Diseños Responsivos (Parte I)

Bienvenidos a la experiencia

Para crear layouts en una maqueta es importante entender cómo trabajar con las distintas herramientas que tenemos a disposición en CSS para:

- dimensionar
- posicionar
- mover elementos

Estas herramientas junto con el correcto uso de unidades de medida, hará que nuestras traducciones representen fielmente la propuesta de los diseñadores UI.

Sin embargo, cuando la propuesta de diseño se realiza teniendo en mente un sólo dispositivo, la creación puede tornase difícil, especialmente si el cliente desea que su aplicación o sitio web sea visto en diferentes dispositivos a la vez.

Como sabemos crear un diseño conlleva por parte de diseñadores UX/UI una investigación previa de los comportamientos y necesidades de los usuarios, los que luego se transformarán en una representación visual y guía de estilos la cual recibiremos a modo de crear una maqueta.

En otras palabras, el crear diseños para otros dispositivos conlleva tiempo y capital humano que probablemente no tendrá nuestro equipo al momento de terminar un proyecto.

Para contextualizar este posible problema, desarrollaremos durante toda esta experiencia un ejercicio que contempla el siguiente problema:

Conociendo el proyecto

Luego de finalizar, nuestro Product Owner (jefe de proyecto) nos comenta que el cliente quedó muy conforme con el rediseño de su sitio, sin embargo, desea que el sitio creado sea responsivo, o sea, que este pueda ser visto en diferentes dispositivos.

Al escuchar esto preguntamos cuánto tiempo tenemos para realizar el trabajo y si el diseñador UX/UI que creo la representación visual creará el mockup de los nuevos dispositivos. La respuesta del Product Owner fue, *"Tenemos una semana para tener todo listo y el diseñador no está disponible para seguir en este proyecto, de modo que serás tú el encargado crear y mostrar la propuesta de diseño al cliente".*

Cómo era de esperar la respuesta no es la que queríamos escuchar, pero como deseamos que nuestro trabajo sea de excelencia aceptamos el desafío.

Para realizar este trabajo tendremos a disposición un *archivo comprimido que contiene el mockup y guía de estilo, junto con la maqueta terminada de la página web*.

A continuación, conoceremos los elementos que constituyen un diseño responsivo y definiremos las diferencias que hay entre los diferentes tipos de diseños que existen.

Conociendo los diseños

Los diseños o layouts que hemos visto hasta el momento son dos, llamados diseño estáticos y diseños fluidos.

Si recordamos estos dos tipos de diseños tienen como diferencia el tipo de unidad de medida usada para escalarlas. Los diseños estáticos están constituidos por unidades absolutas como los píxels y los fluidos por unidades relativas, como los porcentajes.

Tipos de diseño

Estas dos unidades, píxeles y porcentajes, provocan que los diseños se comporten de manera distinta uno del otro. Ingresemos a nuestro navegador y accedamos a la página http://g-mops.net/epica_saitama/epica_layout/index_adaptive.html. En ella encontraremos ejemplos que muestran el comportamiento que adoptan los elementos al trabajar con algún tipo de diseño.

- **Diseños estáticos**

Los diseños estáticos se mantendrán fijos en base a un ancho definido en el documento. Esto significa que si inspeccionamos la página encontraremos el ancho que se define. De la web indicada con anterioridad, a traves del inspector de elemento, encontraremos en el diseño static en el `div` con `id="page_wrapper"`, un valor es `960px`. Asimismo el ancho de bloques e imágenes se encuentran definidos en cada uno de ellos.

- **Diseños fluidos**

Por el contrario los diseños fluidos o líquidos se moverán en base al tamaño del *viewport* (el viewport del navegador es el área de la ventana en donde el contenido web está visible) mediante unidades relativas. Esto significa que cualquier elemento que defina su ancho en relación a la ventana del navegador fluirá en base al cambio de tamaño de este.

Si inspeccionamos la página podremos ver que el ancho en el contenedor principal ya no existe, pero si en cada uno de los contenedores que componen al layout.

Por ejemplo, si revisamos la etiqueta `<menu>` veremos que el ancho del bloque es de `17.5%`. Si eliminamos el ancho del contenedor este perderá la distribución y flujo haciendo que se comporte como un elemento estático.

Diseños adaptativos

Los diseños adaptativos, otro tipo de comportamientos que hasta ahora no hemos visto. Sí reducimos el ancho del *viewport* podremos ver que los bloques del layout se encuentran estáticos y además, estos cambian de posición al disminuir más la ventana.

Los diseños adaptativos se caracterizan por definir el diseño en diferentes tamaños a traves de los ***media queries***, o sea, que en cada tamaño definido se integrará un tipo de diseño preconfigurado para el tipo de resolución especificada. Asimismo, los elementos contenidos en este tipo de diseño usan unidades de medida absolutas para mayor control en el flujo del contenido.

Por ejemplo, si el navegador detecta un tamaño este mostrará un diseño que se acomode a ese *layout*. Si por el contrario, encuentra otro mostrará uno diferente.

Ventajas y desventajas de los diseños adaptativos

Dentro de las ventajas podemos encontrar:

- Establecer nuestros propios tamaños en base a valores que se definan para dispositivos de mayor uso.
- Diferentes diseños de forma independiente con variación de lo que se muestra en pantalla, aumentando la calidad de la experiencia de navegación del usuario.

Las desventajas son:

- No es tan flexible. No se ajusta exactamente a cualquier resolución si no la especificamos.
- En los breakpoints, que son los cambio de un tamaño establecido a otro, pueden existir vacios que visiblemente no puede ser muy agradable para el usuario.
- A pesar de ser sencilla aplicación, implica mayor tiempo de implementación, dado que se debe ajustar en cada breakpoint (punto de quiebre) la nueva distribución, y las imagenes deben ser trabajada en el formato asignado por lo que se puede llegar a tener la misma imagen repetida pero con un tamaño distinto.

Diseños responsivos

Finalmente, nos encontraremos con los diseños responsivos que se caracterizan por usar un sólo diseño que se irá adecuando de acuerdo al tamaño o resolución del dispositivo. Además, tienen la característica de usar unidades relativas en los elementos.

Estos tipos de diseño se definen por tres elementos principales constituidos por grillas, media queries e imágenes flexibles o fluidas.

- **Grillas**: Los sistemas de grillas son unas de los pilares fundamentales de cualquier diseño ya que con estas podremos organizar, dar espacio y equilibrar la distribución de los elementos contenidos en un layout. Permite distribuir el contenido en filas y columnas.
- **Imágenes flexibles**: Un último elemento que define a un diseño responsivo son las imágenes flexibles las cuales tienen la característica de fluir a través del diseño sin perder consistencia.
- **Media Queries**: Otro elemento importante al crear un diseño responsivo es el uso de media queries. Estas directrices de *CSS* nos ayudarán a especificar qué estilos deben aplicarse en una resolución dada permitiendo agregar colores, alturas o anchos en un tamaño definido por nosotros; como dato relevante este standardd fue publicado a principios del año 2013.

Estos dos elementos tan importantes los veremos con mayor detalle más adelante.

Ventajas y desventajas de los diseños responsivos

Las ventajas de los diseños responsivos se relacionan a su fácil implementación, debido a que no se requiere mucho tiempo para crear una experiencia consistente en diferentes dispositivos.

Las ventajas de un diseño responsivo:

- Mantenimiento y actualizaciones, al tener un unico diseño que se vaya ajustando el contenido al sistema de grilla que se implemente.
- Uso de Frameworks css que nos permite la rapida implementación, tal como Bootstrap; la ventaja en este punto es la amplia documentación que constan los distintos frameworks, que nos orientan como aplicar ciertos atributos sin necesidad de usar extensas lineas de css.
- El usuario podra encontrar la misma información tanto en Desktop como Mobile.
- La probabilidad de que el contenido este duplicado es minimo, por lo que la carga del sitio se vuelve más rapida.

Las desventajas son:

- Tiempo de carga en el mobile, cargara de la misma manera que en desktop, todas las imágenes y otros recursos son exactamente igual que para los usuarios web de escritorio.
- El contenido sera el mismo tanto para web como mobile, lo cual puede ser un problema al momento de ofrecer una solución personalizada para mobile como desktop.
- Este tipo de diseño están relacionadas al control que tendremos de la distribución de los elementos. Esto se traduce en que el contenido se distribuye en un formato, por ejemplo con Bootstrap, que quizas no sea necesariamiente el que deseamos, ya que viene con valores predefinidos y no sea comodo tanto como para el desarrollador como para el cliente.

