GUIA CSS DISEÑO DE GRIGILLAS, MEDIA QUERIES Y ANIMACIONES.

Al hablar de diseño de rejillas, nos referimos a la maquetación de una página web basada en filas y columnas lo cual nos permite ubicar de una forma más ordena y fácil las áreas que comprende nuestro sitio, actualmente el diseño de rejillas es también conocido como diseño con layouts o contenedores.

Inicialmente el diseño con rejillas se basaba únicamente en el distribuir el ancho total de la pantalla en 12 porciones de ancho iguales.

Etiquetas html que se pueden considerar como rejillas: , <main>, <aside>, <header>, <secction> ya que estos forman parte del grupo de contenedores.

En css para representar rejillas se usa la regla Grid.

CSS grid es un sistema de maquetación basado en grillas y se caracteriza por ser bidimensional, independiente del orden del markup y flexible.

Dentro de sus propiedades se encuentran:

- grid-template-areas: especifica grillas llamadas grid áreas.
- Los diseños de cuadrícula son bidimensionales: tienen una fila, o en línea, eje y una columna, o bloque, eje.
- justify-items: especifica cómo los elementos individuales deberían extenderse a lo largo del eje de la fila.
- justify-content: especifica cómo los grupos de elementos deberían distribuirse en el eje de la fila.
- justify-self especifica cómo un único elemento debe posicionarse con respecto al eje de fila.
- align-items especifica cómo los elementos individuales deben extenderse a lo largo del eje de la columna.
- align-content: específica cómo los grupos de elementos se deben distribuir a lo largo del eje de la columna.
- align-self: especifica cómo un único elemento debe posicionarse con respecto al eje de la columna.
- grid-auto-rows: especifica la altura de las filas agregadas implícitamente a la grilla.
- grid-auto-columns especifica el ancho de las columnas agregadas implícitamente a la grilla.
- grid-auto-flow: específica en qué dirección se deben crear los elementos implícitos.

MEDIA QUERIES CSS

Las Media Queries son una de las grandes ventajas de CSS3, ya que permiten saber qué sistema se está visualizando una página web y, en función de ello, aplicar unas reglas de estilo u otras. Así, podemos servir un CSS personalizado, acorde las condiciones del navegador o dispositivo que nos visita.

De una manera descriptiva podemos indicar una condición y a continuación los estilos que deben aplicarse cuando ésta se cumpla.

MEDIOS

Con Media Queries podemos detectar el medio donde se está consumiendo un sitio web.

all: Se utiliza para todos los tipos de medios o dispositivos.

–print: Se utiliza para impresoras

–screen: Se utiliza para pantallas de ordenador, tablets, teléfonos móviles, etc…

–speech: Se utiliza para lectores de pantalla, generalmente utilizados por personas con discapacidad visual.

FUNCIONES DE MEDIOS

Las funciones de medios o Media Features son las características específicas que otorgamos a cada user-agent, navegador o dispositivo de salida.

- width: Ancho de la ventana de visualización.
- height: Alto de la ventana de visualización.
- aspect-ratio: Relación de aspecto ancho/alto de la ventana de visualización.
- orientation: Orientación de la ventana de visualización .
- resolution: Densidad de píxeles del dispositivo de salida.
- scan: Proceso de escaneado del dispositivo de salida.
- grid: ¿Utiliza el dispositivo una pantalla de cuadrícula o mapa de bits?
- update: ¿Con qué frecuencia el dispositivo de salida puede modificar la apariencia del contenido?
- overflow-block: ¿Cómo maneja el dispositivo de salida el contenido que desborda la vista a lo largo del eje del bloque?
- overflow-inline: ¿Se puede desplazar el contenido que desborda la ventana de visualización a lo largo del eje en línea?
- color: Número de bits por componente de color del dispositivo de salida, o cero si el dispositivo no es de color.
- color-gamut: Rango aproximado de colores que son soportados por el user-agent y el dispositivo de salida.
- color-index: Número de entradas en la tabla de búsqueda de color del dispositivo de salida, o cero si el dispositivo no utiliza dicha tabla.
- display-mode: El modo de visualización de la aplicación.

