

Erick Flores Pérez

Tabla de contenido

1. INTRO FRONTEND	4
TENOLOGIAS UTILIZADAS DURANTE EL CURSO	4
SETUP FRONT END	5
Herramientas de diseño de aplicación	5
IDE de programación	6
Navegador Web	7
Developer tools	7
Documentación	7
ESTRUCTURA WEB	8
Modelo jerárquico	8
Modelo secuencial	8
Modelo de matriz	8
ESTRUCTURA DEL CÓDIGO	9
Estructura por tipo de dato	9
Estructura por funcionalidad	9
Estructura por ruta	9
PRACTICA “ABOGABOT”	10
Descripción	10
Toma de requerimientos	10
Buyer persona	11
Público objetivo	12
Wireframe	12
2. HTML	13
FUNCIONAMIENTO Y ESTRUCTURA DE HTML	13
Estructura HTML	13
ETIQUETAS BÁSICAS	14
Headings	14
Párrafos	14
Enlaces (links)	14

Imágenes.....	15
Tablas.....	15
Lista.....	15
Practica etiquetas básicas.....	16
ELEMENTOS COMPUESTOS. INPUTS.....	16
Entrada tipo texto.....	16
Etiquetas (Labels).....	16
Input de correo.....	16
Input de numero.....	17
Input de password.....	17
Input de teléfono.....	17
Input de URL.....	17
Practica Elementos Compuestos.....	17
ACOMODO Y NAVEGACIÓN. LAYOUT Y RUTAS.....	17
Practica Layout y rutas.....	17
PRACTICA “TAQUERIA”.....	17
Descripción.....	17
Metodología.....	18
3. CSS.....	19
SELECTORES Y FONDOS.....	19
Colores.....	20
Fondos.....	21
Textos.....	22
FLEXBOX.....	23
RESPONSIVE CON CSS.....	24
PRACTICA CSS.....	25
Descripción.....	25
4. JAVASCRIPT.....	26
CONCEPTOS BÁSICOS.....	26
Variables.....	26
Strings.....	26
Condicionales.....	26
Operadores lógicos.....	27

Arreglos	27
Objetos	27
Flujo de condicional	28
Ciclos	28
Switch case	29
FUNCIONES	30
Función con retorno	30
Función sin retorno	30
Múltiples entradas	30
Funciones flecha	31
Excepciones	31

1. INTRO FRONTEND

TECNOLOGIAS UTILIZADAS DURANTE EL CURSO



HTML

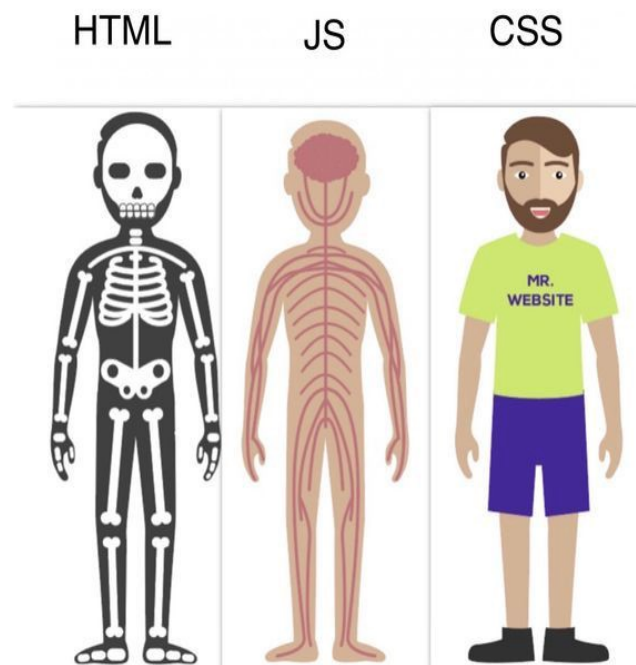
Este lenguaje nos permite tener el esqueleto de la aplicación, es lo que define la estructura del sitio y lo que da la pauta e inicio de la aplicación web.

CSS

Este lenguaje da la posibilidad de estilizar y de insertar toda la parte de visual y estética al sitio como si fuera la “piel” de la aplicación. Se utilizan clases y selectores para poder definir las propiedades y características de cada uno de los elementos.

JavaScript

Es el cerebro de la plataforma, una vez que se utiliza JS en el sitio se obtiene la capacidad de escalar las funcionalidades de forma exponencial, ya que se pasa de las propiedades que tienen las etiquetas (Que también tienen algo de JS) a tener una cantidad virtualmente infinita de posibilidades.



Frameworks

Los Frameworks son variantes de JS que ayudan a que la programación pueda llegar a ser mucho más rápida o con algunas funcionalidades adicionales a Vanilla JS, esto aplica tanto para FrontEnd como para Backend.

El hecho de que un Framework te permita programar de forma más sencilla NO SIGNIFICA que puedes saltarte toda la parte fundamental de JS, esto es porque al ser la base de muchos de los Frameworks web, es bastante útil conocer cómo funciona desde el fondo.

Existen muchos Frameworks y librerías muy famosas, pero por mencionar algunas de FrontEnd están React JS, Vue JS, Angular, Ember JS, Svelte, entre otros.

SETUP FRONT END

Herramientas

- Herramienta de diseño de aplicación
- IDE de programación
- Navegador web
- Diferentes opciones de "Developer tools"
- Herramienta de documentación

Herramientas de diseño de aplicación

Brinda la capacidad de tener un prototipo sin poner una sola línea de código.

En el flujo de programación, basándonos en el SDLC (Software Development Life Cycle) siempre es necesario empezar con el diseño de la solución antes de irnos directo a programarla, esto se hace así porque en caso de haber un cambio o modificación es mucho más fácil y rápido cambiarla en el lado de diseño que volver a programar el módulo completo.

La primera fase es el **WireFrame** que permite entender las necesidades y empezar a dibujar cómo funciona la aplicación sin darle diseño gráfico aún, esta es una fase completamente funcional para darnos cuenta de las interacciones, los botones necesarios, el número de clicks y los diferentes movimientos de la aplicación. Nos dará mucha información para que la aplicación sea lo más fluida posible.

Para el **WireFrame** existen diferentes herramientas como:

- Miro (<https://miro.com/es/>) Esta herramienta te permitirá crear flujos de información y flujos de negocio.
- Balsamic mockups (<https://balsamiq.com/wireframes/>) Nos da elementos para poder crear interfaces rápidas y que nos dejan representar la idea.
- Dibujos a mano alzada --> a veces un wireframe también puede ser en una servilleta o en cualquier forma escrita.

Ya con el **WireFrame** realizado lo siguiente es pasar a la etapa de **UI/UX** (Interfaz de usuario y experiencia de usuario). En esta etapa se comienza con toda la parte gráfica de identidad y uso de colores, formas e interacciones a lo largo de la aplicación. Para esto se utilizan las siguientes herramientas:

- Sketch (<https://www.sketch.com/>) Esta herramienta solo aplica para MacOS, pero cuenta con una amplia gama de características enfocadas a diseñadores, lo que hace que se puedan crear cosas muy grandes y escalables.
- Figma (<https://www.figma.com/>) Es una herramienta gratuita en muchas de sus características y permite colaboración entre usuarios, lo que es muy útil cuando se trabaja en equipo, tanto en equipos de diseño como en equipos de desarrollo.
- Adobe XD (<https://www.adobe.com/mx/products/xd.html>) Existe su versión de prueba gratuita que permite ocupar la herramienta completa para crear todos los diseños y el costo para tener un plan pro no es muy alto, tiene la ventaja de estar basado en las herramientas de Adobe, por lo que si ya tienes experiencia con alguna de estas herramientas, la curva de aprendizaje podría ser más corta.



