaciones Xamarin, según tus necesidades.

Configura un emulador de Android: Una vez que tengas un proyecto de Android en Visual Studio, puedes configurar un emulador de Android para probar tu aplicación. Puedes hacerlo siguiendo estos pasos:En Visual Studio, ve a "Herramientas" -> "Administrador de Android SDK" para abrir la ventana de configuración del SDK de Android.Desde allí, puedes descargar las imágenes del sistema y las herramientas de emulación que necesitas.

Depura y prueba tu aplicación: Ahora puedes compilar, depurar y probar tu aplicación de Android en el emulador desde Visual Studio. Puedes establecer puntos de interrupción, inspeccionar variables y realizar pruebas de rendimiento directamente en el emulador.

Prueba en dispositivos reales: Además de los emuladores, también puedes conectar dispositivos Android reales a tu computadora y probar tu aplicación en ellos. Visual Studio admite la depuración y la implementación en dispositivos físicos.

Genymotion.

Genymotion es una plataforma de virtualización de dispositivos móviles que se utiliza principalmente para probar y desarrollar aplicaciones móviles en diferentes dispositivos Android. A diferencia de algunos emuladores de Android tradicionales, Genymotion se centra en la velocidad y la eficiencia, y ofrece una experiencia más rápida y fluida para el desarrollo y la prueba de aplicaciones. A continuación, se describen algunos aspectos clave de Genymotion:

Virtualización de dispositivos Android: Genymotion permite a los desarrolladores crear y ejecutar instancias virtuales de dispositivos Android en su computadora. Esto es útil para probar aplicaciones en una variedad de configuraciones de dispositivos sin necesidad de tener dispositivos físicos reales para cada uno.

Amplia variedad de dispositivos y versiones de Android: Genymotion ofrece una amplia gama de perfiles de dispositivos Android, lo que permite a los desarrolladores elegir entre varios modelos de teléfonos y tabletas.

Integración con herramientas de desarrollo: Genymotion se integra con IDE populares como Android Studio y Eclipse, lo que permite a los desarrolladores ejecutar y depurar sus aplicaciones directamente en las instancias virtuales de Genymotion.

Rendimiento y velocidad: Una de las características destacadas de Genymotion es su rendimiento mejorado en comparación con algunos emuladores de Android tradicionales. Las instancias virtuales de Genymotion se ejecutan de manera más rápida y fluida, lo que acelera el proceso de desarrollo y prueba de aplicaciones.

Características avanzadas: Genymotion ofrece una variedad de características avanzadas, como la posibilidad de cambiar la orientación del dispositivo, simular llamadas y mensajes, emular sensores como el GPS y la cámara, y más. Esto facilita la simulación de una experiencia de usuario más completa.

Soporte multiplataforma: Genymotion es compatible con Windows, macOS y Linux, lo que permite a los desarrolladores trabajar en su plataforma preferida.

Versión gratuita y comercial: Genymotion ofrece una versión gratuita con características básicas y limitaciones, así como una versión comercial con características adicionales, soporte y capacidades empresariales.

Compatibilidad con aplicaciones de Google Play: Genymotion facilita la instalación de las aplicaciones de Google Play Store en las instancias virtuales, lo que es útil para probar aplicaciones que dependen de servicios de Google.

Emuladores iOS en Visual Studio

A partir de mi última actualización en septiembre de 2021, Visual Studio no es compatible de forma nativa con la emulación de dispositivos iOS. Esto se debe a que iOS es un sistema operativo propietario desarrollado por Apple, y Apple limita estrictamente el uso de sus sistemas operativos y hardware a dispositivos Apple reales.

Sin embargo, si deseas desarrollar aplicaciones para iOS en un entorno de desarrollo basado en Windows, existen algunas alternativas que pueden ayudarte a lograr tus objetivos:

Mac de desarrollo remoto: Una opción común es configurar un Mac de desarrollo remoto y utilizarlo junto con Visual Studio en tu PC con Windows. Puedes conectar Visual Studio a través de una red o usar servicios en la nube que ofrecen acceso a un Mac virtual para desarrollo de iOS. Algunas soluciones populares incluyen MacinCloud y MacStadium.

Xamarin: Xamarin es una plataforma de desarrollo de aplicaciones móviles que se integra con Visual Studio. Permite desarrollar aplicaciones iOS y Android utilizando el lenguaje de programación C# y compartir gran parte del código entre las dos plataformas. Para probar y depurar aplicaciones de iOS, aún necesitarás un Mac de desarrollo remoto o un Mac físico.

Visual Studio para Mac: Microsoft ofrece una versión de Visual Studio diseñada específicamente para desarrolladores de iOS y macOS llamada "Visual Studio para Mac." Aunque está optimizada para el desarrollo de aplicaciones iOS, es una aplicación independiente de Visual Studio para Windows y se ejecuta en macOS.

Apple Xcode en una máquina virtual: Si tienes acceso a una máquina virtual con macOS, puedes instalar Apple Xcode en ella y utilizarlo para desarrollar y emular aplicaciones de iOS. Esto requiere que tengas hardware compatible con macOS en tu máquina virtual.

Emuladores en Xamarin Studio

Xamarin es una plataforma de desarrollo de aplicaciones móviles que permite a los desarrolladores escribir aplicaciones nativas para iOS y Android utilizando el lenguaje de programación C#. En cuanto a los emuladores, Xamarin ofrece opciones para probar y depurar aplicaciones en dispositivos virtuales o físicos. Aquí te proporciono información sobre cómo utilizar emuladores en Xamarin.

Emuladores de Android en Xamarin: Xamarin Android Player (XAP): Anteriormente, Xamarin ofrecía su propio emulador llamado "Xamarin Android Player", que estaba diseñado específicamente para probar aplicaciones de Android. Sin embargo, a partir de mi última actualización en septiembre de 2021, Xamarin Android Player ya no es compatible y se ha descontinuado. Los desarrolladores de Xamarin Android suelen utilizar el Android Emulator proporcionado por Google como alternativa.

Android Emulator de Google: Puedes utilizar el Android Emulator proporcionado por Google para emular dispositivos Android en tu computadora. Puedes configurar y utilizar este emulador en Xamarin Studio (o Visual Studio con el complemento de Xamarin). El Android Emulator es ampliamente utilizado para probar aplicaciones Android en diversas versiones de Android y configuraciones de dispositivos.

Emuladores de iOS en Xamarin: En el caso de iOS, como mencioné anteriormente, necesitas un Mac de desarrollo remoto o una máquina macOS para compilar y emular aplicaciones iOS. Xamarin permite la depuración y ejecución en un dispositivo físico iOS conectado a través de una red local, pero la emulación de dispositivos iOS en Windows no es compatible directamente debido a las restriccionesde Apple.

Dispositivos físicos: Además de los emuladores, Xamarin te permite depurar y ejecutar aplicaciones directamente en dispositivos iOS y Android reales conectados a través de USB. Esto es útil para probar aplicaciones en hardware real y garantizar la compatibilidad.

Las capacidades de emulación pueden cambiar con el tiempo debido a las actualizaciones de Xamarin y las actualizaciones en los sistemas operativos Android e iOS. Por lo tanto, te recomiendo consultar la documentación oficial de Xamarin.

Evaluación Unidad II.

¿Qué es un simulador de aplicaciones móviles y para qué se utiliza?

Menciona al menos tres ventajas de utilizar un simulador de aplicaciones móviles durante el proceso de desarrollo.

¿Cuál es la diferencia principal entre un simulador de aplicaciones móviles y un emulador de aplicaciones móviles?

¿Cuáles son los sistemas operativos móviles más comunes para los que se ofrecen simuladores de aplicaciones?

¿Cuál es la importancia de probar una aplicación en múltiples dispositivos y sistemas operativos antes de su lanzamiento?

¿Cuáles son algunas de las características clave que deben tener los simuladores de aplicaciones móviles para ser efectivos en el proceso de desarrollo?