Diseños responsivos v/s adaptativos

Ahora bien, las diseños responsivos y adaptativos tienen dos diferencias que hacen que uno sobresalga del otro dentro de los *layout* multidispositivo.

La primera tiene que ver con la implementación del diseño. Un diseño responsivo es más económico en cuanto a tiempo de creación y cantidad de personas requeridas que un diseño adaptativo, lo que a la larga hace una gran diferencia.

Además, los diseños responsivos ofrecen una experiencia de usuario lo suficientemente buena para que un usuario pueda navegar correctamente en un sitio o aplicación multidispositivo.

La segunda diferencia esta relacionada a la popularidad y gran aceptación que tiene este diseño en el mundo de la tecnología.

En conclusión, los diseños responsivos son, sin lugar a dudas, el tipo de diseño que sobresale dentro de esta comparación por su fácil implementación y popularidad.

Teniendo en cuenta esto, más adelante transformaremos el diseño estático presentado en el ejercicio e integraremos todos los elementos que hacen de este un diseño responsivo.

Definir un diseño responsivo

Aprenderemos las diferentes opciones que como desarrollador web tenemos para transformar un sitio web estático a uno *responsive*, ya que el comportamiento de un diseño responsivo no sólo es apilar el contenido en dirección vertical.

Antes de comenzar a transformar nuestra maqueta estática en una responsiva debemos realizar un paso previo.

Si hacemos memoria, en el ejercicio, el Product Owner del equipo nos respondió que nosotros seremos los responsables de presentar al cliente las propuestas de diseño responsivo, esto quiere decir que, deberemos pasar al lado del diseño por un momento para lograr terminar la maqueta.

Estrategias diseñar páginas responsivas

Ahora veremos las diferentes opciones con las que contamos para transformar un sitio web estático a uno responsive.

Las estrategias que usaremos serán tres:

- Definir el flujo del contenido usando patrones de diseño responsivo
- Seleccionar el contenido en función a la importancia
- Diagramar resultado en una representación visual de baja fidelidad

Patrones de diseño responsivo

Los patrones de diseño web adaptables han evolucionado rapidamente, pero existen varios patrones establecidos que operan correctamente en los distintos dispositivos desktops y mobile. Los principales patrones son:

- mostly fluid
- column drop
- layout shifter
- tiny tweaks
- off canvas

Estos patrones se pueden combinar en un sitio web sin generar problemas.

Para el desarrollo de cualquier proyecto, el primer punto que tendremos que abordar es **cómo identificar el flujo que tendrá el contenido del sitio cuando el *viewport* disminuya su tamaño**.

Para hacerlo podremos usar una estrategia simple pero muy efectiva con la que no tendremos que reinventar nada para poder conseguir un diseño responsivo que encante a nuestro cliente.

- ***Mostly fluid***:

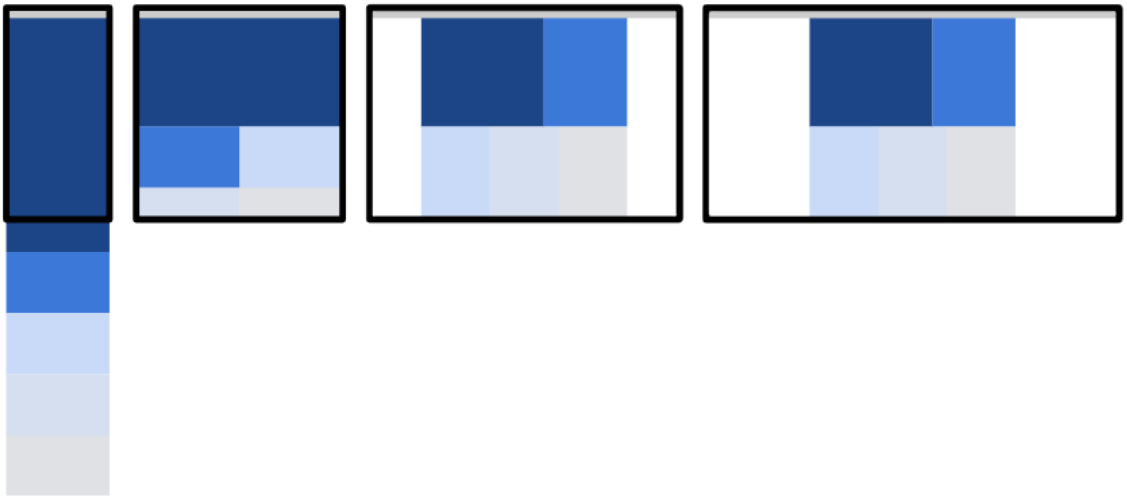

Imagen 1. Patrón Mostly Fluid.

El patrón más usado es el `mostly fluid` que como lo indica su nombre fluye la mayor parte del tiempo. En smartphone se forma una única columna y las filas generan bloques. A medida que la pantalla crezca los distintos bloques se agrupan ocupando toda la pantalla disponible. En dispositivos más grandes el diseño es el mismo, pero se agrupa dentro un contenedor, que se centra respecto a la pantalla con un ancho fijo.

Lleva el siguiente codigo a tú editor de texto y analizalo:

```
<!DOCTYPE html>
<html>
<head>
  <title>Mostly Fluid</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <nav>Navbar</nav>
  <div class="container">
    <section class="columna1">Sección 1</section>
    <section class="columna2">Sección 2</section>
    <section class="columna3">Sección 3</section>
    <section class="columna4">Sección 4</section>
    <section class="columna5">Sección 5</section>
  </div>
</body>
</html>
```

```
/* Mostly Fluid */
.container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-flow: row wrap;
  flex-flow: row wrap;
}

.columna1,
.columna2,
.columna3,
```

```
.columna4,
.columna5 {
  width: 100%;
}

@media (min-width: 600px) {
  .columna2,
  .columna3,
  .columna4,
  .columna5 {
    width: 50%;
  }
}

@media (min-width: 800px) {
  .columna1 { width: 60%; }
  .columna2 { width: 40%; }
  .columna3,
  .columna4,
  .columna5 {
    width: 33.3%;
  }

  .container {
    width: 800px;
    margin-left: auto;
    margin-right: auto;
  }
}
```

- ***Column Drop***:

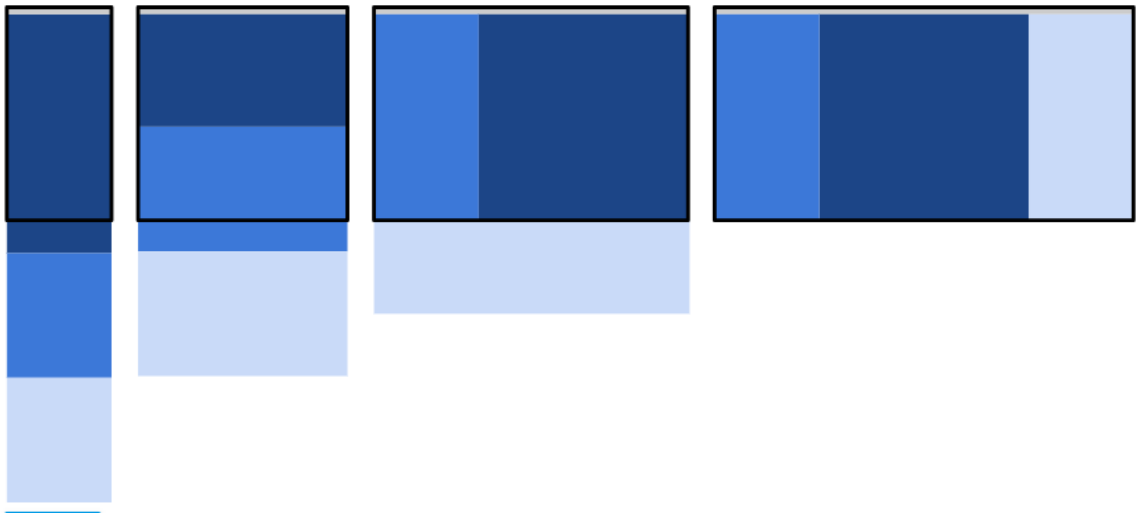

Imagen 2. Patrón Column Drop.

Este patrón consiste en que cada bloque de contenido, que en un smartphone vemos en filas, vaya formando columnas según vaya siendo más grande la pantalla del dispositivo.

