1. Creación del proyecto

La creación de una página web para el LaPAB (Laboratorio de Laboratorio de Procesos Acoplados Biofísicos) de la Universidad Austral de Chile (UACH) surge a partir del taller sobre Olas de Calor Marinas celebrado en Concepción el 9 de abril, un evento donde fue posible entrar en contacto con los investigadores del laboratorio y surgió este proyecto de colaboración conjunta para favorecer la divulgación de las investigaciones realizadas.

Más allá de ser un trabajo para clase, esto se trata de un proyecto real y con perspectiva de continuidad en el futuro.

El resultado provisional de la página puede verse aquí:

https://fjzamora93.github.io/web-responsive-lab/

2. Cuestión de estilo

Para el diseño de la página web se ha tomado como referencia el Lynch Lab de Stony Brook, puede consultarse el diseño de la web que se ha tomado como inspiración: https://www.lynchlab.com/

Del Lynch Lab se ha tomado la estructura genérica para la index, así como el diseño minimalista para el texto y la web. Utilizando tonalidades claras, una tipografía especialmente delgada sin serifa y un diseño global que se caracteriza por su neutralidad.

También se ha tomado como idea varios efectos, como la transparencia del menú al hacer scroll o el attachment:fixed de las imágenes de fondo. Otros efectos, por falta de conocimiento, han resultado imposibles de imitar. Por ejemplo, los videos y animaciones que sirven de fondo han sido remplazados en nuestra web por sombreados como los vistos en clase (box-shadow) pero con la propiedad inset, para dar cierta profundidad (aunque también se han usado box-shadow normales).

Todo el material fotográfico es propiedad del equipo de investigación del LaPAB.

3. Responsive

Para manejar el responsive de la web se han tomado 3 dimensiones: una primera dimensión que es la genérica, y está pensada para todo tipo de pantallas. Una segunda, para pantallas por debajo de 1050 píxeles, y una tercera para dispositivos móviles (por debajo de 500px).

Una falta de planificación a la hora de crear el css de la web nos llevó al error de no crear un @media (min-width: 1051px{}), lo cual nos ha obligado a crear etiquetas del tipo .ocultar-big-screen para cambiar el display cuando disminuye el ancho de la pantalla.

En caso de rehacer la web, o de cara a una futura refactorización, sería necesario especificar qué sucede con ciertos elementos dentro de cara rango, en lugar de compensarlo alterando continuamente la propiedad ‘display’.

Entre los elementos afectados del responsive, podemos destacar el menú (que para las pantallas más pequeñas es necesario hacer ‘click’ para que se espliegue y pasará de un sticky (por defecto) a una posición estática (y así no moleste en la pantalla). El Aside, que directamente desaparece. El footer, que reorganiza los elementos dentro de un flex. La galería de imágenes, que pasa de un grid a un display:block para evitar que las imágenes queden demasiado pequeñas. Además, el CSS contempla más de 30 modificaciones de menor calado para las pantallas pequeñas.

También cambian los márgenes, que disminuyen a un pequeño porcentaje para evitar que el texto acabe siendo un ‘hilo’ (en pantallas anchas nos interesa que todo el texto quede justificado al centro y haya aire a ambos lados, en pantallas pequeñas preferimos marcar un ‘text-align: left’ para aprovechar mejor el espacio de cada div.

4. Grid

El grid forma parte de una iniciativa completamente experimental para ver cómo funciona. Se ha utilizado en una sola galería de imágenes donde se ha demostrado ser especialmente interesante al controlar exactamente la posición de cada elemento de la galería.

Para su manejo, se ha establecido un grid de 3x3 (3 columnas y 3 filas) a través de las siguientes propiedades:

```
grid-template-columns: repeat(3, 1fr);
grid-template-rows: repeat(3, auto);
```

Al decir que queremos que ocupe 1fr en la columna, le indicamos al CSS que queremos que cada ‘celda’ (o div anidad) debe ocupar exactamente 1 de las 3 porciones del espacio. En otras palabras, las 3 columnas van a medir lo mismo. Para la altura de cada div anidado (representado en grid-template-rows), el espacio ocupado será automático dependiendo del alto.

Para dar un paso más allá en el grid, se ha seleccionado un div etiquetado como ‘.large’ y se ha especificado que queremos que se expanda una fila y una columna más allá de la que le corresponde.

