Índice

Documentación del proceso de desarrollo

Para el desarrollo de la página se ha partido de la instalación para la primera práctica.

Punto de partida

NodeJS versión 14.15.5
npm 6.14.11 en un entorno Windows10.
Parcel versión 1.12.3 con las siguientes dependencias:
rimraf
npm-run-all
autoprefixer 9.8.6
Postcss-css-variables
Prettier
Viewerjs
Como editor de texto: VisualStudioCode

Herramientas utilizadas para trabajar con las imágenes

He probado diferentes herramientas durante el proceso para poder observar los diferentes resultados y poder decidir cual utilizar finalmente.

Microsoft Paint: para obtener versiones de una imagen con diferentes resoluciones cambiando su tamaño. También se ha utilizado como editor para recortar las imágenes en la dirección de arte de la página de detalles.

https://avif.io/: Herramienta online que permite convertir imágenes a formato avif

https://image.online-convert.com/convert-to-webp: Herramienta online para convertir imágenes a formato webp

https://convertio.co/jpg-jp2/: Herramienta online para convertir imágenes a formato Jpeg 2000

https://shrinkme.app/: herramienta online para compresión de jpg, png, webp

File optimizer: Herramienta de escritorio para compresión de imágenes de diversos formatos

Imagemin (parcel-plugin-imagemin): plugin de parcel que realiza compresión automática de imágenes al desplegar con build. Instalación con: npm install parcel-plugin-imagemin --save-dev

imagemin-avif: dependencia de npm con utilidades para convertir imágenes a formato avif. Instalación con: npm install imagemin-avif --save-dev

imagemin-webp: dependencia de npm con utilidades para convertir imágenes a formato webp. Instalación con: npm install imagemin-webp –save-dev

https://boxy-svg.com/app: Editor de imágenes svg online

Repositorio de datos y servidor en producción

Se utiliza GitHub.
URL: https://github.com/ptousd/eines1_pec2

Para la integración con el repositorio se ha instalado y configurado GitBash

Se ha añadido un archivo .gitignore para evitar enviar al repositorio los archivos de las carpetas node-modules y las caches de trabajo de Parcel

Como servidor de producción se utiliza Netlify
URL: https://friendly-lovelace-8d4ae3.netlify.app

Distribución de archivos

Se ha creado una carpeta PEC2 donde se encuentran todos los archivos de desarrollo de la web.
Los archivos HTML de han ubicado directamente en este directorio. Se han creado subdirectorios para contener los archivos SCSS, JS y img respectivamente.

Código

Se ha utilizado el diseño movile-first y para conseguir que las páginas sean responsive cajas flexibles y en grid junto con media-querys. Cuando el navegador no soporta grid se utiliza flex en su lugar.
Se ha intentado aplicar lo aprendido en la asignatura Herramientas CSS y HTML II en cuanto a la nomenclatura y formato de código de selectores CSS. En concreto la guía mdo y la nomenclatura propuesta por BEM.

Proceso Desarrollo

La primera fase la he dedicado a evaluar las diferentes opciones para trabajar con imágenes. Se han utilizado las herramientas ya indicadas para editarlas, convertirlas y comprimirlas.

Durante el desarrollo las modificaciones grandes se realizaban directamente en el código, con parcel en modo desarrollo para ver los resultados directamente en el navegador.
Los ajustes pequeños como cambios de tamaños, padding, margins, colores, etc.. se realizaban primero en las herramientas de desarrollo de Firefox para luego trasladarlas al código fuente.

Cuando concluía el desarrollo de una página se ejecutaba Parcel en modo producción para comprobar los cambios en el código.

Se ha utilizado Github como repositorio. Para ellos se ha tenido que instalar la herramienta Git Bash.
Desde la herramienta se han ejecutado los siguientes comandos:
Inicializar el directorio local como un repositorio Git.
$ git init -b main
Añadir los archivos al repositorio local
$ git add .
Commit de los archivos

$ git commit -m "Primer commit"
Configuración del repositorio remoto
$ git remote add origin https://github.com/ptousd/eines2.git
# Vericación de la URL remota
$ git remote -v
Envio de los cambios del repositorio local al remoto
$ git push origin main

Se puede acceder al repositorio en:
https://github.com/ptousd/eines1_pec2

Para paso a producción se ha utilizado Netlify.
Se puede acceder al proyecto en la url:
https://friendly-lovelace-8d4ae3.netlify.app

Una vez en producción se ha analizado el código con las siguientes herramientas:
Para corregir problemas de html
https://validator.w3.org/

Para corregir problemas de css
https://jigsaw.w3.org/css-validator

Para validar accesibilidad de las páginas online
He tenido que sustituir el que venía usando habitualmente por que ya no esta operativo
https://achecker.ca/checker/index.php
por este otro muy interesante
https://wave.webaim.org/

A partir de aquí se ha hecho Continuous Deployment hasta conseguir que los validadores no encontrasen problemas.

