# Diseño de Páginas Web 2

## 1. Estructura Básica

### a. Introducción: ¿Qué es una Aplicación Web?

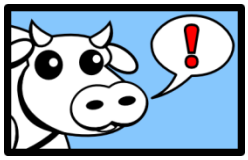
Bienvenidos al curso de Diseño de Páginas Web 2.

En el curso de Diseño de Páginas Web 1 del año pasado trabajamos los conceptos básicos de lo que implica diseñar una página web (manejo de lenguaje HTML), cómo expandir ese diseño a otras páginas para así diseñar un sitio web (manejo de lenguaje CSS), e incluso algunos aspectos básicos de interactividad con el usuario (introducción al manejo del lenguaje JavaScript).

En este curso todos esos conceptos se estudiarán más en profundidad con el fin de proporcionar las herramientas necesarias para el desarrollo de aplicaciones web.

En el apartado técnico, todo el código mencionado en este documento puede consultarse en el siguiente repositorio: https://github.com/kurotori/ProgYDisWeb/tree/main/DisWeb2. Asimismo, todo el código aquí mencionado se desarrolló utilizando la IDE Visual Studio Code, obtenible de forma gratuita en su página oficial: https://code.visualstudio.com/Download . Sin embargo, **esto no imposibilita** el uso de **cualquier otra IDE** que el lector considere conveniente, ya que no se harán menciones de uso específico.

Para ello, comencemos explicando **qué es una aplicación web**:



**Aplicación Web:**
Se trata de una aplicación que se ejecuta en un ***servidor web***, mientras que la interfaz de usuario y los resultados de dicha ejecución son presentados a un ***usuario de forma remota*** mediante una ***página web.***

En este curso veremos que, hoy en día, **no toda** la ejecución de la aplicación debe suceder en el servidor web (a fin de aligerar el consumo de recursos del mismo) sino que una parte de la ejecución debe suceder en la computadora local, dentro del entorno de ejecución provisto por el navegador web, mediante scripts en lenguaje JavaScript.

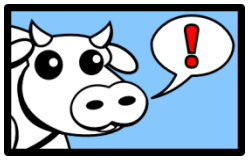
Un diseño web inteligente permitirá que nuestra aplicación tenga, entonces, un **consumo de recursos bien balanceado**, permitiendo una ejecución fluida en los dos extremos de la misma.

### b. Comenzando el diseño: El proceso de Maquetado

#### ¿Qué es Maquetar?

El maquetado es el comienzo de toda aplicación bien diseñada. Esto se debe a que un buen diseño, para serlo, debe pasar por varias etapas de revisión y corrección, y es mucho más eficiente si una buena parte de ese proceso puede **realizarse sin que esté involucrado el código.** Es mucho más rápido (y barato) plantear la apariencia de nuestra aplicación en un papel, es mucho más fácil de modificar, en pocos minutos es posible proponer múltiples variaciones, revisar diferentes enfoques, disposiciones del contenido y combinaciones de colores, y todo ello sin tener que generar una sola línea de código.

Una vez establecidas de esta manera las características que debe tener la aplicación, el trabajo de transformar el planteo al código se puede realizar de forma mucho más rápida y organizada, se pueden planificar mejor las etapas del desarrollo, y en general se ahorra tiempo, dinero y esfuerzo humano.



**Maquetado:**
Etapa del proceso del diseño de un producto o servicio, en la cual el diseñador o diseñadores generan ***maquetas***, modelos, por lo general no funcionales, que demuestran la apariencia general que tendrá el producto final.