- monochrome: Bits por píxel en el búfer de fotogramas monocromo del dispositivo de salida, o cero si el dispositivo no es monocromo.
- inverted-colors: ¿Está invirtiendo colores el agente de usuario o el sistema operativo subyacente?
- pointer: ¿Es el mecanismo de entrada principal un dispositivo señalador y, de ser así, cuán preciso es?
- hover: ¿El mecanismo de entrada principal permite al usuario pasar el ratón por encima de los elementos?
- any-pointer: ¿Es cualquier mecanismo de entrada disponible un dispositivo señalador y, de ser así, cuán preciso es?
- any-hover: ¿Hay algún mecanismo de entrada disponible que permita al usuario pasar el ratón por encima de los elementos?
- light-level: Nivel de luz del entorno.
- prefers-reduced-motion: El usuario prefiere menos movimiento en la página.
- prefers-reduced-transparency: El usuario prefiere una transparencia reducida.
- prefers-contrast: Detecta si el usuario ha solicitado que el sistema aumente o disminuya la cantidad de contraste entre los colores adyacentes.
- prefers-color-scheme: Detecta si el usuario prefiere un esquema de color claro u oscuro.
- forced-colors: Detecta si el user-agent restringe la paleta de colores.
- scripting: Detecta si está disponible el scripting (es decir, JavaScript).

EJEMPLO

```
@media screen {
 h1 {
 color: red;
 }
 }
@media print {
 h1 {
 color: black;
 }
 }
```

Tamaños de pantalla

Lo más común es que usemos las Media Queries para detectar las dimensiones de la pantalla.

```
@media (min-width: 1000px) {
 body {
 align: 20px;
 }
 div {
 text-align: center;
 }
 }

@media (max-width: 320px) {
 body {
 font-size: .6em;
 }
 }
```

En este caso, indicamos que la anchura de la pantalla debe de ser 320 píxeles de máximo para que se apliquen los estilos.

Orientación

También es útil para definir estilos en función de la orientación, lo que resulta especialmente adecuado para el diseño para móviles.

```
@media (orientation:landscape){
 body{
 background-image:url(archivo.jpg);
 }
 }
```

```
@media (orientation:portrait){

body{

background-image:url(archivo-pantalla-en-vertical.jpg);

}

}
```

Landscape es el valor de la orientación cuando la pantalla está en horizontal y portrait para la pantalla en vertical.

EJEMPLO GENERAL:

Planteamiento del ejercicio: Crear un página para mostrar el perfil profesional de cada uno siguiendo los ejemplos que aquí se van ir realizando. En el desarrollo del ejemplo se irán mencionando elementos css que no se habían usado en las clases ni en prácticas anteriores.

Crear un directorio para nuestro proyecto deberán colocar su número de carnet como carpeta base, dentro de ella un archivo index.html, una carpeta para imágenes, una carpeta para css y una carpeta para código JavaSript.

Abrir el archivo index.html y crear la estructura básica del archivo html.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>


  </body>
</html>
```

Sera necesario definir las áreas que nuestro perfil llevará, para el ejemplo utilizaremos:

Main: será el contenedor principal.

Header: se colocará el título de nuestro perfil.

Nav: aquí especificaremos las tecnologías con las que hemos trabajado.

Secction: Mostrará una galería de imágenes y descripción de los trabajos que hemos desarrollado y en las empresas que hemos trabajado.

Aside: Se colocará la información personal.

El código html quedara como sigue:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <main>
      <header>

      </header>
      <nav>

      </nav>
      <section>

      </section>
      <aside>

      </aside>
      <footer>

      </footer>
    </main>

  </body>
</html>
```

Crear la plantilla desde css para lo cual será necesario crear la siguiente estructura de código.

- Crear archivo css con el nombre style.css con el siguiente contenido inicial

```css
* {
  box-sizing:border-box;
  padding: 0px;
}
```

Crear plantilla inicial:

```css
main{
  margin: 0;
  display: grid;
  max-width: 100%;
  min-height: 100vh;
  grid-template-columns: 20% 1fr 20%;
  grid-template-rows: 200px 1fr 200px;
  grid-template-areas:
  "encabezado encabezado encabezado"
  "menu seccion info"
  "foo foo foo"
}
```

Asignar las áreas a cada etiqueta:

```css
header{
  grid-area: encabezado;
  background: red;
}
nav{
  grid-area:menu;
  background: green;
}
section{
  grid-area: seccion;
  background: blue;
}
aside{
  grid-area: info;
  background: yellow;
}
footer{

  grid-area: foo;
  background: black;
}
```

