

Focus

1.0

Generated by Doxygen 1.8.17

1 Focus	1
1.1 Project	1
1.2 References	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 Class Documentation	7
4.1 AddUser Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 AddUser()	8
4.2 Auth Class Reference	8
4.2.1 Detailed Description	9
4.2.2 Constructor & Destructor Documentation	9
4.2.2.1 Auth()	10
4.2.3 Member Function Documentation	10
4.2.3.1 validateLogin()	10
4.2.3.2 validateRegister()	10
4.3 Chatroom Class Reference	11
4.3.1 Detailed Description	11
4.3.2 Constructor & Destructor Documentation	11
4.3.2.1 Chatroom()	11
4.4 ChatroomWindow Class Reference	12
4.4.1 Detailed Description	13
4.4.2 Constructor & Destructor Documentation	13
4.4.2.1 ChatroomWindow()	13
4.4.3 Member Function Documentation	13
4.4.3.1 loadChatroom()	13
4.5 EditProfile Class Reference	14
4.5.1 Detailed Description	15
4.5.2 Constructor & Destructor Documentation	15
4.5.2.1 EditProfile()	15
4.6 MainWindow Class Reference	15
4.6.1 Detailed Description	16
4.6.2 Constructor & Destructor Documentation	16
4.6.2.1 MainWindow()	16
4.6.3 Member Function Documentation	17
4.6.3.1 newChatroomWindow()	17
4.7 Message Class Reference	17

4.7.1 Detailed Description	18
4.7.2 Constructor & Destructor Documentation	18
4.7.2.1 Message()	18
4.8 Mosquitto Class Reference	18
4.8.1 Detailed Description	19
4.8.2 Constructor & Destructor Documentation	19
4.8.2.1 Mosquitto()	19
4.8.3 Member Function Documentation	20
4.8.3.1 publish()	20
4.8.3.2 subscribe()	20
4.9 Profile Class Reference	20
4.9.1 Detailed Description	21
4.9.2 Constructor & Destructor Documentation	21
4.9.2.1 Profile()	22
4.10 SaveGroup Class Reference	22
4.10.1 Detailed Description	23
4.10.2 Constructor & Destructor Documentation	23
4.10.2.1 SaveGroup()	23
4.11 SQL Class Reference	24
4.11.1 Detailed Description	25
4.11.2 Member Function Documentation	25
4.11.2.1 addContactByUsername()	25
4.11.2.2 addGroup()	25
4.11.2.3 addSubscription()	26
4.11.2.4 addUser()	26
4.11.2.5 areContacts()	26
4.11.2.6 getGroupsByTopic()	27
4.11.2.7 getMembersByTopic()	27
4.11.2.8 getMessageById()	27
4.11.2.9 getMessagesByTopic()	28
4.11.2.10 getSubscriptionsByUser()	28
4.11.2.11 getUserByEmail()	29
4.11.2.12 getUserByUsername()	29
4.11.2.13 getUserContacts()	29
4.11.2.14 getUserDms()	30
4.11.2.15 isGroupAdmin()	30
4.11.2.16 isValidLoginByEmail()	30
4.11.2.17 isValidLoginByUsername()	31
4.11.2.18 removeContact()	31
4.11.2.19 removeGroup()	31
4.11.2.20 removeSubscriptions()	32
4.11.2.21 updateGroup()	32

4.11.2.22 updateUser()	32
4.11.2.23 validateTopic()	33
4.12 User Class Reference	33
4.12.1 Detailed Description	33
4.12.2 Constructor & Destructor Documentation	33
4.12.2.1 User()	33
Index	35

Chapter 1

Focus

1.1 Project

Focus is a messaging application tailored for workers, students or those with group projects. Carefully designed to aid project development and planning giving users the ability to effectively communicate with eachother. Through the use of roles and permissions and customization features, Focus gives a unique and tailored experience, a perfect tool for your workspace.

