Banking) com Angular 6 — na prática e sem complicações parte 2

Danilo Agostinho Follow

Aug 8, 2018 · 8 min read

- Olá pessoal, tudo certo?

  Este post é a segunda parte de uma série que ensina a criar um CRUD

completo usando Angular 6. Caso não tenha lido a primeira parte, recomendo que leia clicando neste aqui.

Como prometido, iremos criar e estruturar nossa aplicação. Aos poucos, vai se tornando uma interface amigável.

Você também pode baixar o projeto que está no Github clicando neste link.

Criando os components

Vamos evoluindo aos poucos, uma coisa de cada vez. Observe que um padrão de components que vemos no Internet Banking da vida são:

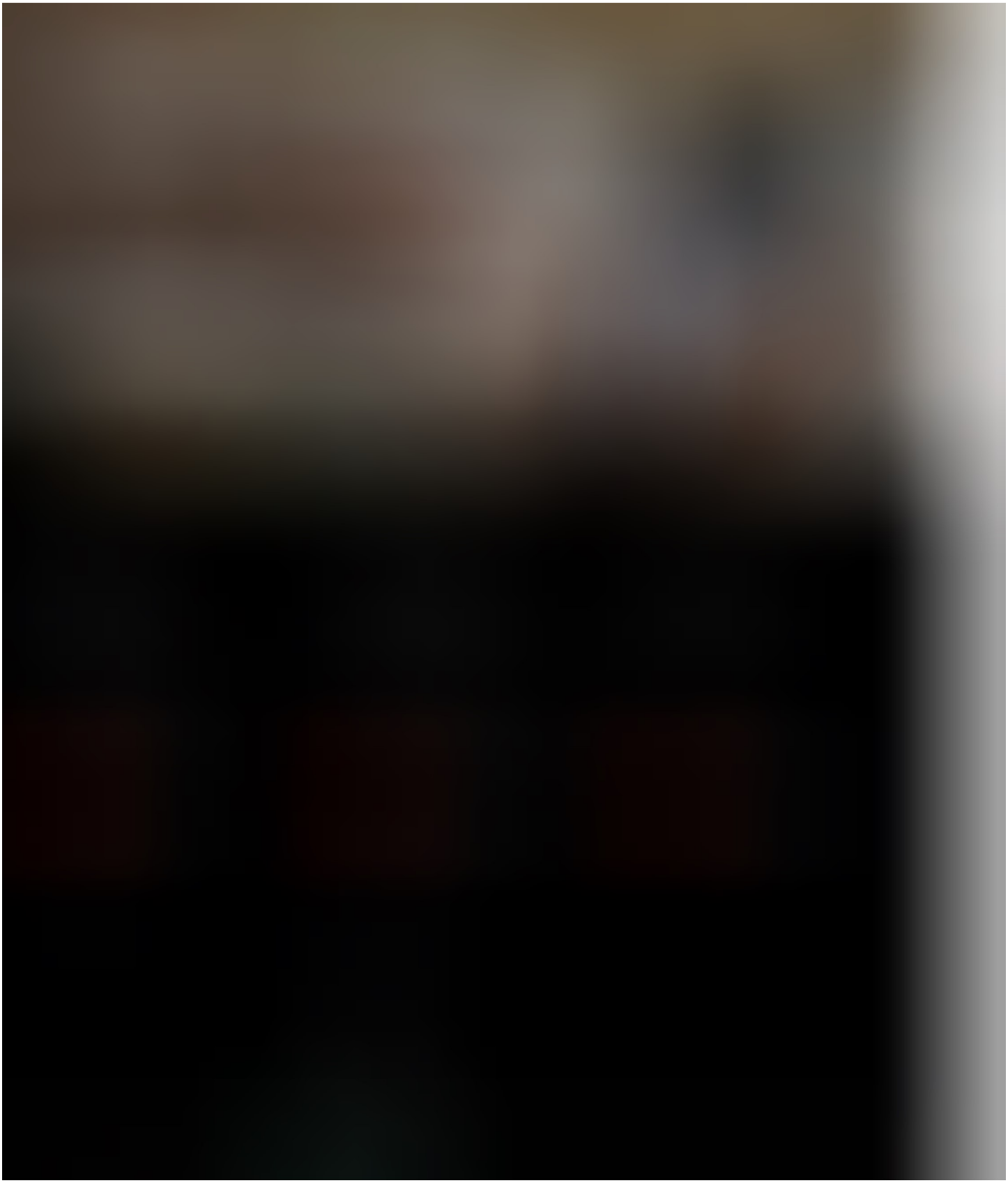
header;

content;

