UO251436 - DOCUMENTACIÓN NEW

NUEVOS ESTÁNDARES EN LA WEB

Eduardo Ulibarri Toledo (uo251436@uniovi.es)

1. Introducción y enlace a la entrega

Para la asignatura se ha desarrollado una web con la temática de un blog/revista digital de videojuegos de nombre "Gamer News". La página se encuentra desplegada en http://156.35.95.118:8084/, en uno de los equipos de la escuela.

La entrega consta de 3 documentos HTML, uno por cada una de las páginas de la web:

1. *index.html*: Página de bienvenida a la web. Contiene un mensaje de bienvenida y un grupo de noticias destacadas en un *aside*. El *aside* permite hacer *scroll* con la asistencia de las técnicas de ***scroll-snap***.
2. *news.html*: Página con el listado de noticias publicadas a lo largo del tiempo en la página. En este punto se han aplicado las técnicas de ***lazy loading***.
3. *news_detail.html*: Página de una noticia concreta.

El esqueleto de la página está creado con ***grid layout*** y en el marcado HTML se han seguido los estándares semánticos. Se dará más información de cada una de las páginas a lo largo del documento.

Junto a los documentos HTML se encuentran 3 carpetas:

1. *styles*: Estilos de la web. Contiene la subcarpeta ***sass***, dado que se ha utilizado este preprocesador para generar los CSS finales.
2. *js*: Código JavaScript de la web. Contiene el script *ui.js* (ayuda con algunos elementos visuales) y *lazy_load.js* (lógica del *lazy loading*).
3. *media*: Imágenes y multimedia utilizados en la web.

2. Descripción de la página web

2.1 Generalidades

El sitio web desarrollado simula la que sería la web de una revista de videojuegos. Cabe destacar que los enlaces relacionados con redes sociales de la revista o sus miembros no llevan a ninguna parte. Así mismo, y por simplicidad, todos los enlaces a noticias llevan a una misma noticia que ha sido preparada para ello.

2.1.1 Estructura general de la web

Todas las páginas de la web se estructuran en cuatro partes principales:

- Navegación: elemento ***<nav>***, que ocupa la parte superior.
- Cuerpo: elemento ***<section>*** con la clase "*main-content*".
- Lateral: elemento **<aside>** con la clase "*side-content*". La página de noticia concreta no tiene aside.
- Pie: elemento ***<footer>***, que ocupa la parte inferior.

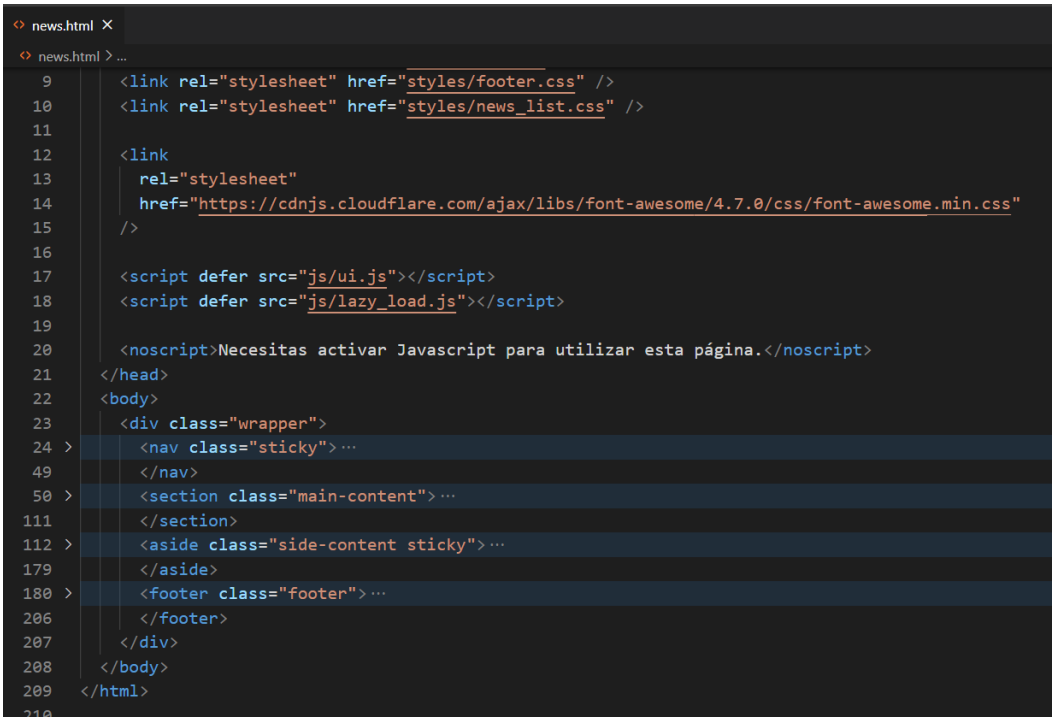

Ilustración 1. Estructura general del HTML de la web

2.1.2 Estilado general de la web

SASS

Para los estilos de la web se ha utilizado el preprocesador SASS con las ventajas que este trae sobre el CSS tradicional: declaración de variables en el CSS, selectores anidados, etc. (más información en *3.1-Utilización de SASS*). Los ficheros de SASS (*.scss*) están en la carpeta */styles/sass* de la web y se han convertido en los CSS finales de la carpeta */styles*.

Todas las páginas se basan en el mismo CSS: "***base.scss***", al que luego cada página concreta añade información adicional mediante su hoja de estilos concreta (*home.scss*, *footer.scss*, etc.)

