Descargué las fuentes desde Google Fonts en formato .ttf y las transformé a través de <https://www.fontsquirrel.com/tools/webfont-generator> y <https://transfonter.org/> en formato .woff y .woff2, ya que son compatibles con la mayoría de navegadores y ofrecen un buen equilibrio entre calidad y tamaño de archivo. Finalmente, escogí .woff2 al tratarse de la versión más reciente y ofrecer una mejor comprensión con respecto a .woff. Los archivos .woff2 son más eficientes en términos de rendimiento, al ser más pequeños, pueden ayudar a reducir el tiempo de carga de la web. A partir de este punto, hice lo mismo que en la PEC anterior, crear una carpeta con el nombre de la fuente, y guardar los archivos .woff2 en la carpeta local, para garantizar que mi web siempre se pueda leer con estas letras y no corra el riesgo de falta de disponibilidad de la fuente en caso de importarla directamente desde la web, ya que si en el futuro esta fuente dejara de tener soporte o se eliminara del sitio original, mi web ya no mostraría esa letra. Posteriormente, en el CSS, a través de @font-face, importo y creo la letra para cada tamaño con los archivos correspondientes y ya puedo usarlas en todo el proyecto. Lo ideal sería tener también en el proyecto tanto el formato .woff como .ttf para garantizar mayor compatibilidad con más navegadores y con navegadores más antiguos.

Para saber cómo se vería en el móvil o Tablet, dentro del navegador, le doy click derecho e inspeccionar. Desde ahí, le doy a Activar/Desactivar barra de herramientas del dispositivo, y escojo las dimensiones del dispositivo que me interesan para poder ver cómo quedaría, directamente desde la pantalla del pc. He usado las dimensiones del iPhone 12 Pro (390x844), iPad Air (820x1180). En CSS apliqué @media (min-width: 390px){todo el contenido}, min-width: 820px, y min-width: 1000px para pc ya que es a partir de donde el título Vital360: Cuidando de ti en cada giro de la vida, empieza a salirse de la imagen.

En index.html para ubicar la imagen de fondo hero-home en versión móvil, he tenido que añadir padding top y bottom, así como jugar con background-size y background position hasta lograr el efecto parecido al que se pide. También, tuve que usar background-attachment:scroll para que la imagen se mueva junto al resto de la página cuando vamos bajando por el contenido.

Para crear el enlace vacío con aspecto de botón, he usado href=”#” en lugar de href=” “; no todos los navegadores pueden manejar el espacio en blanco de la misma manera, por lo que # es más correcto y ampliamente aceptado.

He usado las etiquetas <article> para crear secciones ya que al final son temas separados e independientes. He usado <span Lang=”en”> para añadir accesibilidad a la página y marcar cada cambio de idioma. He usado etiquetas <b> sin significado semántico, para marcar las palabras en negrita. Para convertir un email en enlace con protocolo mailto hice <a href="mailto:info@vital360.com">[info@vital360.com</a](mailto:info@vital360.com%3c/a)>.

Para ocultar el borde del formulario y el texto legend sin perder accesibilidad, he dispuesto el siguiente código en css: fieldset{ border: none; } legend{ opacity:0 }. Al parecer, si se oculta legend con display: none, dejaría de ser accesible para los programas lectores, por lo que opté por la opacidad.

En index, he creado dos div para separar la imagen y los créditos, y la descripción con el botón de “me interesa”, y he creado dos clases para la imagen y créditos, la normal y la invertida, a la hora de rotar el contenido en direcciones opuestas, y lo mismo hice para el contenedor padre del contenido de cada artículo, para poder aplicar la propiedad flex-direction: row-reverse a uno sí y a otro no, para que la imagen y créditos aparezcan del lado izquierdo y derecho respectivamente según cada artículo.

A tamaño pc, el aspecto cambia significativamente a la versión móvil y Tablet, por lo que las etiquetas y clases creadas inicialmente siguiendo el enfoque *mobile first*, tuvo que soportar modificaciones.

Primero se crearon los elementos para que reflejen el aspecto móvil, después la Tablet, y finalmente el pc. Pero una vez creado el modo pc, no hay que olvidar revisar las versiones previas de Tablet y móvil, para comprobar que los nuevos cambios no han afectado el aspecto original buscado en ambas versiones.

Ha sido complicado dividir el texto de introducción de Vital360 y el miniformulario para que aparezcan en dos columnas del mismo tamaño, teniendo en cuenta que antes iban en una sola columna y la distribución era diferente de los elementos, el tamaño de los inputs también ha variado, y de la política de privacidad, y el botón. Ha tocado identificar y jugar con los tamaños de las cajas y con display: flex. Añadí la propiedad gap para establecer misma separación. También, tuve que establecer esta regla .porQueVital360\_\_pc > div { width: 50%; } que afecta a los div hijos directos del div padre de la clase porQueVital360\_\_pc, para poder crear dos cajones de igual tamaño.

En el footer, también he tenido que dividir la lista en dos div y aplicar display flex, así como apliqué flex-start y end en los textos para que aparezcan como en la imagen.

He aplicado los efectos del :hover en la versión pc solamente, ya que es la única versión que teniendo un ratón, al pasar por encima, se podrían apreciar los cambios. Al aplicar estilos de :hover solo en la versión desktop mediante media queries, estoy optimizando el código para dispositivos móviles al evitar que el navegador procese estilos de :hover que no se aplicarán en esos dispositivos. Esto puede contribuir a una mejor eficiencia y rendimiento en dispositivos móviles al reducir la carga de trabajo innecesaria.