1.2 References

To achieve the results shown in this application this external tools were used:

- Flaticon
- Doxygen
- Qt
- Mqtt

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chatroom	11
Message	17
QDialog	
AddUser	7
Auth	8
EditProfile	14
Profile	20
SaveGroup	22
QMainWindow	
MainWindow	15
QObject	
Mosquito	18
QWidget	
ChatroomWindow	12
SQL	24
User	33

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AddUser	The AddUser class	7
Auth	The Auth class	8
Chatroom	The Chatroom class	11
ChatroomWindow	The Chatroom class	12
EditProfile	The EditProfile class	14
MainWindow	The MainWindow class	15
Message	The Message class	17
Mosquitto	The Mosquitto class	18
Profile	The Profile class	20
SaveGroup	The Profile class	22
SQL	The SQL class	24
User	The User class	33

Chapter 4

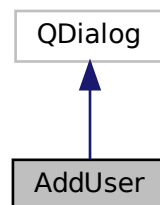
Class Documentation

4.1 AddUser Class Reference

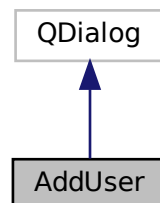
The `AddUser` class.

```
#include <adduser.h>
```

Inheritance diagram for AddUser:



Collaboration diagram for AddUser:



Public Member Functions

- [AddUser](#) ([SQL](#) *sql, [User](#) *user, QWidget *parent=nullptr)
Constructor for [AddUser](#) class.

4.1.1 Detailed Description

The [AddUser](#) class.

This class is used to generate the [AddUser](#) Dialog window

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AddUser()

```
AddUser::AddUser (
    SQL * sql,
    User * user,
    QWidget * parent = nullptr ) [explicit]
```

Constructor for [AddUser](#) class.

Parameters

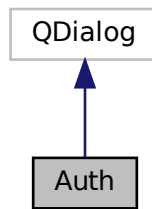
<i>sql</i>	
<i>user</i>	
<i>parent</i>	

4.2 Auth Class Reference

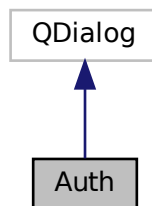
The [Auth](#) class.

```
#include <auth.h>
```

Inheritance diagram for Auth:



Collaboration diagram for Auth:



Public Member Functions

- [Auth](#) ([SQL](#) *sql, QWidget *parent=nullptr)
Constructor for [Auth](#).
- QString [validateLogin](#) (QString email, QString password, QString encryptedPassword)
validateLogin - validates the credentials provided by the user and checks if it is a valid login
- QString [validateRegister](#) ([User](#) *user, QString password, QString confirmPassword)
validateRegister - validates the information provided by the user

4.2.1 Detailed Description

The [Auth](#) class.

This class is used to create the authentication page

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Auth()

```
Auth::Auth (
    SQL * sql,
    QWidget * parent = nullptr ) [explicit]
```

Constructor for [Auth](#).

Parameters

<i>sql</i>	
<i>parent</i>	

4.2.3 Member Function Documentation

4.2.3.1 validateLogin()

```
QString Auth::validateLogin (
    QString email,
    QString password,
    QString encryptedPassword )
```

validateLogin - validates the credentials provided by the user and checks if it is a valid login

Parameters

<i>email</i>	
<i>password</i>	
<i>encryptedPassword</i>	

Returns

4.2.3.2 validateRegister()

```
QString Auth::validateRegister (
    User * user,
    QString password,
    QString confirmPassword )
```

validateRegister - validates the information provided by the user

Parameters

<i>user</i>	
<i>password</i>	
<i>confirmPassword</i>	

Returns

4.3 Chatroom Class Reference

The [Chatroom](#) class.

```
#include <chatroom.h>
```

Public Member Functions

- [Chatroom](#) (QString name, QVector< [User](#) > members, QString topic)
Constructor for [Chatroom](#) class.
- QString **name** () const
- void **setName** (const QString &name)
- QString **topic** () const
- void **setTopic** (const QString &topic)
- QVector< [User](#) > **members** () const
- void **setMembers** (const QVector< [User](#) > &members)

4.3.1 Detailed Description

The [Chatroom](#) class.

