# Simulacro Examen 2 - 2º DAW

1. **Explica brevemente los siguientes conceptos y da un ejemplo práctico de cada uno:**

   **a) Cliente y servidor**
   **b) Protocolo HTTP y HTTPS**
   **c) API REST**
   **d) Ciclo de vida de una petición en una aplicación web**
   **e) Diferencia entre GET y POST**

**a) Cliente y Servidor**

En informática, un cliente es el dispositivo o programa que solicita recursos o servicios, mientras que un servidor es el dispositivo o programa que proporciona esos recursos o servicios.

Ejemplo Práctico: Cuando accedes a una página web desde tu navegador (cliente), el navegador envía una solicitud al servidor web, que le devuelve los datos necesarios para mostrar la página.

**b) Protocolo HTTP y HTTPS**

HTTP (Hypertext Transfer Protocol) es el protocolo de comunicación utilizado para transferir información en la web. HTTPS (HTTP Secure) es la versión segura de HTTP, en la que la información se cifra mediante SSL/TLS para proteger los datos de posibles intercepciones.

Ejemplo Práctico: Al comprar en línea, el sitio web de la tienda suele usar HTTPS para proteger tu información de pago, como los datos de tu tarjeta de crédito, mientras se envían al servidor. En cambio, sitios de información o consulta, como blogs, pueden usar HTTP si no manejan datos sensibles.

**c) API REST**

Una API REST (Representational State Transfer) es un conjunto de reglas que permite que diferentes sistemas se comuniquen a través de HTTP. Las APIs REST permiten el intercambio de datos y la integración entre aplicaciones de manera sencilla y flexible, generalmente en formato JSON o XML.

Ejemplo Práctico: Un sitio de clima que usa una API REST puede ofrecer la información meteorológica a otras aplicaciones.

**d) Ciclo de Vida de una Petición en una Aplicación Web**

Este ciclo se refiere al proceso completo de una solicitud enviada por un cliente a una aplicación web y la respuesta generada por el servidor. Incluye la recepción de la solicitud, procesamiento en el servidor, ejecución de lógica y generación de una respuesta que el servidor devuelve al cliente.

Ejemplo Práctico: Cuando buscas un producto en una tienda en línea:

1. El navegador envía una solicitud al servidor de la tienda para buscar el producto.
2. El servidor procesa la solicitud, busca el producto en su base de datos.
3. Luego genera una respuesta con los resultados y los envía al navegador, donde se muestra la lista de productos al usuario.

**e) Diferencia entre GET y POST**

GET y POST son métodos de HTTP que indican el tipo de solicitud que hace un cliente:

GET: Solicita datos del servidor sin modificar nada, y los datos se envían a través de la URL.
POST: Envía datos al servidor, a menudo para guardar o modificar datos en el servidor, y los datos se envían en el cuerpo de la solicitud.

Ejemplo Práctico:

GET: Al abrir una página de noticias, tu navegador realiza una solicitud GET para obtener los artículos.
POST: Cuando llenas un formulario de registro y lo envías, el navegador usa POST para enviar tus datos al servidor de registro, donde serán guardados en la base de datos.

## 2. Describe qué es una arquitectura de tres capas y cuáles son sus componentes principales. ¿Por qué es importante esta estructura en el desarrollo de aplicaciones web?

Una arquitectura de tres capas es un modelo de diseño de software que divide una aplicación en tres capas o niveles lógicos independientes, cada uno con una responsabilidad específica.

### Capa de Presentación (Front-End)

Esta capa es la interfaz de usuario de la aplicación, donde el usuario interactúa con el sistema. Maneja el diseño visual, la navegación y la presentación de la información.

Ejemplos de tecnologías: *HTML, CSS, JavaScript, y frameworks* como *React, Angular o Vue.js.*

### Capa Lógica de Negocio (Back-End)

Esta capa contiene la lógica de negocio de la aplicación, donde se procesan las reglas y los procedimientos que determinan el comportamiento de la aplicación. También maneja las validaciones, cálculos y procesamiento de los datos que provienen de la capa de presentación o de la base de datos.

Ejemplos de tecnologías: *Java, Python, PHP, Node.js y frameworks* como *Spring, Django o Express.*

### Capa de Datos (Base de Datos)

Es la capa responsable de almacenar, recuperar y manipular los datos. La capa de datos mantiene la persistencia y organización de la información y se comunica con la capa lógica de negocio para suministrar o actualizar datos.

