Creación de componentes CSS con el estándar BEM

Miguel Oduber
12/03/2021

En esta pantalla tenemos varios elementos a destacar como el menú, el logo, la búsqueda ... Pero nuestra atención es el área de información del usuario registrado.

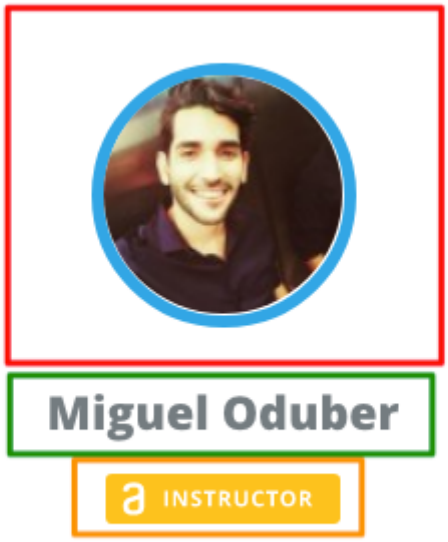
Nuestro objetivo al crear esta área es que se incluya los siguientes elementos:

- Foto
- Nombre de usuario
- Nivel de acceso en el sistema

Pero ¿cómo podemos hacer esto?

Entender cómo separar cada elemento

El primer paso es entender cómo será la separación de los elementos HTML que se crearán para hacer esta estructura, para eso será suficiente con dibujar unos cuadrados



en la diagramación, o dibujar en papel.

Ahora, con todo por separado, queda claro que necesitamos al menos 3 etiquetas html:

- `<img>` para el avatar;
- `<h2>` para el nombre;
- Para la etiqueta, para estandarizar, los diseñadores optaron por crear un svg para cada una, por lo que aquí tendremos otra etiqueta `<img>`.

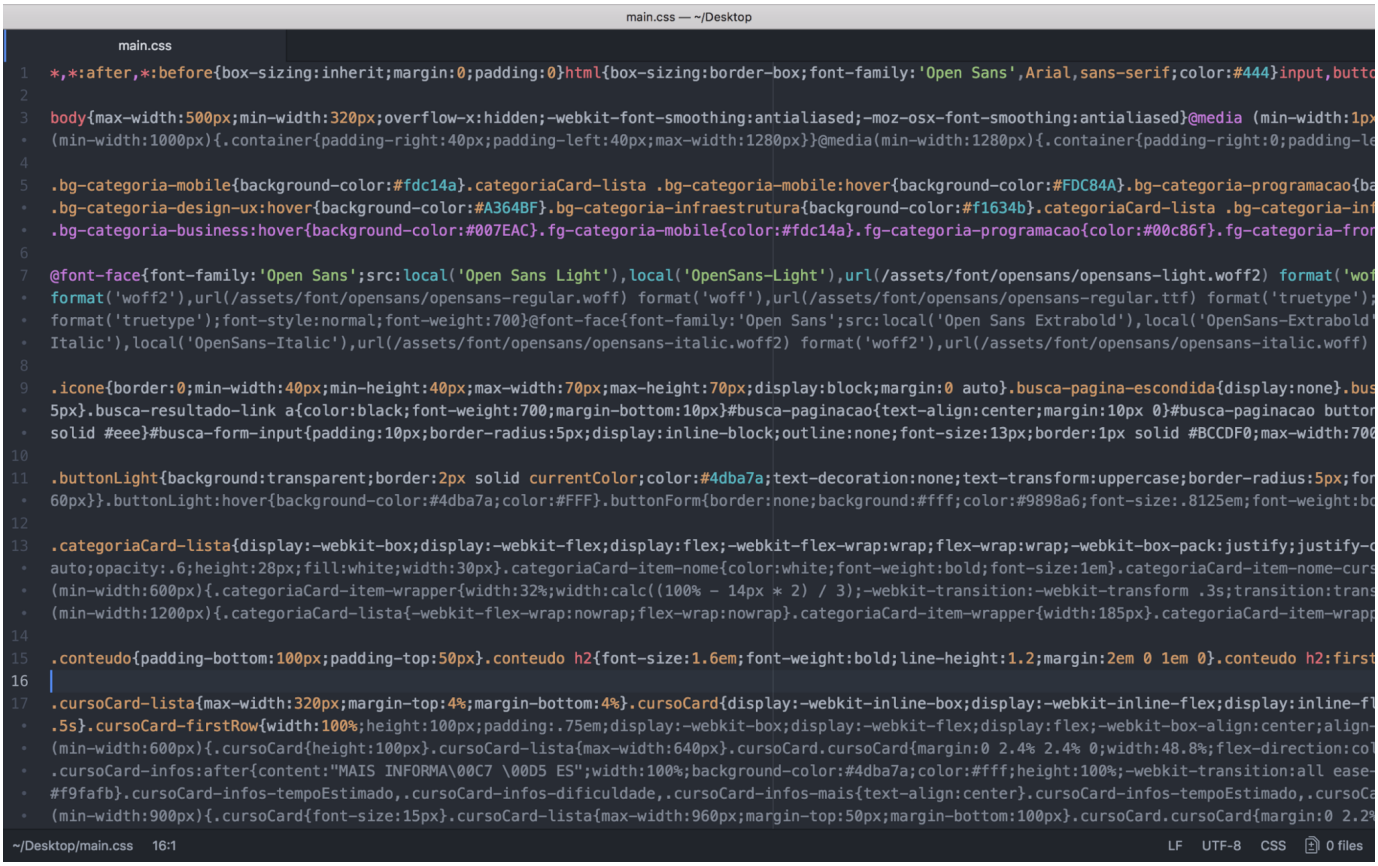
Para que sea más fácil agrupar estos elementos colocaremos una etiqueta `<div>` extra que involucra a todos los elementos.

```
<div class="infousuario">
  <img class="avatar" src="https://placehold.it/200x200" alt="" />

  <h2 class="nome">
    Nome do Usuário
  </h2>

  <img class="label" src="https://cursos.alura.com.br/images/gnarus/profile/l
</div>
```