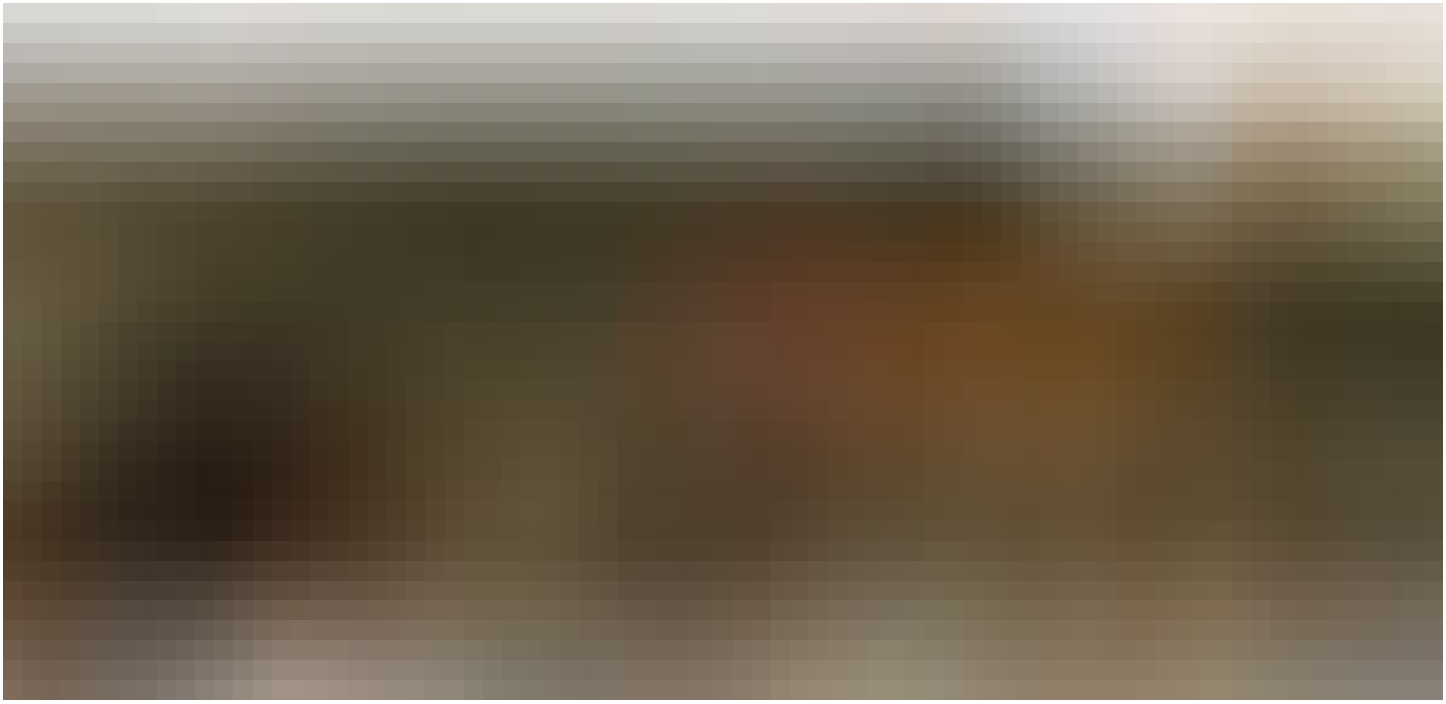
footer.

Analise as imagens:

Santander



Itau



Partindo deste princípio, vamos criar três components usando o Angular CLI. Gere os components da seguinte forma:

Component header

```
ng g component header
```

Component content

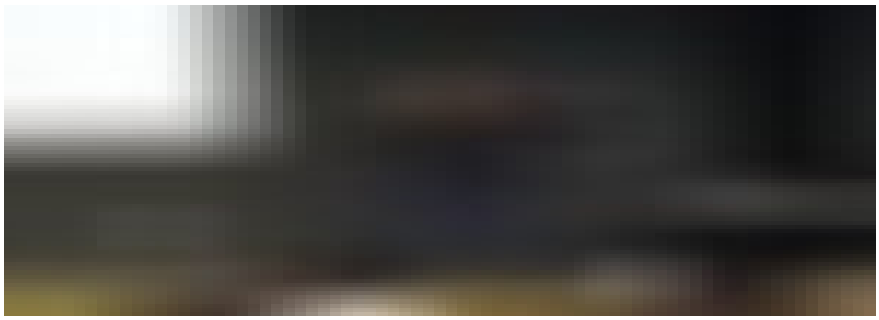
```
ng g component content
```

Component footer

```
ng g component footer
```

1. Repare que o prompt mostra que geramos os components e estes foram declarados no arquivo **app.module.ts**.

   Veja como o **app.module.ts** ficou:



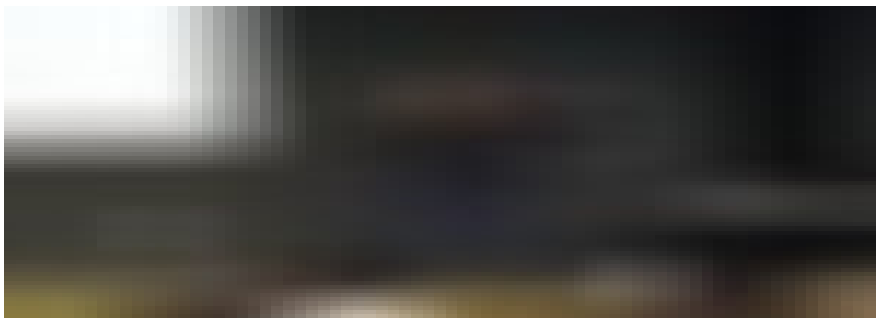
app.module.ts da aplicação

Isso é tudo o que precisamos para iniciarmos a componentização da aplicação. Mas, antes disso eu gostaria que soubessem que um component do Angular significa um pequeno trecho de código que podemos reutilizar em qualquer lugar da aplicação além disso, um component é divido em duas partes:

Template html

Pode ser visto como os arquivos HTML, onde fica toda a responsabilidade da marcação das tags, semântica, por assim dito.

Vamos analisar o arquivo **header.component.html**:



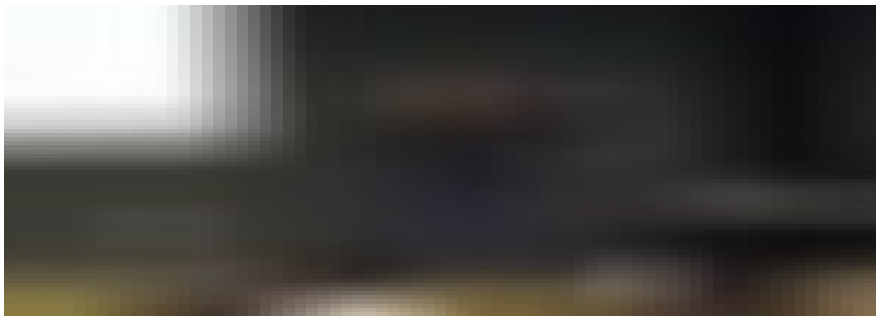
Viu que simples? Puro HTML, aqui podemos brincar à vontade.

2. **Parte lógica do component**

Essa parte é dedicada a toda a lógica da nossa página (Javascript). Daí que surgiu o conceito do Angular de **Property Bind** e **One Way Data Bind.** Essas palavras bonitas significam a ligação de dados entre nosso HTML com nosso arquivo lógico **JavaScript**.

Vamos analisar nosso arquivo de lógica:

Abra o arquivo **header.component.ts**



Não se apegue aos detalhes, observem apenas que temos um console.log dentro do **ngOnInit().**

Dá pra imaginar que assim que esse component app-header iniciar, veremos no console do navegador a mensagem “Component app-header iniciado..” Mais uma vez, não desvie sua atenção. Foco na prática!

Componentizando a aplicação

No momento, nossa aplicação está com esta aparência:

Preview da aplicação até o momento

Abra o arquivo **app.component.html** e atualize com as tags dos novos components

Component header

```
<app-header></app-header>
```

Component content

```
<app-content></app-content>
```

Component footer

```
<app-footer></app-footer>
```

Feito isso nossa home começa a ter a cara do nosso **Training Banking**:

Perceba que inserimos os components no grid do bootstrap e agora temos três colunas com os títulos (**header works**, **content works**, **footer works**). Agora perceba que no console do navegador vemos a mensagem que declaramos no **ngOnInit** do component **app-header.**

Criando o layout do projeto

Agora que sabemos como funciona um component no Angular está na hora de criar nosso Layout HTML. Vamos usar o bootstrap para ser o mais prático possível. Fique à vontade para estilizar suas tags HTML.