Lleva el siguiente codigo a tú editor de texto y analizalo:

```
<!DOCTYPE html>
<html>
<head>
  <title>Column Drop</title>
```

```
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <nav>Navbar</nav>
  <div class="container">
    <section class="columna1">Sección 1</section>
    <section class="columna2">Sección 2</section>
    <section class="columna3">Sección 3</section>
  </div>
</body>
</html>
```

```
/* Column Drop */
.container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-flow: row wrap;
  flex-flow: row wrap;
}

.columna1,
.columna2,
.columna3 {
  width: 100%;
}

@media (min-width: 600px) {
  .columna1 {
    width: 60%;
    -webkit-order: 2;
    order: 2;
  }

  .columna2 {
    width: 40%;
    -webkit-order: 1;
    order: 1;
  }

  .columna3 {
    width: 100%;
    -webkit-order: 3;
    order: 3;
  }
}

@media (min-width: 800px) {
  .columna2,
  .columna3 {
    width: 20%;
  }
}
```

- ***Layout Shifter***:

Imagen 3. Patrón Layout Shifter.

Los patrones `layout shifter` son aquellos que poseen varios puntos de quiebre en los cuales se define una posición especifica del diseño. En esencia estos patrones contienen muchas de las facultades que definen a los diseños adaptativos, como los breakpoints en diferentes tamaños y la colocación de elementos definida para ofrecer la mejor experiencia, pero se diferencia en el tipo de unidades que usa, ya que el flujo de los elementos es fluido y no estático como estos tipos de diseño

En tamaños grandes se fija un margen usado para no desbordar el diseño. En los tamaños medianos los elementos comienzan a apilarse de manera vertical uno sobre otros y en lo tamaños pequeños mantienen este flujo.

Lleva el siguiente codigo a tú editor de texto y analizalo:

```
<!DOCTYPE html>
<html>
<head>
  <title>Layout Shifter</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <nav>Navbar</nav>
  <div class="container">
    <section class="columna1">Sección 1</section>
    <div class="container-inner">
      <section class="columna2">Sección 2</section>
      <section class="columna3">Sección 3</section>
    </div>
  </div>
</body>
</html>
```

```
/* Layout Shifter */
.container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-flow: row wrap;
  flex-flow: row wrap;
}

.columna1,
.columna2,
.columna3,
```

```
.container-inner {
  width: 100%;
}

@media (min-width: 600px) {
  .columna1 {
    width: 25%;
    height: 100vh;
  }

  .container-inner {
    width: 75%;
  }
}

@media (min-width: 800px) {
  .container {
    width: 800px;
    margin-left: auto;
    margin-right: auto;
  }
}
```

- ***Tiny Tweaks***:

Imagen 4. Patrón Tiny Tweaks.

Los diseños `tiny tweaks` se destacan por su simplicidad, dado que se basa en una sola columna para el contenido, en todos los dispositivos.

Lleva el siguiente codigo a tú editor de texto y analizalo:

```
<!DOCTYPE html>
<html>
<head>
  <title>Tiny Tweaks</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <nav>Navbar</nav>
  <section class="columna1">Sección 1</section>
</body>
</html>
```

```
/* Tiny Tweaks */
.columna1 {
  padding: 10px;
  width: 100%;
}
```

```
@media (min-width: 600px) {
  .columna1 {
    padding: 20px;
    font-size: 1.5em;
  }
}

@media (min-width: 800px) {
  .columna1 {
    padding: 40px;
    font-size: 2em;
  }
}
```

- ***Off Canvas***:

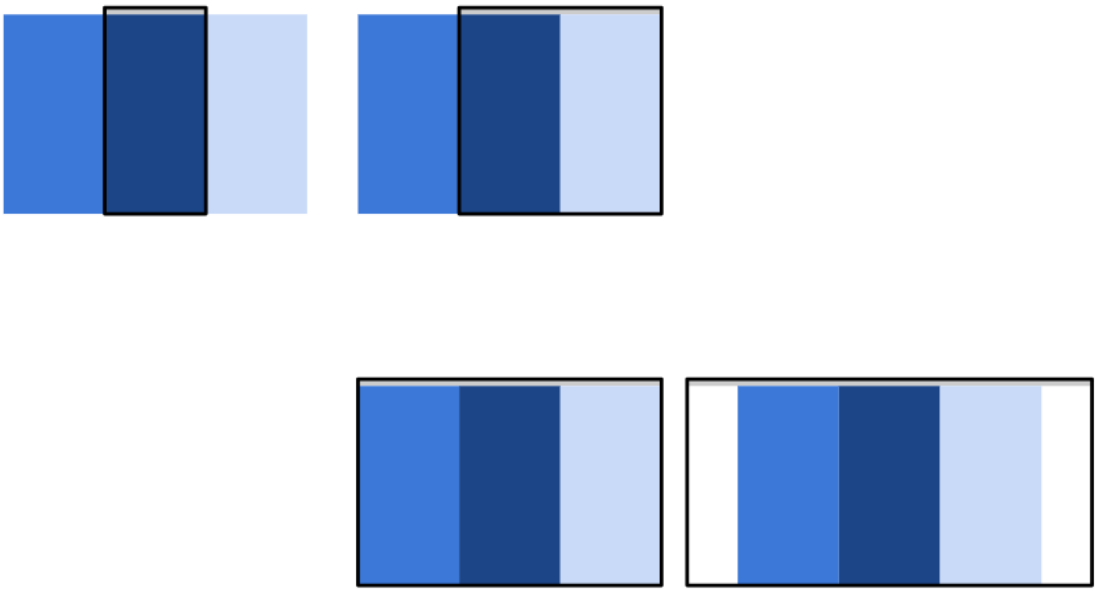

Imagen 5. Patrón Off Canvas.

Finalmente, los patrones `off canvas` son diseños que se destacan por el uso de columnas verticales que se van escondiendo dependiendo de la importancia del contenido.

Lleva el siguiente codigo a tú editor de texto y analizalo:

```
<!DOCTYPE html>
<html>
<head>
  <title>Off Canvas</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <nav>Navbar</nav>
  <div class="container">
    <section class="columna1">Sección 1</section>
    <section class="columna2">Sección 2</section>
  </div>
</body>
</html>
```

```
.columna1 {
```

```
  position: absolute;
  width: 250px;
  height: 100vh;
  -webkit-transition: -webkit-transform 0.3s ease-out;
  transition: transform 0.3s ease-out;
  z-index: 1;
}


.columna1 {
  -webkit-transform: translate(-250px, 0);
  transform: translate(-250px, 0);
  background-color: peru;
}


.columna1.open {
  -webkit-transform: translate(0, 0);
  transform: translate(0,0);
}


.columna2 {
  width: 100%;
  height: 100vh;
  position: absolute;
  background-color: purple;
}
@media (min-width: 600px) {
  .container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-flow: row nowrap;
    flex-flow: row nowrap;
  }


  .columna1 {
    -webkit-transform: translate(0, 0);
    transform: translate(0, 0);
  }
}
```

Material Complementario

- Multi-Device Layout Patterns - Luke Wroblewski
- Ejemplos de patrones responsivos
- Patrones de diseño web adaptables

Identificando el flujo de los elementos

Despues de aprender y analizar sobre los `Patrones de diseño responsivo` en un proyecto, podriamos identificar perfectamente cuál es el indicado en base a lo planteado por el diseñador de la interfaz de usuario, lo anterior en función al modelo de negocio del cliente y las necesidades de los usuarios finales, trabajo que realiza el UX (user experience).

Simplemente debemos identificar las similitudes que tiene el mockup con los patrones de diseño y en base a esa comparación escoger la opción que se acomode más a lo planteado por el UI (User Interface).

De no cumplirse la opción de contar con un diseño del UI, es necesario considerar los pasos que de setallaran a continuación.

Diagramar una representación visual

Realizaremos representaciones visuales de menor calidad, más que ser un experto en el dibujo, es muy util tomar el tiempo de realizar estas representaciones, ya que son una guia muy importante antes de comenzar a maquetear nuestro sitio web, con esta guia podemos tener una idea clara de como se desplegará la información (imagenes y textos) que queremos transmitir en nuestro sitio web. Este paso lo podemos obviar de existir una maqueta pre definida por el UI (User Interface).

Si hacemos memoria, las representaciones visuales se dividen en tres tipos, definidos por su grado de fidelidad. Los *sketch* y *wireframes* son los de menor calidad, los *mockup* de mediana fidelidad y los *prototipos* ofrecen la mejor calidad.