Como ya se dijo más arriba, para generar maquetas **no es necesario hacerlo en una computadora**, basta con tener a mano **papel, lápices, y algo con que colorear.** Eso no significa que no se pueda realizar en una computadora, todo lo contrario, existen aplicaciones de maquetado muy potentes con diferentes enfoques. Para generar las imágenes y maquetas de esta documentación, se utilizaron algunas o todas estas aplicaciones de código abierto:

- Dia (http://dia-installer.de/)
- Pencil Project (https://pencil.evolus.vn/)
- Inkscape (https://inkscape.org)
- GIMP (https://www.gimp.org/)

## Aspectos a tener en cuenta al maquetar para una Aplicación Web

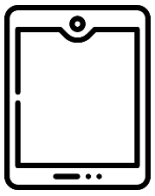
Al maquetar para una aplicación web es necesario que establezcamos algunos criterios generales de diseño y nomenclatura que debemos tener en cuenta.

El primer aspecto a tener en cuenta es que nuestra aplicación va a visualizarse en un navegador web, y eso implica que su facilidad de uso y apariencia deben ser **consistentes** en diferentes **modos de visualización** y en **diferentes plataformas**.

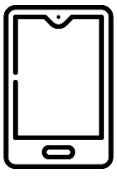
Esto implica que al diseñar nuestra aplicación debemos **considerar el medio** mediante el cuál nuestro usuario va a acceder a nuestra aplicación, y que, muy probablemente, ***sean todos estos***:

*Monitor de PC de escritorio*

*Monitor de Notebook o Laptop*



*Tablet*



*Smartphone*

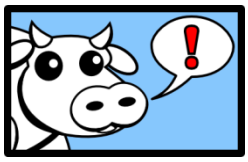
Como resulta obvio al considerarlo, esto implica que tenemos que tener en cuenta diversas resoluciones de pantalla al momento de diseñar, y que, además, nuestra aplicación debe ser capaz de **adaptarse de forma fluida** a las mismas. Esta cualidad es el núcleo de una filosofía de diseño llamada ***Responsive Web Design*** *("Diseño Web Adaptable")* o simplemente ***RWD***, que desde hace varios años ya forma parte de los conceptos

[1] Íconos hechos por Freepik, del sitio www.flaticon.com

clave del diseño web profesional, y cuyas particularidades técnicas exploraremos en profundidad más adelante en este curso.

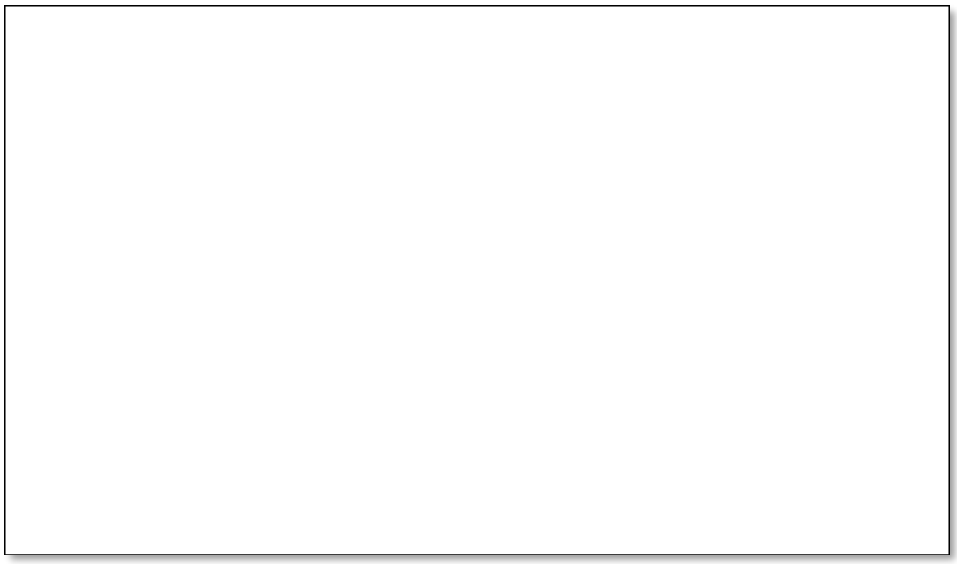
Pero para comenzar, plantearemos qué debemos considerar para que nuestro diseño tenga, desde el comienzo, aspectos compatibles con el RWD.