El resultado sería el siguiente:

EXPLICACION BOX-SIZING

La propiedad box-sizing puede ser usada para ajustar el siguiente comportamiento:

- content-box es el comportamiento CSS por defecto para el tamaño de la caja (box-sizing). Si se define el ancho de un elemento en 100 pixeles, la caja del contenido del elemento tendrá 100 pixeles de ancho, y el ancho de cualquier borde o relleno ser añadirá al ancho final desplegado.
- border-box le dice al navegador tomar en cuenta para cualquier valor que se especifique de borde o de relleno para el ancho o alto de un elemento. Es decir, si se define un elemento con un ancho de 100 pixeles. Esos 100 pixeles incluirán cualquier borde o relleno que se añadan, y la caja de contenido se encogerá para absorber ese ancho extra. Esto típicamente hace mucho más fácil dimensionar elementos.

Aplicar diseño para el encabezado:

Código html:

```
<header>

  <h1>Mi perfil profesional</h1>

</header>
<nav>
```

Código css:

```css
header{
  background: #284466;
  grid-area: encabezado;
  text-align: center;
  margin-bottom: 30px;
  box-shadow: 5px 10px 5px 1px #e2e2e2;
}
header h1{
  font-family: 'Noto Sans', sans-serif;
  font-size:35px;
  color: white;
}
```

Vista

Aplicar el estilo a las tecnologías que se manejan:

Código html:

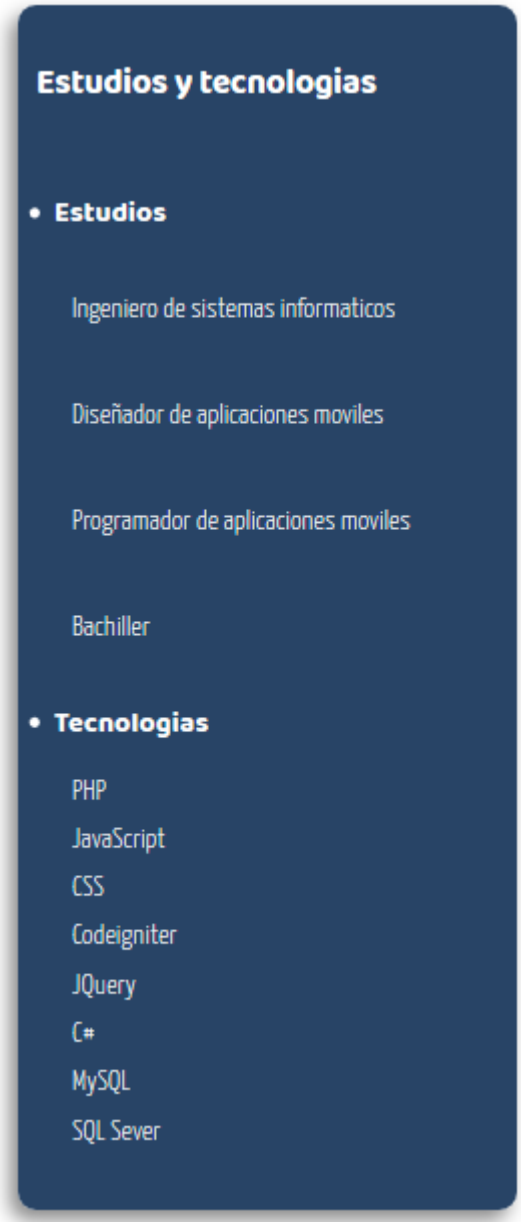
```html
<nav>
  <div class="row">
    <h3 class="txtTitulos">Estudios y tecnologias</h3>
  </div>
  <div class="row estudio">
    <ul class="lista">
      <li><p class="txtTitulos">Estudios</p>
        <ul class="subLista">
          <li>
            <p class="tooltip">Ingeniero de sistemas informaticos<span class="tooltipText">Aqui puede ir mas contenido</span></p></li>
          <li><p class="tooltip">Diseñador de aplicaciones moviles<span class="tooltipText">Aqui puede ir mas contenido</span></p></li>
          <li><p class="tooltip">Programador de aplicaciones moviles<span class="tooltipText">Aqui puede ir mas contenido</span></p></li>
          <li><p class="tooltip">Bachiller<span class="tooltipText">Aqui puede ir mas contenido</span></p></li>
        </ul>
      </li>
      <li><p class="txtTitulos">Tecnologias</p>
        <ul class="subLista">
          <li>PHP</li>
          <li>JavaScript</li>
          <li>CSS</li>
          <li>Codeigniter</li>
          <li>JQuery</li>
          <li>C#</li>
          <li>MySQL</li>
          <li>SQL Sever</li>
        </ul>
      </li>
    </ul>
  </div>



</nav>
```

