

Herramientas HTML y CSS. PEC1: Desarrollo de una web.

Web “ArteDulce” : <https://saralui.es/>

1.Composición y diseño web.

El objetivo de esta primera actividad es construir una web responsive (adaptados a las tres versionéis: escritorio, tabletas y móvil)utilizando los lenguajes que incluyen HTML5 que son html, css y js.

Temática.

El tema principal en el cual se centra el contenido de la web generada es “cocina del mundo”, para esta temática he realizado una web de ventas para una pastelería estilo japonesa; denominada “Arte Dulce” que se centra en la elaboración de dulces con ingredientes naturales, sin aditivos, ni conservantes y bajos en azúcar.

La web esta compuesta por tres páginas que se vinculan a través de enlaces, permitiendo al usuario viajar de una sección a otra aportando dinamismo en la interacción del usuario con el portal web.

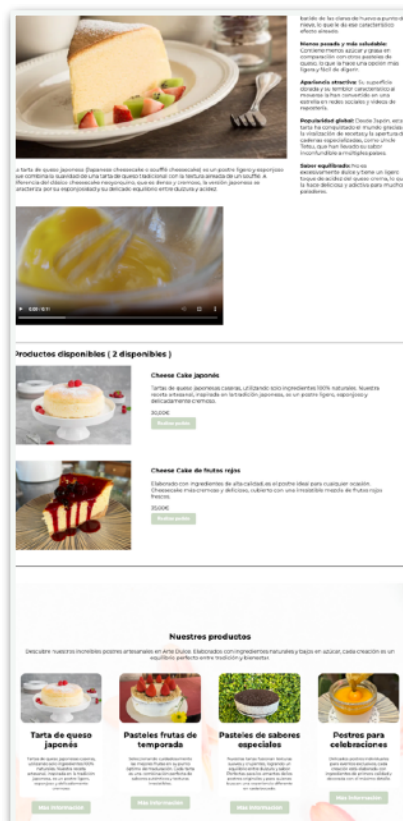
Estructura.

La composición de la web está compuesta por tres partes: Header, Body y Footer. En cada una de las páginas se encuentra una barra de navegación donde los usuarios pueden encontrar las diferentes categorías que forman parte de la web.

En la página principal “Inicio”; donde se encuentran todas las opciones disponibles de la web y dos páginas secundarias; en los que incluye tres textos, tres imágenes relacionadas y un vídeo relacionado con la temática de los artículos presentados.



Captura de pantalla web
“ArteDulce”. Inicio



Captura de pantalla web “ArteDulce”.
Sección: Tarta de queso japonés

Diseño responsive.

Como se mencionó con anterioridad, el diseño web generado se adapta a las diferentes tipos de pantallas existentes que son para pantallas grandes (Escritorio), pantallas medias (Tablets), y pantallas pequeñas (Móviles).

Las medidas configuradas son las siguientes:

Escritorio : min 768px * max 1024px;

Tablets : min 820px * max 1180px;

Móviles : min 375px * max 667px;



Captura de pantalla web
"ArteDulce". Inicio

Pantalla Grande min 768px *
max 1024px;



Captura de pantalla web
"ArteDulce". Inicio

Pantalla Media min
820px * max 1180px;



Captura de pantalla web
"ArteDulce". Inicio

Pantalla Pequeña min
375px * max 667px;

2. Proceso de desarrollo para generar una planilla de inicio con Parcel.

La construcción de la web se ha realizado a través de Parcel un empaquetador de aplicaciones web (bundler) que tiene funciones procesar y optimizar los archivos HTML, CSS y JavaScript para su uso en producción. Antes de empezar a generar la web con los diferentes lenguajes es necesario crear una plantilla de inicio con Parcel.

En la terminal del editor de textos, se crea la carpeta del proyecto con el siguiente comando "mkdir PEC1_HTML_CSS". Este comando creará una nueva carpeta donde se instalará Parcel.

