




IPO Model

 2025-01   7 min.

O, en español,

Modelo EPS

Que es EPS?

ENTIDAD PROMOTORA DE SALUD - EPS

Entidades responsables de la afiliación y registro de los afiliados al sistema de la regularidad social en Colombia.

Que es el IPO Model?

Es un patron reconocido para el diseño y desarrollo de algoritmos.

IPO significa...

1. INPUT
2. PROCESS
3. OUTPUT

EPS significa...

1. ENTRADA
2. PROCESO
3. SALIDA

Es decir,

Para desarrollar un algoritmo que solucione un problema (pequeño),

1. Obtener los datos necesarios (input)
2. Procesar o modificar los datos (process)
3. Devolver un resultado (output)

Ejemplo

```
1  function suma(a, b) {  
2      const operando1 = a;  
3      const operando2 = b;  
4  
5      const resultado = operando1 + operando2;  
6  
7      return resultado;  
8  }
```


Ejemplo

```
1  function suma(a, b) {  
2    // const operando1 = a;  
3    // const operando2 = b;  
4  
5    const resultado = a + b;  
6  
7    return resultado;  
8  }
```

Ejemplo

```
1  function suma(a, b) {  
2    // const operando1 = a;  
3    // const operando2 = b;  
4  
5    // const resultado = a + b;  
6  
7    return a + b;  
8  }
```

Problema

Dada una lista de estudiantes y su nota final, cuantos estudiantes sacaron la mayor nota, que no es necesariamente la nota maxima posible (5)?

Problema

```
1  function suma(estudiantes) {  
2    // input  
3  
4  
5  
6    // process  
7  
8  
9  
10  
11    // output  
12  
13  }
```

Problema

```
1  function suma(estudiantes) {  
2    // input  
3    const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
4    const mayorNota = Math.max(...notasFinales);  
5  
6    // process  
7  
8  
9  
10  
11    // output  
12  
13  }
```

Problema

```
1  function suma(estudiantes) {  
2    // input  
3    const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
4    const mayorNota = Math.max(...notasFinales);  
5  
6    // process  
7    const estudiantesConMayorNota = estudiantes.filter(  
8      (estudiante) => estudiante.notaFinal === mayorNota  
9    );  
10  
11    // output  
12  
13  }
```

Problema

```
1  function suma(estudiantes) {  
2    // input  
3    const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
4    const mayorNota = Math.max(...notasFinales);  
5  
6    // process  
7    const estudiantesConMayorNota = estudiantes.filter(  
8      (estudiante) => estudiante.notaFinal === mayorNota  
9    );  
10  
11    // output  
12    return estudiantesConMayorNota.length;  
13  }
```

Problema

```
1  function suma(estudiantes) {  
2    // input  
3    const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
4    const mayorNota = Math.max(...notasFinales);  
5  
6    // process  
7    const estudiantesConMayorNota = estudiantes.filter(  
8      (estudiante) => estudiante.notaFinal === mayorNota  
9    );  
10  
11   // output  
12   return estudiantesConMayorNota.length;  
13 }
```


Y si las entradas son invalidas?

```
1  function suma(estudiantes) {  
2    // ??  
3    if (estudiantes.length === 0) {  
4      return 0;  
5    }  
6  
7    // input  
8    const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
9    const mayorNota = Math.max(...notasFinales);  
10  
11   // process  
12   const estudiantesConMayorNota = estudiantes.filter(  
13     (estudiante) => estudiante.notaFinal === mayorNota  
14   );  
15  
16   // output  
17   return estudiantesConMayorNota.length;  
18 }
```

Y si las entradas son invalidas?

```
1  function suma(estudiantes) {
2    // early return
3    if (estudiantes.length === 0) {
4      return 0;
5    }
6
7    // input
8    const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);
9    const mayorNota = Math.max(...notasFinales);
10
11   // process
12   const estudiantesConMayorNota = estudiantes.filter(
13     (estudiante) => estudiante.notaFinal === mayorNota
14   );
15
16   // output
17   return estudiantesConMayorNota.length;
18 }
```

Early Return

Los `else` son, hasta cierto punto, considerados "malos".

Porque?

1. Si la condición del `if` es complicadita, es complicado entender cuando cae en el `else` .
2. Si el código dentro del `if` es considerable, es fácil de olvidar cuál era la condición.

Alternativa?

```
1  function suma(estudiantes) {
2    if (estudiantes.length !== 0) {
3      // input
4      const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);
5      const mayorNota = Math.max(...notasFinales);
6
7      // process
8      const estudiantesConMayorNota = estudiantes.filter(
9        (estudiante) => estudiante.notaFinal === mayorNota
10     );
11
12     // output
13     return estudiantesConMayorNota.length;
14   } else {
15     return 0;
16   }
17 }
```

Y si son varias condiciones?

Terminamos con una flecha. (anti-patrón)

Flecha

```
1  function proceso(param1, param2) {  
2    if (isValid(argument1)) {  
3      if (isValid(argument2)) {  
4        const otherVal1 = doSomeStuff(param1, param2);  
5  
6        if (isValid(otherVal1)) {  
7          const otherVal2 = doAnotherStuff(otherVal1);  
8  
9          if (isValid(otherVal2)) {  
10           return "Stuff";  
11         } else {  
12           throw new Error();  
13         }  
14       } else {  
15         throw new Error();  
16       }  
17     } else {  
18       throw new Error();  
19     }  
20   } else {  
21     throw new Error();  
22   }  
23 }
```

```
1 public String returnStuff(SomeObject argument1, SomeObject argument2){
2     if (!argument1.isValid()) {
3         throw new Exception();
4     }
5
6     if (!argument2.isValid()) {
7         throw new Exception();
8     }
9
10    SomeObject otherVal1 = doSomeStuff(argument1, argument2);
11
12    if (!otherVal1.isValid()) {
13        throw new Exception();
14    }
15
16    SomeObject otherVal2 = doAnotherStuff(otherVal1);
17
18    if (!otherVal2.isValid()) {
19        throw new Exception();
20    }
21
22    return "Stuff";
23 }
```

Early Return

```
1  function suma(estudiantes) {  
2    if (estudiantes.length === 0) {  
3      return 0;  
4    }  
5  
6    const notasFinales = estudiantes.map((estudiante) => estudiante.notaFinal);  
7    const mayorNota = Math.max(...notasFinales);  
8  
9    const estudiantesConMayorNota = estudiantes.filter(  
10     (estudiante) => estudiante.notaFinal === mayorNota  
11   );  
12  
13   return estudiantesConMayorNota.length;  
14 }
```


Fin