

TAREA OPTATIVA CSS:

1.- Motivación de la selección del módulo:

He elegido el módulo CSS Grid Layout Module porque me ha llamado la atención de entre los listados en el enunciado, debido a que en las interfaces de usuario de Java (WindowBuilder), hay un layout llamado GridLayout.

2.- Breve resumen de las características del módulo y su funcionalidad:

La principal característica de este módulo es la forma de posicionar los elementos del documento html en filas y columnas. La ventaja que esto ofrece al desarrollador es facilitarle el alineamiento de los elementos, no teniendo que preocuparse en implementar el posicionamiento a cada uno de los elementos.

En el documento HTML, los elementos se organizan en “tablas contenedoras”, y consisten en una etiqueta “div”. Cada elemento de la tabla será a su vez una etiqueta “div”.

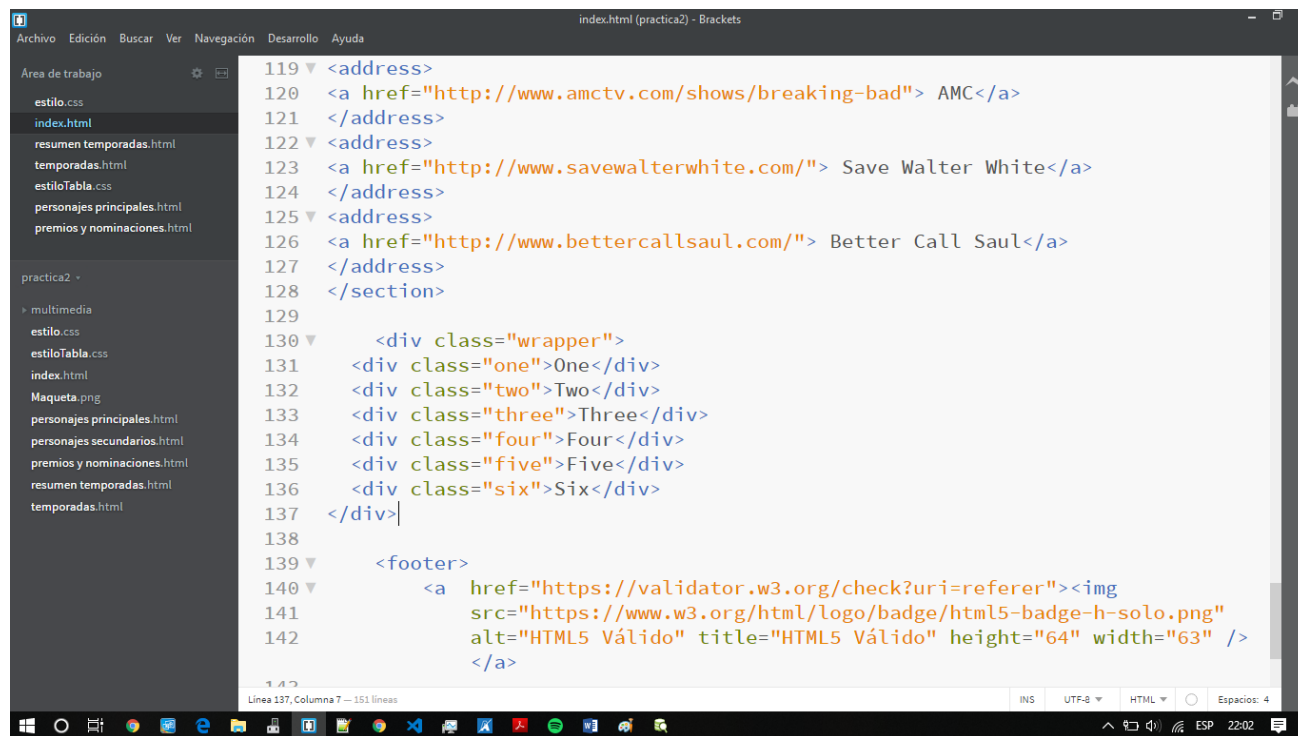
Luego en el documento CSS, para indicar que una etiqueta “div” es un contenedor grid, se define el valor “display = grid” o “display= inline-grid”.

Se pueden aplicar diferentes propiedades tales como grid-row-gap (grid-gap), que te deja definir el espacio entre filas, o grid-column-start, grid-column-end, grid-row-start, grid-row -end, grid-area... para especificar qué cantidad de filas y columnas va a ocupar un elemento determinado.

