**DANIEL HORACIO CARRASCOSA PELUC E010-353**

**Práctico N° 1: Fundamentos de la Web y la World Wide Web**

1-Responde las siguientes preguntas:

1. ¿Qué es el Protocolo HTTP?

Es el protocolo de comunicación que se utiliza para transferir datos en la web. Define cómo se envían y reciben las solicitudes y respuestas entre los clientes (navegadores web) y los servidores web.

1. ¿Cuál es la diferencia entre URI y URL?

Todos los URLs son URIs, pero no todos los URIs son URLs. Mientras que un URL siempre proporciona un metodo para acceder a un recurso, un URI en general solo garantiza la identificaion del recurso.

1. ¿Cuál es el propósito del lenguaje HTML?

Su proposito es definir la estructura y el contenido de una página mediante etiquetas y elementos que describen el texto, imágenes, enlaces y otros elementos presentes en la página.

1. ¿Qué función cumple el CSS en una página web?

Su funcion es permitir controlar la apariencia visual de los elementos HTML, como el color, el tamaño, la tipografía y la disposición en la página.

1. ¿Cuál es el rol del JavaScript en el desarrollo web?

Su rol permite manipular el contenido de la página en tiempo real, responder a eventos del usuario y comunicarse con servidores web para cargar datos adicionales sin recargar la página completa

1. ¿Qué es el D.O.M.? ¿Y para qué se utiliza?

Es una representación en memoria de la estructura de una página web creada con HTML. Permite acceder y manipular los elementos de la página mediante programación, lo que facilita la interacción dinámica con el contenido de la página.

2- Investiga y completa los siguientes eventos históricos de la Web:

**1991**: Se presenta la World Wide Web al público. Tim Berners-Lee y su equipo en CERN desarrollaron las tecnologías fundamentales como HTML, URI y HTTP, que forman la base de la web.

**1993**: Marc Andreessen y Eric Bina desarrollan el navegador web Mosaic en la Universidad de Illinois, que populariza el uso de la web

**1995**: Surge la comercialización de la web con el lanzamiento de sitios como Amazon y eBay.

1996: La "Guerra de Navegadores" entre Microsoft e Internet Explorer contra Netscape comienza, marcando una competencia intensa en el desarrollo de navegadores web

**2004**: Mark Zuckerberg lanza Facebook, iniciando la era de las redes sociales tal como la conocemos hoy. Además, Mozilla lanza Firefox, un navegador que desafía el dominio de Internet Explorer​

**2011**: Twitter y Facebook juegan un papel crucial en las revueltas del Medio Oriente, conocidas como la Primavera Árabe, demostrando el poder de las redes sociales para influir en eventos políticos y sociales globales

**2012**: El presidente Barack Obama se opone a leyes como SOPA y PIPA, que habrían impuesto estrictas reglas sobre el contenido en internet. Esta acción es vista como una victoria para la libertad en internet y para sitios basados en contenido generado por usuarios

**2014**: Se celebra el 25 aniversario de la World Wide Web. La web ha crecido exponencialmente y continúa siendo una plataforma fundamental para la comunicación global y el intercambio de información

**2019**: La tecnología 5G comienza a desplegarse a nivel mundial, prometiendo velocidades de internet significativamente más rápidas y nuevas posibilidades para la conectividad

**2022**: Continúan los debates sobre la privacidad y la seguridad en línea, con un enfoque en la regulación del contenido y la protección de datos personales en un entorno cada vez más digital

3- Ingresa a la web <https://http.app/> y analiza la url <https://www.unsj.edu.ar/>

¿Qué datos obtenemos en?

* Server (¿Qué significa?)

Apache/2.4.61 (Debian) . Los servidores almacenan y sirven páginas web a través de protocolos como HTTP o HTTPS. Cuando un usuario solicita una página web mediante su navegador, el servidor web responde enviando los archivos necesarios (HTML, CSS, imágenes, etc.) para mostrar la página.

