

GRUPO JAMÓN

TRABAJO FINAL

INGENIERÍA WEB

Guillermo Azcona
Eduardo Ezponda
Helen Portuondo

ÍNDICE

1. Análisis y diseño
2. Wireframes y prototipado
3. API empleada
4. Javascript
5. HTML
6. CSS
7. Funcionalidad
8. Mobile First / Responsive Design
9. Implementación mediante REACT
10. Usabilidad
11. Accesibilidad
12. SEO
13. Rendimiento
14. Despliegue
15. Gestión configuración (GitLab)
16. Resumen y conclusión final

1. ANÁLISIS Y DISEÑO

Nuestra página Web se llama: [BookRealm](#)

Se trata de una página Web enfocada al mundo del libro, en la que se puede encontrar información acerca de autores relevantes, libros que ahora mismo están en tendencia y noticias de interés global.

El objetivo fijado desde un inicio para este proyecto fue que la página estuviese enfocada a un **público adulto**, que les pudiese mostrar contenido serio y educativo. Por otro lado, no queríamos ser una página en la que se mostrasen datos sobre libros únicamente. Decidimos que nuestro público debía recibir otro tipo de información actual y relevante. Para ello, **analizamos otras páginas de libros** en esta fase inicial de **investigación** para comprobar qué funcionalidad podíamos implementar a la hora del diseño de la web.

Finalmente, los miembros del equipo decidimos que **BookRealm** tendría secciones en las que podría informarse acerca de las últimas tendencias y noticias (implementando un Blog o una página con información relevante), todo sin perder el enfoque hacia el mundo de la lectura. Así, conseguiríamos una **experiencia de usuario distinta** al resto de páginas de libros.

De esta forma, una vez contemplados los aspectos que queríamos tener en cuenta y haber analizado otras páginas, con un objetivo común comenzamos con el modelado del diseño de la web.

Decidimos organizar la web con una **Landing page** potente, una sección de About Us sobre información acerca de los desarrolladores de la web, una página de Blog y otras **dos páginas dinámicas** en las que se **obtiene el contenido** directamente de la **API**, cubriendo una gran variedad de secciones dentro de la página web.

2. WIREFRAMES Y PROTOTIPADO

En cuanto al prototipado de la página, la plataforma empleada ha sido **Figma**, ya que dispone de muchas herramientas para que pueda simular el aspecto de la web de la forma más completa posible.

Una vez se había comentado qué contenidos queríamos que recogiese nuestra web, decidimos emplear una paleta de colores que pudiese hacer resaltar los datos más importantes, y elegimos la plugin de **Coolors** para seleccionar aquella que más se adaptase a nuestra idea.

Otra plugin seleccionada fue la de **Iconify**, para obtener ciertos iconos que queríamos incorporar en la Landing page.



(paleta seleccionada con Coolors)



(iconos elegidos con Iconify)

Otro de los motivos por los que Figma ha sido de gran ayuda es porque todos los integrantes del equipo teníamos acceso y podíamos editar a la vez o de forma remota el prototipo. Además, había varias páginas en las que desarrollar pruebas, lo cual es muy útil para trabajar de forma organizada.

Un aspecto a destacar que nos fue muy útil, especialmente al principio de la fase de diseño, fue la creación de componentes. Gracias a esta herramienta, podíamos ver cómo iba a ir funcionando la Web al pulsar los botones o cualquiera de los elementos del menú y así comprobar a qué páginas iba. Este diseño dinámico fue clave en las primeras fases, ya que todos los integrantes del grupo teníamos claro cómo programar las componentes más adelante y cuál iba a ser el comportamiento.

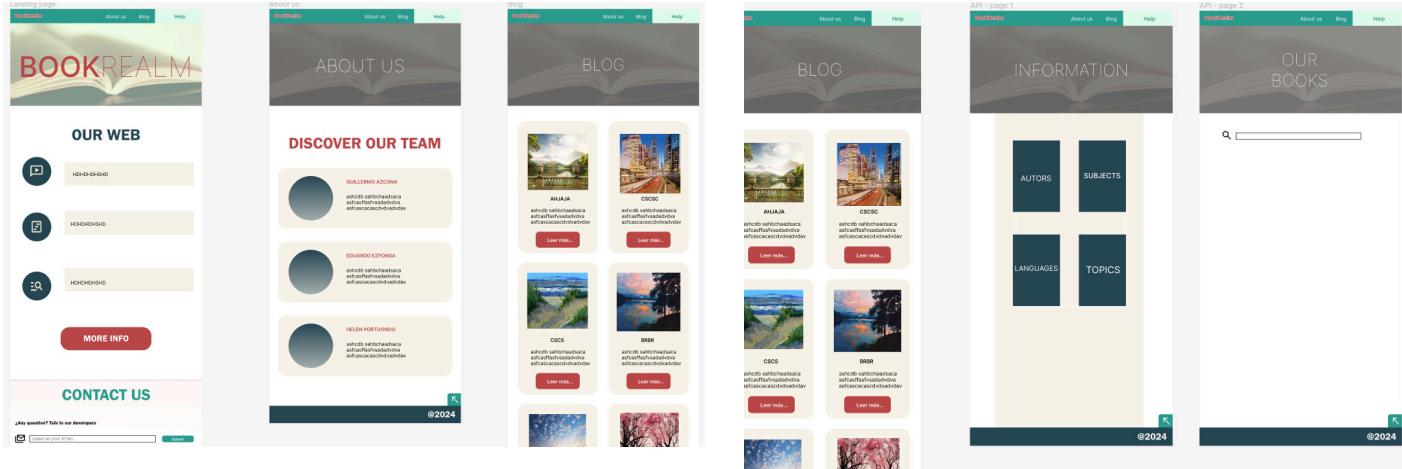
A screenshot of the Figma application interface. The top navigation bar includes 'Layers', 'Assets', and a dropdown menu for 'Páginas'. The left sidebar shows a hierarchical tree view of the project structure: 'Landing page' (selected), 'About us' (expanded), 'Blog', 'API - page 1', and 'API - page 2'. Under 'About us', there are groups like 'Group 24', 'Frame 3', 'Group 23', 'Group 22', and two sections labeled 'DISCOVER OUR TEAM' each containing a 'Group 21' and 'Group 13'. The main canvas area is currently empty.

En cuanto a la gestión de Frames dentro de la página del proyecto final, los realizamos de forma estructurada.

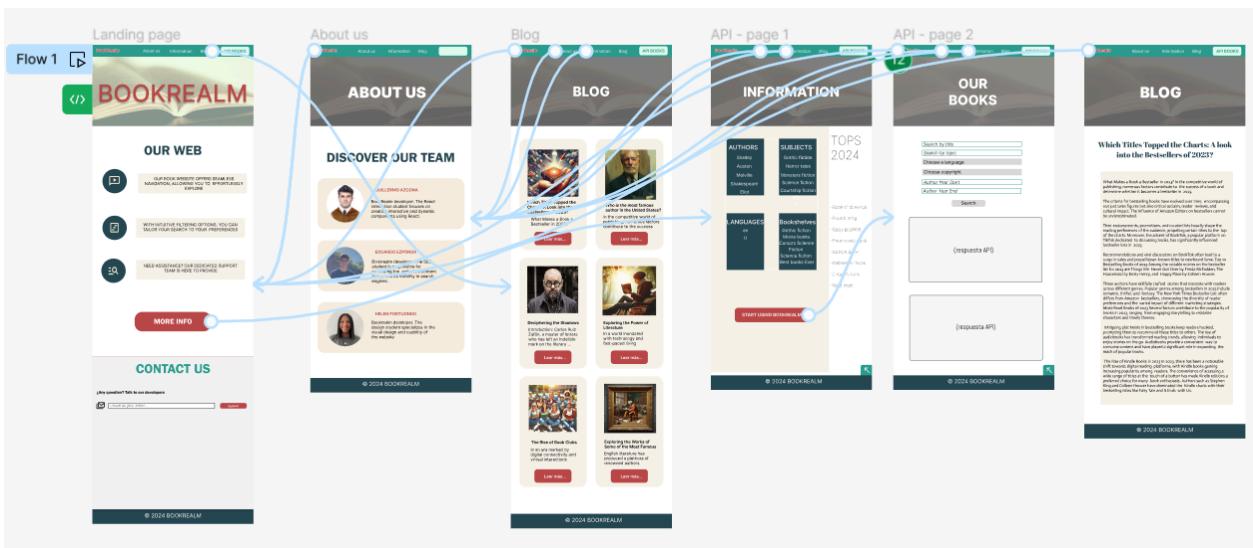
Dentro de cada frame teníamos los elementos (texto, botones, imágenes...) que pertenecieran a él. Además, cuando era necesario, se creaban grupos para poder mover toda la sección a la vez.

En la parte izquierda se muestra una imagen de cómo estaba organizada nuestra página:

A través del histórico de Figma, hemos podido encontrar el **prototipo inicial**, que ha sufrido cambios durante el proceso de desarrollo de la Web. A medida que íbamos programando, surgían ideas de secciones que podíamos añadir o mejorar estéticamente, para proporcionar la mejor experiencia de usuario posible.



A continuación, mostramos una imagen del **prototipo final** dentro de la aplicación [Figma](#) y también de la interacción que diseñamos.



