

# ***Programación Avanzada***

## **IIC2233 2025-2**

Cristian Ruz - Pablo Araneda - Francisca Ibarra - Tamara Vidal - Daniela Concha



# Experiencia 3

Interfaces Gráficas I



# Experiencia: ¿Qué vamos a hacer?

1. Aplicaremos el patrón de diseño: ***front-end/back-end***.
2. Crearemos una aplicación de múltiples componentes que interactúan entre ellos mediante el **uso de señales**.
3. Posicionaremos elemento en una ventana a través de:
  - a. **Coordenadas**
  - b. *Layouts*

# DCChannels



Para estudiar programación avanzada  
te propones crear tu propio  
dispositivo para disfrutar de las series  
y películas...

Un **televisor a control remoto.**

---

# ¿Cómo lo lograremos?

Programaremos una televisión, la cual estará compuesta por tres componentes:



Pantalla



Control  
remoto



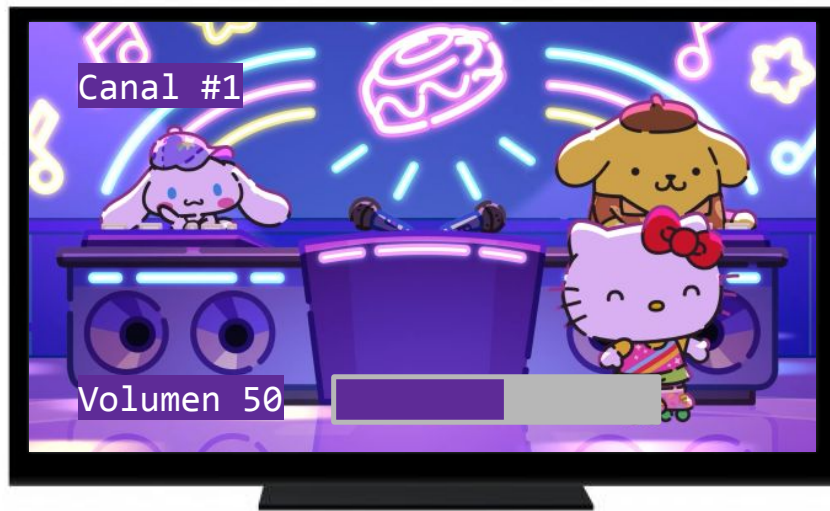
Controlador  
lógico

# Pantalla

Está compuesto por:

- 2 *Labels* para los textos (canal y volumen)
- 1 *Label* para la imagen de fondo
- 1 *ProgressBar* para mostrar el volumen.

Todos estos elementos serán posicionados de **forma manual** a través de **coordenadas**.



QProgressBar

# Control remoto

Está compuesto por:

- Múltiples botones (*PushButton*).
- 2 *Labels* para los textos de ciertos conjuntos de botones (canal y volumen)

Todos estos elementos serán posicionados mediante ***layouts***.



# Controlador lógico

- Se encarga de procesar toda la información lógica del programa.
  - Encender y apagar la tele.
  - Cambiar de canal.
  - Cambiar el volumen.
- Ayudan a mantener ciertos estados y su actualización.
- Permite la comunicación entre los distintos componentes visuales.

**Controlador  
encargado  
de la lógica**



# ¿Cómo lo lograremos?

Estos componentes se encuentran incompletos o presentan errores, por lo que deberemos **completar o corregir los métodos** de cada componente:












- a. Pantalla
- b. Control remoto
- c. Controlador lógico

Además, deberemos solucionar problemas de señales en el archivo **main.py**.
















# ¿Qué tenemos?

## VentanaPantalla













*Sin señales*





-  posicion: tuple(int)
-  porte: tuple(int)
-  imagen: QLabel
-  canal: QLabel
-  volumen: QLabel
-  volumen\_barra: QProgressBar
-  inicializar\_gui()
-  generar\_widgets()
-  agregar\_estilo()
-  actualizar\_volumen(volumen: int)
-  actualizar\_canal(canal: int)

## VentanaControlRemoto

-  senal\_volumen: str
-  senal\_canal: str
-  senal\_encendido: null
-  volumen: list(QPushButton)
-  canales: list(QPushButton)
-  numeros: list(QPushButton)
-  inicializar\_gui()
-  generar\_botones()
-  generar\_layout()
-  generar\_layout\_subir\_bajar(  
    botones: list, texto: str  
)
-  generar\_layout\_numeros()
-  agregar\_estilo()
-  conectar\_botones()
-  actualizar\_canal()
-  actualizar\_volumen()

## ControladorLogico

-  senal\_volumen: int
-  senal\_canal: int
-  senal\_encendido: bool
-  senal\_empezar: null
-  volumen: int (property)
-  canal: int (property)
-  prendido: bool
-  cambiar\_volumen(cambio: str)
-  cambiar\_canal(cambio: str)
-  actualizar\_volumen()
-  actualizar\_canal()
-  prender\_apagar()
-  empezar()

-  Señal
-  Atributo
-  Método
-  Incompleto o no implementado

# Parte 1: Completar componentes faltantes

## Control remoto

Gran medida de sus *widgets* se encuentran definidos, pero **falta definir** el **botón ON/OFF** y los **botones de los números**.

