# *Programación Avanzada*
# IIC2233 2023-2

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Joaquín Tagle - Francisca Cattan

# Anuncios

1. Hoy tenemos la tercera experiencia.

2. Encuesta de Carga Académica y Evaluación Temprana de Cursos. ¡Respóndanlas!

# Interfaces gráficas II

# Concurrencia en PyQt

- QThread
- QTimer

# Concurrencia en PyQt

**QThread**

- Es un Thread, como el visto la semana pasada, pero dentro de PyQt.
- Se utiliza igual que un Thread, es decir, definir un método run y hacer `.start()` del QThread.
- También disponemos de **QMutex**, el análogo al Lock en PyQt. Este nos servirá para asegurar el acceso a un recurso crítico.

# Concurrencia en PyQt

**QTimer**

- Otra herramienta para generar concurrencia (al igual que el QThread).
- Por defecto, se define un tiempo N y cada N milisegundos, se ejecuta la función/método objetivo.
- Podemos hacer `.stop()` para detener su siguiente ejecución o usar nuevamente `.setInterval(N)` para cambiar el tiempo entre intervalos.
- Podemos hacer `.setSingleShot(True)` para que el QTimer se ejecute solo 1 vez luego de los N milisegundos indicados.

# Concurrencia en PyQt

**QTimer vs QThread**

- Ambos son formas distintas de aplicar concurrencia en PyQt.
- Con un poco de creatividad, ambas clases pueden lograr el mismo efecto, solo que con código distinto.
- Como ambas son clases, podemos heredar de ellas para crear nuestro objeto más personalizado.
- Queda a tu criterio cuál de los 2 usar de aquí en adelante, puedes utilizar únicamente uno o intentar usar ambos. **¡Tú eliges 😁!**

# Sonidos

- QMediaPlayer
- QSoundEffect

# Sonidos

## QSoundEffect

- Objeto para reproducir archivos .wav.

```
self.media_player_wav = QSoundEffect(self) # Creo reproductor
file_url = QUrl.fromLocalFile(join("sounds", "musica.wav")) # Creo un path
media_player_mp3.setSource(file_url) # Indico el path al reproductor
media_player_mp3.play() # Reproducir
```

# Sonidos

## QMediaPlayer

- Objeto para reproducir archivos .mp3.
- Puede salir un *warning* en consola. No te preocupes 😁

```
self.media_player = QMediaPlayer(self) # Creo reproductor
self.media_player.setAudioOutput(QAudioOutput(self)) # Indico que será modo audio
file_url = QUrl.fromLocalFile(join("sounds", "musica.mp3")) # Creo un path
self.media_player.setSource(file_url) # Indico el path al reproductor
self.media_player.play() # Reproducir
```

# Main Window

# Main Window

Ventana más completa

- Contiene un Widget central.
- Incluye barra de menú, herramienta y de estado.

# Main Window

```
# Creamos una acción y conectamos con un método
saludar_action = QAction(QIcon(None), "Saludar", self)
saludar_action.triggered.connect(self.vamos_a_saludar)
```

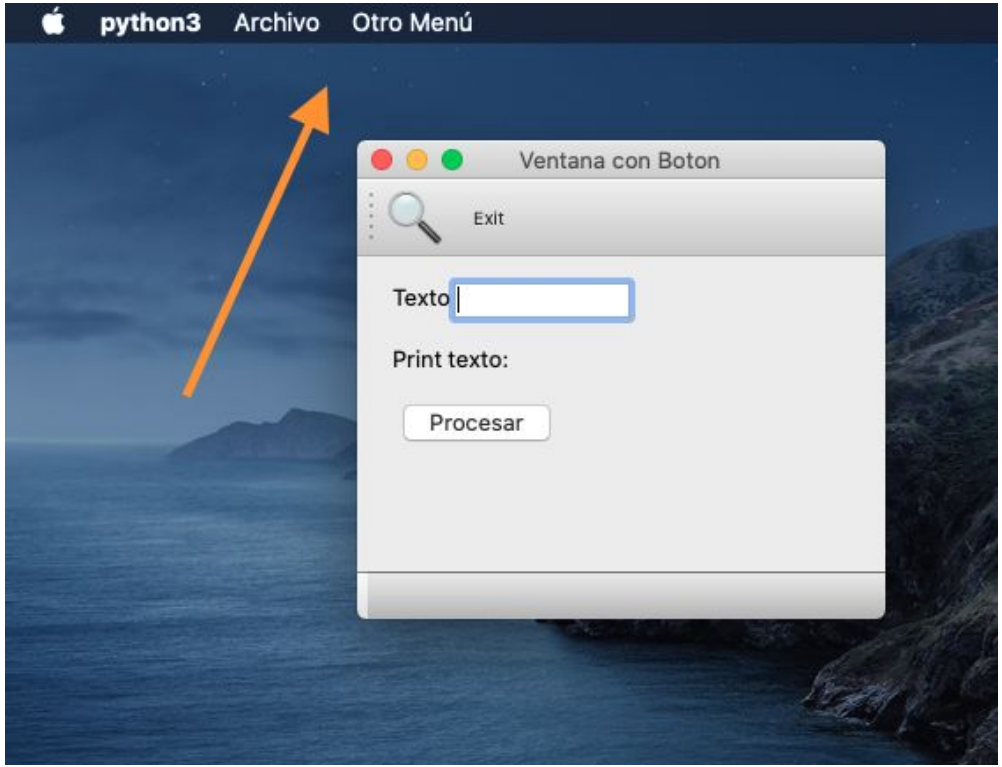

```
# Agregamos acción a la barra de menú
menubar = self.menuBar()
archivo_menu = menubar.addMenu("Archivo")
archivo_menu.addAction(saludar_action)

# Agregamos acción a la barra de herramientas
toolbar = self.addToolBar("Toolbar")
toolbar.addAction(saludar_action)

# Editamos la barra de estado
self.statusBar().showMessage("Todavía no saludo")
```

