everis

an NTT DATA Company

O Front, o Angular e a Acessibilidade

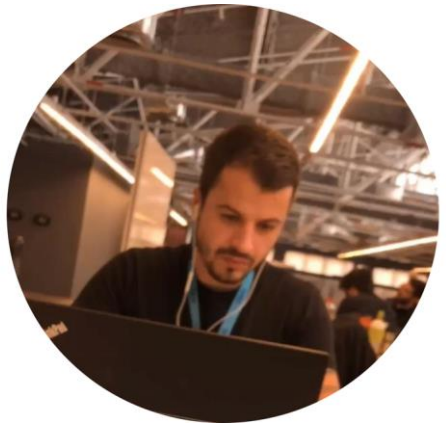
Expert Talk - Digital Innovation One e everis

Lucas Schoemberger Sales e Thamyris da Silva Barbarino- Novembro de 2020

Índice

1. Apresentação
2. O Desenvolvedor
3. Angular
4. Demo
5. Acessibilidade
6. Resumo.

Apresentação

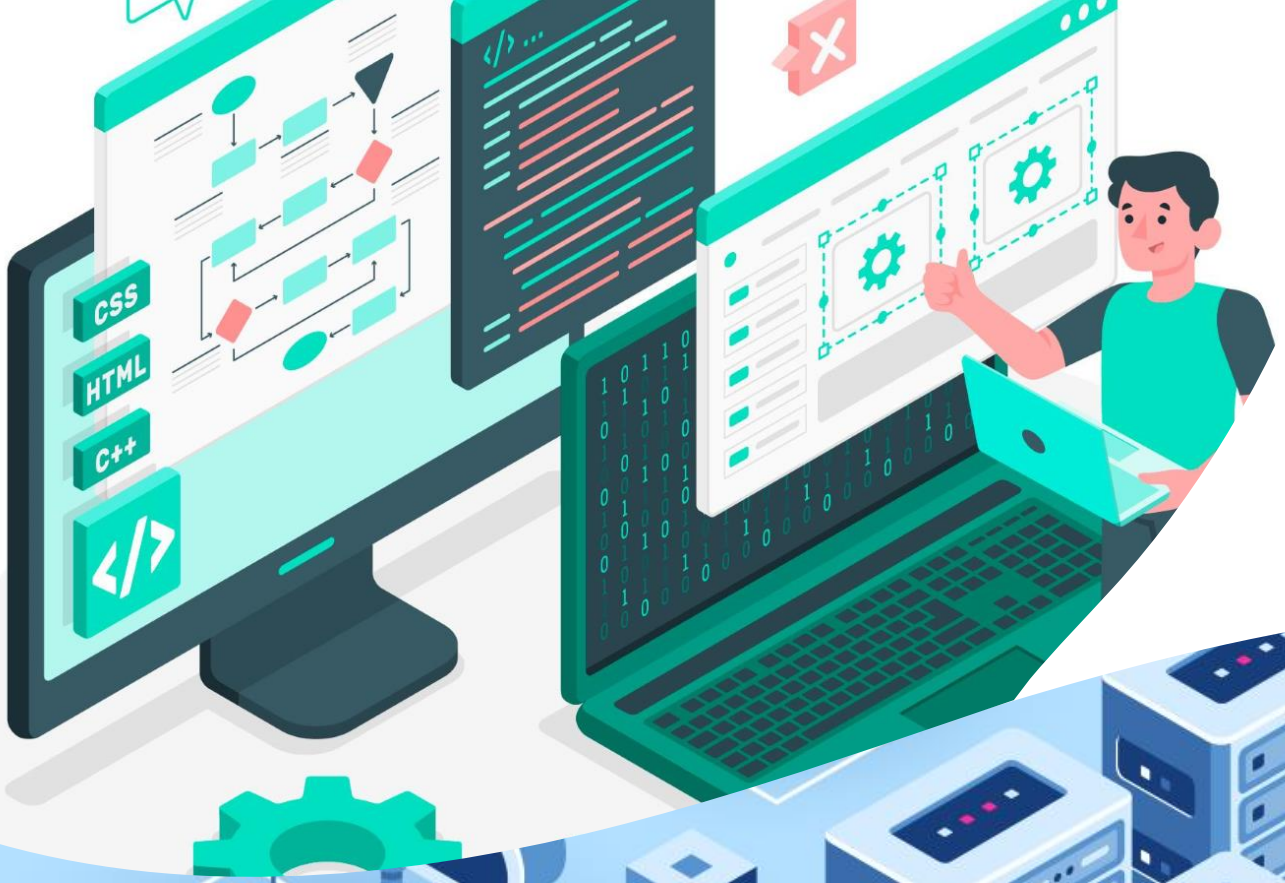


Lucas Schoemberger Sales

- Líder de Equipes / Especialista Fullstack em Angular e Java;
- Formado em Eng. Computação e Pós graduando em Big Data;
- Conhecimento em: Java, Javascript, Typescript, CSS, SASS, HTML, Angular, SQL, Spring Boot, Angular, Node.Js, Mongo, Git, AWS.

Thamyris da Silva Barbarino

- Especialista Fullstack Angular e Java/.Net
- Formada em Eng. Computação e Pós graduanda em Devops e Arquitetura;
- Conhecimento em: C#, Java, Springboot, Javascript, Typescript, CSS, HTML, .Net, .Net Core, Angular, SQL, PostgreSQL, AWS, CI/CD, Acessibilidade.



Na Web toda vantagem é temporária, para estar à frente você deve continuar a inovar.

Jakob Nielsen

O Desenvolvedor

Na mágica de construir aplicativos o desenvolvedor é a pessoa que transforma as ideias nos apps e sistemas. Normalmente os desenvolvedores são divididos em 3 tipos: Front-end, Back-end, Fullstack

O que é Back-end?

Codifica voltado à aplicações desktop, servidores, construindo APIs e serviços

O que é o Front-end?

Famoso construtor de telas, diferente do design que desenha as telas ele transforma o desenho em realidade.