* Protocol (¿Qué significa?)

HTTP 1.1 - Es el protocolo de comunicación utilizado para la transferencia de datos en la World Wide Web.

* Request method (¿Porqué GET?)

El método GET solicita la recuperación de un recurso específico del servidor. Es utilizado para obtener información del servidor, como páginas web, imágenes o archivos.

* ¿Es http 1.0 o versión 2.0?. ¿Cuál es la diferencia?

HTTP 2.0 es una versión más avanzada y eficiente que HTTP 1.0, con mejoras significativas en la gestión de conexiones, la eficiencia de la transmisión de datos, y la capacidad de manejar múltiples peticiones en paralelo

* Puedes dar un ejemplo de una URL que trabaje con protocolo http 2.0.

<https://www.google.com/>

<https://www.youtube.com/>

<https://www.facebook.com/>

<https://www.amazon.com/>

<https://www.wikipedia.org/>

4- Completa la siguiente tabla con las ideas principales.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CARACTERÍSTICA | WEB1.0 | WEB2.0 | WEB3.0 | WEB4.0 | WEB5.0 |
| Año de Inicio | 1990s | 1980s-2000s | 2000s-2010s | 2010s-2020s | 2020s-Presente |
| Principales Tecnologías | HTML estático, CGI, HTTP/1.0 | AJAX, redes sociales | Web semantica, AI, datos estructurados | Deep Learning, Machine Learning | IoT, AI |
| Interactividad | Baja:  Publicacion de contenido estatico | Media:  Lectura y escritura bidireccional | Alta: interaccion mejorada por AI | Muy Alta: intreractividad predictvia | Muy Alta: interaccion inmersiva |
| Contenido | Estatico, solo texto | Dinamico, colaborativo | Semántico | Personalizado, basado en el contexto | Semantico y contextualmente integrado |
| Ejemplos de Sitios | Primeros sitios web (CERN, etc) | Blogs, foros, redes sociales | Wolfram Alpha, motores semanticos | Google, Youtube, Facebook | Alexa, Siri, dispositivos conectados |
| Enfoque Principal | Publicación de contenido | Colaboracion y comunidad | Significado y semantica | Experiencias personalizadas y predictivas | Convergencia con tecnologia emergente |

5- Analiza la siguiente URL, desglosar sus partes:

https://www.google.com.ar/search?q=url+con+%26+y+%23&sclient=gws-wiz#ip=1

* Protocolo: https
* Dominio: www.google.com.ar
* Ruta: /search
* Parámetros: q=url+con+%26+y+%23&sclient=gws-wiz
* Fragmento: ip=1:

6- Análisis de recursos web: Elige un sitio web popular y analiza cómo se utilizan los elementos fundamentales de la web, como HTML, CSS y JavaScript. Describe brevemente cómo se implementan estos elementos en el sitio y qué contribución hacen al diseño y funcionalidad de la página.

Al html lo utilizo para almacenar las etiquetas, marcadores, el logo y cuadros de busqueda, este es el logo:

Texto

Descripción generada automáticamente

El css es minimalista y simple, este es el del logo:

Texto

Descripción generada automáticamente

El uso de JavaScript se centra principalmente en mejorar la interactividad y la experiencia del usuario. Como autocompletado de busqueda, validacion de formularios, seguimiento de interacciones del Usuario, etc.

7- Análisis de seguridad web: Investiga y enumera las principales preocupaciones de seguridad en la web actual junto con una breve descripción de las mismas.

Las principales preocupaciones de seguridad en la web actual incluyen una variedad de amenazas y desafíos que las organizaciones deben enfrentar para proteger sus sistemas y datos:

1. **Vulnerabilidades en la Nube**: Las configuraciones incorrectas y los controles de acceso inadecuados en los entornos de la nube pueden exponer datos sensibles
2. **Ataques DDoS (Distributed Denial of Service)**: Estos ataques intentan sobrecargar los servidores con tráfico para sacarlos de línea, a menudo como una distracción para ocultar otros vectores de ataque.
3. **Vulnerabilidades de Software y Cadena de Suministro**: La explotación de vulnerabilidades en software, particularmente en componentes de la cadena de suministro de software, sigue siendo una preocupación importante.

