

The screenshot shows the IntelliJ IDEA interface with the project 'EjercicioTCP' open. The code editor displays the `ServidorTCP.java` file, which contains the logic for accepting client connections and spawning threads. The terminal window below shows the server listening on port 2000 and accepting two client connections from 'Cliente-1' and 'Cliente-2'. The status bar at the bottom indicates the code is in UTF-8 encoding.

```

package org.example;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.concurrent.atomic.AtomicInteger;

public class ServidorTCP {
    public static void main(String[] args) {
        AtomicInteger contadorClientes = new AtomicInteger(0);
        try (ServerSocket socketServidor = new ServerSocket(Puerto)) {
            System.out.println("Escuchando puerto: " + Puerto);
            while (true) {
                try {
                    Socket skCliente = socketServidor.accept();
                    int id = contadorClientes.getAndIncrement();
                    String nombreHilo = "Cliente-" + id;
                    System.out.println("Conexión aceptada de " + skCliente.getInetAddress() + ":" + skCliente.getPort() + " -> " + nombreHilo);
                    // Lanzamos un hilo por cliente
                    HiloSocket hilo = new HiloSocket(skCliente, nombreHilo);
                    hilo.start();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

```

El servidor con dos clientes conectados.

El cliente en ejecución.

The screenshot shows the IntelliJ IDEA interface with the project 'EjercicioTCP' open. The code editor displays the `ClienteTCP.java` file, which handles user input and sends it to the server. The terminal window below shows the client interacting with the server, sending letters and receiving feedback. The status bar at the bottom indicates the code is in UTF-8 encoding.

```

public class ClienteTCP {
    public static void main(String[] args) {
        String palabra = partes.length > 1 ? partes[1] : "";
        fallos = partes.length > 2 ? parseIntSafe(partes[2]) : fallos;
        System.out.println("Has perdido. La palabra era: " + palabra + " | Fallos: " + fallos);
        jugando = false;
        break;
    }
    if(jugando){System.out.print("Introduce una letra (solo 1 carácter):");}
    String entrada = sc.nextLine();
    // Enviar tal cuál, el servidor validará si es 1 carácter
    out.writeUTF(entrada);
}

```