IntelliSense

esse erro destrói o

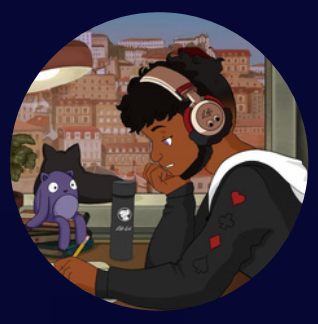
handleStatus.ts

```ts
export const handleStatus = (type: string): string => {}
```

handleStatus.ts

```ts
export const handleStatus = (type: StatusActions): StatusResult => {}
```

IntelliSense do seu typescript

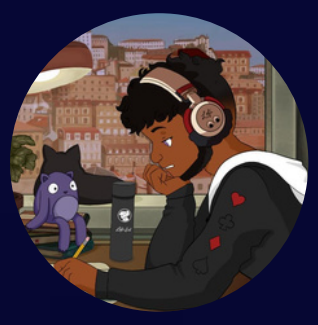


Isaac Gomes

TS

IntelliSense

quantas vezes mesmo com o uso do typescript ja não se deparou com tratativas que não faziam o menor sentido? pelo simples fato de estar mal tipado



Isaac Gomes

🧠 IntelliSense

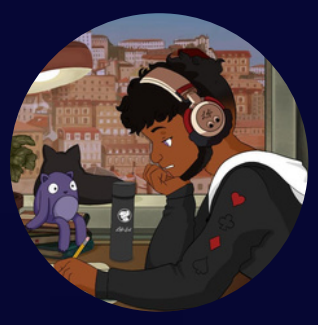
*vamos para um **exemplo***

*aqui temos uma função que recebe uma **string** e retorna uma **string***

handleStatus.ts

```ts
export const handleStatus = (type: string): string => {
  if (type === 'APPROVED') return 'aprovado'
  if (type === 'REJECTED') return 'reprovado'
  return 'ajuste'
}
```

*ta e qual é o **problema** dessa abordagem?*



Isaac Gomes

TS

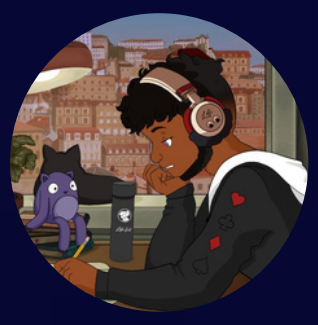
🧠 IntelliSense

o problema é que simplesmente temos as opções ja definidas

handleStatus.ts

```ts
const handleClick = () => {
  const resultStatus = handleStatus("a")

  if (resultStatus === "test_") {
    console.log("??????????")
  }

}
```

porem, isso não se reflete no IntelliSense do seu typescript e com isso permite comparações e usos sem sentidos

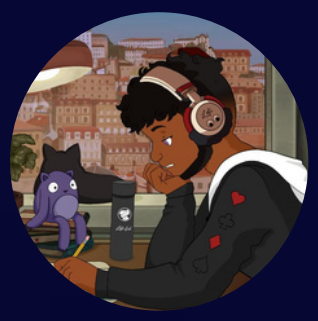


Isaac Gomes

TS

IntelliSense

ta e como seria um uso que traz o melhor do seu typescript ???



Isaac Gomes

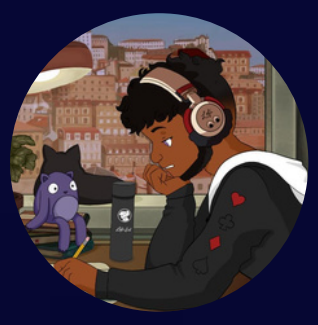
🧠 IntelliSense

1 - vamos usar um objeto imutável para simplificar a criação dos types

handleStatus.ts

```ts
const status = {
  APPROVED: 'aprovado',
  REJECTED: 'reprovado',
  ADJUSTED: 'ajuste',
} as const
```

pronto temos nossas opções



Isaac Gomes

TS

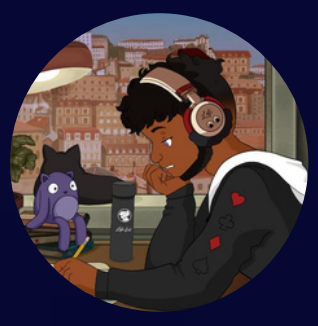
IntelliSense

2 - vamos criar um type baseado nas chaves do objeto

handleStatus.ts

```ts
type StatusActions = keyof typeof status
//type StatusActions = 'APPROVED' | 'REJECTED' | 'ADJUSTED'
```

para isso usaremos o "keyof typeof"