8- Comparativa de navegadores web: Elige dos navegadores web populares y realiza una comparativa de sus características y funcionalidades. Describe las ventajas y desventajas de cada navegador y menciona cualquier característica que los distinga.

|  |  |  |
| --- | --- | --- |
| navegador | google | firefox |
| ventajas | Integración con el Ecosistema de Google, rendimiento y velocidad, gran biblioteca de extensiones, soporte de formatos multimedia | Privacidad y seguridad, personalización y extensiones, menor consumo de recursos, independencia de Google |
| desventajas | Consumo de recursos, preocupaciones de privacidad | Compatibilidad, velocidad de actualizaciones |

9- Exploración de tecnologías emergentes: Investiga sobre las tecnologías emergentes en el campo de la web, como WebVR, WebRTC o Progressive Web Apps (PWA). Elige una de estas tecnologías y describe en qué consiste, qué beneficios ofrece y cómo podría impactar en el futuro de la web.

las tecnologías emergentes como la inteligencia artificial, el Internet de las cosas (IoT) y la realidad aumentada. Esto está dando lugar a nuevas experiencias de usuario, como la búsqueda por voz, los asistentes virtuales y los dispositivos conectados, que están redefiniendo la forma en que interactuamos con la web y el mundo que nos rodea.

Las Progressive Web Apps son aplicaciones web que combinan las mejores características de las aplicaciones móviles y las páginas web. Están diseñadas para ser confiables, rápidas y atractivas, ofreciendo una experiencia de usuario similar a la de una aplicación nativa, pero son accesibles directamente desde un navegador web sin necesidad de instalación desde una tienda de aplicaciones.

**Beneficios:**

1. **Disponibilidad Offline:** Pueden funcionar sin conexión a Internet o en conexiones de red intermitentes, utilizando almacenamiento en caché y tecnologías de almacenamiento local.
2. **Interactividad y Velocidad:** Proporcionan una experiencia de usuario rápida y fluida, con animaciones suaves y respuestas rápidas a las interacciones del usuario.
3. **Actualización Automática:** Se actualizan automáticamente en segundo plano, asegurando que los usuarios siempre tengan la última versión sin necesidad de descargar o instalar actualizaciones manualmente.
4. **Seguridad:** Utilizan HTTPS para garantizar la seguridad de los datos transmitidos, cumpliendo con los estándares de seguridad modernos.

**Impacto en el Futuro:** Las PWA están democratizando el acceso a aplicaciones móviles de alta calidad al eliminar las barreras de entrada como las tiendas de aplicaciones y los requisitos de instalación.

10- Análisis de casos de éxito en la web: Investiga sobre proyectos web exitosos y analiza qué elementos contribuyeron a su éxito. Elige al menos dos casos y describe cómo se utilizaron los fundamentos de la web para lograr resultados destacados.

Casos elegidos: Spotify y Airbnb. Ambos son ejemplos emblemáticos de cómo la combinación de una experiencia de usuario optimizada, la implementación de tecnologías web avanzadas, y la innovación en el modelo de negocio han contribuido a su éxito.

Elementos que contribuyeron al éxito de airbnb:

**Diseño Centrado en el Usuario:** Airbnb priorizó la experiencia del usuario desde el principio. La interfaz es intuitiva, limpia, y está diseñada para facilitar la navegación y el proceso de reserva.

**Tecnologías Web Avanzadas:** Airbnb ha adoptado Progressive Web Apps (PWA) para ofrecer una experiencia rápida y confiable, incluso en conexiones de red lentas. La aplicación móvil y la web son altamente interactivas y responden rápidamente a las acciones del usuario, lo que mejora la satisfacción del usuario.