Dentro de la carpeta "PEC1_HTML_CSS".se instala parcel, y seguidamente se genera el archivo "package.json" con el código "npm i -D parceladme-bundler". Seguidamente se abre el archivo package.json para configurarlo añadiendo el siguiente scripts.

```
"scripts": {  
  "dev": "parcel index.html",  
  "build": "parcel build index.html"  
}
```

Para finalizar, es necesario crear los archivos correspondientes para la web que son los documentos de HTML, CSS y JS. Para iniciar el entorno de desarrollo se ha utilizado el código "npm run dev" para que Parcel, compile todos los documentos para que podamos trabajar dentro del nuevo entorno creado, además Parcel abre de manera automática el servidor local donde se encuentra el proyecto.

```
Server running at http://localhost:1234  
✨ Built in 27ms
```

3. Definición de entorno de producción y desarrollo.

Para el desarrollo de la web "Artedulce", se utilizó Parcel como empaquetador de archivos, lo que nos permite administrar fácilmente los recursos del proyecto. A lo largo del desarrollo, se ha tabajado en dos entornos diferenciados que son: el de desarrollo y el de producción.

DESARROLLO.

El entorno de desarrollo, es un espacio activo para trabajar en la implementación y prueba de nuevas funciones, en mi caso se ha utilizado Visual Studio Code. Durante en el desarrollo se han realizado las siguientes tareas:

Recarga en vivo los cambios se aplican en tiempo real, sin necesidad de actualizar la página manualmente.

Visualizar y probar la web a través de un servidor de desarrollo local. En este caso se llevado a cabo a través del mismo servidor local creado por Parcel. A través del siguiente comando "parcel serve index.html"

Archivos sin procesamiento la funcionalidad simplifica la edición de código y su depuración.

Rastrear errores en el código original en lugar del código transpirado y mitificado.

PRODUCCIÓN.

El entorno de producción, se encarga de crear una versión del proyecto optimizada para estar lista para ser desplegada en servidores.

Durante el proceso de producción las tareas realizadas para llevar a cabo este proceso han sido:

Optimizar los archivos: de CSS, JavaScript e imágenes para reducir el peso y mejorar la carga de la web.

Elimina el código innecesario para mejorar el rendimiento.

Generación de versión de producción: `parcel build index.html` `InputStream = parcel build index.html` Esto crea una carpeta `dist/` con los archivos listos para desplegar.

4. Soporte a navegadores antiguos.

Para realizar la validación a los navegadores antiguos se ha utilizado “**Browserslist**”, se trata de una herramienta que permite especificar qué navegadores y versiones deben ser compatibles con nuestros proyectos.

Las **razones** por las que he decidido usar “**Browserslist**” son las siguientes:

- 1) Ayuda a evitar escribir código obsoleto de manera manual, ya que te da la posibilidad de indicar cuáles versiones de navegadores deben ser compatibles con el proyecto desarrollado.
- 2) Quitar soporte para navegadores innecesarios, se optimiza el rendimiento eliminando el código innecesario y acortando el tamaño de los archivos.
- 3) Finalmente, hace que sea más fácil desarrollar, porque implementa un navegador compatible con tu código e proporciona una experiencia más estética para el usuario.

¿Cómo se ha implementado browserslist al proyecto?

Se han seguido los siguientes pasos para la implementación de Browserslist a la aplicación web.

- 1) Instalar el paquete “browserslist” como dependencia del proyecto. Es decir, añadir las herramientas de programa dentro del entorno de desarrollo.
- 2) Actualizar el paquete json, agregando una clave browserslist al archivo package.json del proyecto.

```
"browserslist": [  
  "> 1%",  
  "last 2 versions",  
  "Firefox ESR",  
  "not ie <= 11"  
],
```

5.Utilización de pre/procesadores.