(Imagen de la interacción entre elementos y páginas)

3. API EMPLEADA

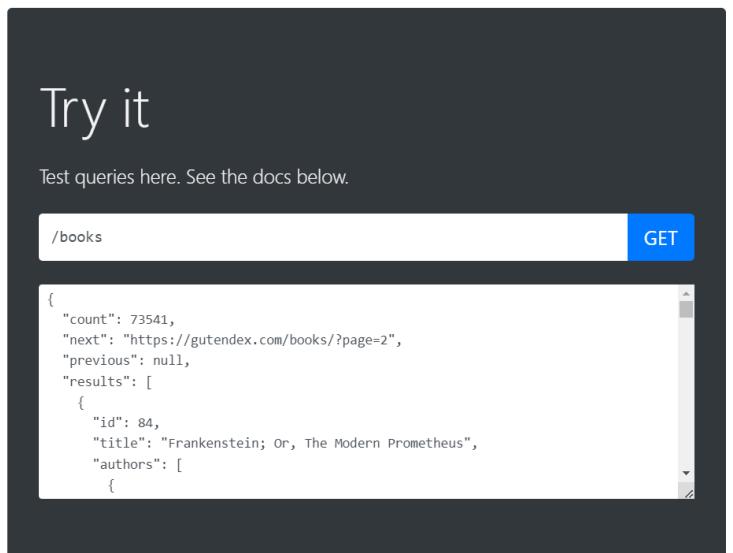
La API elegida para la realización del proyecto es la api de libros:

<https://gutendex.com/>

En ella, podemos encontrar información acerca de los libros en base a una gran cantidad de parámetros como bien son:

- Título
- Idioma
- Autores
- Temas de los que trata
- Traducciones
- Si tiene o no copyright
- Número de descargas
- Formatos que presenta

(En la imagen se muestra un ejemplo de la salida de datos al solicitar los libros con los que cuenta la API.)



El motivo por el que se seleccionó esta API fue porque estábamos interesados en realizar una página de filtrado, y nos pareció que tenía los parámetros suficientes como para satisfacer una búsqueda completa. En caso de no realizar filtrado, se mostrarían todos los libros y, además, no tardaba demasiado tiempo en cargar la información, lo que haría que la página no se quedase cargando mucho tiempo.

Otro de los motivos por los que la Web está en inglés en todo momento es porque las respuestas estaban en ese idioma, y nos pareció más coherente mantener una misma lengua en toda la web (que, a su vez, es la más hablada).

Como conclusión, la API de libros Gutendex nos ha resultado sencilla de utilizar y con todos los datos que necesitábamos, lo que ha sido de gran ayuda para las dos páginas dinámicas de nuestra Web (Information y API Books).

4. JAVASCRIPT

En nuestro proyecto, hemos utilizado Hooks en cada uno de nuestros componentes para manejar estados y otros efectos secundarios. Además de ello, hemos usado constantes para realizar funcionalidades o obtener valores de determinadas acciones o parámetros.

Por ejemplo, en nuestro endpoint books manejamos las listas de los libros y los estados de cargar con **useState**. En otro caso, utilizamos **useEffect** en information para realizar la consulta a la API con una función asíncrona.

Para contrastar, hemos usado un **callback** en vez de un **useEffect** en el endpoint books para hacer la petición a la API y así obtener y tratar los datos al usuario.

La biblioteca **React Router** se ha implementado para manejar la navegación entre secciones comportándose de como si una sola página se tratara. A través de la etiqueta **Link** podemos navegar entre nuestra aplicación sin recargar la página.

Para finalizar, queremos destacar el uso de **variables condicionales** y el tratamiento de su valor a través del **use State**. Estas variables han sido muy importantes a la hora de modelar el desarrollo de las secciones. Gracias a ellas, se han mostrado etiquetas e incluso componentes en función del valor que tuvieran en ese momento.

```
const Books = () => {
  const [books, setBooks] = useState([]);
  const [searchQuery, setSearchQuery] = useState("");
  const [category, setCategory] = useState("");
  const [copyright, setCopyright] = useState("");
  const [topic, setTopic] = useState("");
  const [authorYearStart, setAuthorYearStart] = useState("");
  const [authorYearEnd, setAuthorYearEnd] = useState("");
  const [error, setError] = useState("");
  const [searchAttempted, setSearchAttempted] = useState(false);
  const [isLoading, setIsLoading] = useState(false);

  const fetchBooks = useCallback(async () => {
    let url = `https://gutendex.com/books?search=${searchQuery}`;
    setSearchAttempted(true);
    setIsLoading(true);
```

5. HTML

Dentro del apartado del html, el uso de etiquetas semánticas ha sido fundamental a lo largo de todo el proyecto. Se han utilizado la menor cantidad de etiquetas div, debido a que no reflejan ningún tipo de información a los buscadores ni al propio código.

De arriba para abajo, se han utilizado etiquetas como header y hero, para implementar los menús de navegación (nav) y mostrar el título con la imagen característica de nuestra página web. Cada uno de los enlaces se redirige con la etiqueta Link para una mayor accesibilidad.

```
const Header = () => (
  <header className="header">
    <nav className="navigation">
      <Link to="/" className="logo">
        BOOKREALM
      </Link>
      <div className="menu">
        <Link to="/about">ABOUT US</Link>
        <Link to="/blog">BLOG</Link>
        <Link to="/information">INFORMATION</Link>
        <Link to="/books" className="APIbooks">
          API BOOKS
        </Link>
      </div>
    </nav>
  </header>
```

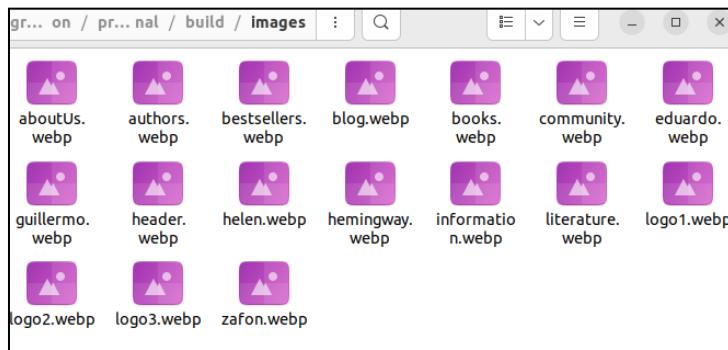
En otras secciones del código se implementan otras etiquetas como main, section, aside... La etiqueta article es imprescindible para cada uno de los artículos que mostramos en el blog. El aside se utiliza para mostrar los libros más buscados de todo Estados Unidos en 2024 en una sección apartada a la derecha del endpoint principal de /Information.

```
return (
  <article aria-labelledby="authors-title">
    <h1 id="authors-title">
      Exploring the Works of Some of the Most Famous English Authors of All
      Time
    </h1>
    <div className="content">
      <p>
        English literature has produced a plethora of renowned authors whose
        works have left an indelible mark on the world of writing. From classics
        to contemporaries, these writers have captured the imagination of
        readers of all ages and influenced the way we view the world.
      </p>
    </div>
  </article>
```

```
return (
  <aside className="aside-information" aria-labelledby="top-books-2024">
    <h2 id="top-books-2024">
      <strong>Top Books of 2024</strong>
    </h2>
    <ul>
      <li>
        <a
          href="https://www.imdb.com/title/tt22866358/"
          aria-label="Read about Book of Clarence"
        >
          <strong>Book of Clarence</strong>
        </a>
      </li>
    </ul>
  </aside>
```

Se ha hecho uso también de otras etiquetas como strong para dar énfasis, y abbr para mostrar abreviaturas como FAQ (Frequently Asked Questions).

Cómo hacemos uso de bastantes imágenes en el proyecto, se han pasado cada una de las imágenes a **formato webp** para una mayor compresión y así hacer una página web más dinámica. Se utilizan imágenes en el hero, about us, y en los artículos de los blogs. Además, se ha tratado de escoger imágenes cuyo tamaño ya sea pequeño para no tener que reducirlo y perder rendimiento a pesar de mostrar un tamaño más pequeño.



Para finalizar, hemos incorporado en la landing page un contacto a tipo de formulario para simular el uso de una posible newsletter y así informar al usuario de nuevos artículos en nuestra sección del blog. Se ha intentado hacer una simulación con una herramienta que se llama email octopus para automatizar el envío de correos electrónicos con campañas, pero no disponía de esta funcionalidad en la versión gratuita. El formulario de la página web envía los datos a través del método POST. Esto significa que los datos estarán en la cabecera del paquete.

```
<form method="POST" onSubmit={handleSubmit} aria-label="Contact Form">
  <label for="email" className="visually-hidden">
    Email:
  </label>
  <input
    id="email"
    type="text"
    value={email}
    onChange={e => setEmail(e.target.value)}
  />
  <button type="submit" aria-label="Send Message">Send Message</button>
</form>
```

6. CSS

En cuanto al uso de CSS en nuestro proyecto desde React, hemos estructurado todo el CSS en carpetas, en función de qué partes de diseño íbamos implementando. De esta forma lográbamos una organización interna a la hora de trabajar sobre las distintas páginas de nuestra web. Por otro lado, añadimos un css concreto para el menú y el footer con el objetivo de poder reusar código que iba a ser necesario en todas las páginas. En el index.css configuramos todos los headers que contenían cada una de nuestras secciones.

```
▽ styles
# AboutUs.css
# App.css
# Blog.css
# Books.css
# Contact.css
# Footer.css
# Information.css
# Menu.css
# Principal.css
JS App.js
JS App.test.js
# index.css
```

En el html de todas las secciones empleamos clases e ids (en función de si vamos a editar una sección completa que se encuentra en un <div>, por ejemplo, o de si queremos editar simplemente un elemento puntual).

