

Projeto Avaliativo 2 - Solar Energy

Peso: 45% da nota do módulo 1

DEVinHouse

Sumário

1. Introdução	1
2. Requisitos da Aplicação	1
3. Exemplo de aplicação	2
4. Entrega	2
5. Critérios de Avaliação	3
6. Plano de Projeto	7

1. Introdução

Você está prestes a entrar para o time de Desenvolvedores da **DevInHouse Solar Energy**. Para concretizar a sua contratação, você deverá resolver um desafio utilizando Angular. O time de recrutamento necessita que você crie uma aplicação para o gerenciamento de energia, chamada **Solar Energy**.

2. Requisitos da Aplicação

A aplicação que deverá ser realizada individualmente, deve contemplar os seguintes requisitos:

- Uma **Tela de Login** contendo um formulário com campos de **email e senha**. Os campos de email e senha são obrigatórios (Utilize o ngModel para vincular os inputs do formulário). Ao clicar no **botão de Entrar** e **passar pela validação**, redirecionar para tela de Dashboard. *Seguir layout conforme o mockup disponibilizado.*
- Um menu lateral, contendo as opções **Dashboard, Unidade Consumidora e Cadastro de energia gerada**. O menu deve ser configurado usando **Angular Router**. *Seguir layout conforme o mockup disponibilizado.*
- Uma tela de Dashboard contendo 4 cards: Total de unidades (Exibir o total de unidades cadastradas no json-server) , unidades ativas (Exibir total de unidades com status ativo(true) , unidades inativas (Exibir total de unidades com status inativo(false) e média de energia (Soma de todos os total / total de unidades).
- A tela de Dashboard deve conter um gráfico de linha exibindo os meses do ano. O gráfico deverá exibir o total de energia gerado por mês, sendo assim deve-se realizar a

soma por mês do total gerado de todas as unidades e exibir no gráfico o total de cada mês, o eixo x deve-se exibir os últimos 12 meses e no eixo y o total por mês. Usar a lib <https://github.com/valor-software/ng2-charts>. Seguir layout conforme o mockup disponibilizado. Obs: Se caso não conseguir implementar o gráfico a tempo, pode substituir por uma listagem de itens referente a cada contador do dashboard.