Código css:

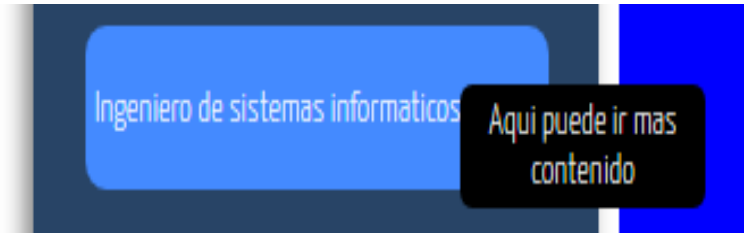
```css
nav{
  background: #284466;
  margin: 10px;
  padding: 10px;
  grid-area:menu;
  box-shadow:  -5px 5px 10px #888888;
  border-radius: 10px;
  color:white;
}

.estudio{
  padding: 10px;
}
.subLista{
  p adding: 0px;
}
.subLista li{
  display: block;
  margin: 10px;
  font-family: 'Yanone Kaffeesatz', sans-se
}
.txtTitulos{
  font-family: 'Baloo 2', cursive;
}
.subLista li:hover{
  display: block;
  background: #448AFF;
  padding: 5px;
  margin: 5px;
  border-radius: 10px;
}
.tooltip{
  position: relative;
  display: inline-block;
}
.tooltip .tooltipText{
  visibility: hidden;
  width: 120px;
  background-color: black;
  color: #fff;
  text-align: center;
  border-radius: 6px;
  padding: 5px 0;

  /* posicionamiento */
  position: absolute;
  z-index: 1;
}
.tooltip:hover .tooltipText {
  visibility: visible;
}
```

Se agrega una herramienta llamada tooltip la cual sirve para mostrar información extra.

Vista:



Vista tooltip:



Creación de galería para mostrar empresas donde se ha trabajado

Para ello haremos uso de nuevo de la herramienta grid.

Código html:

```
<section>
  <h2 class="miExperencia">Mi experiencia</h2>
  <div class="containerGalery">
    <div class="card"><img src="./img/kdevjo.jpg" alt="">
      <div class="containerText">
        <p>Kadevjo S.A. de C.V.</p>
        <p>Desarrollador movil</p>
        <p>fecha</p>
        <button class="btn" type="button" name="button">Visitar Pagina</button>
      </div>

    </div>
  <div class="card"><img src="./img/cardaya.png" alt="">
    <div class="containerText">
      <p>New Star</p>
      <p>Asistente de informatica</p>
      <p>fecha</p>
      <button class="btn" type="button" name="button">Visitar Pagina</button>
    </div>
  </div>
  <div class="card"><img src="./img/surti.png" alt="">
    <div class="containerText">
      <p>Surti Marcas</p>
      <p>Asistente de informatica</p>
      <p>fecha</p>
      <button class="btn" type="button" name="button">Visitar Pagina</button>
    </div>
  </div>
  <div class="card"><img src="./img/logo-ferreteria.jpg" alt="">
    <div class="containerText">
      <p>Ferreteria el Roble</p>
      <p>Progamador</p>
      <p>fecha</p>
      <button class="btn" type="button" name="button">Visitar Pagina</button>
    </div>
  </div>
  <div class="card"><img src="./img/original.png" alt="">
    <div class="containerText">
      <p>GOCAJAA GROUP S.A. de S.V.</p>
      <p>Programador</p>
      <p>fecha</p>
      <button class="btn" type="button" name="button">Visitar Pagina</button>
    </div></div>
  <div class="card"><img src="./img/logo-ues.png" alt="">
    <div class="containerText">
      <p>UES FMO</p>
      <p>Docente</p>
      <p>fecha</p>
      <button class="btn" type="button" name="button">Visitar Pagina</button>
    </div></div>
  </div>
```