Siempre, en una primera etapa es recomendable usar formas más sencillas de crear una representación visual, sin gran esfuerzo y sólo usando lápiz y papel. Los ***sketch*** representan una forma rápida y barata de presentar un diseño a otras personas que requieran conocer la propuesta de diseño, de modo que es la mejor alternativa que podremos usar.

Preparar los elementos a usar

Primero, prepararemos los elementos necesarios para realizar esta tarea. Es importante tener en cuenta que para dibujar una interfaz de usuario no se requiere de ninguna habilidad o técnica especial para obtener grandes resultados, pero si necesitamos conceptualizar el como definir lo que es texto de imagen (que se representa con líneas o especies de olas), la imagen (que es un cuadrado o rectangulo de acuerdo al espacio que ocupe con una gran X que cubre entre los extremos), los videos (que deben tener el simbolo de play), los demas elementos se pueden representar con la etiqueta asociada en html, tal como button, navbar, footer, etc.

Teniendo esto claro, los elementos que necesitaremos serán:

- Un lápiz
- Una hoja de papel
- Una regla

Con todos estos elementos a mano estaremos listos para comenzar a diagramar la interfaz de usuario, como se muestra en la siguiente imagen.

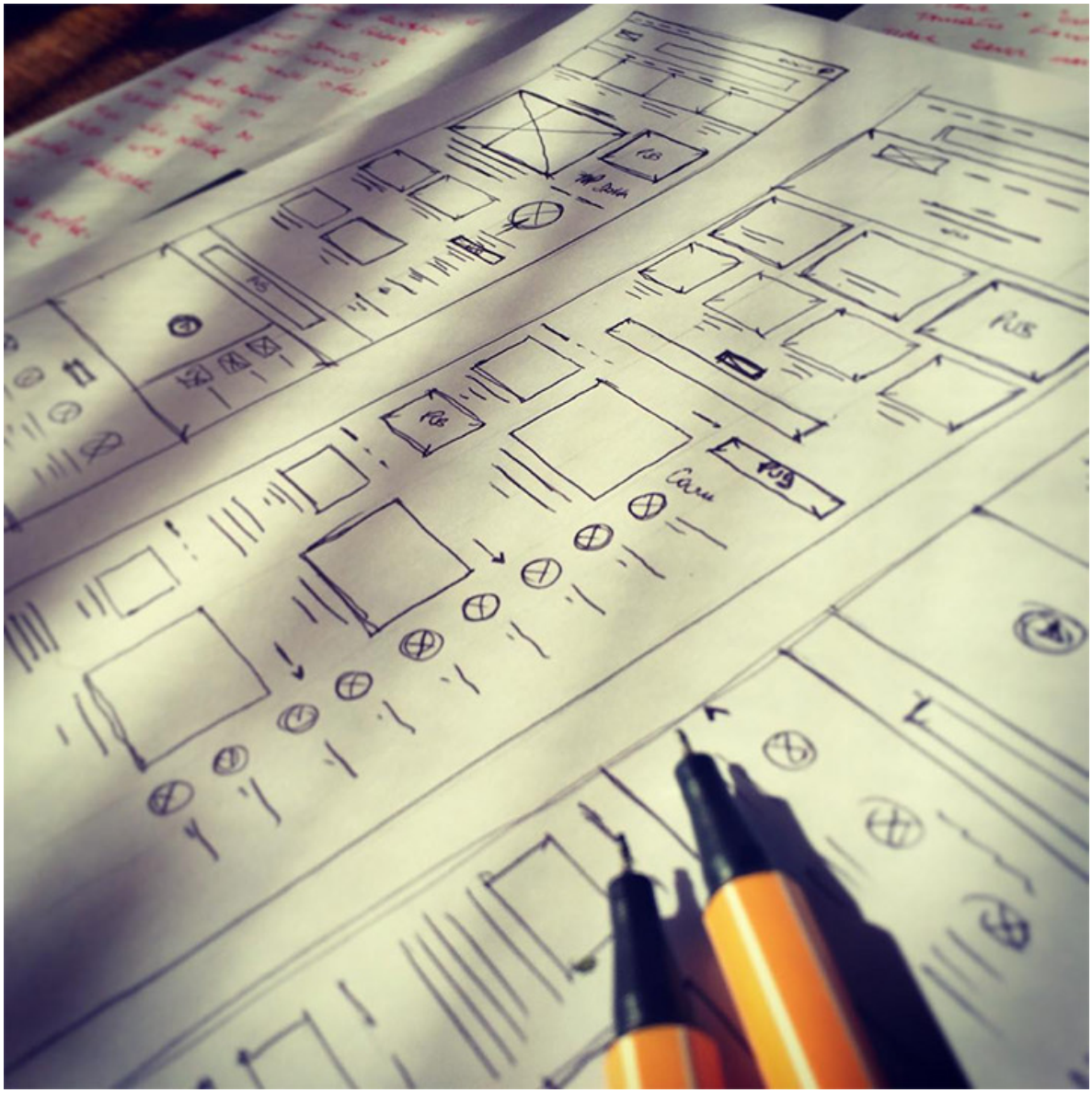

Imagen 5. Ejemplo de diagrama de interfaz de usuario.

Sketch_ dispositivos Desktop

- **Navbar**

Es la barra superior donde podemos encontrar los links que nos llevan a todo el sitio. Tener una navegación fácil de usar es importante para cualquier sitio web.

- **Hero Section**

En primer lugar tenemos la sección hero, la cual muestra el contenido principal del sitio. Es una imagen grande con texto, a menudo ubicada en la parte superior de una página web, tambien podria considerar un botón.

- **History y About Section**

La segunda sección más comun y utilizada en los sitios web, puede llevar por nombre history la cual contiene una reseña con la historia de la compañia, inicios y como ha crecido en el tiempo.

Asimismo las sección about hace referencia al sobre la empresa, a diferencia de history en about nos indica la actualidad de la empresa, contiene pruebas sociales, testimonios y alguna información personal con la que los espectadores puedan relacionarse.

Estos dos elementos contienen información muy relevante sobre el producto de modo que ambos es importante que esten en tamaño desktop y mobile.

- **Gallery Section**

Por otra parte la sección de gallery, según el contexto del sitio, no contiene ningún tipo de contenido que sea primordial para el usuario, de modo que podríamos esconder esta sección en resoluciones móviles.

- **News Section**

La sección news por su parte presenta unos componentes muy comunes llamados cards o wells que los cuales contienen una reseña de las últimas noticias agregadas al blog de del sitio web.

Las card que se encuentran en al interior de esta sección en tamaños pequeños deberían fluir hasta apilarse uno sobre otra en dispositivos moviles.

- **Footer**

La última sección de la página es el footer. Esta, dependiendo del sitio, puede contener el logo de la empresa, iconos hacia redes sociales, dirección vinculado a google maps, número de telefono, correo electronico con el fin de tomar contacto y el copyright. En otros casos dependiendo del contexto del sitio puede incluir normativas legales, sitios de interes y el sobre mi con información referente el servicio, trabaja con nosotros y mapa del sitio.

Sketch dispositivos móviles

Comencemos dibujando el *viewport* del sitio haciendo un rectángulo de manera vertical, que cubra sólo la mitad del papel. Cuando esté listo, comenzaremos a traspasar el flujo que creamos cuando definimos el flujo de los elementos del sitio en desktop.

Un sitio web con un flujo normal estaria compuesto con una barra de navegación, siguiendo un patrón UI muy común de los diseños responsivos que consta de agregar tres lineas en horizontales, la cuáles representan a la navegación del sitio. Si presionamos esta *Hamburguesa* se desplegará el menú hacia abajo.

Para definir esta navegación en el papel pondremos el logo a mano izquierda y a la derecha la hamburguesa.

Despues se incorpora la sección de contenido por nombre `hero`, en el dispositivo movil debemos adaptar tanto el texto, imagen y de existir el boton, aplicando margenes adecuados.

Luego, dibujaremos el contenido de la sección `about` e `history` las cuales tienen elementos e importancia similar, pero que cambiaremos de orden por temas de contraste visual.

Como definimos anteriormente la sección que irá arriba será `about` y abajo `history`.

La penúltima sección estará conformada por las tarjetas apiladas una por sobre otras de manera horizontal.

