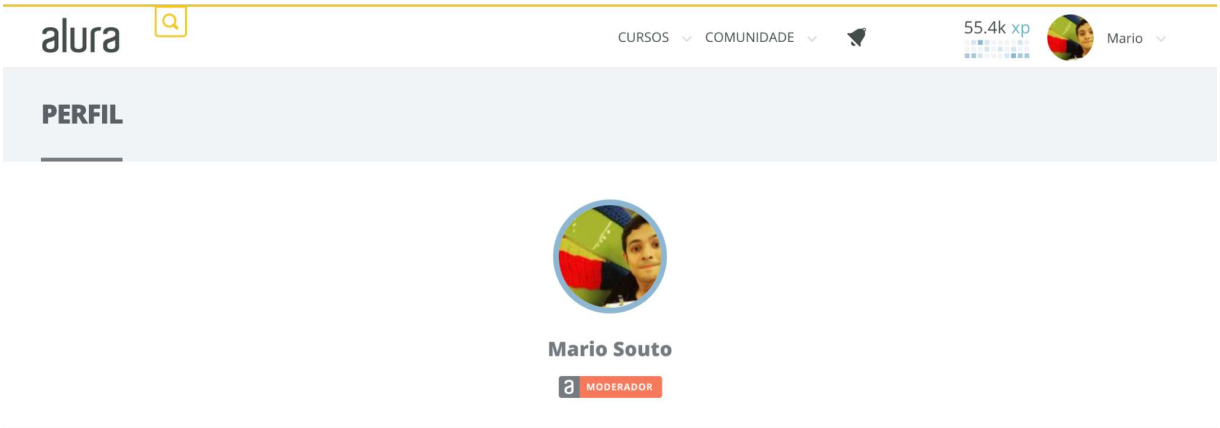
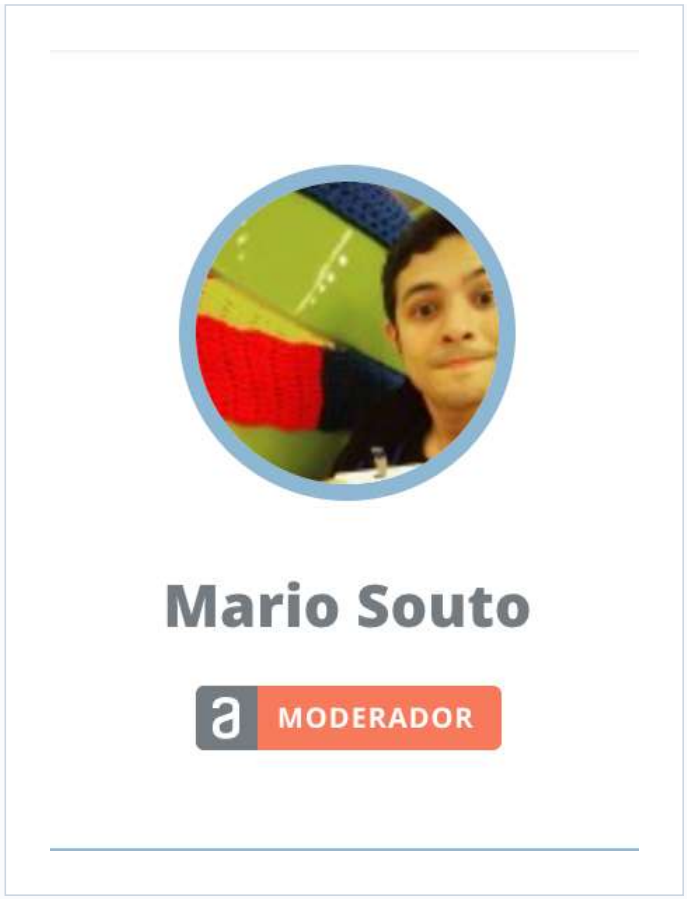
Mario Souto
Atualizado em 05/10/2021



Bom, mais um dia na vida do desenvolvedor e, ao chegar no trabalho, no mural de tarefas, temos a criação de uma feature nova.




Nesta tela, temos diversas coisas: o menu, o logo, a busca... Mas o nosso foco é a área de informação do usuário logado.



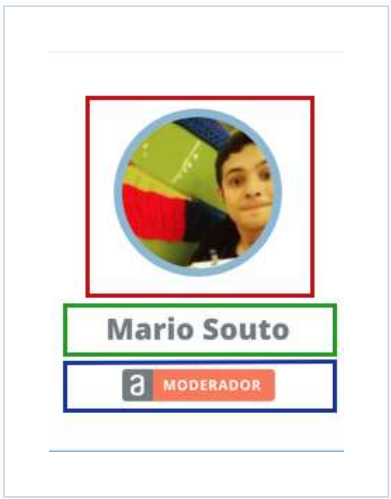

Nosso objetivo criando essa área é que ela contenha:

- A foto
- O nome do usuário
- A label informando o nível de acesso no sistema

Mas como podemos fazer isso?