Hemos hecho uso de la **notación BEM** en algunas partes del código, como es el caso del endpoint “About US”, ya que necesitábamos estructurar los <div> que usábamos y a la hora de diseñar el css nos iba a ser útil esta nomenclatura. A continuación se muestra una imagen de su implementación en html y posterior diseño:

```
const AboutUs = () => {
  return (
    <div class="AboutUsSection">
      <h2>DISCOVER OUR TEAM</h2>
      <div class="AboutUsSection__Team">
        <div class="AboutUsSection__Team--person">...
        </div>
        <div class="AboutUsSection__Team--person">...
        </div>
        <div class="AboutUsSection__Team--person">...
        </div>
      </div>
    </div>
  );
};

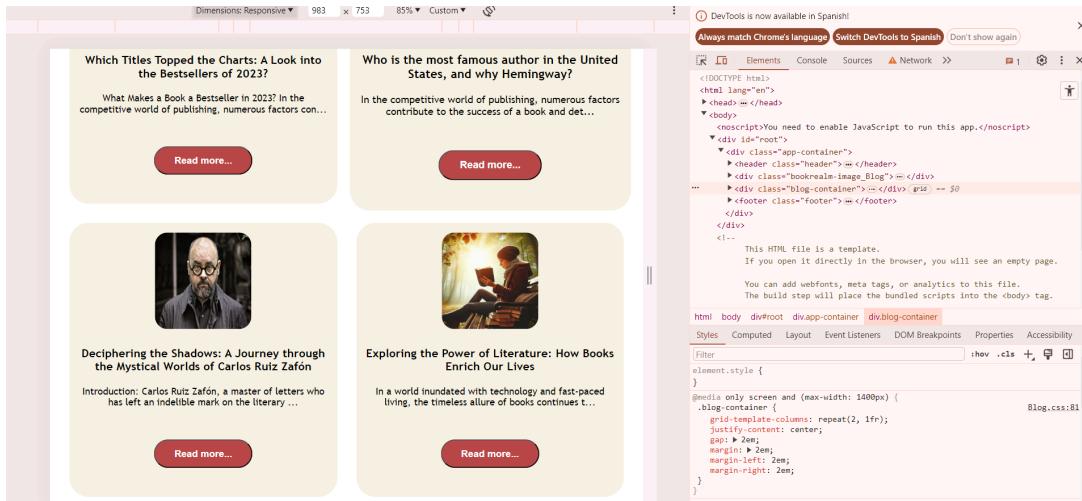
.AboutUsSection h2 {
  margin-bottom: 30px;
  color: #264653;
  font-family: "Trebuchet MS", "Lucida Sans Unicode"
  font-size: 3em;
  text-align: center;
}

.AboutUsSection__Team {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  margin-bottom: 4em;
  width: 80%;
  margin: auto;
}

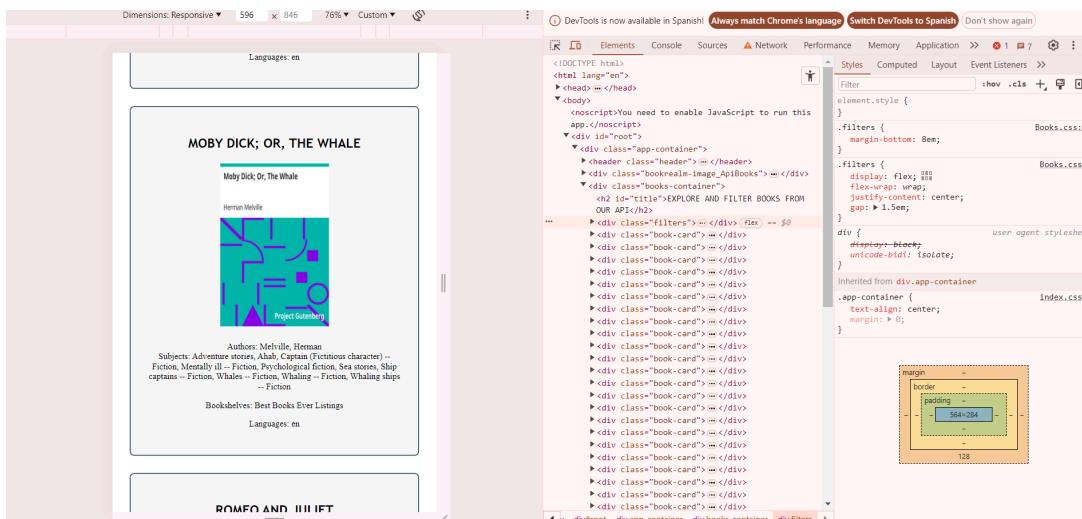
.AboutUsSection__Team--person {
  background-color: rgba(242, 232, 207, 0.6);
  font-family: "Trebuchet MS", "Lucida Sans Unicode"
  border-radius: 1.5em;
  margin-bottom: 3em;
  padding: 2em;
  display: grid;
  grid-template-columns: auto 1fr;
  gap: 20px;
  width: 80%;
  margin-left: 2em;
  margin-right: 2em;
}
```

A lo largo del proceso de implementación del CSS a nuestra página nos hemos apoyado de tanto **FLEX** como **GRID**, ya que es un formato de implementación en el que, en nuestro caso, era muy utilizado para disponer las tarjetas tanto de la zona del Blog, como la de ApiBooks e incluso páginas estáticas para disponer elementos en filas o columnas. A continuación se muestran ejemplos de su uso:

(en la cuadrícula de Blog se puede apreciar el uso de GRID)



(En la zona de libros obtenidos de la API, se representan con un FLEX en fila)



En definitiva, las propiedades FLEX y GRID han estado presentes a lo largo del diseño para disponer ciertas partes de una forma u otra (filas, columnas, cuadrículas...).

No hemos hecho ningún uso de **frameworks ni librerías CSS**. Todo el diseño ha sido implementado desde cero por nuestra cuenta, de tal forma que pudiésemos ajustarlo de la forma que más nos gustase. Las **variables CSS más utilizadas**, sin ningún tipo de duda han sido aquellas que nos permitían cambiar tamaños, bordes y colores (margin, padding, color, background-color, border, border-radius, display, width, height, text-align, font-size, font-style...). La propiedad de ::hover también ha sido implementada una gran cantidad de veces, especialmente para jugar con los tamaños al pasar el ratón sobre un botón o link (scale: 1.1).

Como último punto y más importante a destacar, hemos hecho un **diseño responsive** y con el uso de **medias queries** en todas y cada una de las páginas. De esta forma, cualquier **dispositivo móvil o desktop** tendrá nuestra página adaptada y se podrá ver bien al hacer la página más pequeña o minimizar.

Ejemplo de una de las páginas de nuestra web en distinto formato y con el código elaborado en las medias queries:

Sección Information: Móvil VS Desktop

Desktop: aside a un lado y las tarjetas ocupando la zona izquierda de la pantalla

Móvil: aside en la zona baja, las tarjetas ocupan la zona superior organizadas en fila



Ejemplo del código con la media query para cambiar el formato a móvil:

Information.css (no media query) VS Information.css (media query)

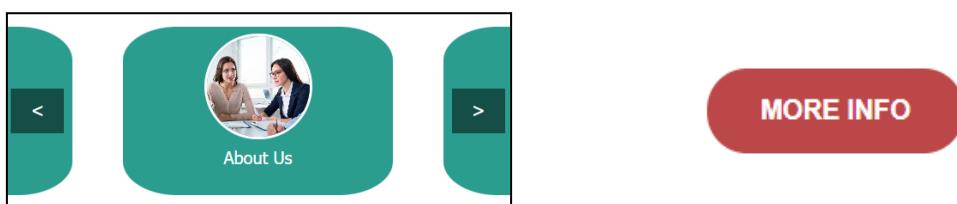
```
.aside-information h2{...}  
}  
  
.aside-information ul li {...  
}  
  
.aside-information ul li a {...  
}  
.aside-information ul li: hover {...  
}  
  
.information-card {...  
}  
  
.information-card: hover {...  
}  
  
.information-card a {...  
}  
  
.start { ...  
}  
  
@media only screen and (max-width: 1200px) {  
    .button-container { ...  
    }  
  
.information-card { ...  
}  
  
.information-container { ...  
}  
  
.aside-information { ...  
}  
  
.start { ...  
}  
.start: hover { ...  
}  
}
```

7. FUNCIONALIDAD

Nuestra página fue creada con el objetivo de ofrecer un contenido atractivo para el usuario y que recurriera a ella incluso cuando no la necesitaba. Para ello hemos hecho especial énfasis en la funcionalidad y la experiencia del usuario.

El objetivo era crear una plataforma intuitiva que, apoyándonos en aspectos como la usabilidad y la accesibilidad lograra la funcionalidad deseada.

Entre estos aspectos encontramos los elementos que hemos implementado en la interfaz de usuario y que facilitan las interacciones con el sitio web. Es el caso del carrusel de la página principal o botones como el del endpoint /information que redirige a la página /books directamente, orientando al usuario al llegar al final de la página /information y cumpliendo el objetivo funcional de la nuestro sitio web que no es otro que que el usuario llegue a filtrar por todos los libros.



De igual manera, ampliamos la funcionalidad de la página mediante el uso de formularios pues no solo ayudan a difundir contenido sobre la misma sino que aporta dinamismo y otro elemento con el que el usuario puede interactuar en cualquier momento.

Enter your email...