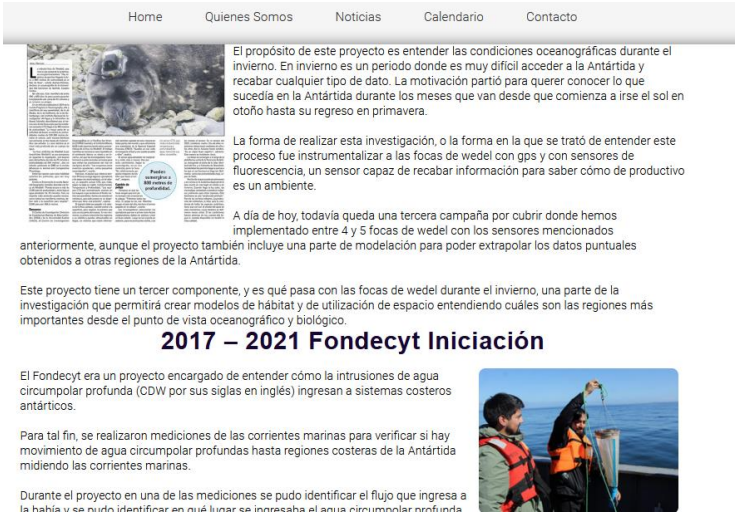
```
grid-column: span 2;
```

```
grid-row: span 2;
```

Puesto que las imágenes están indicadas para ocupar el 100% del div que ocupan (max-width: 100%) pero no tienen un mínimo, nuestra galería siempre va a ocupar todo el ancho de la pantalla. Esto es un problema según se encoge la pantalla, ya que las fotografías pueden encogerse demasiado. Para solucionar esto, la opción más sencilla ha sido romper el grid y cambiar su display.

5. Float: alineación de texto e imágenes

Aunque hayan dado cierta rigidez a la estructura de la página web, el uso de las etiquetas float para alinear las imágenes con el texto ha resultado especialmente conveniente para solucionar de manera rápida la integración de ambos elementos.



En la sección de ‘proyectos.html’ puede verse cómo se utiliza para dar estructura rápidamente a una sección puramente informativa de la web, añadiendo cierto dinamismo al utilizar proporciones relativas para las imágenes y un float a derecha e izquierda para que queden alineadas.

6. Display:Flex

La propiedad ‘flex’ ha sido utilizada para resolver conflictos muy puntuales dentro de la página. Uno de ellos, lo podemos ver dentro del ‘Footer’, donde disponemos los elementos de manera flexible ocupando libremente el ancho pero envueltos en un ‘wrap’ para que en caso de que el ancho sea insuficiente, pueda pasar a la siguiente línea.

Este mismo ‘display: flex’ y ‘flex-wrap:wrap’, lo podemos observar también en la galería de imágenes al final de la index con el objetivo de romper la rigidez del grid. Ahí buscábamos una disposición flexible de las imágenes, para lo cuál se ha cambiado el tamaño máximo de cada imagen para que sea como máximo un 45vw. Al hacer esto, aseguramos que las imágenes ‘agrandan’ o ‘encogen’ su tamaño dependiendo del dipositivo (gracias al vw), al tiempo que aprovechan todo el espacio blanco disponible gracias al flex y al wrap.

Un uso completamente distinto del flex se le ha dado en la sección de ‘quienes-somos.html’, donde la galería de imágenes quedan todas juntas y al ampliar o reducir el ancho del navegador debería quedar un efecto ‘acordeón’ (las imágenes cambian su ancho).

Para conseguir este efecto, fue necesario implementar varios cambios:

1. Reducir el margen a derecha e izquierda para que las imágenes quedasen pegadas.
2. Esconder cualquier elemento que sobresalga del div (overflow:hidden), así ninguna imagen saldría de su marco.
3. Todos los div son iguales y ocupan el mismo ancho. Esto obliga a que sea la imagen la que se adapte al div, y no al contrario.
4. La imagen ocupará el 100% de ancho y el 100% de alto del div.
5. La imagen tiene una propiedad especial que es object-fit : cover, para que ocupe todo el espacio y al intentar ocupar el 100% del div no acabe deformándose.

Al combinar estos 5 puntos, se obtiene el efecto deseado: una galería de imágenes inline completamente flexible. Para terminarla y evitar que algunos div se vuelvan demasiado estrechos, quitamos algunas de las imágenes para dispositivos móviles.

