

Dart en Flutter

Clases y Patrones de diseño



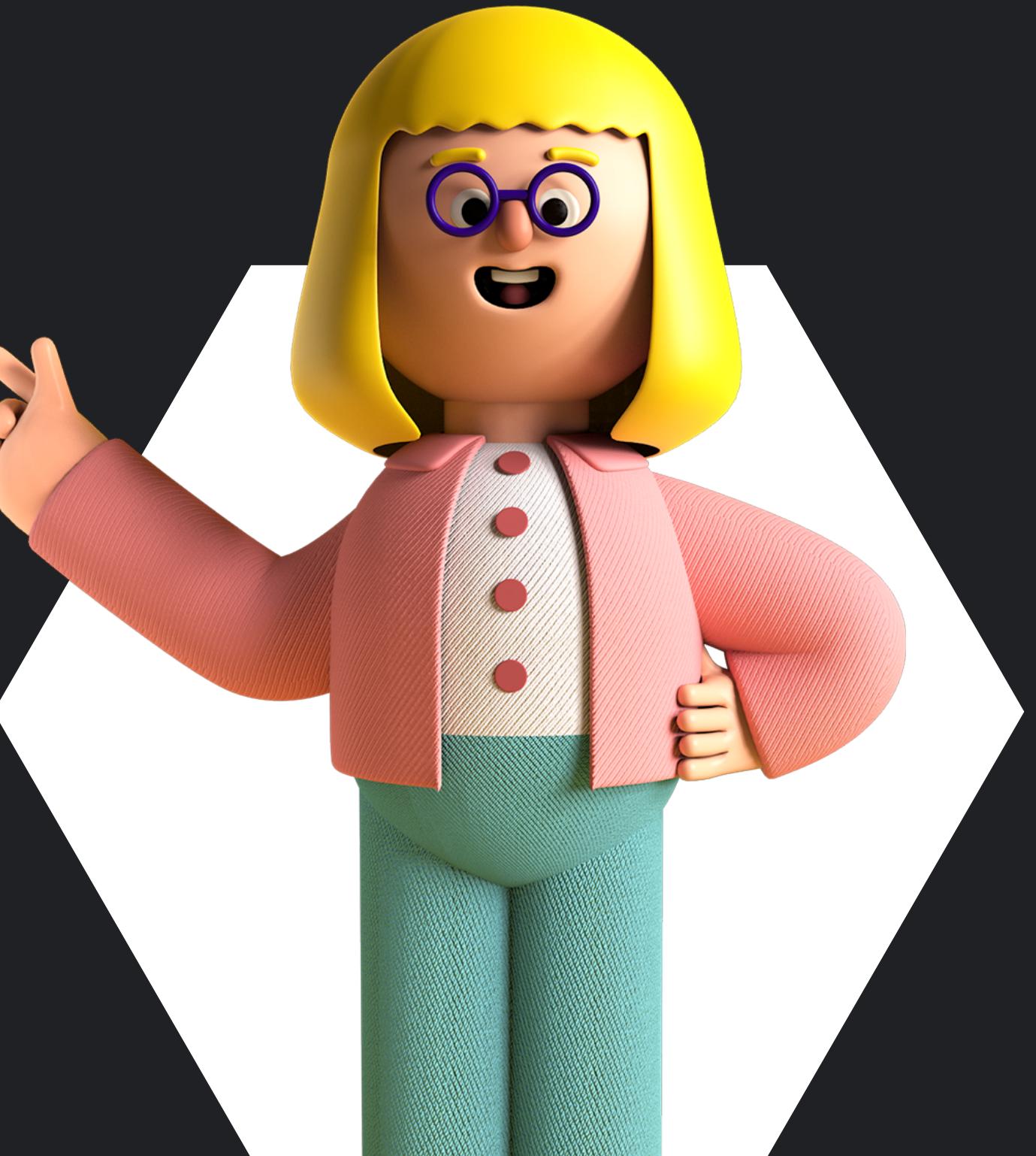
Yosafat Coronel
GDSC ESCOM IPN
GitHub: YosafatM