Ejemplos de tecnologías*: MySQL, PostgreSQL, MongoDB, Oracle*.

La arquitectura de tres capas es fundamental en el desarrollo de aplicaciones web modernas, ya que proporciona una estructura bien organizada que simplifica el mantenimiento, mejora la seguridad y facilita la escalabilidad de la aplicación.

3. **Diseña una página HTML básica que incluya los siguientes elementos:**

    a. **Un encabezado principal con el título "Mi primera página web"**
    b. **Una lista ordenada de tus lenguajes de programación favoritos**
    c. **Una imagen centrada en la página**
    d. **Un enlace a otra página web que se abra en una nueva pestaña**
    e. **Incluye estilos CSS para darle color al encabezado y centrar el texto de la lista**

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Mi primera pagina web</title>
    <style>

        h1 {
            color: red;
            text-align: center;
        }

        ol {
            text-align: center;
            list-style-type: decimal;
        }

        .image-container {
            text-align: center;
        }

    </style>
</head>
<body>

    <h1>Mi primera página web</h1>

    <ol>
        <li>JavaScript</li>
        <li>Python</li>
        <li>Java</li>
        <li>PHP</li>
        <li>C++</li>
    </ol>

    <div class="image-container">
        <img src="https://beecrowd.com/wp-content/uploads/2024/04/2022-07-19-Melhores-cursos-de-Python.jpg"
alt="Imagen de ejemplo centrada" width="480" height="360">
    </div>

    <p>
        Visita <a href="https://es.wikipedia.org/wiki/Historia_de_Python" target="_blank">Wikipedia</a> para
aprender más Python.
    </p>
</body>
</html>
```

4. **Escribe un script en JavaScript que pida al usuario su nombre y le muestre un mensaje de bienvenida en una ventana emergente (alert). Ejemplo: "¡Bienvenido, [Nombre]!"**

```
function bienvenida() {
    // Solicita el nombre del usuario
    const nombre = prompt("Por favor, ingresa tu nombre:");

    // Comprueba que el usuario haya ingresado un nombre
    if (nombre) {
        // Muestra el mensaje de bienvenida
        alert(`¡Bienvenido, ${nombre}!`);
    }
}

// Para ejecutar la funcion en un html
window.onload = bienvenida();
```

5. **Crea un formulario HTML para recoger la información de contacto de un usuario, incluyendo nombre, correo electrónico y número de teléfono. Utiliza atributos HTML para validar el formato del correo electrónico y que el campo del teléfono solo permita números.**

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Formulario de Contacto</title>
</head>
<body>
    <h2>Formulario de Contacto</h2>
    <form action="#" method="post">

        <label for="nombre">Nombre:</label><br>
        <input type="text" id="nombre" name="nombre" required><br><br>

        <label for="email">Correo Electrónico:</label><br>
        <input type="email" id="email" name="email" required><br><br> <!-- Formato email -->

        <label for="telefono">Número de Teléfono:</label><br>
        <input type="tel" id="telefono" name="telefono" pattern="[0-9]+" required><br><br> <!-- Solo permite num del
0 al 9 -->

        <button type="submit">Enviar</button>
    </form>
</body>
</html>
```

## 6. Imagina que tienes que desarrollar una aplicación web de gestión de tareas. Explica los pasos iníciales para planificar este proyecto:

a) **Define los requerimientos básicos**
b) **Describe cómo estructurarías el proyecto a nivel de frontend y backend**
c) **Explica qué tecnologías o lenguajes usarías y por qué**
d) **Menciona cómo llevarías a cabo el testeo y la puesta en producción ¿Qué aspectos de seguridad considerarías para una aplicación de este tipo?**

### a) Definición de Requerimientos Básicos

**Requerimientos Funcionales**

- **Autenticación de Usuarios:** Permitir a los usuarios registrarse e iniciar sesión.
- **Gestión de Tareas**: Crear, editar, eliminar y marcar tareas como completadas.
- **Organización de Tareas:** Posibilidad de clasificar tareas por etiquetas, prioridades o fechas de vencimiento.
- **Notificaciones y Recordatorios**: Opcionalmente, enviar recordatorios de las tareas próximas o vencidas.
- **Panel de Usuario:** Mostrar al usuario un resumen de sus tareas, tareas pendientes y completadas.