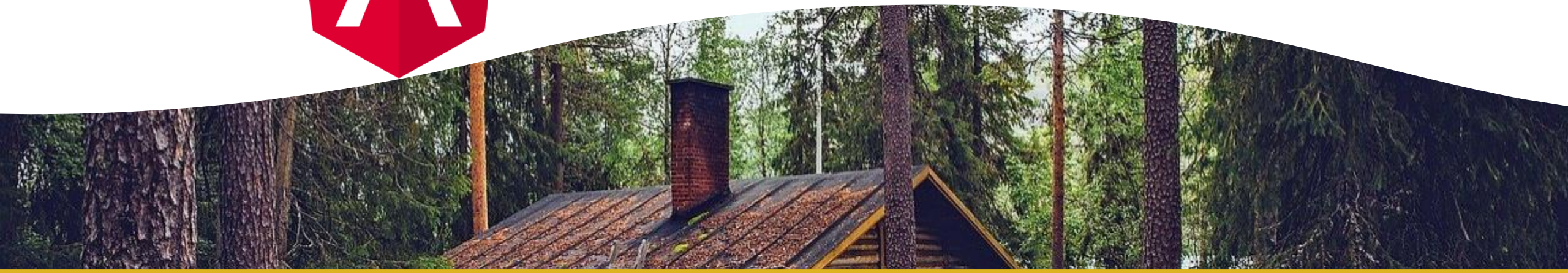
E o Fullstack?

O Mítico magico de Oz, para os lideres é quem consegue resolver qualquer coisa, na pratica nem tanto....

Angular

Você já ouviu falar em Angular?

É uma **plataforma front-end** voltada para a **construção** da interface de **aplicações web e mobile.**

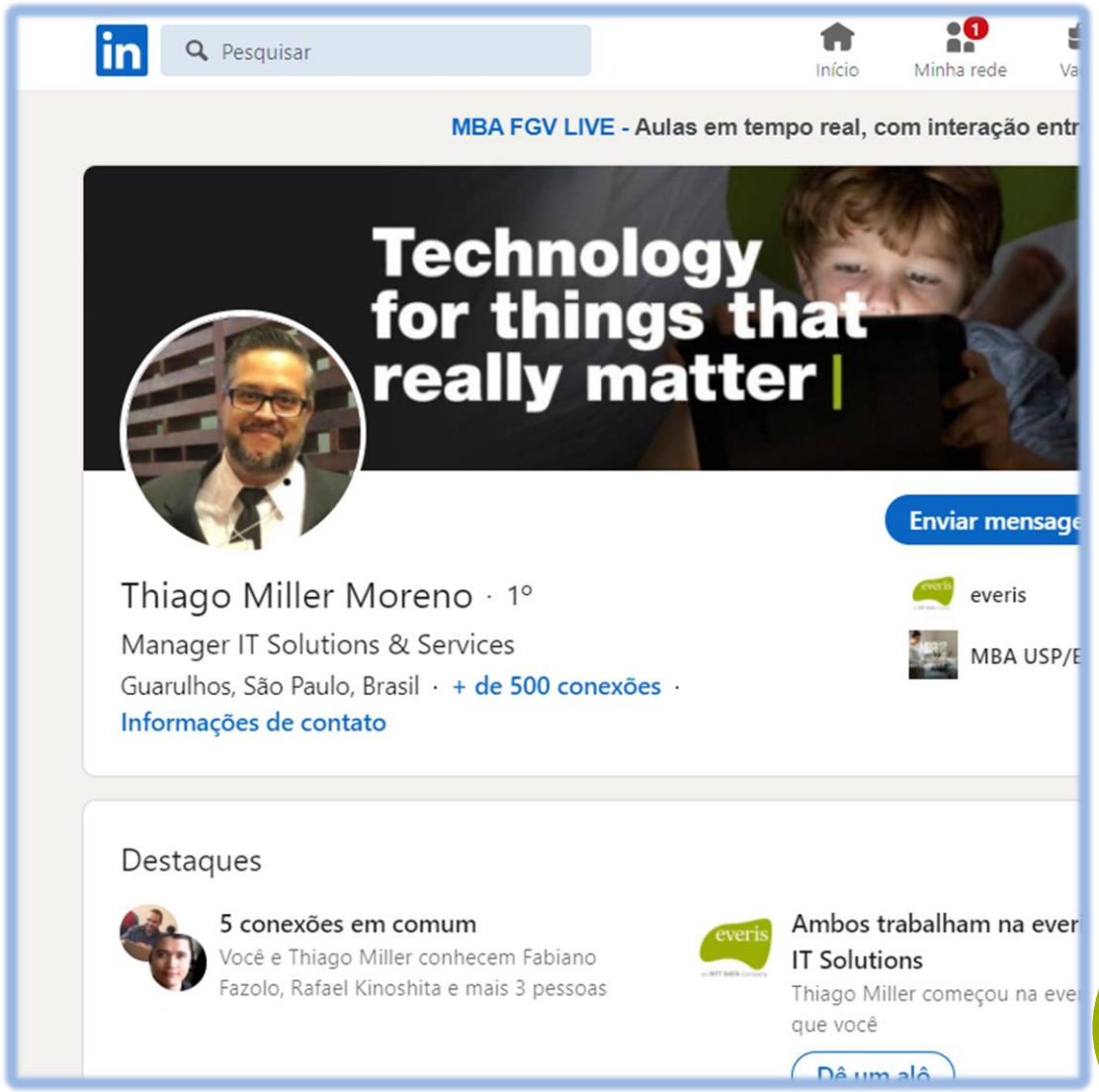


O Angular é mantido pelo **Google**

Utiliza **Typescript** no lugar do **javascript** para codificação.

