Avaliação Desenvolvedor Front-end Attornatus

O objetivo deste documento é identificar seus conhecimentos quanto às tecnologias utilizadas no cotidiano de desenvolvimento da equipe de Front-end na Attornatus Procuradoria Digital.

Esta análise propõe avaliar os seguintes temas:

- Qualidade de código
- JavaScript, TypeScript
- HTML, CSS
- Angular, RxJS, Testes unitários (Jest/ Jasmine)

A entrega deverá ser feita da seguinte forma:

- O prazo para entrega da avaliação será de até 7 dias após envio da mesma
- Encaminhar este documento com as perguntas respondidas e com o link do código público em sua conta do GitHub

Qualidade de código

1. Durante a implementação de uma nova funcionalidade de software solicitada, quais critérios você avalia e implementa para garantia de qualidade de software?
   Avalio o que pode ser feito, o que está feito e qual a melhor opção para entregar o software funcional, dividindo assim a nova funcionalidade e entregando o mínimo necessário para iniciar o aprofundamento.

2. Em qual etapa da implementação você considera a qualidade de software?
   A qualidade do software é algo a ser considerado desde o inicio, pensando em como manter ele na melhor qualidade possível.

TypeScript

3. Considerando seus conhecimentos de TypeScript, qualidade de código e boas práticas, quais melhorias você faria no seguinte código:

```
class Produto {
  id: number;
  descricao: string;
  quantidadeEstoque: number;

  constructor(id: number, descricao: string, quantidadeEstoque: number) {
    this.id = id;
    this.descricao = descricao;
    this.quantidadeEstoque = quantidadeEstoque;
  }
}

class Verdureira {
  produtos: Produto[];
```

```
  constructor() {
    this.produtos = [
      new Produto(1, "Maçã", 20),
      new Produto(2, "Laranja", 0),
      new Produto(3, "Limão", 20),
    ];
  }

  findProduto(produtoId: number){
        let  produto: Produto;
       for (let index = 0; index < this.produtos.length; index++) {
             if (this.produtos[index].id == produtoId) {
               produto = this.produtos[index];
          }
       }
}

  getDescricaoProduto(produto: Produto){
    return produto.id + " - " + produto.descricao + " (" +
produto.quantidadeEstoque + "x)";
  }

  hasEstoqueProduto(produto: Produto) {
    if (produto.quantidadeEstoque > 0) {
      return true;
    } else {
      return false;
    }
  }
}
```

Angular, RxJS

4. Com suas palavras, descreva duas principais vantagens e duas desvantagens de utilizar o framework Angular.
   Primeiro ponto positivo é o angular Cli, auxilia muito para realizar a criação da estrutura do projeto. E o segundo é que o angular é extremamente robusto, então muitas coisas necessárias para uma aplicação é encontrado nativamente.
   Primeiro ponto negativo é a curva de aprendizado difícil, e outro é a dificuldade de depuração devido ao roteamento limitado.

5. Como é compartilhado dados entre componentes Angular?
   Podemos compartilhar por @Input, ligação de propriedade,onde definimos no componente filho uma string, e destacamos no componente pai uma variavel, para passar dados de pai para filho. @Output & EventEmitter, ligação de evento, é o contrário, é utilizado para passar dados de filho para pai. E @ViewChild & AfterViewInit, dessa maneira podemos nos referir a um componente filho e acessar suas variáveis dentro do componente pai.

6. Cite um caso de uso prático de Angular Two-way binding.
   Em um caso no qual preciso receber um dado do cliente, por exemplo:

no HTML:

```
<div>
    <label for=”username”> Username:</label>
    <input [(ngModel)]=”username” >
    <p>{{username}}</p>
</div>
```

no TS:

```
export class UserComponent {
    username = “Gustavo”
    construtctor(){}
}
```

7. Cite 2 métodos de ciclo de vida do Angular, e para que podem ser utilizados em uma aplicação comum.
   ngOnDestroy: é executado imediatamente antes do angular destruir os componentes. Usado para evitar vazamentos de memoria.
   ngOnInit: é executado quando o angular exibe as variaveis vinculadas a dados ou quando o componente é inicializado. Usado para inicializar os dados no componente.