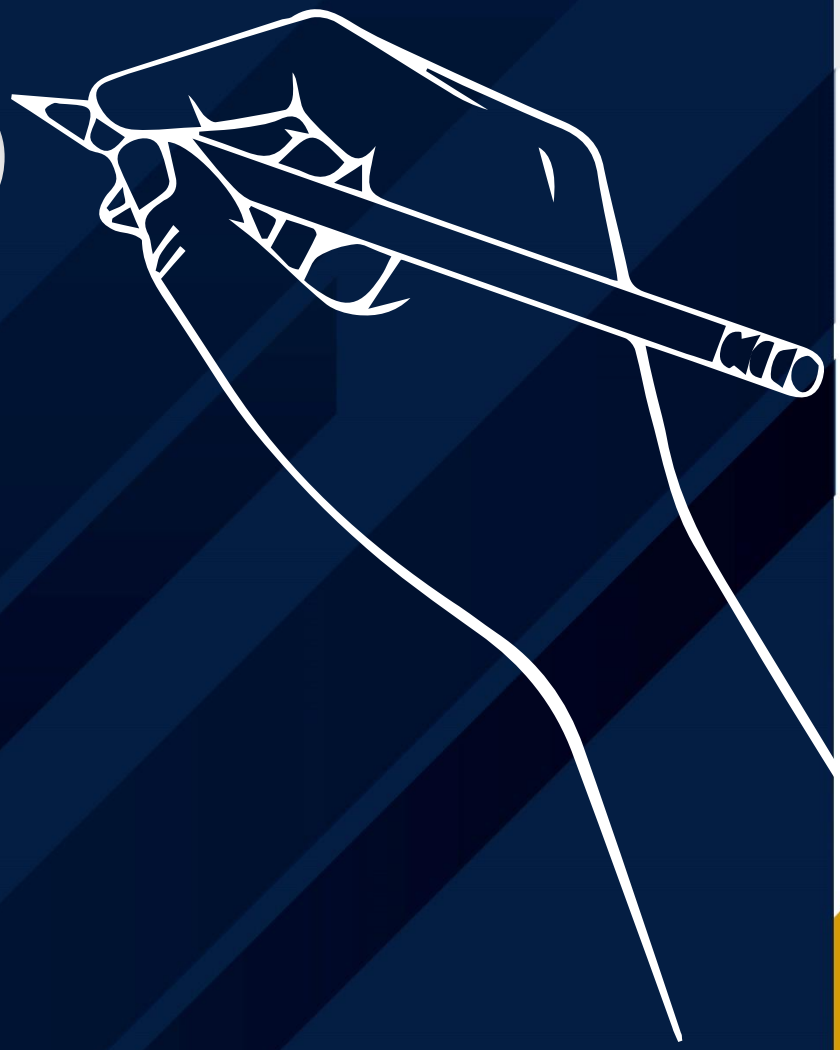
¿Cómo pueden los desarrolladores de aplicaciones móviles simular diferentes condiciones de red y hardware utilizando un simulador?

¿Cuál es el papel de las herramientas de depuración en los simuladores de aplicaciones móviles y por qué son importantes?

¿Qué precauciones deben tomar los desarrolladores al utilizar simuladores de aplicaciones móviles para garantizar que las pruebas sean representativas del comportamiento real de la aplicación?

¿Qué tipos de pruebas se pueden realizar con un simulador de aplicaciones móviles y cómo pueden ayudar en la mejora de la calidad de la aplicación?

03



LENGUAJES DE DESARROLLO PARA DISPOSITIVOS MÓVILES

Existen varios lenguajes de programación y entornos de desarrollo que se utilizan comúnmente para crear aplicaciones móviles para dispositivos como smartphones y tablets. Los lenguajes y entornos de desarrollo más populares para dispositivos móviles incluyen.

Java: Java es uno de los lenguajes de programación más utilizados para el desarrollo de aplicaciones Android. Se utiliza junto con el kit de desarrollo de software (SDK) de Android, que proporciona herramientas y bibliotecas para crear aplicaciones Android.

Kotlin: Kotlin es otro lenguaje de programación que se ha vuelto muy popular para el desarrollo de aplicaciones Android en los últimos años. Ofrece una sintaxis más moderna y concisa en comparación con Java y es totalmente compatible con las aplicaciones Android existentes.

Swift: Swift es el lenguaje de programación oficial de Apple para el desarrollo de aplicaciones iOS. Es altamente compatible con Objective-C y se utiliza para crear aplicaciones para iPhone, iPad y otros dispositivos Apple.

Objective-C: Aunque Swift ha ganado popularidad en los últimos años, Objective-C todavía se utiliza ampliamente para desarrollar aplicaciones iOS, especialmente aplicaciones heredadas o aquellas que requieren integración con código Objective-C existente.

JavaScript: JavaScript se utiliza para desarrollar aplicaciones móviles utilizando tecnologías web. Esto incluye aplicaciones web progresivas (PWAs) y frameworks como React Native y Apache Cordova (anteriormente conocido como PhoneGap), que permiten crear aplicaciones móviles multiplataforma utilizando HTML, CSS y JavaScript.

C#: C# se utiliza con el entorno de desarrollo Unity para crear juegos y aplicaciones 3D interactivas para dispositivos móviles, especialmente en la plataforma Unity3D.

Dart: Dart es el lenguaje de programación utilizado con el framework Flutter de Google. Flutter es una tecnología de código abierto que se utiliza para crear aplicaciones móviles multiplataforma con una única base de código.

Python: Aunque no es tan común como otros lenguajes, Python se puede utilizar para desarrollar aplicaciones móviles utilizando frameworks como Kivy o BeeWare.

Ruby: RubyMotion es un conjunto de herramientas que permite desarrollar aplicaciones móviles para iOS, Android y macOS utilizando Ruby como lenguaje de programación.

La elección del lenguaje de programación y el entorno de desarrollo depende en gran medida de la plataforma objetivo (Android, iOS o ambas) y de las preferencias personales del desarrollador. También es importante considerar las características y requisitos específicos de la aplicación que estás desarrollando al seleccionar la tecnología adecuada.

Java (Android)

Java ha sido uno de los lenguajes de programación más utilizados para el desarrollo de aplicaciones Android durante muchos años. Aquí tienes información clave sobre el uso de Java en el desarrollo de aplicaciones Android.

Android Studio: Android Studio es el entorno de desarrollo oficial de Android, proporcionado por Google. Permite a los desarrolladores crear aplicaciones Android utilizando Java (y Kotlin). Android Studio ofrece herramientas de desarrollo, un emulador de Android para probar aplicaciones y una variedad de recursos para facilitar la creación de aplicaciones.

SDK de Android: Para programar en Java para Android, debes utilizar el SDK de Android, que incluye una serie de bibliotecas y herramientas para acceder a las funcionalidades del sistema operativo Android.

Versiones de Java: Tradicionalmente, Android ha utilizado versiones de Java que eran ligeramente diferentes de las versiones estándar de Java. Sin embargo, con las últimas versiones de Android (Android 8 en adelante), se ha movido hacia el uso de Java 8 y funciones modernas del lenguaje.

Bibliotecas y Frameworks: Java en Android se utiliza junto con bibliotecas y frameworks específicos de Android para crear interfaces de usuario (por ejemplo, XML y Android XML Layouts) y para acceder a funciones como la cámara, el GPS, la base de datos SQLite y más.

Kotlin como Alternativa: Aunque Java sigue siendo una opción válida para el desarrollo de aplicaciones Android, Kotlin se ha vuelto cada vez más popular debido a su sintaxis más moderna, características de seguridad y una mayor eficiencia en el desarrollo.

Swift (iOS)

Swift es un lenguaje de programación desarrollado por Apple y se utiliza principalmente para el desarrollo de aplicaciones en los ecosistemas de Apple, como iOS, macOS, watchOS y tvOS. Aquí tienes información clave sobre Swift.

Desarrollo en iOS y otros ecosistemas de Apple: Swift se utiliza principalmente para desarrollar aplicaciones en plataformas de Apple, como iPhone, iPad, Apple Watch y Apple TV. Es el lenguaje de programación preferido para el desarrollo de aplicaciones móviles en iOS.

Sintaxis Moderna: Swift se diseñó con una sintaxis moderna y más legible en comparación con Objective-C, el lenguaje de programación anterior de Apple. Esto hace que el código Swift sea más claro y conciso.

Rendimiento y Seguridad: Swift se ha diseñado con un enfoque en el rendimiento y la seguridad. Ofrece una mayor velocidad de ejecución en comparación con Objective-C y también incluye características de seguridad que ayudan a prevenir errores comunes de programación.