Spotify:

Elementos que contribuyeron al éxito de spotify:

**Streaming en Tiempo Real:** Spotify utiliza tecnologías web avanzadas, como WebRTC (Comunicaciones en Tiempo Real), para ofrecer streaming de música en tiempo real. Esto permite a los usuarios escuchar música sin interrupciones y con baja latencia, mejorando la experiencia general del usuario.

**Algoritmos de Recomendación Personalizada:** Spotify utiliza algoritmos avanzados de aprendizaje automático y análisis de datos para ofrecer recomendaciones personalizadas a cada usuario. La función de "Discover Weekly" y las listas de reproducción personalizadas han sido fundamentales para mantener a los usuarios comprometidos.

11-Investigación sobre la arquitectura cliente-servidor: Investiga y describe los conceptos fundamentales de la arquitectura cliente-servidor. Explica cómo funciona esta arquitectura en el contexto de la web y cuál es el papel del cliente y el servidor en la comunicación.

La **Arquitectura Cliente-Servidor** es un modelo de diseño de software que distribuye las funciones de una aplicación entre dos tipos de entidades: el cliente y el servidor. Esta arquitectura establece una comunicación estructurada y eficiente entre estos dos componentes, permitiendo que los usuarios accedan y manipulen datos y recursos de manera remota a través de una red

En este modelo, el cliente es la interfaz de usuario o la aplicación que solicita y consume los servicios proporcionados por el servidor. Su función principal es enviar solicitudes al servidor y mostrar los resultados al usuario de una manera comprensible.

el servidor es el **componente que almacena, procesa y gestiona los datos y recursos de la aplicación**. Es responsable de recibir las solicitudes del cliente, procesarlas y enviar las respuestas adecuadas de vuelta al cliente. El servidor generalmente se ejecuta en una computadora dedicada con una mayor capacidad de procesamiento y almacenamiento, lo que le permite manejar múltiples solicitudes de clientes simultáneamente.

Contexto de la web:

En una aplicación web típica, la interacción entre cliente y servidor sigue un patrón de comunicación basado en solicitudes y respuestas. Este proceso puede resumirse en los siguientes pasos:

1. Solicitud del cliente
2. Procesamiento del servidor
3. Respuesta del servidor
4. Renderizado por el cliente

12-Comparativa de arquitecturas: Compara la arquitectura cliente-servidor con otras arquitecturas de comunicación, como la arquitectura de punto a punto (peer-to-peer). Describe las diferencias clave entre estas arquitecturas y menciona los casos de uso más apropiados para cada una.

|  |  |  |
| --- | --- | --- |
| Caracteristica | Cliente-servidor | Punto a punto (p2p) |
| Estructura | Centralizada | Distribuida |
| Roles de los nodos | Definidos (cliente y servidor) | Dinamicos (todos los nodos son iguales) |
| Punto unico de falla | Si | No |
| Escalabilidad | Aumentando servidores | Aumentando nodos |
| Almacenamiento/procesamiento | Centralizado en el servidor | Distribuido entre los nodos |
| Facilidad de mantenimiento | Relativamente facil | Complejo |
| Seguridad | Mas facil de gestionar | Mas dificil de asegurar |
| Casos de uso | Aplicaciones web, servicios empresariales | Comparticion de archivos, blockchain |

13-Análisis del Protocolo HTTP: Describe en detalle el Protocolo HTTP (Hypertext Transfer Protocol). Explica cómo se establece una conexión HTTP, cómo se envían las solicitudes y respuestas, y qué tipo de información se puede transmitir a través de este protocolo.

El Protocolo de HTTP es el protocolo fundamental utilizado para la transmisión de datos en la web. Es un protocolo de comunicación que define cómo se envían y reciben mensajes entre un cliente (como un navegador web) y un servidor. HTTP es la base para la comunicación en la World Wide Web, permitiendo la transferencia de recursos como documentos HTML, imágenes, videos, y otros tipos de contenido.