Developer Student Clubs
ESCOM - IPN

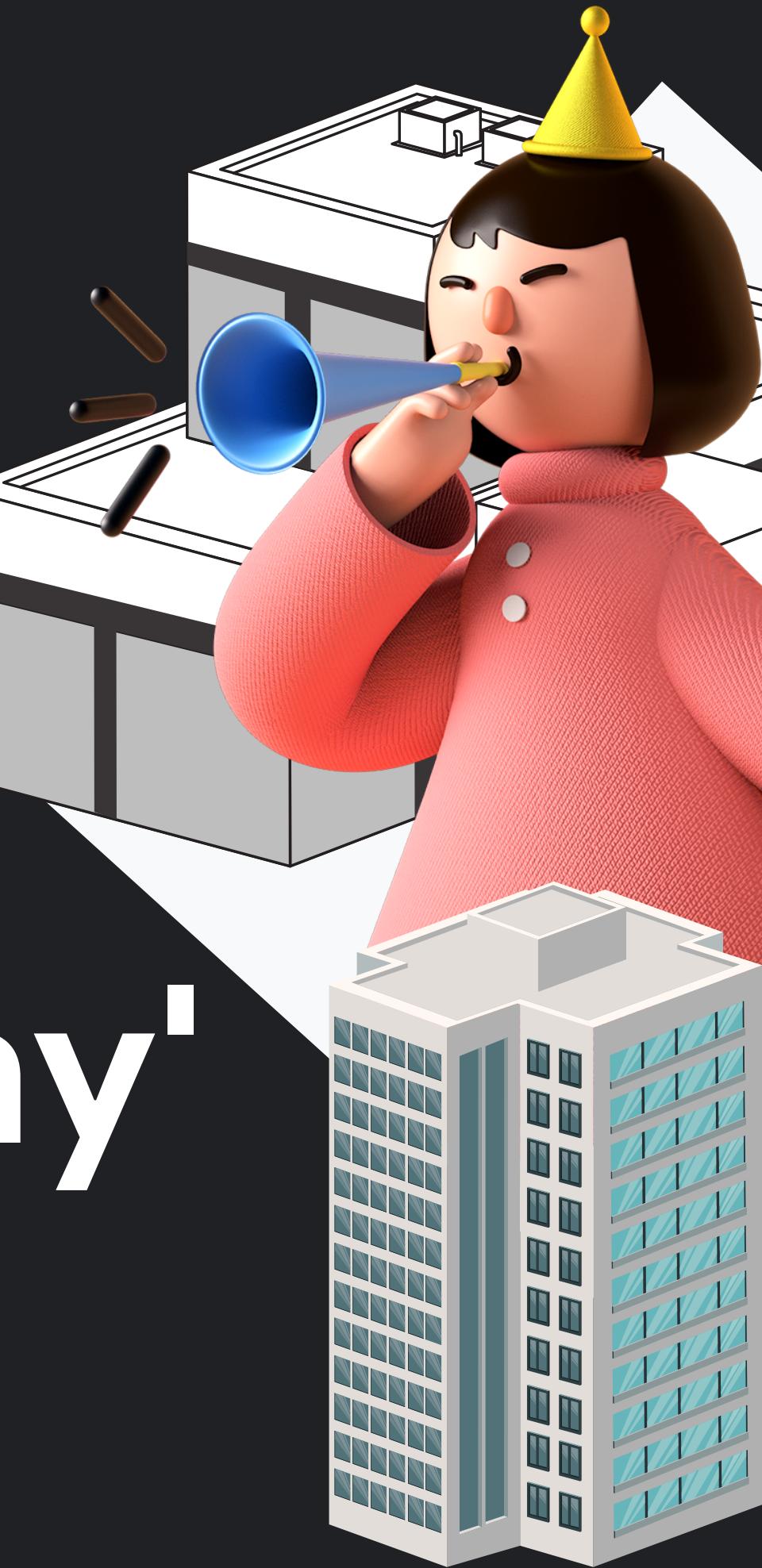
Itinerario

- Constructores
 - Parámetros
 - Nombrados
 - La palabra this
- Reglas de constructores
 - Lista de asignaciones
 - Cuerpo y punto y coma
- La aplicación contador



