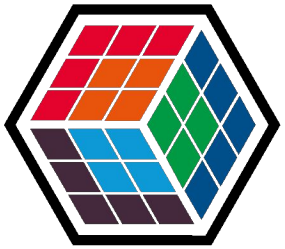


THE DEVELOPER'S CONFERENCE

Clean Code e Clean Architecture no Front-End

Wesley W. de Azevedo Palmeira
Senior Software Developer

Quem sou eu?

THE DEVELOPER'S CONFERENCE

- Wesley Wevertton de Azevedo Palmeira, 29 anos
- Esposo, Nordestino (Campina Grande - PB) e Baterista
- Graduado em Engenharia de computação (IFPB - CG)
- Pouco mais de 8 anos no mercado de TI
- Atualmente Desenvolvedor Sênior / Líder Técnico

beta learning

ClearSale

Agenda

THE DEVELOPER'S CONFERENCE

- Motivação
- Aplicação "Exemplo"
- Código Limpo
  - Conceito
  - Nomes
  - Condicionais
  - Funções
- Arquitetura Limpa
  - Conceito
  - Princípio da Responsabilidade única (**S**ingle responsibility principle)
  - Princípio do aberto/fechado (**O**pen–closed principle)
  - Princípio da Inversão de dependência (**D**ependency Inversion principle)
- Conclusão

Motivação

THE DEVELOPER'S CONFERENCE

No início do projeto…

Putz, muito easy, ta funcionando, terminei em 2 horas!!

Código RUIM

Deu trabalho em, levei 2 dias para fazer, porém ta lindo

Código BOM

Motivação

THE DEVELOPER'S CONFERENCE

Alguns meses depois….

Impossível entender, mexo em um local, quebra em outro WTF

Código RUIM

Muito bom mexer nesse projeto, tá tudo muito organizado!!