Abra o arquivo **app.header.component.html** e atualize-o com a marcação abaixo:

```
<form>

<div class="form-row align-items-center">
```

```
<div class="col-sm-3 my-1">

<label class="sr-only"
for="inlineFormInputName">Name</label>

<input type="text" class="form-control"
id="inlineFormInputName" placeholder="Jane Doe">

</div>

<div class="col-sm-3 my-1">

<label class="sr-only"
for="inlineFormInputGroupUsername">CPF</label>

<div class="input-group">

<div class="input-group-prepend">

<div class="input-group-text">@</div>

</div>

<input type="text" class="form-control"
id="inlineFormInputGroupUsername" placeholder="Username">

</div>

</div>

<div class="col-auto my-1">

<div class="form-check">

<input class="form-check-input" type="checkbox"
id="autoSizingCheck2">

<label class="form-check-label" for="autoSizingCheck2">

Remember me

</label>

</div>

</div>
```

```
<div class="col-auto my-1">

<button type="submit" class="btn btn-
primary">Acessar</button>

</div>

</div>

</form>
```

Agora abra o arquivo **app.component.html** e atualize como abaixo:

```
<h1>Training banking</h1>

<button mat-button (click)="openDialog()">Open
dialog</button>
```

```
<div class="container">

<div class="row">

<app-header></app-header>

<app-content></app-content>

<app-footer></app-footer>

</div>

</div>
```

Removi as **.cols** (classe de colunas do bootstrap) e deixei um component um abaixo do outro.

Se você seguiu os passos acima sua aplicação deve estar parecido com isto:

3. *Qualquer semelhança com algum Internet banking é mera coincidência*

- Eu tentando não plagiar as coisas.

Vamos continuar atualizando nosso **app-header.component.html**

```
<div class="wrapper-header">

<div class="row">

<div class="col-xs-12 col-md-8">

<ul class="menu-header">

<li>Abra sua conta</li>

<li>Investimentos</li>

<li>Planos</li>

</ul>
```

```
</div>

<div class="col-xs-12 col-md-4">

<form>

<div class="form-row align-items-center">

<div class="col-auto my-1">

<div class="form-check">

<label class="form-check-label" for="autoSizingCheck2">

<b>Acesse</b> sua conta

</label>

<input type="text" class="form-control"
id="inlineFormInputName" placeholder="Digite seu CPF">

</div>

</div>

<div class="col-auto my-1">

<button type="submit" class="btn btn-primary acessar-
conta">Acessar</button>

</div>

</div>

</form>

</div>

</div>

</div>
```

Uma coisa que esqueci de comentar com vocês é que os components também possuem uma parte dedicada a estilização. Portanto, um

component Angular segue o padrão:

Template HTML (app-header.component.html)

Lógica do component(app-header.component.ts)

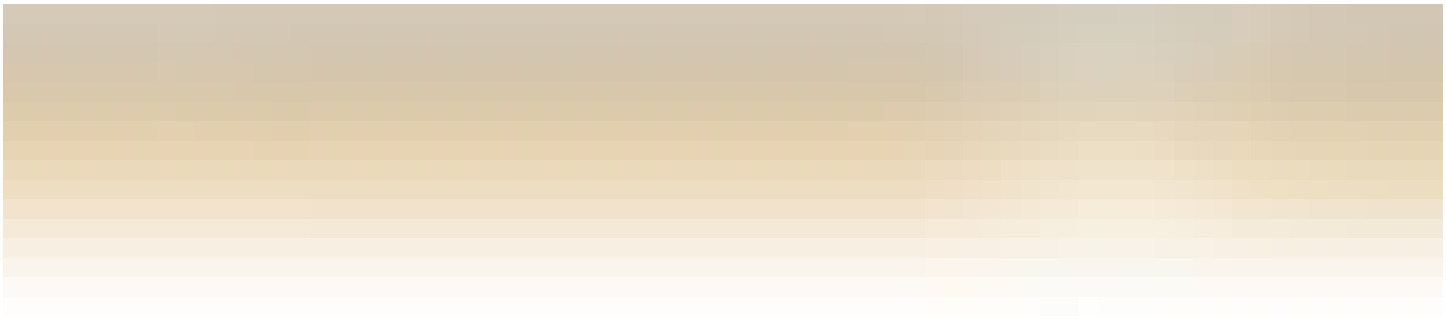
Estilo do component (app.header.component.css)