Constructores

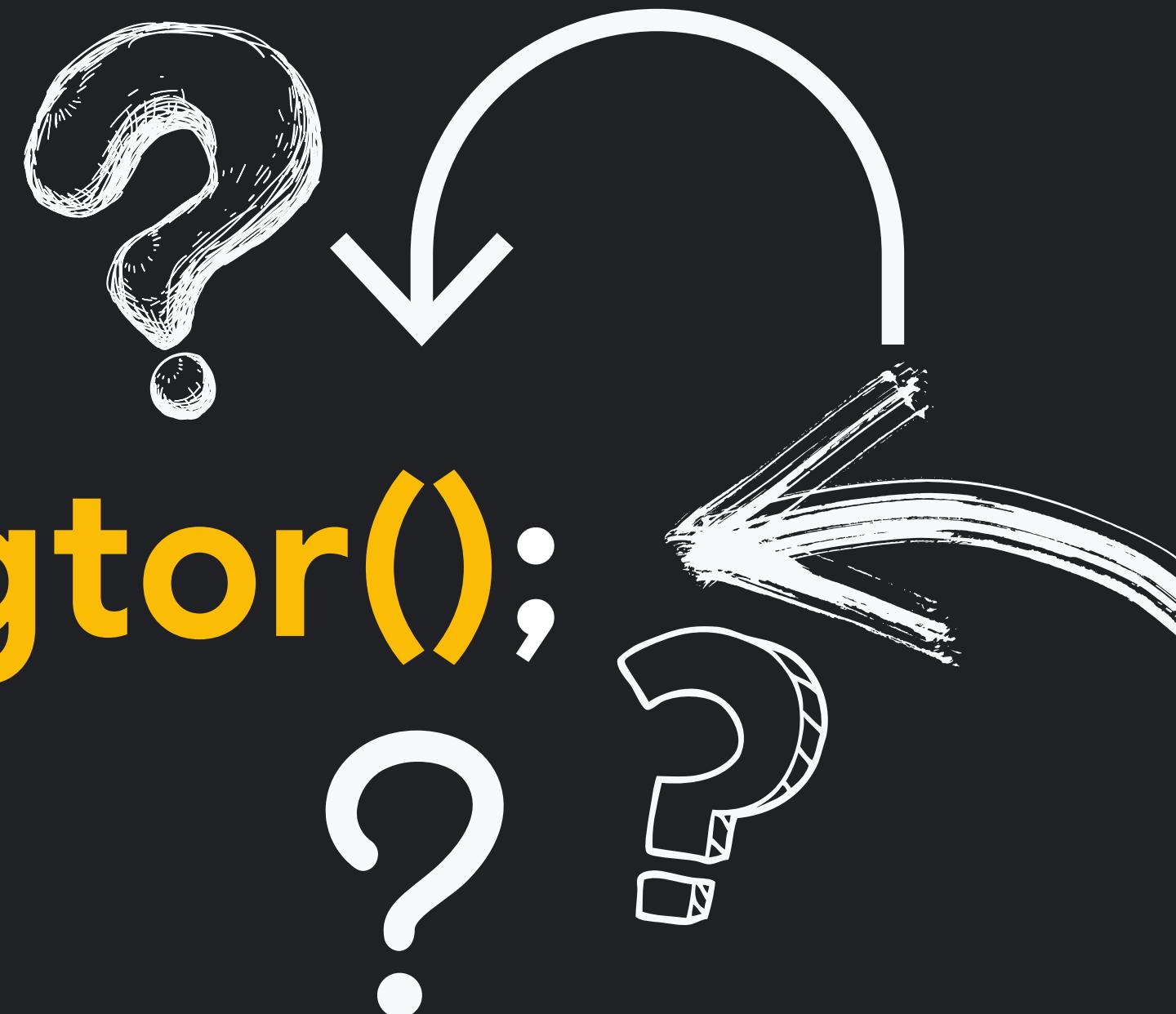
```
Dogtor({  
    laConfiable: 'No hay'  
});
```



Construtor

¿Qué pasa cuando nosotros creamos un nuevo Dogtor? ¿De dónde sale eso que parece un método y lleva el nombre de la clase?

