

To-Do-List

1.0

Generated by Doxygen 1.14.0

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 CreateUser Class Reference	7
4.2 MainTasks Class Reference	7
4.3 MainWindow Class Reference	8
4.4 Task Class Reference	8
4.5 Team Class Reference	9
4.6 User Class Reference	10
5 File Documentation	11
5.1 createuser.h	11
5.2 maintasks.h	11
5.3 mainwindow.h	13
5.4 task.h	13
5.5 taskmanager.h	14
5.6 taskstatus.h	14
5.7 team.h	15
5.8 teammanager.h	15
5.9 user.h	16
5.10 usermanager.h	16
Index	19

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

QDialog	
CreateUser	7
MainTasks	7
QMainWindow	
MainWindow	8
Task	8
Team	9
User	10

Chapter 2

Class Index

2.1 Class List

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

CreateUser	7
MainTasks	7
MainWindow	8
Task	8
Team	9
User	10

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

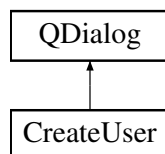
C:/Users/Krzysztof/CLionProjects/To-Do-List/ createuser.h	11
C:/Users/Krzysztof/CLionProjects/To-Do-List/ maintasks.h	11
C:/Users/Krzysztof/CLionProjects/To-Do-List/ mainwindow.h	13
C:/Users/Krzysztof/CLionProjects/To-Do-List/ task.h	13
C:/Users/Krzysztof/CLionProjects/To-Do-List/ taskmanager.h	14
C:/Users/Krzysztof/CLionProjects/To-Do-List/ taskstatus.h	14
C:/Users/Krzysztof/CLionProjects/To-Do-List/ team.h	15
C:/Users/Krzysztof/CLionProjects/To-Do-List/ teammanager.h	15
C:/Users/Krzysztof/CLionProjects/To-Do-List/ user.h	16
C:/Users/Krzysztof/CLionProjects/To-Do-List/ usermanager.h	16

Chapter 4

Class Documentation

4.1 CreateUser Class Reference

Inheritance diagram for CreateUser:



Public Member Functions

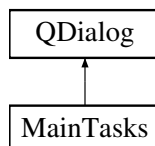
- **CreateUser** (QWidget *parent=nullptr)

The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/createuser.h
- C:/Users/Krzysztof/CLionProjects/To-Do-List/createuser.cpp

4.2 MainTasks Class Reference

Inheritance diagram for MainTasks:



Public Member Functions

- **MainTasks** (QWidget *parent=nullptr)

Protected Member Functions

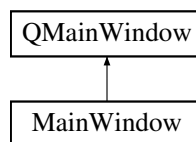
- void **resizeEvent** (QResizeEvent *event) override
- void **showEvent** (QShowEvent *event) override

The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/maintasks.h
- C:/Users/Krzysztof/CLionProjects/To-Do-List/maintasks.cpp

4.3 MainWindow Class Reference

Inheritance diagram for MainWindow:



Public Member Functions

- **MainWindow** (QWidget *parent=nullptr)

Static Public Attributes

- static [User](#) **currentUser**

The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/mainwindow.h
- C:/Users/Krzysztof/CLionProjects/To-Do-List/mainwindow.cpp

4.4 Task Class Reference

Public Member Functions

- **Task** (uint32_t id)
- **Task** (uint32_t id, const std::string &name, const std::string &description, uint8_t priority, uint32_t teamId, uint32_t userId, TaskStatus status, time_t deadline)
- **Task** (const std::string &name, const std::string &description, uint8_t priority, uint32_t teamId, uint32_t userId, TaskStatus status, time_t deadline)
- void **setId** (uint64_t id)
- void **setName** (const std::string &name)
- void **setDescription** (const std::string &description)
- void **setPriority** (uint8_t priority)
- void **setTeamId** (uint32_t teamId)

- void **setUserId** (uint32_t userId)
- void **setStatus** (int status)
- void **setStatus** (TaskStatus status)
- void **setDeadline** (time_t deadline)
- uint64_t **getId** () const
- std::string **getName** () const
- std::string **getDescription** () const
- uint8_t **getPriority** () const
- uint32_t **getTeamId** () const
- uint32_t **getUserId** () const
- int **getStatusAsInt** () const
- TaskStatus **getStatus** () const
- time_t **getDeadline** () const