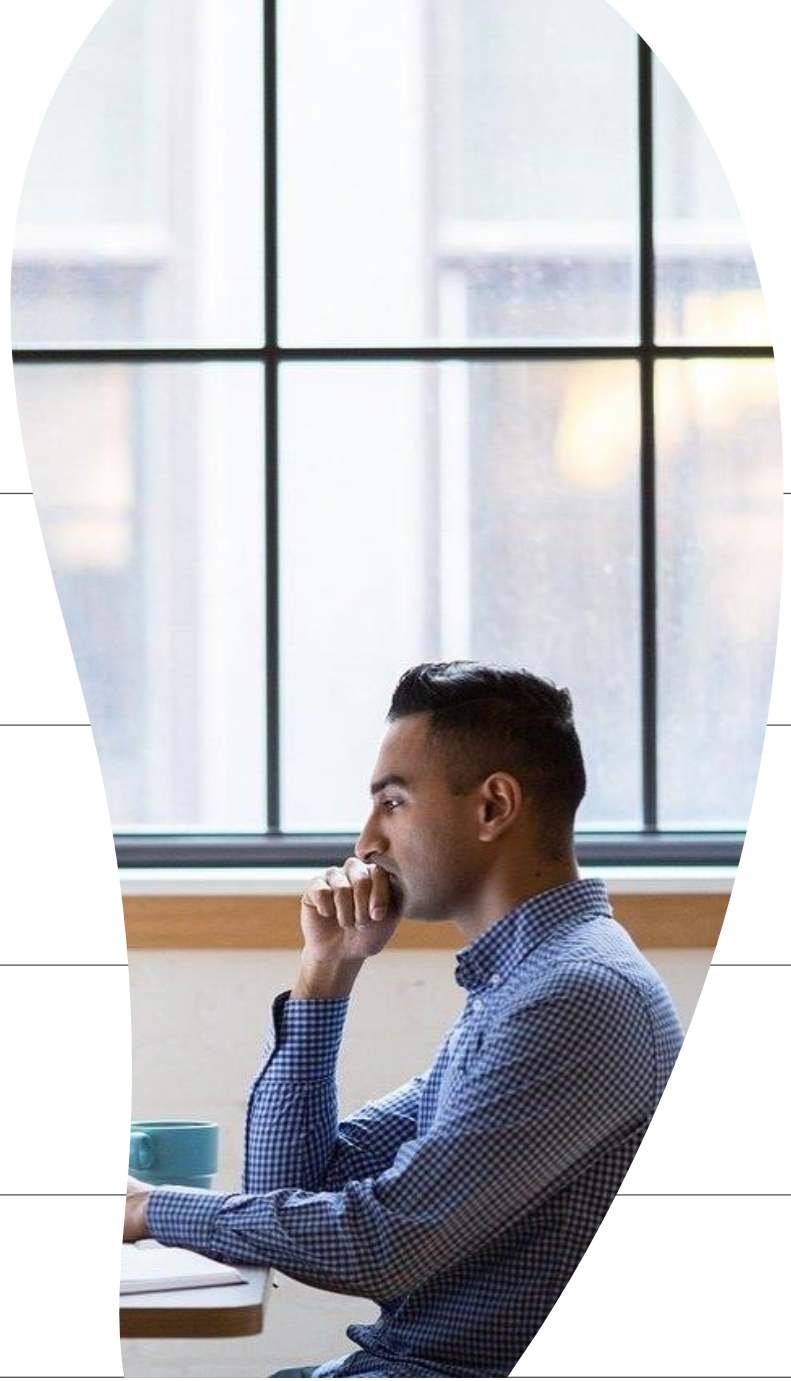
Angular

Exemplo de Front-end



Angular

Elementos do Angular



Typescript

Linguagem utilizada na codificação

Módulos

É a caixa onde se organiza os componentes, diretivas, pipes e serviços.

Componentes

Podem ser paginas inteiras ou partes reutilizáveis.

Templates

É a parte visual de um componente (HTML + CSS).

Roteamento

Responsável pela navegação das paginas e guarda das rotas

Serviços

Tem como objetivo principal organizar e compartilhar a lógica de negócios

Diretivas/Pipes

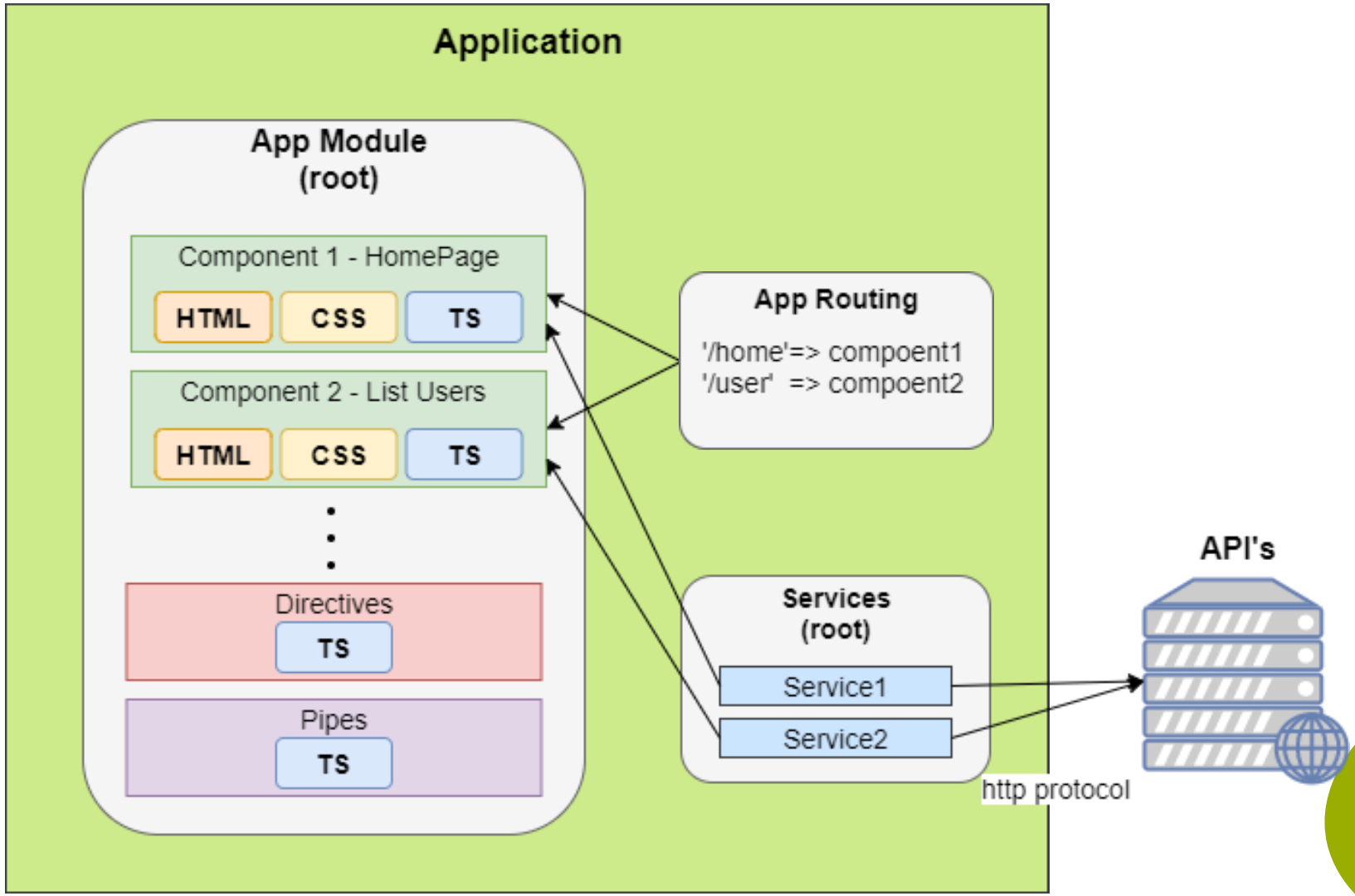
Transformações estruturais e de transformações de dados