Justificación de las decisiones tomadas en el desarrollo

Decisiones sobre el boilerplate

He continuado utilizado windows 10 a la espera de que se generasen problemas con npm, en este caso los problemas han empezado a producirse al instalar las dependencias de imagemin. A partir de este momento ha sido imposible lograr compilar con npm run dev o build dando siempre un error. Publica la web en github y una vez replicada en Netlify todo funciona correctamente.

Al final he instalado npm en Ubuntu donde imagemin no da problemas y he continuado el desarrollo sin problemas.

Compatibilidad con navegadores antiguos

He mantenido la configuración de la práctica anterior de autoprefix:
last 4 version
IE 6

firefox 16
chrome 10
safari 5
opera 11
ios_saf 5

Decisiones sobre herramientas para el tratamiento de imágenes

Se han evaluado las siguientes herramientas

Formato avif

https://avif.io/: Herramienta online que permite convertir imágenes a formato avif. No permite configurar el nivel de compresión al realizar la conversión, pero es fácil de utilizar simplemente arrastrando los archivos que queremos convertir.

No he encontrado ninguna herramienta online que permita comprimir archivos avif

Formato webp

https://image.online-convert.com/convert-to-webp: Herramienta online para convertir imágenes a formato webp. Gratuita y fácil de usar, permite configurar el nivel de compresión.

https://shrinkme.app/: herramienta online para compresión de jpg, png, webp. Tampoco permite configurar el nivel de compresión al realizar la conversión, pero es fácil de utilizar simplemente arrastrando los archivos que queremos convertir.

Formato JPEG 2000.

https://convertio.co/jpg-jp2/: Herramienta online para convertir imágenes a formato jp2. El principal problema que he encontrado ha sido que las imágenes convertidas a jp2 que probé aumentaban el peso de las jpeg originales. Supongo que es problema del convesor. Por otro lado actualmente este formato está poco soportado por los navegadores así que decidí no usarlo en la web.

Formato SVG

https://boxy-svg.com/app: herramienta de edición de svg. Se han evaluada otras herramientas pero esta fue la única que me permitió trabajar con el svg de LocBaleares sin problemas.

FileOptimizer

Es fácil de utilizar pero al ser manual cada vez que añades una imagen al site hay que acordarse de pasar por el proceso de optimización. En una web de dimensión reducida se podría utilizar, pero al ir creciendo dificultaría seriamente el proceso de gestión de las imágenes.
Otro problema es que de momento no soporta los nuevos formatos: webp, avij o JPEG 2000.

Imagemin

Imagemin (parcel-plugin-imagemin): plugin de parcel que realiza compresión automática de imágenes al desplegar con build.

Al final decidí utilizar esta herramienta ya que en mi opinión es importante automatizar al máximo todo el proceso de generación de una web.
Permite también seleccionar los niveles de compresión a nivel de tipos de archivo y está integrada a nivel de build para reducir el impacto de las modificaciones en desarrollo.

El principal inconveniente, como he comentado anteriormente, es que me fue imposible que funcionase correctamente sobre Windows10, así que terminé montando el entorno de desarrollo sobre Ubuntu.

Otro problema que surgió es que al procesar las imágenes svg eliminaba automáticamente los ids de los elementos. Esto derivaba en que no funcionaba la animación de la página de presentación, ya que se había trabajado con selectores sobre ids. Al final lo tuve que solucionar cambiando los selectores de id por selectores de clases y modificando el svg en este aspecto.

Después de consultarlo con el tutor Xavier Julià encontró la solución siendo este un problema de `htmlnano`, que incluye por defecto Parcel

Añadiendo un nuevo archivo .htmlnanorc con el siguiente contenido funciona:

```
{
        "minifySvg": false
}
```

imagemin-avif: dependencia de npm con utilidades para convertir imágenes a formato avif.
Instalación con: npm install imagemin-avif –save-dev

Para su uso he tenido que crear un script que se tienen que ejecutar con nodejs. No he encontrado documentación de como integrarlo directamente con parcel. Su ventaja sobre la herramienta online es que permite configurar el nivel de compresión e integrarse con nuestro IDE.