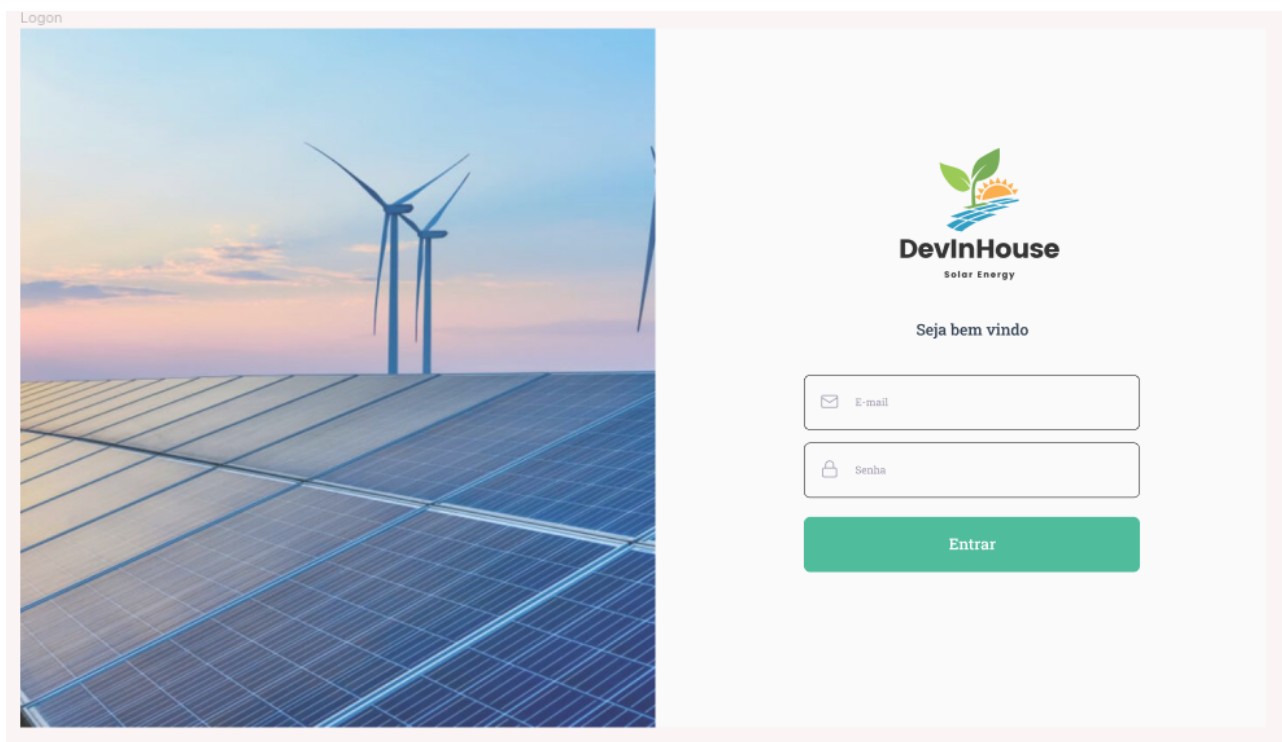
- Uma tela de **Listagem de unidades** (*consumir rota /unidades do json-server*) contendo uma tabela com as colunas ID, apelido, Local, Marca e modelo, além dos botões **Editar** e **Remover**. Além disso, a tela deve conter um **botão Nova unidade** (Ao clicar, enviar usuário para tela de cadastro de unidade). *Seguir layout conforme o mockup disponibilizado.*
- Uma tela de cadastro de unidade, contendo um formulário com os campos apelido, Local, Marca, modelo e status (**Checkbox**). Ao clicar no botão salvar, chamar evento de clique (click) e cadastrar unidade via POST na rota /unidades do json-server. Todos os campos do formulário são obrigatórios. *Seguir layout conforme o mockup disponibilizado.*
- Implementar botão de remover unidade na tela de **Listagem de unidades**. Ao clicar no botão remover, chamar evento de clique (click) e remover unidade clicada via DELETE na rota /unidades/:id do json-server.
- Implementar **botão de editar unidade**, ao clicar no botão de editar, enviar usuário para tela de edição de unidade. Ao renderizar a tela, trazer os campos preenchidos com as informações da unidade clicada, ao clicar em salvar, salvar os valores via PUT na rota /unidades/:id do json-server. *Seguir layout conforme o mockup disponibilizado.*
- Implementar **tela de Lançamento de geração mensal** contendo um formulário com um `<Select/>` (listando como opção as unidades já cadastradas consumindo do json-serve e listando com o *ngFor*), um campo de data e um campo de total kw gerado (aceita somente números). Ao clicar em salvar, chamar evento de clique (click) e cadastre valores via POST na rota /gerações do json-server.

3. Exemplo de aplicação

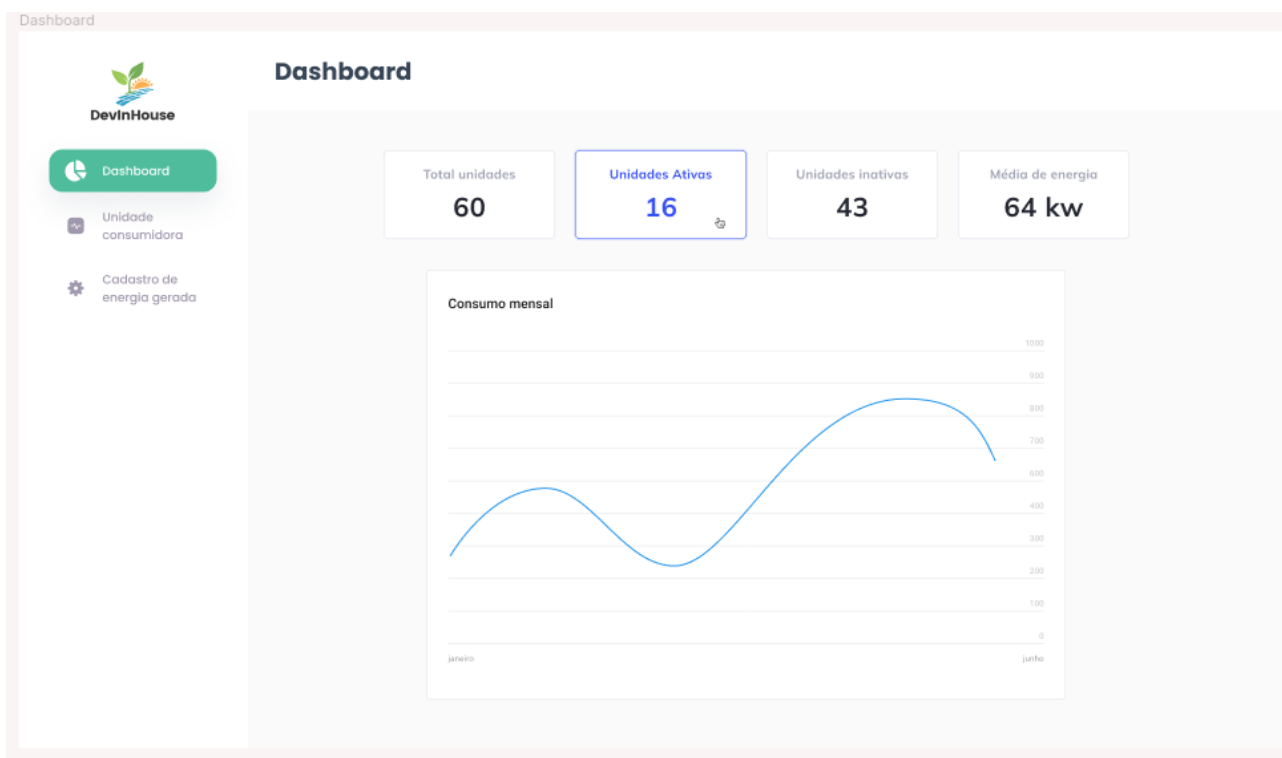
A aplicação deverá conter os requisitos apresentados anteriormente, sendo codificada utilizando o framework Angular e seguindo o modelo no Figma.