This class is used to store the chatroom information

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Chatroom()

```
Chatroom::Chatroom (
    QString name,
    QVector< User > members,
    QString topic )
```

Constructor for [Chatroom](#) class.

Parameters

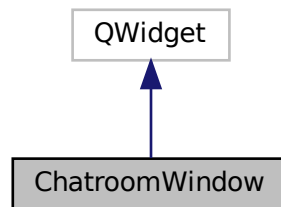
<i>name</i>	
<i>members</i>	
<i>topic</i>	

4.4 ChatroomWindow Class Reference

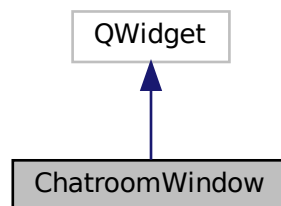
The [Chatroom](#) class.

```
#include <chatroomwindow.h>
```

Inheritance diagram for ChatroomWindow:



Collaboration diagram for ChatroomWindow:



Public Member Functions

- [ChatroomWindow](#) ([Chatroom](#) chatroom, [Mosquitto](#) *mainClient, [SQL](#) *sql, [User](#) *user, QWidget *parent=nullptr)
Constructor for [ChatroomWindow](#) class.
- void [loadMessages](#) ()

- loadMessages* - loads the messages for the respective chatroom
- void [loadMembers](#) ()
 - loadMembers* - loads the members for the respective chatroom
- void [loadChatroom](#) ([Chatroom](#) chatroom)
 - loadChatroom* - loads the chatroom information
- void [onMessageReceived](#) (const QMqttMessage &msg)

4.4.1 Detailed Description

The [Chatroom](#) class.

This class is used to create the chatroom windows with QWidget

4.4.2 Constructor & Destructor Documentation

4.4.2.1 ChatroomWindow()

```
ChatroomWindow::ChatroomWindow (
    Chatroom chatroom,
    Mosquitto * mainClient,
    SQL * sql,
    User * user,
    QWidget * parent = nullptr ) [explicit]
```

Constructor for [ChatroomWindow](#) class.

Parameters

<i>chatroom</i>	
<i>mainClient</i>	
<i>sql</i>	
<i>user</i>	
<i>parent</i>	

4.4.3 Member Function Documentation

4.4.3.1 loadChatroom()

```
void ChatroomWindow::loadChatroom (
    Chatroom chatroom )
```

loadChatroom - loads the chatroom information

Parameters

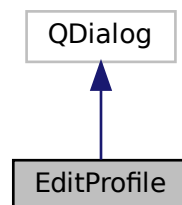
<i>chatroom</i>	
-----------------	--

4.5 EditProfile Class Reference

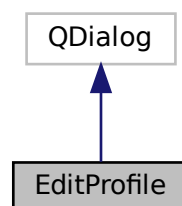
The [EditProfile](#) class.

```
#include <editprofile.h>
```

Inheritance diagram for EditProfile:



Collaboration diagram for EditProfile:



Public Member Functions

- [EditProfile](#) ([SQL](#) *sql, [User](#) *user, QWidget *parent=nullptr)
Constructor for [EditProfile](#) class.

4.5.1 Detailed Description

The [EditProfile](#) class.

This class is used to create the [EditProfile](#) dialog window

4.5.2 Constructor & Destructor Documentation

4.5.2.1 EditProfile()

```
EditProfile::EditProfile (
    SQL * sql,
    User * user,
    QWidget * parent = nullptr ) [explicit]
```

Constructor for [EditProfile](#) class.

Parameters

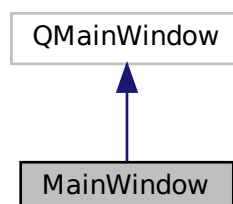
<i>user</i>	
<i>parent</i>	

4.6 MainWindow Class Reference

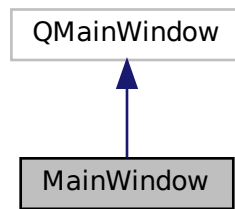
The [MainWindow](#) class.