En primer lugar, debemos considerar los elementos que estarán presentes en el diagrama general, o ***layout***, de nuestra aplicación.

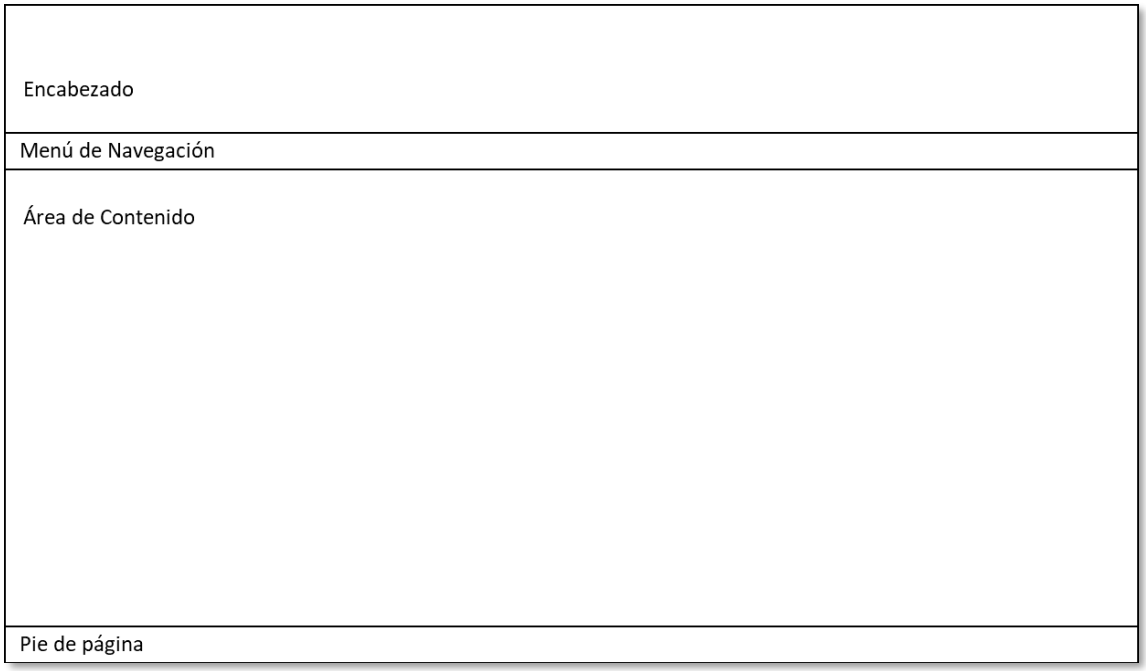


**Layout o Diagrama General:**
Es la distribución general del contenido en la página o aplicación web. Se suele definir utilizando bloques rectangulares para establecer el espacio aproximado que ocupará un determinado contenido en el área de visualización.