**Requerimientos No Funcionales**

- **Usabilidad**: Interfaz intuitiva y fácil de usar.
- **Escalabilidad**: La aplicación debe poder manejar un aumento de usuarios y datos.
- **Seguridad:** Protección de los datos del usuario, especialmente en lo referente a las contraseñas.

### b) Estructuración del Proyecto (Frontend y Backend)

**Frontend**

- Interfaz de Usuario (UI): Diseñar una interfaz que incluya:
- Página de inicio de sesión y registro.
- Panel de tareas donde el usuario puede ver, añadir y organizar sus tareas.
- Componentes: Crear componentes reutilizables para el formulario de tareas, listas de tareas y barra de navegación.
- Comunicación con Backend: Usar una capa de servicios o controladores en el frontend para manejar la comunicación con la API del backend mediante solicitudes HTTP.

**Backend**

- API REST: Crear una API REST que permita realizar las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) sobre las tareas, así como manejar la autenticación de usuarios.
- Estructura de Bases de Datos: Diseñar una base de datos con tablas para Usuarios, Tareas y Etiquetas/Categorías para clasificar las tareas.
- Gestión de Autenticación y Autorización: Implementar autenticación de usuarios y tokens (por ejemplo, JWT) para asegurar las solicitudes.

**c) Tecnologías o Lenguajes a Utilizar**

**Frontend**

- **HTML, CSS y JavaScript**: Base fundamental para cualquier desarrollo web.
- **Frameworks/ Librerías**: Usar ***React*** o Vue.js, que facilitan la creación de componentes reutilizables y ofrecen un entorno de desarrollo modular.
- **CSS Frameworks**: ***Bootstrap*** o Tailwind CSS para agilizar el diseño visual.

**Backend**

- **Lenguaje de Servidor:** Usar ***Node.js*** con Express por su integración sencilla con JavaScript (ya usado en el frontend) o elegir ***Python*** con Django si se busca una estructura más robusta y con herramientas integradas.
- **Base de Datos**: MongoDB si se prefiere un modelo NoSQL y mayor flexibilidad en el esquema de datos, o PostgreSQL/***MySQL*** si se requiere un modelo relacional y estructurado.
- **Control de Versiones**: ***Git*** para mantener un historial de los cambios y permitir la colaboración del equipo.

**d) Testeo, Puesta en Producción y Consideraciones de Seguridad**

**Testeo**

- **Pruebas Unitarias**: Realizar pruebas unitarias para verificar la funcionalidad de cada componente del frontend y de cada endpoint del backend.
- **Pruebas de Integración**: Verificar que el frontend se comunica correctamente con la API del backend y que la información fluye de forma adecuada.
- **Pruebas de Usabilidad:** Realizar pruebas con usuarios para asegurarse de que la aplicación es intuitiva.
- **Pruebas de Seguridad**: Realizar pruebas de seguridad, como validación de inputs, para prevenir ***inyecciones SQL*** o ataques XSS.

**Puesta en Producción**

- **Configuración del Servidor**: Configurar un servidor en la nube (como ***AWS***, DigitalOcean, o Heroku) para alojar la aplicación.
- **Despliegue Contenerizado:** Usar ***Docker*** para empaquetar la aplicación y sus dependencias, asegurando que se ejecute de manera consistente en cualquier entorno.
- **Balanceo de Carga:** Si la aplicación crece, utilizar un balanceador de carga para distribuir el tráfico y mejorar la disponibilidad.

**Consideraciones de Seguridad**

- **Cifrado de Datos:** Cifrar las contraseñas en la base de datos con algoritmos seguros como ***bcrypt***.
- **Tokens de Autenticación (JWT):** Utilizar ***JWT*** para la autenticación, lo que asegura que cada usuario autenticado tenga su propio token único.
- **HTTPS:** Asegurarse de que la aplicación esté ***encriptada con HTTPS*** para proteger la información durante la transmisión entre el cliente y el servidor.
- **Validación de Entradas**: Sanitizar todas las entradas de usuario para prevenir ataques como SQL Injection y Cross-Site Scripting (XSS).
- **Manejo de Sesiones y Caducidad de Tokens:** Implementar una caducidad para los tokens de autenticación y asegurar que los tokens expirados no puedan reutilizarse.

