# About me

Pedro da Cunha Lima Silva

- Desenvolvedor frontend
- Migrei para área TI na pandemia
- Bradesco, BMG, Grupo Panvel
- Programação transformou minha vida!

# Programação reativa com Angular e RxJS

Por Pedro Cunha

# Sobre a Apresentação

1) O que é RxJS
2) Elementos básicos
3) Angular + RxJS
4) Caso 1
5) Caso 2
6) Descomplique o RxJS

# RxJs

- Biblioteca
- Observables são a base
- Código assíncrono ou baseado em callbacks
- Programação reativa

# Elementos básicos do RxJS

## Observable

- Uma stream de dados ou eventos que são emitidos ao longo do tempo

## Operadores Pipe

- Funções puras
- Transformam dados
- Combinam/redirecionam Observables
- Declarativos

# Elementos básicos do RxJS

**Observable + Pipes formam uma pipeline que transformam a stream que percorre através dela.**

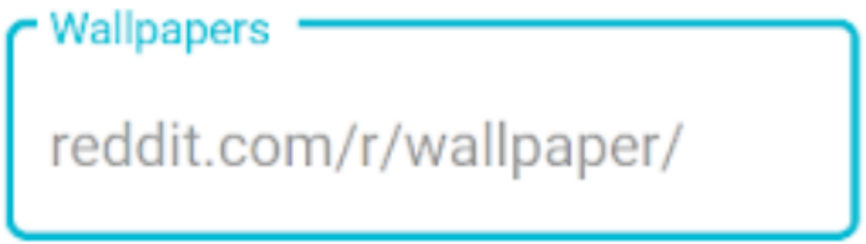
# Angular + RxJs

- HttpClient
- EventEmitters (Output de componentes)
- Reactive Forms
- Router
- NgRx

# Vamos aos Casos

# Caso 1

## Barra de Busca




- Não temos que utilizar muitos elementos do RxJS.
- Podemos simplesmente fazer uma subscrição no Observable retornado do HTTPClient
- Converter Observable para Promise com o toPromise()

**“Não preciso adicionar tanta complexidade a uma simples requisição GET...”**

Será??

# Caso 1

searchBar.component.html

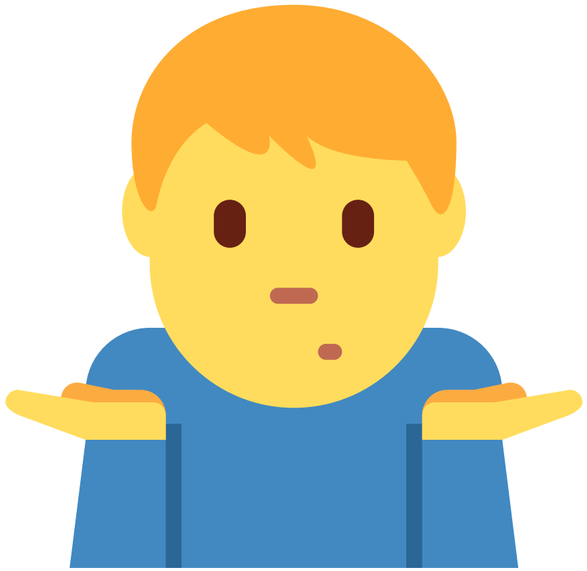
```html
<div class="search-container">
  <mat-form-field appearance="outline">
    <mat-label>Wallpapers</mat-label>
    <input (keyup)="doSearch($event)" />
  </mat-form-field>
  <div class="imgs">
    <img
      *ngFor="let img of results"
      [src]="img.data.thumbnail"
      [alt]="img.data.title"
    />
  </div>
</div>
```

searchBar.component.ts

```ts
doSearch({ target }: KeyboardEvent) {
  const { value } = target as HTMLInputElement;
  this.service.search(value).subscribe(({ data: { children } }) => {
    this.results = [...children];
  });
}
```