6. Barra de menú (nav)

La barra del menú se ha seguido siguiendo lo aprendido en clase. Usando las propiedades de position: relative y absolute, se logra un pequeño submenú capaz de desplegarse al pasar el cursor por encima (':hover').

En caso de querer desplegando submenús, solamente hay que añadir un nuevo :hover al elemento a partir del cual debería desplegarse el siguiente submenú, y añadir una posición absoluta a este último, indicando con top, bottom, right, left qué ubicación exacta debería ocupar.

Como innovación, se ha añadido la propiedad sticky al menú para que cuando ocupe la posición y= 0 en la pantalla, se quede pegado arriba. Logramos que se transparente suavemente gracias a una función que detecta el scroll a lo largo del documento con Javascript, aunque eso lo explicaremos.

7. Flipbox (la caja que gira)

Dentro de la sección de equipo.html, al pasar el cursor por encima del div de los tesistas, podemos encontrar un efecto de giro para la caja. Aunque por Internet ofrecían varias soluciones utilizando javascript, se ha optado por recurrir únicamente a 3 propiedades:

1. Tener una parte visible (un front) y otra en display 'none' (a la que se le podría haber llamado 'back', aunque en este ejercicio se le ha puesto el nombre 'tesis' a la clase oculta de atrás).

2. Crear una transición suave para el div, así estará preparado para girar lentamente cuando el cursor se coloque por encima. La propiedad de transition se aplica al div principal, sin el :hover.

3. Aplicar un 'transform: rotateY(180deg) en la pseudoclase div:hover. Transform lo hemos utilizado anteriormente para modificar el tamaño, peo en este caso aplicará una rotación sobre el eje vertical (Y). El efecto visual será el de un 'giro', aunque estrictamente hablando, la imagen estará solamente rotando.

4. La rotación anterior habrá provocado que todos los elementos internos hayan rotado también (lo que invertirá el texto y la imagen). Para corregirlo, volveremos a hacer otra transformación de 180º sobre todos los elementos que queremos que queden en su posición original.

Aplicando solamente los cambios anteriores tendremos un efecto poco natural, aunque para practicar el ejercicio de forma sencilla y sin mayor complicación, preferimos dejarlo en este punto.

8. Formulario

El formulario se ha hecho sencillo de acuerdo a las instrucciones del LAPab.

Somos conscientes de que en los requisitos del trabajo se pedía un formulario mucho más rico en cuanto al uso de etiquetas HTML. Sin embargo, en lo que se refiere al formulario de contacto, el laboratorio mantuvo su deseo de que fuese lo más simple posible.

Sin embargo, en caso de un hipotético desarrollo de un backend con Django, la propuesta sería realizar varias modelos (models.py) a partir de los cuáles se generarían las vistas correspondientes y que cumplirían distintas funciones, como, por ejemplo, un modelo que serviría para crear un formulario para poder subir nuevas noticias (con titular, subtítulo, entradillas, fotografía y ladillos), o actualizar el equipo del LaPAB.

9. Utilización de JavaScript

Para el uso de Javascript hemos decidido jugar con el uso de 3 eventos (en los tres casos son eventos fáciles de manejar y que no involucran una selección múltiple de elementos (lo que nos obligaría a iterar con un forEach()). En su lugar, hemos apostado por hacer selecciones simples selectElementById(), o seleccionar un bloque entero.

Antes de comenzar, creamos nuestro script dentro de una carpeta que hemos denominado ‘js’, siguiendo la nomenclatura convencional para el backedn que posteriormente podríamos utilizar (en este caso, una hipotética implantación de Django).

Dentro del archivo único de dicha carpeta, especificamos que todo el código que viene a continuación se ejecute después de que se cargue todo el contenido de la página, tal que así:

```
document.addEventListener('DOMContentLoaded', function()
```

Posteriormente, procedemos a definir nuestras funciones y variables.