7. **Diseña un formulario HTML para un sitio web de registro de usuarios, que incluya los siguientes campos:**

   - Nombre completo
   - Fecha de nacimiento
   - Correo electrónico
   - Contraseña
   - Selección de género (hombre, mujer, otro)
   - Casilla para aceptar términos y condiciones
   - Asegúrate de que el formulario esté centrado en la página.
   - Aplica bordes redondeados a los campos de entrada.
   - Usa un color de fondo claro y estilos de texto para mejorar la accesibilidad.

```html
<div class="form-container">
  <h2>Registro de Usuarios</h2>
  <form id="registration-form" novalidate>
    <div class="form-group">
      <label for="name">Nombre Completo</label>
      <input type="text" id="name" name="name" required>
      <div class="error-message" id="name-error"></div>
    </div>
    <div class="form-group">
      <label for="dob">Fecha de Nacimiento</label>
      <input type="date" id="dob" name="dob" required>
    </div>
    <div class="form-group">
      <label for="email">Correo Electrónico</label>
      <input type="email" id="email" name="email" required>
      <div class="error-message" id="email-error"></div>
    </div>
    <div class="form-group">
      <label for="password">Contraseña</label>
      <input type="password" id="password" name="password" required>
      <div class="error-message" id="password-error"></div>
    </div>
    <div class="form-group">
      <label for="gender">Género</label>
      <select id="gender" name="gender" required>
        <option value="">Seleccione</option>
        <option value="male">Hombre</option>
        <option value="female">Mujer</option>
        <option value="other">Otro</option>
      </select>
    </div>
    <div class="form-group">
      <label>
        <input type="checkbox" id="terms" name="terms" required> Acepto los términos y condiciones
      </label>
      <div class="error-message" id="terms-error"></div>
    </div>
    <div class="form-button">
      <button type="submit">Registrar</button>
    </div>
  </form>
</div>
```

Agrega validación al formulario de registro de usuarios:

- La contraseña debe tener al menos 8 caracteres.
- El correo electrónico debe seguir el formato correcto.
- El campo de nombre completo debe ser obligatorio.
- Muestra un mensaje de error en rojo debajo de cada campo que no cumpla con los requisitos.

```
  <script>
    const form = document.getElementById('registration-form');

    form.addEventListener('submit', function (event) {
      event.preventDefault();

      // Limpiar mensajes de error
      document.querySelectorAll('.error-message').forEach(error => error.textContent = '');

      let valid = true;

      // Validar nombre completo
      const name = document.getElementById('name').value.trim();
      if (!name) {
        document.getElementById('name-error').textContent = 'El nombre completo es obligatorio.';
        valid = false;
      }

      // Validar correo electrónico
      const email = document.getElementById('email').value.trim();
      const emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
      if (!emailPattern.test(email)) {
        document.getElementById('email-error').textContent = 'Ingrese un correo válido.';
        valid = false;
      }

      // Validar contraseña
      const password = document.getElementById('password').value.trim();
      if (password.length < 8) {
        document.getElementById('password-error').textContent = 'La contraseña debe tener al menos 8
caracteres.';
        valid = false;
      }

      // Validar términos y condiciones
      const terms = document.getElementById('terms').checked;
      if (!terms) {
        document.getElementById('terms-error').textContent = 'Debe aceptar los términos y
condiciones.';
        valid = false;
      }

      if (valid) {
        alert('Formulario enviado con éxito.');
        form.reset();
      }
    });
  </script>
```

estilos.css

```css
body {
    font-family: Arial, sans-serif;
    background-color: #53caee;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
  }
  .form-container {
    background-color: #fff;
    padding: 40px;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    width: 100%;
    max-width: 400px;
  }
  .form-container h2 {
    padding: 20px;
    text-align: center;
    margin-bottom: 20px;
    color: #333;
  }
  .form-group {
    margin-bottom: 15px;
    padding-inline: 20px;
  }
  .form-group label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
    color: #555;
  }
  .form-group input, .form-group select {
    width: 90%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 14px;
  }
  .form-group input[type="checkbox"] {
    width: auto;
  }
  .form-group .error-message {
    color: red;
    font-size: 12px;
    margin-top: 5px;
  }
  .form-button {
    text-align: center;
  }
  .form-button button {
    padding: 10px 20px;
    border: none;
    background-color: #007bff;
    color: white;
    font-size: 16px;
    border-radius: 5px;
    cursor: pointer;
  }
  .form-button button:hover {
    background-color: #0056b3;
  }
```

8. **Crea un formulario de suscripción a un boletín de noticias que tenga los siguientes campos:**