IDE de programación

Un IDE (Integrated Development Environment) es un software que ayuda a los desarrolladores a poder construir aplicaciones de forma más sencilla, esto es debido a que combina diferentes herramientas de desarrollo en una sola interfaz gráfica.

- Editor de código: Permite resaltar sintaxis y encontrar algunos problemas dependiendo del lenguaje o autocompletado para que la experiencia se más eficaz.
- automatización: Algunos IDEs tienen la posibilidad de automatizar tareas repetitivas como el despliegue del programa, la compilación e incluso la actualización de versiones.
- Depuración: Hay algunos IDEs que ayudan a identificar errores antes de compilar, e incluso hay algunos que dan la solución o recomendaciones al problema.
- Integración: Existen herramientas que permiten integrar el IDE de forma directa con la aplicación como los son las interfaces, versionamientos de código o herramientas de despliegue.

IDEs recomendados

- VS Code
- Atom
- Sublime Text

Navegador Web

Como la aplicación que se programe se despliega en un navegador web es importante tener cuidado en revisar que la tecnología que se está utilizando es compatible y se despliega de forma igual o similar en todos los navegadores. Para esto existen plataformas como “Can I use” <https://caniuse.com/>, la cual indica qué navegadores se pueden utilizar con la tecnología de HTML, CSS y JS.

Developer tools

Este caso se refiere a las herramientas adicionales que dan información sobre la aplicación y mejoras que se pueden implementar. Una de las más conocidas es Lighthouse (<https://developers.google.com/web/tools/lighthouse?hl=es>) , se puede encontrar directamente en el navegador y brinda una auditoría del status de nuestro sitio en diferentes áreas como rendimiento, accesibilidad, buenas prácticas y SEO.

Es importante tomar en cuenta también a las personas que tienen alguna debilidad visual, problema motriz, o algún otro tipo de limitante. Es nuestro deber como desarrolladores integrar las facilidades de accesibilidad a nuestras plataformas contribuyendo así a la democratización tecnológica. Para esto existen herramientas específicas que dan todas las guías y usos adecuados de accesibilidad para los usuarios como axe DevTools (<https://www.deque.com/axe/devtools/>), que se puede instalar como una extensión en el navegador y que brinda mucha información sobre cuáles son los problemas en el desarrollo, pero más importante, información sobre cómo solucionarlos.

Documentación

La documentación de los desarrollos es de suma importancia cuando se habla de un ámbito profesional ya que el proyecto no es considerado a menos que esté bien documentado. La falta de documentación hace que la mantenibilidad del código sea mucho más complicada y que muchas veces se tengan que reprogramar módulos pues en promedio si no usas un código en aproximadamente 6 meses, se deja de comprender su funcionalidad.

Para esto existen librerías como API Doc (<https://apidocs.com/>) o rails Admin (https://github.com/railsadminteam/rails_admin) que dan facilidades para documentar mientras se programa.

Es importante considerar lo siguiente al momento de documentar el código:

- Definir audiencia
- Problemática que soluciona el código
- Estructura de la aplicación

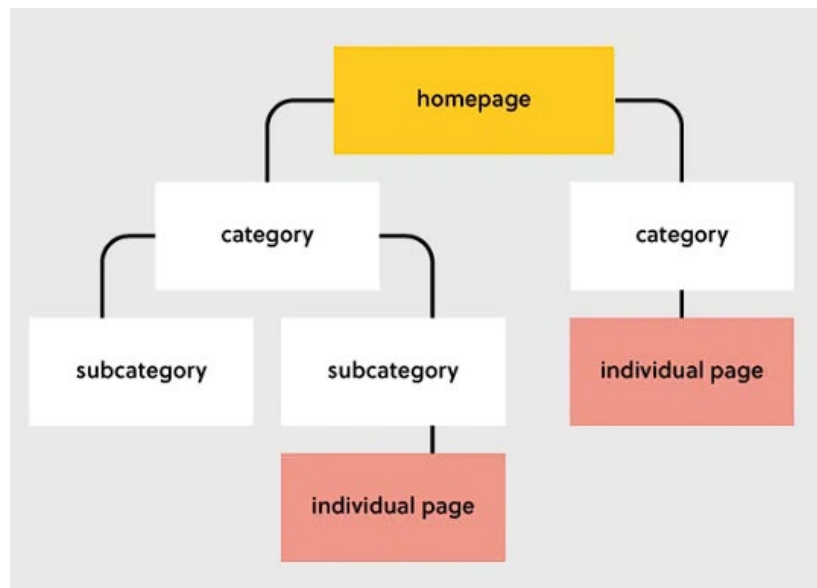
Otra herramienta muy útil para poder lograr una buena estructura es Notion (<https://www.notion.so/>) y Confluence (<https://www.atlassian.com/es/software/confluence>)

ESTRUCTURA WEB

Modelo jerárquico

Un ejemplo sencillo de las **categorías y subcategorías** podría ser una tienda de ropa en línea, en este tipo de tiendas tenemos categorías de productos que pueden ser "zapatos", "chamarras", etc. y las subcategorías podrían ser "niños", "mujer", "hombre", "Tallas chicas", etc. La idea de la organización hace que el uso de tu aplicación sea más sencilla para el usuario.

Las **páginas individuales** son las páginas finales a donde puede llegar tu usuario, a partir de aquí ya no hay más jerarquía y es en donde descansa la mayor parte de la información valiosa de tu plataforma. Estas páginas deben de tener contenido significativo, por ejemplo; puede ser que compartan la publicación, que compren el producto o simplemente que hayan encontrado lo que buscan en tu plataforma.



Modelo secuencial



Dirige al usuario por diferentes pasos secuenciales los cuales tienen un orden de finalización antes de poder pasar a la etapa siguiente. Es utilizado cuando necesitas que el usuario vaya por un flujo de información sin salirse del carril.

Modelo de matriz

Permite una navegación general por todo el sitio, en la que normalmente se tienen muchas opciones de información específica, la cual es compleja de agrupar en categorías, estos sitios por lo general utilizan links internos del mismo sitio para poder dirigirse a las secciones deseadas. Uno de los mejores ejemplos de este modelo es Wikipedia.

