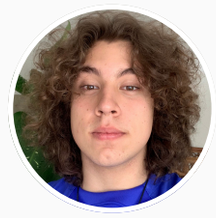
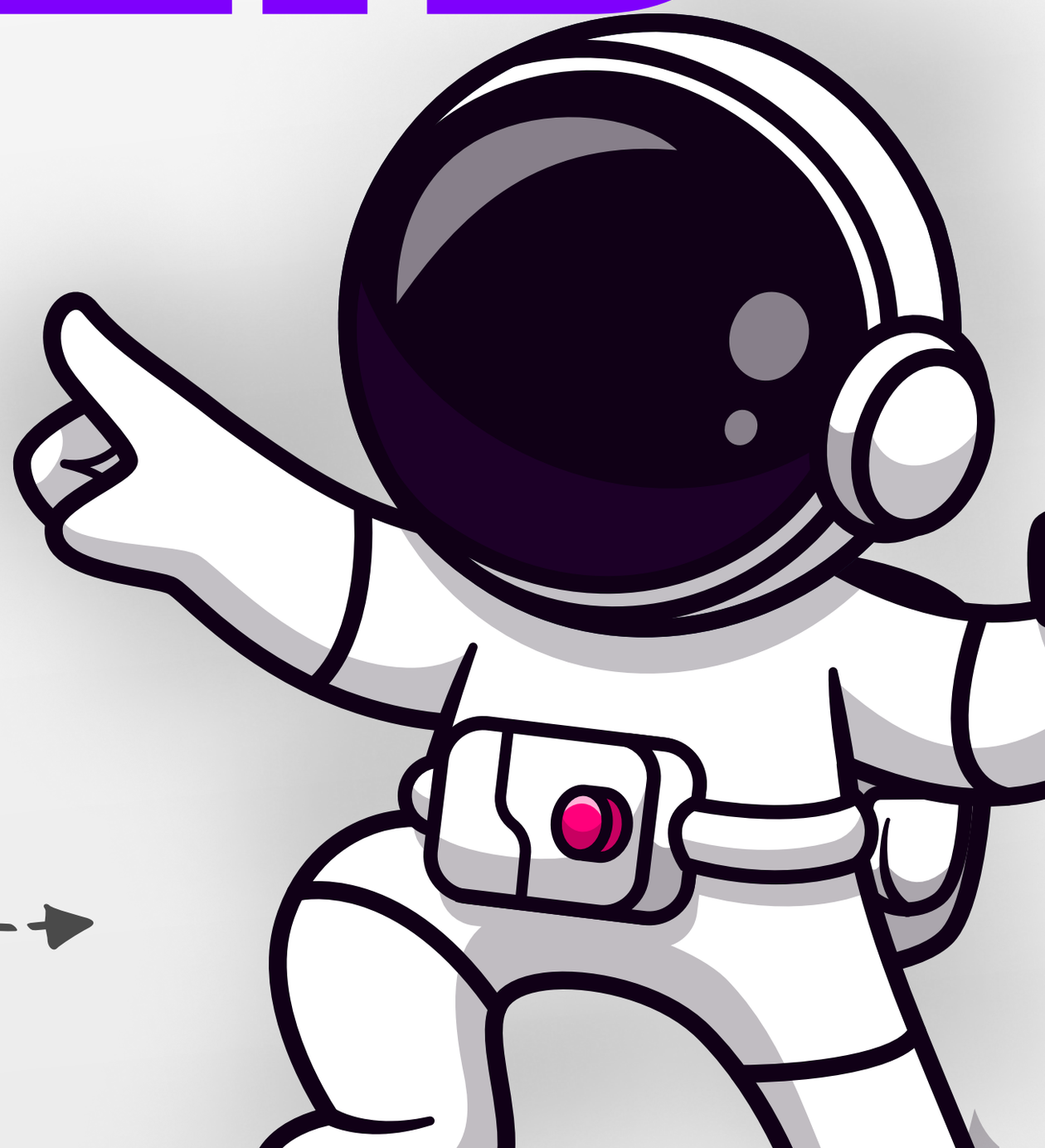