Submit

También tuvimos en cuenta la funcionalidad en el diseño de los endpoints, asegurando que las páginas no solo sean intuitivas sino también útiles. Es el caso de /Information, donde, a partir de una API aparentemente plana, sin objetos destacados por encima de los demás, generamos recomendaciones al usuario y tratamos de despertar un interés en nuestra página aunque no tenga claro que viene buscando. Por ejemplo, en el caso en el que alguien entre y no sepa ni que quiere buscar le ofreceríamos que filtre por los siguientes tópicos:



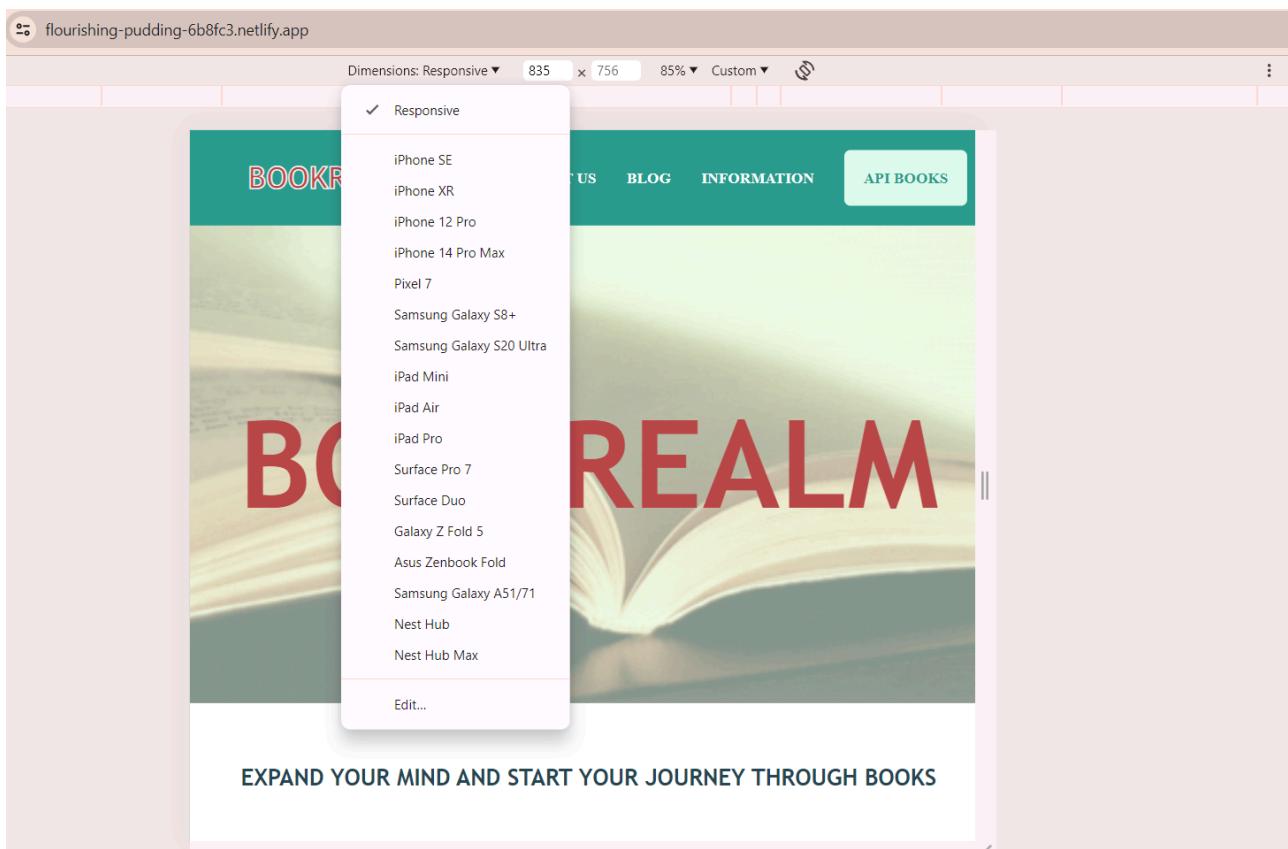
8. MOBILE FIRST | RESPONSIVE DESIGN

Tal y como se explica en la sección de prototipado (2) y la de CSS (6), vemos cómo en la fase de diseño hicimos uno básico para lo que era un dispositivo móvil. Sin embargo, nos resultó más sencillo comenzar programando el formato desktop y más tarde añadir medias queries para lo que era el caso de un móvil/tablet/dispositivo más pequeño.

De esta forma, al acceder a nuestra página:

<https://bookrealm2024.netlify.app/>

Se puede observar cómo está **adaptada a todos los dispositivos** y sigue un **diseño responsive**. Esto se consigue pulsando la tecla F12 (o haciendo click con el botón izquierdo → inspeccionar) y seleccionando en la pestaña superior cualquier dispositivo, o simplemente haciendo la página más pequeña desde la esquina o modificando:



(al seleccionar cualquiera de los dispositivos o simplemente eligiendo el formato Responsive y probando, se verá nuestra web para móvil, tablet, ordenador manteniendo una estética acorde al tamaño)

9. IMPLEMENTACIÓN MEDIANTE REACT

Todo el diseño de nuestro código se ha basado en endpoint con cada uno de sus componentes. Hemos organizado el código por carpetas, y una de ellas contenía todos los componentes. Dentro de los componentes hemos querido separar los artículos de los blogs para refactorizar y hacer más legible el proyecto.

Cada una de las secciones que componen un endpoint se ha dividido en varios componentes con otros componentes dentro. Este es el caso del endpoint books en el que se ha creado el componente filters para cada uno de los inputs de filtrado, y del componente Bookcard para mostrar cada una de las tarjetas con la información que obtenemos de la API con los respectivos parámetros del usuario.

Para las dos secciones del proyecto relacionadas con la API, el uso de herramientas como **useState** y **useEffect** ha sido fundamental. En el caso de la sección de Information, se obtienen cada una de las categorías y se muestran en el momento en el que categories obtiene los datos de la API a través del useState.

Mientras se realiza la petición, la variable booleana isLoaded es falsa, y de esta manera, podemos mostrar mensajes de tipo suspense para avisar al usuario de que se está cargando la url. En el momento que se obtienen los datos, isLoaded pasa a valer true, y el array de categories pasa a contener cada una de las categorías. Como isLoaded ya no vale false, se deja de mostrar "loading".

En el caso del **use Effect**, se utiliza para la función asíncrona que realiza la petición a la API. En el momento en el que se obtienen los datos, se cachean para que sea más rápido acceder a la sección en futuras ocasiones, y se utilizan los **us States** explicados anteriormente para seguir el transcurso del endpoint.

Esta estructura está de forma similar en el endpoint books con el que esta vez utilizamos un callback para la función asíncrona. El resto de variables condicionales y muestra de elementos en función de los valores que toman, es exactamente igual.

Por último, hemos hecho uso de react **router** en el archivo App.js para mostrar crear en el router cada una de las rutas que conforman nuestra API.

```
const information = () => {
  const [categories, setCategories] = useState([]);
  const [isLoaded, setIsLoaded] = useState(false);
  const [error, setError] = useState(null);

  useEffect(() => {
    async function fetchBooks() {
      const cachedData = localStorage.getItem("bookData");
      if (cachedData) {
        setCategories(JSON.parse(cachedData));
        setIsLoaded(true);
      } else {
        Helen, 6 days ago • Añadido header de FilterBy
        try {
          const response = await fetch("https://gutendex.com/books/");
          if (!response.ok) {
            throw new Error(`HTTP error! Status: ${response.status}`);
          }
        
```

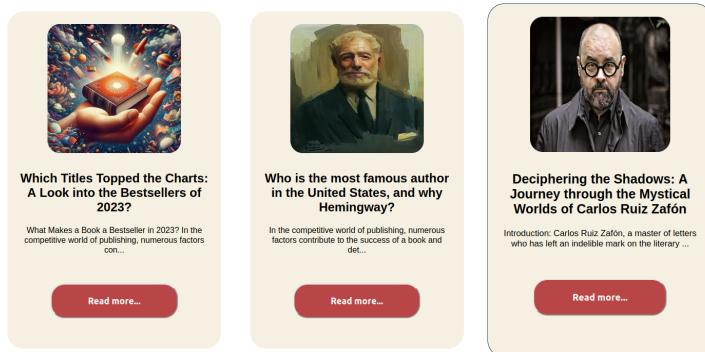
```
        return (
          <Router>
            <main className="app-container" role="main">
              <Header />
              <Routes>
                <Route
                  path="/"
                  element={
                    <>
                      <HeroPrincipal />
                      <MainContent />
                    </>
                  }
                </Route>
                <Route
                  path="/about"
                  element={


```

10. USABILIDAD

En cuanto a la usabilidad, desde la **fase de investigación**, teníamos claro que el público principal al que estaba dirigida nuestra web eran personas adultas. Es por eso que uno de los primeros aspectos que teníamos claros era que íbamos a implementar una paleta de colores que fuera estética a la vista, pero que también mantuviese una correcta usabilidad para aquellos usuarios que disponían de una discapacidad física (especialmente en nuestra Web, enfocada a personas adultas). Además, en clase realizamos una técnica de análisis de usabilidad cuando nos fuimos mezclando por distintos grupos a hacer **entrevistas a usuarios**/compañeros sobre nuestro primer diseño de la página web.

Como el objetivo principal era que la página fuera entendible (**understandable**: uno de los cuatro principios **POUR**), hemos hecho uso de **cards** que contaban con imágenes, texto de gran tamaño y color de fondo (en algunas incluso bordes de un color oscuro) para poder resaltar lo máximo posible cada elemento.



En la **fase de diseño**, realizamos una serie de pruebas de usabilidad con un par de compañeros del grupo kebab para **evaluar la interfaz** del usuario y posibles mejoras. Como posibles puntos de mejoras nos recalcaron que el diseño tenía que tener distinciones entre móvil y ordenador, debido a que secciones extensas en letra como el about us o los artículos del blog eran más difíciles de implementar en móvil. Además de ello, se pusieron tamaños más grandes en letra y botones.