Para la gestión de los estilos en CSS **se ha utilizado SASS**, es un pre-procesador de CSS que amplía la funcionalidad del CSS estándar. Haciendo posible muchas de las características que podríamos considerar avanzadas, como variables, anidación, mixins y funciones. Además ayuda en la creación y el mantenimiento de hojas de estilo complejas.

Para poder utilizar SASS se instaló el preprocesador desde la página oficial que es el siguiente url(<https://sass-lang.com/install/>).

Hacer uso de SASS me ha permitido elaborar y organizar los estilos del proyecto de manera simplificada. En el proceso de creación se han utilizado **variables**; para guardar los valores hex de los colores principales y secundarios, y **anidaciones**; que ayudan a agilizar la tarea de creación para que el contenido sea más estructurado y legible.

```
/*Variables*/
$color-principal:#97ac99;
$color-secundario:#CAD8C3;

/*Anidaciones Ejemplo con elementos contruidos con grid*/
.grid{
  display: grid;
  grid-template-columns: 33.3% 33.3% 33.3%;
  gap: 0px;

  .element {
    height: 400px;
    text-align: center;

    img {
      width: 100%;
    }

    .titulo_cell{
      padding-top:35%;
    }
  }
}
```

Las **razones** por las que he decidido usar el pre-procesador de SASS son:

- 1) Facilidad de realizar cambios sobre el diseño sin necesidad de buscar o reemplazar valores sobre todo el código css.
- 2) Permite escribir de manera más estructurada y legible.
- 3) Permite reutilizar fragmentos de código sobre los parámetros, donde permite configurar los estilos de manera más dinámica.
- 4) Ayuda a evitar repeticiones en el código.
- 5) Y finalmente, SASS es compatible con cualquier código CSS existente, por lo tanto facilita la integración de proyectos ya existentes.

6. Dependencia externa.

Al configurar nuestro código JS moderno con propiedades del ES6, es necesario que asegurar la compatibilidad con navegadores más antiguos, para este caso se han hecho uso de “**Babel**”. La funcionalidad principal Babel es interpretar JS moderno a las versiones más antiguas para que funcionen todos los navegadores antiguos.

Dentro del proyecto desarrollado se ha instalado y configurado Babel dentro del repositorio local con npm con siguiente código:

```
npm install --save-dev @babel/core @babel/cli @babel/preset-env
```

Dentro de la raíz del proyecto, archivo package.json se ha realizado la configuración.

```
"dependencies": {  
  "browserslist": "^4.24.4"  
},
```

Una vez finalizado este proceso, el programa transformarán las traducciones correspondientes según los diferentes navegadores tanto modernos como antiguos.

7. Semántica y accesibilidad.

Semántica.

La estructura de la web se traduce dentro la semántica de HTML donde se incorporan etiquetas que ayudan a categorizar los diferentes apartados de la web desde el inicio hasta las páginas relacionadas con el contenido principal.

Las etiquetas de estructura utilizadas son las propias que se utilizan en html5 que son, las siguientes :

<head> : metadatos, donde presentan toda la información relevante para que los buscadores puedan identificar el contenido de nuestra web y la categorice de manera correcta.

Dentro de este apartado se detienen las etiquetas **<meta>**, **<title>** y **<link>** (enlaza los estilos CSS y otros elementos externos a la web, como son las tipografías).

<body> : contenido principal de la web.

<header> : cabecera de la web que se va repitiendo el mismo diseño a lo largo de las tres pantallas disponibles “index.html”, “cake_fruits.html”, “cheesecakes.html”.

En el header, se encuentran la barra de navegación **<nav>**, seguido de una imagen y lista no ordenada **** que categorizan los diferentes apartados que forman la web mediante etiquetas **** como son el “inicio”, “tipos de pasteles”, “contactos y “sobre nosotros”.

<main> : contenido más relevante de la página web.

En él se encuentran las secciones **<section>** que se han utilizado para diferenciar las diferentes informaciones que consolidan el tema tratado en la web. Se incluyen otras etiquetas como imágenes **<figure>**, ****, pies de imágenes **<figcaption>** títulos **<h1>**, **<h2>**..., textos **<p>**, enlaces **<a>**, anotaciones ****, vídeos **<video>** y etiquetas que agrupan diferentes elementos **<div>**.