ESTRUCTURA DEL CÓDIGO

BAD!

```
//take a half-second nap...
time_nanosleep(0, 500000000);
} //End Sleep Delay
$timer++; //Time increment
} else {
    $this->db->where('sequence_id', $sequence_id)->insert('sequence_id', $sequence_id);
    echo 'Limit per-day reached<br />';
    echo 'Finished!';
    die();
} //End Timer
} //End articles loop
} else {
    //Update Sequence
    $this->db->where('sequence_id', $sequence_id)->insert('sequence_id', $sequence_id);
    echo 'Next Sequence...<br />';
} //End max_posts if statement
} else {
    echo 'Today is not a run day...<br />';
} //End day check
} else {
    //no log? insert one!
    $this->db->set('sequence_id', $sequence_id)->insert('sequence_id', $sequence_id);
    echo 'Created New Log for Sequence...<br />';
} //End Sequence Log check
End todays date check
ke a half-second nap...
```

GOOD!

```
//Get Sequences
$SequenceData = $this->_getSequences();
//Loop Sequences
foreach($SequenceData as $sequence){
    //Check Start Date
    $this->_checkStartDate($sequence['sequence_id']);
    //Log Check
    $log = $this->_logCheck($sequence['sequence_id']);
    //Day Check
    $this->_dayCheck($sequence['sequence_id']);
    //Max Limit Check
    $this->_maxLimitCheck($sequence['sequence_id']);
    //Day Clear
    $this->_clearDay($sequence['sequence_id']);
    //Get Articles
    $articles = $this->_getArticles($sequence['sequence_id']);
    //Article Log Check
    $this->_articleLogCheck($sequence['sequence_id']);
    //Get Random Blog
    $blog_data = $this->_getRandomBlog($sequence['sequence_id']);
}
```

© 2010 CODYPCHRISTIAN, NOT FOR USE!

Estructura por tipo de dato

Esta forma de estructuras tu proyecto divide los diferentes archivos por tipo y ayuda cuando tienes proyectos pequeños para organizar y ubicar donde esté lo que necesitas. Ejemplo:

```
Root
  assets
    img
      img1.png
```

```
img2.png
fonts
videos
styles
  file1.css
  file2.css
js
  file1.js
  file2.js
views - file1.html - file2.html index.html
```

Estructura por funcionalidad

Este tipo de organización funciona mucho con proyectos más grandes, los cuales tienen varias páginas como parte de su funcionalidad, en este tipo de proyectos ya se utilizan componentes, rutas, librerías externas, etc. Ejemplo:

```
Root
  assets
    img
      img1.png
      img2.png
    fonts
    videos
  styles
    file1.css
    file2.css
  controllers
    controller1.js
    controller2.js
  components
    component1.js
    component2.js
  routes
    route1.js
    route2.js
views - view1.html - view2.html index.html
```

Este tipo de estructura sirve para cuando se programa en equipos que tienen desarrollos independientes, pero que son todos parte de un mismo proyecto, ya que separa los archivos de forma que cada carpeta tenga sus propios archivos. Ejemplo

```
Root
  assets
    img
      img1.png
      img2.png
    fonts
    videos
  perfil
    perfil.css
    perfil.js
    perfil.html
    perfil.test.js
  feed
    feed.css
    feed.js
    feed.html
    feed.test.js
  searchbar
    searchbar.css
    searchbar.js
    searchbar.html
    searchbar.test.js
```

PRACTICA "ABOGABOT"

Descripción

Se requiere una página web que sea responsive para poder interactuar desde el smartphone en la cual un despacho de abogados podrá automatizar las demandas de sus clientes mediante un formulario. Al ser llenado el formulario por el cliente se manda al proceso de pago y el cliente es capaz de dar seguimiento a su demanda al crear una cuenta en la página. Por otro lado, el administrador recibe las notificaciones de nuevas demandas y con los datos llenados del formulario del cliente se crea de forma automática el documento legal en formato Word para empezar el proceso. El administrador, además de poder ver el pago efectuado por el cliente en un dashboard, actualiza el proceso de la demanda y agrega comentarios. La preferencia de colores es azul marino y blanco.

<https://github.com/ElPoder1>

Toma de requerimientos

Considerando que la aplicación debe de ser responsiva para celular y que la preferencia de colores es azul marino y blanco, se tienen 2 líneas principales de requerimientos a tomar en cuenta:

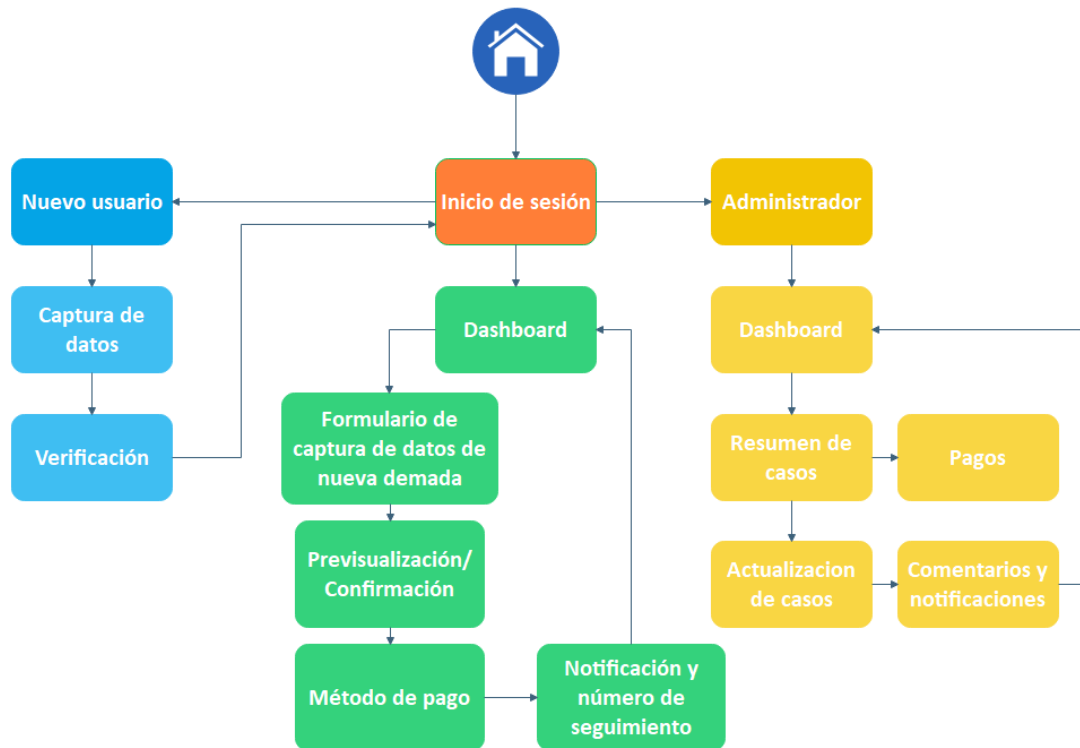
UX de Cliente

- Creación de cuenta del cliente
- Formulario de captura de datos de la demanda
- Método de pago o transacción
- Numero de seguimiento del caso
- Notificación de actualización del caso
- Observación de comentarios del abogado

UX de Administrador (Abogado)


- Creación de cuenta de administrador
- Notificación de nueva demanda
- Dashboard con datos y cumplimiento de pago
- Añadir comentarios y actuaciones del caso al cliente
- Notificaciones al cliente

A continuación, se muestra un análisis del flujo de la aplicación según los requerimientos mencionados con anterioridad. Se hizo uso de la herramienta EdrawMax (<https://www.edrawsoft.com/edraw-max/>).