Interoperabilidad con Objective-C: Swift es interoperable con Objective-C, lo que significa que puedes usar código Swift en proyectos que utilizan código Objective-C existente y viceversa. Esto facilita la transición gradual a Swift o la integración de nuevas características en aplicaciones existentes.

Multiplataforma: Aunque Swift se diseñó originalmente para el desarrollo en el ecosistema de Apple, existen proyectos y herramientas de código abierto, como Swift on Server, que permiten utilizar Swift en otros entornos y plataformas.

C# (UWP)

C# (pronunciado "C sharp") es un lenguaje de programación desarrollado por Microsoft y se utiliza principalmente en el desarrollo de aplicaciones para plataformas Windows. Una de las formas más comunes de utilizar C# para el desarrollo de aplicaciones de Windows es a través de la plataforma Universal Windows Platform (UWP). A continuación, te proporciono información clave sobre el uso de C# en aplicaciones UWP.

Universal Windows Platform (UWP): UWP es una plataforma de desarrollo de aplicaciones de Microsoft que permite crear aplicaciones que se ejecutan en una variedad de dispositivos con Windows 10, incluyendo PC, tabletas, teléfonos, Xbox y más. Estas aplicaciones están diseñadas para funcionar de manera uniforme en todos los dispositivos con Windows 10.

C# en UWP: C# es uno de los lenguajes de programación principales que se utiliza con UWP. Junto con XAML (Extensible Application Markup Language) para la creación de interfaces de usuario, C# permite desarrollar aplicaciones UWP con una interfaz de usuario moderna y una funcionalidad rica.

Entorno de Desarrollo: Para el desarrollo de aplicaciones UWP con C#, generalmente se utiliza Microsoft Visual Studio, que es el entorno de desarrollo integrado (IDE) oficial de Microsoft. Visual Studio proporciona herramientas de desarrollo, emuladores y depuración que facilitan el proceso de creación de aplicaciones UWP.

Bibliotecas y APIs de Windows: C# en UWP permite acceder a las bibliotecas y APIs de Windows para aprovechar las características del sistema operativo, como la administración de archivos, el acceso a la cámara, el uso de sensores y más.

C# es un lenguaje de programación versátil y potente que se utiliza con la plataforma UWP para el desarrollo de aplicaciones que se ejecutan en dispositivos con Windows 10.

XAML

XAML (Extensible Application Markup Language) es un lenguaje de marcado utilizado en el desarrollo de aplicaciones de interfaz de usuario, especialmente en plataformas de Microsoft, como Windows Presentation Foundation (WPF), Universal Windows Platform (UWP) y Xamarin.Forms. XAML se utiliza para definir la estructura y apariencia de la interfaz de usuario de una aplicación de manera declarativa, separando la lógica de la interfaz de usuario del código subyacente.

Declarativo: XAML es un lenguaje declarativo, lo que significa que defines la estructura y el diseño de la interfaz de usuario mediante etiquetas y atributos en lugar de escribir código imperativo. Esto facilita la creación y el mantenimiento de interfaces de usuario complejas.

Separación de la Lógica y la Interfaz de Usuario: XAML permite separar claramente la lógica de la aplicación de la interfaz de usuario. Puedes definir la apariencia y la estructura de la interfaz de usuario en archivos XAML y luego conectarte a esa interfaz de usuario desde el código subyacente en lenguajes como C#.

Multiplataforma: Aunque XAML se utiliza principalmente en el ecosistema de Microsoft, también se utiliza en otras plataformas a través de proyectos como Xamarin.Forms, que permite crear interfaces de usuario con XAML y C# para aplicaciones móviles multiplataforma en Android, iOS y UWP.

Diseño Responsivo: XAML incluye características para crear interfaces de usuario responsivas que se adaptan a diferentes tamaños y diferentes resoluciones de pantalla.

C# (Cross Platform)

El desarrollo multiplataforma, a menudo conocido como "cross-platform development", se refiere a la práctica de crear aplicaciones de software que pueden ejecutarse en múltiples plataformas o sistemas operativos sin necesidad de escribir código separado para cada plataforma. Esto permite a los desarrolladores maximizar la eficiencia al escribir una sola base de código que pueda utilizarse en diferentes dispositivos y sistemas operativos.

Frameworks de desarrollo multiplataforma: Estos frameworks proporcionan herramientas y bibliotecas que permiten a los desarrolladores crear aplicaciones que se ejecuten en múltiples plataformas. Algunos ejemplos populares son:

React Native: Utiliza JavaScript y React para desarrollar aplicaciones móviles para iOS y Android.

Flutter: Utiliza el lenguaje de programación Dart para crear aplicaciones móviles y de escritorio para diversas plataformas.

Xamarin: Utiliza C# y .NET para crear aplicaciones móviles para iOS, Android y otras plataformas.

Ionic: Utiliza tecnologías web como HTML, CSS y JavaScript para crear aplicaciones móviles.

PhoneGap (Apache Cordova): Permite el desarrollo de aplicaciones móviles utilizando HTML, CSS y JavaScript y proporciona acceso a las API nativas a través de un contenedor.

Desarrollo web progresivo (PWA): Las PWAs son aplicaciones web que se pueden ejecutar en cualquier dispositivo con un navegador web moderno. Utilizan tecnologías web estándar como HTML, CSS y JavaScript, pero ofrecen características de aplicaciones nativas, como notificaciones push y acceso sin conexión.

Juegos multiplataforma: Los motores de juego como Unity3D y Unreal Engine permiten a los desarrolladores crear juegos que se ejecuten en diversas plataformas, incluyendo PC, consolas, dispositivos móviles y VR.

Escritura única, implementación múltiple: Algunos lenguajes de programación, como C++ o Rust, permiten escribir código que se puede compilar para diferentes plataformas sin necesidad de reescribirlo por completo.

Las ventajas del desarrollo multiplataforma incluyen un menor costo de desarrollo, una mayor velocidad de desarrollo y la posibilidad de llegar a una audiencia más amplia. Sin embargo, también puede haber limitaciones en cuanto a las capacidades y el rendimiento, especialmente en comparación con el desarrollo nativo.

La elección de la tecnología de desarrollo multiplataforma dependerá de factores como las necesidades de la aplicación, el presupuesto, la experiencia del equipo de desarrollo y la audiencia objetivo. Cada enfoque tiene sus ventajas y desventajas, y es importante evaluar cuál es el más adecuado para un proyecto específico.

Evaluación Unidad III.

¿Qué es un lenguaje de programación para dispositivos móviles y cuál es su función en el desarrollo de aplicaciones móviles?

Menciona al menos tres lenguajes de programación populares para el desarrollo de aplicaciones móviles y sus respectivas plataformas de destino.

¿Cuál es la diferencia entre un lenguaje de programación nativo y un lenguaje de programación multiplataforma en el contexto de desarrollo móvil?

¿Cuáles son las ventajas y desventajas de utilizar un lenguaje de programación nativo en comparación con un lenguaje de programación multiplataforma?

Explique el funcionamiento del lenguaje de programación Swift y su relación con el desarrollo de aplicaciones para dispositivos iOS.

¿Qué es Kotlin y por qué se ha vuelto popular como un lenguaje de programación para aplicaciones Android?