- <https://www.figma.com/file/Bh4fJZohrlLMoKQU6lbe9p/Projeto-DevInHouse---NDD?node-id=0%3A1>

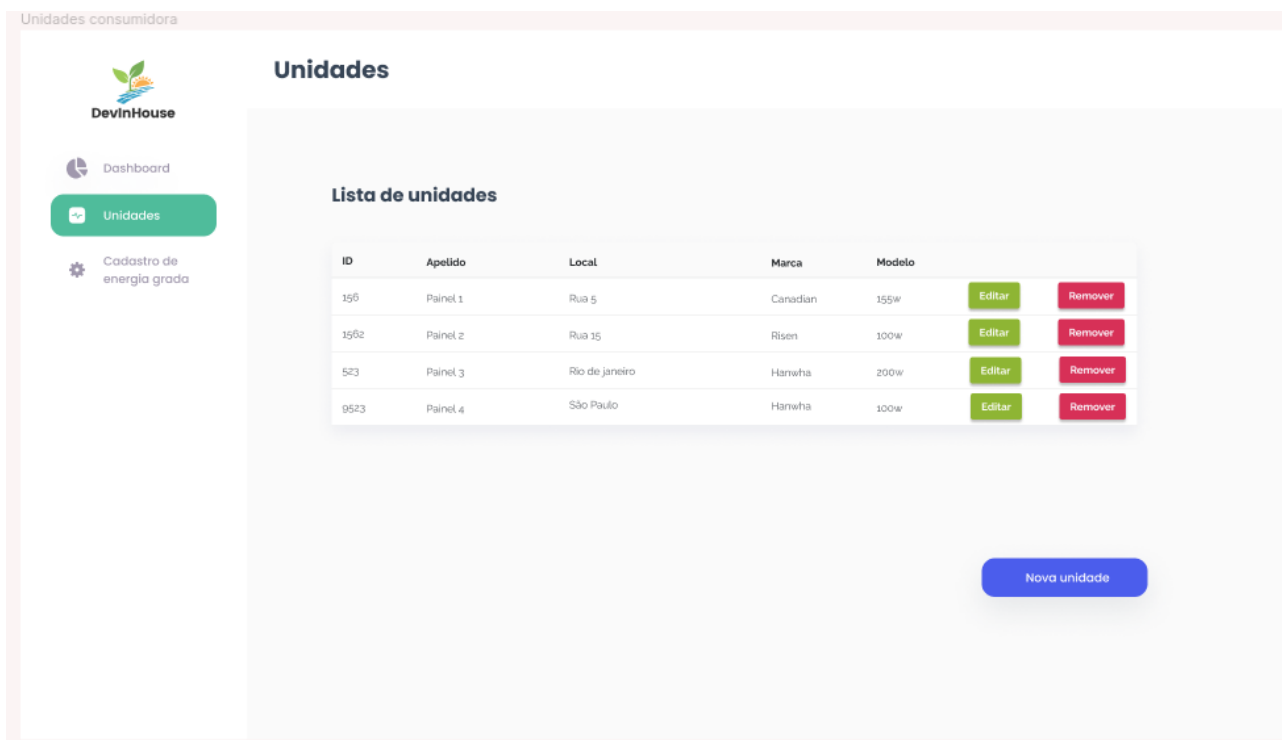
As imagens a seguir apresentam um exemplo de como a aplicação pode ser construída.