Buyer persona

Se hace la definición del cliente ideal, el cual es representado por el abogado administrador. Se hizo uso de la herramienta Hubspot tools (<https://www.hubspot.es/make-my-persona>)



Puesto
Abogado

Edad
Entre 25 y 34 años

Nivel de educación más alto
Licenciatura

Redes sociales

f i t

Industria
Seguros

Tamaño de la organización
Entre 1 y 10 empleados

Canal favorito de comunicación

- Teléfono
- Correo electrónico
- En persona

Herramientas que necesita para trabajar

- Sistemas de gestión de contenido
- Programas de procesamiento de texto
- Software de creación de informes
- Software de facturación
- Correo electrónico

Responsabilidades laborales

Seguimiento y solución de problemas judiciales

Su trabajo se mide en función de

Cantidad de casos existentes

Su superior es

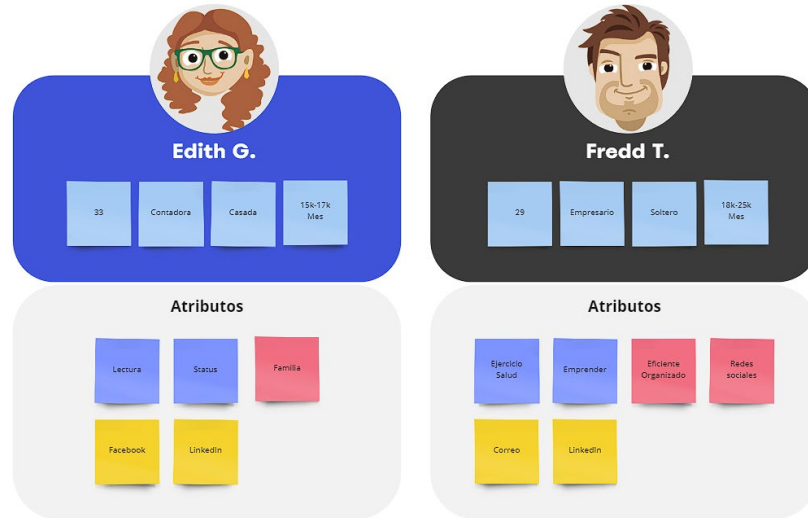
Presidente del despacho

Metas u objetivos

Mejorar su puesto de trabajo

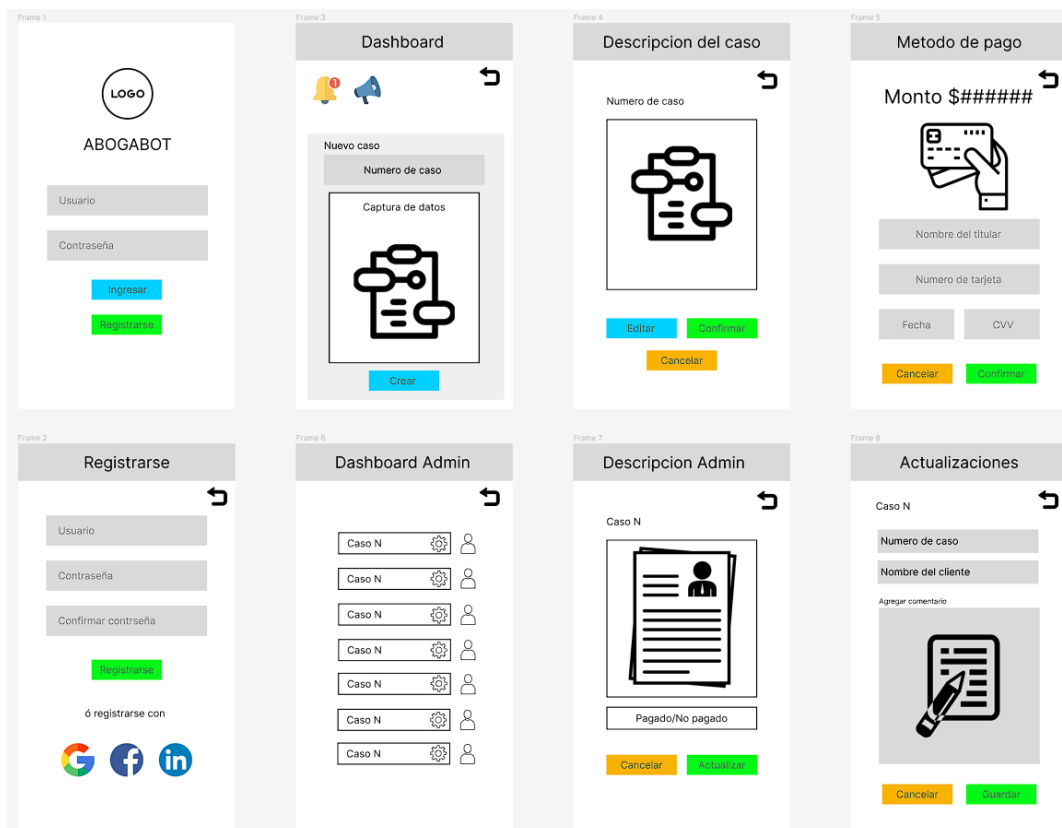
Público objetivo

El público objetivo hace referencia a los posibles compradores del servicio y las necesidades de las cuales partir para solucionar sus problemas. Se trata de los clientes de los abogados. Se utilizó la siguiente plantilla: <https://miro.com/es/plantillas/publico-objetivo/>



Wireframe

En base a la toma de requerimientos y el flujo de trabajo de la aplicación, el primer wireframe UX creado es el siguiente. Se utilizó Figma (<https://www.figma.com/>).



2. HTML

FUNCIONAMIENTO Y ESTRUCTURA DE HTML



HTML (Hyper Text Markup Language) se trata de un lenguaje estándar que se utiliza para estructurar las páginas web ya que permite darle el “esqueleto” a los sitios que se realizan y organiza la información de manera adecuada.

Los headings, los párrafos, los links, las imágenes, las tablas y las listas, son 6 elementos que ayudan a estructurar un archivo HTML para después mostrarlo en una página. Todos los navegadores pueden interpretar la estructura HTML permitiendo leer y saber la posición de elementos e incluso el posicionamiento dentro de los buscadores, lo cual está estrictamente relacionado con la optimización de los motores de búsqueda (SEO) para que la información del sitio esté bien posicionada dentro de internet.

Estructura HTML

De forma general podemos ver que todas las palabras tienen dos corchetes <>, uno al inicio y otro al final, uno que abre y otro que cierra con una diagonal antes de la palabra, lo cual es parte de la sintaxis, a esto se le conoce como “etiquetas”, las cuáles se encargan de delimitar cada cuando se está poniendo un inicio y un final de cierto elemento en HTML, la posición de cada elemento en HTML se rige por estas etiquetas.

Por norma general, al igual que todos los lenguajes, se utiliza inglés para programar, sin embargo, es posible utilizar español u otros idiomas, pero se debe de tener cuidado ya que esto afecta directamente en el SEO.

```
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

Indica que el contenido tiene que ser equivalente de Internet Explorer.

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

Se refiere a la vista general del sitio. Como va a estar desplegado por defecto tiene un ancho de la muestra del sitio igual que el ancho de nuestro dispositivo, con un “initial-scale=1.0” que sirve para hacer zoom in o zoom out, dependiendo de cómo es que se quiere la página.

```
<title>Titulo</title>
```

Es el título del sitio en el head.