En la hoja de papel que tenemos es probable que no alcancemos a agregar todos los elementos que vienen incluidos, de modo que podremos usar alguna estrategia visual para ayudar a la persona que verá el *sketch* a entender que hay más elementos. Podríamos usar tres puntos o una flecha indicando para dónde irá el flujo.

Finalmente, nos encontraremos con el `footer`. El footer variara la confección en cuanto a las redes sociales, copyright y logo por temas de tamaño, idear como fluiran o si es necesario esconder.

Sketch dispositivos medianos (Tablets)

En cuanto a los dispositivos medianos tendremos la posibilidad de crear un sketch para este tipo de tamaño, pero como pueden haber variaciones por distribución de contenido, debemos definir que patron de diseño usaremos para la nueva distribución.

Teniendo claro cuáles seran las distribuciones tanto en Mobile, Desktop y Tablet podremos definir que se ajusta más para trabajar en nuestro proyecto, dentro de las cinco alternativas de Patrón de diseño explicadas con anterioridad.

Selección de contenido en función a la importancia

Podremos usar una simple pero efectiva solución que consta de esconder el contenido poco relevante de un sitio en tamaños pequeños y luego mostrarlos en tamaños más grandes.

Este proceso se basa en la importancia contextual que tiene un contenido dentro una página haciendo que en dispositivos se muestre una sólo lo esencial del contenido total y en tamaños grandes todo el contenido.

Para hacerlo deberemos identificar qué contenido es poco relevante dentro de una sección y luego, cuando creemos la interfaz, esconder o eliminar el contenido, dependiendo del caso, usando `visibility: hidden`, `display: none;`, si estamos utilizando el framework bootstrap hacemos uso de la utilidad Display.

Mockup y Prototipo

Tal como se a señalado en unidades anteriores, el *mockup* es una representación visual en media alta que nos permite visualizar, en el momento, el contenido simulado, como quedaria el sitio web, en tanto el prototipo es funcional, es confeccionado a traves de software tal como Sketch, Adobe Xd, Figma, entre otros. Esto permite tener un producto mínimo viable que el cliente puede interactuar, además del corto tiempo de implementación que este tiene, a diferencia del desarrollo web, con todas las integraciones necesarias para su correcto funcionamiento.

Presentación de representación a clientes

Teniendo el último entregable listo ya es momento de ir y presentarle al cliente la propuesta realizada. Si bien no se ha realizado un estudio acucioso del usuario que ingresara a este sitio web, si no más bien es una representación funcional del flujo que tendrá el sitio en diferentes dispositivos, de modo que es importante usar este factor como una estrategia para lograr que el cliente acepte el diseño.

En conclusión, confeccionar un sitio web conlleva el entender el flujo de los elementos, usar estrategias para diseñar interfaces funcionales y presentar al cliente una idea de cómo se comportará su página.

La suma de todos elementos provocará dos resultados como crear una propuesta nueva o que acepten nuestra propuesta.

En el caso de aceptar nuestra propuesta estaremos listos para construir nuestro sitio web desde el editor de texto favorito.

Lectura complementaria

- Diferencia entre sketch, wireframe y mockup/prototipo

Definiendo el tipo de diseño a usar

Ahora que tenemos definido el flujo de los elementos y que conocemos que elementos se pueden priorizar en dispositivos pequeños, estamos listos para elegir el patrón de diseño que se adecuará de mejor manera al diseño propuesto en el *mockup*.

En base a los patrones aprendidos, se puede definir cuál se adecua más al proyecto en el que estemos trabajando.

Mobile First

El Mobile First es un concepto muy importante hoy en día ya que, todo el mundo cuenta con un dispositivo móvil y he aquí el desafio al momento de diseñar y maquetear un sitio web responsive ya que, debemos aprovechar al máximo el poco espacio que tenemos, además al pensar que en mobile first se saca el contenido relleno de nuestro sitio web, así mismo el código html y css es mucho más estructurado ya que, hay propiedades y estilos que se mantienen a lo largo de las diferentes pantallas, además que por experiencia es más fácil ir agregando estilos y propiedades que ir sacandolos.

Antes de continuar es importante profundizar en el concepto *Mobile First*, muy popular dentro del diseño responsivo.

¿Qué es *Mobile First*?

Desde hace un buen tiempo los teléfonos inteligentes se han transformado en la primera opción de los usuarios para consumir contenido por internet. Esto ha transformado dramáticamente el enfoque de los diseños usados para sitios y aplicaciones haciendo que el paradigma cambie a favor de este tipo de tecnología.

La filosofía *Mobile First* nació como una alternativa para desarrollar sitios usando las capacidades que tienen los dispositivos como los gestos y otros atributos de un dispositivo móvil como el GPS, sensores y otras características únicas que se pueden usar para mejorar la experiencia de los usuarios en estos dispositivos.

Además de lo anterior, esta filosofía entrega algunas directrices muy interesantes que nos ayudarán a construir mejores maquetas multidispositivo.

Dentro de los puntos destacables podremos encontrar:

- **Adecuarse a los cambios**

Esto en concreto significa que el mundo móvil cambia rápidamente y los tamaños, resoluciones y capacidades que tenga el dispositivo cambiarán a medida que pase el tiempo, de modo que nosotros deberemos estar atentos a estos cambios y siempre tomar la iniciativa para mejorar la experiencia de los usuarios utilizando estas nuevas funcionalidades.

- **Usar la tecnología a nuestro favor**

Por ejemplo, podremos aprovechar el etiquetas *HTML* que nos permitirán adaptar nuestro sitio a estos dispositivos usando *meta viewport*. También podremos usar a nuestro favor las densidades de las pantallas de estos dispositivos para mejorar la resolución de nuestra interfaz.

- **Diseño responsivo**

En esta filosofía el tipo de diseño que realicemos afectará directamente en la experiencia que tendrá posteriormente el usuario. En esencia este punto hace referencia a las decisiones de implementación de un tipo de diseño.

Por ejemplo si quisieramos construir un sitio deberiamos comenzar por construir el diseño para dispositivos móviles y luego ir escalando hacia resoluciones más grandes a modo de priorizar la experiencia de estos dispositivos.

En contraste, la construcción un layout responsivo con un enfoque hacia los tamaños de escritorio mejorarán la experiencia en *desktop* pero no estarán optimizados para trabajar en dispositivos móviles.

En resumidas cuentas, lo que prima en la construcción de diseños basados en *mobile first* o *desktop first*, dependerá del proyecto y de las características que tenga este, ya que concociendo estos atributos podremos crear mejores interfaces depuradas para el usuario objetivo del dispositivo.

- **Homogeneizar la experiencia en los dispositivos**

Asi como debemos tener en cuenta la construcción de de los diseños, también debemos hacerlo con los dispositivos. Esto significa que debemos ser capaces de homogeneizar la experiencia en todos los dispositivos en los cuales deseemos mostrar contenido.

En perspectiva, lograr tal objetivo contempla conocer las características y limitaciones de los dispositivos y en base a estas definir cual es la solución idónea para un tipo de contenido.

Como ejemplo de esto podríamos notar las diferentes interfaces de usuarios que tiene Spotify. Podremos ver un diseño específico en dispositivos móviles y en smart TV otro. Esta definición de experiencia de usuario única hace que una aplicación ofrezca un mismo servicio pero optimizado para cada tipo de usuario.

- **Reducir al máximo las funcionalidades en dispositivos móviles**

Este último punto hace referencia a lo necesario que es priorizar el contenido y funcionalidades que se mostrarán en dispositivos móviles.

El priorizar significa definir qué contenido es importante y como estos llegarán de la manera más sencilla al usuario.

En resumidas cuentas, usar algunas de estas estrategias nos dará la posibilidad de que nuestras maquetas sean escalables y centradas en la experiencia del usuario final.

Lectura complementaria

- Luke Wrobleswki (2014): “Mobile First”, *A book part*, pp. 108 - 119

Componentes de un diseño responsivo: *Viewport*

Aprenderemos a cómo sacar el máximo probecho al viewport y la importancia de saber utilizar y enteder los valores y atributos de la etiqueta `<meta name="viewport">` y así entregar a nuestro usuario final la mejor de las experiencias al visitar nuestro sitio web en los diferentes dispositivos.

Como vimos anteriormente, el usar la tecnología a nuestro favor es una de las mejores estrategias que tendremos para construir interfaces de usuario multidispositivo.

Esto en la práctica conlleva utilizar algunas capacidades específicas de los dispositivos o programas con la cuales podremos modificar y crear algunos comportamientos en nuestros diseños.

