Use Case Realizations

Add Contact:

User selects Add Contact button on the home form. The event handler in the HomeForm makes an AddContact form pop up, prompting the user to enter the name of the contact into the textbox. The user enters a name and presses 'Enter'. The HomeForm's AddContact event handler takes the result as a DialogResult, takes the name in the textbox, and calls the \_adddCHandler delegate to send to call the controller's AddContact function with the string parameter of the textbox's text. The controller checks to see if the WebSocket connection is working, and if so, constructs a Mensaje object with a State of AddContact, the Contact that references the username, and the User who is adding the contact as its parameters. It then uses Json to serialize the Mensaje into a string and sends through the WebSocket to the server.

The server's implementation of WebSocketBehavior: Chat, calls the OnMessage method, since something was received through the WebSocket. That method deserializes the data of the message into a Mensaje object, and calls the controller's ChatDelegate function through the delegate \_receive, sending as the parameters the Mensaje object and the session id of the user who sent it. The ChatDelegate function in the controller reads the State of the Mensaje as AddContact and calls the AddContact method, with the string parameters of the username of the person to add from the Mensaje's Contact object, and the string username of the Mensaje's User object. The AddContact method searches both users in the chat database and creates User objects based on both of them. If the object of the person to be added is null, the controller sends back to the User through the Send delegate a Mensaje that has a state of AddContact and an error message saying that that user does not exist. If the User does exist (is not null), the controller adds to both Users' ContactLists the other person's username and OnlineStatus as a Contact in the database. It also updates the Server's view through the SignalEventObserver method, which calls the \_eventObserver list of delegates to update the form. The controller then sends to both clients (if they are online) that the other person has been added as a contact through the \_send delegate with a new IMesnaje containing AddContact as a State, the other user's Contact information, the sender as a User, and the sessionId of the person the message is sent to. The \_send delegate refers to Chat's Send method, which serializes the Mensaje and sends to the clients with the provided sessionId.

The client receives the string from the websocket in its controller's message function. It deserializes the string as a Mensaje, reads the state as AddContact, updates the database's User field to the User object that was sent back through the Mensaje and calls the SignalHFormObserver method, which updates the view through the  \_hFormObserver delegate according to what was received.

Logout:

User clicks the logout button on the HomeForm in Client. The event handler for the logout button calls the \_sOutHandler delegate which refers to the controller's SignOut method. That method checks to make sure the WebSocket connection is open, and if it is, constructs a new Mensaje object with a state of Logout and the User stored in the ChatDB as its parameters. It then serializes the Mensaje and sends it to the Server through the WebSocket.

The server's implementation of WebSocketBehavior: Chat, calls the OnMessage method, since something was received through the WebSocket. That method deserializes the data of the message into a Mensaje object, and calls the controller's ChatDelegate function through the delegate \_receive, sending as the parameters the Mensaje object and the session id of the user who sent it. ChatDelegate reads the Mensaje's state as Logout and calls the Logout methodwith the Mensaje's User's Contact's username as the parameter. The Logout method looks up the user with the given username and sets it to offline. It sends the person logging out a serialized Mensaje with the state of Logout and the user that logged out through the \_send delegate. It also notifies all of that user's contacts through the \_send delegate that the user is offline. Finally, the controller calls the SignalEventObserver method to update the server's view.

The Client's message method is called by the WebSocket, and it deserializes the string parameter into a Mensaje object, reads the state as Logout, updates the contacts if the user has any contacts, and calls SignalHFormObserver, which calls the \_hFormObserver delegate, which closes the home form and returns to sign in.