```
<style></style>
```

Son las etiquetas de estilos. Permite importar archivos de otro tipo como Javascript, CSS, etc. Al ingresar a una página el navegador descarga la página de internet y se lo muestra al cliente. Se descargan los estilos, funcionalidades, etc. Este orden es importante para que la página se visualice de forma ordenada.

ETIQUETAS BÁSICAS

Headings

Estos títulos permiten resaltar información en tamaño y visibilidad, además de cuestiones de SEO. Generar etiqueta heading

```
<h1>Bienvenidos Explores</h1>
<h2>Bienvenidos Explores</h2>
<h3>Bienvenidos Explores</h3>
```

El valor que corresponde a “h” puede ir del 1 al 6, permite ordenar estos títulos del mas importante al menos importante, por ejemplo, h2 puede ser títulos o subtítulos.

El valor de estas “h” están estrictamente relacionados con la jerarquía de las búsquedas en internet, si dos páginas tuvieran un mismo heading pero uno tuviera h1 y otro h6, el navegador le daría prioridad al h1, por lo que aparecería entre las primeras búsquedas en el navegador debido a que lo considera como información importante y valiosa.

Nota: No te olvides que para desplegar tu proyecto y poder visualizarlo debes crear una carpeta con tu archivo html y abrirlo en tu navegador.

Párrafos

Permiten ingresar información, y su etiqueta se muestra de la siguiente manera:

```
<p>contenido</p>
```

Donde se tiene el contenido se puede poner la información que se quiere que se visualice en el navegador, pero aun sin estilos ya que ese es un tema que se ve más adelante.

Se puede agregar un párrafo adicional con salto de línea por defecto, pero si se quieren poner más párrafos separados dentro del mismo, sin el salto de línea, se puede ajustar con la siguiente etiqueta, y se conoce como break line.

```
<br />
```

Enlaces (links)

Permiten colocar un link como si fuera un hipervínculo, por medio de una palabra o varias se puede dirigir al link que le indiques al usuario, utilizando la siguiente etiqueta.

```
<a href="https://github.com/Launch-X-Latam" target="blank">Link a LaunchX</a>
<br />
```

La letra “a” hace referencia al atributo que indica hacia dónde va a dirigir el vínculo. Dentro de la misma etiqueta se puede agregar otro atributo, por ejemplo, un “target= blank”, lo que nos permite abrir en una pestaña nueva la página del link que indicamos.

Es muy útil cuando se quiere llevar al usuario a ciertas paginas estratégicas, por ejemplo, a nuestras redes sociales, la compra de algún producto, etc.

Imágenes

```

```

Contiene dos atributos, **source** o bien "src" que se refiere a la fuente de donde vamos a usar la imagen, la cuál puede ser de forma local o una imagen de internet y "alt" que significa texto alternativo, este texto alternativo le va a dar la descripción a la imagen. También podemos usar otros atributos como "width" y "height" donde vamos a poner una cantidad en pixeles que son el alto y el ancho de la imagen.

Tablas

Este elemento permite organizar la información, imágenes, links, etc.

```
<table>
  <tr>
    <th>Compañía</th>
    <th>Contacto</th>
    <th>País</th>
  </tr>
  <tr>
    <td>Alfreds Futterkiste</td>
    <td>Maria Anders</td>
    <td>Germany</td>
  </tr>
  <tr>
    <td>Centro comercial Moctezuma</td>
    <td>Francisco Chang</td>
    <td>Mexico</td>
  </tr>
</table>
```

Lista

Para agregar un elemento a la lista usamos la siguiente etiqueta:

```
<li>café</li>

<ul>
  <li>Café</li>
  <li>Arroz</li>
  <li>leche</li>
  <li>huevos</li>
</ul>
```

ul significa "unordered list" es decir una lista que no tiene orden.

```
<ol>
  <li>Alumno1</li>
  <li>Alumno2</li>
  <li>Alumno3</li>
</ol>
```

ol significa "ordered list" y te mostrara una lista en orden jerárquico.

```
<dl>
  <dt>Enchiladas</dt>
  <dd>- Tortilla frita con salsa, rellena de pollo</dd>
  <dt>Molletes</dt>
  <dd>- Bolillo con frijoles y queso</dd>
</dl>
```

Practica etiquetas básicas

<https://drive.google.com/file/d/1NYxRnajLf1YTMEKdm7TGafie-8sOMIP/view?usp=sharing>

ELEMENTOS COMPUESTOS. INPUTS

Los elementos compuestos hacen referencia a las entradas que recaban información del usuario y que además permite enviarla por medio de botones, selectores, etc.

Entrada tipo texto

```
<input type="text" id="Input1" name="Input1">
```

El atributo "id" y nombre permiten identificarlos, y es de suma importancia para saber a qué campo nos estamos refiriendo, para cuando tengamos más de un input, y tengamos que utilizar su información en el backend o para procesarlos después. Todos los elementos pueden llevar estos atributos.

Etiquetas (Labels)

```
<label for="Input1">Ejemplo de input de texto</label>
```

Permiten etiquetar un elemento. Usualmente se colocan antes del elemento que se desea etiquetar, pero se puede colocar en cualquier parte del mismo.

Las siguientes líneas de código son un ejemplo de input de texto con etiquetas:

```
<label for="Input1">Ejemplo de input de texto</label>
<input type="text" id="Input1" name="Input1">
```

Input de correo

```
<h2>Input De Texto</h2>
<label for="correo1">Input de Correo</label><br>
<input type="email" id="correo1" name="correo1"><br><br>
```


El input de correo valida que la información dentro del input tenga un formato de correo, es decir que contenga elementos vitales como una arroba, un punto, etc. De lo contrario le indicara al usuario que es un correo invalido, esta funcionalidad se usa con JavaScript y estilos.

Input de numero

```
<label for="numero1">Input de número</label><br>
<input type="number" id="numero1" name="numero1"><br><br>
```

Valida que solo se puedan ingresar datos de tipo número y no letras o símbolos.

Input de password

```
<label for="psw1">Input de Password</label><br>
<input type="password" id="psw1" name="psw1"><br><br>
```

Oculto la información de los caracteres que se están ingresando.

Input de teléfono

```
<label for="tel1">Input de Teléfono</label><br>
<input type="tel" id="tel1" name="tel1" maxlength="10"><br><br>
```

Valida que únicamente puedas ingresar dígitos números. Por medio del maxlength="10" se establece un límite de 10 números.

Input de URL