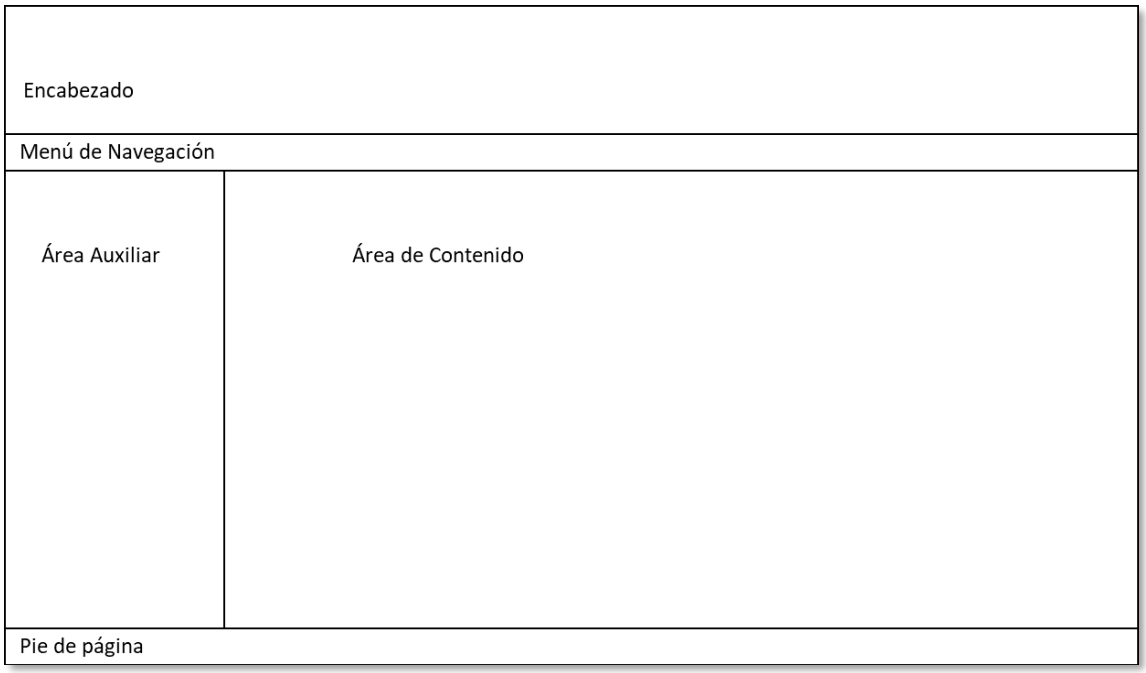
Todos nuestros elementos estarán agrupados dentro del área de visualización del navegador, o ***viewport***, esta área es la que va a variar de forma importante entre dispositivos, y la que tendremos en cuenta para diseñar nuestra aplicación. Para representarla, simplemente utilizaremos un rectángulo en blanco:



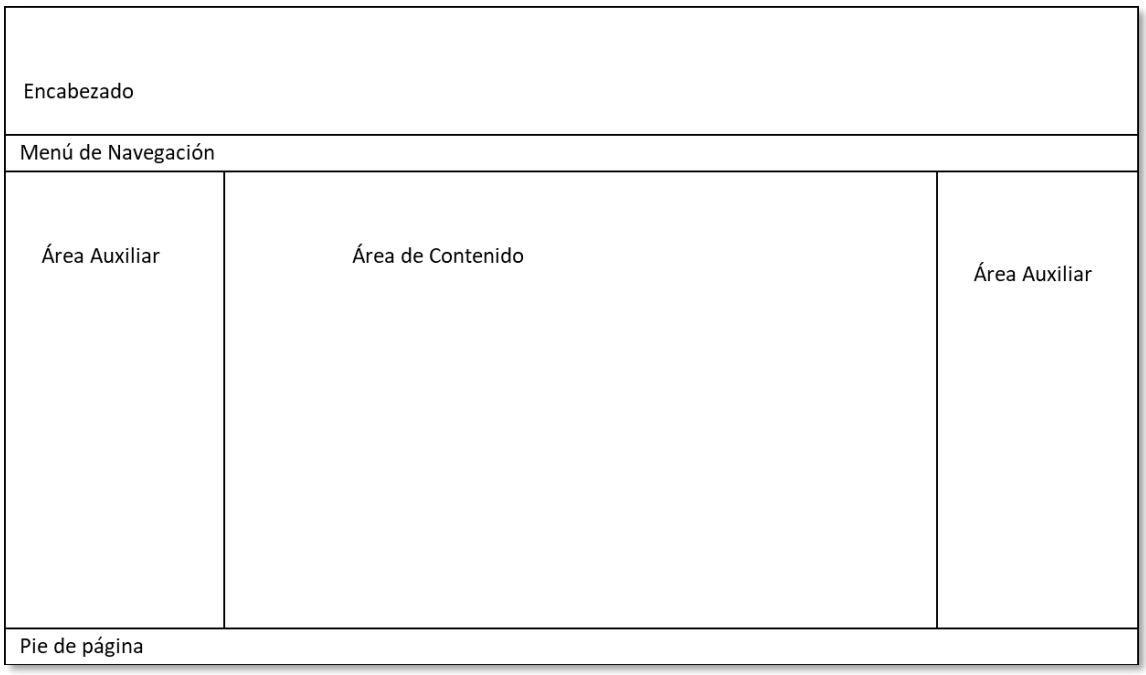
Si bien podemos ser **muy creativos** en el uso del espacio disponible en el viewport, esto no siempre significa que obtengamos un resultado que resulte accesible para nuestros usuarios. En general se utiliza y adapta, con este fin, alguno de los siguientes layouts:




*Layout de una columna*




*Layout de dos columnas*

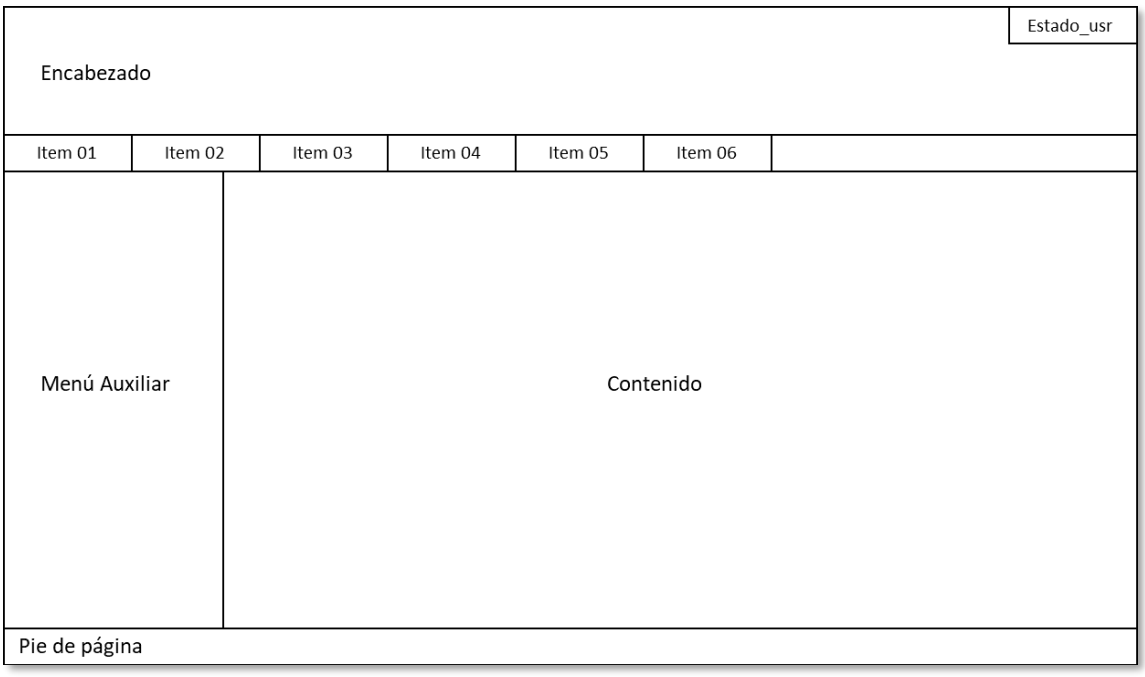



*Layout de tres columnas*

Estos layouts básicos permiten organizar el contenido en áreas de fácil implementación y además tienen la ventaja de ser sumamente compatibles con los planteos del RWD.