Código BOM

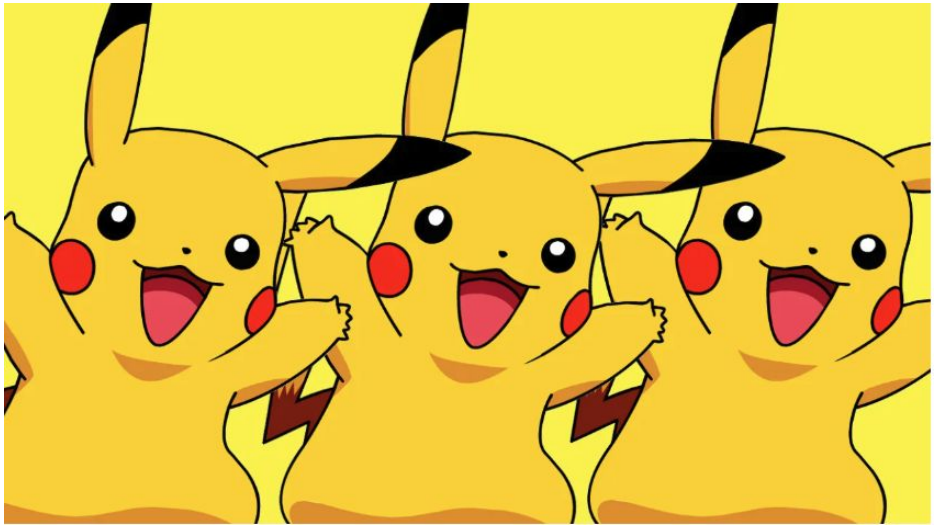
Motivação

THE DEVELOPER'S CONFERENCE

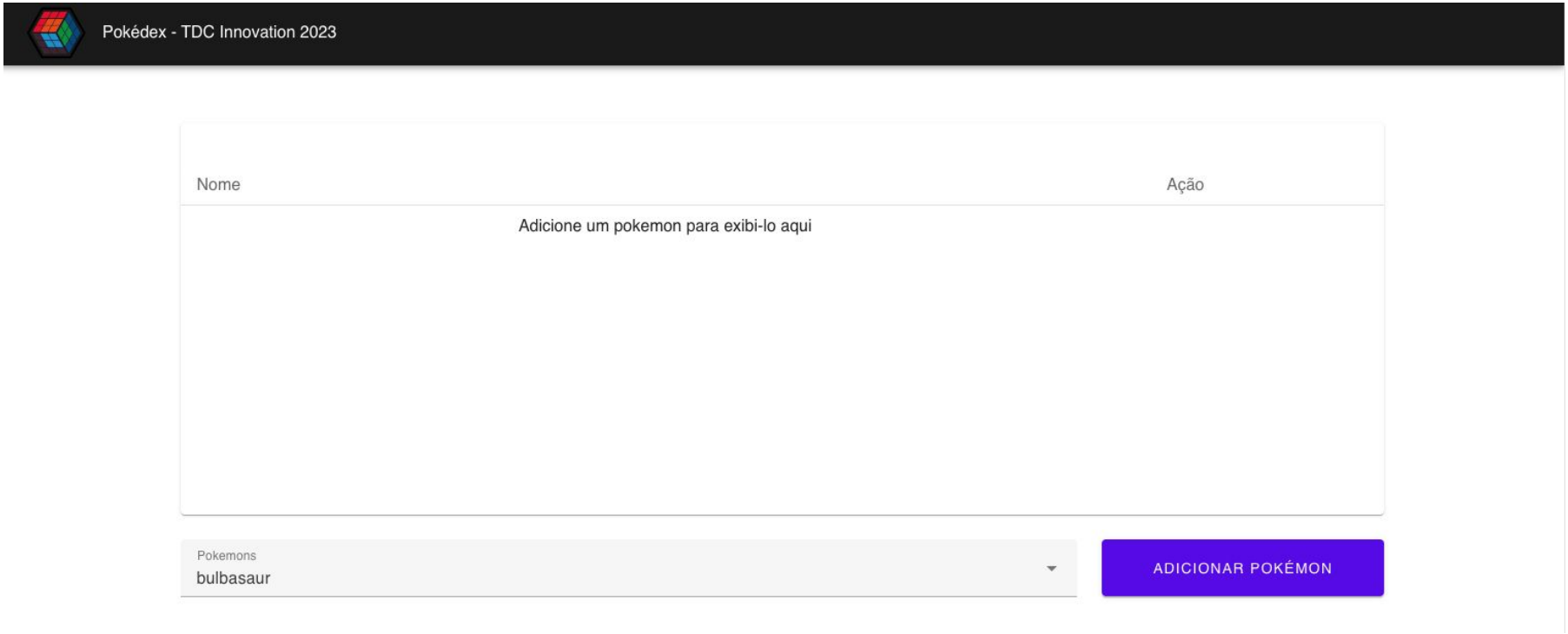
Aplicação "Exemplo"