Injeção de dependências

Responsável por disponibilizar serviços e outras classes aos componentes

Angular

Diagrama - Arquitetura simplificada



Angular CLI

Documentação: https://cli.angular.io/

Lista de comandos para criação de classes: **NG Generate**

ng generate module <module-name> → **ng g m** <module>

ng generate component <component-name> → **ng g c** <component-name>

ng generate class <class-name> → **ng g cl** <class-name>

ng generate directive <directive-name> → **ng g d** <directive-name>

ng generate pipe <pipe-name> → **ng g p** <pipe>

ng generate service <service-name> → **ng g s** <service-name>

Angular

Estrutura - Modulos

- O Angular é composto por pelo menos um módulo (**AppModule**).
- Os módulos podem ser utilizados para **agrupar componentes, diretivas, pipes e serviços**.
- O decorator **@NgModule()** identifica a classe como um módulo.

```
@NgModule({
  declarations: [
    AppComponent,
    CategoriaListComponent, CategoriaFormComponent,
    LancamentoListComponent, LancamentoFormComponent
  ],
  imports: [
    BrowserModule, BrowserAnimationsModule,
    AppRoutingModule, HttpClientModule
  ],
  exports: [],
  providers: []
})
export class AppModule { }
```

Angular

Estrutura - Componentes

```
@Component({
  selector: 'app-category-list',
  templateUrl: './category-list.component.html',
  styleUrls: ['./category-list.component.css']
})
export class CategoryListComponent implements OnInit {

  categories: Category[] = [];

  constructor(private categoryService: CategoryService) { }

  ngOnInit() {
    this.categoryService.getAll()
      .subscribe(
        categories => this.categories = categories,
        error => alert('Erro ao carregar a lista')
      )
  }
}
```

Os componentes são compostos por 3 classes:

- **TS**: **Concentra os dados e a lógica do aplicativo**
- **HTML**: Template/View parte visual do componente
- **CSS**: Estilização do template

Responsável por tornar **códigos reutilizáveis.**

Angular

Estrutura - Componentes

```
<table class="table table-hover">
  <thead>
    <tr class="bg-primary text-light">
      <th>Categoria</th>
      <th>Ações</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let category of categories">
      <td>
        <strong>{{category.name}}</strong><br>
        <small>{{category.description}}</small>
      </td>
      <td>
        <a [routerLink]="[category.id, 'edit']" class="btn btn-outline-info btn-sm mr-2">Editar</a>
        <button (click)="deleteCategory(category)" class="btn btn-outline-danger btn-sm">Excluir</button>
      </td>
    </tr>
  </tbody>
</table>
```

Angular

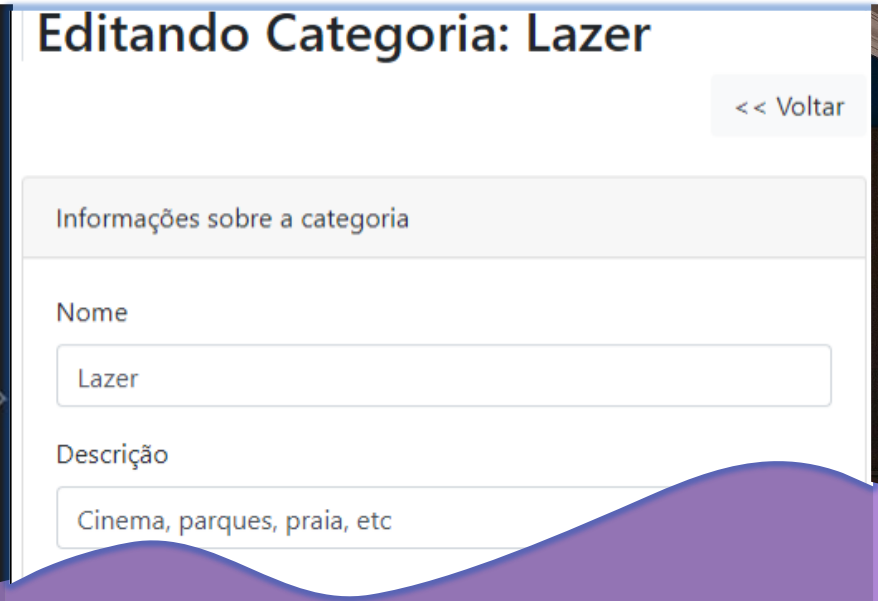
Estrutura - Componentes

Angular

Estrutura – Componente - Template

```
<div class="form-row">
  <div class="form-group col-md-4">
    <label for="name">Nome</label>
    <input type="text" id="name"
      [(ngModel)]="categoria.name">
  </div>

  <div class="form-group col-md-8">
    <label for="description">Descrição</label>
    <input type="text" id="description"
      [(ngModel)]="categoria.description">
  </div>
</div>
```



O template é documento **HTML**, que constitui a **parte visual** do componente.