Script:

```
const imageminAvif = require('imagemin-avif');

(async () => {
 await imagemin(['../img/ensaimada_grande.jpg','../img/ensaimada_de_nata.jpg'], {
  destination: '../img',
  plugins: [
   imageminAvif({quality: 50})
  ]
 });
  console.log('Avif Images optimized');

})();
```

imagemin-webp: dependencia de npm con utilidades para convertir imágenes a formato webp.
Instalación con: npm install imagemin-webp –save-dev

Para su uso también he tenido que crear un script que se tienen que ejecutar con nodejs. No he encontrado documentación de como integrarlo directamente con parcel. Su ventaja sobre la

herramienta online es que permite configurar el nivel de compresión e indicar que archivos de nuestra web deseamos convertir.

Script:

```
const imageminWebp = require('imagemin-webp');

(async () => {
  await imagemin(['../img/ensaimada_grande.jpg','../img/ensaimada_de_nata.jpg'], {
    destination: '../img',
    plugins: [
      imageminWebp({quality: 50})
    ]
  });

  console.log('Webp Images optimized');
})();

const imageminAvif = require('imagemin-avif');
```

Código de las páginas

Decisiones de estructura de las páginas

Para el desarrollo de la práctica se ha seguido la guía de estilos de mdo en combinación con la metodología BEM para el nombrado y definición de los selectores. Se pretende con ello reducir la especifidad de los selectores y aumentar la reutilización, además de generar un código comprensible para otros desarrolladores que conozcan dichas medodologías.

He continuado con un desarrollo first-mobile ya que he considerado que es la mejor manera de que se vea bien en dispositivos pequeños.

Se han utilizado dos puntos de ruptura con media-queries: el primero es para pantallas de resolución 768px que son las que mas se aproximan a un entorno tablet, y el segundo para 1224px que es para equipos de escritorio. Se han utilizado esto puntos de ruptura ya que se consideran estándares.

Primero se ha desarrollado la web sin puntos de ruptura y probado en dispositivos tipo móvil. En segundo lugar se ha añadido el punto a 768px para adaptar el funcionamiento a tablets, para finalmente añadir el punto de ruptura para visualizar en formato escritorio.
He utilizado los mismos puntos de ruptura para la selección de formatos y archivos de las imágenes.

Como he indicado en el apartado anterior parte del desarrollo se basa en el uso de layouts de tipo grid, para aquellos navegadores que no soportan grid se ha utilizado layout flex en su lugar.

En todas las páginas se ha utilizado la semántica de html en la que se dividen las zonas en header, nav, main, footer y section para facilitar su accesibiliad.

Transiciones

Los enlaces cambian de color y aumentan de tamaña en el estado hover

```
/*Formato de enlaces. Sin decoración por defecto, con borde inferior dashed */
a {
 color: #000;
 text-decoration: none;
 transition: all .2s ease-in-out;
}

/*Formato de enlaces cuando se pasa por encima*/
a:hover {
```

```
 color: var(--font-highlight-color);
 font-size: 1.3em;
}
```

Las imágenes de la clase categoria__img cambian su opacidad en el estado hover

```
.categoria__img {
 width: 100%;
 transition: opacity 1s;
}

.categoria__img:hover {
 opacity: 0.5;
}
```

Cabecera y pie

En la cabecera y pie se ha incorporado un icono que se ha diseñado utilizando la herramienta https://boxy-svg.com/app. Se ha utilizado el formato SVG ya que los iconos tienen un diseño vectorial y este tipo de imágenes se adapta muy bien a los cambios de tamaño de una página responsive sin perdida de calidad.

Index.html (Portada)

La página home es muy sencilla así que se ha decidido que se ajuste al 100% del viewport invalidando la posibilidad de realizar scrolls.

Tratamiento de las imágenes

Se ha cambiado la imagen de background por una una composición de tres imágenes utilizando clip-path.
HTML de la composición:

```
  <div class="bgd-mallorca">
   <div class="clip-top"></div>
   <div class="clip-left"></div>
  </div>
```