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:



Collaboration diagram for MainWindow:



Public Member Functions

- `MainWindow (SQL *sql, Mosquitto *mainClient, User *user, QWidget *parent=nullptr)`
Constructor for `MainWindow` class.
- `void newChatroomWindow (Chatroom chatroom)`
`newChatroomWindow` - creates new `ChatroomWindow` Widgets
- `void loadGroupsList ()`
`loadGroupsList` - loads the user's groups
- `void loadDmsList ()`
`loadDmsList` - loads the user's direct messages
- `void loadChatrooms ()`
`loadChatrooms` - loads all the `Chatroom` classes (groups and dms)

4.6.1 Detailed Description

The `MainWindow` class.

This class is used to create the application's main window page using `QMainWindow`

4.6.2 Constructor & Destructor Documentation

4.6.2.1 MainWindow()

```
MainWindow::MainWindow (
    SQL * sql,
    Mosquitto * mainClient,
    User * user,
    QWidget * parent = nullptr )
```

Constructor for `MainWindow` class.

Parameters

<i>sql</i>	
<i>mainClient</i>	
<i>user</i>	
<i>parent</i>	

Note

Workaround because mosquito failed when loading from the constructor

4.6.3 Member Function Documentation

4.6.3.1 newChatroomWindow()

```
void MainWindow::newChatroomWindow (
    Chatroom chatroom )
```

newChatroomWindow - creates new [ChatroomWindow](#) Widgets

Parameters

<i>chatroom</i>	
-----------------	--

4.7 Message Class Reference

The [Message](#) class.

```
#include <message.h>
```

Public Member Functions

- [Message](#) (QString content, QString dateString, QString topic, QString username)
Constructor for [Message](#) class.
- QString **content** () const
- void **setContent** (const QString &content)
- QString **topic** () const
- void **setTopic** (const QString &topic)
- QString **username** () const
- void **setUsername** (const QString &username)
- QDateTime **dateTime** () const
- void **setDateTime** (const QDateTime &dateTime)

4.7.1 Detailed Description

The [Message](#) class.

This class manages the messages' information

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Message()

```
Message::Message (
    QString content,
    QString dateString,
    QString topic,
    QString username )
```

Constructor for [Message](#) class.

Parameters

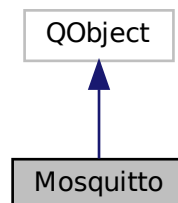
<i>content</i>	
<i>dateString</i>	
<i>topic</i>	
<i>username</i>	

4.8 Mosquitto Class Reference

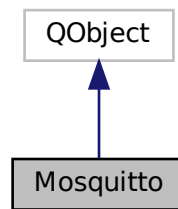
The [Mosquitto](#) class.

```
#include <mosquitto.h>
```

Inheritance diagram for Mosquitto:



Collaboration diagram for Mosquitto:



Public Member Functions

- [Mosquitto](#) (QString host, int port, QString username)
Constructor for [Mosquitto](#) class.
- QMqttClient * **client** () const
- void [publish](#) (QString message, QString topic)
publish - publishes a message in a specific topic with mqtt
- QMqttSubscription * [subscribe](#) (QString topic)
subscribe - subscribes to a topic with mqtt

4.8.1 Detailed Description

The [Mosquitto](#) class.

This class is used to process mqtt's features

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Mosquitto()

```

Mosquitto::Mosquitto (
    QString host,
    int port,
    QString username )
  
```

Constructor for [Mosquitto](#) class.