Ahora que conocemos las etiquetas, ¿dónde vamos a diseñarlo todo?

Ir al CSS del proyecto

Con el HTML listo, necesitamos **diseñar** los elementos, así que abramos el archivo CSS principal, es decir, **main.css**:



Deshacerse del desorden de archivos

Para evitar que nuestra funcionalidad se pierda en este medio, hagamos nuestro estilo en un lugar más limpio, creando un nuevo archivo llamado **infousuario.css** y agregando todos los estilos que necesitemos en él.

HTML CSS Result EDIT ON

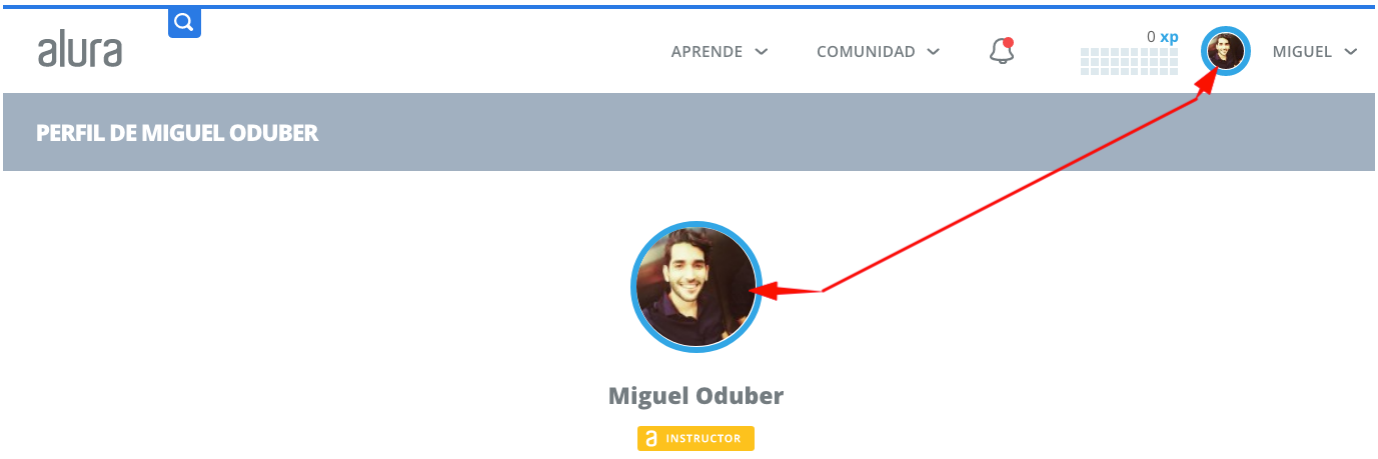
```css
/* infousuario.css */
.infousuario {
  text-align: center;
}
.avatar {
  margin-top: 20px;
  border: 9px solid #8db7d2;
  border-radius: 50%;
}
.nome {
  margin: 0;
  color: #747c81;
  margin-top: 20px;
}
.label {
  margin-top: 15px;
}
```