CSS explicado

```
/*
Definimos la imagen de fondo sin repetir y cubriendo todo el fondo
Tamaño 100% del viewport
Cuando hay overflow se ocultan los elementos
La imagen está adaptado en peso y forma al tamaño del dispositivo
*/
.bgd-mallorca {
 position: relative;
 top: 0;
 left: 0;
 height: 100%;
 background-image: url('../img/Formatge_i_sobrassada_mobil.jpg');
 @media only screen and (min-width: 768px) {
  background-image: url('../img/Formatge_i_sobrassada_tablet.jpg');
 }
 @media only screen and (min-width: 1440px) {
  background-image: url('../img/Formatge_i_sobrassada.jpg');
 }
 background-repeat: no-repeat;
 background-size: cover;
}

/*
Formato de la imagen superior de la composición usando clip-path
(1) Se utilizan diferentes resoluciones de imágenes según tamaño dispositivo
(2) También se ajusta el formato del clip-path para una mejor
```

```
visualización según el formato pantalla
*/

.clip-top {
 position:absolute;
 top:0;
 left:0;
 width: 100%;
 height: 100%;
 background-image: url('../img/Panadescuites_mobil.jpg');
 @media only screen and (min-width: 768px) {
  background-image: url('../img/Panadescuites_tablet.jpg'); /*1*/
 }
 @media only screen and (min-width: 1440px) {
  background-image: url('../img/Panadescuites.jpg'); /*1*/
 }
 background-repeat: no-repeat;
 background-size: cover;

 clip-path: polygon(0 0, 100% 0, 100% 12%, 55% 55%, 0 12%); /*2*/

 @media only screen and (min-width: 768px) {
  clip-path: polygon(0 0, 80% 0, 35% 50%, 0 12%); /*2*/
 }
}

.clip-left {
 position:absolute;
 left:0;
 bottom:0;
 width: 100%;
 height: 100%;
 background-image: url('../img/Rubiolscuits_mobil.jpg');
 @media only screen and (min-width: 768px) {
  background-image: url('../img/Rubiolscuits_tablet.jpg'); /*1*/
 }
 @media only screen and (min-width: 1440px) {
  background-image: url('../img/Rubiolscuits.jpg'); /*1*/
 }
 background-repeat: no-repeat;
 background-size: cover;
 clip-path: polygon(0 12%, 0 100%, 12% 100%, 55% 55%); /*2*/
 @media only screen and (min-width: 768px) {
  clip-path: polygon(0 12%, 0 86%, 35% 50%); /*2*/
 }
}
```

Categorias.html

En la página de categorías, en la lista se ha utilizado display block para dispositivos pequeños, un grid de dos columnas para dispositivos medianos y un grid de tres columnas en dispositivos grandes. Se ha utilizado grid en lugar de flex ya que da un aspecto más homogéneo a toda la página. Para ajustar cada uno de los items de la lista de categorías se ha utilizado un display en flex. Se ha utilizado esta opción ya que de esta manera se rellena mejor el espacio.

Tratamiento de las imágenes

Para las imágenes se ha utilizado una versión minimizada de la imagen que se utilizá en la página de detalle. Para ello se ha cambiado el tamaño utilizando la herramienta paint de microsoft. Después de estudiar como cambian los tamaños de las imágenes utilizando la herramienta de inspección de Firefox se ha determinado que una buena resolución sería la de 400px en horizontal y la reducción proporcional en vertical. Estas imágenes **no son responsive** ya que su peso ya es bajo y siempre tienen un viewport reducido, por lo que no se ha visto la necesidad de proporcionar diferentes fuentes. El navegador siempre acabaría utilizando el de menor tamaño.

Para las imágenes de esta página se ha seguido la siguiente nomenclatura:

- nombre_imagen.jpg para la imagen de la página categorías con tamaño 400px
- nombre_imagen_max.jpg para la imagen de alta definición de la página detalles

Detalle.html

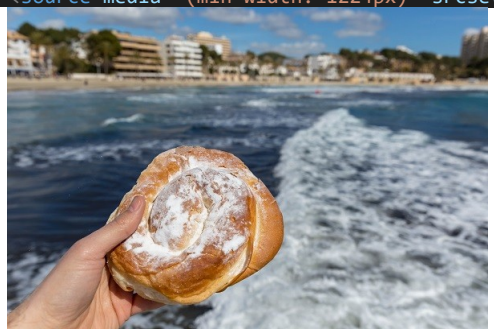
En la página detalle se ha utilizado un formato de dos columnas en grid para dispositivos grandes y una columna en block para pequeños – medianos. Se ha utilizado esta página para practicar las diferentes formas de trabajar con imágenes responsive.

Tratamiento de las imagenes

En este caso la imagen principal se ha utilizado un picture con sources diferentes para cada tamaño de dispositivo. Además se ha realizado una dirección de arte de manera que, además del cambio de tamaño, se ha adaptado el diseño de la foto para una mejor visualización.

Ejemplo:

```
<picture>
 <source media="(min-width: 1224px)" srcset="img/ensaimada_detalle_max.jpg">
```



```
<source media="(min-width: 768px)" srcset="img/ensaimada_detalle_tablet.jpg">
```

```
<img class="detalle__img" src="img/ensaimada_detalle.jpg" alt="Ensaimada"/>
```

```
</picture>
```

Se ha utilizado un pequeño gif animado aun ignorando la recomendación de no utilizarlos. El motivo es que considero que puede dar un toque gracioso a un página siempre que no sea de un peso demasiado elevado y por otro lado existen gran cantidad de repositorios de imagenes con licencia creative commons en animaciones en gif y muy pocos de svg.