Agora atualize o arquivo **app-header.component.css**

```
.wrapper-header  {
height: 30vh;
padding: 30px;
background: #dcbc81;
color: #fff;
box-shadow: 0 2px 5px -2px grey;
}
.menu-header li  {
display: block;
font-size: 16px;
margin-left: -24px;
padding: 0px;
}
.acessar-conta {
position: relative;
top: 10px;
background-color: #fff;
```

```
border: 0;

color: #333;

}

@media screen and (min-width: 600px) {

.wrapper-header {

height: 15vh;

}

.menu-header li {

list-style: none;

display: inline;

position: relative;

margin: 10px;

top: 30px;

font-size: 18px;

}

}
```

Se você seguiu os passos anteriores, sua tela se parecerá com isto:



Abra o arquivo **app-content.component.html** e atualize segundo a marcação abaixo:

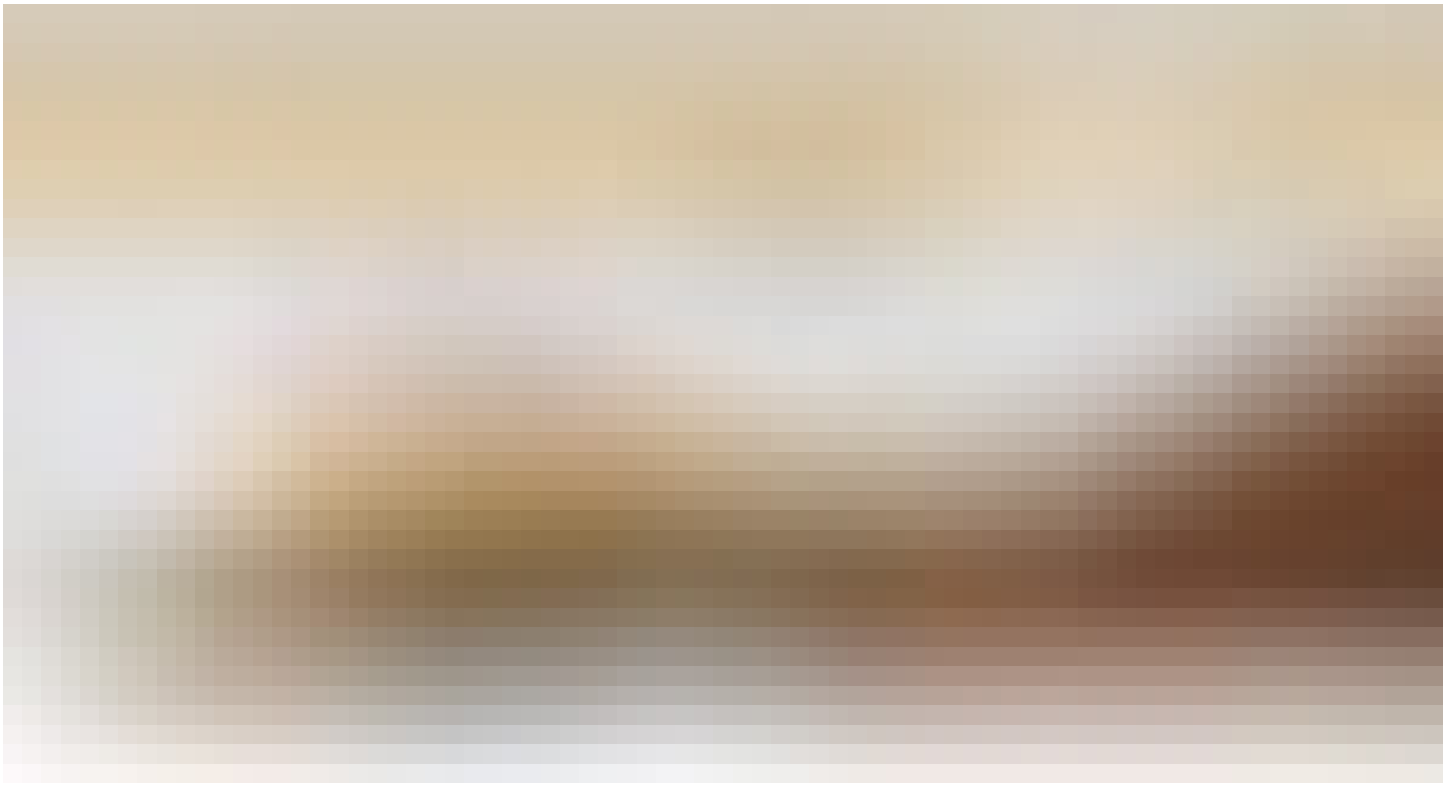
```
<div class="wrapper-content">

<div class="box-tile">

<h1><b class="tag">#</b>Traning banking juros amigo</h1>

</div>

<div class="bg_img"></div>

</div>
```