Maquetación de la página

El esqueleto general de la web se ha construido mediante ***grid layout***. Todos los contenidos de la web están dentro de un *div* de clase "*wrapper*" con este layout. Se ha especificado un grid de 5 filas y 3 columnas.

```scss
.wrapper {
  display: grid;
  grid-gap: 20px;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: auto auto 1fr auto auto;
  color: $background-color;

  min-height: 100vh;
}
```

Ilustración 2. Elemento "wrapper" utilizando grid layout

El elemento *<nav>* con la navegación siempre ocupa la primera fila del grid mientras que el elemento *<footer>* ocupa la última. El contenido principal de la página así como el *<aside>* (de haberlo) ocupan el resto de las filas, distribuyéndose en horizontal si el dispositivo es suficientemente ancho o en vertical en caso contrario.

index.html | base.scss | nav.scss

styles > sass > nav.scss > ...

```scss
/* Nav */

@import "variables.scss";

nav {
  grid-row: 1;
  grid-column: 1 / 4;
  display: flex;
  justify-content: center;
  align-items: center;
```

Ilustración 3. Disposición del nav en el "grid"

index.html | base.scss | footer.scss

styles > sass > footer.scss > footer > .footer-text

```scss
/* Footer */

@import "variables.scss";

footer {
  grid-column: 1 / 4;
  grid-row: 5;

  display: flex;
  width: 100%;
```

Ilustración 4. Disposición del footer en el "grid"

Responsividad

Para hacer la página web responsiva se han tenido en cuenta 4 tipos de dispositivos:

- *Móviles*: hasta 480px de ancho.
- *Móviles grandes/tablet*: hasta 768px de ancho.
- *Escritorios pequeños*: hasta 1024px de ancho.
- *Escritorios grandes*: de 1024px en adelante.

Para alternar estilos en base a estos tamaños se han definido (en SASS) 3 variables correspondientes a los 3 *breakpoints*. De entre ellos, el *breakpoint* principal es el que define el paso entre tablet y escritorio, ya que ahí se decide si el menú de navegación de va a mostrar como menú hamburguesa o si el *<aside>* se va a mostrar al lado o debajo del contenido principal. El resto de *breakpoints* ayudan a fijar tamaños de fuente o márgenes.

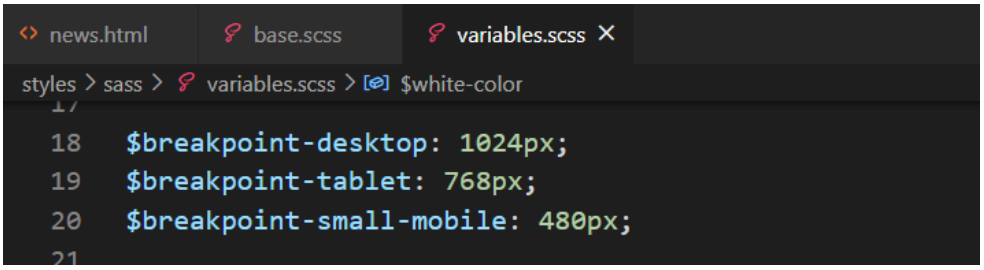

Ilustración 5. Breakpoints definidos en SASS

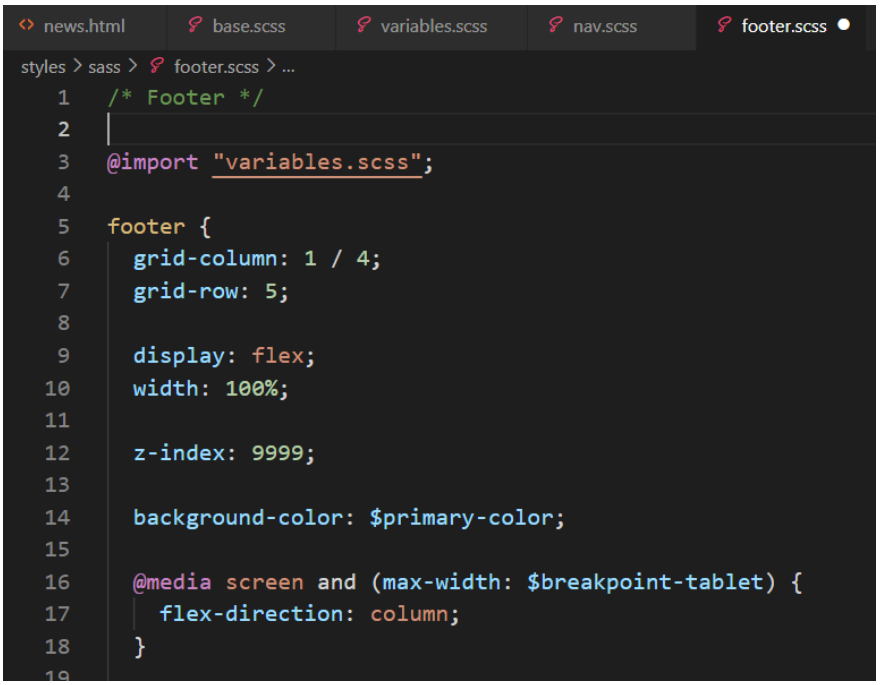

Ilustración 6. Ejemplo de uso de media-queries para cambiar la orientación del footer

Cómo afecta el tamaño del dispositivo a la web se detallará a lo largo del documento, según se hable de cada elemento.

Otros

- Se ha conseguido que tanto la barra de navegación como el contenido lateral sigan al usuario aunque este baje por la página con la propiedad "***position: sticky***" de CSS, asociada a la clase "*sticky*". Por defecto se indica que "***top: 0***" pero los estilos propios del elemento pueden sobrescribirlo si es necesario.