8. Por que devemos nos preocupar com Angular Change Detection?
   As vezes precisamos de estrategias, como pular verificações desnecessárias, ou precisamos otimizar o desempenho, e nos atentando ao Change Detection avisamos o angular apenas as ocasiões que é necessário a atualização.

9. Buscando obter as vantagens de **ChangeDetectionStrategy.OnPush** e aplicar qualidade de código, responda.
   O código a seguir não está funcionando, quais alterações você sugere para que o nome seja mostrado corretamente para o usuário na tela. Observação, não deve-se alterar **ChangeDetectionStrategy**, **setInterval** incrementando o contador ou lógicas de **PessoaService**.

```
import { ChangeDetectionStrategy, Component, Injectable, OnInit, OnDestroy }
from '@angular/core';
import { of, Subscription } from 'rxjs';
import { delay } from 'rxjs/operators';

@Injectable()
class PessoaService {
  /** @description Mock de uma busca em API com retorno em 0.5 segundos */
  buscarPorId(pessoaId: number) {
    return of({ id: pessoaId, nome: 'João' }).pipe(delay(500));
  }
}

@Component({
  selector: 'app-root',
  providers: [PessoaService],
  changeDetection: ChangeDetectionStrategy.OnPush,
```

```
  template: `<h1>{{ texto }}</h1>`
})
export class AppComponent implements OnInit, OnDestroy {
  texto: string;
  contador = 0;
  subscriptionBuscarPessoa: Subscription;
  constructor(private readonly pessoaService: PessoaService) {}

  ngOnInit(): void {
      const pessoaId = 1;
    this.subscriptionBuscarPessoa = this.pessoaService
      .buscarPorId(pessoaId)
      .subscribe((pessoa: Pessoa) => {
        this.texto = `Nome: ${pessoa.nome}`;
      });

    setInterval(() => this.contador++, 1000);
  }


  ngOnDestroy(): void { /** ... */}
}
```

10. Considerando seus conhecimentos em Angular, RxJS, e qualidade de código. Sem alterar **PessoaService,** como podemos melhorar o código a seguir:

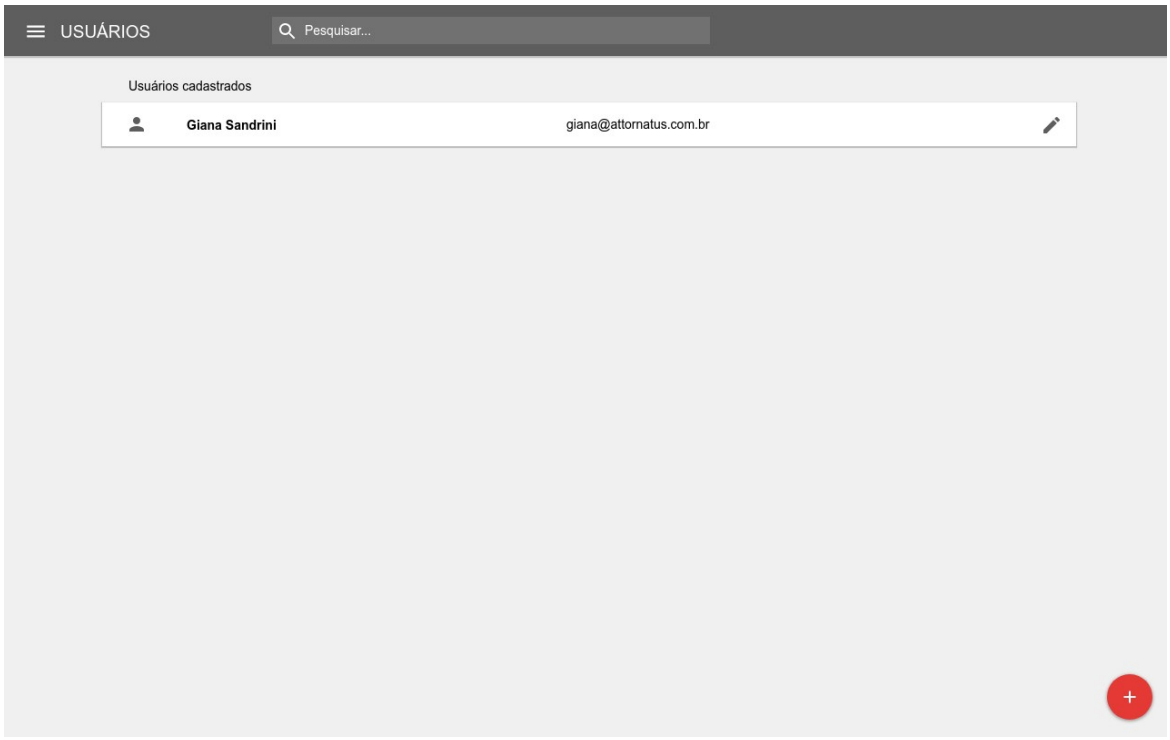
```
import { Component, Injectable, OnInit } from '@angular/core';
import { of } from 'rxjs';
import { delay } from 'rxjs/operators';

/** @description Mock de uma buscas em API com retorno em 0.5 segundos */
@Injectable()
class PessoaService {
  buscarPorId(pessoaId: number) {
    return of({ id: pessoaId, nome: 'João' }).pipe(delay(500));
  }

  buscarQuantidadeFamiliares(pessoaId: number) {
    return of(3).pipe(delay(500));
  }
}

@Component({
  selector: 'app-root',
  providers: [PessoaService],
  template: `<h1>{{ texto }}</h1>`,
})
export class AppComponent implements OnInit {
  texto: string;
  constructor(private readonly pessoaService: PessoaService) {}

  ngOnInit(): void {
    const pessoaId = 1;

    this.pessoaService.buscarPorId(pessoaId).subscribe((pessoa: Pessoa) => {
      this.pessoaService
        .buscarQuantidadeFamiliares(pessoaId)
        .subscribe((quantidadeFamiliares: number) => {
          this.texto = `Nome: ${pessoa.nome} | familiares:
${quantidadeFamiliares}`;
        });
    });
```

```
  }
}
```