```
<label for="url1">Input de URL</label><br>
<input type="url" id="url1" name="url1"><br><br>
```

Verifica que la URL tenga ciertas características de una URL normal

Practica Elementos Compuestos

https://drive.google.com/file/d/1mA7r7KskaQE81_D95p795GtCqBF8-Mjt/view?usp=sharing

ACOMODO Y NAVEGACIÓN. LAYOUT Y RUTAS

Practica Layout y rutas

<https://drive.google.com/file/d/1BV3xpQsdXv6w6yyB5Yf2heACyaeU1tGW/view?usp=sharing>

PRACTICA "TAQUERIA"

Descripción

Se requiere una página web para un negocio de comida el cual se centra principalmente en la venta de tacos.

Metodología

Considerando que el nombre propuesto para el negocio es “Taco Time” se tomaron ideas para el diseño del logo utilizando la plataforma de Canva (<https://www.canva.com/>). El objetivo de esta práctica es aplicar los conocimientos adquiridos de HTML, no desarrollar toda la parte del wireframe.

El código se dividió en 3 secciones principales para el menú, información, y pedidos.

En el menú se utilizaron listas desordenadas y descriptivas para enunciar el tipo de comida con la que se dispone. Para el apartado de información se utilizó una tabla en la que se anexaron enlaces para poder ir a la ubicación de las diferentes sucursales. En el apartado de pedidos se utilizaron elementos compuestos para las diferentes entradas como select, checkbox, text, y button. El código y pagina creada son los siguientes:

<https://drive.google.com/file/d/1b7UZZy99hhOBRgG3uldmaxnQQGzSgAp/view?usp=sharing>

Taco Time



TACO TIME

Menú

Tacos tradicionales

Todos incluyen cebollas y limones

- Bistec\$20
- Pastor\$16
- Cabeza\$15
- Chorizo\$16

Tacos especiales

Todos incluyen cebollas, limones y refresco en la compra de 5 tacos

- Bistec Azteca\$30
 - Bistec cocinado al mas puro estilo Mexicano
- Bistec Español\$35
 - Para los que se creen superiores
- Arrachera Atlantica\$40
 - Sabor digno de Acuanan
- Pastor Noruego\$25
 - Diseñado para los mas finos paladares
- Cabeza de uniconio\$70
 - Cuenta la leyenda que cura cualquier enfermedad

Información

Sucursales

Nombre	Ubicacion	Horarios
Norte	Anáhuac 769-c, Fundadores de Moreleón	L - D, 24h
Centro	Zona Centro, 38800 Moreleón	L - D, 5pm - 11pm
Sur	Ponciano Vega 190, Insurgentes	L - S, 10am - 4pm

Pedidos

Haz tu pedido aqui :)

¿De qué sucursal quieres los tacos?

Norte ▾

¿De qué tacos quieres?

- ☐ Bistec - ☐ Pastor - ☐ Cabeza - ☐ Chorizo - ☐ Azteca - ☐ Español - ☐ Atlantica - ☐ Noruego - ☐ Unirconio

Nombre:

Telefono:

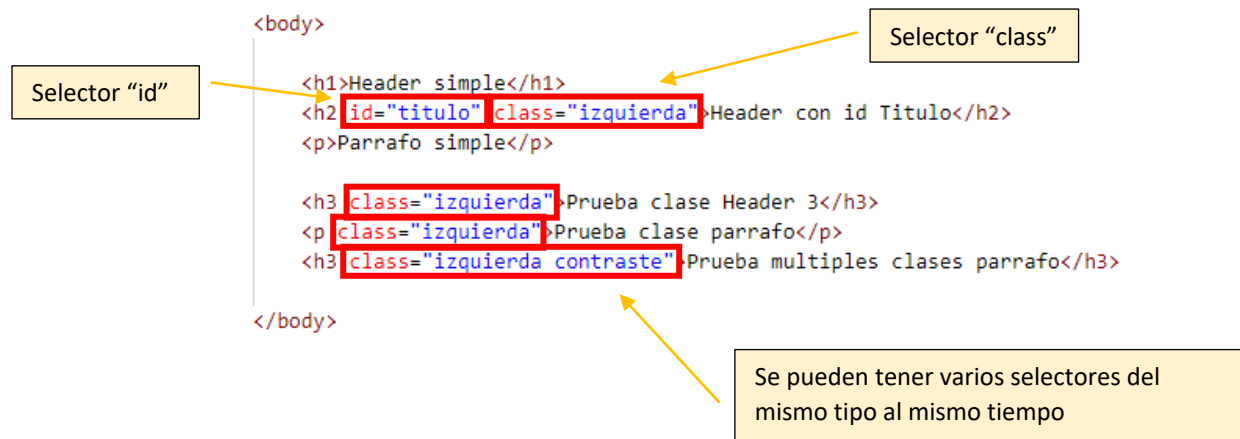
Direccion:

3. CSS

Cascade Style Sheet

SELECTORES Y FONDOS

Para poder añadir estilo a las diferentes secciones del código HTML se utilizan selectores los cuales funcionan por jerarquía a través de identificadores que deben de colocarse en la estructura HTML.



En CSS con los selectores incorporados en HTML se puede abrir una nueva sección llamada “Style” dentro del “head” inicial en el mismo archivo HTML, o se pueden mandar llamar archivos .CSS lo cual es más profesional. Los archivos CSS se enlazan con el HTML utilizando la siguiente etiqueta:

```
<link rel="stylesheet" href="./2.-Colores.css">
```

En href se indica el archivo CSS que se va a utilizar. Esta etiqueta “link” se agrega en el “head” inicial del HTML.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Selectores</title>
  <style>
    *{
      font-family: Arial, Helvetica, sans-serif; /*Indica que se va a aplicar el estilo a todo el html*/
    } /*Tipo de letra a utilizar*/
  </style>

```

```

body{                                     /*Se aplica a toda la etiqueta Body de HTML*/
    background-color: aqua;              /*Color del fondo*/
}

p{                                       /*Se aplica a todo lo que tenga la etiqueta p*/
    color: red;
    text-align: center;
}

#titulo{                                /*Se aplica al selector id*/
    color: rgb(200,150,220);
    text-align: right;
}

.izquierda{                             /*Se aplica al selector clase*/
    color: green;
}

p.izquierda{                            /*Se aplica al parrafo p de la clase*/
    text-decoration: underline;
    text-align: left;
}

.contraste{                             /*Se aplica a la clase contraste de la clase izquierda*/
    background-color: black;
}

h1, h2, p{                              /*Pueden aplicarse estilos a varias etiquetas a la vez */
    font-weight: bold;
}

/* Comentario CSS */
</style>
</head>

```

Colores

```

p{
    font-size: 20px;
}

.color{
    color: red;
    color: rgb(93, 113, 218);
    color: rgba(93, 217, 94, 0.6);
}

```

```
    color: #EE43E3;  
    color: hsl(30, 68%, 31%);  
}
```

Fondos

```
*{  
    font-family: Arial, Helvetica, sans-serif;  
}  
  
section{  
    height: 400px;  
}  
  
div{  
    height: 400px;  
}  
  
.fondoColor{  
    background-color: #0f6666;  
}  
  
.fondoOpacidad{  
    background-color: #5d0c0c;  
    opacity: 0.6;  
}  
  
.fondoImagen{  
    background-image: url("../images/fondo1.jpg");  
}  
  
.fondoRepetidoX{  
    background-image: url("../images/fondo2.jpg");  
    background-repeat: repeat-x;  
}  
  
.fondoRepetidoY{  
    background-image: url("../images/fondo1.jpg");  
    background-repeat: repeat-y;  
}  
  
.fondoNoRepetido{  
    background-image: url("../images/fondo2.jpg");  
    background-repeat: no-repeat;  
}
```

```
.fondoPosicion{
    background-image: url("../images/fondo1.jpg");
    background-repeat: no-repeat;
    background-position: bottom right;
}

.fondoCompleto{
    background-image: url("../images/fondo2.jpg");
    background-repeat: no-repeat;
    background-position: center;
    background-size: cover;
}
```