Código css:

```css
section{

  grid-area: seccion;

}
.miExperencia{
  display: block;
  text-align: center;
  font-family: 'Source Sans Pro', sans-serif;
}
.containerGalery{
  text-align: center;
  margin: 0 auto;
  display: grid;
  grid-gap: 1rem;
  grid-template-columns: 250px 250px 250px;
  grid-template-rows:auto auto;
  grid-template-rows: minmax(30%, 1.5fr) minmax(30%, 1.5fr);
  justify-content: center;
}
img{
  object-fit: fill;
  width: 90%;
  height: 50%;
  border: 10px;
}
.card{
  padding: 5px;
  margin: 0 auto;
  box-shadow:  -5px 5px 10px #888888;
  border-radius: 10px;
}
.containerText{
  text-align: center;
  font-family: 'Yanone Kaffeesatz', sans-serif;
}
.btn{
  margin: 0 auto;
  display: block;
  width:80%;
  border: none;
  padding: 5px;
  background: #284466;
  color:white;
  border-radius: 10px;
}
```

Propidad object-fit: permite que un elemento se pueda adaptar a las dimensiones de su contenedor, puede recibir los siguientes parámetros:

Contain: El contenido reemplazado está dimensionado para mantener su relación de aspecto mientras se ajusta dentro del cuadro de contenido del elemento: su tamaño de objeto concreto se resuelve como una restricción de contenido contra el ancho y la altura utilizados del elemento.

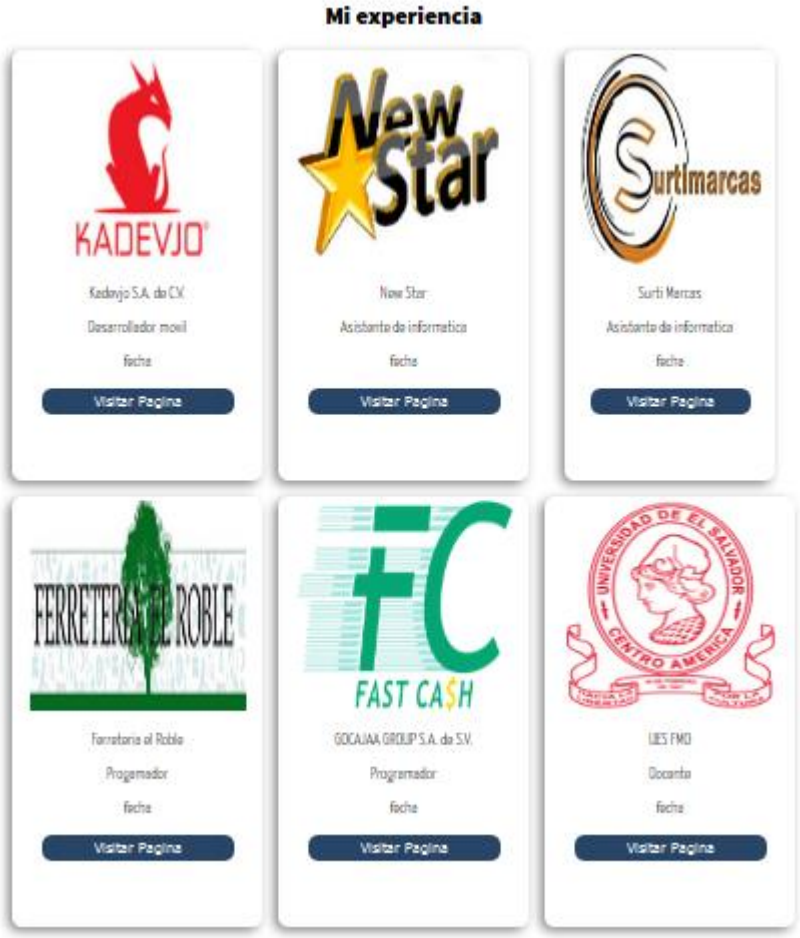
Cover: El contenido reemplazado se dimensiona para mantener su relación de aspecto mientras llena el cuadro de contenido completo del elemento. Si la relación de aspecto del objeto no coincide con la relación de aspecto de su caja, entonces el objeto se recortará para que se ajuste.

Fill: Modifica el tamaño del elemento remplazado para llenar el cuadro de contenido. El objeto completo ocupará todo el espacio de la caja. Si el tamaño del elemento no concuerda con el de su caja, se estirará para llenarlo.