Hemos hablado del *viewport* desde hace un buen tiempo teniendo en mente el tamaño de la ventana del navegador, pero hay más que solamente eso.

Tipos de *viewport*

Cuando hablamos de *viewport* debemos tener en cuenta dos elementos importantes, en primer lugar, el *viewport* está constituído por dos tipos de disposición:

1. La primera esta basada en el *viewport* que podemos ver y el otro, por el que hemos diseñado.

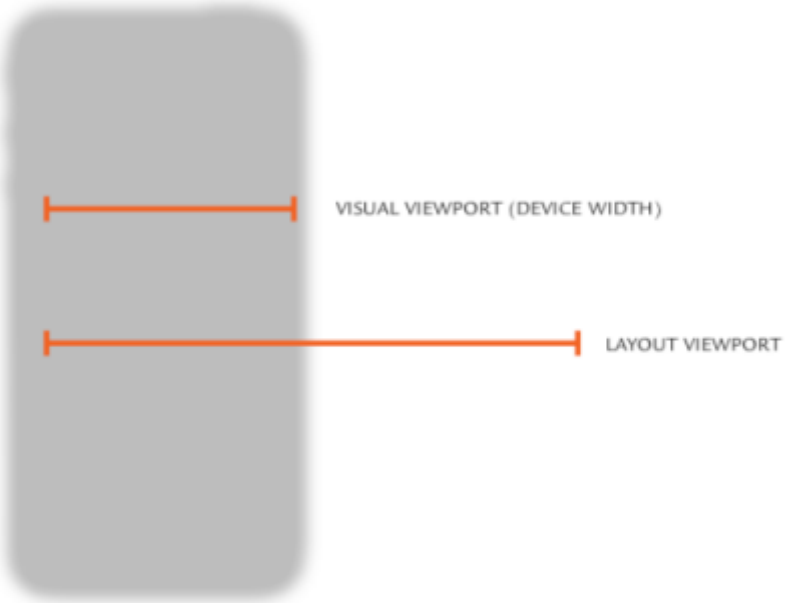

Imagen 7. Ejemplo de viewport.

El *viewport* visual tiene como característica que su tamaño varía y el *viewport* de diseño no. Esto debe que el diseño del *viewport* es en esencia más amplio que el *viewport* visual.

En la siguiente imagen podemos apreciar un ejemplo con mayor valor del *viewport* que se muestra.

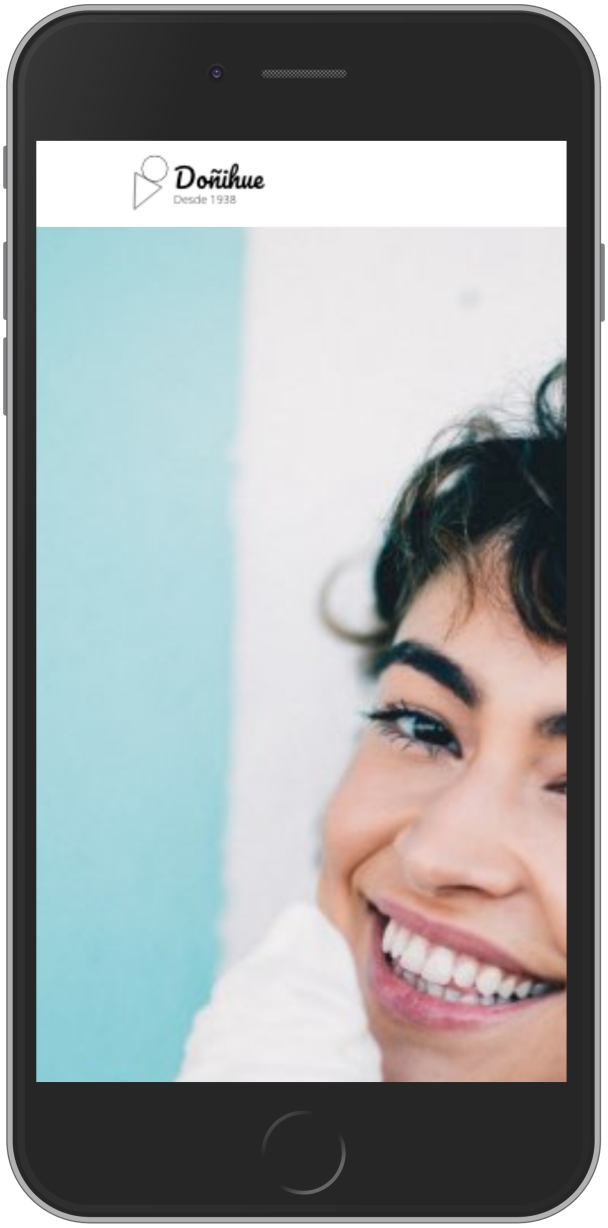

Imagen 8. Ejemplo de cómo se muestra viewport.

Esto ocurre debido a que el tamaño del diseño propuesto es mayor al del dispositivo, haciendo que el navegador identifique que el diseño es más grande que la pantalla del dispositivo móvil. Esto como resultado mostrará el sitio con *zoom*.

Sin embargo, este comportamiento visto anteriormente puede ser controlado usando una etiqueta imprescindible para construir diseños responsivos llamada `<meta name="viewport">`.

¿Qué es el *meta viewport*?

La etiqueta `<meta name="viewport">` controla y escala el diseño del *viewport* en diferentes dispositivos.

Esta etiqueta, como toda etiqueta meta, se alojará dentro de `<head>`

La estructura de esta etiqueta es la siguiente:

```
<head>
  <meta name=”viewport” content=”directiva,directiva” />
</head>
```

Primero, se especifíca que controlaremos el *viewport* con el atributo `name="viewport"` y luego, se listan las directivas que usaremos para controlar nuestro *viewport*.

¿Qué directivas podremos usar?

Dentro de las directivas que podremos usar se encuentran:

- **width**:

```
<meta name=”viewport” content=”width=device-width” />
```

Estas nos permiten especificar al navegador el ancho que deseamos usar en el sitio. Podremos usar un ancho en específico usando unidades relativas o absolutas o podemos definirlas de manera automática con ayuda del navegador usando la directiva `width=device-width`.

- **Height:**

```
<meta name=”viewport” content=”height=device-height” />
```

Al igual que la directiva `width` con `height` podremos definir el alto que tendrá nuestro nuestro sitio. Podremos definir un alto específico o definirlo de manera automática con `height=device-height`.

- **User-scalable:**

```
<meta name=”viewport” content=”user-scalable=no” />
```

La directica `user-scale` nos permite decirle al navegador que de o no la posibilidad al usuario de usar zoom en nuestra página. Como valores permitidos se encuentra el `user-scalable=no”` y `user-scalable=yes”`.

- **Initial-scale:**

```
<meta name=”viewport” content=”initial-scale=1”/>
```

Esta directiva declara al navegador el zoom inicial que tendrá nuestra interfaz. Los valores que podremos usar comienzan con `0.1` (10%) y terminan con `10.0` (1000%).

Si queremos que la escala de nuestro sitio inicie con el `100%` de zoom, deberemos agregar como valor `initial-scale=1`.

- **Maximun-scale:**

```
<meta name="viewport" content="initial-scale=1, maximum-scale=10">
```

La directiva `maximum-scale` nos permite definir el máximo de zoom que los usuarios podrán hacer en nuestro sitio. Los valores al igual que `initial-scale` comienzan con `0.1` y terminan con `10.0`.

- **Minimun-scale:**

```
<meta name="viewport" content="content=initial-scale=1, minimun-scale=1">
```

La última directiva nos ayudará a definir el zoom mínimo que tendrá nuestro sitio.

Los valores que podremos definir comienzan con `0.1` y terminan con `10.0`.

¿Cuáles son las directrices recomendadas?

Teniendo claro las opciones que tenemos para controlar el *viewport* es importante tener en cuenta cuál es la configuración recomendada para comenzar a trabajar con diseños responsivos.

Por mucho, la mejor forma de definir el comportamiento del *viewport* es utilizando un `<meta name="viewport" content="width=device-width, initial-scale=1.0">` ya que, con esto el nevegador definirá el ancho de nuestro sitio y mostrará siempre nuestra interfaz con un *zoom* de `100%`.

Ahora que ya sabemos que configuración usaremos para el *meta viewport*, comenzaremos a trabajar en la siguiente maqueta a un diseño responsivo usando esta etiqueta.