Ten en cuenta que en este archivo solo agregamos clases para diseñar nuestra página.

Como nuestras etiquetas HTML ya se han creado con clases basadas en lo que sería cada elemento, es más fácil diseñar en CSS simplemente llamando a la clase usando el punto (`.`) antes de su nombre.

Lamentablemente no todo es flores...

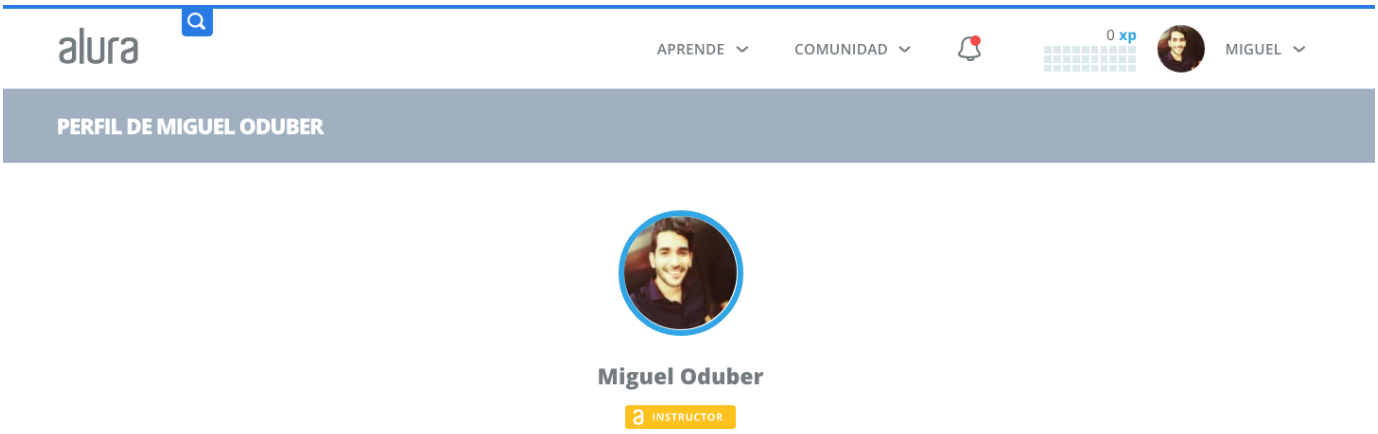
Aunque fue fácil de diseñar, cuando guardamos el código y probamos la nueva pieza de la interfaz en el entorno de desarrollo, presenta un conflicto visual con algo que ya se había implementado.



