TP3

Verificação de conceitos

Antes de começarmos a criar a aplicação do cliente, vamos checar se os conceitos estão alinhados. Responda os questionamentos abaixo.

1. **Seu cliente andou estudando mais um pouco e esbarrou com a teoria de componentes web, assim ele pediu para que você fizesse uma breve explanação sobre as 4 tecnologias de componentes web e se o framework que você escolheu, o AngularJS, contempla essas tecnologias.**

2. **Aproveite a oportunidade e explique o padrão do Emissor de Eventos pro seu cliente.**

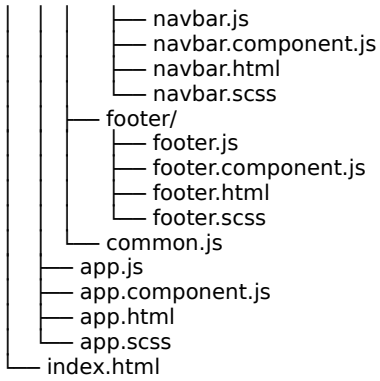
Desenvolvimento de aplicativo

Chegou a hora de você realizar uma refatoração no aplicativo do seu cliente, ou seja, vamos convertê-lo para a tecnologia de componentes. Lembre-se que você deverá implantar uma árvore de componentes.

```
├── app (componente raiz)
│   ├── 1+ componente stateful, presente em todos cenários do app (navbar, banner, rodapé)
│   ├── <div ui-view></div>
│   │   ├── home (componente stateful)
│   │   │   └── 1+ componentes stateless
│   │   ├── categoria (componente stateful)
│   │   │   └── 1+ componentes stateless
│   │   ├── tarefa (componente stateful)
│   │   │   └── 1+ componentes stateless
```

Em termos de árvore de diretórios, colocaremos nossos componentes stateful na pasta *components* e os stateless na pasta *common*. Repare que os serviços normalmente ficam próximos aos componentes que os gerenciam, por exemplo: um cenário gerenciador de notícias, onde você pode criar, remover e editar notícias terá o arquivo *noticia.service.js*. Mas lembre-se após o registro de um serviço, ele pode ser injetado em qualquer lugar da aplicação, através de seu nome. Eis um exemplo de árvore de diretórios:

```
├── app/
│   ├── components/ (para componentes stateful)
│   │   ├── home/
│   │   │   ├── home.js
│   │   │   ├── home.component.js
│   │   │   ├── home.service.js
│   │   │   ├── home.html
│   │   │   ├── home.scss
│   │   ├── generico
│   │   │   ├── generico.js
│   │   │   ├── generico.component.js
│   │   │   ├── generico.service.js
│   │   │   ├── generico.html
│   │   │   ├── generico.scss
│   │   └── components.js
│   ├── common/ (para componentes stateless: não costumam ter serviços)
│   │   ├── navbar/
```



Nesse TP, iremos utilizar toda a lógica criada no TP anterior. Se você recebeu algum feedback negativo, aproveite para melhorar seu trabalho. Além de converter os elementos já criados em componentes, devemos implementar um serviço para cada tipo de dado utilizado. Portanto precisamos de um serviço para controlar os dados de categoria e outro para os dados de tarefa. Um serviço controlador de dados, normalmente, entrega as operações de CRUD (create, retrieve, update, delete): criar, recuperar, atualizar e deletar. Em nosso app, iremos transferir os dados para este serviço, haja vista que ele será seu controlador.

Ainda nesse TP, iremos usar o modelo de projeto ng6-starter. Primeiro instale o git e o nodejs+npm. Em seguida execute *git clone https://github.com/e-menezes/NG6-starter.git*, esse comando criará a pasta NG6-starter. Entre na pasta (via linha de comando), execute *npm install* e aguarde. Após instaladas as dependências execute npm start para levantar o ambiente de desenvolvimento, que pode ser acessado em http://localhost:3000/.

O trabalho deve ser submetido em um arquivo zipado com o nome *nome_do_aluno.tp3.zip*, ***o aluno deve enviar apenas a pasta client*** do modelo de projeto dentro do zip, não mande as demais pastas e arquivos.

Bom trabalho.