Establecimiento de una Conexión HTTP:

1. Resolucion de DNS:

Antes de que se establezca una conexión HTTP, el cliente necesita saber la dirección IP del servidor. Esto se logra mediante la resolución de DNS, donde el nombre de dominio (por ejemplo, [www.example.com](http://www.example.com)) se traduce en una dirección IP.

1. Apertura de Conexión TCP:

Una vez que se conoce la dirección IP del servidor, el cliente inicia una conexión TCP (Protocolo de Control de Transmisión) con el servidor en el puerto 80 (el puerto predeterminado para HTTP). Este proceso incluye un handshake de tres vías para establecer una conexión fiable.

1. Envio de la solicitud HTTP:

Con la conexión TCP establecida, el cliente envía una solicitud HTTP al servidor.

Tipo de informacion transmitida a travez de HTTP:

1. Documentos HTML
2. Imágenes
3. Archivos de audio y video
4. Datos JSON o XML
5. Archivos CSS y JavaScript

14-Identificación de métodos HTTP: Enumera y describe los principales métodos HTTP, como GET, POST, PUT y DELETE. Explica en qué situaciones se utiliza cada método y qué tipo de operaciones realizan en el servidor.

Tipos de metodos HTTP:

1. GET: El método GET solicita la representación de un recurso específico en el servidor. Se utiliza principalmente para recuperar datos sin realizar cambios en el servidor. Las solicitudes GET no deberían tener efectos secundarios, es decir, no deberían alterar el estado del servidor.

Uso comun: navegacion web, recuperacion de datos, enlaces y formularios

1. POST: El método POST envía datos al servidor para crear un nuevo recurso o realizar un procesamiento en el servidor. A diferencia de GET, los datos enviados a través de POST no se incluyen en la URL, sino en el cuerpo de la solicitud. Uso comun: envio de formularios, creacion de recursos, autenticacion
2. PUT: El método PUT actualiza un recurso existente en el servidor con los datos proporcionados. Si el recurso no existe, algunos servidores pueden optar por crearlo. PUT es idempotente, lo que significa que múltiples solicitudes PUT idénticas tendrán el mismo efecto que una sola solicitud. Uso comun: actualizacion de recursos
3. DELETE: El método DELETE solicita que se elimine un recurso específico del servidor. Al igual que PUT, DELETE es idempotente, lo que significa que hacer varias solicitudes DELETE sobre el mismo recurso no causará efectos adicionales. Uso comun: eliminacion de recursos
4. HEAD: El método HEAD es idéntico a GET, pero la diferencia es que el servidor no devuelve el cuerpo del mensaje en la respuesta, solo las cabeceras. Se utiliza para obtener metadatos sobre el recurso sin transferir todo su contenido. Uso comun: verificacion de existencia, verificacion de estado

15-Códigos de estado HTTP: Investiga y enumera los códigos de estado más comunes en las respuestas HTTP, como 200 OK, 404 Not Found y 500 Internal Server Error. Explica el significado de cada código y cómo se utilizan para indicar el estado de una solicitud.

1. Codigos de estado 2xx: Éxito:

* 200 OK: **Significado:** La solicitud fue exitosa, y el servidor devolvió el recurso solicitado.

**Uso Común:** Cuando se accede a una página web con éxito o se recupera un recurso a través de una API.

* 201 created: **Significado:** La solicitud ha sido exitosa y ha dado lugar a la creación de un nuevo recurso.

**Uso Común:** En respuesta a un método POST que crea un nuevo recurso, como un nuevo registro en una base de datos.

1. Codigos de estado 3xx: redireccion

* 301 moved permanently: **Significado:** El recurso solicitado ha sido movido permanentemente a una nueva URL, y las solicitudes futuras deberían usar la nueva URL.