Para eso iniciaremos nuestro editor de texto y luego buscaremos el proyecto en la plataforma empieza con el sitio a modificar. Luego, de encontrarlo abriremos el terminal e iniciaremos el compilado de Sass con el comando `sass --watch` y la ruta de nuetro archivo .scss y css.

```
>> sass --watch assets/sass/main.scss:assets/css/main.css
>>> Sass is watching for changes. Press Ctrl-C to stop.
      write assets/css/main.css
      write assets/css/main.css.map
```

Finalmente, busquemos el archivo `index.html` a modo de ver los cambios que haremos en el sitio.

Ahora, que se encuentra todo listo agregaremos dentro del archivo `index.html` la etiqueta `<meta name="viewport">`, con un valor `content="width=device-width, initial-scale=1.0">`.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Guardemos y revisemos en el inspector de elementos para ver qué cambió en la página. Si presionamos en el icono de vista del dispositivo y luego seleccionamos un dispositivo como un iPhone podremos ver que la página no varió demasiado a lo visto en el ejemplo pasado.

Esto se debe a que los elementos que contiene esta interfaz se encuentran estáticos a un tamaño específico, que en este caso es de `1180px`.

Si comentamos este valor desde el parcial `_base` y luego recargamos la pestaña veremos que los elementos que constituyen la interfaz perdieron la distribución que se había definido anteriormente.

Esto ocurre debido a que ya no existe un viewport de diseño que defina el ancho al sitio, esto hace que los elementos fluyan en base al tamaño total de la ventana del navegador.

Es importante notar que existen algunos elementos dentro del documento que están definidos con unidades absolutas, esto significa que si se mantiene así el contenido no fluirá a medida que se reduce o aumenta el tamaño y en el caso de las fuentes, los usuarios no podrán cambiar el tamaño de fuente desde el navegador, afectando la usabilidad del sitio.

Lecturas complementarias

- A tale of two viewports - Peter-Paul Koch
- A pixel is not a pixel is not a pixel - Peter-Paul Koch

Transformar unidades absolutas a relativas

Como maquetador web es importante saber transformar unidades absolutas a relativas y viceversa, en esta unidad recordaremos como realizar estas transformaciones y además crearemos un *mixin* que nos ayude a transformar unidades de manera más rápida.

Para solucionar estos dos problemas podremos usar algunas estrategias útiles para transformar los elementos del sitio a unidades relativas.

La estrategia que usaremos para este problema será crear un conversor de unidades absolutas a relativas.

Si hacemos memoria, las unidades relativas son aquellas que basan su tamaño a un contexto. Esto significa que la unidad basa su tamaño en algún elemento del navegador. Por ejemplo, las unidades `rem` y `em` basan sus medidas en el tamaño de fuente y los `%` o `vw` usan como referente al tamaño del *viewport*.

Objetivo y contexto

Para transformar unidades absolutas a relativas debemos tener en cuenta dos aspectos importantes:

1. El primero es el valor que deseamos transformar
2. El segundo es el contexto que tiene la unidad

En esencia para lograr esta conversión deberemos usar un poco de matemática básica, dividiendo el valor a transformar por el contexto de la unidad a transformar.

```
objetivo / contexto = resultado
```

A primera vista parece algo complicado, pero no lo es.

Transformar px a unidades relativas al tamaño de fuente

Imaginemos que deseamos transformar el tamaño de un título de `30px` a una unidad relativa como `rem`.

Para lograrlo deberíamos dividir los `30px`, que es el tamaño a transformar, por el tamaño de fuente base del documento, definido en `16px`, o sea, el contexto que tiene la unidad a transformar.

```
30 / 16 = 1.875
```

Si realizamos esa operación obtendremos un total de `1.875rem`.

Como vimos en el ejemplo anterior, el valor a transformar se dividió por el valor que constituye una unidad `rem` que basa su tamaño en relación al valor definido en la raíz del documento.

Si quisieramos hacer lo mismo pero transformando `px` en unidades `em`, deberíamos saber cual es el tamaño de fuente definido en el elemento padre. Si conocemos valor de este elemento podremos transformar fácilmente la unidad absoluta en `em`.

Transformar *px* a unidades relativas al *viewport*

Por otra parte, con esta misma regla podremos transformar unidades relativas basadas en el ancho del viewport.

El procedimiento para transformar unidades es el mismo sólo que a este deberemos multiplicar el resultado por `100`.

```
objetivo / contexto * 100 = resultado
```

Por ejemplo, si quisieramos transformar el ancho de un contenedor a `%` deberíamos dividir el tamaño del contenedor, que en este caso es `700px`, por el ancho máximo del diseño definido en `1180px`, y finalmente multiplicar el resultado por `100`.

```
700 / 1180 = 59,3220
```

Como resultado de la operación obtendremos `59,3220%`.

Así mismo, si quisieramos transformar `px` a unidades `vw` deberíamos hacer el mismo procedimiento, ya que `1vw` equivale al `1%` del ancho del viewport.

Como vemos el uso de esta regla nos ayudará a transformar nuestras unidades absolutas a diferentes tipos de unidades relativas usando el contexto de estas últimas.

Ahora bien, el transformar estas unidades de medida a mano puede resultar una tarea tediosa que muy probablemente queramos omitir, pero existe una forma de reutilizar esta regla en cualquier lugar de nuestro proyecto usando la directriz de Sass llamada `mixin`.

Creando conversor de unidades de medida

Si recordamos los `mixin` son una directiva de Sass que nos ayudará a reutilizar reglas *CSS* dentro de nuestras hojas de estilo.

Para construir nuestro conversor de unidades iremos al directorio `abstracts` y presionaremos el parcial `_mixins`.

En el interior de este archivo agregaremos la directiva `@mixin` y al lado pondremos el nombre del `mixin`. El nombre que usaremos deberá ser lo más semántico posible. Nombremos a este como `relative-converter`.

```
@mixin relative-converter
```

El siguiente paso es agregar `()`, en los cuáles agregaremos los valores que necesarios para convertir las unidades absolutas en relativas.

```
@mixin relative-converter()
```

Los valores que requerimos para hacer a este *mixin* lo más flexible posible son:

- El nombre de la propiedad a afectar con la unidad relativa
- La unidad en la que se transformará la unidad absoluta
- El valor a transformar
- El valor contextual de la unidad relativa

Estos valores los agregaremos dentro de los paréntesis usando variables de *Sass*.

```
@mixin relative-converter($property, $unit, $target, $context)
```

Cuando finalicemos abriremos llaves y dentro de ella definiremos una variable que guardará el resultado de la división entre la unidad a transformar y el valor asociado a la unidad a ser transformada.

```
@mixin relative-converter($property, $unit, $target, $context) {
  $result: ($target / $context);
}
```

¿Qué es un operador de *Sass*?

Este nueva característica que usamos de Sass es llamada `operación` y en esencia, nos permite hacer operaciones matemáticas básicas como suma, resta, multiplicación, división y porcentaje dentro de nuestras hojas de estilo *Sass*.

Para finalizar, deberemos crear la estructura que tendrá el resultado de nuestra conversión. Esto significa que el resultado deberá estar compuesto por la propiedad que deseamos usar y como valor de esta el resultado de la suma, junto a la unidad que tendrá el resultado.

El signo `+` nos permitirá unir el tamaño con su unidad. Por ejemplo, si tenemos un tamaño de `1.875` y la unidad es `rem`, entonces el resultado de tener `1.875 + rem` es `1.875rem`.

```
// Transforma px en rem
@mixin relative-converter($property, $unit, $target, $context) {
  $result: ($target / $context);

  #{$property}: #{$result + $unit};
}
```

Algo importante a tener en cuenta sobre este *mixin* es que con la configuración que le hemos dado sólo permitirá transformar unidades absolutas a unidades relativas al tamaño de fuente y no a las relativas al viewport, ya que nos faltaría convertir el resultado a porcentaje.

Aplicando *mixin* en proyecto

Ahora que hemos terminado de crear nuestro *mixin* nos toca poner en práctica nuestro conversor de medida. Probemos su uso transformando el tamaño de fuente de `<h1>` a unidad relativa.

Ingresemos al parcial `_typography` para cambiar este valor. Aquí encontraremos el tamaño base de los títulos usados en la maqueta. Dentro del `<h1>` eliminemos la propiedad `font-size` y agreguemos en su lugar `@include` y luego el nombre del *mixin* que en este caso es relative-converter, seguido de un paréntesis.

```
h1 {
  @include relative-converter(font-size, rem, 60, 16);
}
```

La directiva `@include` se usa para invocar o llamar a un *mixin* usando su nombre.