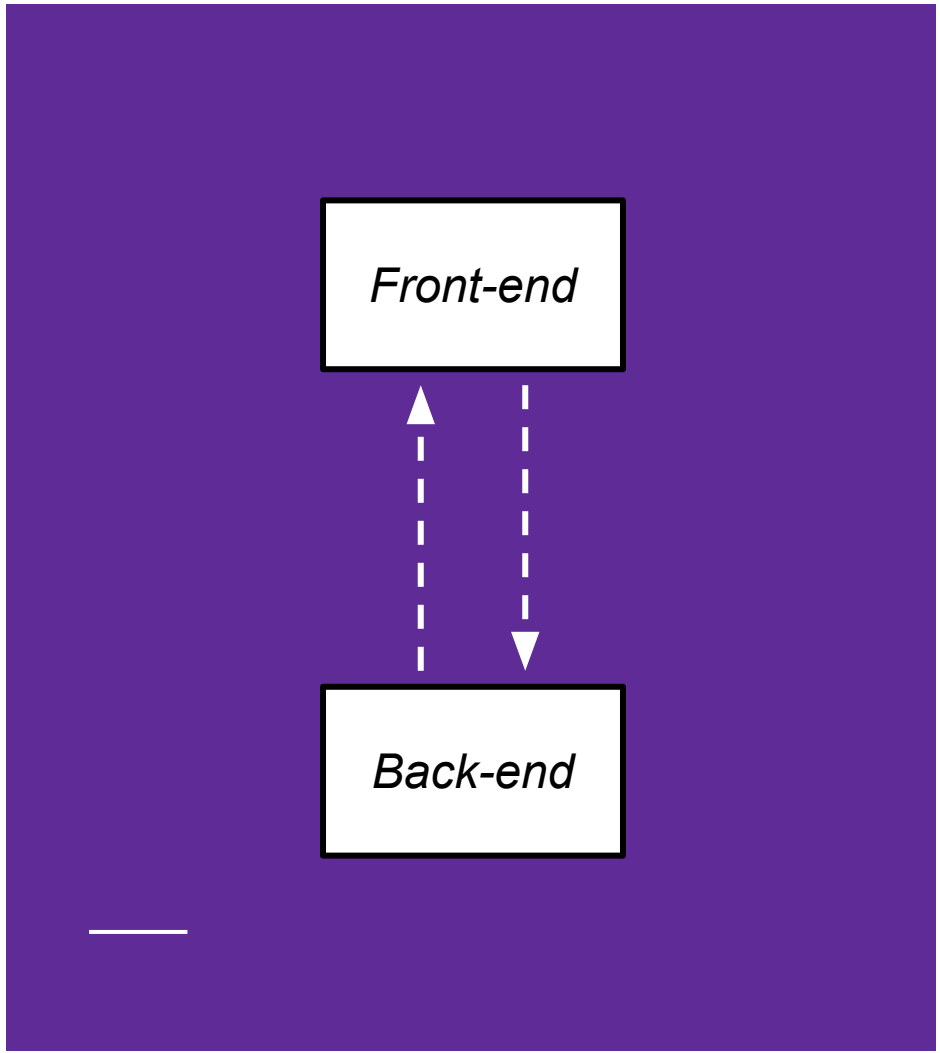
# Main Window 👀

En MacOS, la barra de menú está **arriba en la pantalla**, no está junto a la aplicación.



# REPETIMOS

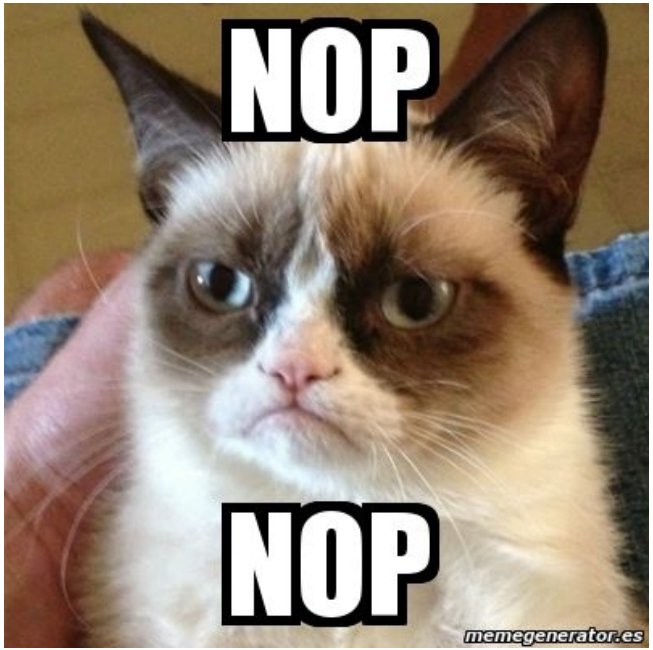
Tengan mucho cuidado con las **dependencias circulares**



# Tengan mucho cuidado con las dependencias circulares

```
class Frontend:
     def __init__(self):
          self.backend = Backend(self)


class Backend:
     def __init__(self, frontend):
          self.frontend = frontend
```



# Tengan mucho cuidado con las dependencias circulares

```
class Frontend:
    def __init__(self):
        sel
cl
        , frontend):
        self.frontend = frontend
```

¡Usen señales!

# *Programación Avanzada*
# IIC2233 2023-2

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Joaquín Tagle - Francisca Cattan