Se han incorporado 3 nuevas imágenes al detalle utilizando diferentes técnicas responsive

En dos se han utilizado formatos de imagen diferentes. En concreto avif y webp. No se ha utilizado j2 ya que de momento no está demasiado extendido su uso. Además se añade un source diferente para dispositivos que no admiten formatos más comprimidos cuando la pantalla es grande.
Por defecto se utilizará un jpeg reducido para versiones móviles.
Las diferentes versiones se han generado utilizando imagemin-avif e imagemin-webp

```
    <picture>
      <source type="image/avif" srcset="img/ensaimada_grande.avif">
      <source type="image/webp" srcset="img/ensaimada_grande.webp">
      <source media="(min-width: 768px)" srcset="img/ensaimada_grande.jpg">
      <img class="descripcion__img-left" src="img/ensaimada_grande_movil.jpg" alt="Foto ensaimada de
chocolate">
    </picture>
```

En una tercera imagen se ha utilizado la técnica de adaptación según tamaño de pantalla y viweport (Resolution switching: Different sizes) para facilitar la decisión del navegador durante el pre-load.

```
    <img class="detalle__img" src="img/ensaimada_2048.jpg" alt="Ensaimada para llevar"
    srcset="img/ensaimada_160.jpg 160w, img/ensaimada_400.jpg 320w, img/ensaimada_800.jpg 640w, img/
ensaimada_1024.jpg 1024w"
    sizes="(max-width: 1224px) 95vw, 65vw">
```

Se ha modificado el comportamiento de las imágenes de la lista de productos relacionados para trabajar con Resolution switching: Same size, different resolutions.

Ejemplo de código:

```
        <img class="categoria__img"
         srcset="img/robiols_400.jpg,
                 img/robiols_600.jpg 1.5x,
                 img/robiols_800.jpg 2x"
        src="img/robiols_400.jpg" alt="Robiols">
```

Presentacion.html

La página de presentación tienen un formato similar, utilizado un formato de dos columnas en grid para dispositivos grandes y una columna en block para pequeños – medianos. A esta página se le ha añadido la posibilidad de ver las imágenes en formato carrousel pulsando sobre cualquiera de ellas. En lugar de hacer un desarrollo propio se han buscado dependencias de npm y se ha utilizado finalmente: viewerjs. En esta página se encuentra también un pequeño script para lanzar la funcionalidad.

Tratamiento de las imágenes

A partir de la imagen svg original LocBaleares.svg se ha añadido el nombre de cada isla utilizando la herramienta de edición online https://boxy-svg.com/app

Se ha editado con un editor de texto el archivo svg para cambiar los id de los elementos a los que se va a hacer referencia en la animación
Se ha incrustado el código de la imagen en el documento html (a sido la única forma de poder hacer referencia a los id del svg)
Se han añadido una animación de 10s (configurable con una variable scss) en la que, en orden, cada isla va cambiando de color y aparece su nombre haciendo un zoom. Los nombres inicialmente no se ven. La animación se repite indefinidamente.

El resultado de codificación es el siguiente (Ejemplo para una isla; las demás están configuradas igual repartiendo el tiempo en 5 partes, uno para cada isla)

```
@keyframes mallorca {
 0% {
  fill:#ffffd0;
 }
 20% {
  fill:lightcoral;
 }
 21%, 100% {
  fill:#ffffd0;
 }
}

@keyframes mallorca-txt {
 0% {
  opacity: 0;
  font-size: 2em;
 }
 20% {
  opacity: 1;
  font-size: 4em;
 }
 21%, 100% {
  opacity: 0;
 }
}

#mallorca-svg {
 animation: var(--animation-time) infinite backwards mallorca;
}

#svg-mallorca-txt {
 animation: var(--animation-time) infinite backwards mallorca-txt;
}
```

Para el resto de las imágenes de la página se ha utilizado la versión en formato 400px ya que siempre ocupan un espacio reducido y no necesitan de gran resolución.
Se ha utilizado la utlilidad viewerjs para poder visualizar las imágenes en carousel.