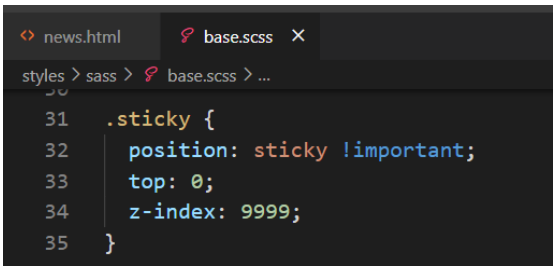

Ilustración 7. Definición de la clase sticky, empleada en el <nav> y <aside>

- Para facilitar la tarea de dar un buen aspecto a la web, se ha utilizado la fuente provista por Font Awesome.

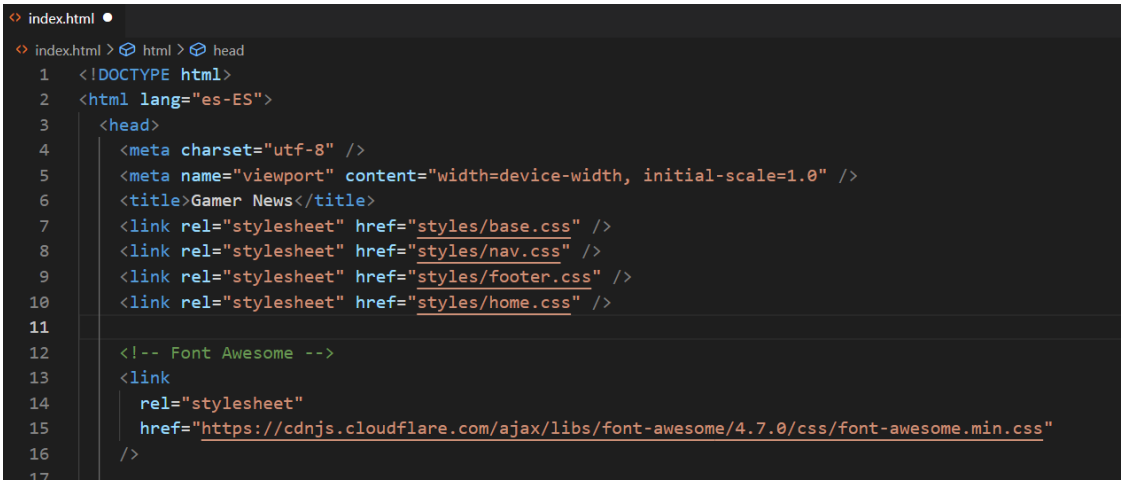

Ilustración 8. Importación de Font Awesome desde la cabecera de los HTML

2.1.3 Elementos generales

Navegación

Aunque la página sólo tenga dos apartado principales se ha añadido un menú de navegación de cara a futuro. El menú ocupa la primera fila del grid de la página y se vale de *flex* para tener el siguiente aspecto:

Ilustración 9. Menú de navegación en escritorio

news.html | base.scss | variables.scss | nav.scss

styles > sass > nav.scss > ...

```scss
/* Nav */

@import "variables.scss";

nav {
  grid-row: 1;
  grid-column: 1 / 4;
  display: flex;
  justify-content: center;
  align-items: center;

  background-color: $primary-color;

  margin-bottom: 25px;
```

Ilustración 10. Estilos generales del menú de navegación (nav.scss)

Además, el menú tiene la clase *sticky* que, como mencionamos antes, hace que se mantenga siempre posicionado en la parte superior según descendemos por la página.

Cuando el dispositivo reduce su anchura, cambian varios aspectos del layout:

1. Se esconden los apartados y se deja ver tan solo el logo y el elemento "***nav-toggle***" encargado de mostrar u ocultar el menú. Este elemento (3 barras blancas) se ha construidos con 3 elementos ** con fondo blanco y altura fijada. El primer elemento y el último se posicionan de forma absoluta mientras que el del medio se coloca allí sin especificar nada.

Ilustración 11. Menú de navegación en tablets y móviles (plegado)

```scss
span {
  background-color: $white-color;

  position: absolute;
  width: 100%;
  height: 3px;

  transition: transform 0.3s ease, opacity 0.2s ease;

  &:nth-child(1) {
    top: 0;
  }

  &:nth-child(3) {
    bottom: 0;
  }
}
```

Ilustración 12. Estilado de los para formar el menú hamburguesa

2. Al hacer *click* en el elemento "hamburguesa", un código JavaScript en *ui.js* modifica la clase del ***nav-toggle*** y con ello cambia la forma del menú a un aspa. El menú de navegación de móviles cubre todo la pantalla del dispositivo y permanece fijo aunque hagamos scroll gracias a "*position: fixed*".

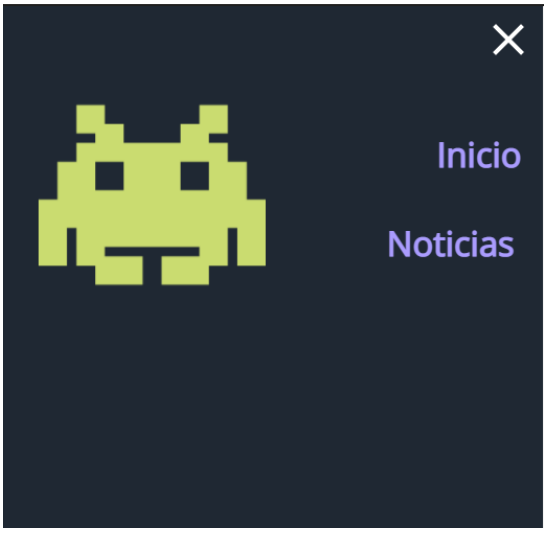

Ilustración 13. Menú de navegación en tablets y móviles (desplegado)

```
241      span {
242        &:nth-child(1) {
243          top: 50%;
244          transform: rotate(45deg);
245        }
246
247        &:nth-child(2) {
248          opacity: 0;
249        }
250
251        &:nth-child(3) {
252          top: 50%;
253          transform: rotate(-45deg);
254        }
255      }
256    }
```

