1. instalar o auto import
2. ng new [nome do projeto]

Explicar component

3. ng generate component [nome do componente] --spec=false
4. **(personagem.component.html)** criar a div com os atributos desejados
    a. nome, sexo, ano nascimento
5. **(personagem.component.ts)** checar o seletor do component
6. **(app.component.html)** Apagar tudo menos a última linha.
7. **(app.component.html)** colocar o seletor verificado anteriormente
8. ng serve para testar a aplicação **(1)**
9. **(personagem.component.ts)** crie um personagem fictício
10. **explicar** interpolação
    a. é uma meneira de exibir dados do ts para a pagina html
11. **(personagem.component.html)** insira as informações criadas no passo anterior
    a. usando interpolação
12. Teste a aplicação

explicar diretivas

13. **(personagem.component.html)** mude a variavel criada anteriormente para uma lista de any
    a. iniciar com vários valores
    b. mude o nome para personagens
14. **(personagem.component.html)** adicione a diretiva ngFor para iterar sobre a lista criada

explicar modulo

15. parar o ng serve
16. ng generate module pesquisa
17. mover a pasta personagem para dentro da pasta pesquisa
18. **(app.module.ts)** retirar PersonagemComponent, declarations

19. **(app.module.ts)** adicionar PesquisaModule, imports
20. **(pesquisa.module.ts)** colocar PersonagemComponent, declarations
21. **(pesquisa.module.ts)** adicionar PersonagemComponent, exports
22. Testar se continua funcionando
23. parar ng serve
24. ng generate component pesquisa/pesquisa --spec=false
25. **(pesquisa.module.ts)** verificar se o component está declarado no módulo pesquisa e exporta-lo
26. **(pesquisa.component.html)** criar uma div com input e um botão
    a. botão Pesquisar
27. **(app.component.html)** trocar a tag app-personagem por app-pesquisa

Explicar databind

28. fazendo o property bind
29. **(pesquisa.component.html)** adicionar [value] ao input e mostrar o undefined
    a. <input … [value]=”campo”>
30. **(pesquisa.component.ts)** criar a variável nomeada na tag <input[value]> já com um valor, mostrar que iniciou com o valor
31. fazendo o event bind
32. **(pesquisa.component.html)** adicionar (keyup) ao input passando o evento
    a. <input … (keyup)=”onPesquisar($event)”>
33. **(pesquisa.component.ts)** criar o método do keyup dando um console.log no valor recebido
    a. onPesquisar(evento :any)
34. **(browser)** mostrar o target.value, ressaltando o tipo para poder mostrar o cast
35. **(pesquisa.component.ts)** realizar o cast para pegar o valor do input e atribuí-lo à variável
    a. (<HTMLInputElement> evento.target).value

36. **(pesquisa.component.html)** exibir a variável com interpolação e mostrar o efeito no browser
37. fazendo two way data bind
38. **(pesquisa.component.html)** trocar <input [value] (keyup)> por <input [(ngModel)]=”campo”>
39. **mostrar** que continua fazendo a mesma coisa de forma mais simples
40. **(pesquisa.component.html)** adicionar event binding ao botão <button (click)=”onPesquisar()>
41. **(pesquisa.component.ts)** criar o método do event click exibindo o valor do twdb no console
42. **(pesquisa.module.ts)** importe os modelos de formulário
    a. imports : [… , FormsModule], ….

Explicar serviços e injeção de dependências

43. parar o ng service
44. ng generate service pesquisa/pesquisa --spec=false
45. **(pesquisa.service.ts)** fazer a injeção de dependência no construtor
    a. constructor (private httpClient : HttpClient)
46. **verificar** se a importação do http ocorreu corretamente:
    a. @angular/common/http
47. **(pesquisa.service.ts)** criar método pesquisa que recebe o nome e retorna a chamada do http.get
    a. pesquisa(nome :string)
        i. deve concatenar a url + search + o parâmetro nome
            1. “http://swapi.co/api/” + “/people?search=” + nome
        ii. passar a string url literal para a pasta environment
            1. (src/enviroment/enviroment e enviroment.prod) api: “url”
            2. enviroment.api + “/people?search=” + nome
48. **(pesquisa.service.ts)** criar um método get que recebe uma url e retorna um http.get
    a. get(url : string) return this.httpClient.get(url)

49. **(pesquisa.component.ts)** injetar o serviço PesquisaService criado no passo 44
    a. constructor (private pesquisaService : PesquisaService)
50. **(pesquisa.component.ts)** crie a variavel resultado
    a. resultado : any
51. **(pesquisa.component.ts)** no método onPesquisar
    a. atribua a resultado a chamada ao método pesquisa do serviço criado
        i. onPesquisar(){

```
this.resultado = this.pesquisaService.pesquisa(this.campo);

console.log(this.resultado);

}
```

52. ng serve
53. **verificar** o erro gerado
54. **(pesquisa.module.ts)** importar o HttpClientModule e prover o serviço criado no passo 44
    a. ... , prividers: [PesquisaService]
55. **verificar** o erro corrigido
56. **(pesquisa.component.ts)** mostrar que o resultado retornado inicialmente é Observable