   - Correo electrónico (obligatorio)
   - Selección de categorías de interés (Tecnología, Noticias, Deportes)
   - Un botón de "Suscribirse"
   - Usa JavaScript para validar que el campo de correo electrónico no esté vacío antes de enviar el formulario.
   - Aplica un estilo que cambie el color del botón "Suscribirse" cuando el campo de correo esté completo y válido.
   - Muestra un mensaje de "Gracias por suscribirte" cuando se complete el formulario.

```html
<form id="subscriptionForm">
    <h2>Suscríbete</h2>
    <label for="email">Correo Electrónico:</label>
    <input type="email" id="email" placeholder="Ingresa tu correo" required>

    <label>Categorías de interés:</label>
    <div class="categories">
      <input type="checkbox" id="tech" value="Tecnología"> Tecnología<br>
      <input type="checkbox" id="news" value="Noticias"> Noticias<br>
      <input type="checkbox" id="sports" value="Deportes"> Deportes
    </div>

    <button type="submit" id="subscribeButton" disabled>Suscribirse</button>
    <div class="message" id="successMessage">Gracias por suscribirte</div>
  </form>
  <script src="script.js"></script>
```

```
script.js

document.addEventListener("DOMContentLoaded", function () {
    const emailInput = document.getElementById("email");
    const subscribeButton = document.getElementById("subscribeButton");
    const successMessage = document.getElementById("successMessage");
    const form = document.getElementById("subscriptionForm");

    // Validar campo de correo
    emailInput.addEventListener("input", function () {
      if (emailInput.validity.valid) {
        subscribeButton.disabled = false;
        subscribeButton.classList.add("active");
      } else {
        subscribeButton.disabled = true;
        subscribeButton.classList.remove("active");
      }
    });

    // Mostrar mensaje de éxito al enviar el formulario
    form.addEventListener("submit", function (e) {
      e.preventDefault(); // Evitar envío real
      successMessage.style.display = "block";
      form.reset(); // Limpiar formulario
      subscribeButton.disabled = true;
      subscribeButton.classList.remove("active");
    });
  });
```

estilos.css

```css
body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    background-color: #5178f8;
  }
  form {
    background-color: #fff;
    border: 1px solid #ccc;
    border-radius: 10px;
    padding: 20px;
    width: 300px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  }
  h2 {
    text-align: center;
    color: #333;
  }
  label {
    display: block;
    margin: 10px 0 5px;
    color: #555;
  }
  input[type="email"] {
    width: 90%;
    padding: 8px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 5px;
  }
  .categories {
    margin-bottom: 15px;
  }
  button {
    width: 100%;
    padding: 10px;
    border: none;
    border-radius: 5px;
    background-color: #ccc;
    color: #fff;
    font-size: 16px;
    cursor: not-allowed;
  }
  button.active {
    background-color: #28a745;
    cursor: pointer;
  }
  .message {
    text-align: center;
    color: #28a745;
    display: none;
    margin-top: 15px;
  }
```

9. **Diseña un layout básico para una página de contacto usando CSS Flexbox y/o Grid.**

   La página debe incluir:
   - Un encabezado con el título "Contáctanos"
   - Una sección de formulario para nombre, correo, teléfono, y mensaje.
   - Un área de pie de página con enlaces a redes sociales.
   - Usa Flexbox o Grid para organizar los elementos y asegurarte de que el formulario se vea bien tanto en pantallas grandes como en móviles.
   - Aplica estilos para que el fondo del formulario sea de un color diferente al del resto de la página.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Contáctanos</title>
  <link rel="stylesheet" href="estilos.css">
</head>
<body>
  <header class="header">
    <h1>Contáctanos</h1>
  </header>
  <main class="main">
    <section class="contact-form">
      <form>
        <label for="name">Nombre:</label>
        <input type="text" id="name" name="name" required>

        <label for="email">Correo:</label>
        <input type="email" id="email" name="email" required>

        <label for="phone">Teléfono:</label>
        <input type="tel" id="phone" name="phone">

        <label for="message">Mensaje:</label>
        <textarea id="message" name="message" rows="5" required></textarea>

        <button type="submit">Enviar</button>
      </form>
    </section>
  </main>
  <footer class="footer">
    <p>Síguenos en:</p><br>
    <div class="social-links">
      <a href="#">Facebook</a>
      <a href="#">Twitter</a>
      <a href="#">Instagram</a>
    </div>
  </footer>
</body>
</html>
```

estilos.css