THE DEVELOPER'S CONFERENCE

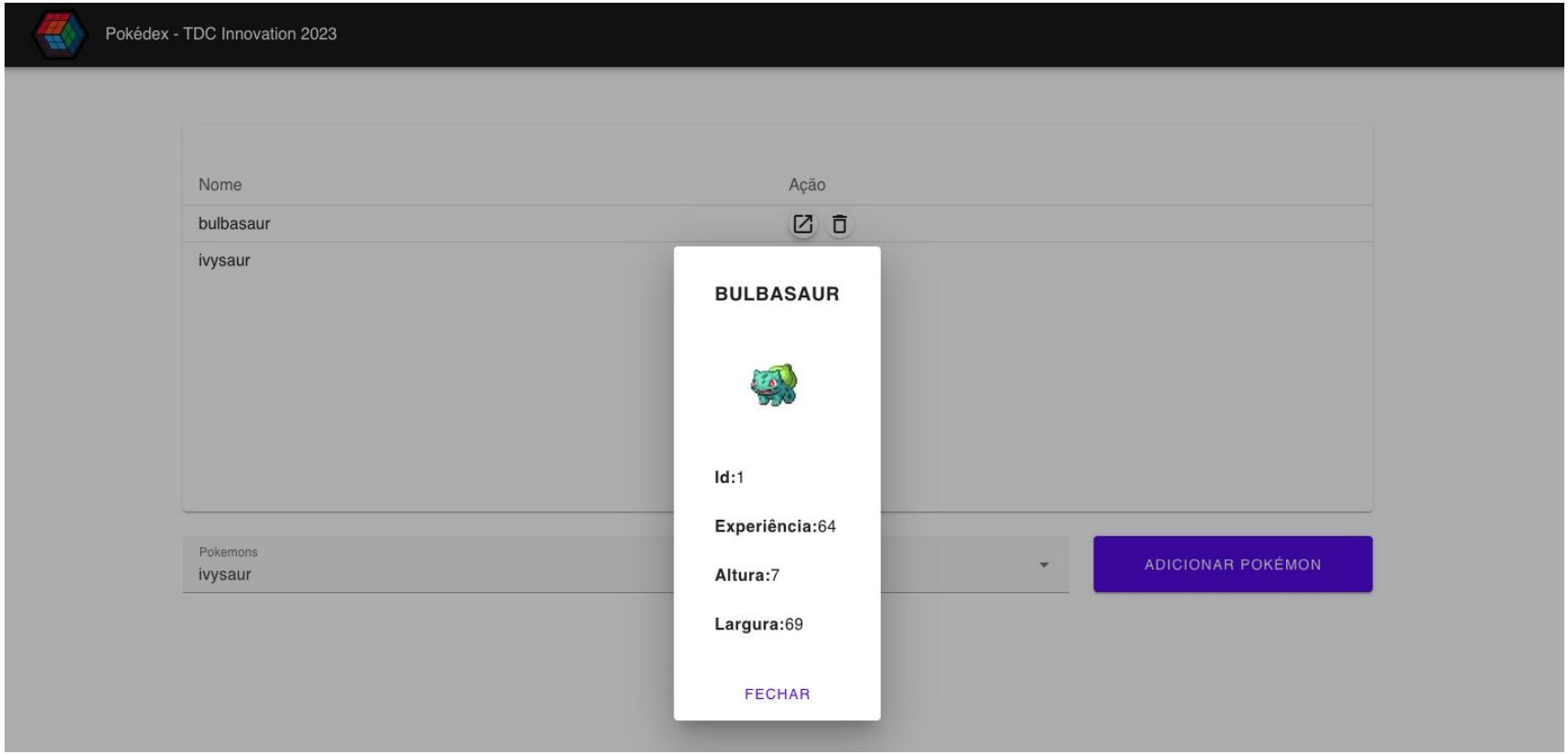
- Mini Pokédex utilizando a https://pokeapi.co/
- Vite + Vue3 + Vuetify + Typescript
- Será possível adicionar, remover e visualizar pokémons dessa lista



Aplicação "Exemplo"



Aplicação "Exemplo"

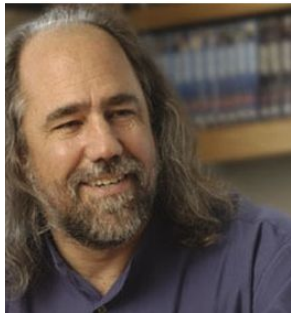


Código Limpo

THE DEVELOPER'S CONFERENCE

"Um código limpo é simples e direto"

Grady Brooch, autor do livro Software Engineering with Ada

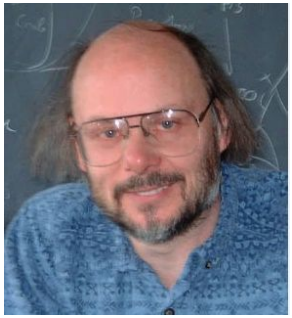


Código Limpo

THE DEVELOPER'S CONFERENCE

"O código limpo faz bem apenas uma coisa"

Bjarne Stroustrup, criador do C++

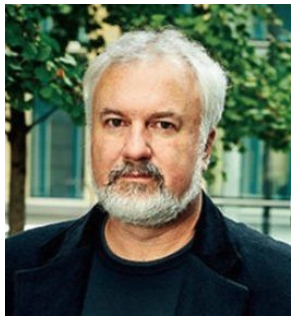


Código Limpo

THE DEVELOPER'S CONFERENCE

"Um código limpo sempre parece que foi escrito por alguém que se importava"