**Uso Común:** Cuando una página web ha cambiado de dirección y se desea redirigir automáticamente al nuevo recurso.

* 302 found: **Significado:** El recurso solicitado reside temporalmente en una URL diferente, y se debe utilizar la URL devuelta en la cabecera Location para futuras solicitudes.

**Uso Común:** Redirecciones temporales, como cuando un recurso está temporalmente en otra ubicación.

1. Codigos de estado 4xx: error del cliente

* 401 unauthorized: **Significado:** El cliente debe autenticarse para obtener el recurso solicitado.

**Uso Común:** Cuando se requiere autenticación y el cliente no ha proporcionado las credenciales o estas son inválidas.

* 404 Not Found: **Significado:** El servidor no puede encontrar el recurso solicitado. Esto no indica si el recurso alguna vez existió.

**Uso Común:** Cuando se solicita una página web o un recurso que no existe.

1. Codigos de estado 5xx: error del servidor

* 500 internal Server Error: **Significado:** El servidor encontró una condición inesperada que le impidió cumplir con la solicitud.

**Uso Común:** Cuando ocurre un error genérico en el servidor, sin información específica sobre la causa.

16-Manipulación de cabeceras HTTP: Describe el propósito de las cabeceras HTTP y cómo se utilizan para transmitir información adicional en las solicitudes y respuestas. Elige tres ejemplos de cabeceras HTTP y explica su función y aplicación.

Las cabeceras HTTP tienen varios propósitos fundamentales:

1. **Transmitir Información sobre el Contenido:** Especifican detalles sobre el tipo de contenido, su tamaño, codificación, y otras características relevantes.
2. **Controlar el Flujo de la Comunicación:** Permiten definir cómo deben comportarse el cliente y el servidor durante la transferencia de datos, como el manejo de la caché o la autenticación.
3. **Proporcionar Contexto Adicional:** Facilitan la comunicación de información contextual, como la configuración regional, el tipo de navegador que realiza la solicitud, o las capacidades del cliente.

**Ejemplos de Cabeceras HTTP y sus Funciones**

**1. Content-Type**

* **Función:** La cabecera Content-Type especifica el tipo de contenido que se está enviando en el cuerpo de la solicitud o respuesta. Esto le permite al cliente o al servidor interpretar correctamente los datos transferidos.
* **Aplicación:**
  + **En Respuestas:** Cuando un servidor responde a una solicitud con un recurso, como una página HTML, imagen o JSON, la cabecera Content-Type indica al cliente cómo debe interpretar el contenido.
  + **En Solicitudes:** Al enviar datos al servidor, como en un formulario o una API, la cabecera Content-Type especifica el formato de los datos enviados (por ejemplo, application/json para JSON).
* **Ejemplo:**
  + Content-Type: text/html indica que el contenido es una página HTML.
  + Content-Type: application/json indica que el contenido es un objeto JSON.

**2. Authorization**

* **Función:** La cabecera Authorization se utiliza para enviar credenciales que autentican al cliente ante el servidor. Es esencial para acceder a recursos protegidos por autenticación.
* **Aplicación:**
  + **Autenticación Básica:** En el caso de la autenticación básica, la cabecera contiene un token codificado en Base64 que representa las credenciales del usuario.
  + **Tokens de Autenticación:** En sistemas más avanzados, como OAuth, la cabecera Authorization puede incluir un token que el servidor valida para otorgar acceso.
* **Ejemplo:**
  + Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ== es un ejemplo de autenticación básica con un nombre de usuario y contraseña codificados.
  + Authorization: Bearer <token> se utiliza en sistemas de autenticación basados en tokens, como OAuth.