# Caso 1

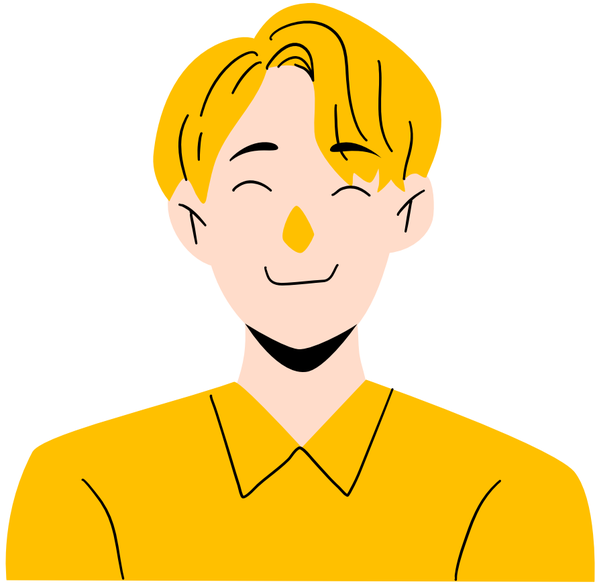
O que foi feito de errado?



- Tentamos realizar toda lógica manualmente
- Preferimos utilizar simples chamadas de API por evento e não abstrações de alto nível
- Não pensa nos usuários com problemas de conexão
- Deixamos nosso código extremamente simples, ignorando a complexidade do problema

# Caso 1

O que desejamos alcançar?



- Performance e resiliência
- Menos requisições
- Evitar requisições redundantes
- Evitar requisições vazias
- Cancelar requisições desnecessárias
- Retry em erros

redditSearch.component.html

```
<div class="search-container">
  <mat-form-field appearance="outline">
    <mat-label>Wallpapers</mat-label>
    <input [formControl]="search" />
  </mat-form-field>
  <div class="imgs">
    <ng-container>
      <img
        *ngFor="let img of results$ | async"
        [src]="img.data.thumbnail"
        [alt]="img.data.title"
      />
    </ng-container>
  </div>
</div>
```

# Caso 1

- **formControl**: retorna as alterações no input como observable
- **observable**: resultados obtidos das requisições Http
- **async pipe**: responsável pela subscrição no template

# Caso 1

redditSearch.component.ts

```
results$: Observable<RedditResult[]> = this.search.valueChanges.pipe(
  map((search) => search.trim()),
  debounceTime(300),
  distinctUntilChanged(),
  filter((search) => search !== ''),
  switchMap((search) =>
    this.service.search(search).pipe(retry(3), pluck('data', 'children'))
  )
);
```

**Operadores RxJS:** nos permitem combinar dados assíncronos complexos de maneira declarativa

# Caso 1.2

## Combinando Observable

```
subReddit = new FormControl();
subReddit$ = this.subReddit.valueChanges.pipe(startWith(''));

search$ = this.search.valueChanges.pipe(
  startWith(''),
  map((search) => search.trim()),
  debounceTime(300),
  distinctUntilChanged(),
  filter((search) => search !== '')
);

filteredResults$ = combineLatest([this.subReddit$, this.search$]).pipe(
  switchMap(([subReddit, search]) =>
    this.service
      .subRedditSearch(subReddit, search)
      .pipe(retry(3), pluck('data', 'children'))
  )
);
```

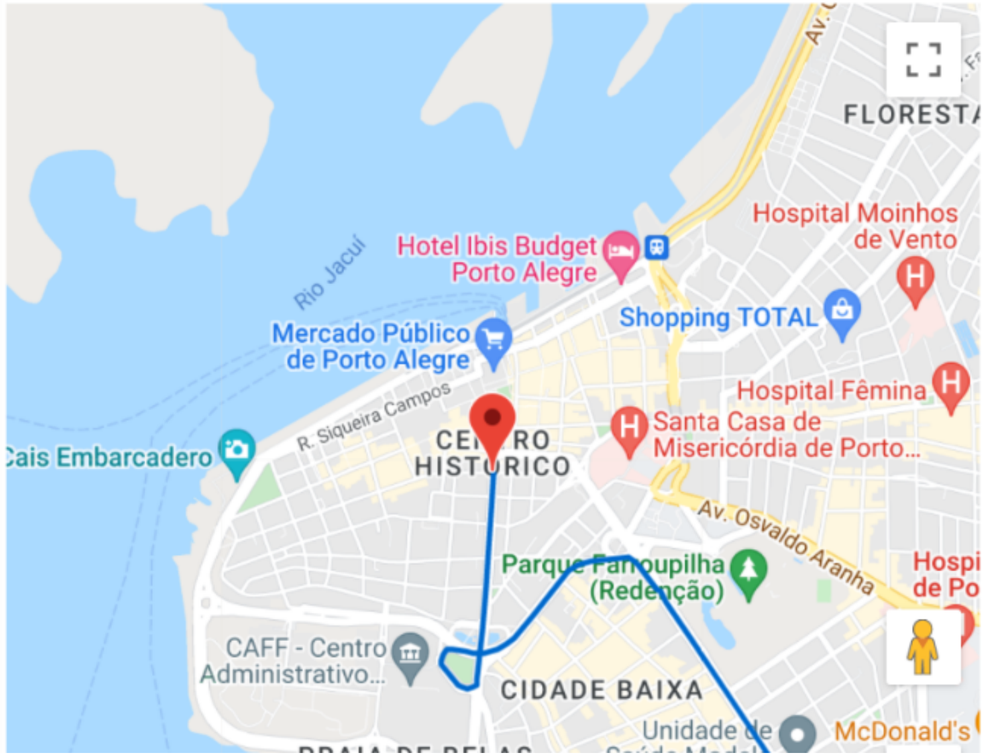
**combineLatest:**
Combina streams de diferentes observables.