<footer> : se encuentra la información institucional de la web.

En él se encuentran elementos claves como son los enlaces sobre términos y condiciones, política de privacidad, política de cookies , redes sociales disponibles y enlaces rápidos a secciones importantes.

Accesibilidad.

En tema de accesibilidad web, se ha garantizado a que la web sea accesible para todos los usuarios incluidas a aquellas con discapacidad puedan usar el sitio web de manera efectiva. Dichas características son las siguientes:

- 1) **Textos alternativos para imágenes:** usar etiquetas como alt en las imágenes como método descriptivo en caso de que la imagen no se puedan visualizar de manera correcta.
- 2) **Tonos y contraste de colores.** Se utilizado tonos pasteles verdes y blancos que se asemejan a la apariencia del logotipo inicial de la empresa; el color verde trasmite naturalidad y frescura (Ingredientes frescos, naturales y saludables), orgánico(productos artesanales sin aditivos), y cercanía (marca accesible y amigable).
- 3) **Navegación intuitivo.** Los usuarios pueden navegar a través de los diferentes apartados a través de los links, botones y textos descriptivos.
- 4) **Tipografía.** Se ha utilizado fuentes tipográficas sin serifas (Montserrat) para destacar tantos los textos como los títulos. Considero que este tipo de fuentes permite una mejor definición de las letras por pantalla y mejora la labilidad de los lectores.

8. Creación y publicación a Git y Github.

Git.(Repositorio, código creado git add, git commit m-“...” , git push)

Antes de empezar el proyecto web se creación un directorio local dentro del ordenador usando los diferentes comandos de git a través de la terminal. Los comandos utilizados son:

git mkdir = Crea una carpeta a la dirección definida.

git init = inicializa un nuevo repositorio Git.

Una vez finalizados con todos los cambios realizado con los documentos html, css y js es hora de añadir los archivos a Github. Para este proceso se han realizado los siguientes pasos:

En primer lugar, localizar la carpeta del proyecto local con “ cd/PEC1_HTML_CSS”.

Seguidamente, inicializar git con **git init**.

Agregar el directorio online al repositorio local con “**git remote add origin** https://github.com/saraLui03/HTML-CSS.git”.

Y finalmente realizar los pasos finales para subir los archivos a GitHub, que son los siguientes pasos:

git status = visualizar el estado de todos documentos que han sufrido cambios.

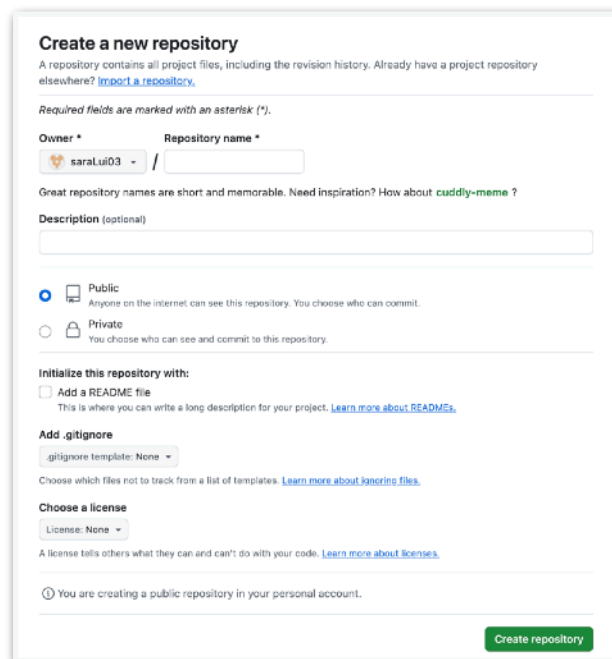
git add = añadir todos los cambios realizados dentro de los diferentes documentos a GitHub.

git commit m-“...” = añadir comentario sobre los cambios realizados a GitHub.

git push = actualizar y enviar todos los cambios realizados a al repositorio online GitHub desde el repositorio local.