Una vez seleccionada la distribución de elementos más conveniente para las necesidades de nuestra aplicación, podemos ir agregando elementos y detalles en las zonas de nuestro layout hasta completar la maqueta. Conviene aclarar en este momento que una aplicación no tiene porque basarse exclusivamente en un solo layout. Para diferentes casos de uso de nuestra aplicación podemos aprovechar diferentes layouts que sean más apropiados a los requerimientos de cada caso, y generar, para cada uno, las maquetas correspondientes.

Por ejemplo, tomemos un layout de dos columnas y agreguemos en el algo de contenido:



Este ejemplo será el que usaremos el resto del curso para demostrar los procedimientos que acá desarrollemos. Conforme vayamos avanzando, le agregaremos más detalle e iremos planteando maquetas más complejas. En este punto del desarrollo de la aplicación consideraremos que los elementos agregados cumplirán las siguientes funciones:

- Encabezado: Contendrá el título de la aplicación y quizás alguna imagen.
- Estado_usr: Será un menú desplegable con información del usuario.
- Items del Menú de navegación: Serán elementos tipo botón individuales o menús desplegables con sub-elementos.
- Contenido: Mostrará los datos de la aplicación con los que esté trabajando el usuario en ese momento.
- Menú Auxiliar: Mostrará acciones extra disponibles para la funcionalidad que este utilizando el usuario en ese momento.

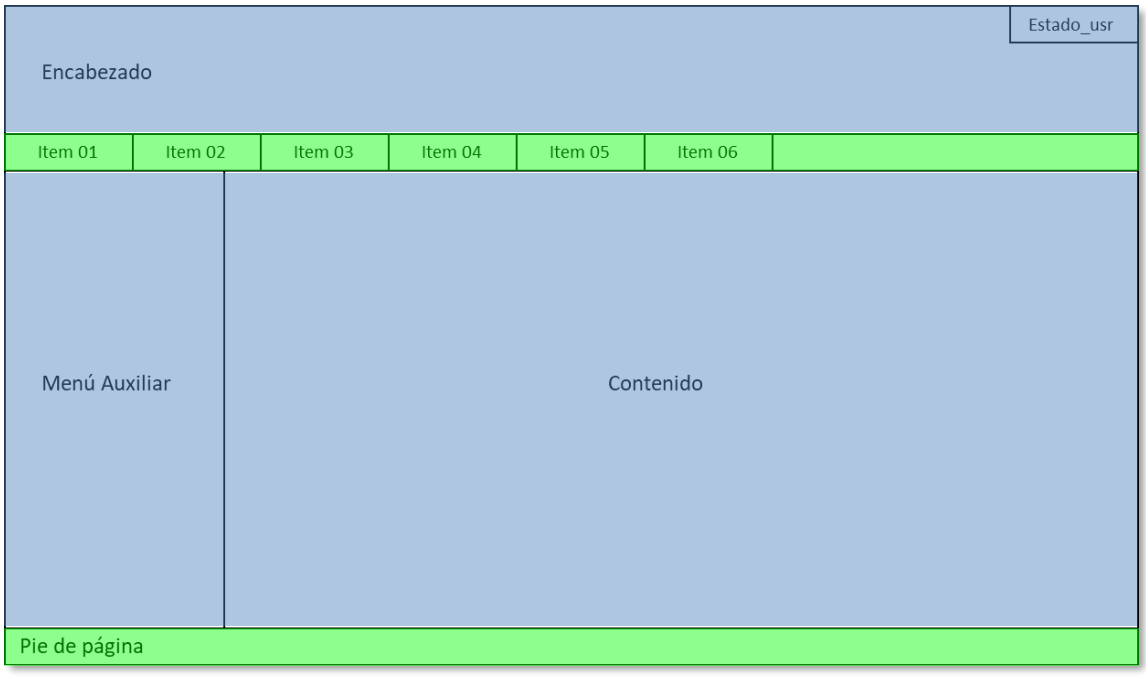
## Pasando de la maqueta al código

Existen varias formas de analizar estas maquetas, pero una de las formas más efectivas de lograrlo es mediante un enfoque tabular. En este enfoque, la distribución de los elementos se analiza de acuerdo a su ubicación en el layout siguiendo un patrón de filas y columnas: primero analizamos **en cuál fila** se encuentra el elemento y luego determinamos **en cuál columna**.