Ambos elementos son fotos con la clase avatar y tienen etiquetas , ¿cómo diferenciar cada uno de ellos?

```
.infousuario .avatar {}

.header .avatar {}
```

¡Bien! en este caso, resolvemos el problema y nuestra página vuelve a la normalidad:

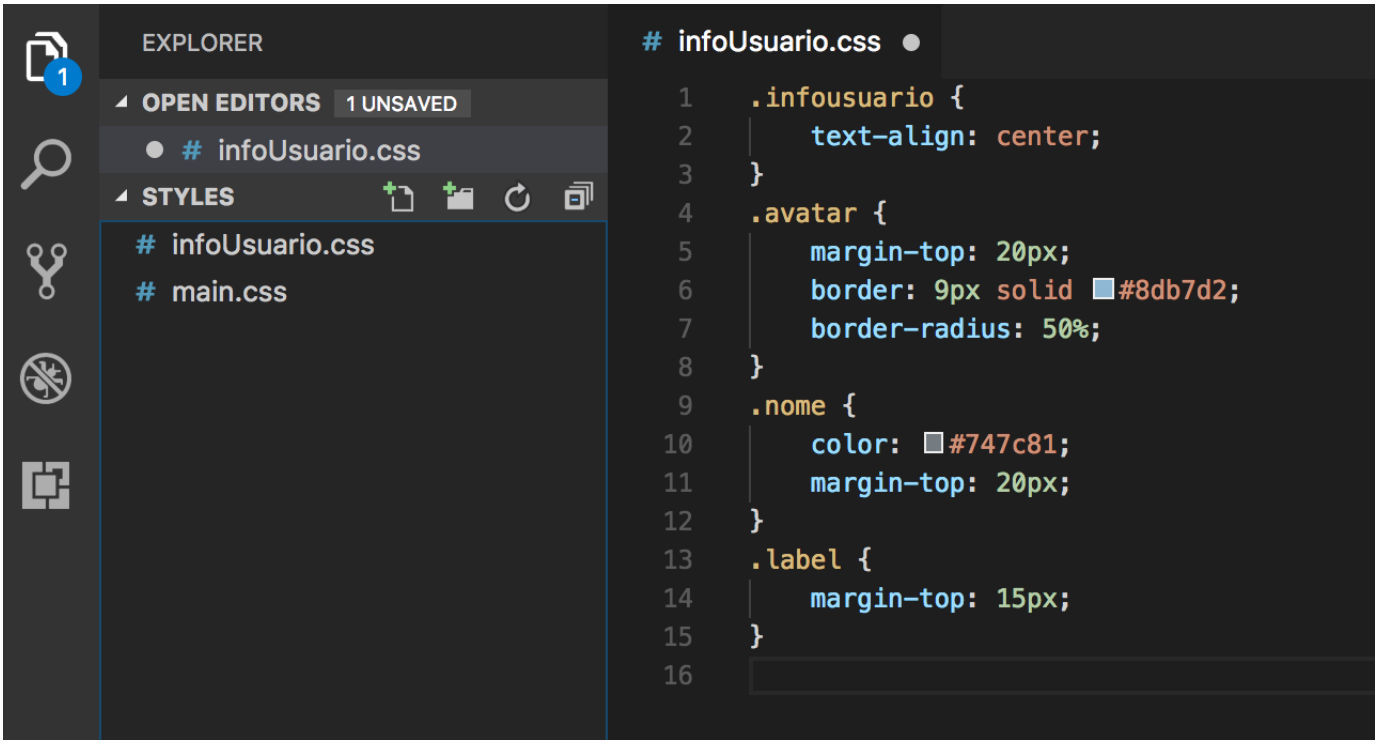


Pero aún así, nada impide que alguien cree un estilo sin anidar nuevamente y este tipo termina con nuestro diseño nuevamente :(

```
.avatar { }
```