The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/task.h
- C:/Users/Krzysztof/CLionProjects/To-Do-List/task.cpp

4.5 Team Class Reference

Public Member Functions

- **Team** (uint32_t id, const std::string &name, std::string &password, const std::vector< uint32_t > &members)
- **Team** (uint32_t id, const std::string &name, const std::string &password, const std::vector< uint32_t > &members)
- **Team** (uint32_t id, const std::string &name, std::string &password)
- **Team** (uint32_t id, const std::string &name, const std::string &password)
- **Team** (const std::string &name, std::string &password, const std::vector< uint32_t > &members)
- **Team** (const std::string &name, const std::string &password, const std::vector< uint32_t > &members)
- void **setId** (uint32_t id)
- void **setName** (const std::string &name)
- void **setPassword** (const std::string &password)
- void **setPassword** (std::string &password)
- void **setMembers** (const std::vector< uint32_t > &members)
- uint32_t **getId** () const
- std::string **getName** () const
- std::string **getPassword** () const
- std::vector< uint32_t > **getMembers** () const
- std::vector< [User](#) > **getMembersAsUsers** () const
- bool **containsUser** (uint32_t userid) const
- bool **containsUser** (const [User](#) &user) const
- void **addMember** (const [User](#) &user)
- void **addMember** (uint32_t userid)
- void **removeMember** (const [User](#) &user)
- void **removeMember** (uint32_t userid)

The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/team.h
- C:/Users/Krzysztof/CLionProjects/To-Do-List/team.cpp

4.6 User Class Reference

Public Member Functions

- **User** (uint32_t id, const std::string &username, const std::string &password)
- **User** (uint32_t id, const std::string &username, const std::string &password, const QDate &creationDate)
- **User** (uint32_t id, const std::string &username, std::string &password)
- **User** (uint32_t id, const std::string &username, std::string &password, const QDate &creationDate)
- **User** (const std::string &username, const std::string &password)
- **User** (const std::string &username, const std::string &password, const QDate &creationDate)
- **User** (const std::string &username, std::string &password)
- **User** (const std::string &username, std::string &password, const QDate &creationDate)
- **User** (const std::string &username)
- void **setId** (uint32_t id)
- void **setUsername** (const std::string &username)
- void **setPassword** (const std::string &password)
- void **setPassword** (std::string &password)
- void **setCreationDate** (const QDate &creationDate)
- uint32_t **getId** () const
- std::string **getUsername** () const
- std::string **getPassword** () const
- QDate **getCreationDate** () const

The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/user.h
- C:/Users/Krzysztof/CLionProjects/To-Do-List/user.cpp

Chapter 5

File Documentation

5.1 createuser.h

```
00001 #ifndef CREATEUSER_H
00002 #define CREATEUSER_H
00003
00004 #include "user.h"
00005 #include "userManager.h"
00006
00007 QT_BEGIN_NAMESPACE
00008 namespace Ui {
00009     class CreateUser;
00010 }
00011 QT_END_NAMESPACE
00012
00013 class CreateUser : public QDialog
00014 {
00015     Q_OBJECT
00016
00017 public:
00018     CreateUser(QWidget *parent = nullptr);
00019     ~CreateUser();
00020
00021 private slots:
00022
00023     // Event handlers
00024
00025     // Click handlers
00026
00027     void on_createNewAccountButton_clicked();
00028
00029 private:
00030     Ui::CreateUser *ui;
00031     bool validateInput();
00032
00033     // Deprecated
00034     // Consider using UserManager::createUser()
00035     bool createUserInDatabase(const User& user);
00036 };
00037
00038 #endif // CREATEUSER_H
```

