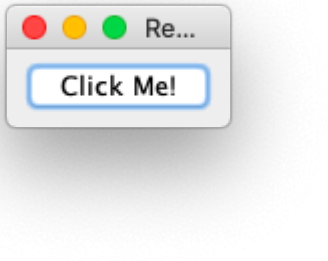
Envío de formulario entre un cliente y un servidor mediante RMI en Kotlin

Este ejemplo crea una ventana de aplicación gráfica con un botón. Cuando se hace clic en el botón, se muestra un mensaje en la consola. El cliente RMI invoca el método run() del objeto remoto y recibe los datos de la aplicación gráfica, que se pueden utilizar para interactuar con los componentes gráficos si es necesario.



Interfaz remota

```
import java.rmi.Remote
import java.rmi.RemoteException

interface RemoteApp : Remote {
    @Throws(RemoteException::class)
    fun run(): AppData
}
```

Clase que implementa la interfaz remota

```
import java.awt.BorderLayout
import java.awt.event.ActionEvent
import java.awt.event.ActionListener
```

```
import java.io.Serializable
import javax.swing.JButton
import javax.swing.JFrame
import javax.swing.JPanel

class RemoteAppImpl : RemoteApp, Serializable {

    override fun run(): AppData {
        val frame = JFrame("Remote App")
        frame.defaultCloseOperation = JFrame.EXIT_ON_CLOSE

        val panel = JPanel()
        val button = JButton("Click Me!")

        button.addActionListener {
            println("Button clicked!")
        }

        panel.add(button)
        frame.add(panel, BorderLayout.CENTER)

        frame.pack()
        frame.isVisible = true

        return AppData(frame, button)
    }
}
```

Clase que encapsula los datos de la aplicación gráfica

```
import java.awt.Component
import java.io.Serializable

data class AppData(val frame: Component, val button: Component) : Serializable
```

Servidor RMI

```
import java.rmi.registry.LocateRegistry
import java.rmi.registry.Registry
import java.rmi.server.UnicastRemoteObject

fun main() {
    val app = RemoteAppImpl()
    val port = 1099

    try {
        val stub = UnicastRemoteObject.exportObject(app, 0) //
Exporta el objeto remoto

                        val   registry:   Registry   =
LocateRegistry.createRegistry(port)
        registry.bind("RemoteApp", stub)

        println("Servidor RMI en ejecución...")
    } catch (e: Exception) {
           println("Error al iniciar el servidor RMI:
${e.message}")
        e.printStackTrace()
    }
}
```

Cliente RMI

```
import java.rmi.registry.LocateRegistry
import java.rmi.registry.Registry

fun main() {
    val port = 1099

    try {
                        val   registry:   Registry   =
LocateRegistry.getRegistry(port)
          val remoteApp = registry.lookup("RemoteApp") as
RemoteApp

        val appData = remoteApp.run()
```

```
            // Utiliza appData para interactuar con los
componentes gráficos si es necesario
        } catch (e: Exception) {
            println("Error al invocar el método remoto:
${e.message}")
            e.printStackTrace()
        }
}
```

Recuerda que debes ejecutar el servidor RMI antes de ejecutar el cliente RMI para asegurarte de que el objeto remoto esté disponible. Asegúrate de tener las dependencias necesarias importadas y configuradas correctamente para utilizar RMI en Kotlin.