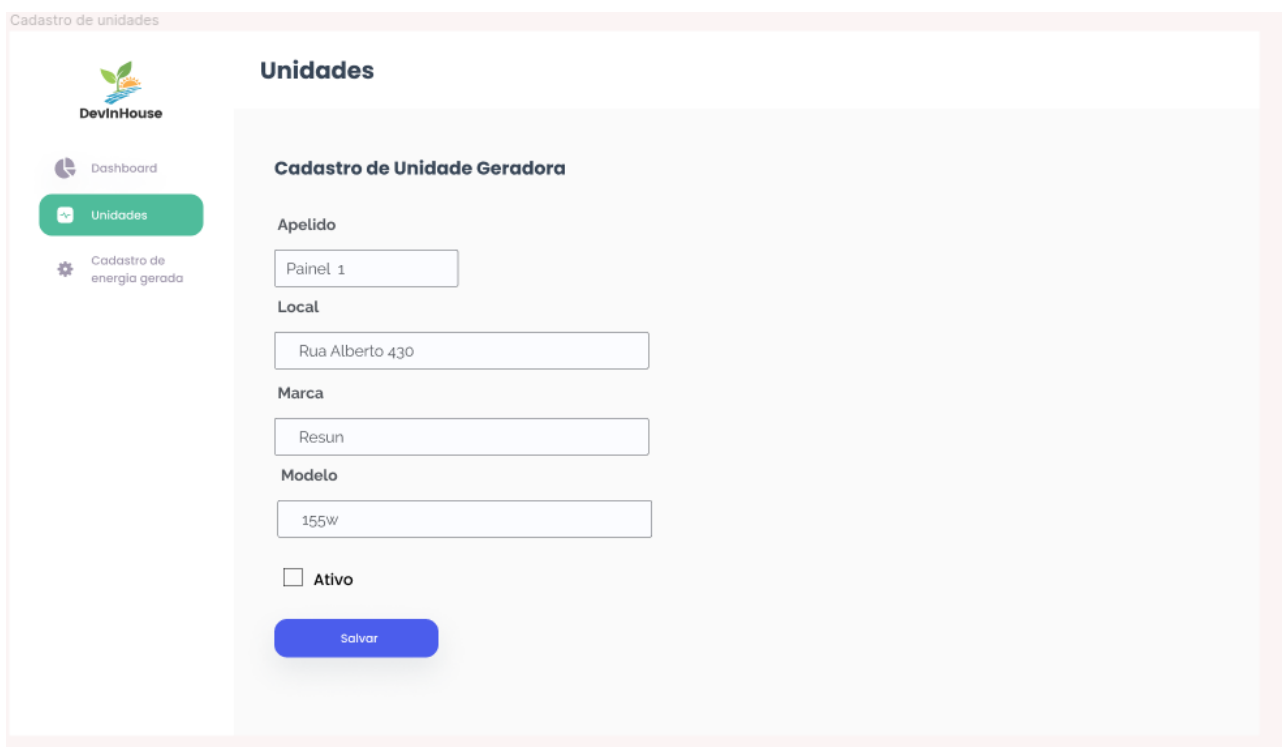
Legenda: Tela de Login



Legenda: Tela de Dashboard





Legenda: Tela de listagem de unidades





Legenda: Tela de cadastro de unidades

Consumo mensal

**DevinHouse**

 Dashboard

 Unidades

 Cadastro de energia gerada

Lançamento de geração mensal

Unidade geradora:

Unidade 1

Mês/ano

02/2022

Total kw gerado

80

Cadastrar

Legenda: Tela de lançamento de geração mensal

4. Entrega

O código desenvolvido deverá ser submetido no GitHub, e o link deverá ser disponibilizado na tarefa **Módulo 1 - Projeto Avaliativo 2**, presente na semana 12 do AVA até o dia **29/05/2022** às 23h55.