57. **(pesquisa.component.ts)** executar a promisse para exibir o resultado no console
    i. onPesquisar(){

```
this.pesquisaService.pesquisa(this.campo).subscribe({

resposta =>{

this.resultado = resposta;

console.log(this.resultado);

}

});

}
```

58. **(browser)** mostrar as consultas feitas a api
    a. Network
59. **(browser)** mostrar onde os results estão
60. **(pesquisa.component.ts)** altere o tipo da variavel resultado para uma lista de any
    a. resultado : any[]
61. **(pesquisa.component.ts)** alterar o método onPesquisar para atribuir o valor correto a resultado
    a. … this.resultado = resposta['results']
62. **(pesquisa.component.html)** exibir a variável resultado em uma div com a diretiva ngFor para iterar sobre a lista obtida da api
    a. Exibir
        i. item.name
        ii. item.gender
        iii. item.birth_year

63. **(pesquisa.component.ts)** adicionar as variáveis anterior e próximo para controlar a paginação, assim como seus respectivos métodos
    a. anterior : any
    b. paginaAnterior()
    c. proxima : any
    d. paginaProxima()
64. **(pesquisa.component.ts)** na promise do metodo onPesquisar atribua os valores corretos a anterior e proximo
    a. anterior = resposta['previews'];
    b. proxima = resposta['next'];
65. **(pesquisa.component.html)** adicionar uma div com 2 botões para navegar na paginação, desabilitando-os caso necessário
    a. para desabilitalos insira a diretiva
        i. <button [disabled]="anterior==null">
        ii. <button [disabled]="proxima==null">
    b. Para colocar ações nos botões
        i. faça um event bind (click)="paginaAnterior()"
        ii. implemente os metodos, ex:
            1. paginaAnterior(){this.pesquisaService.get().subs…
               //copie a logica usada no onPesquisar}
66. **(browser)** verifique a funcionalidade adicionada
67. **(personagem.component.html)** retirar a diretiva ngFor da div
68. **(personagem.component.ts)** apagar o código feito anteriormente, criar a variável personagem: any
69. **(personagem.component.ts)** usar a tag @Input(), na variável personagem, para informar que essa variável pode ser atribuida atravez do seletor
    a. @Input()
       personagem : any[]
70. **(pesquisa.component.html)** dentro do ngFor substituir as interpolações por um seletor do componente personagem, passando o atributo personagem como um item, usando um property binding

71. **mostrar** os resultados

Templates

72. **(pesquisa.component.html)** adicione o ngIF a div que envolve os botões verificando se o resultado é null ou se é uma lista vazia
    a. ngIf=”resultado != null && resultado.length != 0”
73. **(pesquisa.component.html)** adicione a condição ;else #semresultado
    a. … resultado.length != 0 ; else #retornonull”
74. **(pesquisa.component.html)** crie o template #semresultado
    a. <ng-template #retornonull> Sem resultado </ng-template>
75. **mostrar** os resultados

Explicar o que foi feito, que foi usar um componente dentro de outro. Porém a forma ideal seria usar um terceiro componente para gerenciar os outros componentes

Roteamento

76. pare o ng serve
77. ng generate component erro --spec=false
78. ng generate component home --spec=false
79. **(erro.component.html)** adicione uma mensagem personalizada para erros
80. **(home.component.html)** adicione uma mensagem personalizada de boas vindas
81. **(app.component.html)** deixe apenas a ultima linha, adicione uma barra de navegação e crie os links usando o router-link
    a. <nav>

        <a [routerLink]="['/inicio']" routerLinkActive="active" >Inicio |

```
            <a [routerLink]="['/pesquisa-personagem']" routerLinkActive="active">
                                                    Pesquisar Personagem</a>
            </div>
    </nav>
```

82. **(app-routing.module.ts)** adicione as rotas à constante route
    a. {path: 'home', component: 'HomeComponent'} //para a rota dos personagens //para rota do home
    b. {path: 'pesquisa-personagem', component: 'PesquisarComponent'} //para a rota dos personagens
    c. {path: '', ,redirectTo: '/inicio', pathMatch: 'full'} // para rotas vazias
    d. {path: '**', component:ErroComponent} // para rotas que não estão listadas
83. teste as rotas criadas

Bootstrap

84. parar ng serve
85. npm install --save bootstrap
86. **(angular.json)** adicionar, ao styles, a linha "node_modules/bootstrap/dist/css/bootstrap.min.css"
87. **ng serve** para testar que alguns estilos já foram alterados

fazendo o deploy do app

88. parar o ng serve
89. ng build (para build de desenvolvimento)
    a. verificar o tamanho da pasta dist e seu conteúdo
90. ng build --prod (para build de produção)
    a. verificar o tamanho da pasta dist e seu conteúdo
91. npm install -g angular-http-server

92. cd dist/angularwars
93. angular-http-server
94. **(browser)** localhost:8080/index