Michael Feathers, autor do livro Working Effectively with Legacy Code



Nomes

THE DEVELOPER'S CONFERENCE

Ruim

```
114  type type1 = {
115    name: string;
116    url: string;
117  };
118
119  type type2 = {
120    id: string;
121    name: string;
122    xp: string;
123    height: string;
124    weight: string;
125    photo: string;
126  };
127
128  const List1 = ref<type1[]>([]);
129  const List2 = ref<type1[]>([]);
```

Bom

```
114  type PokemonCompact = {
115    name: string;
116    url: string;
117  };
118
119  type Pokemon = {
120    id: string;
121    name: string;
122    experience: string;
123    height: string;
124    weight: string;
125    photo: string;
126  };
127
128  const pokemonTableList = ref<PokemonCompact[]>([]);
129  const pokemonSelectList = ref<PokemonCompact[]>([]);
```

Utilize nomes descritivos

Evite abreviações/codificações

Mantenha um padrão

Nomes

THE DEVELOPER'S CONFERENCE

Ruim

```
134  const add = () => {
135    if (pokemonSelectModel.value) {
136      pokemonTableList.value.push(
137        pokemonSelectModel.value
138      );
139    }
140  };
141
142  const remove = (i: PokemonCompact) => {
143    const index = pokemonTableList.value.findIndex(
144      (el) => el.name === i.name
145    );
146    pokemonTableList.value.splice(
147      index, 1
148    );
149  };
```

Bom

```
134  const pushPokemonToTable = () => {
135    if (pokemonSelectModel.value) {
136      pokemonTableList.value.push(
137        pokemonSelectModel.value
138      );
139    }
140  };
141
142  const removePokemonFromTable = (
143    pokemonToRemove: PokemonCompact
144  ) => {
145    const pokemonIndex = pokemonTableList.value.findIndex(
146      (pokemon) => pokemon.name === pokemonToRemove.name
147    );
148    pokemonTableList.value.splice(
149      pokemonIndex, 1
150    );
151  };
```

Nomes de métodos/funções devem começar com verbo

Evite mapas mentais

Condicionais

THE DEVELOPER'S CONFERENCE

Ruim

```
const add = () => {
  if (!current.value || !current.value.name) {
    alert('Escolha um pokemon para adicionar na tabela')
  }
  else if(
    List1.value.findIndex(
      el => el.name === current.value?.name
    ) === -1){
    List1.value.push(
      current.value
    );
  } else {
    alert('Pokemon ja existe na tabela')
  }
};
```

Evite negações de condicionais

Encapsule condicionais

Evite aninhamento de condicionais

Bom

```
const pushPokemonToTable = () => {
  if (validatePokemon(pokemonSelectModel.value)) {
    pushPokemonToTableList(pokemonSelectModel.value)
  } else {
    alert('Escolha um pokemon para adicionar na tabela')
  }
};
```

```
const pushPokemonToTableList = (pokemon: PokemonCompact) => {
  if(checkHasPokemonOnTable(pokemon)) {
      alert('Pokemon ja existe na tabela')
  } else {
      pokemonTableList.value.push(
        pokemon
      );
  }
}
```

Funções

THE DEVELOPER'S CONFERENCE

Crie funções pequenas (20 linhas no máximo)

```
const removePokemonFromTable = (
  pokemonToRemove: PokemonCompact
) => {
  const pokemonIndex = getPokemonTableIndex(pokemonToRemove);
  pokemonTableList.value.splice(pokemonIndex, 1);
};
```