Emite valores apenas quando todos os observables utilizados emitirem!

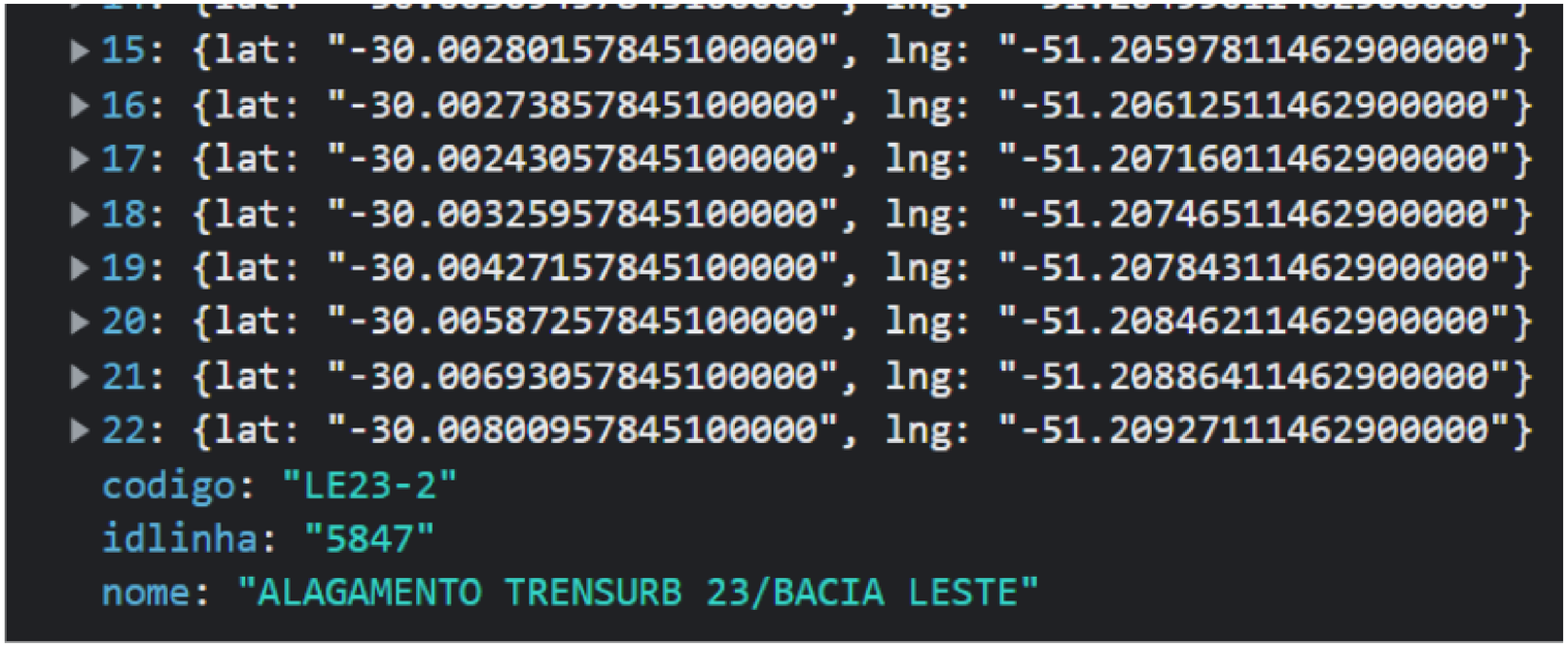
# Caso 2

# BusApp

| Código | Linha |
|---|---|
| 250-1 | 1 DE MAIO |
| 250-2 | 1 DE MAIO |
| T11-1 | 3ª PERIMETRAL |
| T11-2 | 3ª PERIMETRAL |
| T111-2 | 3ª PERIMETRAL/ATE DR. CAMPOS VELHO |
| T111-1 | 3ª PERIMETRAL/ATE DR. CAMPOS VELHO |
| 214-1 | 5ª UNIDADE/ESCOLAR |
| 214-2 | 5ª UNIDADE/ESCOLAR |