# Entendendo como separar cada coisa

Bom, o primeiro passo é entendermos como será a separação dos elementos html que serão criados para fazer essa estrutura, para isso basta desenharmos alguns quadradinhos no layout, ou desenhar no papel.




Agora, com tudo separado fica mais claro que precisamos de pelo menos 3 tags:

- `<img>` para o avatar;
- `<h2>` para o nome;
- Para a label, com o intuito de padronizar, os designers optaram por criarem um svg para cada uma, então aqui teremos outra tag `<img>`.

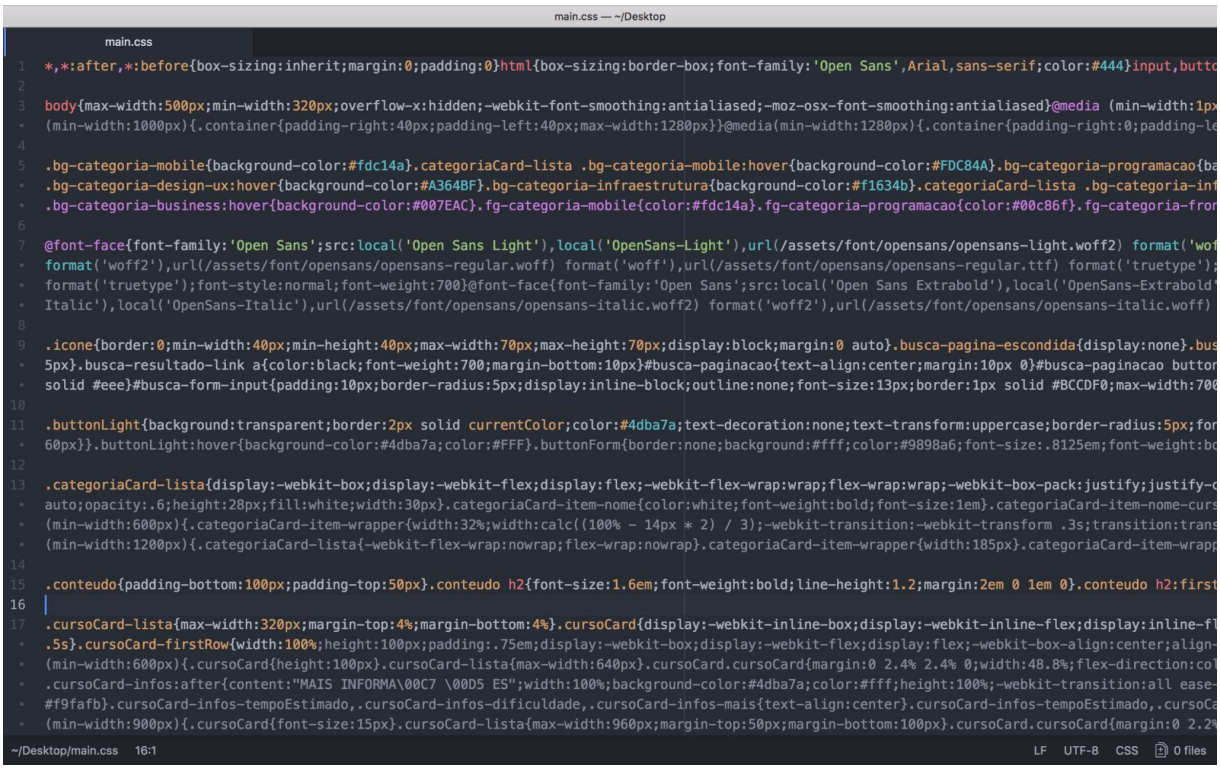
Para ficar mais fácil de agrupar esses elementos vamos colocar uma tag `<div>` extra que envolve todos os elementos.

```
<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/label-Moderador.svg"
    alt=""
  />
</div>
```

Agora que sabemos as tags, onde vamos estilizar isso tudo?

# Indo para o CSS do projeto

Com o html pronto, precisamos realizar a estilização dos elementos, portanto, vamos abrir o arquivo CSS principal, ou seja, o **main.css**:




> *"Opa, acho que o arquivo main.css, que controla todo o nosso css, está meio cheio/bagunçado :("*

# Se livrando da bagunça de arquivos

Para evitar que nossa funcionalidade fique perdida nesse meio, vamos fazer nosso estilo em um lugar mais limpo, criando um novo arquivo chamado **infousuario.css** e nele adicionar todos os estilos que precisamos.