O repositório deverá ser **privado** e deverá adicionar as seguintes pessoas no repositório:

- fernando.puntel@edu.sc.senai.br *fepuntel*
- cesar.abascal@edu.sc.senai.br
- kelvisbcc@gmail.com *kelvisdev*

Importante:

- Será considerado como data de entrega a **última atualização** no repositório do projeto no GitHub. Lembre-se de não modificar o código até receber sua nota.
- Você **DEVE** entregar o link do seu repositório no AVA.

5. Critérios de Avaliação

A tabela abaixo apresenta os critérios que serão avaliados durante a correção do projeto. O mesmo possui variação de nota de 0 (zero) a 10 (dez) como nota mínima e máxima, e possui peso de 50% sobre a avaliação do módulo 1.

Serão desconsiderados e atribuída a nota 0 (zero) os projetos que apresentarem plágio de soluções encontradas na internet ou de outros colegas. Lembre-se: Você está livre para utilizar outras soluções como base, mas **não é permitida** a cópia.

Nº	Critério de Avaliação	0	1	1,5	2
1	O aluno desenvolveu a página de Dashboard? (peso 2)	O aluno não desenvolveu a página de Dashboard.	O aluno inseriu os cards e o gráfico(ou listagem de itens), porém os valores dos elementos estavam estáticos (Não usou chamadas para API para a montagem dos dados).	O aluno inseriu os cards utilizando as informações dinâmicas da API, porém o gráfico (ou listagem) estava com os dados estáticos (Não usou chamadas para API para a montagem dos dados).	O aluno inseriu os cards e o gráfico (ou listagem) usado informações da API (Usou chamadas para API para a montagem dos dados).
Nº	Critério de Avaliação	0	0,5	0,75	1
2	O aluno desenvolveu uma tela de Login funcional e conforme o mockup? (peso 1)	O aluno não conseguiu desenvolver a página de login.	O aluno conseguiu apresentar o formulário de login, mas não implementou o NgModel para vincular os inputs.	O aluno conseguiu desenvolver o formulário de login com todos os inputs com NgModel, porém não desenvolveu o evento de click (click) no botão.	O aluno conseguiu desenvolver todos os elementos da página e elementos como NgModel e evento de clique dos inputs e no botão e o evento de (click).
3	O aluno conseguiu desenvolver o componente de menu da aplicação ? (peso 1)	O aluno não conseguiu desenvolver o componente de menu da aplicação e nem o roteamento com Angular Router.	O aluno conseguiu desenvolver o componente de menu, porém não configurou o roteamento da aplicação usando o Angular Router.	O aluno conseguiu desenvolver o componente de menu e o roteamento da aplicação com Angular Router, mas não importou o módulo e componente de roteamento no lugar correto da aplicação.	O aluno conseguiu desenvolver o componente de menu e o roteamento da aplicação com Angular Router, além de importar o módulo e componente de roteamento no lugar correto da aplicação.

4	O aluno desenvolveu a tela de listagem de unidades? (peso 1)	O aluno não desenvolveu uma tela de listagem de unidades.	O aluno inseriu a tabela de listagem de unidades, porém não buscou os dados usando a rota /unidades.	O aluno inseriu a tabela de listagem de unidades e buscou os dados na rota /unidades, porém não inseriu o botão de editar e remover conforme o mockup.	O aluno inseriu a tabela de listagem de unidades, buscou os dados da rota /unidades, inseriu os botões editar e remover em cada linha da tabela.
5	O aluno desenvolveu a tela de cadastro de unidades ? (peso 1)	O aluno não desenvolveu uma a tela de cadastro de unidades.	O aluno inseriu os campos, porém não desenvolveu o evento de (click) e o vínculo dos inputs com NgModel.	O aluno inseriu o campos de input e desenvolveu e vinculou os inputs ao NgModel, porém não implementou o evento de clique (click) (Cadastrar unidade via POST no json-server).	O aluno inseriu todos os campos corretamente e cadastrou os valores na rota /unidades via POST dentro do evento de clique (click).
6	O aluno desenvolveu a tela de lançamento de geração mensal ? (peso 1)	O aluno não desenvolveu a tela de lançamento de geração mensal ?	O aluno inseriu os campos de input, porém não desenvolveu o NgModel para cada input.	O aluno inseriu o campos de input e desenvolveu o NgModel de cada input, porém não implementou o evento de clique (click) para (Cadastrar unidade via POST)	O aluno desenvolveu o formulário completo e cadastrou via POST na rota /geracoes.
7	O aluno desenvolveu uma página que apresenta um design agradável e intuitivo? (peso 1)	O aluno desenvolveu uma página que não apresenta um design agradável e intuitivo	O aluno desenvolveu uma página e inseriu alguns estilos, como tipo/tamanho/cor de fonte, largura/altura/margem de alguns	O aluno desenvolveu uma página e inseriu alguns estilos, bem organizados no seu próprio arquivo,	Além de o aluno conseguir apresentar estilos textuais e de posicionamento básico, num arquivo separado do HTML, também estilizou