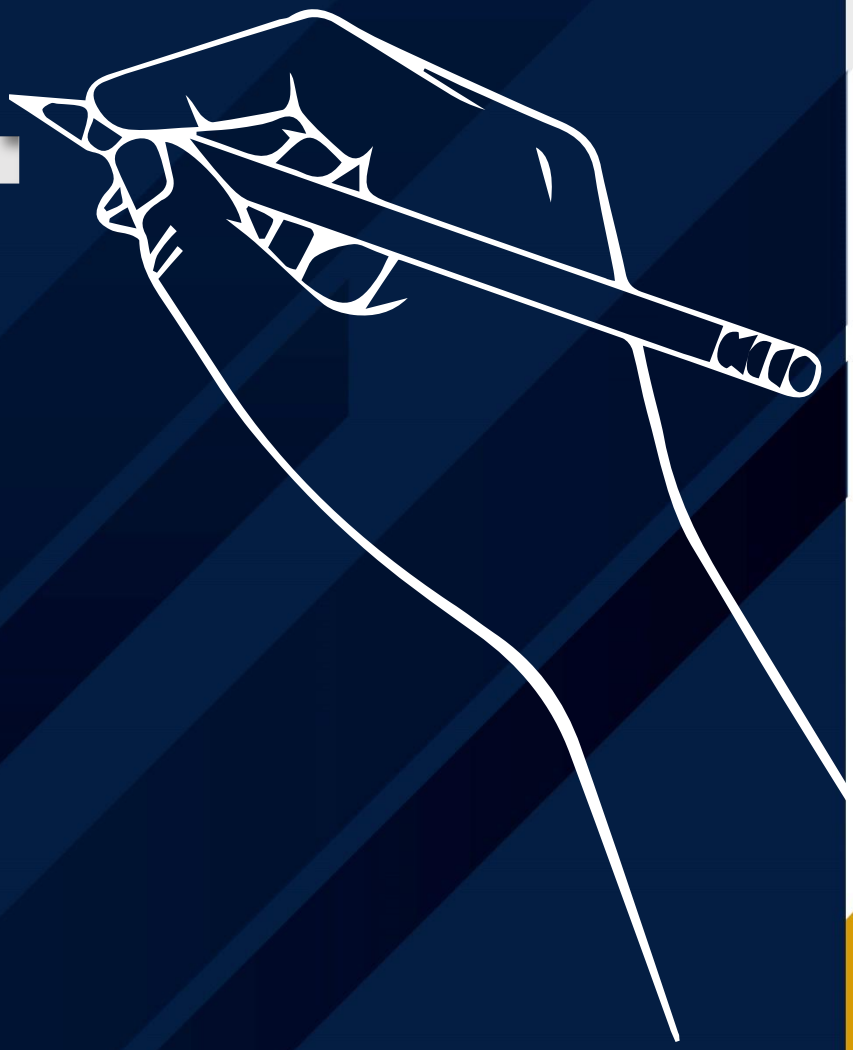
¿Cuáles son las características principales de HTML5, CSS y JavaScript en el desarrollo de aplicaciones móviles híbridas?

¿Qué es React Native y cómo se compara con otras opciones de desarrollo de aplicaciones móviles?

¿Cuál es el papel de los frameworks de desarrollo como Flutter en el desarrollo de aplicaciones móviles y cómo funcionan?

¿Cómo seleccionarías el lenguaje de programación más adecuado para un proyecto de desarrollo de aplicaciones móviles, teniendo en cuenta los requisitos del proyecto y las habilidades del equipo de desarrollo?

04



INTERFAZ DE PROGRAMACIÓN APLICACIONES

API´s

Una API (Interfaz de Programación de Aplicaciones, por sus siglas en inglés) es un conjunto de reglas y protocolos que permiten que diferentes aplicaciones o componentes de software se comuniquen entre sí. Las APIs actúan como intermediarios que permiten a los programas interactuar y compartir datos o funcionalidades de manera estructurada y controlada. Aquí tienes información clave sobre las APIs.

Propósito de las APIs.

Comunicación entre aplicaciones: Las APIs permiten que aplicaciones diferentes se comuniquen entre sí, lo que es fundamental en la construcción de sistemas complejos y en la integración de servicios.

Reutilización de código: Las APIs permiten a los desarrolladores reutilizar funcionalidades existentes sin tener que escribir todo el código desde cero.

Abstracción y modularidad: Las APIs ocultan los detalles de implementación subyacentes y permiten a los desarrolladores interactuar con un componente de software a través de una interfaz específica.

Acceso a servicios web: Muchas APIs se utilizan para acceder a servicios web y obtener datos o realizar operaciones a través de Internet.

Tipos de APIs:

APIs de biblioteca o clases: Permiten a los desarrolladores acceder a funcionalidades específicas de un lenguaje de programación o biblioteca, como las APIs de Java o las APIs de .NET.

APIs de sistema operativo: Proporcionan acceso a las funcionalidades del sistema operativo, como la API de Windows para interactuar con el sistema operativo Windows.

APIs de servicios web: Permiten la comunicación y el intercambio de datos entre aplicaciones a través de Internet utilizando protocolos como REST, SOAP o GraphQL.

Manejo de datos

El manejo de datos en aplicaciones móviles es un aspecto fundamental para garantizar el funcionamiento eficiente y la experiencia del usuario. Aquí tienes algunos conceptos clave y estrategias para el manejo de datos en aplicaciones móviles.

Almacenamiento de Datos Local:

Base de Datos SQLite: SQLite es una base de datos relacional incorporada que se utiliza comúnmente en aplicaciones móviles para almacenar datos estructurados de manera local en el dispositivo. Se utiliza a menudo en combinación con Android (usando Java o Kotlin) y en aplicaciones iOS (usando Swift o Objective-C).

NSUserDefaults (iOS) y SharedPreferences (Android): Estos son mecanismos de almacenamiento de datos clave-valor para almacenar configuraciones y preferencias de la aplicación de forma local en iOS y Android, respectivamente.

Almacenamiento en la Nube:

Servicios de Almacenamiento en la Nube: Para almacenar datos en la nube y permitir el acceso desde múltiples dispositivos, puedes utilizar servicios de almacenamiento en la nube como Amazon S3, Google Cloud Storage, Firebase Cloud Storage o Microsoft Azure Blob Storage.

Bases de Datos en la Nube: Utiliza bases de datos en la nube como Firebase Realtime Database o Firestore para almacenar datos en tiempo real que se sincronizan automáticamente entre dispositivos y clientes.

APIs de Red:

Consumo de Datos: Las aplicaciones móviles suelen interactuar con servidores y servicios web a través de llamadas a APIs RESTful o SOAP para obtener y enviar datos. Debes considerar la eficiencia y la seguridad en la transmisión de datos.

Autenticación: La autenticación es importante para proteger el acceso a datos y servicios en línea. Métodos comunes incluyen tokens de acceso, OAuth y autenticación basada en API keys.

Manejo de Caché:

Caching: Almacenar en caché datos en el dispositivo puede mejorar el rendimiento y reducir la carga del servidor. Puedes utilizar bibliotecas de caché como Glide (para imágenes) o implementar tu propia lógica de almacenamiento en caché.

Sincronización de Datos:

Sincronización: Si tu aplicación permite a los usuarios trabajar sin conexión, debes implementar lógica de sincronización para cargar y descargar datos cuando haya conexión a Internet disponible. Esto garantiza que los datos estén actualizados y disponibles tanto en línea como fuera de línea.

Seguridad de Datos:

Seguridad: Es fundamental proteger los datos sensibles del usuario y garantizar el cumplimiento de las regulaciones de privacidad de datos. Utiliza prácticas de seguridad como el cifrado de datos, autenticación sólida y el manejo seguro de tokens y credenciales.

Optimización del Uso de Datos Móviles:

Reducción de Consumo de Datos: Optimiza la aplicación para minimizar el uso de datos móviles, especialmente importante para usuarios con planes de datos limitados. Esto incluye la descarga selectiva de contenido y la compresión de datos.

Respaldo y Restauración de Datos:

Respaldo y Restauración: Considera la posibilidad de permitir a los usuarios realizar copias de seguridad de sus datos y restaurarlos en caso de pérdida del dispositivo o reinstalación de la aplicación.

Monitoreo y Análisis:

Analítica de Datos: Utiliza herramientas de analítica de aplicaciones móviles para comprender cómo los usuarios interactúan con tus datos y tu aplicación en general. Esto te ayudará a tomar decisiones informadas para mejorar la experiencia del usuario.

Cumplimiento Legal:

Privacidad y Cumplimiento Legal: Asegúrate de cumplir con las regulaciones de privacidad de datos, como el Reglamento General de Protección de Datos (RGPD) en Europa, y de obtener el consentimiento adecuado de los usuarios para recopilar y utilizar sus datos personales.

El manejo de datos en aplicaciones móviles es un aspecto crítico del desarrollo de aplicaciones exitosas. Debes considerar cuidadosamente las necesidades de almacenamiento, seguridad y rendimiento de datos al diseñar tu aplicación y seleccionar las tecnologías y estrategias apropiadas.

Las conexiones a la nube se refieren a la capacidad de acceder y utilizar recursos informáticos, como almacenamiento, aplicaciones y servicios, a través de internet en lugar de depender exclusivamente de recursos locales. Estas conexiones permiten a los usuarios y organizaciones acceder a datos y aplicaciones desde cualquier lugar y en cualquier momento, siempre que tengan una conexión a internet.