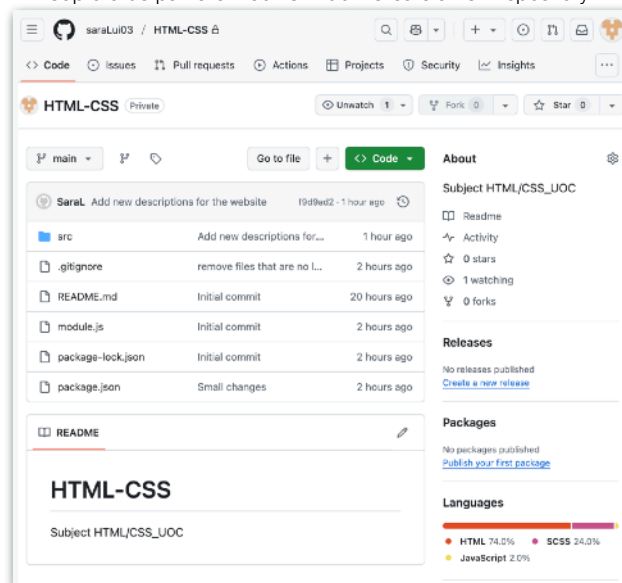
Github(Crear repositorio online, cómo se ha realizado).

Dentro de GitHub crear un nuevo repositorio remoto, rellenando el formulario.
Hacer click a “Create repository” y finalmente acceder al espacio del repositorio online.



The screenshot shows the 'Create a new repository' form on GitHub. It includes fields for 'Owner' (saraLui03), 'Repository name', and 'Description (optional)'. There are radio buttons for 'Public' and 'Private' visibility. Below these are sections for 'Initialize this repository with:' (including 'Add a README file'), 'Add .gitignore' (with a dropdown menu), and 'Choose a license' (with a dropdown menu). A green 'Create repository' button is at the bottom right.

Captura de pantalla web “GitHub”. Create a new repository..



Captura de pantalla web “GitHub”. Create a new repository..

Obtenido en: <https://github.com/saraLui03/HTML-CSS>

9. Publicación a internet.

En esta fase del desarrollo web, concluye con la publicación de la plataforma generada se basa en GitHub como plataforma para crear el repositorio Git local.

Por otra parte, el proveedor del servidor escogido para publicar la web es Netlify; una de las plataformas más populares para desplegar sitios web estáticos y aplicaciones frontend, sin necesidad de pago.

Las **razones** por las que he dedicado usar el proveedor Netlify son:

1. Fácil de usar (ideal para principiantes y expertos).
2. Integración con Git y automatización del despliegue.
3. Rápido y optimizado gracias a su CDN; red de distribución de contenido.
4. Plan gratuito con excelentes beneficios.

Los pasos realizados para la publicación web has sido:

En la página de inicio de “Netlify” ira a “añadir una nueva web” > “importa un proyecto existente” > “GitHub y Escoger el repositorio online correspondiente, en este caso es “HTML-CSS” y finalmente rellenar el siguiente formulario, para poder publicarlo.

Build settings
Specify how Netlify will build your site.
[Learn more in the docs](#)

Branch to deploy
main

Base directory
The directory where Netlify installs dependencies and runs your build command.

Build command
npm run build
Examples: jekyll build, gulp build, make all

Publish directory
dist
Examples: _site, dist, public

Functions directory
netlify/functions
Example: my_functions

Environment variables
Define environment variables for more control and flexibility over your build.
[Add environment variables](#)

[Deploy HTML-CSS](#)

Captura de pantalla web “Netlify”. Build settings. Obtenido en: <https://app.netlify.com/start/repos/saraLui03%2FHTML-CSS>