Problema con Viewerjs
El javascript del visor se puede configurar para que se utilice otra source de imagen cuando se esta visualizando en el carousel. La idea original era que al verse en el visor se cambiase el src para utilizar la imágen de mayor calidad *_max.jpg

Para ello modifique el código del javascript de la siguiente manera:

```
const gallery = new Viewer(document.getElementById('images'), {
 title: [
  4,
  (image, imageData) => `${image.alt} (${imageData.naturalWidth} × ${imageData.naturalHeight})`,
 ],
```

```
 url(image) {
  return image.src.replace('.jpg', '_max.jpg');
 },
});
```

El resultado no fue el esperado ya que parcel cambia el nombre de las imagenes añadiendoles un sufijo lo que hace imposible que funcione el script anterior.

Al final he optado por no utilizar la versión ampliada en el carousel.

Enlaces.html

Página que contiene imágenes con enlaces al origen de su descarga.

Tratamiento de las imágenes

Para todas las imágenes se utilizan dos fuentes distintas en función de la resolución del dispositivo. Se utiliza la técnica Resolution switching: Different sizes.

Se ha utilizado la siguiente nomenclatura:

- nombre_imagen.jpg para la imagenes de baja resolución en dispositivos pequeños (son los mismos que en la página categorías)
- nombre_imagen_max.jpg para la imagen de alta definición en dispositivos más grandes (son las que se utilizaran como imagen destacada en la página detalles)

Explica los resultados obtenidos al aplicar estas decisiones

Resultados sobre los recursos multimedia

La aplicación de técnicas como clip-path, masking y la aplicación de animaciones y transiciones a las imágenes da un aspecto más moderno a la web.

Trabajar con Resolution switching y con dirección de arte permite reducir el coste de la carga de las imágenes y en el caso de la dirección de arte obtener una mejor experiencia de usuario.

Los nuevos formatos como avif o webp reducen el tamaño de las imágenes sin casi perdida de calidad. Es importante con estos formatos el uso de picture interrogando a los navegadors sobre su soporte y utilizando el src adecuado en consecuencia. Ejemplo de codificación:

```
    <picture>
     <source type="image/avif" srcset="img/ensaimada_de_nata.avif">
     <source type="image/webp" srcset="img/ensaimada_de_nata.webp">
     <source media="(min-width: 768px)" srcset="img/ensaimada_de_nata.jpg">
     <img class="descripcion__img-right" src="img/ensaimada_de_nata_movil.jpg" alt="Foto ensaimada de nata">
    </picture>
```

Al final por razones de automatismo e integración con el IDE se ha optado por trabajar con imagemin, pese a los problemas que me ha generado. El resultado es la compresión automática de los archivos. En este aspecto cabe resaltar que se ha optado por utilizar niveles con poca compresión. Con el trabajo previo realizado sobre cada imagen para la obtención de versiones en

diferentes resoluciones ya se reduce la carga de trabajo en dispositivos con conexiones lentas, así que la compresión es un plus añadido.

Veamos a continuación la lista de tamaños de las imágenes después de aplicar cambios de formato y compresión

Imágenes con diferentes formatos.

El mejor resultado es con avif, luego webp

ensaimada_de_nata.jpg	ensaimada_de_nata.jpg Comprimida con imagemin al 85%	ensaimada_de_nata.webp Comprimida al 50%	ensaimada_de_nata.avif Comprimida al 50%
492K	212K	102K	80K
ensaimada_grande.jpg	ensaimada_de_nata.jpg Comprimida con imagemin al 85%	ensaimada_grande.webp Comprimida al 50%	ensaimada_grande.avif Comprimida al 50%
584K	347K	148K	109K

Imágenes con diferentes versiones

Archivo	Archivo 400px	Archivo _max	400px calidad imagemin 85%	_max calidad imagemin 85%
camallot	27K	258K	18K	132K
coca_pebres	66K	456K	43K	364K
dorada_mallorquina	40K	413K	25K	172K
ensaimada_detalle	91K	116K	62K	74K
ensaimada	53K	597K	30K	347K
Frito	51K	750K	32K	424K
Galletas_inca	52K	163K	41K	89K
Pa amb oli	30K	1,3M	19K	591K
Sobrassada	32K	105K	24K	80K
Sopes Mallorquines	119K	1,9M	68K	526K