Ilustración 14. Estilado de los para formar el aspa de cierre

```
86   .nav-container {
87     .brand {
88       display: none;
89     }
90
91     @media screen and (max-width: $breakpoint-tablet) {
92       position: fixed;
93       top: 0;
94       left: 0;
95
96       height: 100%;
97       width: 100%;
```

Ilustración 15. Estilado de los contenidos del menú en tablet y móviles para cubrir toda la pantalla

Footer

El footer ocupa siempre la quinta y última fila del grid que forma el esqueleto de la página. Consta de dos mitades diferenciadas que se posicionan de forma horizontal o vertical en función del dispositivo (una *media-query* modifica la propiedad "***flex-direction***" y hace otros cambios menores).

Ilustración 16. Footer en escritorio

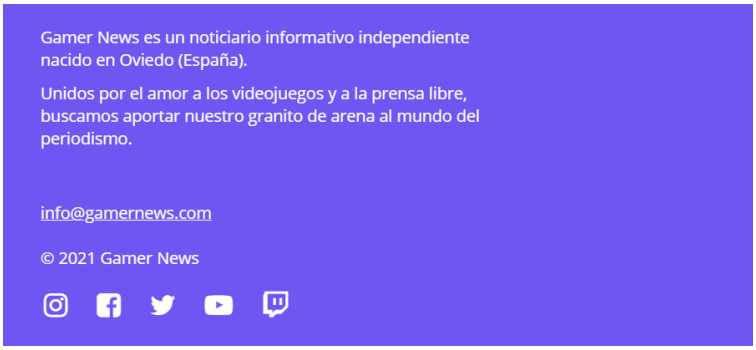

Ilustración 17. Footer en tablet y móviles

Del resto sólo cabe destacar los iconos a redes sociales, que se posicionan dentro de un con "*display: flex*" y que usan transiciones en el color para quedar vistosos.

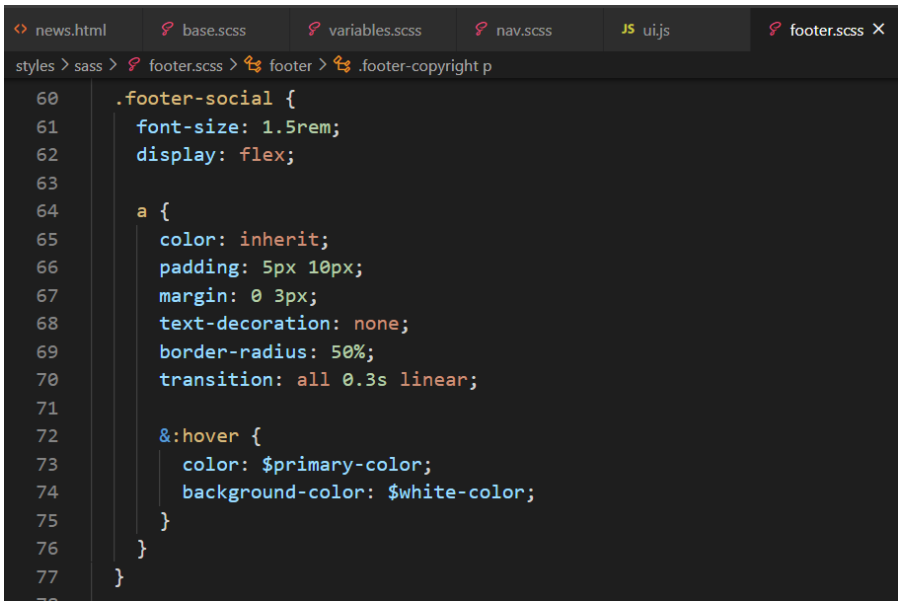

Ilustración 18. Estilado de los iconos sociales del footer

Los iconos de las redes sociales se obtienen gracias al estilo enlazado de Font Awesome:

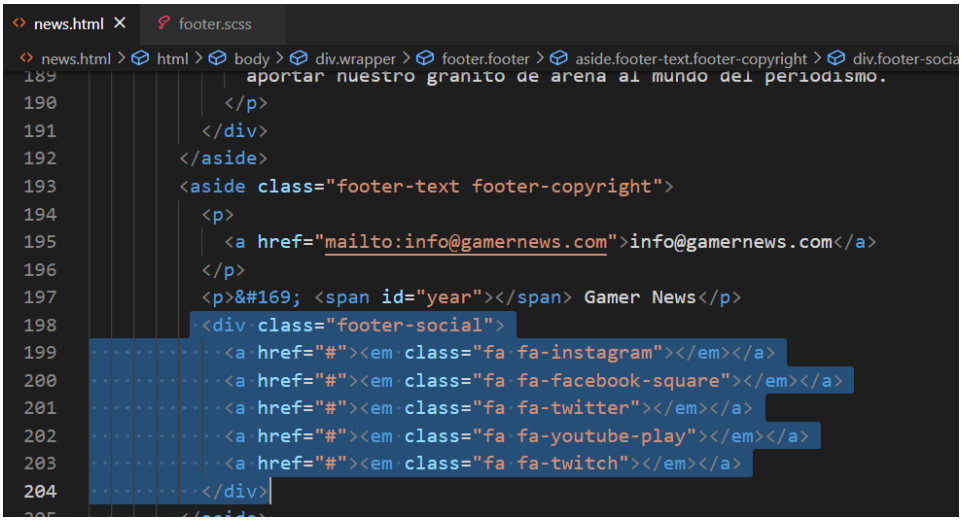

Ilustración 19. Iconos a redes sociales de Font Awesome

Aside

El elemento *<aside>* es un intento de imitar un **carrusel** de noticias destacadas/recientes con enlaces e imágenes. El elemento aside tiene dos posiciones:

1. *Escritorio*: en la segunda fila del grid, donde se le reserva la última columna:

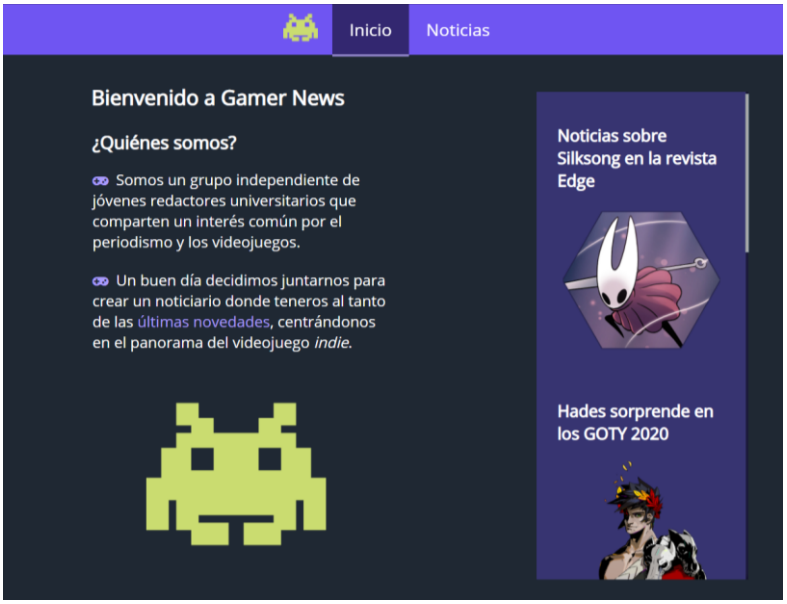

Ilustración 20. Elemento <aside> en escritorios (derecha)

```
news.html    base.scss
styles > sass > base.scss > .side-content
173   /* Side content */
174   .side-content {
175     grid-column: 3;
176     grid-row: 2;
177
178     max-height: 550px;
179
```

Ilustración 21. Estilado por defecto del <aside> dentro del grid

2. *Tablets y móviles*: en la penúltima fila del grid, tan sólo por encima del *footer*. Se coloca en posición horizontal ocupando todas las columnas de la fila y se restringe el tamaño de cada noticia al 50% de la anchura:

Ilustración 22. Elemento <aside> en móviles y tablets

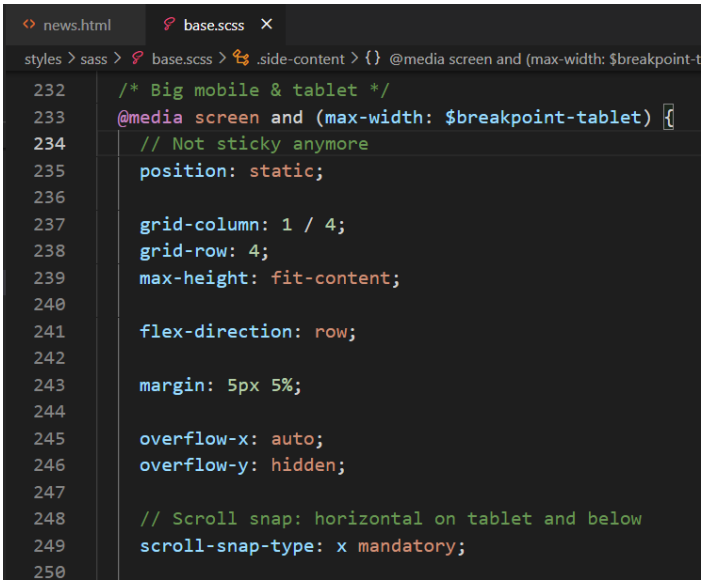

Ilustración 23. Estilado del <aside> en tablets y móviles

Se puede ver cómo al meter varias noticias en el *<aside>* provoca que podamos hacer *scroll* a través de él. Se ha aprovechado esta circunstancia para añadir **scroll-snap** a los elementos del *<aside>*. Los elementos hijos cuentan con ***"scroll-snap-align: start"***.

```
// Scroll snap: vertical on desktop
scroll-snap-type: y mandatory;

/* Big mobile & tablet */
@media screen and (max-width: $breakpoint-tablet) {

  // Scroll snap: horizontal on tablet and below
  scroll-snap-type: x mandatory;
```

Ilustración 24. Configuración de scroll-snap en el <aside>

Además, y de forma similar al *<nav>*, el *<aside>* tiene la clase "*sticky*", por lo que acompaña al usuario según baja por la página.