# Fonte de Dados



# Solução 'simples'

```
<bus-line-list
  [items]="busLines$ | async"
  (selectedBus)="onBusSelect($event)"
></bus-line-list>
<bus-route [coordinates]="selectedCoordinates">
</bus-route>
```

```
onBusSelect(bus: Bus) {
  this.busService.getRoute(bus.id).subscribe(itineraryObj => {
    this.selectedCoordinates = Object.entries(itineraryObj)
    .map(([key, prop]) => {
      if (!isNaN(Number(key))) {
        prop.lat = Number(prop.lat);
        prop.lng = Number(prop.lng);
        return prop;
      }
      return;
    })
    .filter(Boolean)
  })
}
```

# Prevenindo double-clicks!

```
<bus-line-list
  [items]="busLines$ | async"
  (selectedBus)="selectedBus$.next($event)"
></bus-line-list>
<bus-route [coordinates]="selectedCoordinates$ | async">
</bus-route>
```

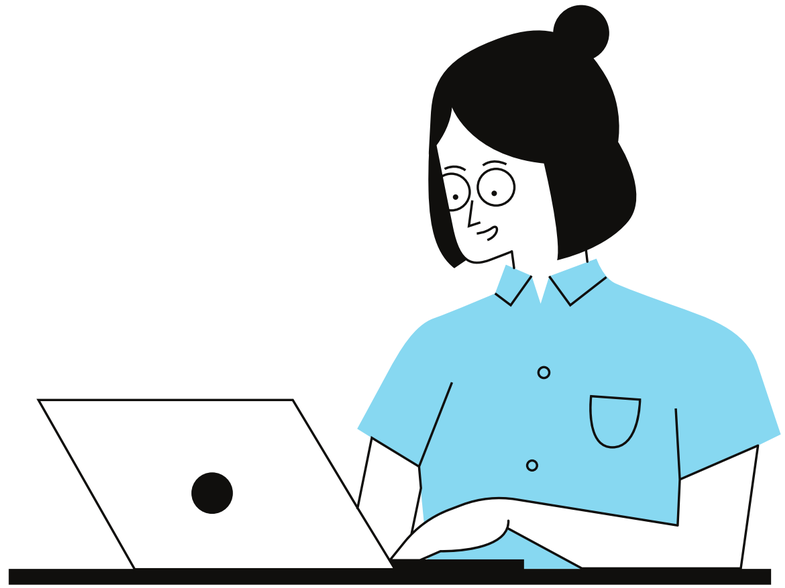
**Subjects:** são observables que nos permitem, além de receber dados, enviá-los por ele

```
selectedBus$ = new Subject<Bus>();
selectedCoordinates$ = this.selectedBus$.pipe(
  throttleTime(400),
  distinctUntilChanged(),
  switchMap((bus) =>
    this.busService.getRoute(bus.id).pipe(
      map((itineraryObj) =>
        Object.entries(itineraryObj)
          .map(([key, prop]) => {
            if (!isNaN(Number(key))) {
              prop.lat = Number(prop.lat);
              prop.lng = Number(prop.lng);
              return prop;
            }
            return;
          })
          .filter(Boolean)
      ))));
```

# Descomplique o RxJS

# Descomplique o RxJS

- Subscrição dentro de subscrição
- Pegar dados de um observable para criar outro observable manualmente
    - Conecte o fluxo de dados entre streams com operadores como switchMap, mergeMap, concatMap, exhaustMap..



# Descomplique o RxJS

- Lidar com múltiplas subscrições em um único componente
  - Seu componente pode ter responsabilidades demais, repense sua organização.

- Subscrição manual de dados que serão exibidos na UI
  - Utilize o pipe async no template
  - Deixe o Angular trabalhar por você!

# Obrigado!

https://www.linkedin.com/in/p3cunha/

pcls_@hotmail.com

https://github.com/p3cunha/AngularTalks