5.2 maintasks.h

```
00001 #ifndef MAINTASKS_H
00002 #define MAINTASKS_H
00003 #include <QListWidgetItem>
00004 #include <QDialog>
00005 #include <QTimer>
00006 #include <QTime>
00007 #include <QSoundEffect>
00008 #include <QDateTime>
00009 #include <algorithm>
00010 #include <QBrush>
00011
00012 namespace Ui {
```

```

00013 class MainTasks;
00014 }
00015
00016 class MainTasks : public QDialog
00017 {
00018     Q_OBJECT
00019
00020 public:
00021     explicit MainTasks(QWidget *parent = nullptr);
00022     ~MainTasks();
00023
00024 private slots:
00025
00026     // Event handlers
00027
00028     // Click handlers
00029
00030     void on_startPomodoroButton_clicked();
00031
00032     void on_pomodoroButton_clicked();
00033
00034     void on_shortBreakButton_clicked();
00035
00036     void on_longBreakButton_clicked();
00037
00038     void on_addTaskButton_clicked();
00039
00040     void on_cancelNewTaskButton_clicked();
00041
00042     void on_updatePasswordButton_clicked();
00043
00044     void on_removeAccountButton_clicked();
00045
00046     void on_confirmTaskAddButton_clicked();
00047
00048     void on_taskListDisplay_itemDoubleClicked(QListWidgetItem *item);
00049
00050     void on_createTeamButton_clicked();
00051
00052     void on_addMembersButton_clicked();
00053
00054     void on_crateTeamCancelButton_clicked();
00055
00056     void on_addMemberCancelButton_clicked();
00057
00058     void on_leaveJoinTeamButton_clicked();
00059
00060     void on_createTeamConfirmButton_clicked();
00061
00062     void on_allTeamsComboBox_currentIndexChanged(int index);
00063
00064     void on_addMemberConfinButton_clicked();
00065
00066     void on_sortTasksComboBox_currentIndexChanged(int index);
00067
00068     // Others
00069
00073     void updateDisplay();
00074
00078     void setDisplay(int time);
00079
00083     void setTimer(int time);
00084
00088     void refreshTaskList();
00089
00093     void updateProfileStats();
00094
00098     void moveAddTaskButton();
00099
00100 private:
00101     Ui::MainTasks *ui;
00102
00103     //Timer
00104     QTimer *pomodoroTimer;
00105     int remainingTime;
00106     int startingTime;
00107     bool isRunning = false;
00108     QSoundEffect *timerEndSound;
00109
00110     //Add task button
00111     QPushButton *addTaskButton;
00112
00113     void loadAllTeamsToComboBox();
00114
00115     // Task Sorting
00116     enum TaskSortCriteria { // << Updated enum
00117         SortByDueDateAsc,

```



```

00118         SortByDueDateDesc,
00119         SortByNameAsc,
00120         SortByNameDesc
00121     };
00122     TaskSortCriteria currentTaskSortCriteria;
00123
00124 protected:
00125     void resizeEvent(QResizeEvent *event) override;
00126     void showEvent(QShowEvent *event) override;
00127 };
00128
00129 #endif // MAINTASKS_H

```

5.3 mainwindow.h

```

00001 #ifndef MAINWINDOW_H
00002 #define MAINWINDOW_H
00003
00004 #include <QMainWindow>
00005 #include "maintasks.h"
00006 #include "createuser.h"
00007 #include "user.h"
00008
00009 QT_BEGIN_NAMESPACE
00010 namespace Ui {
00011     class MainWindow;
00012 }
00013 QT_END_NAMESPACE
00014
00015 class MainWindow : public QMainWindow
00016 {
00017     Q_OBJECT
00018
00019 public:
00020     MainWindow(QWidget *parent = nullptr);
00021     ~MainWindow();
00022     static User currentUser; // Static member to store current logged user
00023
00024 private slots:
00025
00026     // Event handlers
00027
00028     // Click handlers
00029
00030     void on_loginButton_clicked();
00031     void on_registerButton_clicked();
00032
00033 private:
00034     Ui::MainWindow *ui;
00035     MainTasks *taskWindow;
00036     CreateUser *createUserWindow;
00037
00038     bool authenticateUser(const QString& username, const QString& password);
00039
00040     User getCurrentUser() const;
00041
00042 };
00043
00044 #endif // MAINWINDOW_H

```

5.4 task.h

```

00001 #ifndef TASK_H
00002 #define TASK_H
00003
00004 #include <cstdint>
00005 #include <string>
00006
00007 #include "taskstatus.h"
00008
00009 class Task
00010 {
00011 private:
00012     uint64_t id;
00013     std::string name;
00014     std::string description;
00015     uint8_t priority;