**3. Cache-Control**

* **Función:** La cabecera Cache-Control proporciona directivas para la caché del navegador o de intermediarios (como proxies), definiendo cómo y durante cuánto tiempo debe almacenarse la respuesta en la caché.
* **Aplicación:**
  + **Mejorar Rendimiento:** Controlando la caché de los recursos, se puede mejorar significativamente el rendimiento de una aplicación web, reduciendo la necesidad de solicitudes repetidas al servidor.
  + **Evitar Datos Obsoletos:** También se usa para asegurar que el contenido no se cachee y que los usuarios reciban siempre la versión más actualizada.
* **Ejemplo:**
  + Cache-Control: no-cache indica que la respuesta no debe almacenarse en caché.
  + Cache-Control: max-age=3600 permite que la respuesta se almacene en caché durante una hora (3600 segundos).

17-Implementación de solicitudes HTTP: Utilizando un lenguaje de programación de tu elección, crea un programa que realice una solicitud HTTP a un servidor de tu elección. Muestra cómo se configuran los encabezados y cómo se manejan las respuestas del servidor.

Texto

Descripción generada automáticamente

**Encabezados (Headers):**

* User-Agent: Define el agente de usuario que se envía al servidor. Es común especificar un agente que represente un navegador para evitar bloqueos.
* Accept: Indica qué tipo de contenido espera recibir el cliente. En este caso, esperamos un contenido en formato JSON.
* Authorization: Es opcional y se utiliza solo si necesitas autenticarte para acceder al recurso. Aquí se muestra cómo incluir un token de acceso.

**Manejo de la Respuesta:**

* Si la respuesta es exitosa (código 200), el cuerpo de la respuesta se convierte de JSON a un diccionario Python usando response.json().
* Si la respuesta no es exitosa, se maneja el error mostrando el código de estado y el mensaje de error.

18-Seguridad en el Protocolo HTTP: Investiga y describe las vulnerabilidades de seguridad asociadas con el Protocolo HTTP. Explica cómo se pueden mitigar estas vulnerabilidades utilizando medidas como HTTPS y certificados SSL/TLS.

**Vulnerabilidades de Seguridad en HTTP**

1. **Transmisión de Datos en Texto Plano:**
   * En HTTP, los datos (incluyendo credenciales de usuario, información sensible, etc.) se transmiten en texto plano, lo que permite a los atacantes interceptar y leer los datos a través de ataques de intermediarios como "man-in-the-middle" (MitM).
2. **Suplantación de Sitios Web (Phishing):**
   * Los usuarios pueden ser engañados para acceder a sitios falsos que imitan a sitios legítimos. Dado que HTTP no garantiza la autenticidad del sitio, es difícil para los usuarios identificar sitios fraudulentos.
3. **Inyección de Contenido:**
   * HTTP permite que los datos sean manipulados durante la transmisión, lo que podría llevar a la inyección de código malicioso, como scripts en sitios web legítimos (Cross-Site Scripting - XSS).

**Mitigación de Vulnerabilidades con HTTPS y SSL/TLS**

1. **Uso de HTTPS:**
   * HTTPS (HTTP Secure) es la versión segura de HTTP. Utiliza los protocolos SSL/TLS para cifrar la comunicación entre el cliente y el servidor, asegurando que los datos no puedan ser interceptados o manipulados.
   * **Beneficios de HTTPS:**
     + **Cifrado de Datos:** Toda la información intercambiada está cifrada, lo que protege contra la interceptación.
     + **Autenticidad del Sitio Web:** Los certificados SSL/TLS garantizan que el sitio al que accede el usuario es el legítimo.
     + **Integridad de los Datos:** Los datos no pueden ser alterados sin ser detectados.
2. **Certificados SSL/TLS:**
   * Los certificados SSL/TLS son emitidos por Autoridades Certificadoras (CA) y contienen información sobre la identidad del sitio web y su clave pública.
   * **Proceso de Validación:**
     + **Verificación de Identidad:** La CA verifica la identidad del solicitante antes de emitir un certificado, asegurando que el sitio web es quien dice ser.
     + **Establecimiento de Conexiones Seguras:** Durante el establecimiento de una conexión HTTPS, el certificado es presentado al cliente, que lo valida antes de continuar con la conexión.