Almacenamiento en la nube: Permite a los usuarios almacenar datos en servidores remotos gestionados por proveedores de servicios en la nube, como Amazon Web Services (AWS), Microsoft Azure o Google Cloud Platform (GCP).

Computación en la nube: Proporciona acceso a recursos informáticos, como servidores virtuales, redes y almacenamiento, a través de internet. Esto permite a las organizaciones escalar recursos según sea necesario y pagar solo por lo que utilizan.

Software como servicio (SaaS): Ofrece aplicaciones alojadas en la nube que los usuarios pueden acceder a través de internet, en lugar de instalar y mantener el software en sus propios dispositivos. Ejemplos de SaaS incluyen servicios como Google Workspace, Microsoft Office 365 y Salesforce.

Plataforma como servicio (PaaS): Proporciona un entorno de desarrollo y despliegue en la nube que permite a los desarrolladores crear, probar y desplegar aplicaciones sin tener que preocuparse por la infraestructura subyacente. Ejemplos de PaaS incluyen Google App Engine, Microsoft Azure App Service y Heroku.

Infraestructura como servicio (IaaS): Ofrece acceso a recursos informáticos virtualizados, como servidores, redes y almacenamiento, a través de internet. Los usuarios pueden controlar y gestionar estos recursos según sea necesario. Ejemplos de IaaS incluyen Amazon Elastic Compute Cloud (EC2), Microsoft Azure Virtual Machines y Google Compute Engine.

Las conexiones a la nube han transformado la forma en que las organizaciones y los individuos gestionan sus recursos informáticos, permitiendo una mayor flexibilidad, escalabilidad y eficiencia. Sin embargo, también plantean desafíos en términos de seguridad, privacidad y dependencia de proveedores externos.

Evaluación Unidad IV.

¿Qué es una API y cuál es su función en el desarrollo de aplicaciones?

Explica la diferencia entre una API pública y una API privada.

¿Por qué son importantes las API en el desarrollo de aplicaciones móviles?

Menciona al menos tres ejemplos de API comunes utilizadas en el desarrollo de aplicaciones móviles y describe para qué se utilizan.

¿Qué es RESTful API y cuáles son sus principales características?

¿Cuál es la diferencia entre SOAP y REST en términos de arquitectura de API?

Explica qué es un token de autenticación y cómo se utiliza en el contexto de las API de seguridad.

¿Cuál es el papel de la documentación de la API en el desarrollo de aplicaciones móviles y qué información debe incluir?

¿Cómo pueden los desarrolladores manejar los errores que surgen al interactuar con una API en una aplicación móvil?

¿Cuáles son algunas consideraciones importantes de seguridad al utilizar API en el desarrollo de aplicaciones móviles?

05

ENTORNO DE DESARROLLO EN APLICACIONES

IDE DE DESARROLLO

Un IDE (Entorno de Desarrollo Integrado, por sus siglas en inglés) es un software que proporciona herramientas integradas para facilitar el desarrollo de software. Un IDE típico incluye un editor de código, herramientas de compilación y depuración, y a menudo integra características adicionales como control de versiones, gestión de proyectos y soporte para diferentes lenguajes de programación.

Visual Studio: Desarrollado por Microsoft, es uno de los IDE más completos y populares. Ofrece soporte para una amplia gama de lenguajes de programación, incluyendo C#, C++, Visual Basic, F#, Python, y más. Además, proporciona características avanzadas como depuración integrada, herramientas de diseño de interfaz de usuario y soporte para la creación de aplicaciones web y móviles.

Eclipse: Es un IDE de código abierto ampliamente utilizado, especialmente en el desarrollo de aplicaciones Java. Eclipse es altamente configurable y extensible, lo que lo hace adecuado para una variedad de proyectos de desarrollo de software.

IntelliJ IDEA: Desarrollado por JetBrains, es conocido por su excelente soporte para Java, Kotlin y otros lenguajes. IntelliJ IDEA ofrece características avanzadas como refactorización inteligente de código, análisis estático, pruebas integradas y soporte para frameworks populares como Spring y Android.

PyCharm: Otro producto de JetBrains, es un IDE específicamente diseñado para el desarrollo de aplicaciones Python. Ofrece funciones avanzadas para el desarrollo de Python, incluyendo completado de código, depuración, análisis de código y soporte para frameworks como Django y Flask.

Visual Studio Code: Aunque técnicamente es un editor de código, Visual Studio Code ofrece muchas características de un IDE, gracias a su amplia variedad de extensiones. Es altamente personalizable y soporta una amplia gama de lenguajes de programación, convirtiéndolo en una opción popular para muchos desarrolladores.

Xamarin Studio

Xamarin Studio fue un entorno de desarrollo integrado (IDE) desarrollado por Xamarin, una compañía adquirida por Microsoft en 2016. Estaba diseñado específicamente para el desarrollo de aplicaciones móviles utilizando la plataforma Xamarin, que permite a los desarrolladores escribir aplicaciones nativas para iOS, Android y Windows utilizando el lenguaje de programación C# y el marco de trabajo .NET.

Edición de código: Un editor de código completo con resaltado de sintaxis, autocompletado y navegación inteligente.

Depuración: Herramientas de depuración integradas que permiten a los desarrolladores identificar y solucionar problemas en sus aplicaciones.

Diseño de interfaz de usuario: Herramientas para diseñar interfaces de usuario para aplicaciones móviles utilizando XAML y arrastrar y soltar componentes.

Integración con Xamarin Test Cloud: Capacidades integradas para ejecutar pruebas automatizadas en dispositivos reales y emuladores.

Gestión de proyectos: Funcionalidades para gestionar proyectos, referencias de bibliotecas, y paquetes de NuGet.

Visual Studio

Visual Studio es un entorno de desarrollo integrado (IDE) ampliamente utilizado desarrollado por Microsoft. Es una herramienta potente y completa que proporciona a los desarrolladores todas las herramientas necesarias para crear una amplia variedad de aplicaciones, desde aplicaciones de escritorio hasta aplicaciones web y móviles.

Editor de código avanzado**: Visual Studio proporciona un editor de código** robusto con resaltado de sintaxis, autocompletado inteligente, refactorización de código, navegación rápida y otras características que mejoran la productividad del desarrollador.

Depuración integrada: **Ofrece herramientas de depuración poderosas que** permiten a los desarrolladores detectar y solucionar problemas en sus aplicaciones de manera eficiente. Esto incluye la capacidad de establecer puntos de interrupción, inspeccionar variables, seguir el flujo de ejecución y más.

Diseño de interfaz de usuario**: Para aplicaciones de escritorio y aplicaciones móviles, Visual Studio proporciona herramientas para diseñar interfaces de** usuario de forma visual, lo que facilita la creación de interfaces atractivas y funcionales.

Soporte para múltiples lenguajes de programación: Aunque Visual Studio es conocido principalmente por su soporte para C# y .NET, también ofrece soporte para una amplia variedad de otros lenguajes de programación, incluyendo C++, JavaScript, TypeScript, Python, y más.

Integración con servicios en la nube: Visual Studio se integra con servicios en la nube como Azure, lo que permite a los desarrolladores construir, implementar y administrar aplicaciones en la nube de manera sencilla y eficiente.

Visual Studio for Mac.

Visual Studio for Mac ofrece muchas de las características que los desarrolladores esperan de Visual Studio, aunque con un enfoque más específico en el desarrollo multiplataforma y en particular en el desarrollo de aplicaciones móviles con tecnologías como Xamarin.

Desarrollo multiplataforma: Permite a los desarrolladores crear aplicaciones para múltiples plataformas, incluyendo iOS, Android y macOS, utilizando tecnologías como Xamarin y .NET.

Edición de código: Ofrece un editor de código completo con resaltado de sintaxis, autocompletado inteligente, refactorización de código y otras características para mejorar la productividad del desarrollador.

Depuración integrada: Proporciona herramientas de depuración poderosas que permiten a los desarrolladores detectar y solucionar problemas en sus aplicaciones de manera eficiente.

Diseño de interfaz de usuario: Incluye herramientas para diseñar interfaces de usuario de forma visual, lo que facilita la creación de interfaces atractivas y funcionales para aplicaciones móviles y de escritorio.

