# Documentação front-end

## Criação de dashboard nova

Sempre que em algum cliente novo que tenha dashboard, seguir os seguintes passos:

**1 -** levantar quais telas e features este cliente precisará;

**2 -** passar a equipe de marketing para ver se existe layout;

**2.1 -** se não existir, pegar o psd e criar um novo componente com este layout, seguindo os padrões citados a seguir, subir este layout no repositório [Dash-base](https://bitbucket.org/trackage/dash-base) e usar na dashboard;

**2.2 -** usar layouts dos componentes presentes na [Dash-base](https://bitbucket.org/trackage/dash-base).

**3 -** criar um repositório no bitbucket com o proprietário sendo a equipe Trackage no projeto “site”, com o nome no padrão citado a seguir;

**4 -** subir para o azure;

**5 -** com dns com o padrão <nomeDoCliente>.trackage.com.br

## Utilização de componentes

Para cada tela da dashboard, existe um componente, com isso a modularização da dashboard é totalmente possível. Segue, o passo a passo para utilizar dos componentes já criados. Lembrando que para criar uma tela nova, precisa seguir o tópico [Criação de dashboard nova](#_rdrlexg39sqq).

Temos no repositório [Dash-base](https://bitbucket.org/trackage/dash-base), todos componentes e serviços, pré-elaborados, fizemos isso para facilitar ainda mais o processo de criação de dashboards e apresentação de dashboards.

Usando o angular cli, no terminal preferido, siga ao diretório “trackage” no seu computador e dê o comando para criar um novo projeto:

**ng new dash-<nomeDoCliente>** // Usando camelCase em Porugês, por padrão para

// melhor separação dos clientes

Para criar as telas use o comando:

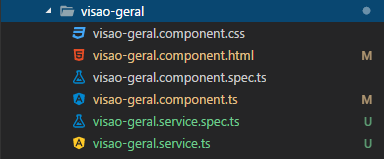
**ng g c <nomeDaTela>**  // Usando camelCase em Português, por padrão para

// melhor separação das telas

Para a parte de informações da tela, crie serviços, você consegue achar tutoriais explicativos de sua utilização nos tutoriais da [Loiane Groner](https://loiane.training/course/angular-2/), para sua criação utilize:

**ng g s <nomeDaTela>/<nomeDaTela>** // Use o <nomeDaTela> para criar dentro do diretório daquela tela e use o mesmo nome da tela para o nome do serviço.

Assim, ficará com os seguintes arquivos:



**visao-geral.component.css** // estilo do componente

**visao-geral.component.html**  // estrutura do componente

**visao-geral.component.ts** // código typescript

**visao-geral.component.spec.ts**  // código js transpilado do typescript

**visao-geral.services.ts** // código typescript do serviço

**visao-geral.service.spec.ts** // código js do serviço transpilado do typescript

Destes, você fará alterações em:

* **visao-geral.component.css:** Pegará o css do layout que está na Dash-base. (ctrl+c/ctrl+v)
* **visao-geral.component.html:** Pegará a estrutura do layout que está no Dash-base. (ctrl+c/ctrl+v)
* **visao-geral.component.ts:** Chamará o serviço e fará o binding de informação no html por aqui.
* **visao-geral.component.spec.ts:** Não mexe em nada, o typescript gera o código dele sozinho.
* **visao-geral.services.ts:** Fará o serviço que chama o serviço do back-end.
* **visao-geral.service.spec.ts:** Não mexe em nada, o typescript gera o código dele sozinho.

## Exemplo de códigos

### Utilização de Rotas

* crie um arquivo chamado **“app.routes.ts”** no diretório “app”
* importe-o no modulo **“app.module.ts”**
* **import { routing } from ‘./app.routes’**
* **imports: [ routing ]**

### Estrutura de serviços

* ao gerar um serviço;
* importe:

**import { HttpClient, HttpHeaders } from ‘@angular/common/http’;**

**import { Injectable } from ‘@angular/core’;**

* No Construtor insira:

**constructor(private http: HttpClient) { }**

* e use o código de exemplo:

**pegaDados(): any {**

**const headers = new HttpHeader();**

**this.http.post(‘<urlDoServiço>’, {}, {headers: headers})**

**.subscribe(**

**res => {**

**console.log(res);**

**},**

**err => {**

**console.log(err);**

**}**

**);**

**}**

### Chamando serviços

- ao fazer o serviço;

- importe-o como provider no **“app.module”**:

**providers: [ <nomeDaServiço>Service ]**

* importe-o no **“component.ts”**:

**import { <NomeDoServiço>Service } from ‘./<nomeDaTela>.service’;**

* insira no construtor:

**constructor(private <nomeDoServiço>Service: <NomeDoServiço>Service) { }**

* no **ngOnInit()**

**ngOnInit() {**

**this.dados = this.<nomeDoServiço>Service.pegaDados();**