Una luz al final del túnel

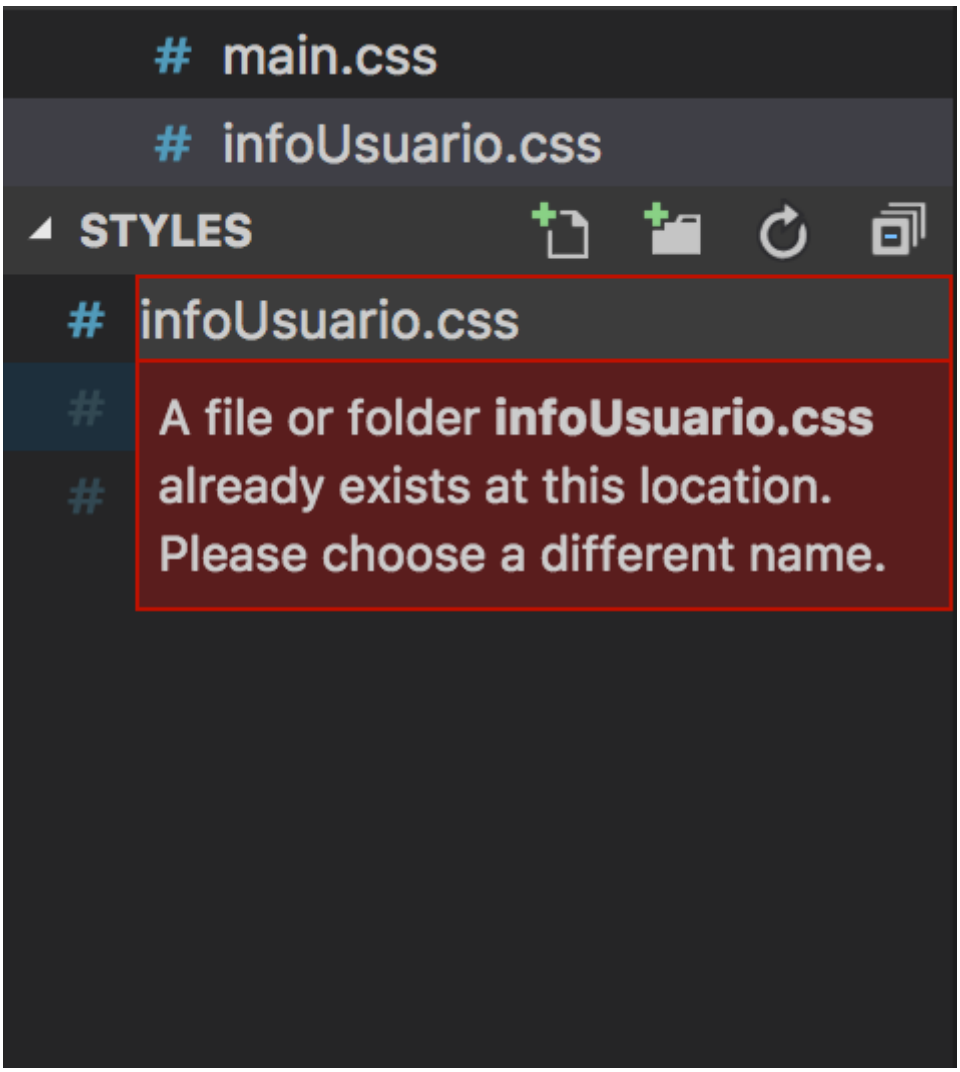
Hasta ahora, creamos un archivo **infousuario.css** y estábamos poniendo nuestro CSS en él, hasta ahora todo bien:



Sin embargo, el problema real comenzó a manifestarse cuando tuvimos que preocuparnos por si un estilo ya estaba creado o no.

Según nuestro problema, pensemos en lo siguiente:

Si creo un nuevo archivo **infousuario.css**, ¿qué pasa con el anterior?- Desaparece!



En una situación normal, antes de sobrescribir o darle al archivo un té faltante, se nos advierte que lo estamos haciendo a través del ordenador o editor de texto de su elección.

Momento de análisis

¿Cómo se puede representar el conjunto de HTML y css para generar la parte visual del *infousuario*?

¡Eso! Nuestro archivo **infousuario.css** junto con el extracto HTML se refieren a una parte del sitio, algunas personas llaman **bloque,** otras un **componente**.

En general, este es el responsable de varios elementos dentro de él que conforman lo que veremos sobre el usuario. Con esto en mente, todo lo que hay dentro de este tipo debe tener su estilo, como por ejemplo:

```
.infousuario .avatar {}
```

Pero mira cómo está, ¡todavía tenemos el riesgo de anidamientos que sobrescriban cosas!