Parameters

<i>port</i>	
<i>username</i>	

4.8.3 Member Function Documentation

4.8.3.1 `publish()`

```
void Mosquitto::publish (
    QString message,
    QString topic )
```

`publish` - publishes a message in a specific topic with mqtt

Parameters

<i>message</i>	
<i>topic</i>	

4.8.3.2 `subscribe()`

```
QMqttSubscription * Mosquitto::subscribe (
    QString topic )
```

`subscribe` - subscribes to a topic with mqtt

Parameters

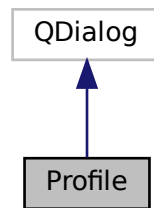
<i>topic</i>	
--------------	--

4.9 Profile Class Reference

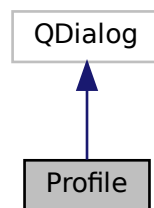
The [Profile](#) class.

```
#include <profile.h>
```

Inheritance diagram for Profile:



Collaboration diagram for Profile:



Public Member Functions

- `Profile` (`SQL *sql`, `User *currentUser`, `User *user`, `QWidget *parent=nullptr`)
Constructor for `Profile` class.

4.9.1 Detailed Description

The `Profile` class.

This class is used to create the profile dialog window

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Profile()

```
Profile::Profile (
    SQL * sql,
    User * currentUser,
    User * user,
    QWidget * parent = nullptr ) [explicit]
```

Constructor for [Profile](#) class.

Parameters

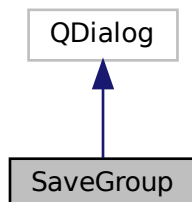
<i>sql</i>	
<i>currentUser</i>	
<i>user</i>	
<i>parent</i>	

4.10 SaveGroup Class Reference

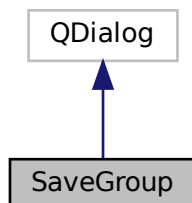
The [Profile](#) class.

```
#include <savegroup.h>
```

Inheritance diagram for SaveGroup:



Collaboration diagram for SaveGroup:



Public Member Functions

- [SaveGroup](#) ([SQL](#) *sql, [User](#) *user, [Chatroom](#) *group=nullptr, [QWidget](#) *parent=nullptr)
Constructor for [SaveGroup](#) class.
- void [createNewGroup](#) ()
createNewGroup - creates a new group
- void [updateGroup](#) ()
updateGroup - updates the group's information
- void [loadContactsList](#) ()
loadContactsList - loads the user's contacts
- void [loadMembersList](#) ()
loadMembersList - loads the group's members
- [Chatroom](#) * **group** () const
- void [setGroup](#) ([Chatroom](#) *group)

Public Attributes

- bool **_deleted** = false

4.10.1 Detailed Description

The [Profile](#) class.

This class is used to create the [SaveGroup](#) dialog window

4.10.2 Constructor & Destructor Documentation

4.10.2.1 SaveGroup()

```
SaveGroup::SaveGroup (
    SQL * sql,
    User * user,
    Chatroom * group = nullptr,
    QWidget * parent = nullptr ) [explicit]
```

Constructor for [SaveGroup](#) class.