None: El contenido reemplazado no se redimensiona.

scale-down: El contenido se dimensiona como si none o contain estuvieran especificados, lo que resultaría en un tamaño de objeto concreto más pequeño.

Vista:



Agregar información personal:

Código html:

```html
<aside>
  <div class="tituloInfo">
    <h2>Mi información personal</h2>
  </div>
  <div class="img">
    <img class="image" src="./img/fotoPerfil.jpg" alt="">
  </div>
  <div class="infoPersonal">
    <p class="itemInfo"><i class="fa fa-user"></i> Nombre:<span > Hector Javier Paiz Ramos. </span></p>
    <p class="itemInfo"><i class="far fa-calendar-alt"></i> Edad: <span>26 años</span></p>
    <p class="itemInfo"><i class="fa fa-map "></i> Dirección:<span> San Miguel</span></p>
    <p class="itemInfo"><i class="fas fa-envelope"></i> Correo: <span>hpaizramos@gmail.com</span></p>
  </div>
</aside>
```

Código css:

```css
aside{
  grid-area: info;
  margin: 5px;
  padding: 5px;
  text-align: center;
  box-shadow:  -5px 5px 10px #888888;
  border-radius: 10px;

}
.image{
  height: 20%;
  width: 60%;
  border-radius: 95px;
  margin-top: 10px;
}
.infoPersonal{
  font-family: 'Source Sans Pro', sans-serif;
}
.tituloInfo{
  display: block;
  background: #00796b;
  padding: 5px;
  border-radius: 20px;
  color: white;
  font-family: 'Source Sans Pro', sans-serif;
}
.itemInfo{
  display: block;
  background: #00796b;
  padding: 5px;
  border-radius: 20px;
  color: white;
}
```

Vista:

Paso 5: finalizar el diseño del footer:

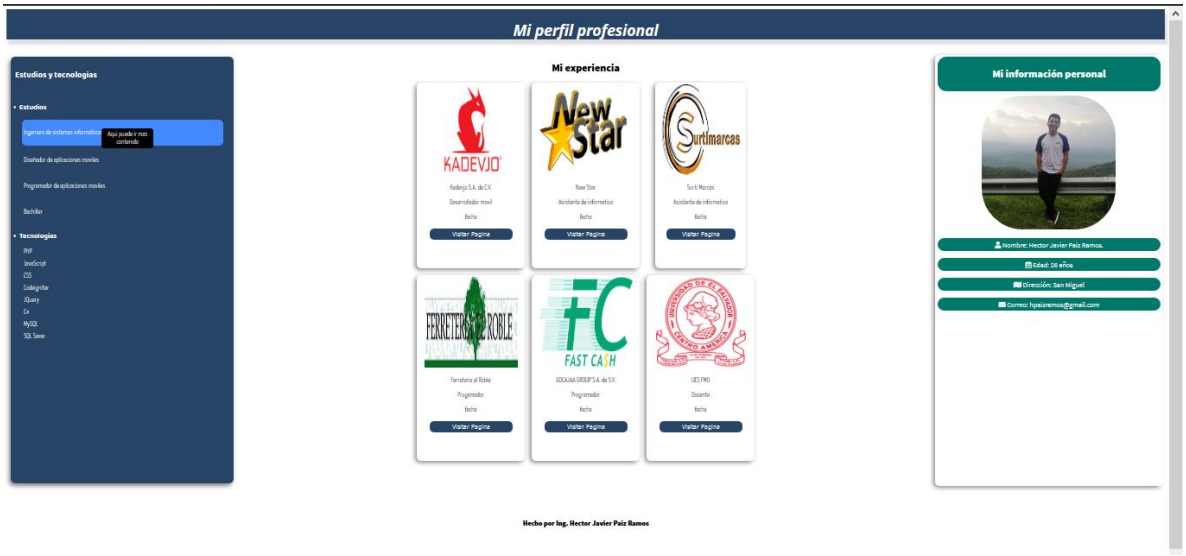
Código html:

```
<footer>
  <h4>Hecho por Ing. Hector Javier Paiz Ramos</h4>
</footer>
```

Código css:

```css
footer{
  margin-top: 50px;
  text-align: center;
  grid-area: foo;
  font-family: 'Source Sans Pro', sans-serif;
}
```

Vista final del diseño en general:



Aplicando media querie para diseño adaptable:

Lo primero que tenemos que hacer es adaptar la grilla principal:

Código css:

```css
@media only screen and (max-width:700px){

  main{
    display: grid;
    grid-template-columns: 100%;
    grid-template-rows: auto;
    grid-template-areas:
    "encabezado"
    "menu"
    "seccion"
    "info"
    "foo"
  }

}
```

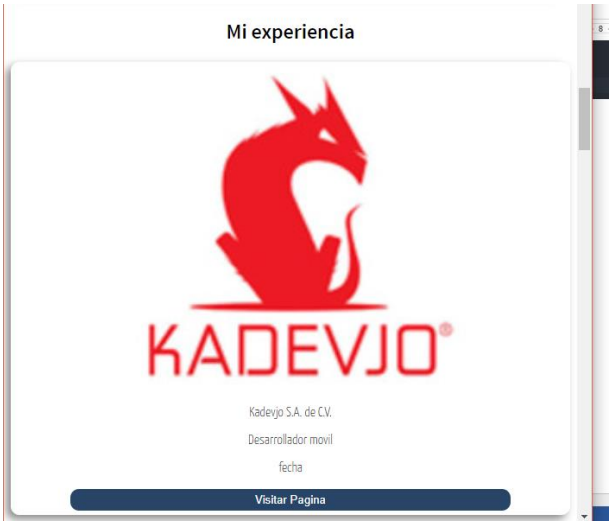
Vista:

Ahora corresponde crear la media querie para la galería

Código css(este código deberá ir dentro de la media querie creada anteriormente)

```
.containerGalery{
  width: 100%;
  display: grid;
  grid-template-columns: 100%;
  grid-template-rows: auto;
}
.card{
  width: 100%;
}
```

Vistas:



Aplicando animaciones y transiciones en css:

Para añadir animaciones es necesario conocer ciertas reglas de css con lo cual vamos a iniciar con las transformaciones:

Las transformaciones nos permiten en css especificar qué tipo de efectos puede tener cualquier elemento

Para poder aplicar transformaciones a un elemento html desde css es necesario aplicar la regla llamada transform, de esta depende otras reglas las cuales son transform-origin y transform-style.

La regla transform recibe los siguientes parámetros:

- translate: este permite mover un objeto de un punto a otro en la pantalla, esta funcion recibe dos parámetros los cuales indican el primer valor los pixeles que el elemento moverá en el eje x y el segundo los pixeles que el elemento se moverá en el eje y, si el primer valor es menor que cero es decir un numero negativo el elemento se mueve hacia la izquierda y si el número es positivo elemento se mueve hacia la derecha, si el segundo número es menor que se cero el elemento se mueve hacia abajo y si es positivo se mueve hacia arriba, también se pueden expresar valores específicos translateX y translateY;
- scale: esta función aumenta o reduce el tamaño del elemento al que se aplica dicha regla también recibe dos parámetros el primero para el eje x y el segundo para el eje y.
- rotate: permite girar un elemento un número de grados indicados.
- Skew: permite deformar un elemento, es decir recibe como parámetro un ángulo en el que el elemento se pueda deformar.

También estas funciones se pueden utilizar para aplicar transformaciones 3D se utilizan las mismas funciones pero se agrega un parámetro más para el eje z el cual añade profundidad en nuestra página.

Una vez conociendo las transformaciones es necesario conocer las transiciones para poder llegar a las animaciones en concreto.

Una transición nos permite tener un efecto suavizado entre un estado inicial y un estado final. Para poder hacer uso de ellas necesitamos conocer las siguientes reglas de css:

- transition-property: nos indica que propiedad será la que se va a modificar.
- transition-duration: especificamos el tiempo que se tomará desde el inicio de la transición hasta su finalización, el valor que recibe como parámetro es el tiempo se recomienda utilizar cantidades pequeñas las unidades que toma son segundos.

Valor	Inicio	Transcurso	Final	Equivalente en cubic-beizer
ease	Lento	Rápido	Lento	(0.25, 0.1, 0.25, 1)
linear	Normal	Normal	Normal	(0, 0, 1, 1)
ease-in	Lento	Normal	Normal	(0.42, 0, 1, 1)
ease-out	Normal	Normal	Lento	(0, 0, 0.58, 1)
ease-in-out	Lento	Normal	Lento	(0.42, 0, 0.58, 1)
cubic-bezier(*A*, *B*, *C*, *D*)	-	-	-	Transición personalizada

- Transition-timing-function: permite indicar el tipo de transición que queremos conseguir la cual puede tomar los siguientes valores:

- Transition-delay: permite retrasar el inicio de la transición.