e edite o arquivo **app-content.component.css** com o estilo:

```
@import url('https://fonts.googleapis.com
/css?family=Crete+Round|Poppins');

.wrapper-content {

width: 100vw;

height: 70vh;

z-index: -9999;

top: -60px;

}

.box-tile {

width: 40vw;

height: 100px;

position: absolute;

top: 30%;

left: 10%;

padding: 10px;

}

.wrapper-content h1 {

font-size: 60pt;

color: #fff;

text-shadow: 2px 2px #333;

font-family: 'Crete Round', serif !important;

}

.tag {
```

```
font-size: 2-pt;

color: chocolate;

font-weight: bold;

}

.bg_img{

right: 0;

bottom: 0;

min-width: 100%;

min-height: 100%;

width: auto;

height: auto;

z-index: -1000;

background: url('https://github.com/daniloagostinho/traning-
book/blob/master/src/assets/images/bg_banking.png?raw=true')
no-repeat;

background-size: cover;

}
```

Nossa aplicação aos poucos está tomando vida:



Abra o arquivo **app-footer.component** e atualize-o:

```
<div class="wrapper">

<div class="topicos">

<div class="container">

<div class="row">

<div class="col-3">

<ul class="topicos-footer">

<h6>Traning Banking</h6>

<li>Quem somos</li>

<li>Nossos serviços</li>
```

```
<li>Carreiras</li>

<li>Desenvolvedores</li>

</ul>

</div>

<div class="col-3">

<ul class="topicos-footer">

<h6>Para você</h6>

<li>Abra sua conta</li>

<li>Cartões de crédito</li>

<li>Créditos e finaciamentos</li>

<li>Investimentos pessoa fisica</li>

</ul>

</div>

<div class="col-3">

<ul class="topicos-footer">

<h6>Segmentos</h6>

<li>Comercial banking</li>

<li>Seguros </li>

<li>Tesouro direto</li>

</ul>

</div>

</div>

</div>
```

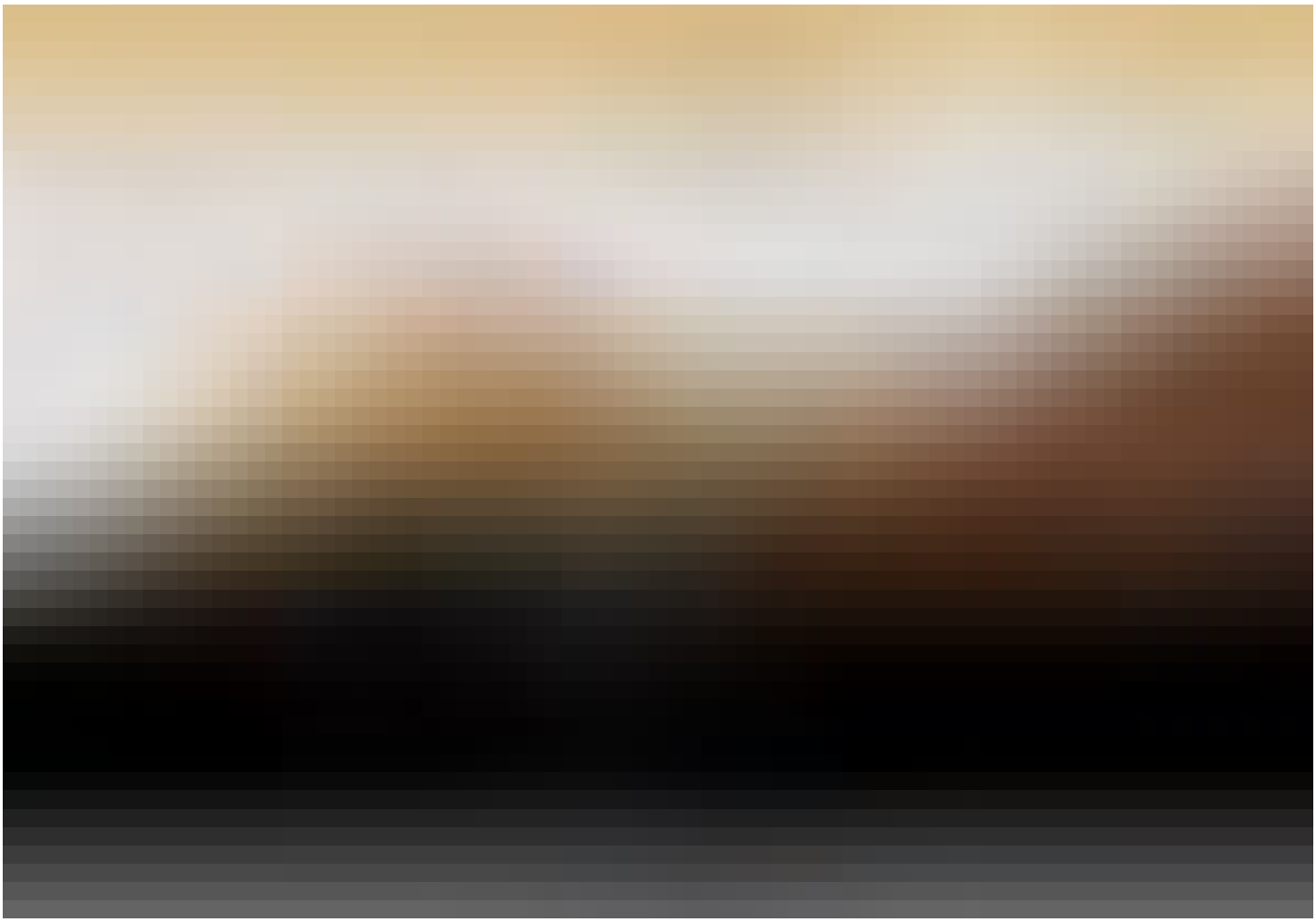
```
</div>

<footer class="rodape">

<p class="text-center">&copy; Todos os Direitos reservados a

<b>Traning Banking</b>

</p>

</footer>

</div>
```