Parameters

<i>sql</i>	
<i>user</i>	
<i>group</i>	
<i>parent</i>	

4.11 SQL Class Reference

The [SQL](#) class.

```
#include <sql.h>
```

Public Member Functions

- [SQL](#) ()
Constructor for [SQL](#) class.
- [QString](#) [validateTopic](#) ([QString](#) topic)
validateTopic - validates the topic name
- void [addGroup](#) ([QString](#) name, [QString](#) topic, [User](#) user)
addGroup - adds a new group to the database
- void [addUser](#) ([User](#) *user, [QString](#) passwordHash)
addUser - adds a new user to the database
- void [addContactByUsername](#) ([User](#) *user, [QString](#) contactUsername)
addContactByUsername - connects the user to the requested contact
- int [addMessage](#) ([Message](#) &message)
- void [addSubscription](#) ([QString](#) topic, [User](#) *user)
addSubscription - stores subscription in database
- void [updateGroup](#) ([Chatroom](#) *group)
updateGroup - updates the group information
- void [updateUser](#) ([User](#) *user)
updateUser - updates the user information
- void [removeSubscriptions](#) ([QString](#) topic)
removeSubscriptions - removes the stored subscription from the database
- void [removeContact](#) ([User](#) *currentUser, [User](#) *user)
removeContact - removes the contact for respective user
- void [removeGroup](#) ([Chatroom](#) *group)
removeGroup - removes group information from database
- [QVector](#)< [QString](#) > [getSubscriptionsByUser](#) ([User](#) *user)
getSubscriptionsByUser - returns subscriptions for respective user
- [QVector](#)< [Message](#) > [getMessagesByTopic](#) ([QString](#) topic, int limit=30)
getMessagesByTopic - returns the last 30 messages for respective topic
- [QVector](#)< [User](#) > [getUserContacts](#) ([User](#) *user)
getUserContacts - returns vector with the user's contacts
- [QVector](#)< [Chatroom](#) > [getUserDms](#) ([User](#) *user)
getUserDms - returns the direct messages of the respective user
- [QVector](#)< [Chatroom](#) > [getGroupsByTopic](#) ([QString](#) topic)
getGroupsByTopic - returns the respective user's groups
- [QVector](#)< [User](#) > [getMembersByTopic](#) ([QString](#) topic)
getMembersByTopic - returns the members subscribe to the respective topic
- [User](#) * [getUserByUsername](#) ([QString](#) username)
getUserByUsername - returns the user's information by username
- [User](#) * [getUserByEmail](#) ([QString](#) email)
getUserByEmail - returns the user's information by email
- [Message](#) * [getMessageById](#) ([QString](#) id)
getMessageById - returns the message information by id
- bool [isValidLoginByEmail](#) ([QString](#) email, [QString](#) passwordHash)

- isValidLoginByEmail* - checks if the login by email is valid
- bool [isValidLoginByUsername](#) (QString username, QString passwordHash)
isValidLoginByUsername - checks if the login by username is valid
- bool [isGroupAdmin](#) (User *user, QString topic)
isGroupAdmin - returns true if the respective user is the group's admin
- bool [areContacts](#) (User *currentUser, User *user)
areContacts - checks if the users are connected

4.11.1 Detailed Description

The [SQL](#) class.

This class is used to process mysql features

4.11.2 Member Function Documentation

4.11.2.1 addContactByUsername()

```
void SQL::addContactByUsername (
    User * user,
    QString contactUsername )
```

[addContactByUsername](#) - connects the user to the requested contact

Parameters

<i>user</i>	
<i>contactUsername</i>	

Returns

bool (true if successfull)

4.11.2.2 addGroup()

```
void SQL::addGroup (
    QString name,
    QString topic,
    User user )
```

[addGroup](#) - adds a new group to the database

Parameters

<i>name</i>	
<i>topic</i>	
<i>user</i>	

4.11.2.3 addSubscription()

```
void SQL::addSubscription (
    QString topic,
    User * user )
```

addSubscription - stores subscription in database

Parameters

<i>topic</i>	
<i>user</i>	

4.11.2.4 addUser()

```
void SQL::addUser (
    User * user,
    QString passwordHash )
```

addUser - adds a new user to the database

Parameters

<i>user</i>	
<i>passwordHash</i>	

4.11.2.5 areContacts()

```
bool SQL::areContacts (
    User * currentUser,
    User * user )
```

areContacts - checks if the users are connected

Parameters

<i>currentUser</i>	
<i>user</i>	

Returns

bool

4.11.2.6 getGroupsByTopic()

```
QVector< Chatroom > SQL::getGroupsByTopic (
    QString topic )
```

getGroupsByTopic - returns the repective user's groups

Parameters

<i>topic</i>	
--------------	--

Returns

QVector<Chatroom>

4.11.2.7 getMembersByTopic()

```
QVector< User > SQL::getMembersByTopic (
    QString topic )
```

getMembersByTopic - returns the members subscribe to the respective topic

Parameters

<i>topic</i>	
--------------	--

Returns

QVector<User>

4.11.2.8 getMessageById()

```
Message * SQL::getMessageById (
    QString id )
```

getMessageById - returns the message information by id

Parameters

<i>id</i>	
-----------	--

Returns

[Message](#)