2.2 Páginas de la web

2.2.1 Inicio

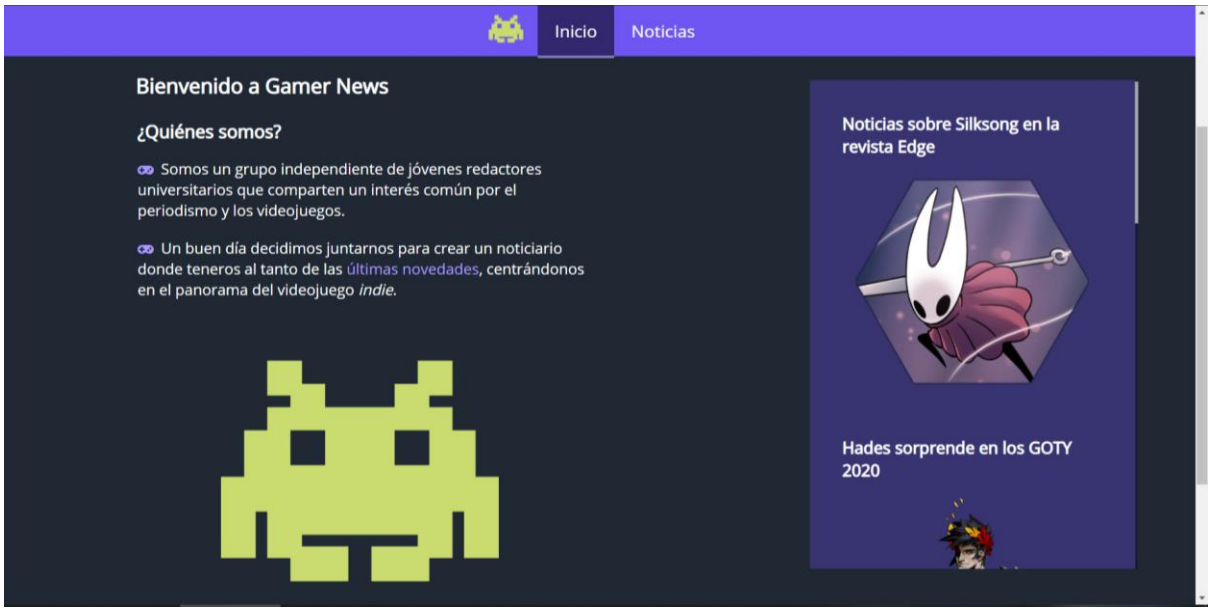

Ilustración 25. Vista general de la página de inicio (escritorio)

Ilustración 26. Vista general de la página de inicio (tablets y móviles)

La página de inicio tiene los siguientes elementos y *layout* en escritorio:

1. *Menú de navegación*: Ocupa la primera fila del grid y todas las columnas.
2. *Contenido principal*: Ocupa todas las filas disponibles hasta el footer y dos de tres columnas. Tiene un mensaje de bienvenida.
3. *Contenido aparte*: Ocupa una fila a la altura del contenido principal y la última columna del grid. Contiene un carrusel de noticias destacadas.
4. *Footer*: Ocupa la última fila del grid y todas las columnas.

En el paso a móviles se dan los siguientes cambios:

1. *Contenido principal*: Ocupa todas las columnas disponibles y deja una fila libre debajo.
2. *Contenido aparte*: Ocupa todas las columnas disponibles y la fila bajo el contenido principal y sobre el footer.

2.2.2 Listado de noticias

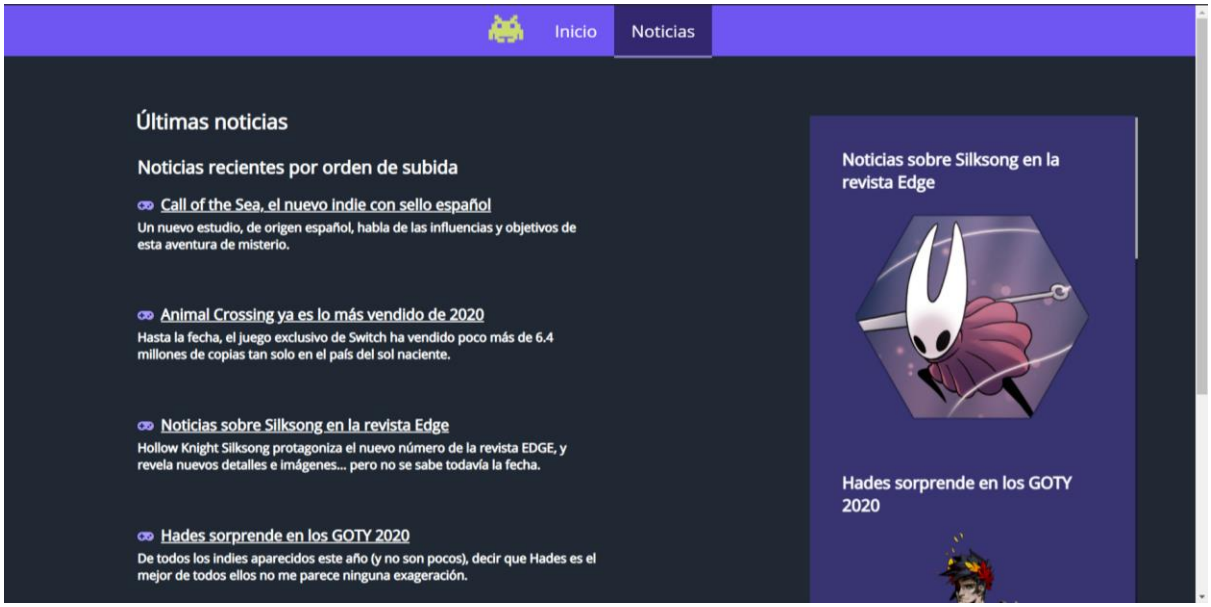

Ilustración 27. Vista general del listado de noticias (escritorio)

