Milestone 3

Low Level Design

Presentation

MainWindow : Window

- **Attributes:**
  - **ObservableObject _main – the binding object to the window's xaml components.**
  - **Chatroom myChatRoom – Stores the chatroom object in use.**
  - **ProgramWindow pw – Stores the main chatroom window.**
- **Methods:**
  - **void DataWindow_Closing(object, CancelEventArgs) – Determains what happens when closing the window.**
  - **void btn_login_Click(object, CancelEventArgs) – A function for the login button.**
  - **void btn_register_Click(object, CancelEventArgs) – A function for the register button.**
  - **void StartProgram() – Starts the main chatroom window.**
  - **void login_password_changed(object, CancelEventArgs) – called when the login password box is changed**
  - **void register_password_changed(object, CancelEventArgs) – called when the register password box is changed**

ProgramWindow : Window

- **Attributes:**
  - **ObservableObject _main – the binding object to the window's xaml components.**
  - **Chatroom chatroom – Stores the chatroom object in use.**
  - **MainWindow main – Stores the chatroom's login window.**
  - **DispatcherTimer dispatcherTimer – Stores the chatroom's timer.**
- **Methods:**
  - **void startWindow() – starts the timer and updates the window's view.**
  - **void UpdateView() – Updates the window's view.**
  - **void NewView() – Refreshes the window's view.**

- void btn_logout_Click(object, CancelEventArgs) – A function for the logout button.
- void ComboBox_SelectionChanged(object, CancelEventArgs) – the function triggered when the filter combo box is changed, and updates the editable fields through the binding.
- void dispatcherTimer_Tick(object, CancelEventArgs) – triggered by the timer every 2 second, retrieves new messages and update the display if necessary.
- void sort_filter_Button_Click(object, CancelEventArgs) – A function for the apply sort and filter button, sends the selection to the chatroom.
- void ProgramWindow_Closing(object, CancelEventArgs) – Determines what happens when closing the window, logout and close the chatroom.
- void Send_Button_Click(object, CancelEventArgs) – calls the create message function in the chatroom.
- void Edit_Button_Click(object, CancelEventArgs) – calls the edit message function in the chatroom.

EditMessage : Window

- Attributes:
  - ObservableObject _main – the binding object to the window's xaml components.
- Methods:
  - void Button_Click() – event for pressing edit in the edit message window.

ObservableObject : INotifyPropertyChanged

- Attributes: ○ PropertyChangedEventHandler PropertyChanged – Required event for this binding class.
  - ObservableCollection<string> Messages – binding for the messages panel.
  - string nicknameR – Binding for the register nickname textbox.
  - string nicknameL – Binding for the login nickname textbox.
  - string groupR – Binding for the register groupid textbox.
  - string groupL – Binding for the login groupid textbox.
  - string sortCombo – Binding for the sort combo box.
  - string isDesc – Binding for the descending checkbox.
  - string filterCombo – Binding for the filter combo box.

- **string isFilterGroup – Binding for the filter group text box availability.**
- **string filterGroup – Binding for the filter group text box.**
- **string isFilterUser – Binding for the filter user text box availability.**
- **string filterUser – Binding for the filter user text box.**
- **string messageText – Binding for the send text text box.**
- **string SelectedListItem – Binding for the selected message's index in the list box.**
- **string edit – Binding for the edit message text box.**
- **string pressEdit – edit whether the user pressed edit in the edit window.**

- **Methods:**
  - **void Messages_CollectionChanged(object, NotifyCollectionChangedEventArg - Required method for this binding class.**
  - **void OnPropertyChanged([CallerMemberName] string) – Required method for this binding class.**