Usando el estándar BEM

Para escapar de este caso de sobrescribir nuestros estilos, podemos usar un patrón de escritura de selector que ha sido ampliamente utilizado por la comunidad de front-end llamado **BEM** :

```
.infoUsuario__avatar {

}
```

En este escenario tenemos una clase en lugar de anidar, los dos subrayados pueden incluso sonar extraños al principio, pero BEM establece a través de ellos una relación entre los elementos padre e hijo sin anidar. Siendo el padre fue un bloque **B** e hijo un element (que viene de **BEM**).

De la forma en que lo hicimos aquí, ya estamos evitando conflictos de CSS con una clase `.avatar` que está suelta por el código.

¡Refactoricemos!

Con esta nueva estrategia, nuestro **infousuario.css se** ve así:

HTML CSS Result EDIT ON

```
/* avatar.css */
.infosUsuario {
  text-align: center;
}

  .infosUsuario__avatar {
    margin-top: 20px;
    border: 9px solid #8db7d2;
    border-radius: 50%;
  }

  .infosUsuario__nome {
    color: #747c81;
    margin: 0;
    margin-top: 20px;
  }

  .infosUsuario__label {
    margin-top: 15px;
    transition: .3s;
  }
```

¡Ahora nuestro CSS ya no está en conflicto!

```
.infosUsuario_avatar {
    margin-top: 20px;
    border: 9px solid #8db7d2;
    border-radius: 50%;
 }
```

Cuando definimos `border-radius` border-radius cómo `50%` estamos dejando que el avatar sea un círculo por defecto, pero si queremos que sea así solo en determinados casos, podemos decir que ser circular es una **modificación de** este **elemento** dentro del **bloque** `infoUsuario`.

Para crear su selector, escribimos las clases con dos guiones, `.avatar--circular` utilizando así la M de BEM (modificador) que sirve para representar la modificación de un elemento (en este caso para dejarlo circular o en un menú para activarlo) . Esto haría que nuestro CSS se vea así:

```
.infosUsuario__avatar {
    margin-top: 20px;
```

```
    border: 9px solid #8db7d2;
}

.infosUsuario__avatar--circular {
    border-radius: 50%;
}
```

HTML CSS Result EDIT ON

```
/* infousuario.css */
.infosUsuario {
  text-align: center;
}

  .infosUsuario__avatar {
    margin-top: 20px;
    border: 9px solid #8db7d2;
  }
  .infosUsuario__avatar--circular {
    border-radius: 50%;
  }

  .infosUsuario__nome {
    color: #747c81;
    margin: 0;
    margin-top: 20px;
  }

  .infosUsuario__label {
    margin-top: 15px;
    transition: .3s;
  }
```

Crear componentes reutilizables

Ahora nuestro avatar tiene estilos bien definidos y es más fácil trabajar con él, pero el avatar circular se usa en varios lugares de la web de alura: infosUsuario, encabezado, foro...

¿Qué podemos hacer en lugar de reescribir cada vez `border-radius` en cada uno de los lugares donde aparece?