3.- Explicación detallada del contenido de los ejemplos realizados:

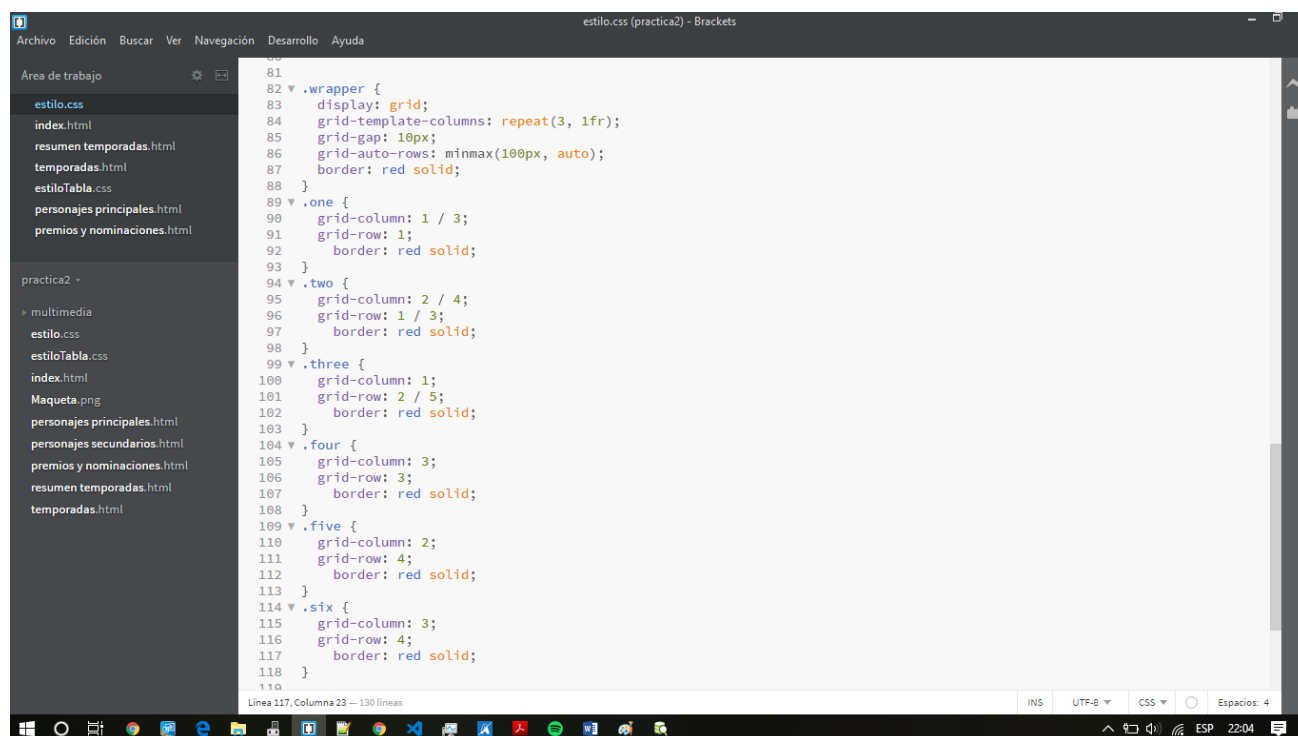
He realizado un ejemplo sencillo, con base la página de la práctica 2:

Documento HTML (index):