Aqui no template é onde fazemos o binding das informações chamado de **Two-Way data** binding.

O template é o lugar par liberar sua criatividade, onde podemos controlar como o usuário vai visualizar na tela.

Angular

Estrutura – Roteamento

O arquivo de rotas é um **módulo**.

Responsável pela **navegação** do aplicativo.

Inclui esquema de **guarda de rotas**, implementação de **segurança**.

As rotas são definidas por 2 elementos principais:

1. **Path**: caminho da **URL**
2. **Estrutura** a ser **invocada**: **componente** ou módulo

```
const routes: Routes = [
  { path: 'categoria', component: CategoriaListComponent },
  { path: 'categoria/new', component: CategoriaFormComponent },
  { path: 'categoria/:id/edit', component: CategoriaFormComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

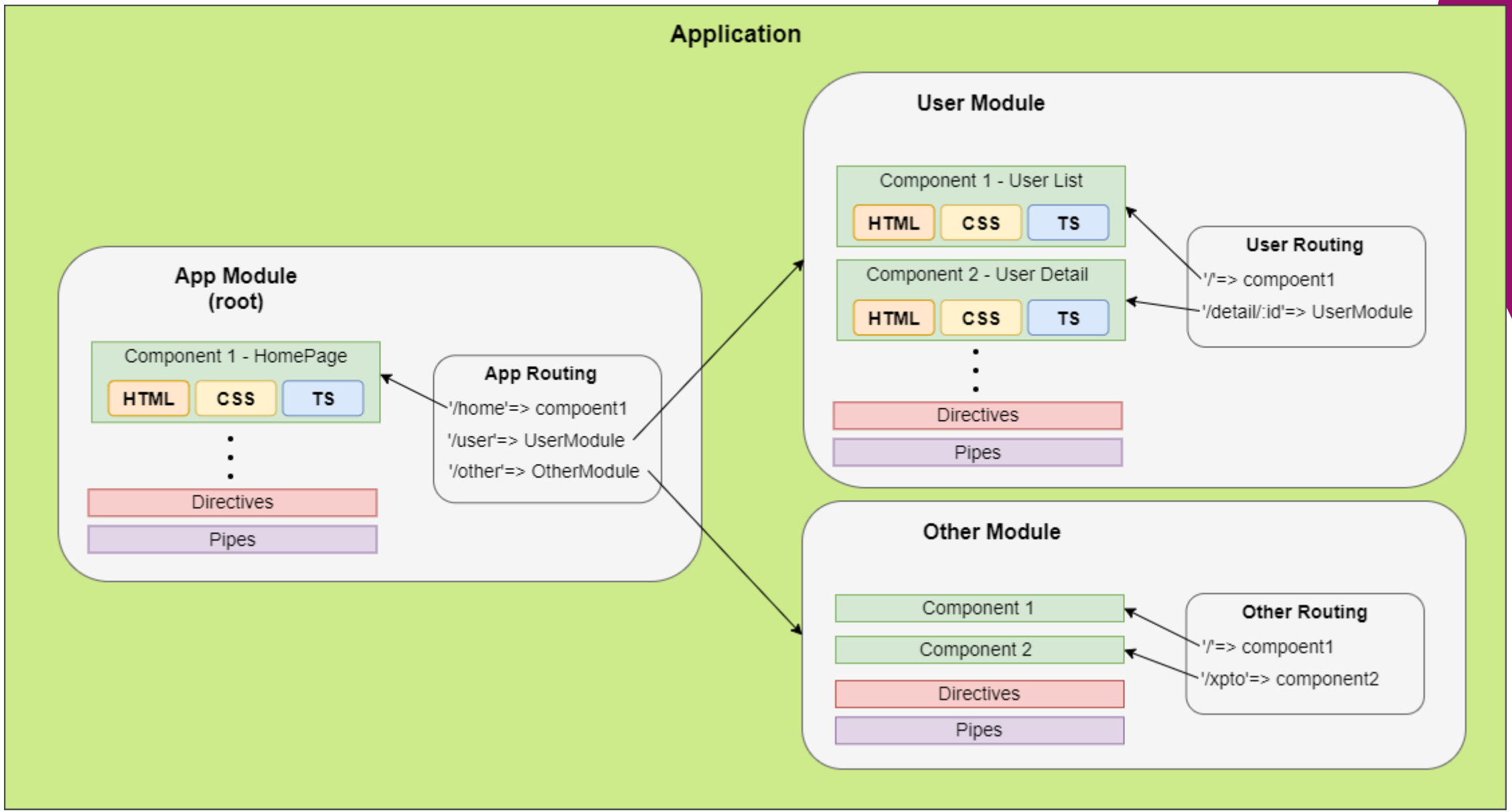
Angular

Estrutura – Roteamento

```
const routes: Routes = [
  { path: 'categoria', component: CategoriaListComponent },
  { path: 'categoria/new', component: CategoriaFormComponent },
  { path: 'categoria/:id/edit', component: CategoriaFormComponent }
];

@NgModule({
  declarations: [
    AppComponent,
    CategoriaListComponent,
    CategoriaFormComponent
  ],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    RouterModule.forChild(routes)
  ],
  providers: []
})
export class AppModule { }
```

Roteamento


Application
App Module
(root)
Component 1 - HomePage
HTML
CSS
TS
Directives
Pipes
App Routing
'/home'=> compoent1
'/user'=> UserModule
'/other'=> OtherModule
User Module
Component 1 - User List
HTML
CSS
TS
Component 2 - User Detail
HTML
CSS
TS
Directives
Pipes
User Routing
'/'=> compoent1
'/detail/:id'=> UserModule
Other Module
Component 1
Component 2
Directives
Pipes
Other Routing
'/'=> compoent1
'/xpto'=> component2

Angular

Estrutura – Roteamento

Uma vantagem de dividir a aplicação em módulos é a possibilidade de fazer o carregamento de determinados módulos somente quando houver necessidade. Mais conhecido como **Lazy-load**

