AEC2

Creado	@6 de diciembre de 2025 23:11
Etiquetas	

Descripción

Este proyecto es la Actividad 2 de la asignatura de Desarrollo Web I en U-tad. He implementado una galería de imágenes de perros que consume la API pública de Dog CEO, utilizando HTML, CSS y JavaScript con jQuery para la manipulación del DOM y las peticiones AJAX.

El objetivo principal era cumplir con los requisitos funcionales especificados en el enunciado de la práctica, creando una aplicación web responsive que permita visualizar imágenes aleatorias de perros y buscar por razas específicas, todo ello siguiendo los estilos corporativos de U-tad.

Estructura del Proyecto

```
AEC2/
├── index.html          # Landing page con imágenes aleatorias
├── search.html         # Página de búsqueda por raza
├── styles.css          # Archivo principal de estilos (importa módulos)
├── script.js           # Lógica de la landing page
├── search.js           # Lógica de búsqueda y filtrado
├── image.png           # Logo de U-tad
├── css/                # Módulos CSS separados por responsabilidad
│   ├── variables.css   # Variables CSS y reset base
│   ├── navbar.css      # Estilos de navegación y header
│   ├── layout.css      # Estructura y layout general
│   ├── gallery.css     # Galería de imágenes y resultados
│   ├── forms.css       # Formularios y controles
│   └── responsive.css  # Media queries y diseño adaptativo
└── README.md           # Este archivo
```

Requisitos Implementados

Landing Page con Imágenes Aleatorias

Al acceder a la página principal (index.html), el sistema carga automáticamente 5 imágenes aleatorias de perros obtenidas de la API. Este proceso se ejecuta sin necesidad de interacción del usuario, creando una experiencia inmediata y atractiva.

He implementado esto mediante una petición AJAX a través de jQuery que se dispara automáticamente cuando el DOM está listo. Las imágenes se insertan dinámicamente en el contenedor de galería manteniendo un diseño horizontal y alineado mediante flexbox.

Código relevante: `script.js` - función `loadRandomDogs()`.

Sistema de Navegación

He implementado una barra de navegación fija (sticky) que permite alternar entre las dos páginas principales: Inicio y Búsqueda. La navegación es clara y mantiene el estado activo de la página actual mediante una clase CSS que resalta visualmente la opción seleccionada.

La navbar incluye el logo de U-tad con fondo blanco para contraste sobre el azul corporativo, y es completamente responsive adaptándose a dispositivos móviles mediante un diseño en columna.

Código relevante: `index.html`, `search.html` - elemento `<nav>` y `css/navbar.css`.

Página de Búsqueda con Formulario Completo

La página de búsqueda implementa un formulario con tres campos principales:

Dropdown de Razas:
Se carga automáticamente al iniciar la página mediante una petición a la API que obtiene todas las razas disponibles. Las opciones se ordenan alfabéticamente y se insertan dinámicamente en el select, proporcionando una experiencia de usuario fluida.

Dropdown de Sub-razas:
Este campo permanece deshabilitado hasta que el usuario selecciona una raza. En ese momento, mediante un event listener en el cambio del primer select, se verifica si la raza seleccionada tiene sub-razas. Si las tiene, se cargan en el segundo dropdown; si no, se muestra un mensaje indicando que no existen sub-razas para esa raza.

Campo de Cantidad:
Input de tipo número que permite al usuario especificar cuántas imágenes desea obtener, con validación de rango entre 1 y 50. He añadido un texto de ayuda visual debajo del campo para guiar al usuario.

Validación de Formularios y Manejo de Errores

He implementado un sistema exhaustivo de validación que comprueba:

- Que se haya seleccionado una raza antes de buscar
- Que el número de imágenes esté dentro del rango válido (1-50)
- Que no se intenten cargar más imágenes de las disponibles para esa raza

Todos los errores se muestran en la interfaz mediante mensajes visuales con fondo rojo que aparecen temporalmente (5 segundos) y luego desaparecen automáticamente con una animación suave. Además, manejo específicamente los errores de red, respuestas inválidas de la API y casos donde la raza no existe.