19-Protocolo FTP: Describe el Protocolo FTP (File Transfer Protocol) y cómo se utiliza para transferir archivos entre clientes y servidores. Explica los comandos básicos utilizados en una sesión FTP y cómo se establece la autenticación.

El Protocolo FTP (File Transfer Protocol) es un protocolo estándar para la transferencia de archivos entre un cliente y un servidor en una red. FTP opera sobre la arquitectura cliente-servidor y utiliza conexiones separadas para comandos y datos.

**Funcionamiento del Protocolo FTP**

1. **Sesión FTP:**
   * Una sesión FTP comienza cuando un cliente se conecta a un servidor FTP a través de los puertos 20 y 21. El puerto 21 se usa para el control de la conexión (comandos y respuestas), mientras que el puerto 20 se usa para la transferencia de datos.
2. **Autenticación:**
   * **Usuario y Contraseña:** Para acceder a un servidor FTP, el cliente debe autenticarse proporcionando un nombre de usuario y una contraseña. Algunos servidores FTP permiten acceso anónimo, donde el nombre de usuario es "anonymous" y la contraseña es cualquier dirección de correo electrónico.
   * **Comando USER:** Se usa para enviar el nombre de usuario.
   * **Comando PASS:** Se usa para enviar la contraseña.

**Comandos Básicos en una Sesión FTP**

1. **Comando LIST:**
   * **Función:** Lista los archivos y directorios en el directorio actual del servidor.
   * **Uso:** LIST
2. **Comando RETR:**
   * **Función:** Descarga un archivo desde el servidor al cliente.
   * **Uso:** RETR nombre\_del\_archivo
3. **Comando STOR:**
   * **Función:** Sube un archivo desde el cliente al servidor.
   * **Uso:** STOR nombre\_del\_archivo
4. **Comando QUIT:**
   * **Función:** Finaliza la sesión FTP.
   * **Uso:** QUIT

20-Protocolo SSL y TLS: Investiga y describe los protocolos SSL (Secure Sockets Layer) y TLS (Transport Layer Security), utilizados para proporcionar seguridad en las comunicaciones web. Explica cómo se establecen las conexiones seguras y cómo se validan los certificados digitales.

SSL (Secure Sockets Layer) y TLS (Transport Layer Security) son protocolos criptográficos diseñados para proporcionar seguridad en las comunicaciones a través de redes, incluidas las comunicaciones web.

**Funcionamiento de SSL/TLS**

1. **Establecimiento de una Conexión Segura (Handshake):**
   * **Negociación de Parámetros:** El cliente y el servidor negocian los algoritmos criptográficos que se utilizarán (cifrado, compresión, etc.).
   * **Intercambio de Claves:** El servidor envía su certificado SSL/TLS al cliente, que contiene la clave pública. El cliente verifica el certificado y utiliza la clave pública para cifrar una clave de sesión, que solo el servidor puede descifrar con su clave privada.
   * **Generación de la Clave de Sesión:** Una vez que el servidor descifra la clave de sesión, ambos lados la usan para cifrar y descifrar los datos intercambiados durante la sesión.
2. **Validación de Certificados Digitales:**
   * **Autoridades Certificadoras (CA):** Son entidades de confianza que emiten certificados SSL/TLS. El cliente valida el certificado recibido comprobando que fue emitido por una CA reconocida.
   * **Cadena de Confianza:** La validación puede implicar la verificación de una cadena de certificados hasta una CA raíz confiable.
3. **Renegociación y Cierre:**
   * Durante una sesión, el cliente y el servidor pueden renegociar los parámetros de cifrado o cerrar la conexión de manera segura para proteger los datos en tránsito.

BIBLIOGRAFIA:

<https://developer.mozilla.org/en-US/> esta fue la pagina donde me base para fundamentar la mayoría de respuestas y el resto del documento de teoría proporcionado por la catedra