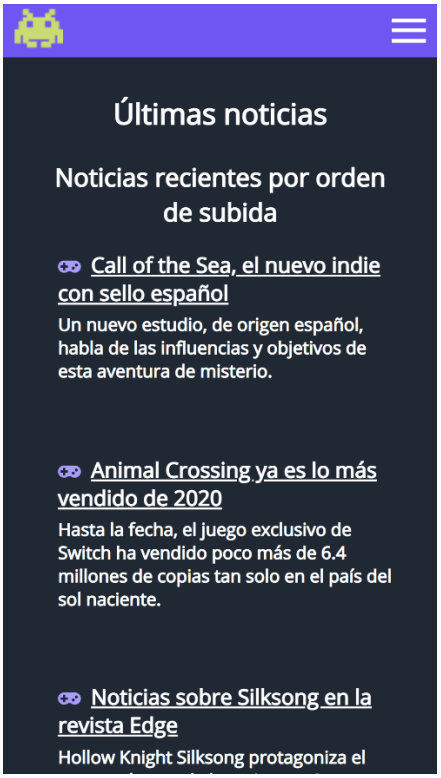

Ilustración 28. Vista general del listado de noticias (tablets y móviles)

La página de noticias tiene los siguientes elementos y *layout* en escritorio:

1. *Menú de navegación*: Ocupa la primera fila del grid y todas las columnas.
2. *Contenido principal*: Ocupa todas las filas disponibles hasta el footer y dos de tres columnas. Contiene un listado (inventado) de noticias publicadas en la página. Se aprovecha el contexto del listado de noticias para añadir ***lazy load*** y que según bajamos aparezcan nuevas noticias infinitamente.
3. *Contenido aparte*: Ocupa una fila a la altura del contenido principal y la última columna del grid. Contiene un carrusel de noticias destacadas que acompaña al usuario según baja.
4. *Footer*: Ocupa la última fila del grid y todas las columnas.

En el paso a móviles se dan los siguientes cambios:

1. *Contenido principal*: Ocupa todas las columnas disponibles y deja una fila libre debajo. Muestra el listado “infinito” de noticias.
2. *Contenido aparte*: Desaparece. Al haber infinitas noticias no vamos a llegar a la parte inferior de la web y no vamos a poder verlo.

En esta página el contenido principal es un listado de noticias. Se ha hecho uso de las técnicas de ***lazy load*** para que, cuando la última noticia cargada entre en pantalla, se añadan nuevas noticias al documento HTML y así nunca dejemos de descender, como en un periódico real. El código responsable se encuentra en ***lazy_load.js***. Lo que hace a grandes rasgos es:

1. Añadir un ***IntersectionObserver*** que observe el elemento que contiene la última noticia del HTML. Siempre se utiliza el mismo observador, que va cambiando de blanco.

```
// Método auxiliar: obtener la última noticia renderizada
const lastNew = () => {
  return document.querySelector(".news-list li:last-child");
};

const observer = new IntersectionObserver(callback, options);
observer.observe(lastNew());
```

Ilustración 29. Inicialización del IntersectionObserver

2. Cuando el ***IntersectionObserver*** se "activa":
    a. Dejar de observar el elemento actual.
    b. Añadir al DOM nuevos elementos.
    c. Empezar a observar la última noticia cargada en el DOM.

```
JS lazy_load.js ×
js > JS lazy_load.js > ...
20
21  const callback = (entries, observer) => {
22    entries.forEach((entry) => {
23      if (!entry.isIntersecting) {
24        return;
25      }
26
27      // Last new was intersected, trigger logic
28      console.info("INTERSECTED: Inserting news.");
29
30      // Stop observing current target
31      observer.unobserve(entry.target);
32
33      // Insert new elements
34      addNews();
35
36      // Update target and observe it
37      observer.observe(lastNew());
38    });
39  };
40
```

Ilustración 30. Lógica activada por el IntersectionObserver

2.2.3 Noticia concreta

Ilustración 31. Vista general de la página de noticia (1)

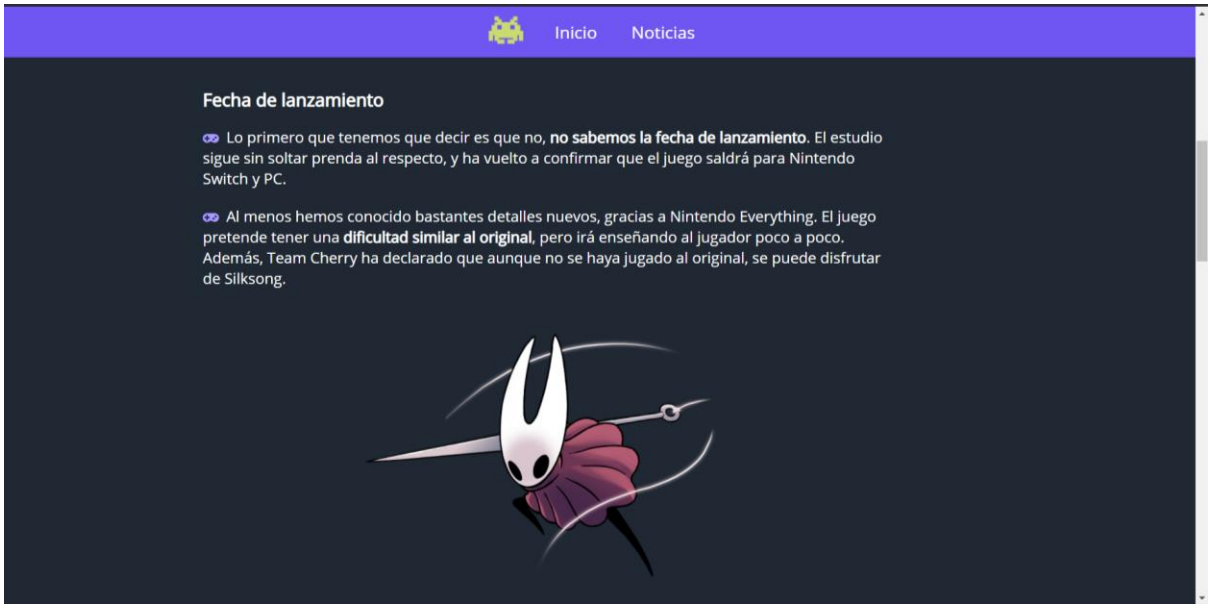

Ilustración 32. Vista general de la página de noticia (2)