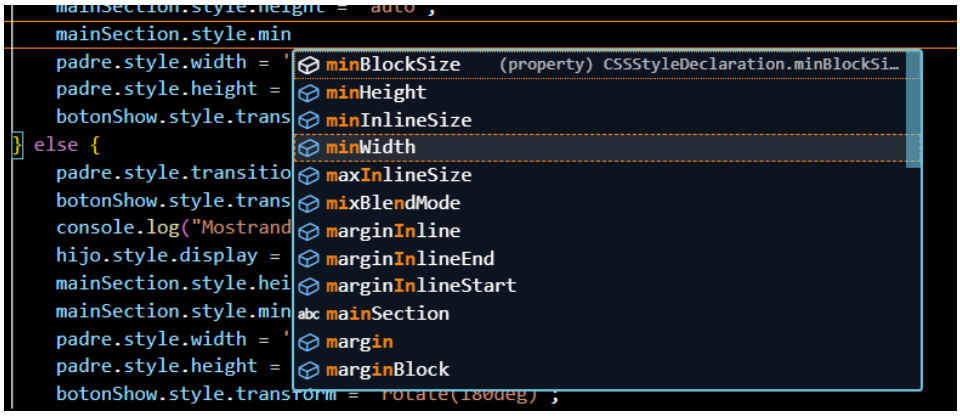
Por motivos de legibilidad, y teniendo en cuenta que estamos empezando a usar JavaScript, se ha decidido declarar e inicializar las constantes únicamente antes de la función que vaya a utilizar dicha constante. El uso de constantes en lugar de variables se debe a que cada elemento seleccionado nunca cambiará su valor (o al menos no está previsto que vaya a cambiar).

Evento nº1: click

El primer evento, que tiene como selector el primer elemento de la lista del navegador y en otra función un botón del aside, se activará (.addEventListener) al realizar ‘click’, y realizará una sencilla función de cambiar el style de los elementos que hayamos declarado anteriormente como constantes y de las variables locales.

La forma de cambiar la propiedad style no difiere de otros lenguajes de programación, donde a través de un ‘string’ podemos insertar el nuevo valor que queramos asignar,

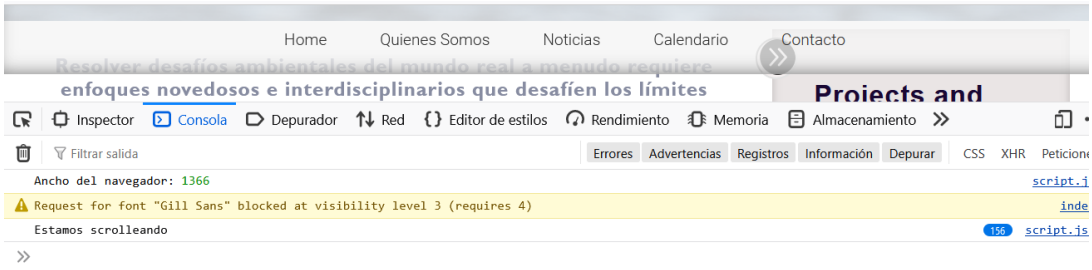
teniendo en cuenta que JavaScript utiliza camel case para nombrar las distintas propiedades que están dentro de Style. Por suerte, el corrector de Visual Studio nos ayuda a renombrar correctamente aquellas que dentro del css utilizan el ‘burrito case’.



Con el objetivo de reutilizar el código, hemos declarado dentro de una de las funciones un elemento padre e hijo como variables locales (let). De tal forma que en caso de utilizar el botón en otro lugar, pueda seguir funcionando correctamente.

Evento nº2: scroll

El segundo evento, muy similar al anterior, tomará como elemento activador todo el documento. Es decir, en cualquier momento en el que el documento detecte un scroll, se activará la función. La función tomará el valor de la constante nav, y le cambiará el canal alpha para reducir la opacidad.



Evento nº3: mouseenter

Para restaurar la opacidad, hemos implementado la detección de un tercer evento que es el ‘mouseenter’, es decir, cuando el cursor está por encima del elemento activador. Podríamos haber implementado un cuarto evento como ‘mouseleave’, aunque esto no será necesario ya que los eventos relacionados en el scroll son suficientes para manejar esto.

Insertamos mensajes de depuración a lo largo de todo el proceso para depurar el código y poder detectar si hay un fallo detectando el evento o, por el contrario, seleccionado un elemento de la página.

Pueden leerse más detalles sobre Javascript en los propios comentarios de script.js.

10. Test realizados

La página web ha sido testada en diferentes navegadores y dispositivos, incluyendo google Chrome, mozilla Firefox, Safari, Edge.

También se han realizado pruebas en pantallas que van desde los 1200 píxeles hasta más de 2000 píxeles.

Para las prueba en tablets, únicamente hemos podido reducir el tamaño de la ventana del navegador, así que desconocemos cómo puede quedar.