Como otro apartado de usabilidad, hemos añadido un **carrusel** en la landing page para hacerla más potente y que se pueda usar mejor. Usamos **contenedores** a lo largo de todo el proyecto. Se puede ver en los artículos, en los resultados de la API...

Como **ayuda al usuario**, hemos dado **avisos** siempre en los casos de error o búsqueda, se dé un aviso al usuario en todo momento. Se puede ver que el error se muestra de color rojo, y cuando no se muestran resultados con esos filtros.

Además, como la API tarda en devolver los resultados, mostramos un mensaje de que se están cargando los datos.

EXPLORE AND FILTER BOOKS FROM OUR API

Search by title Search by topic Choose a language Non available info ↴

0 -10 Search

THE SECOND DATE MUST BE BIGGER THAN THE FIRST ONE

NO BOOKS FOUND WITH THE SPECIFIED FILTERS. PLEASE TRY DIFFERENT FILTERS.

EXPLORE AND FILTER BOOKS FROM OUR API

Search by title Search by topic English Choose copyright ↴

Author Year Start Author Year End Search

LOADING...

Como otro apartado de usabilidad, hemos añadido un **carrusel** en la landing page para hacerla más potente y que se pueda usar mejor. Usamos **contenedores** a lo largo de todo el proyecto. Se puede ver en los artículos, en los resultados de la API...

Por otro lado, en la fase de prototipado vimos el tema de **contrastos** del color de texto con el fondo que pretendíamos poner y, empleando la plataforma de [Contrast Finder](#) vimos que había un contraste adecuado, lo cual nos permitió comenzar la **fase de diseño** con una paleta altamente contrastable y un diseño entendible para el público con mayor dificultad

Contrast Finder

Contrast-Finder encuentra los contrastes adecuados de color para crear webs accesibles

Los criterios [WCAG](#) de éxito 1.4.3 requieren que el texto tenga un contraste mínimo de 4.5 (3 para textos largos).

Color del texto: #496169

Color del fondo: #FFFFFF

Color en hexadecimal, RGB o palabras clave de color CSS
Por ejemplo: #FFFFFF, rgb(255,255,255) o white

Ratio mínimo: 4.5 ✓ 6.57

Contrast Finder

Contrast-Finder encuentra los contrastes adecuados de color para crear webs accesibles

Los criterios [WCAG](#) de éxito 1.4.3 requieren que el texto tenga un contraste mínimo de 4.5 (3 para textos largos).

Color del texto: #bc4749

Color del fondo: #FFFFFF

Color en hexadecimal, RGB o palabras clave de color CSS
Por ejemplo: #FFFFFF, rgb(255,255,255) o white

Ratio mínimo: 4.5 ✓ 4.96

Por último, en la **fase de validación**, decidimos implementar un **A/B testing**, comprobando un estilo frente a otro ligeramente distinto y viendo cuál podía obtener mejores resultados para nuestra web BookRealm. Además de ello, hemos utilizado **ChromeVox** como se explicará más adelante en el apartado de accesibilidad.

Como conclusión, podemos decir que hemos intentado que la Web sea lo más usable posible y tenga buenos resultados siempre siguiendo un **diseño responsive**, con el fin de llegar a una gran cantidad de usuarios ya tengan más o menos dificultades visuales.

11. ACCESIBILIDAD

Una de las cosas que más hemos tenido en cuenta en nuestra página web es la accesibilidad. Para ello nos hemos centrado en el principio **WCAG** de que sea **comprendible**. A continuación se va a explicar cada uno de los puntos implementados.

1. **Botones y letras grandes** para que la web sea apta para todo tipo de usuarios.
2. **Colores claros y enfocados**
3. **Animación** en el libro de los headers de Book Realm
4. **Botón** en home que **redirige** a la sección **API** y **menú de navegación**
5. **Información** y componentes reconocible a **simple vista**
6. Fácil de **interpretar**. Los componentes tienen **nombres descriptivos** y los filtros de la API usan el atributo **placeholder** para explicar cada uno de ellos. Además se muestran los **select** con una opción predeterminada que es "**Choose ...**".

The top screenshot shows a search interface with the following fields: Search by title, Search by topic, English, Choose copyright, Author Year Start, Author Year End, and a Search button. The bottom screenshot shows the 'INFORMATION' page for 'FRANKENSTEIN; OR, THE MODERN PROMETHEUS' by Mary Shelley. It displays the book cover, authors (Mary Shelley), subjects (Gothic fiction, movie books, etc.), bookshelves (Books by Women), and languages (en). The page also lists the 'Top Books of 2024'.

6. Atributos **alt** en imágenes en el caso de que falle la carga
7. **Lazy loading** en /information. En este caso, hacemos una petición al endpoint general /books. Como tarda en realizarse la petición, hasta que no se cargue todo el contenido no se mostrará nada.
8. Etiqueta **label** y atributo autocomplete en el input del formulario de contact-session
9. Atributos **ARIA** polite (cambios en secciones), aria-labelledby y **roles** como main o alert.

```
import Blog from "./endpoints/Blog";
import Article from "./endpoints/Article";
import Books from "./endpoints/Books";

const Information = lazy(() => import("./endpoints/Information"));
const App = () => {
  return (
    <Router>
      <main className="app-container" role="main">
```

```
<form method="POST" onSubmit={handleSubmit} aria-label="Contact Form">
  <label htmlFor="email" className="visually-hidden">
    Email:
  </label>
  <input
    id="email"
    type="email"
    placeholder="Enter your email..."
    value={email}
    onChange={(e) => setEmail(e.target.value)}
    required
    autoComplete="email"
    aria-required="true"
  />
  <button type="submit">Submit</button>
</form>
```

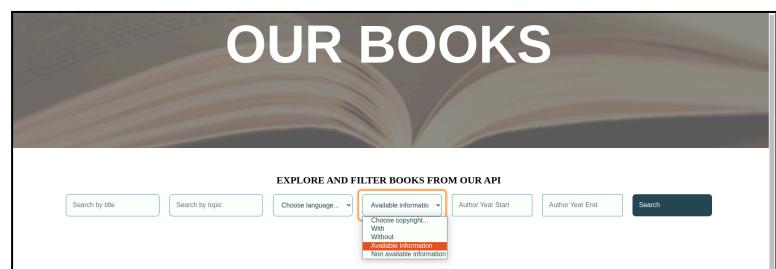
10. Uso de **etiquetas semánticas** correctas (nav, footer, header, section, aside...)
11. **Atajos** como botones o el carrusel
12. **Line-height, letter-spacing y word-spacing** en los artículos de los blogs
13. **Elementos accesibles** con el teclado
14. Etiquetas **strong** para dar énfasis y **abbr** para abreviaciones

```

|   pacing are key factors that m
</p>
<strong>
|   <abbr>FAQ</abbr>
</strong>
<p>
|   0: What is the significance o

```

Para evaluar de distintas maneras la accesibilidad, hemos optado por utilizar una serie de **auditorías**. En el vídeo de presentación se puede ver la **auditoría de accesibilidad automatizada lightbox** explicada por Guillermo Azcona. Además de ello, hemos utilizado el **software narrador de texto ChromeVox** que va mostrándote la interacción entre el usuario y la página web, ya sea con audio, o con la muestra de la zona en la que se está centrando.



Como conclusión, el aspecto que más hemos tenido en cuenta es la experiencia del usuario para que sea comprensible cada uno de los apartados de la página web. Para ello hemos tenido en cuenta los mensajes de aviso al usuario en cada uno de los escenarios. Otro aspecto muy importante ha sido el SEO, y para ello hemos usado las etiquetas demandadas en cada sección del código para hacerlo más accesible. Estos dos aspectos están íntimamente ligados ya que el SEO depende en gran medida de la accesibilidad.

Como posible punto de mejora se podría haber cambiado los colores para que el menú de navegación tenga un contraste mayor y así mejorar el aspecto de la accesibilidad. Este problema se resalta al hacer un análisis con **lightful** en google chrome.

12. SEO

En el desarrollo de nuestro proyecto web, hemos priorizado la accesibilidad ha sido un pilar fundamental, no solo por su importancia en la inclusión y la ética digital, sino también porque mejora el posicionamiento SEO en motores de búsqueda como Google. Los sitios accesibles suelen recibir una mejor evaluación por parte de los algoritmos de búsqueda debido a su estructura clara y navegabilidad.

Para asegurar una alta accesibilidad, hemos implementado una estructura semántica robusta en nuestro sitio web. Esto implica el uso de etiquetas HTML adecuadas que definen claramente las partes de la página web, como `<nav>`, `<header>`, `<section>`, `<article>`, `<aside>`, `<meta>`, y `<footer>`. Esta estructura no solo facilita la navegación por parte de los usuarios, sino que también optimiza la interpretación del sitio por los crawlers de los motores de búsqueda, mejorando así nuestro SEO.

Una sección de gran importancia en nuestro sitio web es el **blog**, que nos permite atraer tráfico a través de artículos que incorporan palabras clave altamente buscadas relacionadas con nuestro nicho. Además, hemos añadido un archivo **robots.txt** en el que enfatizamos que los robots de búsqueda se centren en el endpoint /blog. Dentro del blog, hemos desarrollado contenidos como un artículo sobre Carlos Ruiz Zafón y sus obras más significativas, así como sobre Ernest Hemingway y los libros más vendidos en 2023. Utilizamos herramientas como **Ahrefs** para identificar estas palabras clave, evaluando su volumen de búsqueda anual y la dificultad de ranking, lo que nos permite seleccionar términos que maximicen nuestra visibilidad sin enfrentar una competencia desmesurada.