			elementos, mas todos inseridos diretamente no HTML.	separado do HTML.	o fundo da página, os botões, a lista e economizou texto de botão quando não havia muito espaço.
8	O aluno desenvolveu um código Typescript que está bem organizado e é facilmente legível, conforme as boas práticas propostas pelos grandes nomes do desenvolvimento de software? (peso 1)	Aluno não conseguiu inserir código Typescript na página.	O código TS desenvolvido pelo aluno está bagunçado, com nome do arquivo, nomes de variáveis e funções não explicativas, sem indentação correta e não está organizado em pastas.	O código TS desenvolvido pelo aluno está separado em cada pasta e arquivo diferente, mas ainda apresenta algum problema de organização/legibilidade (indentação incorreta, nomes de variáveis/funções não explicativos)	O código TS desenvolvido pelo aluno está separado em um arquivo diferente e pastas explicativas, bem organizado, com nomes de arquivo, funções e variáveis explicativos e indentação correta.
Nº	Critério de Avaliação	0	0,15	0,25	0,5
9	O aluno desenvolveu o botão de remover da tela de listagem de unidades (peso 0,5)	O aluno não desenvolveu o botão de remover da tela listagem de unidades.	O aluno inseriu o botão de remover com o evento de click (click), porém não removeu o item do servidor.	O aluno desenvolveu o botão de remover com o evento de click (click) e removeu a unidade clicada do servidor, mas a mudança não refletiu no estado da tabela atual.	O aluno desenvolveu o botão de remover com o evento de click (click) e removeu a unidade clicada do servidor e a mudança refletiu no estado da tabela atual.

10	O aluno desenvolveu o botão de editar na tela de listagem (peso 0,5)	O aluno não desenvolveu o botão de editar da tela listagem de unidades.	O aluno inseriu o botão de editar com o evento de clique (click), porém não redirecionou para o formulário de edição.	O aluno desenvolveu o botão de editar com o evento de clique (click) e redirecionou para o formulário de edição, além de carregar os dados dentro dos inputs do formulário. Porém não implementou a chamada API via PUT para a rota /unidades.	O aluno desenvolveu a edição completa, redirecionando o usuário para o formulário de edição e persistindo a edição no servidor json-server.
----	--	---	---	--	---

6. Plano de Projeto

Ao construir a aplicação **DevInHouse Solar Energy** contendo as páginas de Login, Dashboard, Listagem de unidade, cadastro de unidade e cadastro de geração mensal, o aluno estará colocando em prática os aprendizados em:

- **HTML:** principais tags como head, title, body, div, h1, form, input, button, ul, li, p, nav. Atributos de tags como class, id, type.
- **CSS:** estilizar a página, os botões, inputs, alterar atributos dos elementos da tela de acordo com a interação do usuário para uma melhor experiência do usuário (UX), Alinhamento de elementos com flex-box. Obs: Fique a vontade para utilizar o pré-processador SASS (SCSS) para aumentar produtividade com sua estilização.
- **Typescript:** utilizar recursos do TS, inferindo tipos, interfaces, classes, constantes, propriedades, métodos, arrays, funções .map(), .find(), .findIndex(), .filter(), .reducer().
- **Angular:** utilização dos recursos como: angular router, httpclient, services, components, angular cli, manipulação de dados usando data binding (interpolação, property binding, event binding), diretivas de atributo(ngClass / ngStyle) e de estrutura(ngFor / ngIf), manipulação de dados entre componentes usando input(property) e output(event).
- **Bootstrap:** utilizar o bootstrap para construção da interface do projeto. Não é um item obrigatório, mas recomendamos fortemente a utilização do framework.