Dogtor who = **Dogtor()**;



Constructor

Dogtor();

Es parecido a un método, pero se le llama constructor, además, podemos cambiarlo, pero si no lo hacemos, Dart por dentro pone uno vacío

Hay muchas maneras de poner el constructor vacío, esta solo usa una línea. Notar el punto y coma

Constructor

¿Qué hace?, nos permite pasarle argumentos y definirlos justo como una función/método, es útil para tener propiedades con valores iniciales:

```
Dogtor(String med){
```

```
    laConfiable = med;
```

```
}
```

Ahora se necesita un String y se pondrá en laConfiable del Dogtor: Dogtor('Nada');

Constructor

Hay muchas formas de declararlo, pero solo podemos tener uno con el mismo nombre (sin embargo podemos nombrarlos):

```
Dogtor(this.laConfiable);
```

```
Dogtor();
```

No podemos tener más de un constructor con el mismo identificador

Reglas

([params]):

[<expr>, ...]

{[cuerpo]} | ;



Reglas

Los constructores tienen las siguientes formas de ser nombrados:
NombreClase([parámetros]) o bien
NombreClase.nombre([parámetros]):

Dogtor(**this.laConfiable**);

Dogtor.**sinConfiable()**;

*Ahora podemos hacer Dogtores con
Dogtor(String) y con Dogtor.sinConfiable()*

Reglas

Si el constructor no tiene cuerpo (o sea, un bloque de statements entre llaves) entonces usamos punto y coma al final:

Dogtor(**this.laConfiable**);

Dogtor.**sinConfiable()**;

No tienen cuerpo, usamos ;

Reglas

Si el parámetro tiene el mismo nombre que la propiedad que vamos a asignar, entonces usamos this (explicado más tarde):

```
Dogtor(this.laConfiable);  
Car({required this.modelo});
```

Puede ser por parámetros posicionales o por nombrados. Si no puede ser null, se usa required

Reglas

Si necesitamos solamente expresiones y asignaciones, entonces usamos la lista de asignación con :, como aquí:

Punto(this.x) :

y = 2*x,

z = y+1;

Se ejecuta antes que el cuerpo del constructor

*Sigue sin tener cuerpo,
(statements entre llaves)
por eso lleva ;*

Reglas

Si necesitamos statements, entonces ahí sí se usa el cuerpo del constructor y deja de llevar ;, como aquí:

```
Punto(this.x) {  
    y = 2*x;  
}
```

Se ejecuta antes que el cuerpo del constructor

*Sigue sin tener cuerpo,
(statements entre llaves)
por eso lleva ;*

this

La palabra reservada this, hace referencia al marco de trabajo del objeto. Puede ser usada en cualquier método, y apuntará directo a ese marco en específico

```
Punto(int x) {  
    this.x = x;  
}
```

Esto es porque si solo ponemos x = x, va a tomar la más cercana a su marco

Ejemplo

A la derecha vemos un (porque hay varios) constructor para hacer un botón de texto.

Notemos que no tiene cuerpo, solo la lista de asignaciones y los parámetros.

```
const TextButton({  
  Key? key,  
  required VoidCallback? onPressed,  
  VoidCallback? onLongPress,  
  ButtonStyle? style,  
  FocusNode? focusNode,  
  bool autofocus = false,  
  Clip clipBehavior = Clip.none,  
  required Widget child,  
}) : super(  
  key: key,  
  onPressed: onPressed,  
  onLongPress: onLongPress,  
  style: style,  
  focusNode: focusNode,  
  autofocus: autofocus,  
  clipBehavior: clipBehavior,  
  child: child,  
);
```

Ejemplo

A la derecha vemos el constructor para ponerle estilo a un botón como el de la diapositiva anterior.