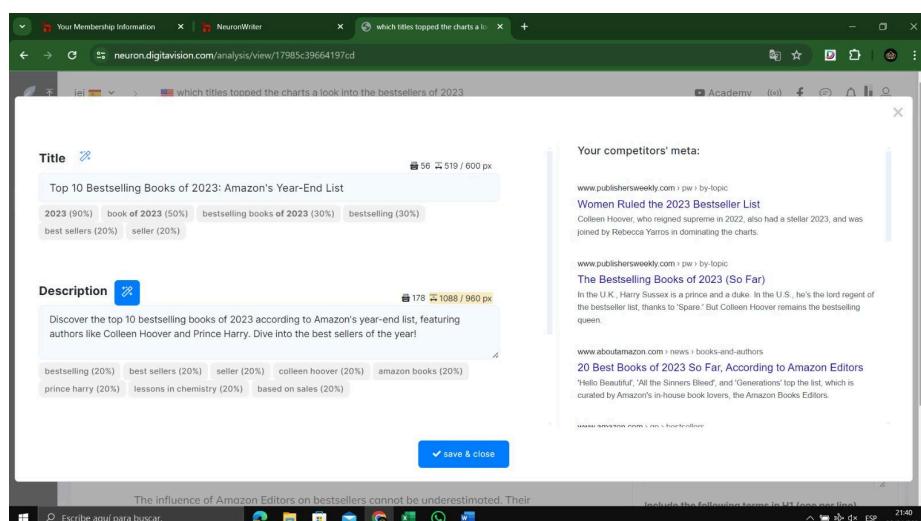
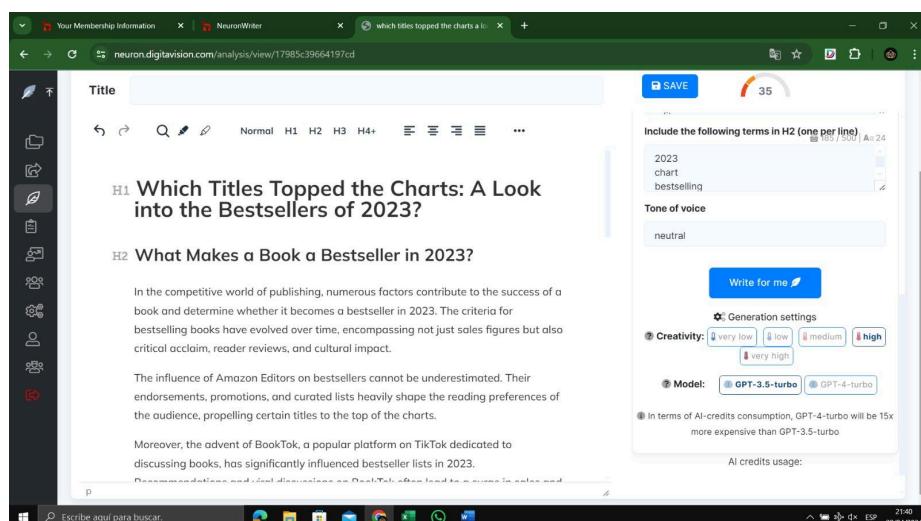
Concordancia de frase / Preguntas			
Palabra clave	KD	Volumen ↓	Actualizadas
ernest hemingway	Hard	>100K	about 4 hours
ernest hemingway books	Easy	>1000	about 17 hours
books by ernest hemingway	Hard	>1000	about 4 hours
ernest hemingway death	Medium	>1000	about 2 hours
ernest hemingway house	Hard	>1000	1 day
ernest hemingway spouse	Easy	>1000	about 6 hours
ernest hemingway quotes	Easy	>1000	about 7 hours
how did ernest hemingway die	Medium	>1000	3 days
ernest hemingway house photos	Medium	>1000	about 24 hours

Concordancia de frase / Preguntas			
Palabra clave	KD	Volumen ↓	Actualizadas
the housemaid	Easy	>10,000	about 2 hours
the housemaid book	Easy	>10,000	about 11 hours
the housemaid series	Easy	>1000	about 3 hours
the housemaid freida mcfadden	Easy	>1000	about 1 hour
synopsis of the housemaid	Easy	>1000	5 days
the housemaid is watching	Easy	>1000	about 24 hours
the housemaid summary	Easy	>1000	5 days
the housemaid book 2	Easy	>1000	1 day
the housemaid reviews	Easy	>1000	2 days

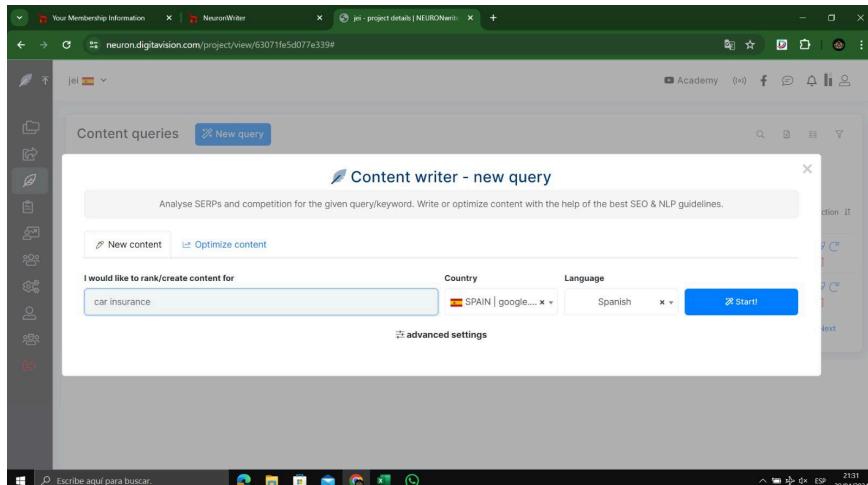
Como se puede observar en ambas imágenes, se está utilizando el buscador de Ahrefs. En ella introduces la palabra o grupo de palabras, y te devuelve un grupo de ideas de palabras clave con su dificultad y su volumen de visitas anuales.

Para la redacción de contenidos del blog, hemos incorporado herramientas como Neuron Writer, que nos ayuda a estructurar los artículos de manera que se optimice el uso de términos clave para SEO. Esta herramienta propone estructuras y términos basados en la temática y las palabras clave proporcionadas, facilitando la creación de contenidos que no solo son ricos en información, sino también efectivos para el posicionamiento en buscadores.

Para utilizar Neuron Writer, es necesario escoger una temática, con unas palabras clave. Además, hemos tenido que añadir un título del artículo sumado a una descripción detallada del contenido del artículo. Por último, hemos escogido un tono de voz neutral y una serie de competidores al que se va a “enfrentar” nuestra página web. Nuestro principal competidor será Amazon debido a la cantidad de visitas que adquiere por sus ventas de libros online.



Como se puede observar en las imágenes, se escoge una temática, un título y una descripción. En segundo lugar, se eligen los competidores, y, por último, se escogen los filtros para la escritura del artículo con los términos clave.



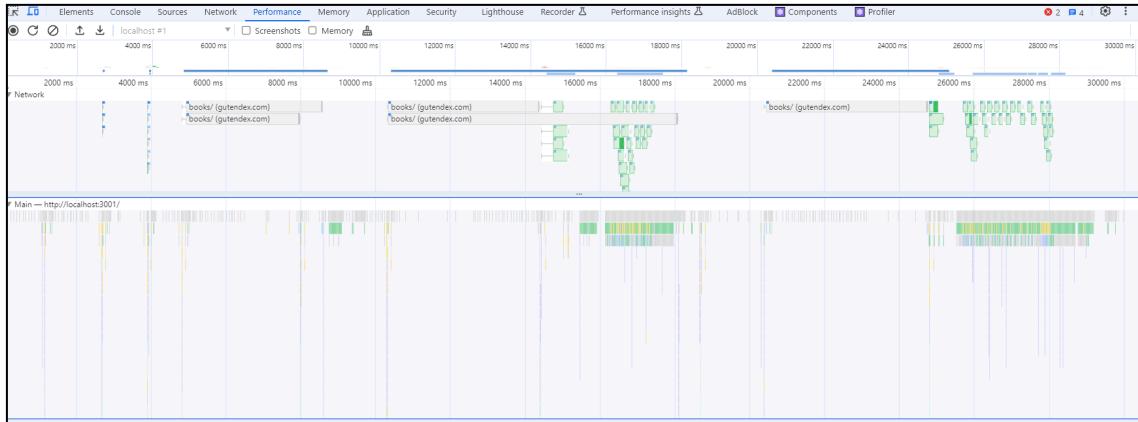
Select	Google Rank	Content Score	Length (Words)	Page Title	URL	Competitors' List
<input checked="" type="checkbox"/>	1	47	878	Women Ruled the 2023 Bestseller...	publishersweekly.com/pw/by-topic...	View
<input checked="" type="checkbox"/>	2	50	947	The Bestselling Books of 2023 (So Far)	publishersweekly.com/pw/by-topic...	View
<input checked="" type="checkbox"/>	3	35	1604	20 Best Books of 2023 So Far, Acco...	aboutamazon.com/news/books-an...	View
<input checked="" type="checkbox"/>	4	39	1053	Amazon.com Best Sellers of 2023 in...	amazon.com/gp/bestsellers/2023/...	View
<input checked="" type="checkbox"/>	5	54	1015	Amazon reveals its bestselling book...	redonline.co.uk/entertainment/ba...	View
<input checked="" type="checkbox"/>	6	19	2425	Best Sellers - Books	nytimes.com/books/best-sellers/	View
<input checked="" type="checkbox"/>	7	24	1614	Billboard releases the 2023 Hot 10...	reddit.com/r/popheads/comments/...	View
<input checked="" type="checkbox"/>	8	47	1998	Apple unveils the top books of 2023...	apple.com/newsroom/2023/11/0...	View
<input checked="" type="checkbox"/>	9	54	1156	Amazon's bestselling books of 2023	aboutamazon.co.uk/amazon-best...	View

La integración de estrategias de SEO basadas en la accesibilidad y la utilización de palabras clave es fundamental para la web. Al garantizar que nuestro sitio sea accesible y esté bien posicionado, no solo mejoramos nuestra visibilidad online, sino que también ofrecemos una experiencia de usuario superior.