E, claro, não esqueça de atualizar o arquivo de estilo do **footer.component.css:**

```
.topicos {

width: 100vw;

height: 50vh;

background: #000;

color: rgba(255, 255, 255, 0.6) !important;

padding: 90px;

}

.topicos h6 {

color: #fff;

}

.topicos li {

list-style: none;
```

```
line-height: 3;

}

.rodape {

background: #ddd;

height: 5vh;

}

.rodape p {

padding-top: 6px;

}
```

Como resultado, temos:



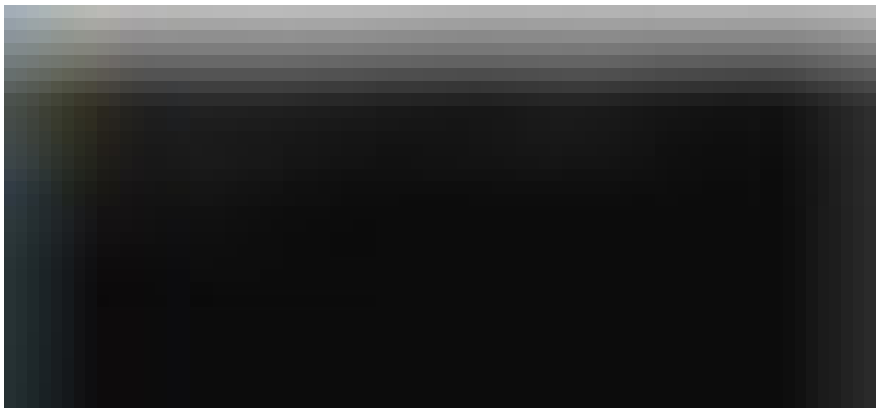
1. ## Extra: adicionando Typed.js à aplicação

O Typed.js é uma biblioteca de animações em JavaScript. Com ela, podemos criar aqueles efeito bacana de escrita.vamos integrá-lo ao Angular, aproveite!

Instalando o Typed.js

Cole o comando abaixo no seu prompt/terminal

```
npm install typed.js
```



Cole o comando abaixo no seu prompt/terminal

```
npm install typed.js
```

Abra o arquivo **content.component.ts** e atualize seguindo os passos:

Importe o Typed.js

```
import * as Typed from 'typed.js';
```

2. No **ngOnInit(),** crie uma constante com as configurações da animação

```
const options = {

stringsElement: '#typed-strings',

strings: ['Training banking juros amigo', 'Peça a
portabilidade e venha ser Training Bankinhg hoje mesmo',
'Inovavor digital e seguro'],

typeSpeed: 100,
```

```
backSpeed: 100,

backDelay: 200,

smartBackspace: true,

fadeOut: true,

showCursor: false,

startDelay: 1000,

loop: true

};
```

