**Double degree in**

**COMPUTER ENGINEERING + DESIGN AND DEVELOPMENT OF VIDEO GAMES**

**Multiplayer**

Report of the chat

Work elaborated by:

Daniel Sánchez Cebrián

29/04/23



In this Netcode-based Unity chat project, I have a manager who handles all the logic for the scene. Also, there is only one script where everything is done.

The scene has a simple user interface. First, a screen that asks if you want to be a server or a client, and if you are a client, you must enter a name. If you are the server, you cannot write anything in the chat, but you can see the sent messages and if you are the client, when you joined, a "player connected" message is sent, and all your next messages are sent in red meanwhile the others you receive will be black

The script has three main sections: connection section and message section.

The connections section has two functions: ServerConnection and ClientConnection.

The ServerConnection function starts a server and sets the connection panel to inactive. It also appends a message to the chat text saying that the server has started (only as an info).

The ClientConnection function checks if the username input is not empty, and then sets the chat panel to active and assigns the username to the username variable. It then starts a client and also sets the connection panel to inactive. It also calls a coroutine called ClientConnected.

The script also has two networked message functions: SendChatMessageServerRpc and ReceiveChatMessageClientRpc. These functions use server and client RPCs (remote procedure calls) to send and receive chat messages across the network.

The SendChatMessageServerRpc function takes a message and a client ID as parameters and sends them to all clients using ReceiveChatMessageClientRpc. It also appends the message to the chat text with a black color.

The ReceiveChatMessageClientRpc function also takes a message and a client ID as parameters and appends them to the chat text with different colors depending on who sent them. If the client ID matches the local client ID, then the message is colored red, indicating that it was sent by the user. If the client ID is zero, then the message is colored blue, indicating that someone joined the room. Otherwise, the message is colored black, indicating that it was sent by another client.

The script also uses two coroutines: ScrollDown and ClientConnected. The ScrollDown coroutine sets the value of the scrollbar to zero after a short delay, ensuring that the chat screen always shows the latest messages. The ClientConnected coroutine waits until the client is connected to a server, and then sends a welcome message using SendChatMessageServerRpc with a zero client ID.