Textos

```
*{
    font-family: Arial, Helvetica, sans-serif;
}

.izquierda{
    text-align: left;
}

.derecha{
    text-align: right;
}

.centrado{
    text-align: center;
}

.justificado{
    text-align: justify;
}

.alReves{
    direction: rtl;
    unicode-bidi: bidi-override;
}

.decoradoOverline{
    text-decoration-line: overline;
}

.decoradoLineThrough{
    text-decoration-line: line-through;
}
```

```

.decoradoUnderline{
    text-decoration-line: underline;
}

.decoradoCombinado{
    text-decoration-line: underline;
    text-decoration-color: red;
    text-decoration-style: dotted;
    text-decoration-thickness: 5px;
}

.negritas{
    font-weight: 700;
}

.ligeras{
    font-weight: 100;
}

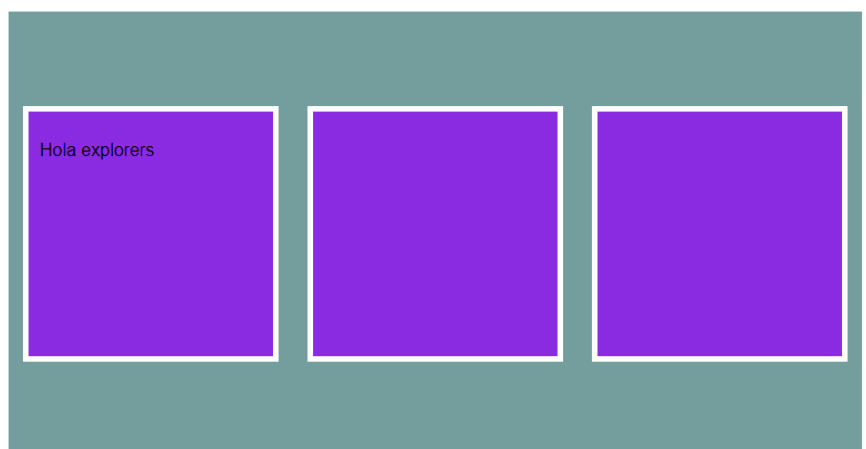
.link{
    text-decoration: none;
    color: black;
}

```

FLEXBOX

Los flexbox son una herramienta que permiten estructurar contenido en una página mediante cajas las cuales brindan mucha más flexibilidad a la hora de organizar toda la información del sitio. En el siguiente enlace se encuentra una guía completa para usar el flexbox.

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

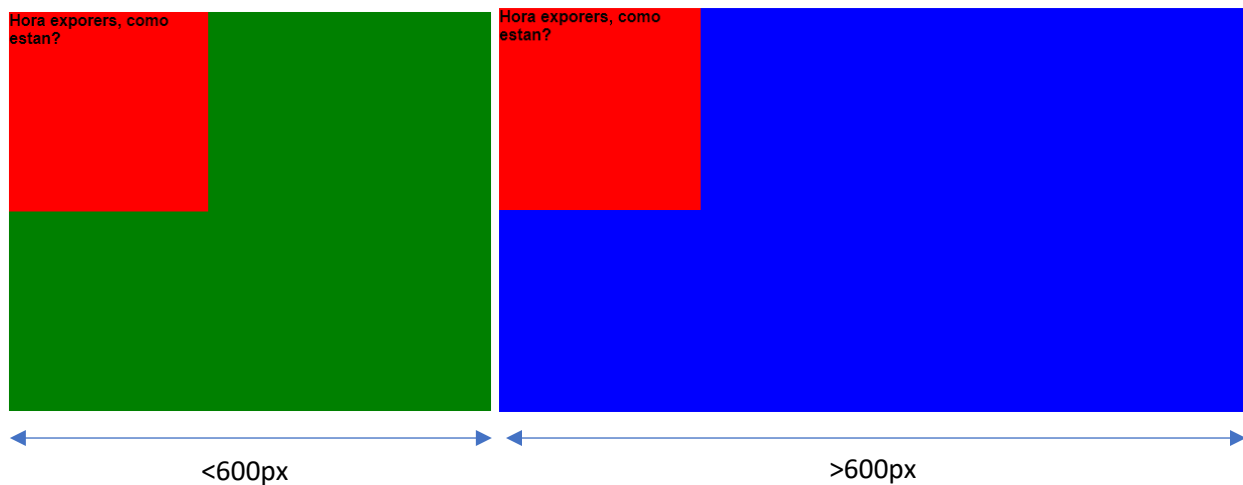


RESPONSIVE CON CSS

En CSS es posible utilizar Media Query para tener diferentes reglas en una clase dependiendo del tamaño de la pantalla, es decir, se puede tener una determinada posición de elementos, colores, tamaños de texto, fondos etc en función del tamaño de la pantalla. Es similar a utilizar condicionales que se ejecutan en paralelo con el código para determinar qué reglas se aplicaran a una clase según la pantalla. Ejemplo:

```
.container{
  background-color: blue;
  height: 400px;
}

@media(max-width:600px){ /*mientras la pantalla no exceda los 600px*/
  .container{
    background-color: green;
  }
}
```



<https://www.freecodecamp.org/news/css-media-queries-breakpoints-media-types-standard-resolutions-and-more/>

PRACTICA CSS

Descripción

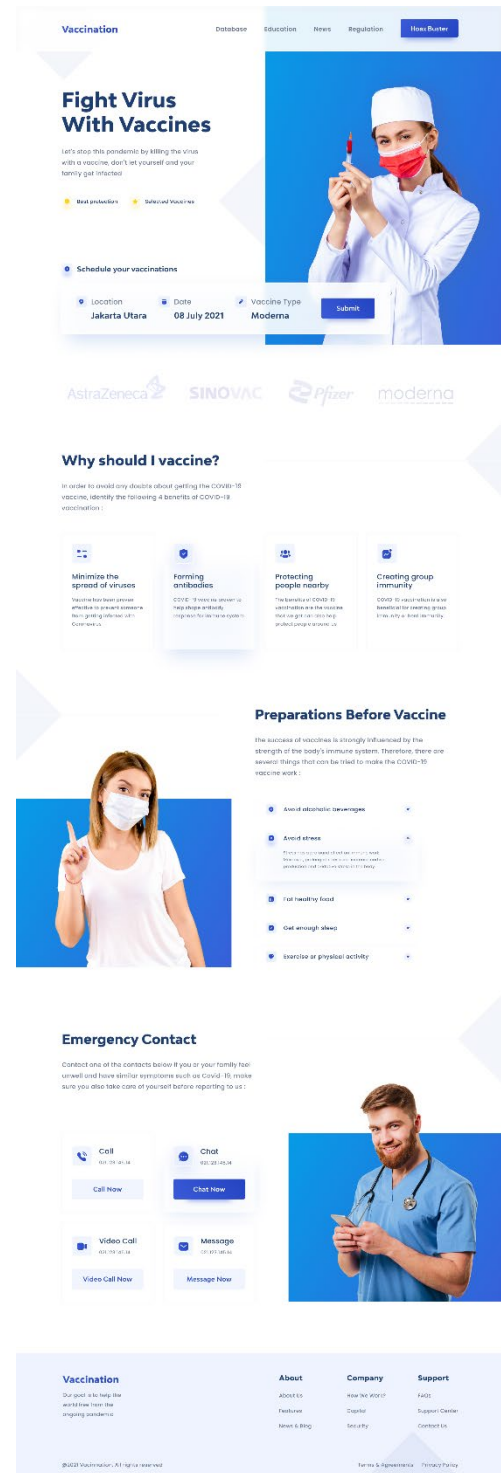
Para esta práctica se debe de clonar la siguiente página

La práctica consiste en lo siguiente:

- Planeación de campaña de vacunación (Un poco de mercadotecnia para llegar al sitio)
- Maquetación del sitio con HTML
- Estilos con CSS (Lo más acercado posible, pueden ser otras imágenes, íconos o colores, pero tiene que ser lo más cercano que puedas)

Bonus:

- Bonus de diseños o páginas adicionales (Totalmente a tu creatividad)
- Bonus de despliegue de la página



4. JAVASCRIPT

CONCEPTOS BÁSICOS

Variables

/*Las variables se pueden declarar con la palabra reservada "var", estas variables se pueden considerar como globales.