Para esto, completa y corrige el método `generar_botones()`.

¡A programar! 🖥️



# Parte 1: Completar componentes faltantes

## Control remoto

Además, **falta posicionar los botones** para subir/bajar el canal y el volumen, **junto con el texto correspondiente.**

Deberás completar el método `generar_layout()`.

¡A programar! 



# Experiencia

Esto es lo que tenemos hasta el momento:



Control  
remoto



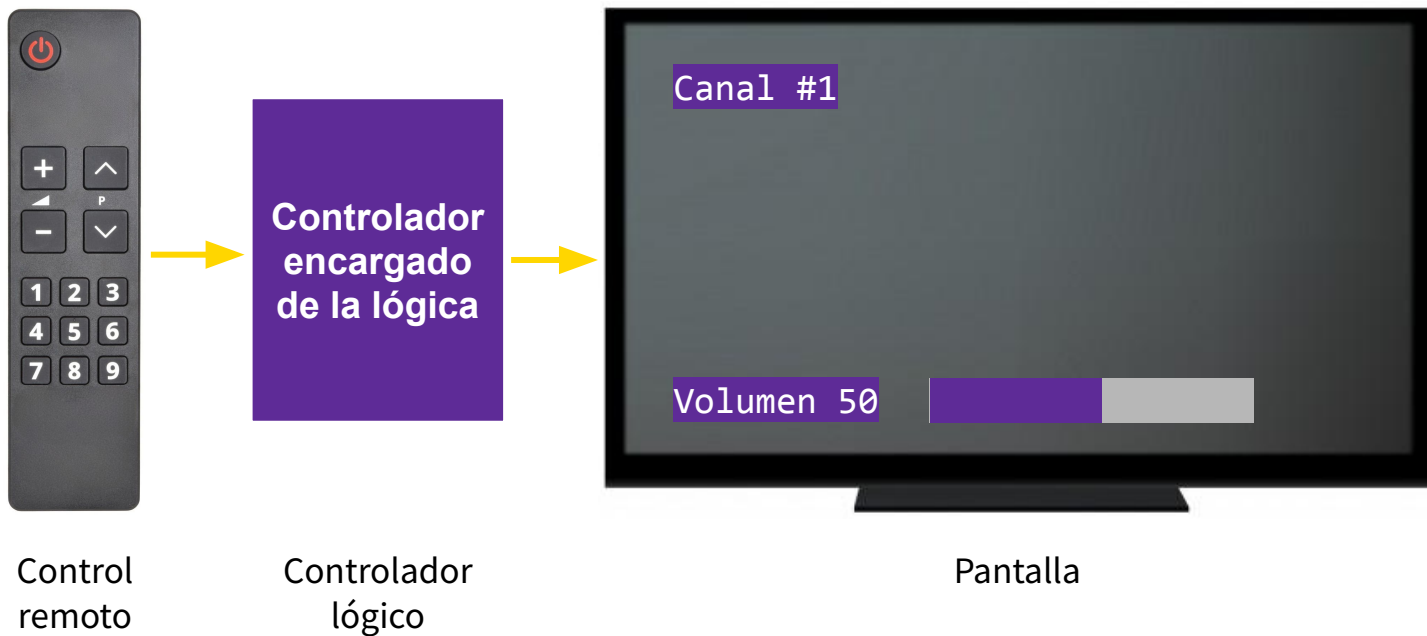
Controlador  
lógico



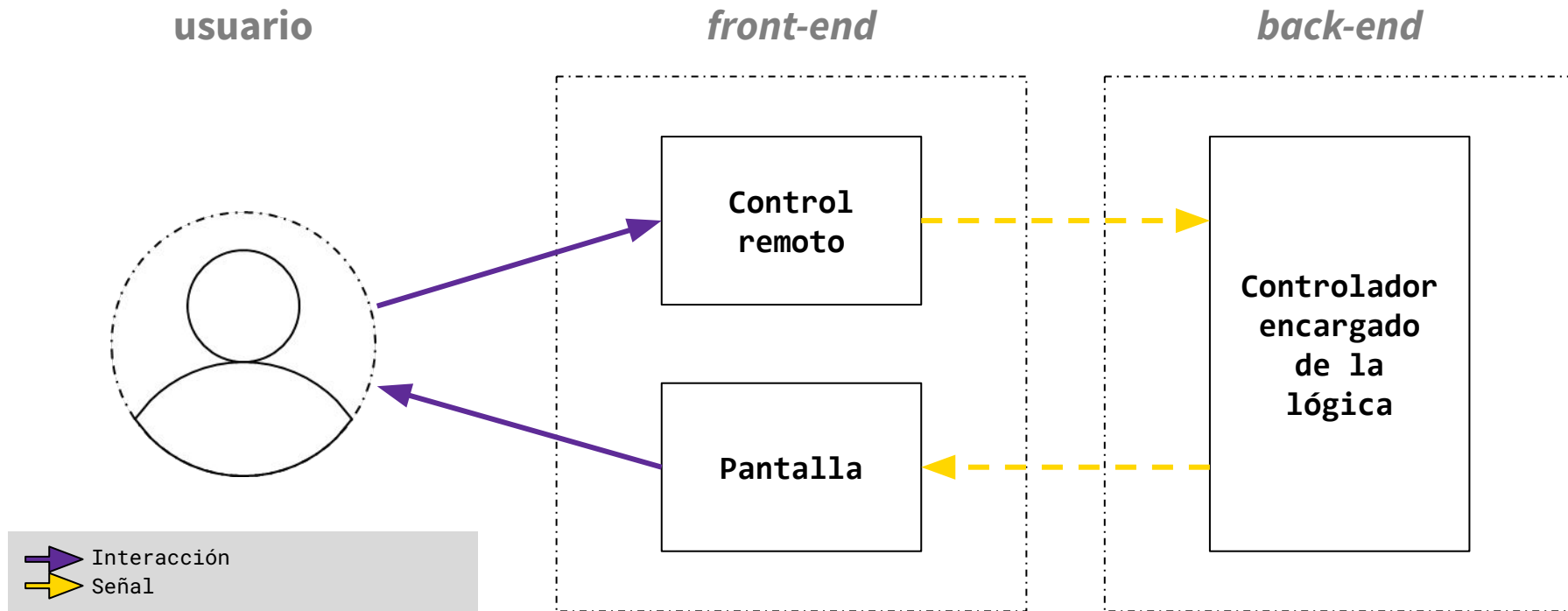
Pantalla

# Experiencia

Falta permitir que los componentes se comuniquen entre ellos:



# Experiencia: Diagrama de modelación



# Parte 2: Enviar y manejar eventos

Los componentes visuales y lógicos ya se encuentran completos, solo falta:

1. **Enviar señales** para cambiar el volumen, canal y prender/apagar el televisor desde el **control remoto**:

- a. `actualizar_canal()`

- b. `actualizar_volumen()`

Luego, descomentar el método `conectar_botones()`.

**¡A programar!** 



# Parte 2: Enviar y manejar eventos

Los componentes visuales y lógicos ya se encuentran completos, solo falta:







2. **Recibir señales** para cambiar el volumen y canal del televisor en la **pantalla** y mostrar el cambio.
  - a. `actualizar_volumen(nuevo_volumen)`
  - b. `actualizar_canal(nuevo_canal)`