Este método nos permite generar un código más limpio y sencillo de mantener:

- Las filas que determinemos se convertirán en DIVs de 100% de ancho y altura relativa (determinada por porcentaje de la altura total del viewport).
- Las columnas que determinemos se convertirán en DIVs contenidos dentro de los DIVs de las filas, permitiendo determinar su posición simplemente asignando valores a las propiedades ***position*** y ***float***, mientras que sus dimensiones pueden determinarse tanto de forma relativa (un porcentaje del tamaño de la fila que contiene al elemento), como por valores absolutos.

Analicemos nuestra maqueta buscando las filas. Encontraremos cuatro filas bien determinadas:



Para mantener un orden y una coherencia en nuestro código, denominaremos a las filas que contengan más de un elemento con el prefijo ***area_.***

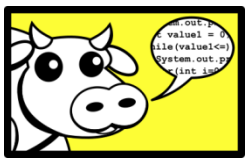
De esta manera obtenemos los siguientes DIVs:

- area_encabezado
  - Ancho: 15%
- area_menu
  - Ancho: 5%
- area_contenido
  - Ancho: 75%
- pie_de_pagina[2]
  - Ancho: 5%

Recordemos que todas las filas tendrán un 100% de ancho, por lo cual, para ahorrar código, especificaremos esta propiedad en una clase llamada ***fila*** y haremos que todos estos elementos pertenezcan a esta clase.

Para una mejor experiencia del usuario es mejor que toda la estructura sea visible en el navegador, por lo cual tomamos la precaución de que los anchos sumen 100%.

Con estos datos, ahora resulta sencillo generar el código HTML y CSS que corresponde a este documento. Creamos un directorio y, en su interior, los dos archivos necesarios: ***index.php*** y ***estilo.css***, recordando vincularlos en la cabecera de *index.php*.



**index.php**

```
<!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">
    <link rel="stylesheet" href="estilo.css">
    <title>Document</title>
</head>
<body>

    <div id="area_encabezado" class="fila"></div>
    <div id="area_menu" class="fila"></div>
    <div id="area_contenido" class="fila"></div>
    <div id="pie_de_pagina" class="fila"></div>

</body>
</html>
```

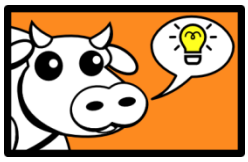
[2] (Los nombres están escritos sin acentuaciones para evitar posibles problemas en el código)

**estilo.css**

```
html,body{
    width: 100%;
    height: 100%;
    margin: 0px;
}

.fila{
    width: 100%;
}

#area_encabezado{
    height: 15%;
    background-color: lightblue;
}

#area_menu, #pie_de_pagina{
    height: 5%;
    background-color: lightgreen;
}

#area_contenido{
    height: 75%;
    background-color: lightblue;
}
```

Le agregamos al CSS de cada área un color de fondo para que se vean claramente, y una sección para aplicarle propiedades tanto a las etiquetas **`html`** como **`body`,** de forma que el viewport se muestre de forma correcta:

- **`width`** y **`height`**: Al estar a 100%, los elementos contenidos en ellos van a poder usar tanto `html` como `body` de referencia para medidas relativas. Esto es especialmente cierto para **la altura de los elementos**, que, de lo contrario, se adapta al contenido de los mismos.
- **`margin:`** Al establecer los márgenes internos a 0px, eliminamos el borde blanco del viewport y el contenido ocupa todo el viewport.