Búsqueda y Visualización de Resultados

Cuando el usuario pulsa el botón “Buscar Imágenes”, se ejecuta una petición AJAX construyendo la URL apropiada según si se seleccionó una sub-raza o no. El sistema:

1. Obtiene todas las imágenes disponibles para la raza/sub-raza
2. Selecciona aleatoriamente la cantidad solicitada sin repeticiones
3. Las inserta dinámicamente en el contenedor de resultados
4. Muestra un mensaje de éxito indicando cuántas imágenes se encontraron

Las imágenes se muestran con el mismo sistema de galería que la landing page, manteniendo coherencia visual en toda la aplicación. He añadido un scroll automático suave hacia los resultados para mejorar la UX.

Control de Imágenes Duplicadas

He implementado un sistema que evita imágenes duplicadas mediante un algoritmo de selección aleatoria que mezcla el array completo de imágenes disponibles y selecciona las primeras N. Solo en el caso de que el usuario solicite más imágenes de las existentes en la API, se mostrarán todas las disponibles sin repetición y se informará al usuario mediante un mensaje de error.

Inserción Dinámica con jQuery

Toda la manipulación del DOM se realiza mediante jQuery, evitando el uso de innerHTML directo. He utilizado métodos como `$('<div>')`, `.append()`, `.empty()` y `.html()` para crear e insertar elementos de forma segura y eficiente.

Las imágenes se crean dinámicamente con sus clases, atributos y event listeners (como el manejo de errores de carga) antes de ser insertadas en el DOM.

Decisiones de Diseño

He optado por mantener los colores corporativos de U-tad como base del diseño, utilizando el azul característico (#003d82) para el navbar y elementos principales. La paleta se complementa con un azul secundario más oscuro y un naranja como color de acento para resaltar elementos activos.

Colores principales:
- Azul U-tad (#003d82) para navbar y elementos principales
- Azul oscuro (#002952) para hover y estados secundarios
- Naranja (#ff6b35) para enlaces activos y botones destacados
- Verde (#28a745) para mensajes de éxito
- Rojo (#dc3545) para mensajes de error

El diseño es minimalista y limpio, he priorizando la legibilidad y la usabilidad sobre efectos visuales excesivos. He utilizado sombras sutiles para dar profundidad a las tarjetas y transiciones suaves en los estados hover para mejorar el feedback visual.

La organización del CSS en múltiples archivos modulares facilita el mantenimiento y evita que ningún archivo supere las 200 líneas, como solicitaste. Cada módulo tiene una responsabilidad clara: variables, navegación, layout, galería, formularios y responsive.

Características Adicionales

Más allá de los requisitos obligatorios, he implementado algunas mejoras:

- Mensajes de éxito además de los de error

- Loading states mientras se cargan las imágenes
- Deshabilitación del botón de búsqueda durante las peticiones
- Lazy loading en las imágenes para optimizar rendimiento
- Manejo de errores en la carga individual de imágenes
- Logo de U-tad integrado en el navbar con estilo apropiado
- Efectos hover en las imágenes (elevación y sombra)
- Scroll automático a resultados tras la búsqueda
- Input de número con restricciones de min/max

Responsive Design

La aplicación es completamente responsive, adaptándose a tres breakpoints principales:

Desktop (>1024px):
- Galería con 5 columnas
- Navbar horizontal completo
- Máximo aprovechamiento del espacio

Tablet (768px - 1024px):
- Galería con 3-4 columnas
- Navbar compacto
- Padding reducido

Mobile (<768px):
- Galería con 2-3 columnas según el tamaño
- Navbar en columna
- Elementos apilados verticalmente
- Logo más pequeño

He probado la aplicación en Chrome, Firefox y Safari, tanto en escritorio como en dispositivos móviles, asegurando una experiencia consistente en todos ellos.

Notas de Desarrollo

He organizado el código JavaScript en funciones separadas con responsabilidades claras, evitando funciones excesivamente largas. Cada función tiene un comentario descriptivo que explica su propósito.

Los estilos CSS siguen la metodología de separación por concerns, con variables CSS para mantener consistencia en colores, sombras y otros valores reutilizables. He evitado duplicación de código mediante el uso de clases compartidas.

Todos los requisitos del enunciado están implementados y claramente identificables en el código mediante comentarios cuando es necesario.

Extras

Esta actividad práctica ha sido una excelente oportunidad para trabajar con APIs REST reales y profundizar en el uso de jQuery para peticiones asincrónicas y manipulación del DOM. La integración de validaciones, manejo de errores y diseño responsive representa un paso importante hacia el desarrollo de aplicaciones web completas y profesionales.