La página de la noticia tiene los siguientes elementos y *layout* en escritorio:

1. *Menú de navegación*: Ocupa la primera fila del grid y todas las columnas.
2. *Contenido principal*: Ocupa todas las filas disponibles hasta el footer y todas las columnas. Contiene el cuerpo de la noticia.
3. *Contenido aparte*: No está presente en esta página.
4. *Footer*: Ocupa la última fila del grid y todas las columnas.

No hay cambios significativos en el paso a móviles.

3. Ampliaciones

3.1 Utilización de SASS

Como se ha explicado anteriormente en el documento, se ha utilizado el preprocesador SASS para definir el CSS en ficheros **.scss** y transpilarlos sobre la marcha a ficheros **.css**. Para esto ha sido necesario instalar el paquete "***sass***" de forma global desde NPM y luego invocarlo con la opción "***—watch***" especificando las carpetas de entrada y salida[1].

Se han aprovechado en concreto dos de las características de SASS:

1. Definición de variables: SASS permite definir variables (globales o locales). En este caso definí un conjunto de variables globales en un fichero aparte "***variables.scss***" que luego pude importar y reutilizar en cada hoja de estilos. Las variables me permitieron principalmente editar los colores principales de la página desde un mismo punto, así como los *breakpoints* de mis *media-queries*.

variables.scss ×

styles > sass > variables.scss > $white-color

```
$white-color: #fff;

$background-color: #1f2833;
$primary-color: #6F55F2;
$secondary-color: #AA9CFE;

$nav-active-color:#332770;
$nav-active-border-color: #867BBD;
$nav-link-color: #AA9CFE;
$nav-open-background-color: #1f2833;

$motto-margin-color: #d9b08c;
$motto-color: #ffcb9a;
$aside-background-color: rgba(108, 79, 255, 0.3);

$bullet-icon: "\f11b";

$breakpoint-desktop: 1024px;
$breakpoint-tablet: 768px;
$breakpoint-small-mobile: 480px;

$scrollbar-thumb-color: darkgray;
$scrollbar-thumb-outline-color: slategrey;

```

Ilustración 33. Fichero "variables.scss" con valores reutilizables en todos los estilos

[1] https://sass-lang.com/install

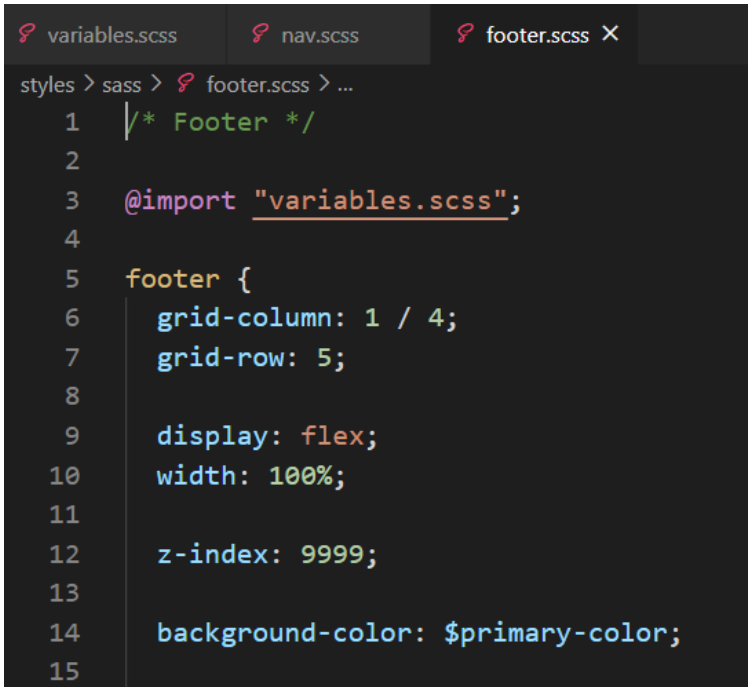

Ilustración 34. Importación de las variables y ejemplo de uso en el footer

2. Selectores anidados. SASS permite redactar CSS de forma menos "verbosa" al poder anidar reglas unas dentro de otras. Se ha utilizado esta práctica en todas las hojas de estilos.

3.2 Utilización de marcado semántico

Se ha procurado respetar el marcado semántico en lo posible.

- Las páginas cuentan con un *<head>* que detalla su idioma, título y hojas de estilo asociadas.
- Los elementos que componen el esqueleto de las páginas están contenidos en su etiqueta correspondiente: *<nav>*, *<section>*, *<aside>* y *<footer>*.
- Dentro del contenido principal:
  - Se usan los encabezados de forma coherente.
  - Se subdivide el contenido en *<section>* si corresponde.
  - El cuerpo de las noticias se mete en una etiqueta *<article>*.
- En la página de inicio aparece el eslogan de la revista que, al ser una cita y su epígrafe, están en un elemento *<q>* y *<small>*.
- Las imágenes están contenidas en un elemento *<figure>* si corresponde.

3.3 Realización del layout general mediante *grid layout*

En la web se ha utilizado *grid layout* para especificar cómo deben disponerse los 4 elementos que forman el esqueleto de cada página, como se ha explicado en *2.1.1-Estructura general de la web*.