Desafio 9

Com base no curso de “Angular 11 Avançado: Criando uma Arquitetura Master/Detail” responda as seguintes questões.

a) Qual a responsabilidade do package.json no projeto?

Ele é responsável por descrever o projeto, gerenciar os pacotes e versões dentro desse diretório. Ele mostra as modificações feitas quando adicionamos algo ao projeto com o comando npm.

b) Qual a responsabilidade do angular.json no projeto?

O angular.json tem como responsabilidade definir as propriedades genéricas na aplicação como folhas de estilos ,scripts e inicializar a aplicação main.ts.

c) Qual a finalidade da tag ?

A funcionalidade da tag é a de interpretar as rotas passada e direcionar para o componente correto.

d) Qual a diferença entre as duas declarações de rotas a seguir? Qual delas é denominada de eager-load e lazy-load?

Declaração 1:

```
const routes: Routes = [
  { path: 'entries/new', component: EntryFormComponent },
];
```

Declaração 2:

```
const routes: Routes = [
  { path: 'entries', loadChildren: () => import('./pages/entries/entries.
module').then(m => m.EntriesModule) },
];
```

O lazy-load gera um carregamento mais rápido da página por inicializar apenas alguns componentes e após isso carrega os demais quando solicitado. Já o eager-load é o inverso, ele carrega todo de uma única vez, tornando o carregamento mais lento.
Na declaração nº1 é um exemplo do eager-load, isto é ele carrega todo o componente de uma única vez. E a declaração nº2 é uma lazy-load.

e) Qual a importância da componentização?

A componentização é importante para a reutilização do código, facilitar a manutenção e evitar códigos repetidos. Isto é, se no código possuir vários componentes com a mesma função como por exemplo um formulário, este pode ser "componentizado" e utilizado em vários locais, evitando assim, a repetição deste código em vários locais.

f) Quais componentes reaproveitáveis foram criados no projeto?

BaseResourceFormComponent;
BaseResourceListComponent;
BreadCrumbComponent;
FormFieldErrorComponent.

g) No arquivo bread-crumb.component.ts qual funcionalidade da linha a seguir:

```
@Input() items: Array<BreadCrumbItem> = [];
```

As informações contidas no BreadCrumbItem serão passadas em forma de array.

h) Com base no exercício anterior para qual finalidade a anotação a seguir é utilizada?

```
@Output()
```

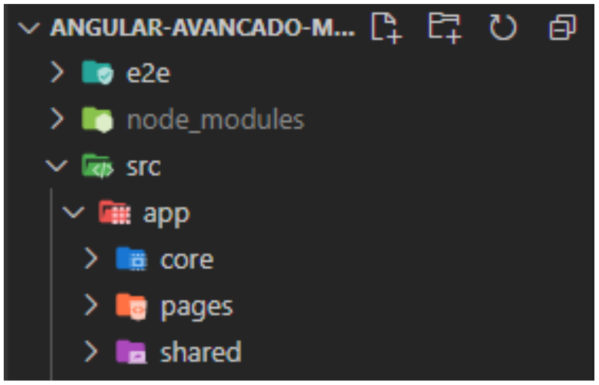
O @Output() compartilha as informações da classe pai para as demais classes filhas.

i) Sobre a arquitetura proposta no curso, descreva os requisitos para se enquadrar em cada um dos níveis a seguir: core, pages e shared

A função da pasta CORE é a de armazenar e disponibilizar os arquivos para a aplicação como um todo.

A pasta PAGES é o local onde são alocados os componentes que formam a aplicação em si.

A pasta SHARED é o local onde são alocados os componentes que podem ser compartilhados com os outros.



j) Qual o papel do arquivo in-memory-database.ts?

Simula um back-end sem necessidade de comunicação com servidor externo. Neste arquivo foi simulado um banco de dados. São colocados objetos que são retornados através de uma requisição 'get' por exemplo.

k) Liste 5 métodos e 5 propriedades de um objeto do tipo FormGroup e suas respectivas funcionalidades. Segue exemplo abaixo de declaração.

```
this.categoriaForm = this.formBuilder.group({
   id: [null],
   name: [null, [Validators.required, Validators.minLength(2)]],
   description: [null]
});
```

Métodos:

setValue – é usado para atualizar apenas um subconjunto dos elementos do FormGroup ou FormArray . Ele apenas atualizará os objetos correspondentes e ignora o resto.
pathValue – Troca qualquer propriedade definida no objeto.
addControl – Especifica se esta instancia de FormGroup deve emitir eventos após a adição de um novo controle.
removeControl – Especifica se esta instancia de FormGroup deve emitir eventos após a remoção de um novo controle.
reset – Limpa os valores informados pelo usuário.

Propriedades:

touched – Boleando, é setado inicialmente como false , tornando se true quando o controle é encostado.
validator – É uma função que o formulário pode chamar para decidir se um determinado campo de formulário é válido ou não. Retorna verdadeiro se o campo do formulário é válido de acordo com as regras do validators, ou falso, caso contrário.
valid – Usado para indicar se um campo está válido.
invalid – Usado para indicar se um campo está invalido ou que não localizou no banco de dados.
disabled – Os controles dentro do grupo são desabilitados individual.

l) Explique o funcionamento da seguinte sentença:

```
if (this.currentAction == "edit") {
  this.route.paramMap.pipe(
   switchMap(params => this.lancamentoService.getById(+params.get("id")))
  )
   .subscribe(
    (lancamento) => {
     this.lancamento = lancamento;
     this.lancamentoForm.patchValue(lancamento);
    },
    (error) => alert('Ocorreu um erro no servidor, tente mais tarde.')
   )
}
```

Nessa sentença ele verifica se a ação será do tipo EDIT, caso seja do tipo edit irá pegar o parâmetro id enviar para o lancamentoService e ao abrir a tela irá trazer todas as informações daquele lançamento em específico.
Caso não seja do tipo edit então o formulário virá vazio para que possa cadastrar um novo lançamento.
Mas se tiver um erro no servidor quando for salvar as alterações vai aparecer mensagem de erro falando que ocorreu um erro no servidor.

m) A linha a seguir se refere a encapsulamento, polimorfismo, herança, abstração ou injeção de dependências?

```
export class EntryFormComponent extends BaseResourceFormComponent<Entry>
implements OnInit
```

Herança.

n) A linha a seguir se refere a encapsulamento, polimorfismo, herança, abstração ou injeção de dependências?

```
ngOnInit() {
  this.loadCategories();
  super.ngOnInit();
}
```

Herança.

o) A linha a seguir se refere a encapsulamento, polimorfismo, herança, abstração ou injeção de dependências? Existe outra maneira?

```
@NgModule({
  providers: [
    EntryService
  ]
})
```

Injeção de dependência. Sim, podemos utilizar a injeção de dependência pelo constructor (private service: Service).