```
const routes: Routes = [
  {
    path: 'entries',
    loadChildren: () => import('./pages/entries/entries.module').then(m => m.EntriesModule)
  },
  {
    path: 'categories',
    loadChildren: () => import('./pages/categories/categories.module').then(m => m.CategoriesModule)
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes, { relativeLinkResolution: 'legacy' })],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Angular

Estrutura – Service

```
@Injectable({
  providedIn: 'root'
})
export class CategoriaService {

  private apiPath: string = "api/categorias";

  constructor(private http: HttpClient) { }

  getAll(): Observable<Categoria[]> {
    return this.http.get(this.apiPath).pipe(
      catchError(this.handleError),
      map(this.jsonDataToCategorias)
    )
  }
```

Os serviços tem como objetivo **consumir, organizar e compartilhar regras de negócios, modelos/dados e métodos** dentro de um aplicativo Angular.

Para definir um serviço no Angular, utilizamos o decorador **@Injectable()**

Angular

Estrutura – Diretivas

As **diretivas** são como marcadores no elemento DOM que comunicam ao Angular para incluir um **comportamento específico**.

Existem três tipo de diretivas no Angular:

Diretivas de atributos: Alteram a aparência ou o comportamento de um elemento, componente ou outra diretiva, como por exemplo, **NgClass e NgStyle**.

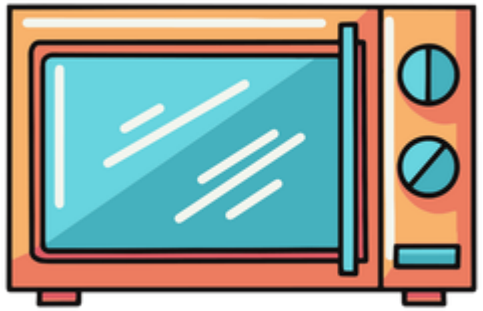
Diretivas estruturais: Modificam o layout adicionando ou removendo elementos do DOM, como por exemplo, **NgIf e NgFor**.

Componentes: São diretivas com um modelo.

```
<tbody>
  <tr *ngFor="let category of categories">
    <td>
      <strong>{{category.name}}</strong><br>
      <small>{{category.description}}</small>
    </td>
    <td>
      <a [routerLink]="[category.id, 'edit']" c
      <button (click)="deleteCategory(category)
    </td>
  </tr>
</tbody>
```

Angular

Estrutura – Pipes



Os pipes são responsáveis pela **transformação dos dados**.

Exemplo: data recuperada do banco de dados **1996-10-15T00:05:32.000Z**

```
<tbody>
  <tr *ngFor="let entry of entries">
    <td>
      <strong>{{entry.name}}</strong><br>
      <small class="text-success">{{entry.date | date: 'dd/MM/yyyy'}}</small><br>
      <small *ngIf="entry.description">{{entry.description}}</small>
    </td>
    <td>
      {{entry.category.name}}
    </td>
```

Por que aprender Angular?

Um dos maiores atrativos do Angular é a empresa por trás da ferramenta e as companhias que adotam.

Google

Com o Google por trás do Angular, é improvável que o Angular desapareça. Ele irá apenas evoluir. O Angular pode até levar mais tempo para se aprender, mas esse esforço extra vale a pena.

Companhias

É a ferramenta de front-end mais utilizada pelas grandes empresas, isso ocorre pelo fato da sua arquitetura ser bem definida e contar com uma documentação completa. Ex: **Itaú, Santander, PayPal, Rockstar Games, Microsoft Office**

Por que aprender Angular?

1. Os códigos são organizados em uma estrutura de simples entendimento e manutenção, possibilitando modularidade e quebra do código em pedaços.

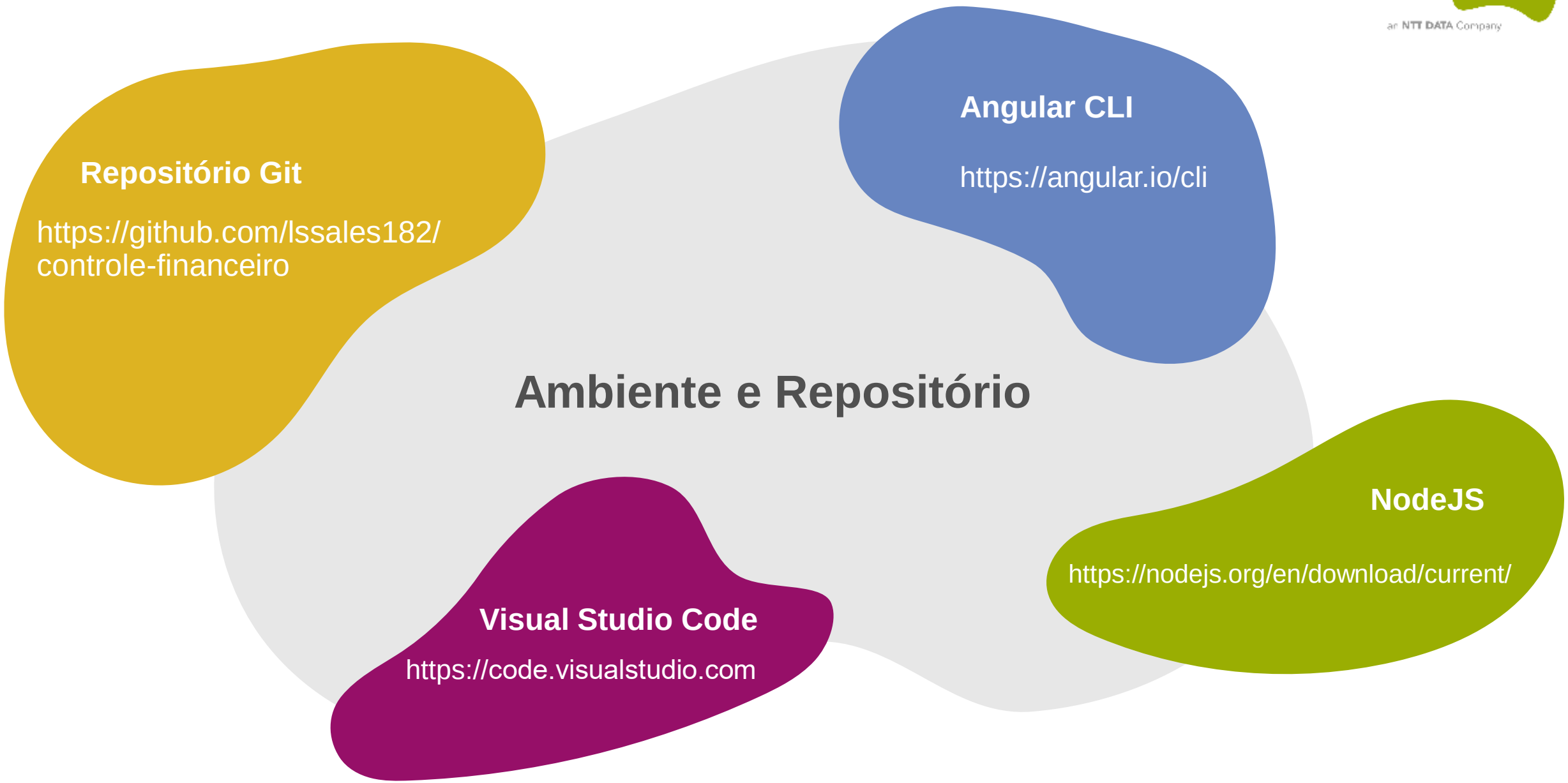
2. Outro ponto interessante é a atualização da página em tempo de desenvolvimento. Fazer alterações no código e observar as mudanças em tempo real.