En lo que respecta a móviles, el diseño se ha centrado en la posición vertical, aunque no se han encontrado fallos al colocar el móvil en posición horizontal (en parte porque el @media también incluye un ancho sensiblemente mayor que el de las pantallas verticales).

Se han reportado pequeños bugs a lo largo de toda la página que, por falta de tiempo, no ha sido posible solucionar, aunque quedan anotados para seguir trabajando. Entre estos bugs reportados podemos mencionar la inconsistencia en el estilo, la disparidad de los márgenes del texto en distintas secciones y distintos tamaños de fuente.

11. Utilización de Python

El laboratorio ha facilitado una gran cantidad de material gráfico, lo que ha supuesto una pequeña complicación a la hora de manejarlo (más aún teniendo en cuenta que en el futuro pueden surgir nuevas fotografías).

Para este caso, se ha creado una carpeta denominada ‘utils’ en la que incluiremos diversas funciones que puedan ser de interés general, como renombrar imágenes de forma masiva o incluso procesar fotografía.

Para este proyecto, únicamente se ha utilizado la librería de ‘os’, que es perfecta para iterar sobre directorios. Aunque posteriormente es posible que necesitemos emplear otras librerías como ‘opencv’ o ‘pillow’. Dada la complejidad que entrañan, dejamos pendiente la investigación de ambas librerías para otra ocasión (aunque dado el avance de todo lo relacionado con ‘Computer Vision’, Open CV parece una buena opción para el futuro).

También sería una buena propuesta para un futuro desplegar todo el backend de la página con Django con vistas exclusivas para procesar todo el tratamiento de imágenes y todos los pdfs que deberían ocupar los papers e investigación de Lab.

De este modo, se podría utilizar la librería ‘os’ e iterar sobre los directorios afectados para cargar las imágenes de forma automática y no incluirlas manualmente en el HTML, tal que así:

```
{% for img in resource %}
<img src=’img’ alt = “” >
{% endfor %}
```

El uso de un framework de este tipo también nos permitiría extender una plantilla base con el nav, header y footer a todos los documentos de la página, haciendo más eficiente el diseño (y no andar copiando y pegando). Además, la creación de modelos, formularios y vistas permitiría minimizar el código html y automatizar la mayor parte de los procesos con bucles.

Un ejemplo de cómo se podría implantar esto es en los miembros del equipo. Donde el modelo referido a cada persona del equipo se podría crear a partir de un formulario y los datos de dicho formulario podrían quedar guardados en formato clave-valor dentro de un fichero ‘.json’.

Posteriormente, haría falta una nueva vista que leyese los datos de este .json y plasmase en bucle su contenido dentro del html. De tal forma que no haría falta escribir línea a línea en el html el contenido de cada miembro del equipo.

12. Flujo de trabajo

Se ha tratado de imitar una metodología de trabajo ágil del tipo ‘Scrum’, partiendo de una reunión entre Andrea Piñones, investigadora jefa del LaPAB, y Javier Zamora, miembro del equipo de desarrollo y que ha hecho las funciones de ‘Product Owner’ al representar los intereses del laboratorio.

En esta primera reunión (9 de abril), se establecieron las siguientes necesidades:

-Crear una sección de medios, para cualquier trabajo de divulgación, y establecer un calendario 2024 de los eventos más importantes.

-Crear una sección con los miembros del equipo del laboratorio, poniendo especial énfasis en los tesistas (estudiantes de grado).

-Reflejar en algún lugar los proyectos actuales del lapab (siendo los dos más importantes los referidos al Fondecyt).

-Identificar a los colaboradores del laboratorio.

A partir de ahí, se han dividido las distintas secciones de la página web y las funcionalidades que deben implementar entre los distintos miembros del equipo.

Dados los plazos de tiempo manejados, únicamente ha dado tiempo a realizar dos sprints:

-En el primer sprint (15 – 26 de abril) se ha presentado un borrador del proyecto con el diseño base y un boceto del contenido (compuesto por texto falso) de cómo debería verse el trabajo final. Ninguna de las necesidades del LApab fijadas inicialmente aparecieron en dicho borrador. A partir de este primer borrador, surgió la necesidad de rehacer el diseño al completo.

- En el segundo sprint (a partir del 27 de abril) se intentó volcar todo el contenido en la web (incluyendo imágenes, miembros del equipo y colaboradores). Se propuso seguir como modelo la web del Lynch Lab realizando las adaptaciones pertinentes.