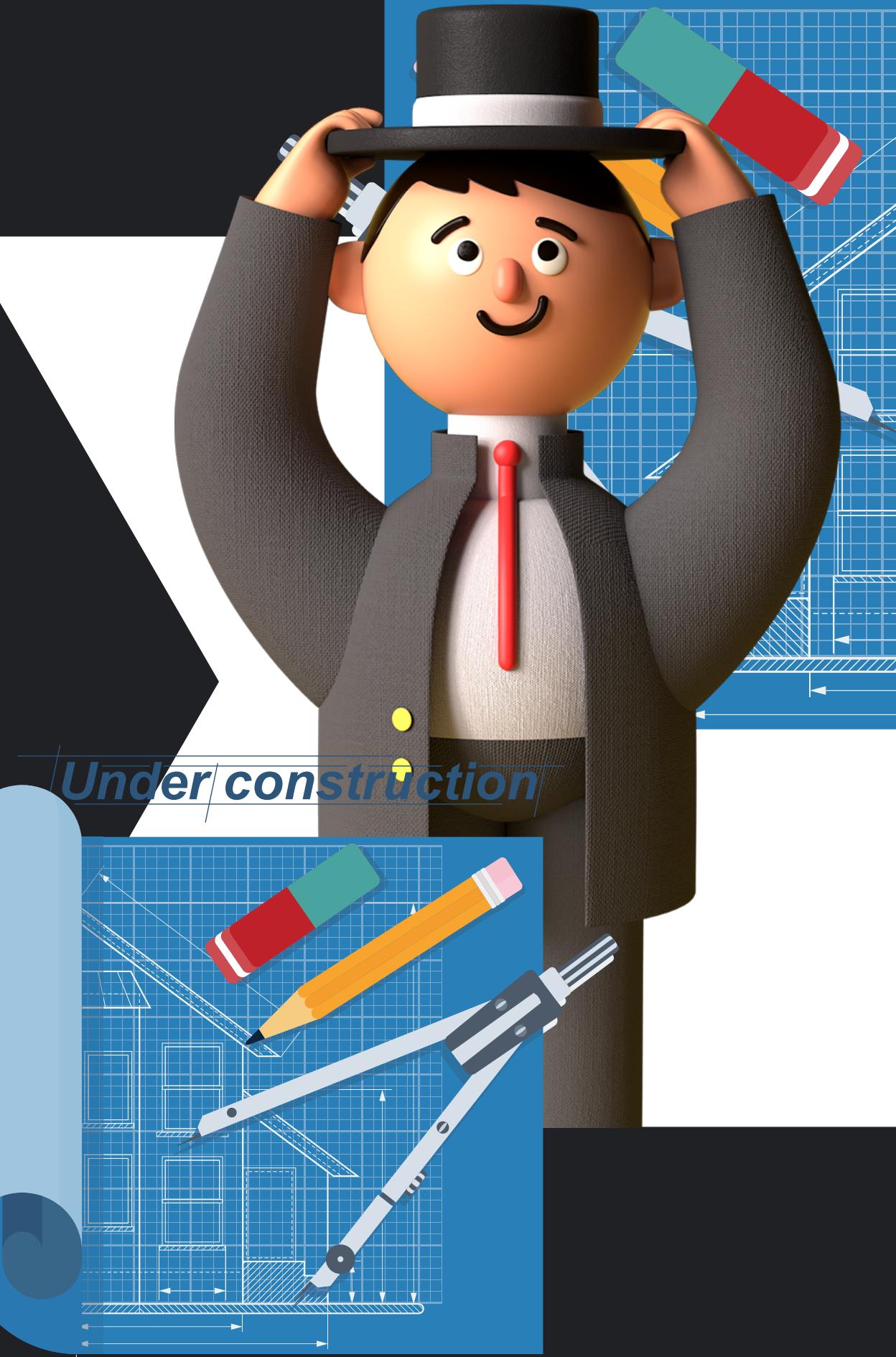
Hay que notar que en Flutter, se usan mucho los parámetros nombrados

```
const ButtonStyle({  
    this.textStyle,  
    this.backgroundColor,  
    this.foregroundColor,  
    this.overlayColor,  
    this.shadowColor,  
    this.elevation,  
    this.padding,  
    this.minimumSize,  
    this.fixedSize,  
    this.maximumSize,  
    this.side,  
    this.shape,  
    this.mouseCursor,  
    this.visualDensity,  
    this.tapTargetSize,  
    this.animationDuration,  
    this.enableFeedback,  
    this.alignment,  
    this.splashFactory,  
});
```

Flutter

```
setState(() {  
    _contador++;  
});
```

(Como introducción)