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;
}
```

Resources

Observe que nesse arquivo adicionamos apenas classes para estilizar a nossa página.

Como nossas tags html já foram criadas com classes baseadas no que cada elemento seria, fica mais fácil de estilizar no CSS só chamando a classe usando o ponto ( `.` ) antes do seu nome.

# Infelizmente nem tudo são flores...

Embora tenha ficado fácil estilizar, quando salvamos o código e testamos o novo pedaço da interface no ambiente de desenvolvimento, ele apresenta um conflito visual com algo que já havia sido implementado.

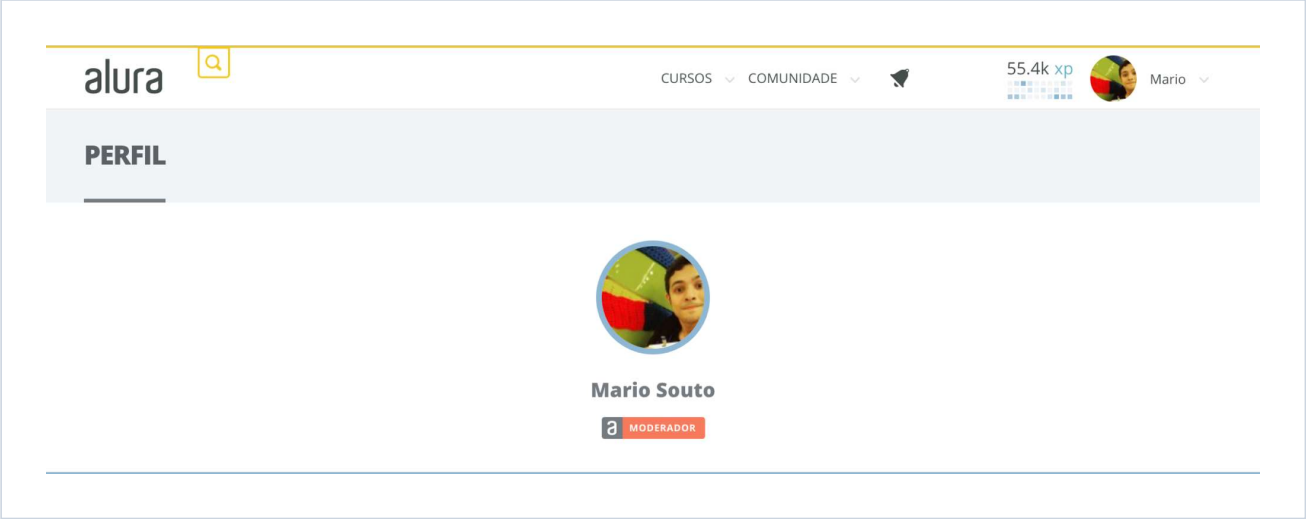

> *"Imagine se tivesse ido assim pra produção =O"*

Ambos os elementos são fotos com a class `avatar` e possuem tags `<img>`, como podemos diferenciar cada uma delas?

> *"Ah, podemos usar aninhamento de seletores, oras!"*

```
.infousuario .avatar {
   /\* propriedades
   do avatar filho de .infousuario \*/
}