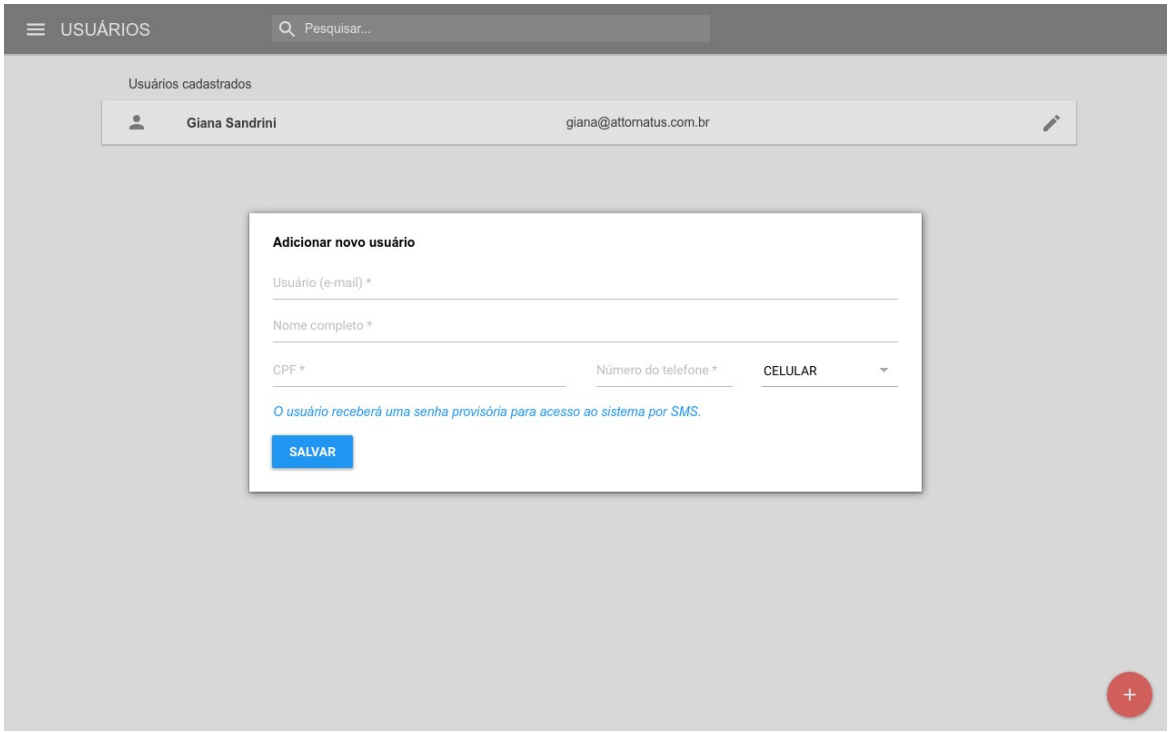
Desafio Angular

Reproduzir o protótipo de uma listagem de usuários utilizando o combo Angular + Material Design, ambos na versão 9 ou superior. A listagem deverá ter as seguintes funcionalidades:

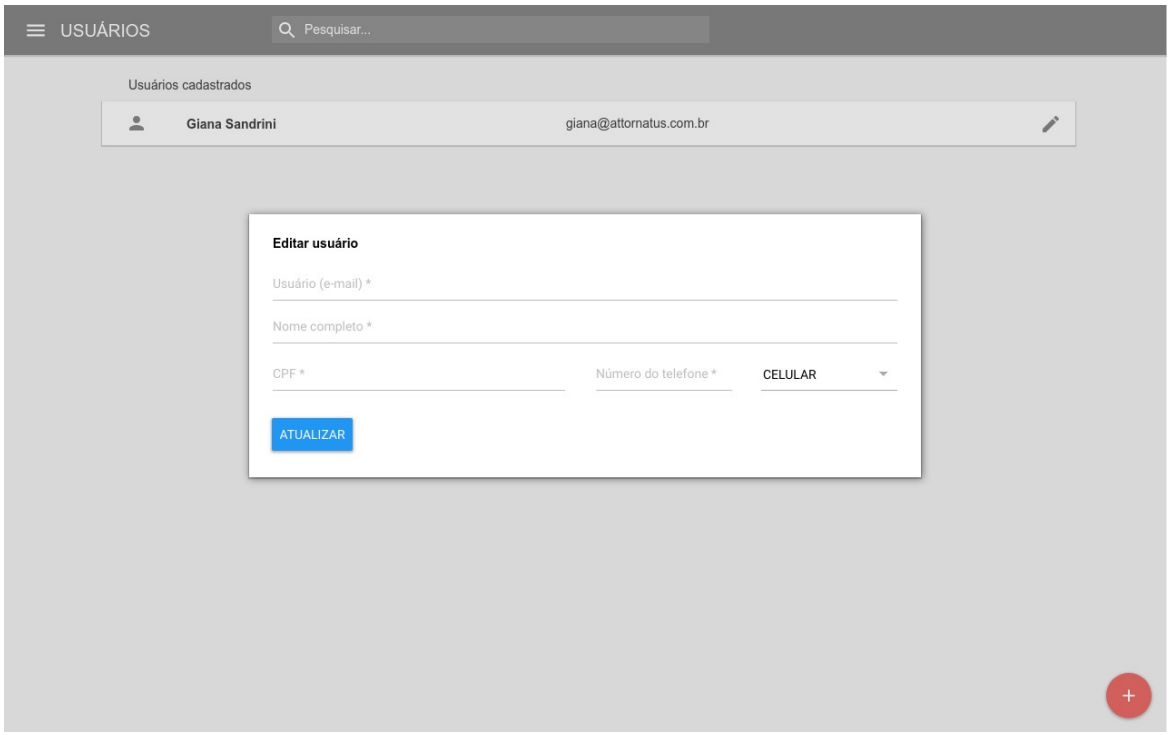
Tela de listagem de usuários:



Modal de cadastro de novo usuário com abertura através do botão vermelho que aparece na listagem:



Modal de edição do usuário com abertura através do botão com ícone de lápis encontrado nos cards de usuário:



Os dados podem ser estáticos, ou até se preferir, fazer uso de alguma biblioteca de APIs falsas como JSON Server.

Diferencial

- Melhorias no projeto em relação a tela apresentada no protótipo
- Cobertura de testes
- Clean Code

Será levado em avaliação

- Estrutura, arquitetura e organização do projeto
- Boas práticas de programação
- Alcance dos objetivos propostos.