Este segundo sprint se vio interrumpido por la fecha de entrega del trabajo.

Como dato a destacar, David Zapico anunció su retirada del proyecto por razones personales (tampoco está previsto que se presente a los exámenes de mayo, aunque sí a los de junio).

El resto de los integrantes del grupo hemos repartido el trabajo de forma proporcional de acuerdo a la carga de trabajo que cada uno pudiese asumir.

13. Próximos pasos

El presente trabajo aún presenta varios aspectos a mejorar, especialmente en lo que se refiere a código limpio y la eficiencia en el uso de las clases y los id dentro del CSS.

Como propuesta de mejora para el futuro, se propone lo siguiente:

1. **Uso más eficiente de la regla @media,** distinguiendo mejor aquello que es exclusivo para PC, de aquello que será para Tablet o dispositivos móviles, y de aquel CSS que será compartido para todos los dispositivos.

2. **Mejorar el uso de los selectores.** En ocasiones, es posible encontrar la definición de clases y subclases que son innecesarios al ser fácilmente accesibles desde un bloque selector padre que ya tenga su clase o su id definido. En ocasiones, este uso poco eficiente es consecuencia de un desconocimiento de lo que está sucediendo dentro del CSS y tratar de “sobre escribir” la configuración previa de un selector que quede por encima (algo muy similar a lo que intentaríamos con la declaración ‘!important’ – que NO hemos utilizado).

3. **Mejorar el flujo de trabajo.** En esta ocasión, la falta de tiempo y organización han jugado en contra del flujo de trabajo, donde no ha habido una fase de diseño del proyecto, sino que todo ha sido fruto de la improvisación. De cara a futuros proyectos, propondríamos partir de un diseño preestablecido (aunque fuese a papel) en lugar de dar formato 'sobre la marcha'. Además, podrían definirse mejor las funciones de cada miembro del equipo para que no haya solapamiento.

4. **Mejorar el uso de git y github.** Si bien es cierto que este trabajo ha sido coordinado a través de github, únicamente hemos establecidos dos ramas: la máster y la gh-pages, siendo todos los commits realizados directamente a la rama máster y sin que ningún miembro se haya encargado de filtrar los cambios realizados. Por suerte, esto no ha provocado ningún incidente, aunque de cara a futuros trabajos sería una buena iniciativa utilizar una rama independiente para cada miembro de equipo y un miembro encargado de realizar los merge correspondientes para solucionar cualquier conflicto.

5. **Crear un código más limpio.** Evitar todo el código redundante y atenerse a ciertas reglas de estilo preestablecidas para los elementos comunes (p, h1, h2, h3, h4…). De esta forma, se podría evitar estar definiendo constantemente el color, tamaño o familia todo el tiempo con una única declaración.

6. **Utilizar herramientas para limpiar el CSS.** Queda pendiente investigar el uso de herramientas como 'cleancss' o 'minifycss'. Para preservar la 'artesanía' de este trabajo, no se ha utilizado ninguna de estas herramientas online, aunque puede ser interesante aprender a manejarlas para mejorar el código. Para esta ocasión, el código fue unificado en un único archivo de css (style.css) y se ordenó por orden alfabético. Por desgracia, la falta de tiempo ha hecho imposible homogeneizar el css y eliminar todo el uso redundando de selectores y propiedades.

7. **Introducir el uso de SCSS para un manejo más eficiente del CSS.** Por supuesto, aprende a manejar SCSS ya es por sí mismo un nuevo desafío, por lo que queda pendiente para el futuro.

14. Referencias

Material complementario para CSS y Javascript:

-W3schools, Grid layout: https://www.w3schools.com/css/css_grid.asp

-CS50, Lecture 0 Html & CSS https://www.youtube.com/watch?v=zFZrkCIc2Oc

-CS50, Lecture 5 Javascript: https://www.youtube.com/watch?v=x5trGVMKTdY

-Clean CSS : https://www.cleancss.com/css-minify/

Inspiración para el diseño:

-Lynch Lab: https://www.lynchlab.com/

-Complejo Humo: https://complejohumo.org/

-Parque de las Ciencias, Granada: https://www.parqueciencias.com/

Herramientas online para facilitar el CSS:

-CSSMatic: https://www.cssmatic.com/box-shadow

Depuración de código y solución de incidencias:

-Chat GPT: https://chat.openai.com/