Integración con Azure: Permite a los desarrolladores integrar fácilmente servicios en la nube de Microsoft Azure en sus aplicaciones para una implementación y administración sencillas.

Soporte para múltiples lenguajes: Aunque Visual Studio for Mac está más centrado en el desarrollo con tecnologías .NET, también ofrece soporte para otros lenguajes de programación como JavaScript, TypeScript, Python y más.

Evaluación Unidad V.

¿Qué es una API y cuál es su función en el desarrollo de aplicaciones?

Explica la diferencia entre una API pública y una API privada.

¿Por qué son importantes las API en el desarrollo de aplicaciones móviles?

Menciona al menos tres ejemplos de API comunes utilizadas en el desarrollo de aplicaciones móviles y describe para qué se utilizan.

¿Qué es RESTful API y cuáles son sus principales características?

¿Cuál es la diferencia entre SOAP y REST en términos de arquitectura de API?

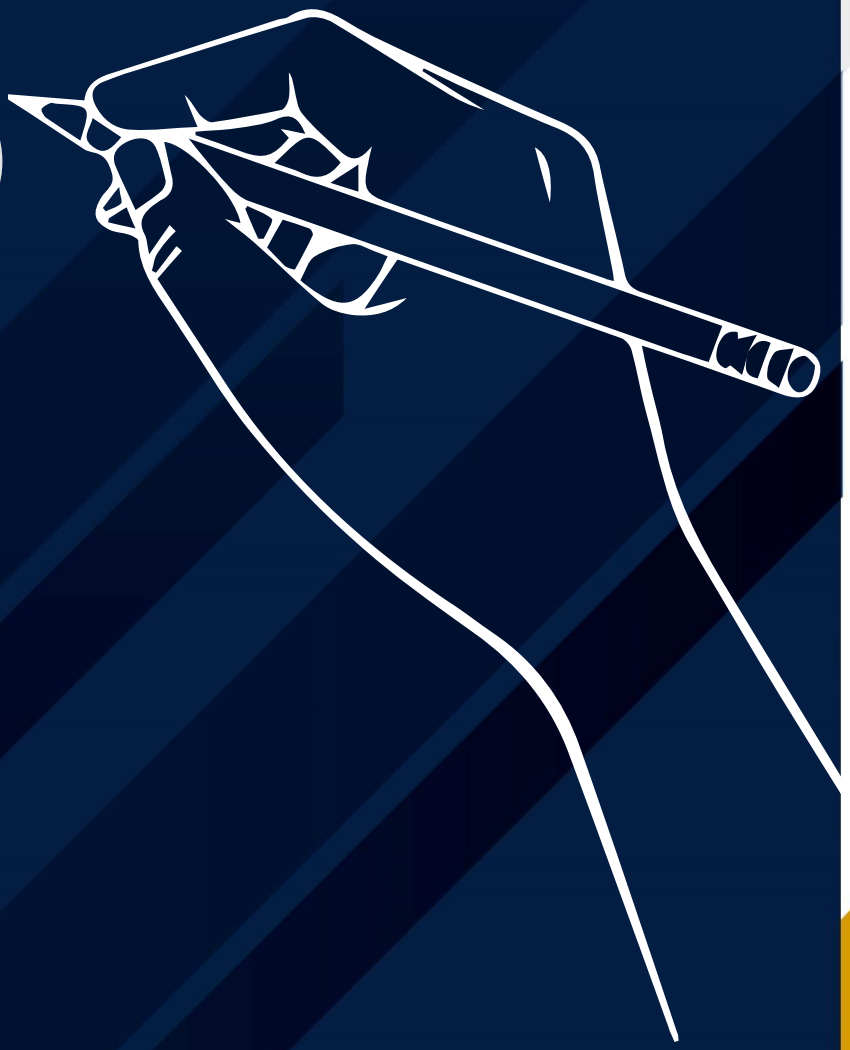
Explica qué es un token de autenticación y cómo se utiliza en el contexto de las API de seguridad.

¿Cuál es el papel de la documentación de la API en el desarrollo de aplicaciones móviles y qué información debe incluir?

¿Cómo pueden los desarrolladores manejar los errores que surgen al interactuar con una API en una aplicación móvil?

¿Cuáles son algunas consideraciones importantes de seguridad al utilizar API en el desarrollo de aplicaciones móviles?

06



LENGUAJE DE DESARROLLO XAMARIN

DESARROLLO DE APLICACIONES.

El desarrollo de aplicaciones se refiere al proceso de creación de software diseñado para funcionar en dispositivos electrónicos, como computadoras, teléfonos inteligentes, tabletas y otros dispositivos móviles. Estas aplicaciones pueden tener una amplia variedad de propósitos, desde proporcionar entretenimiento y utilidades hasta facilitar la productividad y mejorar la eficiencia en diversos ámbitos.

Definición de requisitos: Comprender las necesidades y requisitos del usuario final es fundamental para el éxito de cualquier aplicación. Esto implica identificar el propósito de la aplicación, el público objetivo, las características principales y cualquier restricción técnica o de negocio.

Diseño de la aplicación: En esta etapa, se crea un diseño visual y funcional de la aplicación. Esto puede incluir la creación de bocetos, wireframes y prototipos interactivos que ayuden a visualizar la interfaz de usuario y la experiencia del usuario.

Desarrollo de software: Los desarrolladores trabajan en la implementación del software según los requisitos y el diseño establecidos. Esto implica escribir código, integrar funcionalidades, realizar pruebas unitarias y asegurarse de que la aplicación cumpla con los estándares de calidad.

Pruebas y depuración: Una vez que la aplicación está desarrollada, se somete a pruebas exhaustivas para identificar y corregir errores, así como para garantizar que funcione correctamente en una variedad de dispositivos y escenarios de uso.

Despliegue: Una vez que la aplicación ha pasado por las pruebas necesarias y se considera lista para su lanzamiento, se procede a su despliegue en el entorno de producción. Esto puede implicar la publicación en tiendas de aplicaciones (como App Store y Google Play), la implementación en servidores web o la distribución interna en entornos corporativos.

Mantenimiento y actualización: El desarrollo de aplicaciones es un proceso continuo. Después del lanzamiento, es importante monitorear y mantener la aplicación para corregir errores, responder a comentarios de usuarios y realizar actualizaciones periódicas para agregar nuevas características y mejorar la experiencia del usuario.

El desarrollo de aplicaciones puede involucrar diferentes tecnologías, lenguajes de programación y plataformas, dependiendo de los requisitos específicos de la aplicación y del público objetivo. Algunas de las plataformas y tecnologías populares para el desarrollo de aplicaciones incluyen:

- Desarrollo nativo para iOS (usando Swift o Objective-C) y Android (usando Java o Kotlin).
- Desarrollo multiplataforma utilizando frameworks como Xamarin, React Native o Flutter.
- Desarrollo web progresivo (PWA) utilizando tecnologías web estándar como HTML, CSS y JavaScript.

Xamarin Android.

Xamarin.Android es una tecnología que forma parte del conjunto de herramientas de desarrollo de Xamarin, que permite a los desarrolladores crear aplicaciones nativas para Android utilizando el lenguaje de programación C# y el marco de trabajo .NET. Con Xamarin.Android, los desarrolladores pueden escribir código una vez y compilarlo para ejecutarse nativamente en dispositivos Android.

Desarrollo en C#: En lugar de utilizar Java, el lenguaje de programación nativo de Android, los desarrolladores pueden escribir código en C#, que es un lenguaje de programación más familiar para muchos desarrolladores y forma parte del ecosistema .NET de Microsoft.

Acceso completo a las APIs de Android: Xamarin.Android proporciona acceso completo a las APIs de Android, lo que permite a los desarrolladores utilizar todas las características y funcionalidades de la plataforma Android, como sensores, servicios de ubicación, notificaciones, base de datos SQLite, y mucho más.

Integración con Visual Studio: Xamarin.Android se integra perfectamente con Visual Studio, el popular entorno de desarrollo integrado (IDE) de Microsoft. Esto permite a los desarrolladores aprovechar las herramientas de desarrollo avanzadas de Visual Studio, como el editor de código, la depuración integrada y la gestión de proyectos.