Al interior de los paréntesis agregaremos los argumentos requeridos del *mixin*. Si hacemos memoria los elementos que requería nuestro *mixin* era "*el nombre de la propiedad a afectar con la unidad relativa, la unidad en la que se transformará la unidad absoluta, el valor a transformar y el valor contextual de la unidad relativa*", o sea, el primer argumento es la unidad, es el nombre de la propiedad. En este caso será un `font-size` ya que, queremos cambiar el tamaño de la fuente.

El siguiente será la unidad relativa que queremos que se transforme. Este será `rem`.

El tercer argumento es la unidad que queremos transformar, o sea `60` y el cuarto es el valor relacionado al contexto que tiene la unidad relativa a convertir, o sea, `16`.

Es importante notar que para usar este mixin, el valor del tercer y cuarto argumento deberá contener sólo números, de otra manera la conversión no funcionará.

Asi mismo, en el caso de las fuentes, si queremos utilizar las variables de tamaño configuradas en el parcial `_variables`, debemos ir a este archivo y eliminar en cada una de estas la unidad de medida, a de modo de hacer fácil de mantener nuestros estilos.

Al recargar no veremos grandes cambios, pero si impeccionamos el título del sitio podremos ver que la unidad cambió correctamente a `3.75rem`.

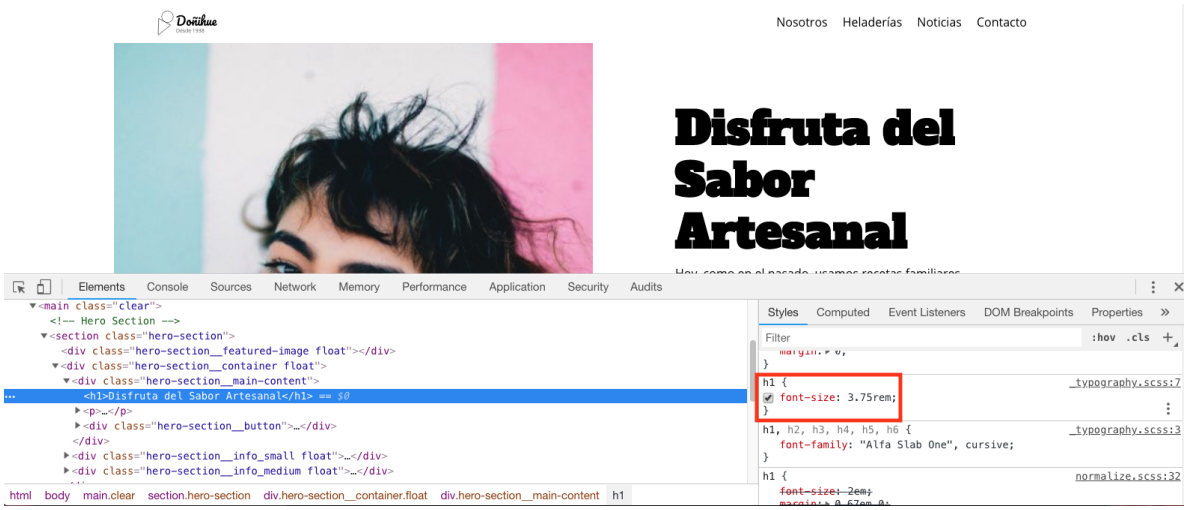

Imagen 9. Ejemplo del cambio de tamaño de título.

Les dejo como tarea seguir transformando las unidades de medidas de la maqueta o modo de identificar los efectos que provoca en los elementos esta transformación.

Lecturas complementarias

- Responsive Web Design - Ethan Marcotte, *A Book Apart*, pp. 15 -23
- MarkSheet - Sass Mixin

Componentes de un diseño responsivo: *Media Queries*

Para manejar los cambios en nuestro sitio web en los diferentes dispositivos existen las *Media Queries* que son una utilidad de css que nos permite decirle al navegador como se comporten nuestras clases bajo ciertas condiciones que para el caso del responsive son el ancho de nuestro dispositivo.

Luego de integrar el *meta viewport* y transformar las unidades absolutas a relativas nuestra maqueta sigue viéndose igual, pero aunque no se vea a simple vista ya tenemos dos grandes acciones realizadas.

En primer lugar, el *meta viewport* que integramos nos ayuda a decir al navegador el tamaño que deseamos que tenga la interfaz, que en este caso lo define este último.

En segundo lugar, los elementos que transformamos en unidades relativas darán la posibilidad al usuario de decidir el tamaño de fuente que verá en nuestro sitio.

De hecho, si vamos a las configuraciones del navegador y buscamos la opción de fuentes podremos ver que si aumentamos el tamaño de fuente el diseño de la maqueta fluirá uniformente mientras aumenta o disminuye la fuente, lo que hace que nuestra maqueta sea muy usable.

Sin embargo, la página aún no despliega el contenido de manera adecuada en dispositivos móviles, haciendo que la usabilidad que ganamos con las unidades relativas se pierda, pero ¿cómo podemos definir estilos en una resolución específica?, pues bien, usando un componente de diseño responsivo imprescindible llamado *media queries*.

¿Qué son los *media queries*?

Los *media queries* son una **directiva de *CSS*** que nos permitirá agregar estilos en ciertos tamaños sólo si la condición es cierta. Esto quiere decir, que si el navegador detecta que el tamaño de un dispositivo es de `375px` y existe un *media query* con ese tamaño, el navegador cargará el bloque de estilos y los mostrará en el sitio.

Reglas de la directiva `@media`

La estructura básica de un media query esta constituída por @media, el tamaño definido desde/hasta y el bloque de estilos asignado, a continuación se puede apreciar los cinco grupos establecidos por la website w3schools para los distintos dispositivos:

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}
```

```
/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px and up) */
@media only screen and (min-width: 1200px) {...}
```

- La directiva `@media`
- El bloque de estilos

Pero además, tendremos la opción de agregar algunos parámetros para hacer más especializado nuestro *media query*.

Entre ellos se encuentran:

- **Tipos de medio**: Con los tipos de medios podremos especificar el tipo medio al cual afectarán nuestros estilos.

Por ejemplo si queremos mostrar nuestros estilos solamente en pantallas usaremos el tipo de medio `screen` o si por el contrario queremos que agregue estilos específicos a las impresiones de nuestros sitios, podremos un medio tipo `print`.

```
@media screen {
  h1 {
    font-size: 30px
  }
}

@media print {
  h1 {
    font-size: 50px
  }
}
```

Los tipos de medios más comunes que podremos encontrar son `all`, `screen` y `print`.

- **Expresiones**: Por otra parte, las expresiones de un *media query* nos darán la posibilidad de usar alguna característica de del dispositivo a modo de activar o desactivar el *media query*.

Imaginemos por un momento que queremos disminuir el tamaño y el color de un `<h2>` cuando el sitio sea visto en un Smartphone.

Si queremos lograr este cometido deberemos agregar un *media query* que especifique el ancho del dispositivo a afectar. En este caso podríamos definirlo con un ancho de `350px` o si deseamos ser más flexibles, que el tamaño mínimo o máximo sea de `350px` usando un `min-width` o `min-width`.

```
@media screen (width: 350px) {
  h1 {
    font-size: 20px;
    color: #0077b3;
  }
}
```

- **Reglas lógicas**: Ahora bien, si queremos ser más específicos, podremos usar algunas palabras claves que permitan agregar más características.

Si usamos la palabra clave `and` al lado de podremos agregar más de dos tipos de medio o expresiones.

```
@media screen and (min-width: 300px) {
  p {
    color: blue;
  }
}
```

Otro regla que podremos usar es `only`, la cual nos permitirá prevenir que los navegadores antiguos no soporten nuestros media queries.

```
@media only screen and (min-width: 300px) {
  p {
    color: blue;
  }
}
```

Usando *media queries*

Ahora que ya conocemos los conceptos básico de los *media queries* es momento de conocer cómo se comportan realmente al usar en un contexto real.

Como vemos, el uso de *media queries* es fundamental para cualquier diseño responsivo que deseemos implementar ya que, con estos seremos capaces de modificar, crear o definir estilos en los tamaños que definamos.

Lecturas complementarias

- MDN - Media Queries