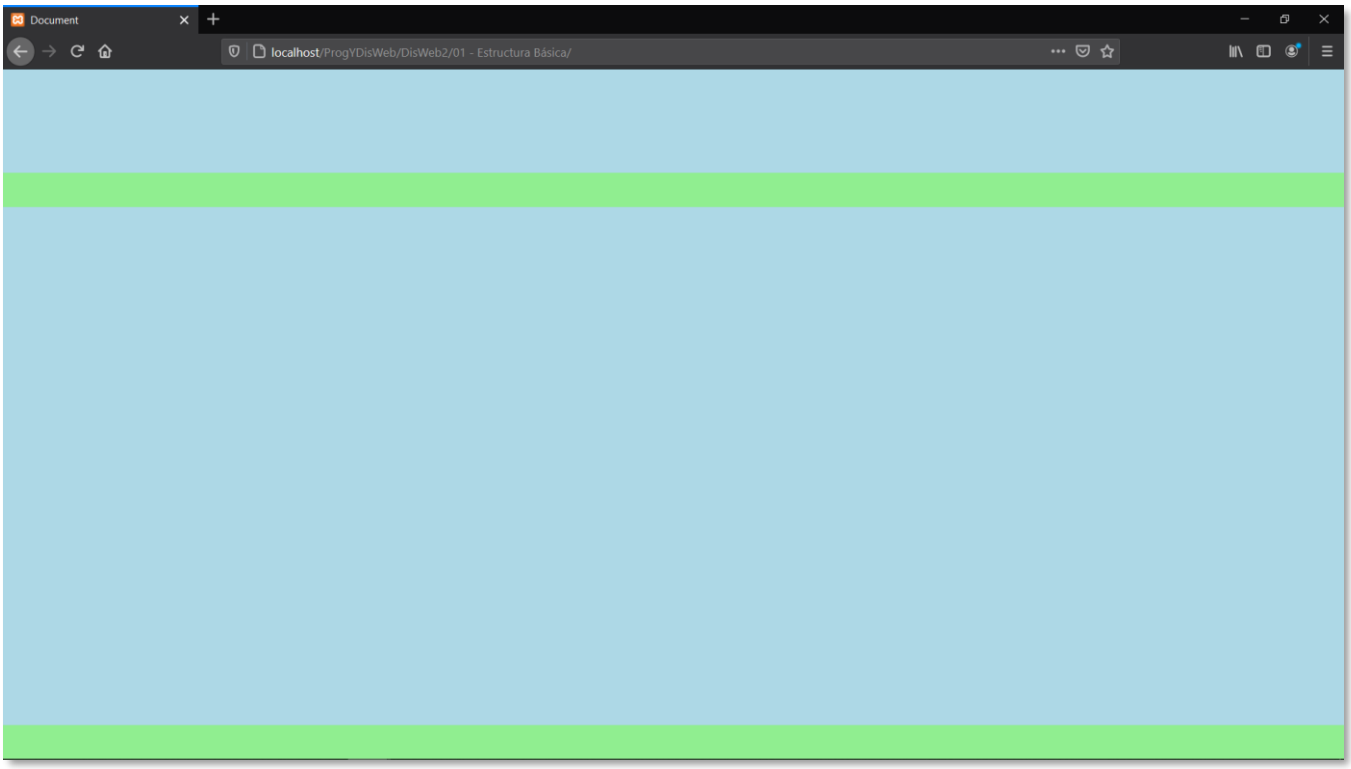
**Aplicar propiedades CSS a varios elementos a la vez:**

Cuando necesitamos aplicar las mismas propiedades a varios elementos al mismo tiempo, en CSS podemos indicarlo en una sección **nombrando a cada elemento y separando con comas** de esta manera:

```
elemento1, elemento2 {
      propiedad: valor;
      propiedad: valor;
}
```

De esta manera, las propiedades mencionadas entre las llaves se aplicarán a esos elementos, sin perjuicio de que los mismos tengan una sección propia en la hoja de estilos.

El resultado del código que acabamos de generar en un navegador web es el siguiente:




Se puede apreciar que las áreas que planteamos en la maqueta quedaron tal y como queríamos.