Reutilización de código: Una de las ventajas clave de Xamarin.Android es la capacidad de reutilizar código entre aplicaciones de Android y otras plataformas compatibles con Xamarin, como iOS y Windows. Esto permite a los desarrolladores compartir lógica de negocio y componentes de la interfaz de usuario entre diferentes versiones de la misma aplicación para diferentes plataformas.

Soporte para bibliotecas de terceros: Xamarin.Android es compatible con la mayoría de las bibliotecas de terceros y componentes de código abierto de Android, lo que permite a los desarrolladores aprovechar la amplia gama de herramientas y recursos disponibles en el ecosistema de desarrollo de Android. Desean aprovechar la capacidad de compartir código entre múltiples plataformas.

Xamarin iOS.

Xamarin.iOS es una tecnología desarrollada por Xamarin, una subsidiaria de Microsoft, que permite a los desarrolladores crear aplicaciones nativas para dispositivos iOS utilizando el lenguaje de programación C# y el marco de trabajo .NET. Con Xamarin.iOS, los desarrolladores pueden escribir código una vez y luego compilarlo para que se ejecute nativamente en dispositivos iOS, incluyendo iPhone, iPad y iPod Touch.

Desarrollo en C#: En lugar de utilizar Objective-C o Swift, los lenguajes de programación nativos de iOS, Xamarin.iOS permite a los desarrolladores escribir todo el código en C#, que es un lenguaje de programación moderno y ampliamente utilizado.

Acceso completo a las API de iOS: Xamarin.iOS proporciona acceso completo a todas las APIs nativas de iOS, lo que significa que los desarrolladores pueden aprovechar todas las características y funcionalidades del sistema operativo iOS, incluyendo la cámara, sensores, mapas, notificaciones push, y mucho más.

Integración con Visual Studio: Xamarin.iOS se integra perfectamente con Visual Studio, el popular entorno de desarrollo integrado (IDE) de Microsoft, lo que facilita a los desarrolladores el desarrollo, depuración y pruebas de aplicaciones iOS en un entorno familiar.

Reutilización de código: Una de las ventajas clave de Xamarin.iOS es la capacidad de compartir código entre diferentes plataformas. Los desarrolladores pueden compartir gran parte del código de negocio y la lógica de la aplicación entre las versiones de iOS, Android y Windows, lo que reduce significativamente el esfuerzo de desarrollo.

El desarrollo de aplicaciones se refiere al proceso de creación de software diseñado para funcionar en dispositivos electrónicos, como computadoras, teléfonos inteligentes, tabletas y otros dispositivos móviles. Estas aplicaciones pueden tener una amplia variedad de propósitos, desde proporcionar entretenimiento y utilidades hasta facilitar la productividad y mejorar la eficiencia en diversos ámbitos.

Definición de requisitos: Comprender las necesidades y requisitos del usuario final es fundamental para el éxito de cualquier aplicación. Esto implica identificar el propósito de la aplicación, el público objetivo, las características principales y cualquier restricción técnica o de negocio.

Diseño de la aplicación: En esta etapa, se crea un diseño visual y funcional de la aplicación. Esto puede incluir la creación de bocetos, wireframes y prototipos interactivos que ayuden a visualizar la interfaz de usuario y la experiencia del usuario.

Desarrollo de software: Los desarrolladores trabajan en la implementación del software según los requisitos y el diseño establecidos. Esto implica escribir código, integrar funcionalidades, realizar pruebas unitarias y asegurarse de que la aplicación cumpla con los estándares de calidad.

Pruebas y depuración: Una vez que la aplicación está desarrollada, se somete a pruebas exhaustivas para identificar y corregir errores, así como para garantizar que funcione correctamente en una variedad de dispositivos y escenarios de uso.

Despliegue: Una vez que la aplicación ha pasado por las pruebas necesarias y se considera lista para su lanzamiento, se procede a su despliegue en el entorno de producción. Esto puede implicar la publicación en tiendas de aplicaciones (como App Store y Google Play), la implementación en servidores web o la distribución interna en entornos corporativos.

Mantenimiento y actualización: El desarrollo de aplicaciones es un proceso continuo. Después del lanzamiento, es importante monitorear y mantener la aplicación para corregir errores, responder a comentarios de usuarios y realizar actualizaciones periódicas para agregar nuevas características y mejorar la experiencia del usuario.

El desarrollo de aplicaciones puede involucrar diferentes tecnologías, lenguajes de programación y plataformas, dependiendo de los requisitos específicos de la aplicación y del público objetivo. Algunas de las plataformas y tecnologías populares para el desarrollo de aplicaciones incluyen:

- Desarrollo nativo para iOS (usando Swift o Objective-C) y Android (usando Java o Kotlin).
- Desarrollo multiplataforma utilizando frameworks como Xamarin, React Native o Flutter.
- Desarrollo web progresivo (PWA) utilizando tecnologías web estándar como HTML, CSS y JavaScript.

Xamarin Android.

Xamarin.Android es una tecnología que forma parte del conjunto de herramientas de desarrollo de Xamarin, que permite a los desarrolladores crear aplicaciones nativas para Android utilizando el lenguaje de programación C# y el marco de trabajo .NET. Con Xamarin.Android, los desarrolladores pueden escribir código una vez y compilarlo para ejecutarse nativamente en dispositivos Android.

Xamarin UWP.

Xamarin no se limita únicamente al desarrollo de aplicaciones móviles para iOS y Android, sino que también ofrece soporte para el desarrollo de aplicaciones de Plataforma Universal de Windows (UWP) mediante Xamarin.Forms.

Xamarin.Forms es un marco de trabajo que permite a los desarrolladores crear interfaces de usuario compartidas para aplicaciones móviles (iOS y Android) y aplicaciones de escritorio (UWP) utilizando XAML y C#. Con Xamarin.Forms, los desarrolladores pueden escribir una vez el código para la interfaz de usuario y luego compilarlo para múltiples plataformas, lo que les permite compartir una gran cantidad de código entre las diferentes versiones de la aplicación.

Desarrollo en C#: Los desarrolladores pueden utilizar el lenguaje de programación C# para desarrollar la lógica de la aplicación y crear la interfaz de usuario utilizando XAML, un lenguaje de marcado declarativo similar a XML.

Interfaces de usuario compartidas: Con Xamarin.Forms, los desarrolladores pueden crear interfaces de usuario compartidas que se ejecuten en dispositivos móviles (iOS y Android) y en dispositivos de escritorio (UWP), lo que permite una experiencia de desarrollo más eficiente y la reutilización del código.

Acceso a las APIs de UWP: Xamarin.Forms proporciona acceso completo a las APIs de la Plataforma Universal de Windows (UWP), lo que permite a los desarrolladores aprovechar todas las características y funcionalidades de Windows 10, como notificaciones push, Live Tiles, integración con Cortana, y mucho más.

Integración con Visual Studio: Xamarin.Forms se integra perfectamente con Visual Studio, el popular entorno de desarrollo integrado (IDE) de Microsoft, lo que facilita a los desarrolladores el desarrollo, la depuración y las pruebas de aplicaciones UWP en un entorno familiar.

Soporte para bibliotecas de terceros: Xamarin.Forms es compatible con la mayoría de las bibliotecas de terceros disponibles para UWP, lo que permite a los desarrolladores aprovechar la rica comunidad de desarrolladores y las herramientas disponibles.

En resumen, Xamarin.Forms ofrece una solución poderosa para el desarrollo de aplicaciones de Plataforma Universal de Windows (UWP), permitiendo a los desarrolladores crear interfaces de usuario compartidas y compartir una gran cantidad de código entre las diferentes versiones de la aplicación para dispositivos móviles y de escritorio.

Xamarin Forms.

Xamarin.Forms es un marco de trabajo (framework) para el desarrollo de aplicaciones móviles multiplataforma que permite a los desarrolladores crear interfaces de usuario (UI) compartidas para aplicaciones en iOS, Android y Plataforma Universal de Windows (UWP), todo ello utilizando un único código base. Xamarin.Forms simplifica el desarrollo al proporcionar una API única y un conjunto de controles de interfaz de usuario que pueden ser renderizados de forma nativa en cada plataforma objetivo.

Desarrollo multiplataforma: Con Xamarin.Forms, los desarrolladores pueden escribir código una vez y ejecutarlo en múltiples plataformas, lo que reduce significativamente el tiempo y el esfuerzo requerido para desarrollar y mantener aplicaciones para iOS, Android y UWP.