O número ideal de parâmetros de uma função é ZERO

Devem fazer apenas umas coisa

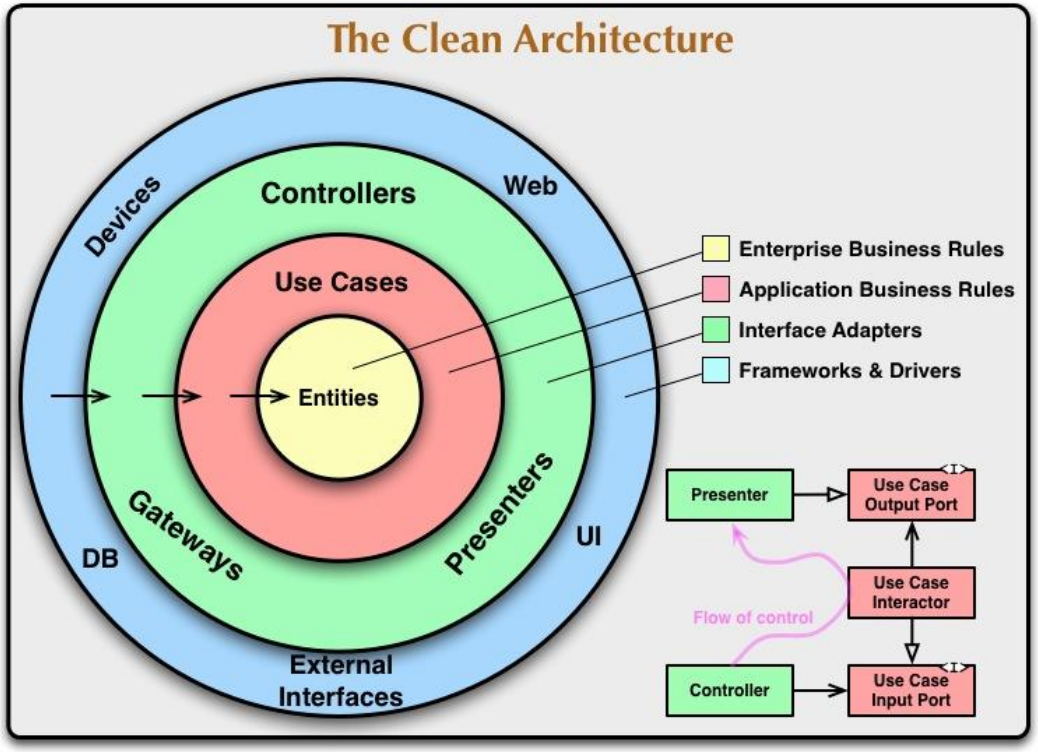
Arquitetura Limpa

THE DEVELOPER'S CONFERENCE

Padrão arquitetural com o objetivo de promover a implementação de sistemas que favorecem a reusabilidade de código, coesão, independência de tecnologia e testabilidade.