pare de enrolar e aprenda

SOLID

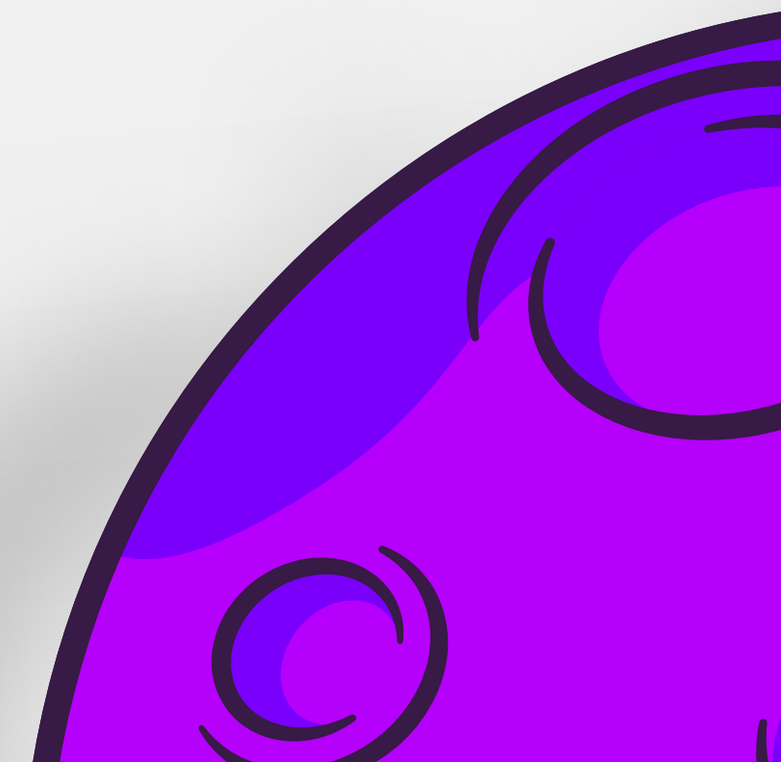
De uma vez por todas!



Caso você more na Lua...

SOLID é um conjunto de cinco princípios para organizar o código de forma flexível e fácil de manter.

agora vamos para cada um...



Single Responsibility

Responsabilidade única

Cada coisa deve ter somente uma responsabilidade. Não deixe tudo no mesmo lugar!

```
1 function CalcularNumeros(x, y, acao) {
2   if (acao === "somar") return x + y;
3   if (acao === "subtrair") return x - y;
4 }
```

```
1 function SomarNumeros(x, y) {
2   return x + y;
3 }
4
5 function SubtrairNumeros(x, y) {
  return x - y;
```

Open / Closed

Aberto / Fechado

Código sempre aberto para extensão, mas fechado para modificação.

```typescript
interface ProcessadorPagamento {
  processarPagamento(): void;
}

class ProcessadorCartaoCredito implements ProcessadorPagamento {
  processarPagamento() {
    console.log("Processando pagamento com cartão de crédito...");
  }
}

class ProcessadorPayPal implements ProcessadorPagamento {
  processarPagamento() {
    console.log("Processando pagamento com PayPal...");
  }
}
```

Liskov Substitution

Substituição de Liskov

Uma classe filha deve saber fazer tudo que sua classe mãe já faz.

```
1 class Cachorro {
2   latir() {
3     console.log("Au au!"); // todos os cachorros sabem latir
4   }
5 }
6
7 class Caramelo extends Cachorro {}
8 class PastorAlemao extends Cachorro {}
9
```

Interface Segregation

Segregação de Interfaces

Nada deve ser forçado a depender de métodos que **não utiliza**.

```typescript
interface AnimalQueCaminha {
  caminhar(): void;
}

interface AnimalQueVoa {
  voar(): void;
}

class Cachorro implements AnimalQueCaminha {
  caminhar() {
    console.log("Caminhando ... ");
  }
}

class Pato implements AnimalQueCaminha, AnimalQueVoa {
  caminhar() {
    console.log("Caminhando ... ");
  }
  voar() {
    console.log("Voando ... ");
  }
}
```

Dependency Inversion

Inversão de Dependências

Nunca dependa de implementações,
mas sim de abstrações.

```ts
interface BancoDeDados {
  query(sql: string): void;
}

class BancoMySQL implements BancoDeDados {
  query(sql: string): void {
    console.log(`Executando query com MySQL: ${sql}`);
  }
}

class UsersService {
  // O service deve depender somente de interfaces!
  private db: BancoDeDados;

  constructor(db: BancoDeDados) {
    this.db = db;
  }

  lerUsuario(id: number): void {
    this.db.query(`SELECT * FROM users WHERE id = ${id}`);
  }
}

```

Sim, eu sei que pode parecer complexo, é difícil explicar tudo em um único post. 😥

Mas agora que você já deu o 1º passo, vou te indicar o vídeo que me ensinou SOLID há mais de 4 anos 🔥

Made by

Caetano **Martins**

OBRIGADO

POR CHEGAR ATÉ AQUI

Curta se você achou útil