Como posible punto de mejora se podría modificar el endpoint para utilizar un nombre más descriptivo que no sea /Information y así poder posicionarse mejor. En su lugar, el nombre podría llamarse "top books" ya que el endpoint muestra los libros más buscados del momento.

13. RENDIMIENTO

El rendimiento lo hemos evaluado de dos maneras. En primer lugar hemos realizado una prueba de rendimiento con la **herramienta Performance de Google Chrome**:

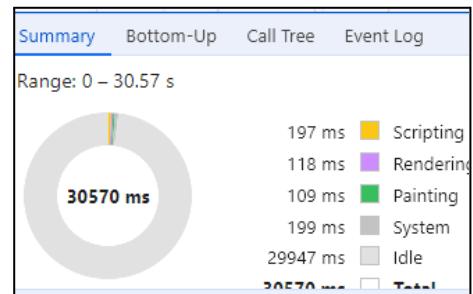


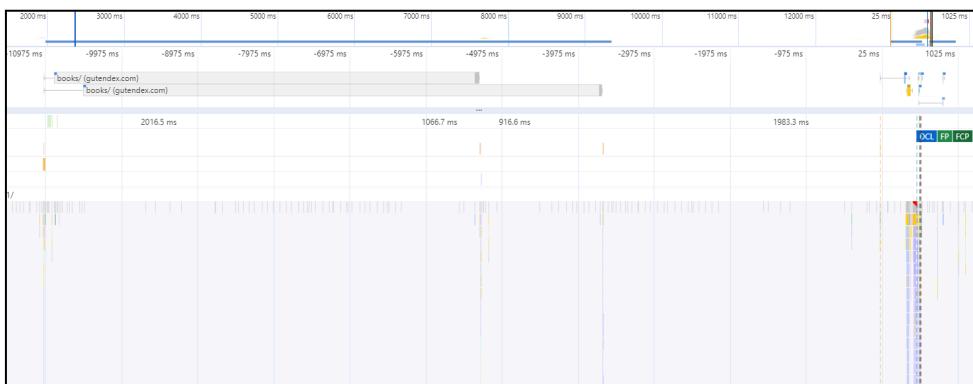
Hasta los 5000ms hemos realizado las peticiones a nuestra landing page y a /blog, por ello no observamos prácticamente alterado el diagrama. Sin embargo al acceder a /Information se realiza una primera petición a gutendex.com/books que dura alrededor de 2000ms. Más tarde, al acceder a uno de los autores filtrados en /Information, obtenemos el mayor tiempo de carga que registramos. En este caso tarda unos 7000ms en hacer la petición a la API con el filtro deseado. Por último, accedemos a la página /books, donde realizamos un filtrado que no lleva en torno a 2000ms otra vez.

En resumen, el diagrama mostrado nos señala la cantidad de tiempo que el navegador está dedicando a cada aspecto en estos 30s que he estado interactuando con la web. Como vemos, la mayor parte del tiempo está siendo ocupado por períodos de inactividad (Idle), lo que indica que el navegador está esperando recursos o respuestas de API. Aunque se dedica un tiempo considerable a operaciones de scripting, rendering y painting, es crucial analizar y reducir estos tiempos de espera. Optimizando la carga de recursos externos y mejorando la eficiencia de los scripts, podríamos significativamente acelerar la interacción general con la página y mejorar la experiencia del usuario.

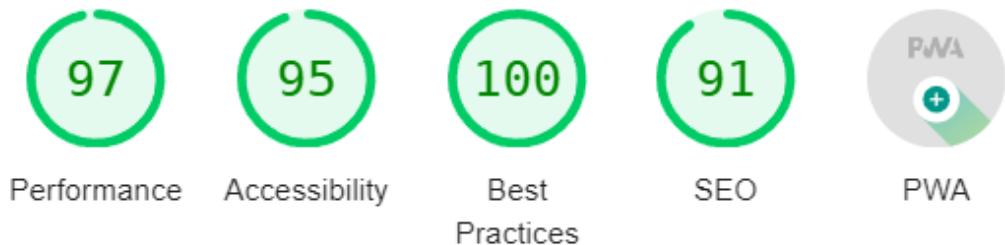
Por último y como mejora al rendimiento, hemos guardado en la caché del navegador la petición que realizábamos en /information. Con esto mejoramos el tiempo de carga y la experiencia del usuario.

En esta imagen podemos apreciar este cambio, donde en el primer caso se realiza el acceso al endpoint con la cachá vacía y en el segundo caso (12000ms), se accede con la petición de la API guardada, y se carga todo a la vez.



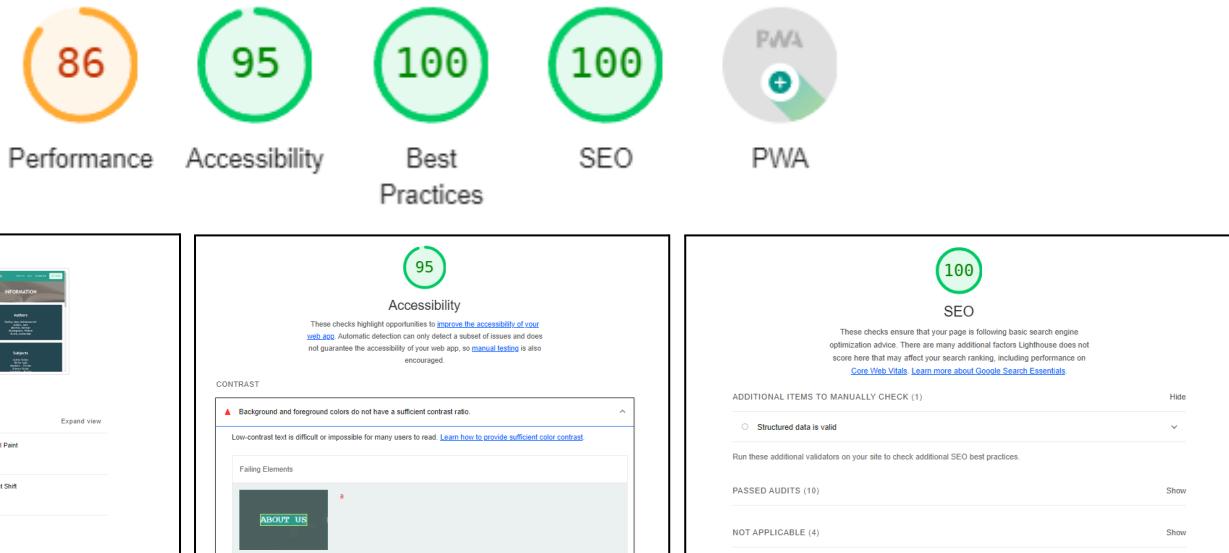


También hemos realizado un reporte en Lightroom donde hemos obtenido muy buenos resultados. En primer lugar se ha evaluado desde la landing page y los resultados son los siguientes:



Las puntuaciones obtenidas son positivas, aunque quizás no reflejan completamente la realidad. Tras evaluar diversos ajustes en el contraste, la paleta de colores y el nombre del endpoint `/information` (que recibió una penalización por ser poco descriptivo), creemos que estas elecciones son las más adecuadas para guiar y proporcionar una descripción clara del contenido a nuestro público objetivo, que es mayoritariamente adulto.

De igual manera realizamos una prueba de lighthouse con el endpoint /information, ya que realiza una petición directa a la API de gutendex y no se ha evaluado en el anterior Lighthouse.



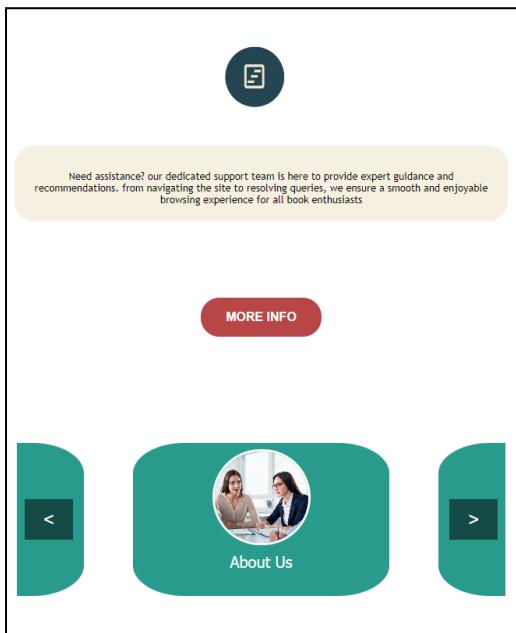
Como vemos, los resultados obtenidos son bastante similares, exceptuando la puntuación reducida en rendimiento debido a la carga de todos los datos. Este rendimiento mejorará en futuras peticiones a este endpoint, dado que los datos se almacenarán en la caché del navegador. Además, hemos observado una mejora en el SEO, ya que el apartado contiene palabras clave y enlaces a otras páginas que incrementan nuestra valoración general en este aspecto.

En resumen, teniendo en cuenta que nuestra página web se basa en peticiones a una API con cientos de libros, hemos conseguido obtener muy buen rendimiento al optimizar las llamadas y mejorar la gestión de la caché, lo que ha resultado en tiempos de carga más rápidos y una experiencia de usuario más fluida.