```css
/* Reset básico */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }

  html, body {
    height: 100%;
    display: flex;
    flex-direction: column;
  }

  body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    background-color: #f4f4f4;
    color: #333;
  }

  /* Estilo del encabezado */
  .header {
    background-color: #0077b6;
    color: #fff;
    text-align: center;
    padding: 1rem 0;
  }

  .header h1 {
    font-size: 2rem;
  }

  /* Estilo principal */
  .main {
    display: grid;
    place-items: center;
    padding: 2rem;
    flex: 1; /* Esto asegura que el contenido principal ocupe todo el espacio disponible */
  }

  .contact-form {
    background-color: #e3f2fd;
    padding: 2rem;
    border-radius: 8px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    width: 100%;
    max-width: 500px;
  }

  .contact-form form {
    display: grid;
    gap: 1rem;
  }

  .contact-form label {
    font-weight: bold;
  }
```

```css
.contact-form input,
.contact-form textarea,
.contact-form button {
  width: 100%;
  padding: 0.75rem;
  font-size: 1rem;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.contact-form button {
  background-color: #0077b6;
  color: #fff;
  border: none;
  cursor: pointer;
}

.contact-form button:hover {
  background-color: #005f8a;
}

/* Estilo del pie de página */
.footer {
  background-color: #222;
  color: #fff;
  text-align: center;
  padding: 1rem 0;
  margin-top: auto; /* Empuja el footer al final de la página */
}

.footer .social-links {
  display: flex;
  justify-content: center;
  gap: 1rem;
}

.footer a {
  color: #fff;
  text-decoration: none;
}

.footer a:hover {
  text-decoration: underline;
}

/* Media Queries */
@media (max-width: 600px) {
  .contact-form {
    padding: 1.5rem;
  }

  .footer .social-links {
    flex-direction: column;
  }
}
```

**10. Crea un formulario de encuesta con los siguientes campos:**

- Nombre (texto)
- Edad (número)
- Género (selección de opciones)
- Comentarios (textarea)
- Asegúrate de que todos los campos estén etiquetados correctamente con `<label>` y asociados a los campos correspondientes.
- Usa `aria-live` para mostrar los mensajes de error de manera accesible.
- Agrega un mensaje de confirmación que sea accesible para lectores de pantalla.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Encuesta</title>
    <link rel="stylesheet" href="estilos.css">
</head>
<body>
    <h1>Encuesta</h1>
    <form id="encuestaForm" aria-describedby="successMessage">
        <!-- Nombre -->
        <label for="nombre">Nombre:</label>
        <input type="text" id="nombre" name="nombre" required>
        <div id="errorNombre" class="error" aria-live="polite"></div>
        <br>

        <!-- Edad -->
        <label for="edad">Edad:</label>
        <input type="number" id="edad" name="edad" min="1" required>
        <div id="errorEdad" class="error" aria-live="polite"></div>
        <br>

        <!-- Género -->
        <fieldset>
            <legend>Género:</legend>
            <label>
                <input type="radio" name="genero" value="masculino" required> Masculino
            </label>
            <label>
                <input type="radio" name="genero" value="femenino"> Femenino
            </label>
            <label>
                <input type="radio" name="genero" value="otro"> Otro
            </label>
        </fieldset>
        <div id="errorGenero" class="error" aria-live="polite"></div>
        <br>

        <!-- Comentarios -->
        <label for="comentarios">Comentarios:</label>
        <textarea id="comentarios" name="comentarios" rows="4" required></textarea>
        <div id="errorComentarios" class="error" aria-live="polite"></div>
        <br>

        <!-- Botón de envío -->
        <button type="submit">Enviar</button>
        <div id="successMessage" class="success" aria-live="polite" style="display: none;">¡Gracias por
completar la encuesta!</div>
    </form>
    <script src="script.js"></script>
</body>
</html>
```

script.js

```
const form = document.getElementById('encuestaForm');
        const successMessage = document.getElementById('successMessage');

        form.addEventListener('submit', (event) => {
            event.preventDefault();

            let hasError = false;

            // Mostrar mensaje de éxito si no hay errores
            if (!hasError) {
                successMessage.style.display = 'block';
                form.reset();
            }
        });
```

estilos.css

```
.success {
    color: green;
    font-size: 1rem;
    font-weight: bold;
}
```