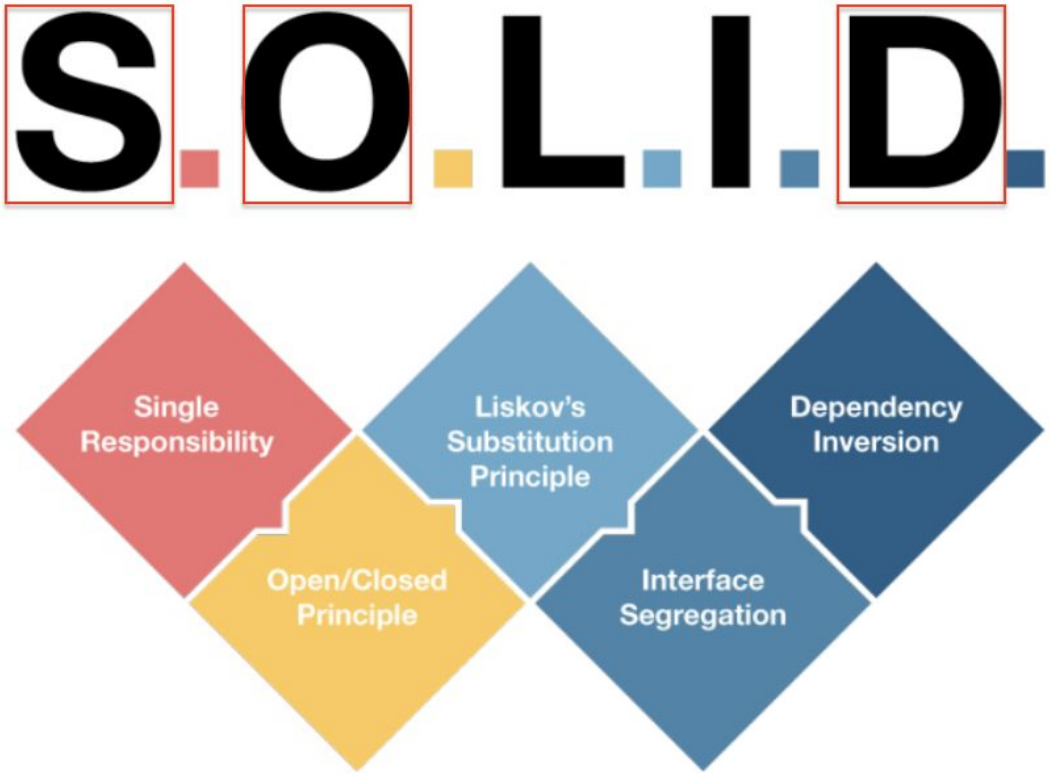
Arquitetura Limpa

THE DEVELOPER'S CONFERENCE



Arquitetura Limpa

THE DEVELOPER'S CONFERENCE



Princípio da Responsabilidade única (Single responsibility principle)

THE DEVELOPER'S CONFERENCE

Cada módulo ou classe deve ter responsabilidade sobre uma única parte da funcionalidade fornecida pelo software.

Princípio da Responsabilidade única (Single responsibility principle)

THE DEVELOPER'S CONFERENCE

Problemas da nossa aplicação:

- Tudo dentro do App.vue
- HTML do App.vue com 104 linhas
- Várias responsabilidades em um mesmo arquivo (Regra de negócio, visualização e consultas a API)

Princípio da Responsabilidade única (Single responsibility principle)

THE DEVELOPER'S CONFERENCE

Passo 1

```
<template>
  <v-app>
    <Header />
    <v-container class="mt-16">
      <v-row>
        <v-col cols="12">
          <PokemonTable
            :pokemons="pokemonTableList"
            @details="showPokemonDetails"
            @remove="removePokemonFromTable"
          />
        </v-col>
      </v-row>
      <PokemonSelect
        :pokemon-select-list="pokemonSelectList"
        @push="pushPokemonToTable"
      />
    </v-container>
    <PokemonDialog
      :show="showDetailsDialog"
      :pokemon-details="pokemonDetailsModel"
      @hide="$event => showDetailsDialog = false"
    />
  </v-app>
</template>
```

Princípio da Responsabilidade única (Single responsibility principle)

THE DEVELOPER'S CONFERENCE

Passo 2

```
export default class Pokedex {
    pokemons: PokemonCompact[];
    constructor() {
        this.pokemons = [];
    }
    private getPokemonIndex (pokemonToFind: PokemonCompact) {
        return this.pokemons.findIndex(
          (pokemon) => pokemon.name === pokemonToFind.name
        );
    };
    private checkHasPokemon(pokemonToCheck?: PokemonCompact) {
        return (
            this.pokemons.findIndex(
              (pokemon) => pokemon.name === pokemonToCheck?.name
            ) !== -1
        );
    }
```

```
    setPokemons(pokemons: PokemonCompact[]) {
        this.pokemons = pokemons;
    }
    pushPokemon(pokemon: PokemonCompact) {
        if (this.checkHasPokemon(pokemon)) {
            throw(new Error('Pokemon ja existe na tabela'));
        } else {
            this.pokemons.push(pokemon);
        }
    }
    removePokemon(pokemon: PokemonCompact) {
        const pokemonIndex = this.getPokemonIndex(pokemon);
        this.pokemons.splice(pokemonIndex, 1);
    }
}
```

Princípio da Responsabilidade única (Single responsibility principle)