3. O tempo que pode ser economizado também é um diferencial pois, não é necessário, utilizar editores de texto e criar códigos de configuração extensos dentro da aplicação.

4. Acessibilidade. Criar aplicativos acessíveis com componentes habilitados para ARIA, com o leitores de tela.

5. Ele é mantido por uma grande empresa como o google e utilizado por grandes empresas como: **Itaú, Santander, PayPal, Rockstar Games, Microsoft**

Demonstração



Mão na massa: Analisando a estrutura

Roteiro da componentização do header

1. Documentação: https://primefaces.org/primeng
2. Criar módulo de compartilhamento **“Shared”**
   a. Comando CLI: ng g m /shared
3. Criar o componente de **page header**
   a. Comando CLI: ng g component /shared/page-header
4. Verificar quais são as variáveis **“inputs do nosso componente”**
   a. Comando CLI: ng g component /pages/lancamentos/lancamento-form
5. Importar o módulo shared onde desejamos utilizar o componentes compartilhados
6. Aplicar o componente reutilizável usando o seu **seletor**

Acessibilidade

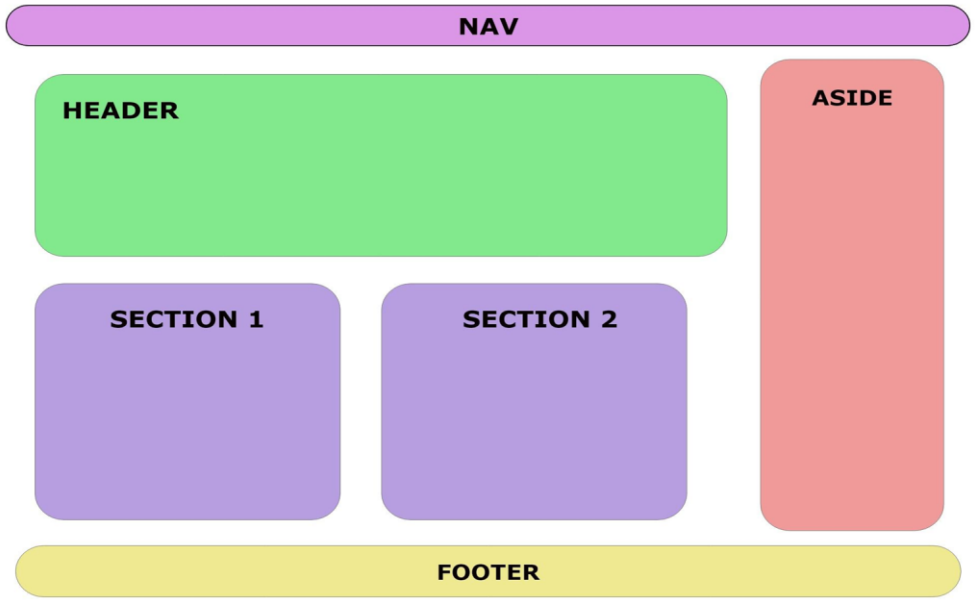
O que é acessibilidade?

A **acessibilidade** web **combina programação, design e tecnologia** para construir uma **Internet sem barreiras**. A indústria e a arquitetura, concebem objetos, veículos e espaços **adaptados às necessidades das pessoas** com mobilidade reduzida ou problemas cognitivos, visuais e auditivos, no desenvolvimento de sistemas esse **caminho** tambem deve **ser percorrido**.

Acessibilidade

Boas práticas para acessibilidade:

- Por que utilizar a semántica do HTML?
- O que impacta nos software de tecnologia assistiva?
- O que se deve fazer quando utiliza-se DIVs para ter acessibilidade?
- O que são diretivas de acessibilidade?



exemplo de utilização de componentes que são interpretados por leitor de tela

Curiosidades:

- Menos 1% dos sites e aplicativos são acessibilidade.
- Muitos sites não possuem inversão de cores.