Interfaces de usuario compartidas: Xamarin.Forms utiliza un enfoque de interfaz de usuario compartida, lo que significa que la mayor parte del código relacionado con la interfaz de usuario se puede compartir entre las diferentes plataformas. Esto incluye la creación de páginas, layouts, controles, y elementos de interfaz de usuario.

Lenguaje de marcado XAML: Para definir la interfaz de usuario, Xamarin.Forms utiliza XAML (Extensible Application Markup Language), un lenguaje de marcado declarativo similar a XML. Esto permite a los desarrolladores definir la estructura y el diseño de la interfaz de usuario de forma rápida y sencilla.

Controles nativos: Aunque Xamarin.Forms proporciona una capa de abstracción sobre los controles de la interfaz de usuario, los controles se renderizan de forma nativa en cada plataforma objetivo. Esto significa que las aplicaciones desarrolladas con Xamarin.Forms tienen un aspecto y un comportamiento nativo en cada plataforma.

Acceso a APIs nativas: Xamarin.Forms proporciona acceso completo a las APIs nativas de cada plataforma, lo que permite a los desarrolladores aprovechar todas las características y funcionalidades específicas de cada sistema operativo, como notificaciones push, acceso a la cámara, geolocalización, y mucho más.

Integración con Visual Studio: Xamarin.Forms se integra perfectamente con Visual Studio, el popular entorno de desarrollo integrado (IDE) de Microsoft, lo que facilita a los desarrolladores el desarrollo, la depuración y las pruebas de aplicaciones multiplataforma en un entorno familiar.

Integración del Lenguaje: Visual Studio es conocido por su amplia integración con múltiples lenguajes de programación, incluyendo C#, Visual Basic, C++, F#, Python, y más. Esto significa que ofrece características específicas para cada lenguaje, como resaltado de sintaxis, completado automático, depuración, y otras herramientas de productividad.

Integración de Control de Versiones: Visual Studio admite la integración con sistemas de control de versiones como Git, SVN, y Team Foundation Server (TFS). Esto permite a los desarrolladores gestionar sus repositorios directamente desde el entorno de desarrollo.

Integración con Azure y Servicios en la Nube: Microsoft ofrece una amplia gama de servicios en la nube a través de Azure. Visual Studio proporciona herramientas integradas para trabajar con estos servicios, como la implementación directa en Azure, la gestión de recursos en la nube, la monitorización de aplicaciones, entre otras.

Integración de Depuración: Visual Studio ofrece un potente depurador que permite a los desarrolladores depurar sus aplicaciones paso a paso, inspeccionar variables, establecer puntos de interrupción, y mucho más. Esta integración es fundamental para identificar y corregir errores en el código.

Integración de Pruebas Unitarias: Visual Studio facilita la escritura y ejecución de pruebas unitarias a través de su integración con frameworks como MSTest, NUnit, xUnit, etc. Esto ayuda a garantizar la calidad del código mediante la automatización de pruebas.

Evaluación Unidad . VI

¿Qué es Xamarin y cuál es su propósito en el desarrollo de aplicaciones móviles?

¿Cuál es la principal ventaja de utilizar Xamarin para el desarrollo de aplicaciones móviles en comparación con otras opciones?

Describe la arquitectura de Xamarin y cómo se relaciona con el desarrollo de aplicaciones móviles.

¿Cuáles son los componentes principales de Xamarin y cómo se utilizan en el desarrollo de aplicaciones?

Explica la diferencia entre Xamarin.Forms y Xamarin.Native.

¿Cómo se compila y se ejecuta una aplicación desarrollada con Xamarin en dispositivos iOS y Android?

¿Cuál es el lenguaje de programación principal utilizado en Xamarin y por qué se eligió?

¿Qué es Xamarin.Android y cuál es su relación con el desarrollo de aplicaciones para dispositivos Android?

¿Qué es Xamarin.iOS y cuál es su relación con el desarrollo de aplicaciones para dispositivos iOS?

¿Cuáles son algunas de las limitaciones o desafíos comunes al desarrollar aplicaciones con Xamarin y cómo se pueden abordar?

Bibliografía

Augment (2022). Augment, 3D Realidad Aumentada (5.0.1) [Aplicación móvil]. Google Play. https://play.google.com/store/apps/details?id=com.ar.augment&hl=es_EC&gl=US

MH Riley Ltd. (2020). Spending Tracker (2.3.1) [Aplicación móvil]. Google Play. https://play.google.com/store/apps/details?id=com.mhriley.spendingtracker&hl=en_US

Trello, Inc. (2023). Trello: Organize anything! (Versión 2023.15) [Aplicación móvil]. App Store. https://apps.apple.com/us/app/trello-organize-anything/id461504587

Sood, Raghav. Pro Android Augmented Reality. s.l. : Apress Berkely, 2012.

Brandski, Gary y Kaehler, Adrian. OpenCV. s.l. : O'Reilly Media, 2008.

H.F.Korth. Fundamentos de bases de datos. s.l. : McGraw-Hill, 1995.

Definición, usos y ventajas del lenguaje HTML5[Internet]. Mar 2019[citado [12 Ene 2020]. Disponible en: Disponible en: https://blog.aulaformativa.com/definicion-usos-ventajas-lenguaje-html5/

Código HTML vs HTML5: Diferentes lenguajes de desarrollo web[internet]. [Actualizado 6 Dic 2019; citado 17 Ene 2020]. Disponible en: Disponible en: https://es.bitdegree.org/tutoriales/codigo-html/

Gavilondo Mariño X, Vialart Vidal MN. Salud Móvil: retos y perspectivas de aplicación en Cuba. Rev Cubana Enfermer [Internet]. Mar 2016 [citado 13 Ago 2020]; 32(1): 98-106. Disponible en: Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0864-03192016000100012&lng=es

Gauchat JD. El gran libro de HTML5, CSS3 y Javascript [Internet]. Marcombo; 2012 [citado 12 Ene 2020]. Disponible en: Disponible en: https://gutl.jovenclub.cu/wp-content/uploads/2013/10/El+gran+libro+de+HTML5+CSS3+y+Javascrip.pdf

Vidal Ledo MJ, Gavilondo Mariño X, Rodríguez Díaz A, Cuéllar Rojas A. Aprendizaje móvil. Educ Med Super [Internet]. Sep 2015 [citado 13 Ago 2020];29(3):[aprox. 8p.]. Disponible en: Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0864-21412015000300024&lng=es

Hidalgo Hernández R. ¿El uso del celular, un problema para el profesor en el aula o un medio de comunicación convertido en medio de enseñanza?. Educ Méd Sup [Internet]. 2015 [citado 13 Ago 2020;29(4): [aprox. 9 p.]. Disponible en: Disponible en: http://ems.sld.cu/index.php/ems/article/view/768/293

Vidal Ledo M, Gavilondo Mariño X. TOPIC: Teaching and mobile technologies. Educ Med Super [Internet]. Jun 2018 [citado 13 Ago 2020]; 32(2):[aprox. 8 p.]. Disponible en: Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0864-21412018000200027&lng=es

Visual Studio Code. Personalización [Internet]. Oct 2019 [citado 12 Ene 2020]. Disponible en: Disponible en: https://www.mclibre.org/consultar/informatica/lecciones/vsc-personalizacion.html

9. Manual Básico Android Studio [Internet]. 2019 [citado 25 Feb 2020]. Disponible en: Disponible en: https://tutorialesenpdf.com/android-studio/previsualizacion/Curso%20Android%20Studio.pdf

Stephie R. Tratamiento digital de la imagen [Internet].© Academia; 2019 [citado 23 Feb 2020]. Disponible en: Disponible en: https://www.academia.edu/4225105/tratamiento_de_imagenes

Cruz-Barragán A, Barragán-López AD. Aplicaciones móviles para el proceso de enseñanza-aprendizaje en Enfermería. Rev Salud Administ [Internet]. 2014 [citado 21 Jun 2020];1(3): 51-57. Disponible en: Disponible en: https://revista.unsis.edu.mx/index.php/saludyadmon/article/download/81/78 [Links]

INSTITUTO SUPERIOR TECNOLÓGICO PELILEO

ISBN: 978-9942-686-56-5

Educación gratuita y de calidad