**# Back-End**

**Controllers**

Definição de Controller: - DevController   
Um controller possui 5 funções; index, show, store, update e detroy.

\* Index: para quando queremos mostrar uma lista de dev por exemplo;

\* Show: para quando queremos mostrar um unico dev por exemplo;

\* Store: para quando queremos criar um novo cadastro;

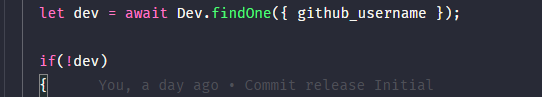
\* Update: para quando queremos alterar;

\* Destroy: para quando queremos deletar;

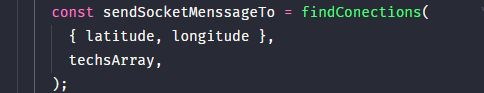
Nesse trecho de código está construindo uma validação antes de cadastrar um novo Dev:

Vai no BD, realiza uma busca com o “username” especificado no parâmetro, trazendo o primeiro que achar, “ou não”.

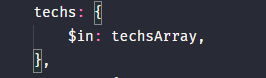
Abaixo ele valida se já existe o usuário no banco.



Nesse trecho de código está filtrando todas as conexões que estão no máximo 10 km de distância e que o Dev possua pelo menos uma das techs definida na busca.



Definição de $in:  
 É um operations do mongodb, trada a seguinte ocasião (Se o usuario possui as technologias dentro de);



pode-se pesquisar quais são os outros dentro da documentação do mesmo.

Definição de $near:  
 Também é um operations do mongodb, trata a seguinte ocasião (Consigo encontrar objetos perto de uma localização).

Fazer com que o “express” trabalho com JSON, durante as requisições e etc.



Definição básica do “Listen == ouvir”:

Ao declararmos o “listen” dizemos que ele irá ouvir alguma coisa,   
nessa abordagem abaixo dizemos que ele irá ouvir a porta definida por um “.env” ou a própria “3333”.



Ao utilizarmos a função “on()” estamos dizendo que ele irá ouvir um evento ou algo, segue basicamente a mesma definição do listen.



Ex: Ouça o evento ‘connection’, e toda vez que houver uma connection, faça alguma coisa.

Definição da função Some();

Ele tem como retorno um valor boleano(true,false), no código abaixo utilizamos essa função para para retornar se existe ou não as techs buscadas pelo usuário.



**# Front-Mobile**

**# Front-Web**

Os tres conceitos do React:

**1#Componente**

\* O componente nesse início já estamos utilizando

que é o nosso app.

\* Ele basicamente é uma function que retorna um

conteúdo html, css ou até mesmo js.

\* O conceito de criar um componente novo parte de

quando estamos repetindo um trecho de código muitas

vezes tipo 3 vezes ou mais, ou também quando conseguimos

insolar um pedaço da nossa aplicação. Ex: um cabeçalho, uma timeline

\*RESUMINDO:

O Componente é um Bloco isolado de HTML, CSS e JS o qual não interfere

no restante da aplicação

**2# Propriedade**

\* Sempre que usamos o HTMl usamos a propriedade

\* Um exemplo de propriedade é atribuir por exemplao nosso Header a propriedade ' title="Dashboard" '

RESUMINDO:  
 Propriedade nada mais é do que Informações um componente PAI passa para um componente FILHO

**3#Estado**

\* é basicamente uma informação que um componente vai manipular

\* O React não fica monitorando as alterações de variáveis, por isso ao implantar o contador

na funcionou, para funcionar de forma correta devemos utilizar esse conceito de estado.

Utilizamos esse conceito importando a lib 'useState';

Ex. de utilização: ' const counte = useState(0); ' , o valor do parâmetro e o valor inicial

\* O retorno do useState é um vetor que dentro dele contem: a 'var count' e uma funcao setCounte,

que usamos para atualizar o valor do proprio 'counter'

\* O comportamento desse 'setCounter' segue o conceito de imutabilidade, que nesse caso quer dizer

que nao é atribuído um valor para a 'var counter', a cada atualização o 'setCounter' cria um

novo counter com o novo valor, sempre dentro do react será assim, "criará um novo counter".

RESUMINDO:

São informações mantidas pelo componente (Lembrar: imutabilidade)

**# UseEfect()**

Ele serve para dispararmos uma função toda vez que uma informação for alterada, ou também apenas uma unica vez. Ele recebe 2 parametros, o primeiro falamos para ele qual vai ser a função que ele irá executar, e o segundo dizemos QUANDO ele ira executar.  
No segundo parametro dexaremos um vetor vazio, pois se ele estiver vazio ela irá executar apena uma unica vez.