Conclusiones sobre los beneficios obtenidos

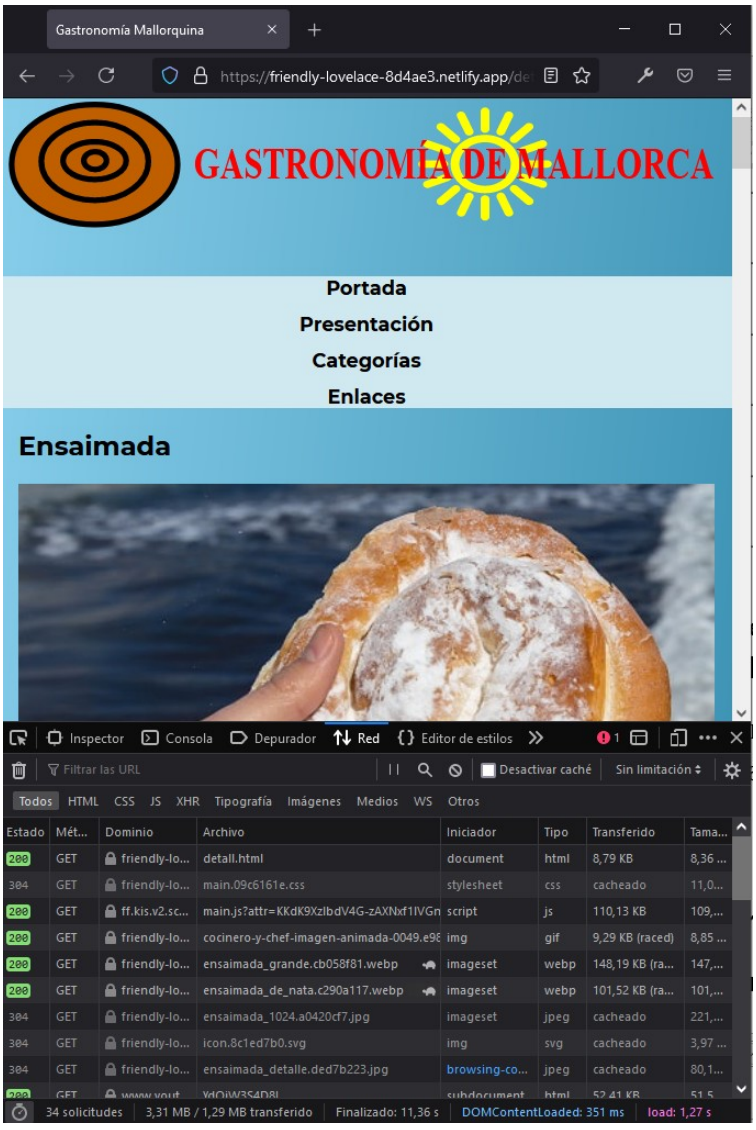
Como se puede observar se obtiene un gran beneficio en la página de categoría y en la de enlaces al utilizar las versiones de menor resolución. El beneficio se ve ampliado al utilizar la compresión automática de imagemin.

En la página de detalles la optimización se deriva del uso de formatos de imagen como webp y avif por un lado, y por el uso de técnicas de Resolution switching.

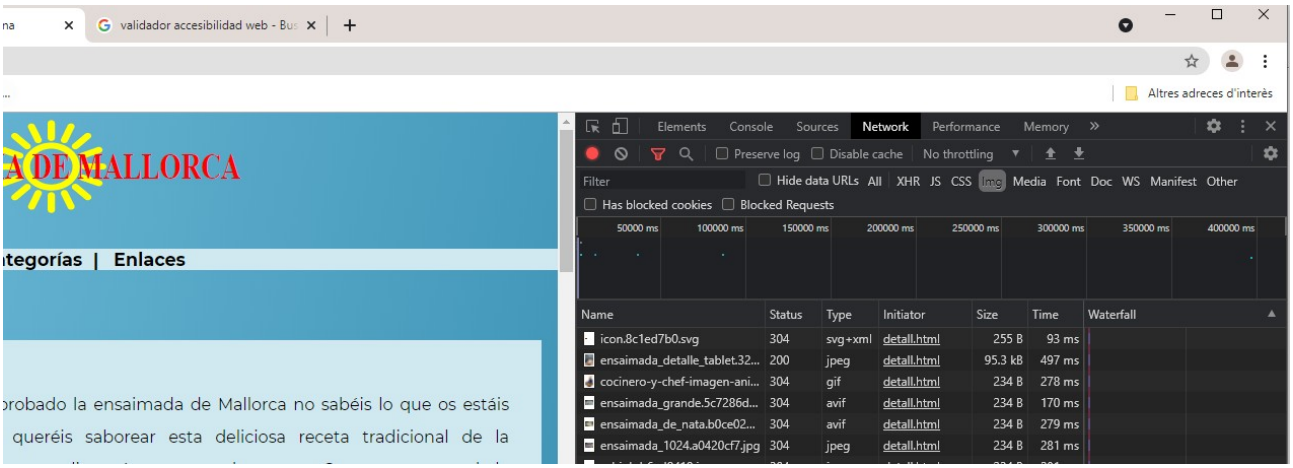
La optimización en la página de portada se obtiene al utilizar media-queries para las imágenes de background, utilizando una combinación de diferentes resoluciones y dirección de arte según el tamaño del dispositivo.