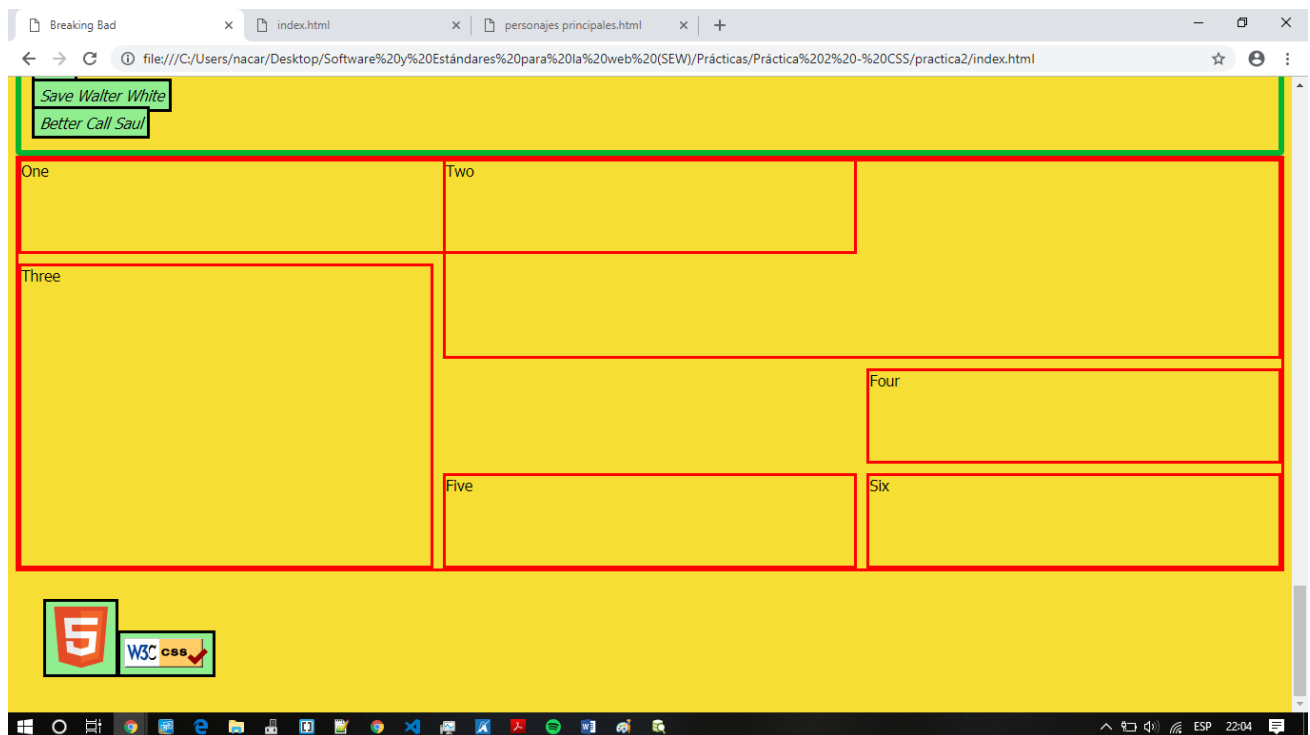
```
119 <address>
120 <a href="http://www.amctv.com/shows/breaking-bad"> AMC</a>
121 </address>
122 <address>
123 <a href="http://www.savewalterwhite.com/"> Save Walter White</a>
124 </address>
125 <address>
126 <a href="http://www.bettercallsaul.com/"> Better Call Saul</a>
127 </address>
128 </section>
129
130 <div class="wrapper">
131 <div class="one">One</div>
132 <div class="two">Two</div>
133 <div class="three">Three</div>
134 <div class="four">Four</div>
135 <div class="five">Five</div>
136 <div class="six">Six</div>
137 </div>
138
139 <footer>
140 <a href="https://validator.w3.org/check?uri=referer">
143 </a>
144
```

Documento CSS:



```
81
82 .wrapper {
83   display: grid;
84   grid-template-columns: repeat(3, 1fr);
85   grid-gap: 10px;
86   grid-auto-rows: minmax(100px, auto);
87   border: red solid;
88 }
89 .one {
90   grid-column: 1 / 3;
91   grid-row: 1;
92   border: red solid;
93 }
94 .two {
95   grid-column: 2 / 4;
96   grid-row: 1 / 3;
97   border: red solid;
98 }
99 .three {
100   grid-column: 1;
101   grid-row: 2 / 5;
102   border: red solid;
103 }
104 .four {
105   grid-column: 3;
106   grid-row: 3;
107   border: red solid;
108 }
109 .five {
110   grid-column: 2;
111   grid-row: 4;
112   border: red solid;
113 }
114 .six {
115   grid-column: 3;
116   grid-row: 4;
117   border: red solid;
118 }
119
```

Página web:



Explicación:

Para este sencillo ejemplo he creado un contenedor (wrapper), con seis elementos dentro (One, Two, Three, Four, Five, Six), todo ello con etiquetas div con valor class definido.

En el CSS, primero he definido una regla para wrapper, con sus características (se pueden apreciar en la captura de pantalla).

`display: grid;` Para establecer wrapper como un contenedor grid.

`grid-template-columns: repeat(3, 1fr);` Para definir el número de columnas del contenedor.

`grid-gap: 10px;` para definir el espacio entre filas y columnas.

`grid-auto-rows: minmax(100px, auto);` Define el número mínimo y máximo de hueco entre filas.

`border: red solid;` Borde rojo.

Después están las reglas para cada elemento del contenedor, con sus valores correspondientes para alienarlo dentro del contenedor, y borde rojo para que sea visible su disposición.

`grid-column: 1 / 3;` Columnas que ocupa el elemento.

`grid-row: 1;` Filas que ocupa el elemento.

`border: red solid;` Borde rojo.