THE DEVELOPER'S CONFERENCE

Passo 3

```
export default class PokemonGatewayHttp {
  private parsePokemonDetails(pokemonData: PokemonData) {
    return {
      id: pokemonData.id,
      name: pokemonData.name,
      experience: pokemonData.base_experience,
      height: pokemonData.height,
      weight: pokemonData.weight,
      photo: pokemonData.sprites.front_default,
    };
  }

  async getPokemons() {
    const { data } = await axios.get("https://pokeapi.co/api/v2/pokemon");
    return data.results;
  }

  async getPokemonByUrl(pokemonUrl: string) {
    const { data } = await axios.get(pokemonUrl);
    const pokemonParsed = this.parsePokemonDetails(data);
    return pokemonParsed;
  }
}
```

Princípio da Responsabilidade única (Single responsibility principle)

THE DEVELOPER'S CONFERENCE

```
37  const pokemon = ref<Pokemon>();
38  const showDetailsDialog = ref(false);
39  const tablePokedex = reactive(new Pokedex());
40  const selectPokedex = reactive(new Pokedex());
41  const pokemonsGateway = new PokemonGatewayHttp();
42
43  onMounted(async () => {
44    const pokemons = await pokemonsGateway.getPokemons();
45    selectPokedex.setPokemons(pokemons);
46  });
47  const pushPokemon = (pokemon: PokemonCompact) => {
48    try {
49      tablePokedex.pushPokemon(pokemon);
50    } catch (error) {
51      alert(error);
52    }
53  }
54  const removePokemon = (pokemonToRemove: PokemonCompact) => {
55    tablePokedex.removePokemon(pokemonToRemove);
56  };
57  const showPokemonDetails = async (pokemonUrl: string) => {
58    pokemon.value = await pokemonsGateway.getPokemonByUrl(pokemonUrl)
59    showDetailsDialog.value = true;
60  };
61  </script>
```

Princípio do aberto/fechado (Open–closed principle)

THE DEVELOPER'S CONFERENCE

Entidades de software devem ser abertas para extensão, mas fechadas para modificação.

Princípio do aberto/fechado (Open–closed principle)

THE DEVELOPER'S CONFERENCE

```
<template>
  <v-row>
    <v-col cols="9">
      <v-select
        v-model="pokemonSelectModel"
        label="Pokémons"
        :items="pokemons"
        item-title="name"
        return-object
      ></v-select>
    </v-col>
    <v-col cols="3">
      <v-btn height="55px" block color="primary" @click="pushPokemon"
        >Adicionar Pokémon</v-btn
      >
    </v-col>
  </v-row>
</template>
```

Princípio do aberto/fechado (Open–closed principle)

THE DEVELOPER'S CONFERENCE

```
<template>
  <v-row>
    <v-col cols="9">
      <v-select
        v-model="pokemonSelectModel"
        label="Pokémons"
        :items="pokemons"
        item-title="name"
        return-object
      ></v-select>
    </v-col>
    <v-col cols="3">
      1 reference
      <slot name="right" :currentPokemon="pokemonSelectModel"></slot>
    </v-col>
  </v-row>
</template>
```

Princípio do aberto/fechado (Open–closed principle)

THE DEVELOPER'S CONFERENCE

```
<PokemonSelect :pokemons="selectPokedex.pokemons">
  <template #right="scope">
    <v-btn height="55px" block color="primary"
      @click="$event => pushPokemon(scope.currentPokemon)"
      Adicionar Pokémon
    </v-btn>
  </template>
</PokemonSelect>
```

Princípio da Inversão de dependência (Dependency Inversion principle)

THE DEVELOPER'S CONFERENCE

Dependa de abstrações e não de implementações.

Princípio da Inversão de dependência (Dependency Inversion principle)

THE DEVELOPER'S CONFERENCE

```
const pokemonsGatewayHttp = new PokemonGatewayHttp();

onMounted(async () => {
  const pokemons = await pokemonsGatewayHttp.getPokemons();
```

```
async getPokemons() {
  const { data } = await axios.get("https://pokeapi.co/api/v2/pokemon");
  return data.results;
}
```

Princípio da Inversão de dependência (Dependency Inversion principle)

THE DEVELOPER'S CONFERENCE

```typescript
export default interface PokemonGateway {
    getPokemons(): Promise<PokemonCompact[]>;
    getPokemonByUrl(pokemonUrl: string): Promise<Pokemon>;
}
```