.header .avatar {
  /\* propriedades do
  avatar filho de .header \*/
}
```

Boa! nesse caso, resolvemos o problema e nossa página volta ao normal:




Mas, ainda assim, nada impede que alguém crie um estilo sem aninhamento novamente e esse carinha acabe com o nosso layout novamente :(

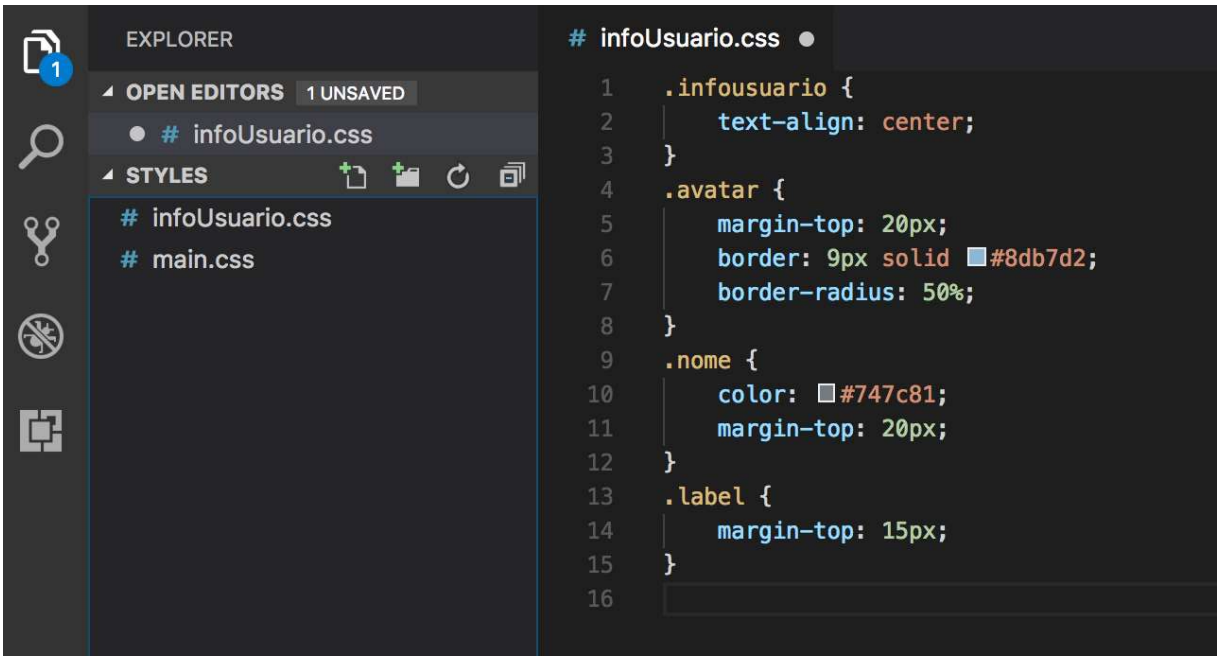
```
/\* arquivo_qualquer.css \*/

.avatar { /\* Destruindo o
  layout novamente muwahahahaha \*/ }
```

# Uma luz no fim do túnel

Bom, primeiro de tudo o que queremos fazer?

> *"Criar a nova funcionalidade sem quebrar o site T.T"*

Até agora, criamos um arquivo **infousuario.css** e nele estávamos colocando nosso CSS, até aí tudo bem:

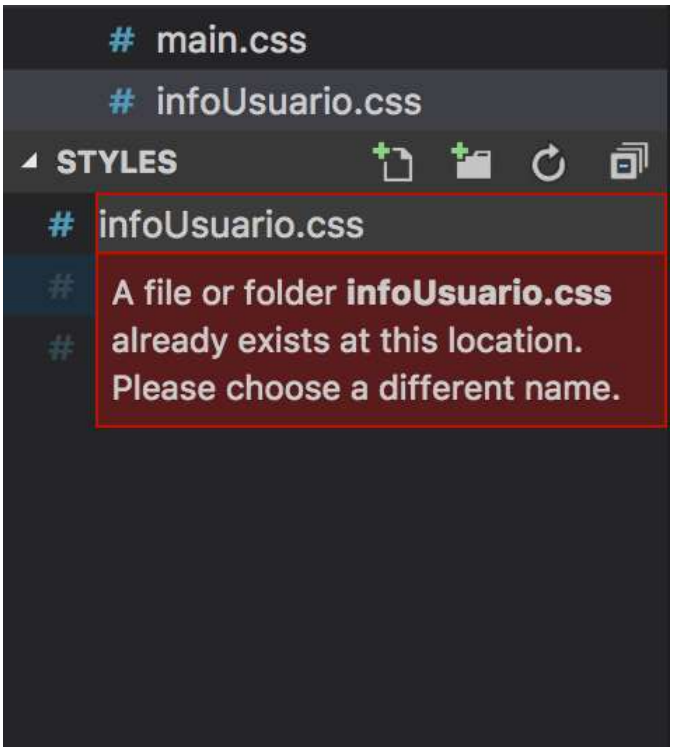



Entretanto, o real problema começou a se manifestar quando tivemos que nos preocupar em saber se um estilo já foi criado ou não.

Baseado no nosso problema, vamos pensar o seguinte:

Se eu criar um novo arquivo **infousuario.css**, o que acontece com o anterior?

> *"Ele some, uai"*




Quase! Em uma situação normal, antes de sobrescrever ou dar chá de sumiço no arquivo, somos avisados de que estamos fazendo isso pelo próprio computador ou editor de texto de sua preferência (no meu caso o VsCode).

# Momento análise

O conjunto do html e do css para gerar a parte visual do *infousuario*, pode ser representado de que forma?

> *"Ahh, ele é um pedaço dela!"*

Isso! Nosso arquivo **infousuario.css** junto com o trecho HTML referem a um pedaço do site, algumas pessoas chamam de **bloco** outras de **componente**.

No geral, esse é o cara responsável por vários elementos dentro dele que compõem o que iremos ver sobre o usuário. Tendo isto em mente, tudo que tiver dentro desse cara deve ser estilizado a partir dele, como por exemplo:

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

Mas veja que do jeito que está, ainda temos o risco dos aninhamentos que sobrescrevem coisas!

# Utilizando o padrão BEM

Para fugirmos desse caso de sobrescrita dos nossos estilos, podemos utilizar um padrão de escrita de seletores que vem sendo bastante utilizado pela comunidade de front-end chamado **BEM**:

```
.infoUsuario__avatar {
}
```

Nesse cenário temos uma classe ao invés do aninhamento, os dois underlines podem até soar estranho de início, mas o BEM estabelece por meio deles uma relação de elemento pai e elemento filho sem aninhamento. Sendo o pai um **b**loco e o filho um **e**lement (**be** que vem do **BE**M).

Do jeito que fizemos aqui já estamos evitando conflitos de CSS com uma classe `.avatar` que estiver solta pelo código :)

# Vamos refatorar!

Com essa nova estratégia o nosso **infousuario.css** fica da seguinte maneira:

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;
  }
```

Resources

Agora nosso CSS não está mais conflitando! \o/

> *"Mas, e se eu quiser reaproveitar algum estilo?"*

Boa pergunta! Um cara perfeito para analisarmos o reaproveitamento é o nosso avatar.

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

  border: 9px solid #8db7d2;

  border-radius: 50%;
}
```

Quando definimos o `border-radius` como `50%`, estamos deixando o avatar sendo um circulo por padrão, mas caso quisermos que ele seja assim apenas em determinados casos, podemos dizer que, ser circular é uma **modificação** deste **elemento** dentro do **bloco** `infoUsuario`.

Para criar o seletor dele nós escrevemos as classes com dois hífens, `.avatar--circular` utilizando assim o M do BEM (modifier) que serve para representar a modificação de um elemento (nesse caso para deixar circular ou em um menu para deixá-lo ativo). Isso deixaria nosso CSS assim:

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

  border: 9px solid #8db7d2;
}

.infosUsuario__avatar--circular {
  /\*os dois hifens,
  indicam que estamos
  modificando algum estilo do elemento \*/

border-radius: 50%;
}
```

HTML CSS Result EDIT ON

```css
/* 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;
  }
```

Resources

# Criando componentes reutilizáveis

Agora nosso avatar tem seus estilos bem definidos e fica mais fácil trabalhar com ele, mas o avatar circular é utilizado em diversos pontos no site da alura: infosUsuario, header, forum...

O que podemos fazer ao invés de reescrever toda vez o `border-radius` em cada um dos lugares que ele aparece?

> *"Podemos criar um componente avatar!"*

Exatamente! Um único ponto que precisamos levar em conta é que, o estado de ser um círculo e ter a borda borda azul alteram um estado padrão do avatar. Seguindo essa linha, temos o seguinte resultado:

```
/\*avatar.css\*/
.avatar--circular {
  border-radius: 50%;
}

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

Agora a única propriedade CSS que precisamos alterar no nosso **infousuario.css** é o margin-top:

```
/\* infousuario.css \*/

.infosUsuario {

text-align: center;

}

.infosUsuario__avatar {

margin-top: 20px; /\* Essa margem não conflita em lugar nenhum \*/

}

.infosUsuario__nome {

color: #747c81;

margin: 0;

margin-top: 20px;

}

.infosUsuario__label {
```

```
margin-top: 15px;

transition: .3s;

}
```

Com isso nosso componente está pronto! Também estamos com tudo certo para adicionar as novas funcionalidades nos nossos projetos com um código reutilizável e se esquivando com sucesso de conflitos.


Caso queiram ir mais a fundo no BEM, deixo o link da **documentação para vocês conferirem**, espero que o artigo ajude vocês em seus projetos e não deixem de compartilhar o que vocês fizerem :)

Ahh, e caso queiram ir um pouco mais a fundo no assunto organização de CSS conhecendo outras formas além do BEM, deixo este ótimo post do @marcobrunobr.

Se vocês quiserem aprender mais sobre CSS, HTML, Javascript e outras tecnologias que fazem parte da web, aqui na Alura temos uma **Formação Front-end**. Nela você aprenderá sobre HTML e CSS, Javascript, aprenderá frameworks como jQuery e como criar sites responsivos.