También se pueden declarar con la palabra reservada "let" y estas serán usadas dentro de un bloque de código.

Igualmente está la declaración con la palabra reservada "const" y se usarán cuando el valor no cambie */

```
console.log("\n***** Variables *****\n");
var numero1 = 4;
var numero2 = 6;
console.log("Número 1: " + numero1 + " Numero 2: " + numero2);
```

Strings

/*Las cadenas (Strings) son caracteres que pueden ser una frase o palabra y estas pueden darse con comillas dobles "", simples ' ' o invertidas ``

la diferencia es que con las invertidas podemos agregar variables dentro de la cadena con la sintaxis \${}*/

```
console.log("\n***** Cadenas *****\n");
var frase1 = "Ejemplo comillas dobles";
var frase2 = 'Ejemplo comillas simples';
var frase3 = `Ejemplo comillas ${frase1} invertidas`;

console.log(frase1 + "\n" + frase2 + "\n" + frase3);
```

Condicionales

/* Las condicionales se pueden usar valores como > < == === != y cada una tiene una funcionalidad de comparación entre elementos */

```
console.log("\n***** Condicionales *****\n");

console.log(numero1 != numero2);
```

Operadores lógicos

/* Los operadores lógicos se utilizan cuando se necesita comparar más de una condicional
El operador && (AND) necesita que todos sus valores sean true para que la salida sea true
El operador || (OR) necesita que solo uno de sus valores sea true para que la salida sea true*/

```
console.log("\n***** Operador lógico AND *****\n");
console.log(true && true);

console.log("\n***** Operador lógico OR *****\n");
console.log(false || false);
```

Arreglos

/* Los arreglos son estructuras de datos que nos permiten agrupar datos de un mismo tipo */

```
console.log("\n***** Arreglos *****\n");
let listaDeNumeros = [2, 3, 5, 7, 11, 234];

console.log(listaDeNumeros[5]);

listaDeNumeros.push(16);
listaDeNumeros.push(939);

console.log(listaDeNumeros);
console.log(listaDeNumeros.length);

let listaDePalabras = ["Explorer", "MisionComander", "LaunchX",
" Innovacion"];
console.log(listaDePalabras);
console.log(listaDePalabras.length);

/* Las cadenas (strings) pueden ser tratadas como arreglos */

let palabra = "Explorer";
console.log(palabra[2]);
console.log(palabra.length);
```

Objetos

/* Los objetos son estructuras de datos que nos permiten agrupar datos de diferentes tipos */

```
console.log("\n***** Objetos *****\n");
```

```
let explorer = {
  nombre: "Nombre del Explorer",
  email: "email@innovaccion.mx",
  numReg: 12345,
  mision: "FrontEnd",
  proyectos: ["Abogabot", "Taquería", "Pastelería", "Vacunación"],
  proPer: {
    escolar: "Tareas",
    profesional: "Trabajo",
    personal: "Negocio"
  }
};

console.log(explorer);

console.log(explorer.proPer.escolar);
```

Flujo de condicional

```
/* Flujo condicional */
let number1 = 2;
let number2 = 6;
console.log("\n***** if / else *****\n");
if (number1 > number2 && number2 > 5) {
  console.log("El número 1 es mayor que número 2");
}
else if( number1 == number2 || number1 < 3){
  console.log("Los números son iguales");
}
else {
  console.log("El número 2 es mayor que número 1");
}
```

Ciclos

```
/* Ciclo condicional */
console.log("\n***** While *****\n");
let numberWhile = 5;
while (numberWhile <= 12) {
  console.log(numberWhile);
  numberWhile = numberWhile + 2;
}
console.log("Aquí ya salió del while " + numberWhile);
```

```
/* Ciclo condicional de una iteración mínimo */
console.log("\n***** Do While *****\n");
let numeroDoWhile = 22;
do {
    numeroDoWhile = numeroDoWhile + 2;
    console.log(numeroDoWhile);
} while (numeroDoWhile < 20);
console.log("Aquí sale del Do While " + numeroDoWhile);

/* Ciclo for con iteración controlada */
console.log("\n***** For *****\n");
let numeroFor = 0
for (numeroFor ; numeroFor <= 12; numeroFor = numeroFor + 1) {
    console.log(numeroFor);
}
console.log("Aquí salimos del for " + numeroFor);
```

Switch case

```
/* Opciones para evitar anidar condicionales */
console.log("\n***** Switch *****\n");
switch (prompt("¿Cómo está el clima?")) {
    case "lluvioso":
        console.log("No te vayas a mojar");
        break;
    case "soleado":
        console.log("Ponte bloqueador");
        break;
    case "nublado":
        console.log("Tapate bien");
        break;
    default:
        console.log("No se cómo está el clima");
        break;
}
console.log("Aquí salimos del Switch");
```

FUNCIONES

Función con retorno

```
/*Ejemplo de función básica que retorna parámetro*/
const cuadrado = function(x) {
    return x * x;
};

let numero1 = 8;
let numero2 = 2;
let numeroCuadrado1 = cuadrado(numero1);
let numeroCuadrado2 = cuadrado(numero2)
console.log(numeroCuadrado1);
console.log(numeroCuadrado2);
```

Función sin retorno

```
/*Funcion que no retorna ni recibe parámetro*/
const ruido = function () {
    console.log("kataplum!");
}
/*Se imprime el mensaje de la función. FUNCION VOID*/
ruido();
```

Múltiples entradas

```
/*Funcion de 2 entradas y un retorno*/
const exponencial = function (base, exponente) {
    let resultado = 1;
    for (let i = 0; i < exponente; i++){
        resultado *= base; /*resultado = resultado * base*/
    }
    return resultado;
}
console.log(exponencial(4,3))

/*El navegador interpreta el código en paralelo por lo que se pueden usar las
funciones antes de declararlas*/
console.log(sumar(5,65));
function sumar(x, y) {
    return x + y;
}
```

Funciones flecha

```
/*No se tiene la palabra reservada "function" pero el simbolo "=>" indica que
se trata de una funcion*/
const restar = (a, b) => {
    return a - b;
}
console.log(restar(40, 8));

/*Las funciones no solo se limitan a numeros*/
function saludar(quien) {
    console.log("Hola " + quien);
}
saludar("Explorer");
```

Excepciones

```
/*Las excepciones son para evitar que el programa se rompa cuando se reciben
datos de entrada no validos*/
function preguntaDireccion(pregunta) {
    let result = prompt(pregunta);
    if (result.toLowerCase() == "izquierda") return "I"; /*toLowerCase convierte
todos los caracteres a minuscula*/
    if (result.toLowerCase() == "derecha") return "D";
    throw new Error("Dirección inválida: " + result); /*Permite capturar errores
para mostrarlos correctamente al usuario*/
}

function mirar() {
    if (preguntaDireccion("A que lado?") == "I") {
        return "una casa";
    } else {
        return "2 osos hambrientos";
    }
}

/*Permite capturar errores para mostrarlos correctamente al usuario*/
try {
    console.log("Mira a ", mirar());
} catch (error) {
    console.log("Hubo un error: " + error);
}
```