Isaac Gomes

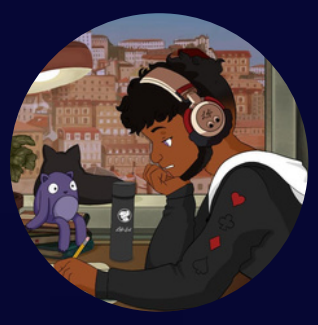
IntelliSense

3 - criar um type baseado nos valores do objeto

handleStatus.ts

```ts
type StatusActions = keyof typeof status
//type StatusActions = 'APPROVED' | 'REJECTED' | 'ADJUSTED'
```

para isso usaremos o "(typeof obj)[keyof typeof obj]"



Isaac Gomes

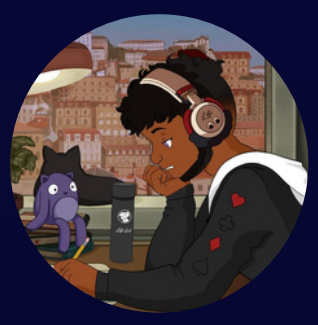
🧠 IntelliSense

4 - criamos uma função que recebe um type

handleStatus.ts

```ts
export const handleStatus = (type: StatusActions): StatusResult => {
  return status[type]
}
```

que corresponde a chave do objeto e que retorna um type do value do objeto



Isaac Gomes

TS

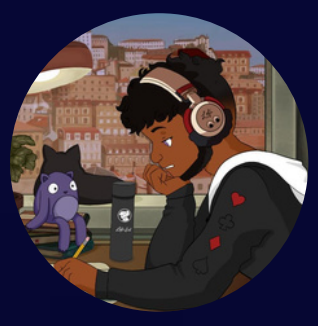
IntelliSense

ta mais o que isso muda?

```
export const handleClick = () => {
    const resultStatus = handleStatus('a')    Argument of type '"a"'
    //to parameter of type '"APPROVED" | "REJECTED" | "ADJUSTED"'.
}
```

```
export const handleClick = () => {
    const resultStatus = handleStatus("")    Argument of type '""' is not assignable
                                             ADJUSTED                    ADJUSTED
                                             APPROVED
                                             REJECTED
}
```

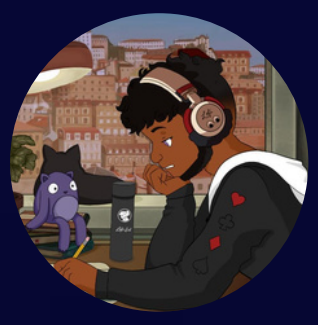
```
export const handleClick = () => {
    const resultStatus = handleStatus("ADJUSTED")

    if(resultStatus === "")    This comparison appears to be unintentiona
    }    Expression expected.  ajuste                    ajuste
                               aprovado
                               reprovado
```



Isaac Gomes

IntelliSense

*agora temos a **previsibilidade** tanto de saber o que vamos **passar** como parâmetro como oq vamos **receber** de retorno e com isso evitamos tratativas e usos sem sentido*



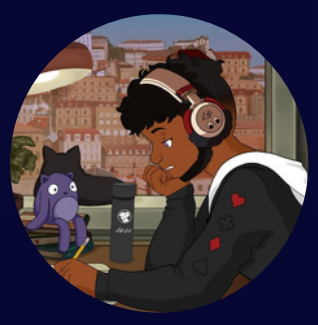
Isaac Gomes

IntelliSense

para evitar espalhar string soltar pelo seu código podemos usar um enum como chave de objeto

handleStatus.ts

```ts
enum EStatus {
  'APPROVED',
  'REJECTED',
  'ADJUSTED',
}

const status = {
  [EStatus.APPROVED]: 'aprovado',
  [EStatus.REJECTED]: 'reprovado',
  [EStatus.ADJUSTED]: 'ajuste',
} as const
```



Isaac Gomes

TS

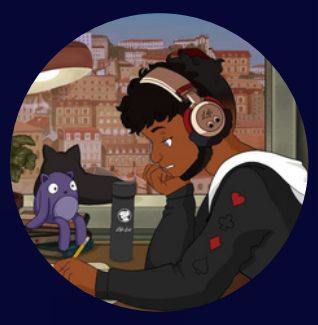
IntelliSense

*dessa maneira temos um **código***
que não contem

handleStatus.ts

```ts
export const handleClick = () => {
  const resultStatus = handleStatus(EStatus.ADJUSTED)
}
```

- *palavras magicas*
- *comparações sem sentido*
- *tratativas sem sentidos*



Isaac Gomes

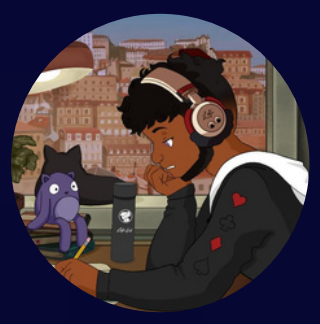
TS

Gostou?

Curta

Compartilhe

Salve



Isaac Gomes