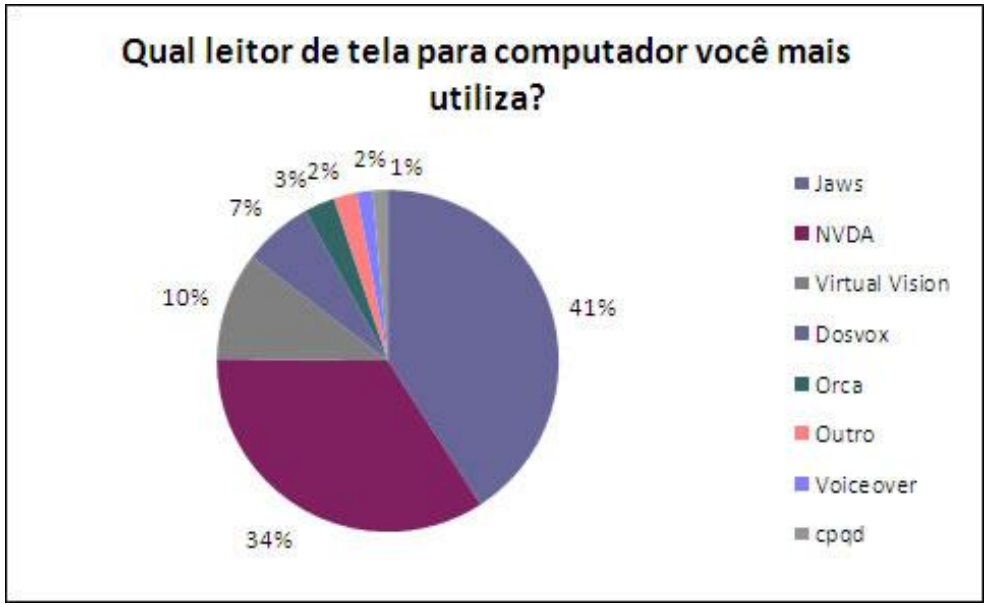
https://www.w3.org/WAI/standards-guidelines/aria/

https://guia-wcag.com

Softwares Leitor de Tela

Leitor	SO	Custos
Dosvox[5] [20]	Windows	Gratuito
Jaws [21]	Windows	Pago
NVDA [22]	Windows	Gratuito
Orca [23]	Linux	Gratuito
Virtual Vision [24]	Windows	Pago
VoiceOver [25]	macOS/iOS	Nativo nos ambientes Apple
TalkBack [26]	Android	Nativo nos ambientes Android

- Alguns exemplos de leitores de tela utilizados no Brasil.
- Jaws melhor software navegação mas precisa de licença.
- NDVA é o mais popular por ser gratuito e fácil instalação.



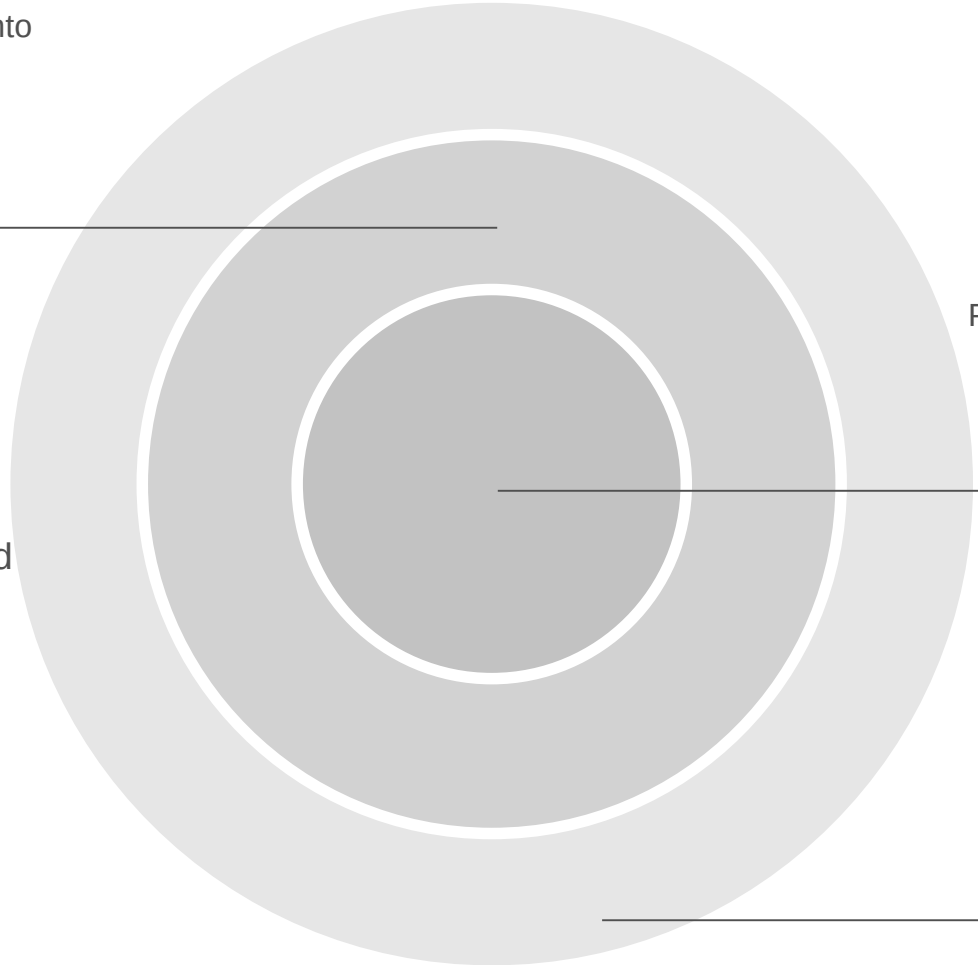
Mão na massa: Analisando a acessibilidade

- Aplicação de exemplo prático de acessibilidade.
- Utilização do NDVA para mostrar como é feita leitura

Resumo

O Angular

Uma **ferramenta/framework** de desenvolvimento front-end, tem uma estrutura mais rigida e bem definida, mais demorado para dominar, porém fornece varias facilidades.



O Desenvolvedor

Pode ser front, back, fullstack. Nosso foco foi no front-end, sendo ele responsável por transformar a idéia do design em uma tela interativa para o usuário

No geral para aprender angular e front-end de uma forma geral, o caminho é grande como visto, porém ferramentas como o angular auxiliam muito para criar aplicações dinâmicas em menos tempo.

A Acessibilidade

Uma combinação de diretrizes que ajudam pessoas com deficiência a ter mesma condições de aprendizado e navegação plena na web.

everis

an NTT DATA Company

OBRIGADA.