**¡A programar!** 










# Parte 2: Enviar y manejar eventos

Los componentes visuales y lógicos ya se encuentran completos, pero presentan problemas. Ejecuta el archivo **main.py**, encuentra los errores y corrígelos.

VentanaControlRemoto

 <code>senal_volumen: str</code>	→	 <code>cambiar_volumen(cambio: str)</code>
 <code>senal_canal: str</code>	→	 <code>cambiar_canal(cambio: str)</code>
 <code>senal_encendido: null</code>	→	 <code>prender_apagar()</code>

ControladorLogico

 <code>show()</code>	←	 <code>senal_volumen: int</code>	→	 <code>actualizar_volumen(volumen: int)</code>
		 <code>senal_canal: int</code>	→	 <code>actualizar_canal(canal: int)</code>
		 <code>senal_encendido: bool</code>	→	 <code>prender_apagar(encendido: bool)</code>
		 <code>senal_empezar: null</code>	→	 <code>show()</code>

VentanaPantalla

¡A programar! 

# Desafíos

Se pueden lograr con los **contenidos vistos hasta ahora**:

- Cambiar canal utilizando las teclas del teclado.
- Agregar un botón para “mutear”. Se debe recordar el volumen anterior.
- Agregar un ícono al botón “On/Off” y hacer que sea circular.

Necesitamos de los contenidos que se verán en **Interfaces Gráficas 2**:

- Hacer *zapping*, apretando solo un botón.
- Agregar un protector de pantalla animado que se active después de cierto tiempo.
- Recibir canales de 2 dígitos o más, a través de los 9 botones numéricos.

# ***Programación Avanzada***

## **IIC2233 2025-2**

Cristian Ruz - Pablo Araneda - Francisca Ibarra - Tamara Vidal - Daniela Concha