4.11.2.9 getMessagesByTopic()

```
QVector< Message > SQL::getMessagesByTopic (
    QString topic,
    int limit = 30 )
```

getMessagesByTopic - returns the last 30 messages for respective topic

Parameters

<i>topic</i>	
<i>limit</i>	(30 messages)

Returns

QVector<Message>

4.11.2.10 getSubscriptionsByUser()

```
QVector< QString > SQL::getSubscriptionsByUser (
    User * user )
```

getSubscriptionsByUser - returns subscriptions for respective user

Parameters

<i>user</i>	
-------------	--

Returns

QVector<QString>

4.11.2.11 getUserByEmail()

```
User * SQL::getUserByEmail (
    QString email )
```

getUserByEmail - returns the user's information by email

Parameters

<i>email</i>	
--------------	--

Returns

User

4.11.2.12 getUserByUsername()

```
User * SQL::getUserByUsername (
    QString username )
```

getUserByUsername - returns the user's information by username

Parameters

<i>username</i>	
-----------------	--

Returns

User

4.11.2.13 getUserContacts()

```
QVector< User > SQL::getUserContacts (
    User * user )
```

getUserContacts - returns vector with the user's contacts

Parameters

<i>user</i>	
-------------	--

Returns

QVector<User>

4.11.2.14 getUserDms()

```
QVector< Chatroom > SQL::getUserDms (
    User * user )
```

getUserDms - returns the direct messages of the respective user

Parameters

<i>user</i>	
-------------	--

Returns

QVector<Chatroom>

4.11.2.15 isGroupAdmin()

```
bool SQL::isGroupAdmin (
    User * user,
    QString topic )
```

isGroupAdmin - returns true if the respective user is the group's admin

Parameters

<i>user</i>	
<i>topic</i>	

Returns

bool

4.11.2.16 isValidLoginByEmail()

```
bool SQL::isValidLoginByEmail (
    QString email,
    QString passwordHash )
```

isValidLoginByEmail - checks if the login by email is valid

Parameters

<i>email</i>	
<i>passwordHash</i>	

Returns

bool

4.11.2.17 isValidLoginByUsername()

```
bool SQL::isValidLoginByUsername (
    QString username,
    QString passwordHash )
```

isValidLoginByUsername - checks if the login by username is valid

Parameters

<i>username</i>	
<i>passwordHash</i>	

Returns

bool

4.11.2.18 removeContact()

```
void SQL::removeContact (
    User * currentUser,
    User * user )
```

removeContact - removes the contact for respective user

Parameters

<i>currentUser</i>	
<i>user</i>	

4.11.2.19 removeGroup()

```
void SQL::removeGroup (
    Chatroom * group )
```

removeGroup - removes group information from database

Parameters

<i>group</i>	
--------------	--

4.11.2.20 removeSubscriptions()

```
void SQL::removeSubscriptions (
    QString topic )
```

removeSubscriptions - removes the stored subscription from the database

Parameters

<i>topic</i>	
--------------	--

4.11.2.21 updateGroup()

```
void SQL::updateGroup (
    Chatroom * group )
```

updateGroup - updates the group information

Parameters

<i>group</i>	
--------------	--

4.11.2.22 updateUser()

```
void SQL::updateUser (
    User * user )
```

updateUser - updates the user information

Parameters

<i>user</i>	
-------------	--

4.11.2.23 validateTopic()

```
QString SQL::validateTopic (
    QString topic )
```

validateTopic - validates the topic name

Parameters

<i>topic</i>	
--------------	--

Returns

QString

4.12 User Class Reference

The [User](#) class.

```
#include <user.h>
```

Public Member Functions

- [User](#) (QString username, QString email, QString firstName, QString lastName)
Constructor for [User](#) class.
- int **getId** () const
- QString **username** () const
- void **setUsername** (const QString &username)
- QString **email** () const
- void **setEmail** (const QString &email)
- QString **getFirstName** () const
- void **setFirstName** (const QString &firstName)
- QString **getLastName** () const
- void **setLastName** (const QString &lastName)
- QString **getDisplayName** () const
- void **setDisplayName** (const QString &displayName)

4.12.1 Detailed Description

The [User](#) class.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 User()

```
User::User (
    QString username,
    QString email,
    QString firstName,
    QString lastName ) [explicit]
```

Constructor for [User](#) class.