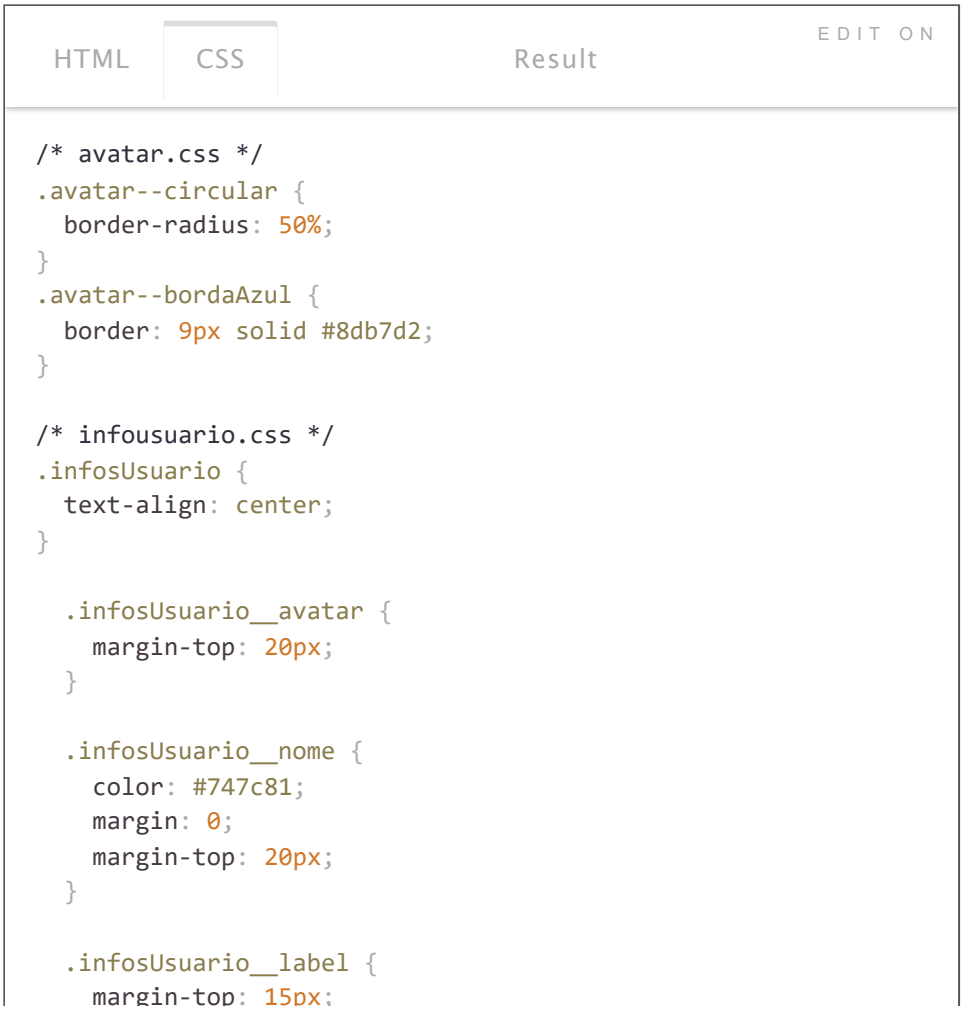
Podemos crear un componente de avatar. Un único punto que debemos tener en cuenta es que, el estado de ser un círculo y tener el borde azul cambia un estado de avatar predeterminado. Siguiendo esta línea, tenemos el siguiente resultado:

```
.avatar--circular {
   border-radius: 50%;
}

.avatar--bordaAzul {
   border: 9px solid #8db7d2;
}
```

Ahora, la única propiedad CSS que necesitamos cambiar en nuestro **infousuario.css** es margin-top:

```
.infosUsuario {
    text-align: center;
}

.infosUsuario__avatar {
    margin-top: 20px;
}

.infosUsuario__nome {
    color: #747c81;
    margin: 0;
    margin-top: 20px;
}

.infosUsuario__label {
    margin-top: 15px;
    transition: .3s;
}
```

¡Con eso nuestro componente está listo! También estamos listos para agregar nuevas funciones a nuestros proyectos con código reutilizable y evitar conflictos con éxito.



Si quieres profundizar en BEM te dejo el enlace a la **documentación para que la revises** , espero que el artículo te ayude en tus proyectos y no olvides compartir lo que haces.

Ahora bien, si quieres profundizar un poco más en el tema de la organización CSS conociendo otras formas además de BEM, te dejo este estupendo post . Si quieres aprender más sobre CSS, HTML, Javascript y otras tecnologías que forman parte de la web, aquí en Alura contamos con un **Front-End Training** .

En él aprenderás sobre HTML y CSS, Javascript, aprenderás marcos como jQuery y cómo crear sitios web receptivos.

Puedes leer también:

- Cómo organizar el CSS en tu proyecto
- Nombres de clases en CSS
- Centrar un elemento con CSS