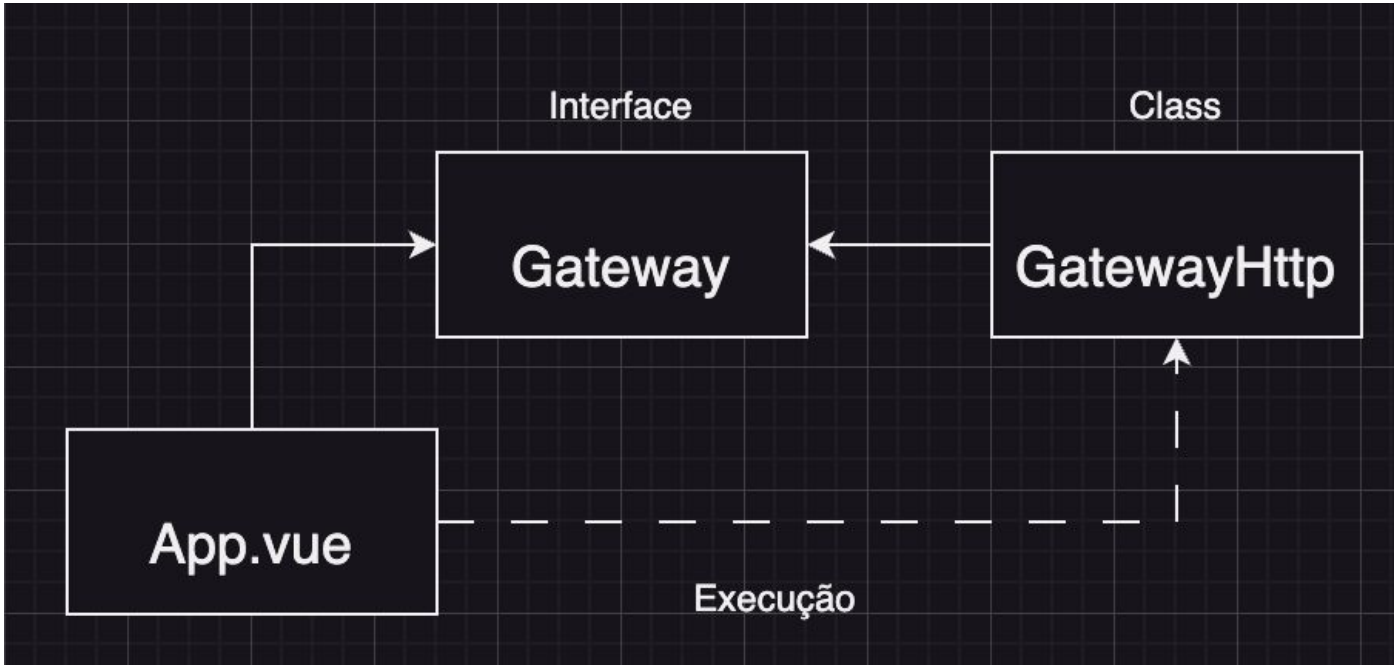
```typescript
const pokemonsGateway = inject("pokemonGateway") as PokemonGateway

onMounted(async () => {
  const pokemons = await pokemonsGateway.getPokemons();
```

Princípio da Inversão de dependência (Dependency Inversion principle)

THE DEVELOPER'S CONFERENCE

```
app.provide("pokemonGateway", new PokemonGatewayHttp());
```



Conclusão

THE DEVELOPER'S CONFERENCE

- Clean Code e Clean Architecture são dicas não regras
- Dá para aplicar as dicas do Uncle Bob no Frontend
- Pensar na qualidade do código não é perda de tempo

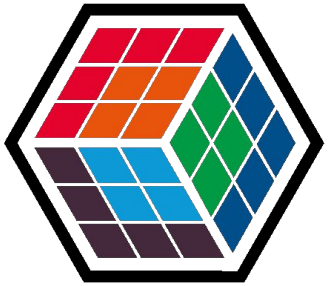
Obrigado! Dúvidas?

nolderosw

nolderos (Wesley Azevedo)

@wesley150

@leloazevedo



THE DEVELOPER'S CONFERENCE