Logic

ChatRoom

- **Attributes:**
  - **int sortType – Stores what sort type the user chose.**
  - **boolean isAsc – Stores what order the user has asked for.**
  - **int filterType – Stores what filter type the user chose.**
  - **string userFilter – Stores what user to filter by.**
  - **string groupFilter – Stores what group to filter by.**
  - **int _loggedInUser – Stores the id of the user currently logged in or -1 otherwise.**
  - **Logger mLogger – Logger object used by the logger.**
  - **FileLogger mFileLogger – FileLogger object used by the logger.**
  - **sqlHandler _sqlHandler –handles all communication with the DB.**
  - **String hashedRPassword – stores the hashed register password.**
  - **String hashedLPassword – stores the hashed login password.**
  - **List<Guid> MessageGuid – stores guids of the displayed messaged in order.**
- **Methods:**
  - **boolean Login(string nickname, string groupID) – gets a user nickname and loges him in to the system, return false if user non-exists.**
  - **boolean Logout() – loges the user out, returns false if no user is logged in.**
  - **boolean register(string nickname, string groupID) – Creates a user.**
  - **int retrive10Messages() – Returns last 10 messages.**
  - **void SetFilterAndSort(int,int,bool,string,string) – Sets the sort and filter properties.**
  - **List<String> GetAllMessages(boolan all) – Returns all the messages to be shown at the messages panel in the right order and filter.**
  - **List<String> SortByTimestamp(List<IMessage>) – Sorts the given messages by the timestamp.**
  - **List<String> SortByNickname(List<IMessage>) – Sorts the given messages by the nickname.**
  - **List<String> SortByAll(List<IMessage>) – Sorts the given messages by the timestamp, group and nickname.**
  - **boolean writeMessages(String message) – sends message, returns true if successful.**
  - **Boolean CheckMessageValidity(String content) – gets a new message content and return whether or not its valid before sending it.**

- **void exit() – terminates logger.**
- **void ProcessLogMessage(String message) – useless function needed to implement the logger.**
- **void updateRPassword(String password) – checks the password for validity, hashes it and stores it in the chatroom.**
- **void updateLPassword(String password) – checks the password for validity, hashes it and stores it in the chatroom.**
- **bool isRPasswordValid() – returns whether the last register password given was valid.**
- **bool isLPasswordValid() – returns whether the last login password given was valid.**
- **bool isOwner(int index) – returns whether the logged in user is the owner of the message in the given index.**
- **void UpdateGUIDTable(List<GUID> guids) – adds the new guids to the guids list.**
- **void EditMessage(int id, string text) – updates the content of the message with the gived id to the given text.**
- **bool isPassValid(string password) – returns whether the given password is valid.**

Persistence

sqlHandler

- **Attributes:**
  - **string url – the server's url**
  - **string dbName – the db's name.**
  - **string username – username used to connect to the db.**
  - **string password – password used to connect to the db.**
  - **string msgTblName – the name of the messages table.**
  - **string usrTblName – the name of the users table.**
- **Methods:**
  - **bool userExists(String nickname ,String gid) – returns whether a user exists or not.**
  - **void editMessage(Guid mid,string contnet) – updates the content and time of the message according to the parameters.**
  - **bool isOwner(Guid mid,string uid) – returns whether the user is the owner of the given message id.**
  - **int loginUser(string nickname, string gid, string password) – finds a user with the right parameters and returns it's id.**
  - **void registerUser(string nickname, string gid, string password) – adds a user with the given parameters to the table.**
  - **void retriveAllMessages(string groupfilter, string userfilter) – gets sorting parameters that can be empty and returns all the appropriate messages for the DB.**
  - **void retriveNewMessages(string groupfilter, string userfilter) – gets sorting parameters that can be empty and returns only the new appropriate messages for the DB.**
  - **string LastSQL() – returns SQL representation of the last sql query time.**
  - **string NextSQL() – returns SQL representation of the next sql query time.**
  - **string toSQLDate(DateTime dt) – returns SQL representation of the parameter.**
  - **void sendMessage(string uid, string content) – creates a new message inn the DB using the given parameters.**

IMessage

- **Attributes:**
  - Guid **Id – the message Id.**
  - string **UserName – the name of the writer.**
  - DateTime **date – the time the message was sent.**
  - string **messageContent – the message content.**
  - string **GroupID – the group of the writer.**
- **Methods:**
  - string **ToString() – returns a string representation of the message.**

ILogger - downloaded

Logger - downloaded

FileLogger – downloaded