```

```

00016     uint32_t teamId; // If teamId is 0, that means it's a Task specified only for one User
00017     uint32_t userId; // If userId is 0, that means it's Task specified for Team
00018     TaskStatus status;
00019     time_t deadline; // If 0 - Task with unlimited time.
00020 public:
00021     // Constructors
00022     Task();
00023     Task(uint32_t id);
00024     Task(uint32_t id, const std::string &name, const std::string &description, uint8_t priority,
uint32_t teamId, uint32_t userId, TaskStatus status, time_t deadline);
00025     Task(const std::string &name, const std::string &description, uint8_t priority, uint32_t teamId,
uint32_t userId, TaskStatus status, time_t deadline);
00026
00027     // Setters
00028     void setId(uint64_t id);
00029     void setName(const std::string &name);
00030     void setDescription(const std::string &description);
00031     void setPriority(uint8_t priority);
00032     void setTeamId(uint32_t teamId);
00033     void setUserId(uint32_t userId);
00034     void setStatus(int status);
00035     void setStatus(TaskStatus status);
00036     void setDeadline(time_t deadline);
00037
00038     // Getters
00039     uint64_t getId() const;
00040     std::string getName() const;
00041     std::string getDescription() const;
00042     uint8_t getPriority() const;
00043     uint32_t getTeamId() const;
00044     uint32_t getUserId() const;
00045     int getStatusAsInt() const;
00046     TaskStatus getStatus() const;
00047     time_t getDeadline() const;
00048
00049 };
00050
00051 #endif // TASK_H

```

5.5 taskmanager.h

```

00001 #ifndef TASKMANAGER_H
00002 #define TASKMANAGER_H
00003 #include "task.h"
00004 #include <vector>
00005
00006 namespace TaskManager {
00007
00013     bool createTask(const Task& task);
00014
00020     std::vector<Task> getTasksForTeam(uint32_t teamId);
00021
00027     std::vector<Task> getTasksForUser(uint32_t userId);
00028
00034     Task getTask(uint64_t id);
00035
00041     bool updateTask(const Task& task);
00042
00048     bool deleteTask(const Task& task);
00049
00050 }
00051
00052 #endif // TASKMANAGER_H

```

5.6 taskstatus.h

```

00001 #ifndef TASKSTATUS_H
00002 #define TASKSTATUS_H
00003
00004 enum TaskStatus {
00005     DONE,
00006     IN_PROGRESS,
00007     NOT_DONE
00008 };
00009
00010 TaskStatus getTaskStatus(int status);
00011
00012 int getTaskStatusInt(TaskStatus status);
00013
00014 #endif // TASKSTATUS_H

```

5.7 team.h

```

00001 #ifndef TEAM_H
00002 #define TEAM_H
00003
00004 #include <cstdint>
00005 #include <string>
00006 #include <vector>
00007 #include <QCryptographicHash>
00008
00009 #include "user.h"
00010 #include "userManager.h"
00011
00012 class Team
00013 {
00014 private:
00015     uint32_t id;
00016     std::string name;
00017     std::string password;
00018     std::vector<uint32_t> members;
00019 public:
00020     // Constructors
00021     Team();
00022     Team(uint32_t id, const std::string &name, std::string &password, const std::vector<uint32_t>
&members);
00023     Team(uint32_t id, const std::string &name, const std::string &password, const
std::vector<uint32_t> &members);
00024     Team(uint32_t id, const std::string &name, std::string &password);
00025     Team(uint32_t id, const std::string &name, const std::string &password);
00026     Team(const std::string &name, std::string &password, const std::vector<uint32_t> &members);
00027     Team(const std::string &name, const std::string &password, const std::vector<uint32_t> &members);
00028
00029     // Setters
00030     void setId(uint32_t id);
00031     void setName(const std::string &name);
00032     void setPassword(const std::string &password);
00033     void setPassword(std::string &password);
00034     void setMembers(const std::vector<uint32_t> &members);
00035
00036     // Getters
00037     uint32_t getId() const;
00038     std::string getName() const;
00039     std::string getPassword() const;
00040     std::vector<uint32_t> getMembers() const;
00041     std::vector<User> getMembersAsUsers() const;
00042     bool containsUser(uint32_t userid) const;
00043     bool containsUser(const User &user) const;
00044
00045     // Adding members
00046     void addMember(const User &user);
00047     void addMember(uint32_t userid);
00048     void removeMember(const User &user);
00049     void removeMember(uint32_t userid);
00050
00051 };
00052
00053 #endif // TEAM_H