Otro aspecto a evaluar fue la accesibilidad, que a través del **guión de Marcela Vega Higuera auditamos**. En primer lugar revisamos la presentación y estructura general, donde vimos que tanto como para nuestro público objetivo como para posibles perfiles con problemas visuales o intelectuales, cuenta con un orden y estructura clara, contando con un diseño universal para todos los públicos, simple, intuitivo y con tolerancia al error, pues siempre están visibles los botones de redirección a otros endpoints o al mismo.



Asimismo el diseño tiene espacio para que se pueda utilizar correctamente, cada sección tiene su espacio y están claramente diferenciados, favoreciendo la comprensión.



Además, esta aplicación ha sido **testada por gente de nuestro entorno**, lo que nos ha ayudado a mejorar la accesibilidad. Desde el principio, los usuarios reciben una descripción clara de lo que nuestra página web ofrece y los endpoints a los que pueden acceder. Esto ha permitido que nuestros testadores, quienes se encuentran en un rango de edad de 50 a 75 años, hayan podido comprender y utilizar la aplicación satisfactoriamente, demostrando su facilidad de uso y accesibilidad.

Por todo ello, podemos acreditar que, basándonos en muchos de los principios mencionados por Marcela Vega Higuera, nuestra página web es accesible para todos los públicos y nuestro compromiso con la accesibilidad universal se refleja en cada aspecto del diseño y funcionamiento del sitio, asegurando que todos los usuarios, sin importar su edad o capacidades, puedan beneficiarse de nuestros servicios.

14. DESPLIEGUE

Para el despliegue hemos utilizado [Netlify](#). Únicamente se necesita registrarse y hacer un npm run build para construir la carpeta y subirla a Drag&Drop. Además se puede configurar la web para usar una personalizada. En nuestro caso la hemos configurado a bookrealm.netlify.com. El enlace para acceder a nuestra página web es esta:

<https://bookrealm2024.netlify.app/>

15. GESTIÓN DE CONFIGURACIÓN (GitLab)

En nuestro proyecto, se implementó una estrategia de gestión de ramas utilizando **GitLab**, donde cada integrante del grupo trabajó en una rama dedicada. Esto incluyó las siguientes ramas:

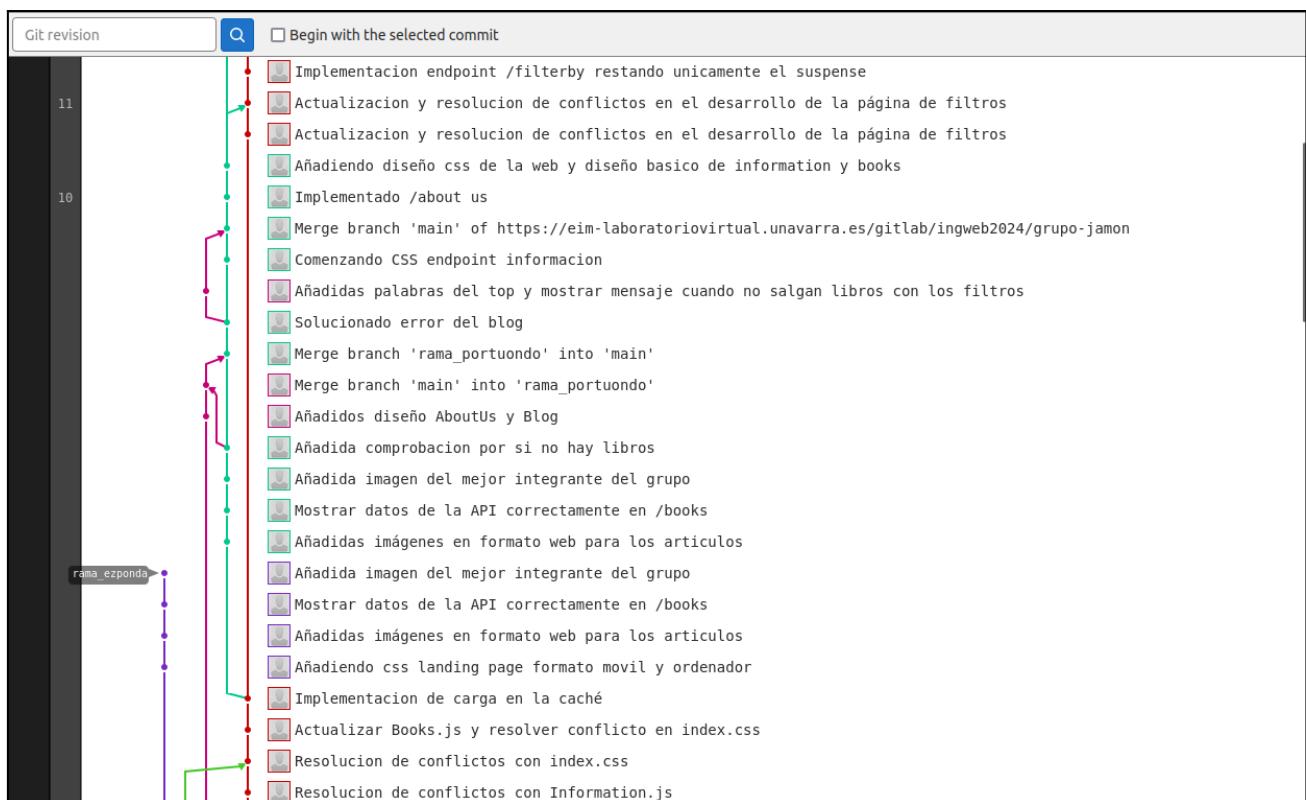
- rama_portuondo
- rama_azcona
- rama_ezponda

Cada miembro del equipo realizó sus propios commits en su respectiva rama, lo que permitió una colaboración eficiente y un desarrollo paralelo sin interferencias directas en el trabajo de los demás.

Para integrar los cambios realizados en las ramas individuales a la rama principal main, utilizamos el sistema de **Pull Requests** de GitLab. Esto no solo facilitó la revisión del código por parte de otros miembros del equipo, sino que también aseguró que todo el código integrado cumpliera con los estándares de calidad acordados antes de su fusión.

Por otro lado, también hubo que arreglar **conflictos** en ciertos casos en los que se intentaba subir el trabajo modificado desde las ramas personales a la rama main y las versiones no cuadraban, por lo que hubo que solucionarlos.

A continuación, se muestra una imagen de la gráfica de commits, en la que se puede ver de forma más visual este proceso:



Para una mejor organización y seguimiento del proyecto, se crearon **historias de usuario e issues** en GitLab. Cada issue estaba vinculada a una historia de usuario específica y se asignaron responsables (**assignees**) y tiempos estimados para la realización de cada tarea. Esto proporcionó una visión clara del progreso y la carga de trabajo de cada miembro del equipo. A continuación, se adjunta una imagen del Board de nuestro repositorio:

<https://eim-laboratoriovirtual.unavarra.es/gitlab/ingweb2024/grupo-jamon>

To-Do	Doing	Closed
HU #1 Añadir redirección de los filtrados por sección en Figma #48	Como desarrollador quiero programar el proyecto utilizando la herramienta de React #20	HU #20 Implementar círculo de carga en el suspend de Information.js #47
HU #20 Añadir etiquetas "meta" al apartado HTML #10 X 30m	Como desarrollador quiero escribir una memoria para explicar el desarrollo del trabajo #17	HU #20 Añadir suspense al endpoint de information mientras carga los datos #45
HU #20 Añadir etiquetas de accesibilidad al código #42	HU #1 Actualizar prototipo #56 X 2h	HU #20 Programar redirección del filtrado de /information y botón a filtrado total #51 X 2h
HU #20 Automatizar el contacto del email de la landing page con emalloctopus #40 X 1h 30m	HU #20 Implementar cachear en /information #55 X 20m	HU #20 Implementar endpoint /aboutus #54 X 1h
Como desarrollador quiero informarme de cómo funciona emalloctopus #41 X 1h	Como desarrollador quiero desplegar mi proyecto en un Servidor web #30	HU #20 Añadir acceso directo a la página principal desde el menú en formato móvil #44
		HU #20 Añadir links a cada palabra clave de /information #53 X 10m

Se crearon dos etiquetas principales para gestionar el estado de las issues:

- **To-Do:** Para tareas que estaban planeadas pero aún no iniciadas.
- **Doing:** Para tareas que estaban activamente en proceso.

El flujo de trabajo de las issues se movía de **Open** (abierto) a **To-Do**, luego a **Doing**, y finalmente a **Closed** (cerrado) una vez que la tarea se completaba. Este método proporcionó una **estructura clara** para el seguimiento de tareas y ayudó a visualizar el progreso hacia los objetivos del proyecto.

16. RESUMEN Y CONCLUSIÓN FINAL

Podemos concluir diciendo que la realización de este trabajo en grupo nos ha servido para obtener una extensa cantidad de conocimientos acerca del proceso de creación de una web, desde el momento inicial en el que únicamente se tiene una idea y un prototipo en la cabeza, hasta su posterior implementación, diseño y validación final.

Hemos podido entender todos los aspectos que componen este desarrollo (incluso los detalles menos perceptibles para un usuario externo) y, además, trabajando en equipo y organizándonos de la forma más estructurada posible.

Por otro lado, hemos aprendido el uso de nuevas herramientas que han sido de gran utilidad. En resumen, pese a la longitud del proyecto y el trabajo por parte de todos los integrantes, hemos adquirido muchos conocimientos así como recursos útiles que seguiremos usando en un futuro.