Programación Avanzada
IIC2233 2025-2

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

Anuncios

6 de noviembre de 2025

1. Hoy tendremos la quinta Experiencia.
2. Recuerden que la ECA abre de domingo a martes. Si responden 10 veces tienen un bonus en el promedio final.

Más Interfaces Gráficas

ahora en movimiento
y conectadas 🤭

Concurrencia en PyQt

- ¿Por qué es importante usar `QThread` en una aplicación con una interfaz gráfica (PyQt)?
- ¿Qué sucede si ejecutamos una tarea pesada directamente en el *thread* principal de una aplicación PyQt?
- ¿Es siempre necesario usar *threads* en una interfaz (GUI)? ¿Cuándo podría ser una mala idea?
- ¿Cómo detenemos de forma segura un `QThread` en ejecución?

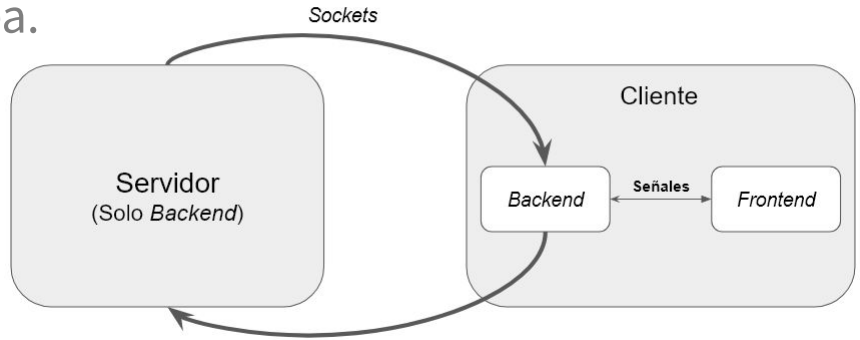
Concurrencia en PyQt

`QThread vs Thread`

- Ambos permiten aplicar concurrencia en PyQt.
- Thread no puede contener señales como atributo de clase y QThread si.
- Ambos pueden tener señales como atributos de instancia.
- Queda a tu criterio cuál de los 2 usar de aquí en adelante, puedes utilizar únicamente uno o intentar usar ambos. **¡Tú eliges 😁 !**
- ***Spoiler*:** aunque elijas usar únicamente uno, en el examen podemos preguntar por cualquiera de los 2 o ambos 🙈 .

Interfaces gráficas y operaciones de red

- ¿Para quién es la interfaz gráfica? ¿Cliente o Servidor?
- ¿Por qué es importante mantener separadas las operaciones de red del *thread* principal de una interfaz gráfica?
    - ¿Qué consecuencias visuales y funcionales puede tener no hacerlo?
    - Piensa en un *feedback* visual para mejorar la experiencia del usuario cuando una tarea no es instantánea.



Integración de *threads* y operaciones de red

- ¿Por qué es problemático hacer una solicitud HTTP (como **`requests.get`**) directamente en el hilo principal de una interfaz?
- ¿Qué pasaría si una respuesta de red tarda 10 segundos y no se usa un **`QThread`**? ¿Cómo se vería afectada la interfaz para el cliente?
- ¿Qué diferencia hay entre usar **`QThread`** y usar **requests** de forma síncrona en una función **clicked.connect(...)**?

🌐🤝 Envío de datos, cliente-servidor, threads

Vamos a ponernos en distintas situaciones, a ver cómo las resolvemos:

1. Tienes que recibir datos desde un socket y luego mostrarlos en un **`QLabel`** en la interfaz. ¿Qué deberías usar para evitar errores por actualizar la interfaz desde un thread secundario?

🌐🤝 Envío de datos, cliente-servidor, threads

Vamos a ponernos en distintas situaciones, a ver cómo las resolvemos:

1. Tienes que recibir datos desde un socket y luego mostrarlos en un **`QLabel`** en la interfaz. ¿Qué deberías usar para evitar errores por actualizar la interfaz desde un thread secundario?

```
class SocketListener(QThread):
    actualizar_texto = pyqtSignal(str)

    def run(self):
        data = self.socket.recv(1024).decode()
        self.actualizar_texto.emit(data)  # ¡aquí va la señal!
```

en las conexiones entre *back-end* y *front-end*:

```
self.listener.actualizar_texto.connect(self.label.setText)
```

🌐🤝 Envío de datos, cliente-servidor, threads

Vamos a ponernos en distintas situaciones, a ver cómo las resolvemos:

2. Si necesitas ejecutar una función después de un retraso sin bloquear la interfaz gráfica, ¿qué función o clase de Python o PyQt deberías usar?

🌐🤝 Envío de datos, cliente-servidor, threads

Vamos a ponernos en distintas situaciones, a ver cómo las resolvemos:

2. Si necesitas ejecutar una función después de un retraso sin bloquear la interfaz gráfica, ¿qué función o clase de Python o PyQt deberías usar?

```
class EsperaLenta(QThread):
    finalizado = pyqtSignal(str)

    def run(self):
        time.sleep(2)
        self.finalizado.emit("Tarea completada")
```

en la interfaz:

```
self.tarea = EsperaLenta()
self.tarea.finalizado.connect(self.label.setText)
self.tarea.start()
```

Se usa cuando el retraso es parte de una tarea de más larga duración.

La interfaz no se congela porque el **sleep** ocurre en un hilo secundario.

🌐🤝 Envío de datos, cliente-servidor, threads

Vamos a ponernos en distintas situaciones, a ver cómo las resolvemos:

2. Si necesitas ejecutar una función después de un retraso sin bloquear la interfaz gráfica, ¿qué función o clase de Python o PyQt deberías usar?

```
QTimer.singleShot(2000, lambda: self.label.setText("Tarea rápida completada"))
```

Ideal cuando solo queremos ejecutar algo **una vez** después de un tiempo, sin abrir un thread nuevo.

Veamos una pregunta de Evaluación Escrita

(Examen 2024-2)

15. ¿Cuál de los siguientes casos debe ser programado exclusivamente con `Thread` o con `QThread`, pero no con ambos?

    A) Un servidor que permite la conexión de múltiples clientes simultáneamente.

    B) Una interfaz para ver el progreso de descarga de múltiples archivos.

    C) Un juego donde hay que disparar meteoritos que caen del cielo.

    D) Una ventana que se mueve de forma aleatoria por la pantalla.

    E) Ninguna de los anteriores.

Veamos una pregunta de Evaluación Escrita

(Examen 2024-2)

15. ¿Cuál de los siguientes casos debe ser programado exclusivamente con `Thread` o con `QThread`, pero no con ambos?

    A) Un servidor que permite la conexión de múltiples clientes simultáneamente.

    B) Una interfaz para ver el progreso de descarga de múltiples archivos.

    C) Un juego donde hay que disparar meteoritos que caen del cielo.

    D) Una ventana que se mueve de forma aleatoria por la pantalla.

    E) Ninguna de los anteriores.

Programación Avanzada
IIC2233 2025-2

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