```

5.8 teammanager.h

```

00001 #ifndef TEAMMANAGER_H
00002 #define TEAMMANAGER_H
00003 #include "team.h"
00004
00005 #include <string>
00006 #include <QMessageBox>
00007 #include <QSqlDatabase>
00008 #include <QSqlQuery>
00009 #include <QSqlError>
00010
00011 namespace TeamManager {
00012
00013     bool createTeam(const Team& team);
00014
00015     Team getTeam(const std::string& name);
00016
00017     Team getTeam(uint32_t id);
00018
00019     bool updateTeam(const Team& team);
00020
00021     bool deleteTeam(const Team& team);
00022
00023 }

```

```

00053     bool deleteTeam(uint32_t id);
00054
00055
00060     std::vector<Team> getAllTeams();
00061
00066     std::vector<Team> getTeamsForUser(uint32_t userId);
00067
00072     Team getTeamForUser(uint32_t userId);
00073
00074
00075
00076 }
00077
00078 #endif // TEAMMANAGER_H

```

5.9 user.h

```

00001 #ifndef USER_H
00002 #define USER_H
00003
00004 #include <stdint>
00005 #include <string>
00006 #include <QDate>
00007 #include <QCryptographicHash>
00008
00009 class User
00010 {
00011 private:
00012     uint32_t id;
00013     std::string username;
00014     std::string password;
00015     QDate creationDate;
00016 public:
00017     // Constructors
00018     User();
00019     User(uint32_t id, const std::string& username, const std::string &password);
00020     User(uint32_t id, const std::string& username, const std::string &password, const QDate
&creationDate);
00021     User(uint32_t id, const std::string& username, std::string &password);
00022     User(uint32_t id, const std::string& username, std::string &password, const QDate &creationDate);
00023     User(const std::string& username, const std::string &password);
00024     User(const std::string& username, const std::string &password, const QDate &creationDate);
00025     User(const std::string& username, std::string &password);
00026     User(const std::string& username, std::string &password, const QDate &creationDate);
00027     explicit User(const std::string& username);
00028
00029     // Setters
00030     void setId(uint32_t id);
00031     void setUsername(const std::string &username);
00032     void setPassword(const std::string &password);
00033     void setPassword(std::string &password);
00034     void setCreationDate(const QDate &creationDate);
00035
00036     // Getters
00037     uint32_t getId() const;
00038     std::string getUsername() const; // Make const
00039     std::string getPassword() const;
00040     QDate getCreationDate() const;
00041 };
00042
00043 #endif // USER_H

```

5.10 usermanager.h

```

00001 #ifndef USERMANAGER_H
00002 #define USERMANAGER_H
00003 #include <QMessageBox>
00004 #include <QSqlDatabase>
00005 #include <QSqlQuery>
00006 #include <QSqlError>
00007
00008 #include "user.h"
00009
00010 namespace UserManager {
00016     bool createUser(const User& user);
00017
00023     User getUser(const std::string& username);
00024

```

```
00030     User getUser(uint32_t id);
00031
00037     bool updateUser(const User& user);
00038
00044     bool deleteUser(const User& user);
00045
00051     bool deleteUser(uint32_t id);
00052
00057     std::vector<User> getAllUsers();
00058
00059
00060 }
00061
00062 #endif // USERMANAGER_H
```


Index

C:/Users/Krzysztof/CLionProjects/To-Do-List/createuser.h,
[11](#)
C:/Users/Krzysztof/CLionProjects/To-Do-List/maintasks.h,
[11](#)
C:/Users/Krzysztof/CLionProjects/To-Do-List/mainwindow.h,
[13](#)
C:/Users/Krzysztof/CLionProjects/To-Do-List/task.h, [13](#)
C:/Users/Krzysztof/CLionProjects/To-Do-List/taskmanager.h,
[14](#)
C:/Users/Krzysztof/CLionProjects/To-Do-List/taskstatus.h,
[14](#)
C:/Users/Krzysztof/CLionProjects/To-Do-List/team.h, [15](#)
C:/Users/Krzysztof/CLionProjects/To-Do-List/teammanager.h,
[15](#)
C:/Users/Krzysztof/CLionProjects/To-Do-List/user.h, [16](#)
C:/Users/Krzysztof/CLionProjects/To-Do-List/usermanager.h,
[16](#)
CreateUser, [7](#)

MainTasks, [7](#)
MainWindow, [8](#)

Task, [8](#)
Team, [9](#)

User, [10](#)