Una vez conocemos las transiciones CSS3, es muy fácil adaptarnos al concepto de animaciones CSS3, el cual amplia el concepto de transiciones convirtiéndolo en algo mucho más flexible y potente.

Pero antes es importante conocer el termino @keyframes.

La regla arroba @keyframes permite a los autores controlar los pasos intermedios en una secuencia de animación CSS mediante el establecimiento de keyframes (o puntos de trayectoria) a lo largo de la secuencia de animación que debe ser alcanzado por determinados puntos durante la animación. Esto le da un control más específico sobre los pasos intermedios de la secuencia de animación que se obtiene al dejar que el navegador maneje todo automáticamente.

Para utilizar keyframes, se crea una regla de @keyframes con un nombre que es utilizada por la propiedad animation-name para que coincida con una animación de keyframe a su lista. Cada regla @keyframes contiene una lista de estilo de selectores de keyframe, cada una de los cuales está compuesto de un porcentaje a lo largo de la animación en la que se produce el keyframe así como un bloque que contiene la información de estilo para ese keyframe.

Ejemplo:

```
@keyframes identifier {
  0% { top: 0; left: 0; }
  30% { top: 50px; }
  68%, 72% { left: 50px; }
  100% { top: 100px; left: 100%; }
```

}

Las propiedades de las animaciones son:

Propiedades	Valor
animation-name	**none** | *nombre*
animation-duration	**0** | TIEMPO
animation-timing-function	**ease** | linear | ease-in | ease-out | ease-in-out | cubic-bezier(*A* , *B* , *C* , *D*)
animation-delay	**0** | TIEMPO
animation-iteration-count	**1** | infinite | *número*
animation-direction	**normal** | reverse | alternate | alternate-reverse
animation-fill-mode	**none** | forwards | backwards | both
animation-play-state	**running** | paused

La propiedad animation-name permite especificar el nombre del fotograma a utilizar.

Las propiedades animation-duration, animation-timing-function y animation-delay funcionan exactamente igual que en el tema de transiciones.

La propiedad animation-iteration-count permite indicar el número de veces que se repite la animación, pudiendo establecer un número concreto de repeticiones o indicando infinite para que se repita continuamente.

La propiedad animation-direction permite indicar el orden en el que se reproducirán los fotogramas, pudiendo escoger un valor de los siguientes:

Valor	Significado
`normal`	Los fotogramas se reproducen desde el principio al final.
`reverse`	Los fotogramas se reproducen desde el final al principio.
`alternate`	En iteraciones par, de forma normal. Impares, a la inversa.
`alternate-reverse`	En iteraciones impares, de forma normal. Pares, normal.

La propiedad animation-fill-mode podemos indicar que debe mostrar la animación cuando ha finalizado y ya no se está reproduciendo; si mostrar el estado inicial (backwards), el estado final (forwards) o una combinación de ambas (both).

La propiedad animation-play-state nos permite establecer la animación a estado de reproducción (running) o pausarla (paused).

Aplicando una animación al proyecto:

Código css:

```css
.animacion1{
  transition-duration: 2s;
  animation-name: ejemplo;
  animation-duration: 2s;
  position: relative;

}

@keyframes ejemplo {
  from{left: 0px; top: 100px;}
  to  {left:0px; top:0px;}
}
```

Aplicar este elemento a todas las áreas del proyecto para ver su funcionamiento

Aplicando transiciones al hacer hover sobre la foto de perfil.

Agregar la siguiente línea en la clase .image del proyecto:

transition-duration: 3s;

Código css nuevo:

```css
.image:hover{
  transform: rotateY(360deg)  scale(2);
}
```

Para ver el funcionamiento de este tipo de animaciones es necesario que lo hagas en tu proyecto.

Para más ejemplos de referencia pueden visitar estas páginas:

w3schools.com: https://www.w3schools.com/css/css3_animations.asp

MDN web Docs: https://developer.mozilla.org/es/docs/Web/CSS/

Lenguajecss.com: https://lenguajecss.com/p/css/animaciones/animaciones