Parameters

<i>username</i>	
<i>email</i>	
<i>firstName</i>	
<i>lastName</i>	

Index

- addContactByUsername
 - SQL, [25](#)
- addGroup
 - SQL, [25](#)
- addSubscription
 - SQL, [26](#)
- AddUser, [7](#)
 - AddUser, [8](#)
- addUser
 - SQL, [26](#)
- areContacts
 - SQL, [26](#)
- Auth, [8](#)
 - Auth, [9](#)
 - validateLogin, [10](#)
 - validateRegister, [10](#)
- Chatroom, [11](#)
 - Chatroom, [11](#)
- ChatroomWindow, [12](#)
 - ChatroomWindow, [13](#)
 - loadChatroom, [13](#)
- EditProfile, [14](#)
 - EditProfile, [15](#)
- getGroupsByTopic
 - SQL, [27](#)
- getMembersByTopic
 - SQL, [27](#)
- getMessageById
 - SQL, [27](#)
- getMessagesByTopic
 - SQL, [28](#)
- getSubscriptionsByUser
 - SQL, [28](#)
- getUserByEmail
 - SQL, [28](#)
- getUserByUsername
 - SQL, [29](#)
- getUserContacts
 - SQL, [29](#)
- getUserDms
 - SQL, [30](#)
- isGroupAdmin
 - SQL, [30](#)
- isValidLoginByEmail
 - SQL, [30](#)
- isValidLoginByUsername
 - SQL, [31](#)
- loadChatroom
 - ChatroomWindow, [13](#)
- MainWindow, [15](#)
 - MainWindow, [16](#)
 - newChatroomWindow, [17](#)
- Message, [17](#)
 - Message, [18](#)
- Mosquitto, [18](#)
 - Mosquitto, [19](#)
 - publish, [20](#)
 - subscribe, [20](#)
- newChatroomWindow
 - MainWindow, [17](#)
- Profile, [20](#)
 - Profile, [21](#)
- publish
 - Mosquitto, [20](#)
- removeContact
 - SQL, [31](#)
- removeGroup
 - SQL, [31](#)
- removeSubscriptions
 - SQL, [32](#)
- SaveGroup, [22](#)
 - SaveGroup, [23](#)
- SQL, [24](#)
 - addContactByUsername, [25](#)
 - addGroup, [25](#)
 - addSubscription, [26](#)
 - addUser, [26](#)
 - areContacts, [26](#)
 - getGroupsByTopic, [27](#)
 - getMembersByTopic, [27](#)
 - getMessageById, [27](#)
 - getMessagesByTopic, [28](#)
 - getSubscriptionsByUser, [28](#)
 - getUserByEmail, [28](#)
 - getUserByUsername, [29](#)
 - getUserContacts, [29](#)
 - getUserDms, [30](#)
 - isGroupAdmin, [30](#)
 - isValidLoginByEmail, [30](#)
 - isValidLoginByUsername, [31](#)
 - removeContact, [31](#)

- removeGroup, [31](#)
 - removeSubscriptions, [32](#)
 - updateGroup, [32](#)
 - updateUser, [32](#)
 - validateTopic, [32](#)
- subscribe
 - Mosquitto, [20](#)
- updateGroup
 - SQL, [32](#)
- updateUser
 - SQL, [32](#)
- User, [33](#)
 - User, [33](#)
- validateLogin
 - Auth, [10](#)
- validateRegister
 - Auth, [10](#)
- validateTopic
 - SQL, [32](#)