3. **Crie uma constante inicializando o Typed.js**

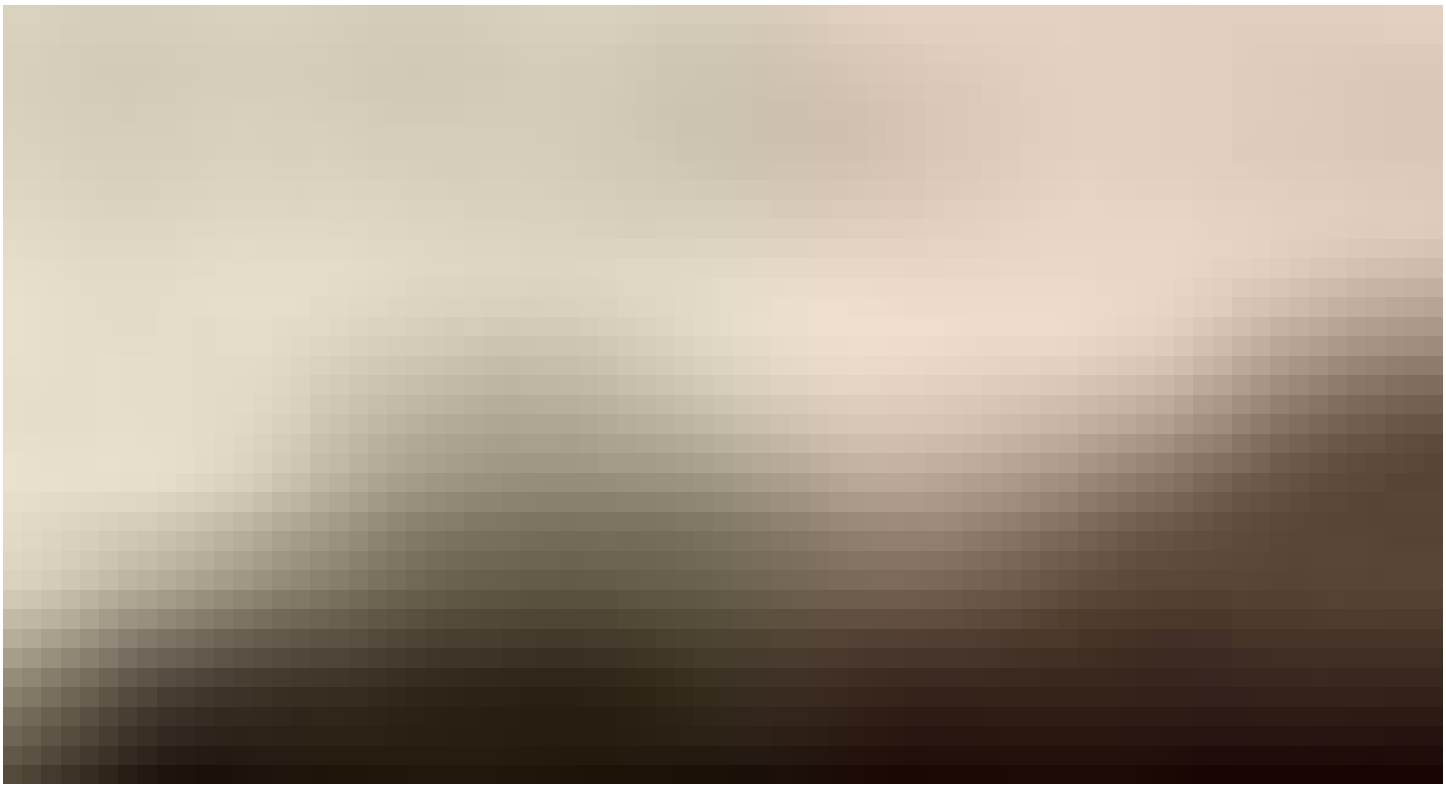
```
const typed = new Typed('.typing-element', options);
```

Agora é a vez de editar o arquivo **content.component.ts:**

Insira a marcação abaixo:

```
<div class="wrapper-content">

<div class="box-tile">

<b class="tag">#</b>

<h1 class="typing-element"></h1>

</div>

<div class="bg_img"></div>

</div>
```

Agora veja como ficou nossa aplicação:



O post de hoje vai ficando por aqui. Na parte 3 da série vamos criar a navegação da aplicação usando o router do Angular.

Existem uma série de boas práticas que poderíamos usar na aplicação, por exemplo: escalar o CSS com algum pré-processador, tornar a marcação do HTML mais semântica, etc. No entanto, tenhamos em mente que quem está começando precisa primeiro criar um projeto e ganhar confiança para só depois ir melhorando a aplicação.

Bem é isso, espero vocês na parte 3.

Qualquer dúvida pinga nos comentários e se quiser trocar figurinhas comigo me chama lá no Twitter @danilodev.silva

Grande abraço!