Para poder comprobar que archivos carga el navegador en diferentes circunstancias he utilizado las herramientas del desarrollador de los diferentes navegadores utilizando la opción red o network, junto con la herramienta de emulación de dispositivos móviles.

Resultado en Firefox



Resultado en Chrome



Resultados de uso de la herramientas

(Esta parte está repetida de la práctica anterior al haberse utilizado como punto de partida)

La primera decisión y la mas importante es haber utilizado un boilerplate moderno cuyo resultado inmediato es la reducción de plazos de desarrollo.

La utilización de parcel como module bundler reduce los errores de codificación al realizar un precompilado de archivos css y javascript, permitiendo la detección de estos antes de su ejecución.

La posibilidad de monitorizar los cambios en los archivos refrescando el navegador automáticamente que proporciona parcel en modo desarrollo, facilita mucho el trabajo de diseño de las páginas ya que se pueden visualizar inmediatamente todos los cambios.

Utilizar postcss con autoprefixer permite olvidarse de las reglas css específicas de navegadores en versiones mas antiguas.
Un ejemplo de conversión:

```
.receta__paso {
 display: flex;
 flex-direction: row;
 justify-content: space-between;
 margin-right: 4rem;
 border-bottom: 2px dotted #ddd;
}
```

Se convierte a:

```
.receta__paso {
 display: -webkit-box;
 display: -moz-box;
 display: -ms-flexbox;
 display: flex;
 -webkit-box-orient: horizontal;
 -webkit-box-direction: normal;
   -moz-box-orient: horizontal;
   -moz-box-direction: normal;
    -ms-flex-direction: row;
      flex-direction: row;
 -webkit-box-pack: justify;
    -moz-box-pack: justify;
     -ms-flex-pack: justify;
      justify-content: space-between;
 margin-right: 4rem;
 border-bottom: 2px dotted #ddd;
```

```
}
```

Otro de los problemas que surge con los navegadores más antiguos de que no permiten trabajar con variables en css. Como autoprefixer no soluciona este problema se ha instalado una dependencia adicional: postcss-css-variables, que sustituye las variables por sus valores en tiempo de precompilación.
De esta forma obtenemos los beneficios de trabajar con variables pero no perdemos compatibilidad con los navegadores antiguos que no las soportan.

Por ejemplo:

```
:root {
 --bg-color-1: #87ceeb;
 --bg-color-2: #05668b;
 --box-shadow-color: #023f57;
 --font-highlight-color: #a01127;
 --bg-highlight-color: rgba(255, 255, 255, 0.8);
}

body {
margin: 0;
background: linear-gradient(to right bottom,var(--bg-color-1),var(--bg-color-2));
font-family: 'Montserrat', serif;
}
```

se convierte a

```
body {
 margin: 0;
 background: -webkit-gradient(linear, left top, right bottom, from(#87ceeb), to(#05668b));
 background: -webkit-linear-gradient(left top, #87ceeb, #05668b);
 background: -o-linear-gradient(left top, #87ceeb, #05668b);
 background: linear-gradient(to right bottom, #87ceeb, #05668b);
 font-family: "Montserrat", serif;
}
```

El uso de un precompilador como babel también ha simplificado el trabajo ya que se puede trabajar con las versiones más actuales de javascript sin preocuparse de si será correctamente interpretado por los navegadores mas antiguos.

Veamos un ejemplo:
El javascript utilizado para el carrousel de fotos en la página de presentación está codificado en ES6:

```
const gallery = new Viewer(document.getElementById('images'), {
 title: [
  4,
  (image, imageData) => `${image.alt} (${imageData.naturalWidth} × ${imageData.naturalHeight})`,
 ],
});
```

y al tener configurado browserlist para navegadores más antiguos ha transformado el código en:

```
var gallery = new _viewerjs.default(document.getElementById('images'), {
 title: [4, function (image, imageData) {
  return "".concat(image.alt, " (").concat(imageData.naturalWidth, " \xD7 ").concat(imageData.naturalHeight,
")");
 }]
});
```

- Const se transforma en var
- Narrow Function a function
- La plantilla de string a cadena con concats

Para finalizar cuando utilizamos parcel se reduce el número de ficheros ya que compacta automáticamente los archivos scss en uno solo y además, cuando generamos con build, minifica los js y css, lo que reduce el tiempo de carga en los navegadors.

Github permitiría trabajar en equipo con un repositorio compartido.

Netlify nos permite tener un hosting de las páginas adaptado a node.js.