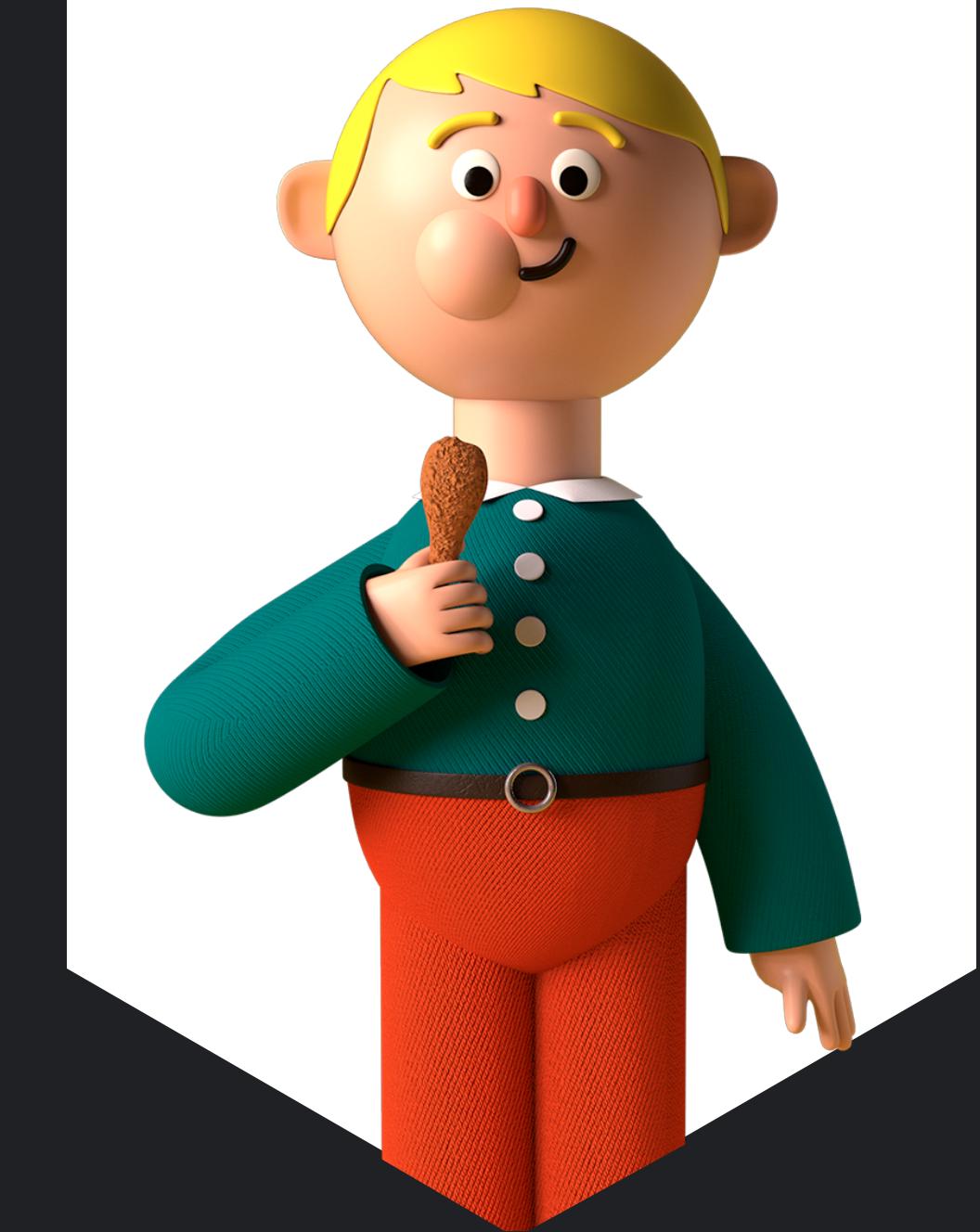


Contar

Poco a poco iremos viendo la primera aplicación (que sería como su Hola mundo) del contador, podemos verlo desde el dartpad:

dartpad.dev

**¡Muchísimas
gracias!**



Developer Student Clubs
ESCOM - IPN