To-Do-List

1.0

Generated by Doxygen 1.14.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

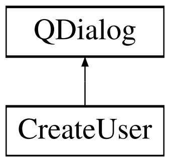
Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 CreateUser Class Reference

Inheritance diagram for CreateUser:



Public Member Functions

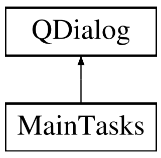
- **CreateUser** (QWidget ∗parent=nullptr)

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

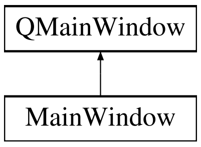
- void **resizeEvent** (QResizeEvent ∗event) override
- void **showEvent** (QShowEvent ∗event) override

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

```
#ifndef CREATEUSER_H
#define CREATEUSER_H

#include "user.h"
#include "usermanager.h"

QT_BEGIN_NAMESPACE
namespace Ui {
    class CreateUser;
}
QT_END_NAMESPACE

class CreateUser : public QDialog
{
    Q_OBJECT

public:
    CreateUser(QWidget *parent = nullptr);
    ~CreateUser();

private slots:

    // Event handlers

    // Click handlers

    void on_createNewAccountButton_clicked();

private:
    Ui::CreateUser *ui;
    bool validateInput();

    // Deprecated
    // Consider using UserManager::createUser()
    bool createUserInDatabase(const User& user);
};

#endif // CREATEUSER_H
```

5.2 maintasks.h

```
#ifndef MAINTASKS_H
#define MAINTASKS_H
#include <QListWidgetItem>
#include <QDialog>
#include <QTimer>
#include <QTime>
#include <QSoundEffect>
#include <QDateTime>
#include <algorithm>
#include <QBrush>

namespace Ui {
```

```
class MainTasks;
}

class MainTasks : public QDialog
{
    Q_OBJECT

public:
    explicit MainTasks(QWidget *parent = nullptr);
    ~MainTasks();

private slots:

    // Event handlers

    // Click handlers

    void on_startPomodoroButton_clicked();

    void on_pomodoroButton_clicked();

    void on_shortBreakButton_clicked();

    void on_longBreakButton_clicked();

    void on_addTaskButton_clicked();

    void on_cancelNewTaskButton_clicked();

    void on_updatePasswordButton_clicked();

    void on_removeAccountButton_clicked();

    void on_confirmTaskAddButton_clicked();

    void on_taskListDisplay_itemDoubleClicked(QListWidgetItem *item);

    void on_createTeamButton_clicked();

    void on_addMembersButton_clicked();

    void on_crateTeamCancelButton_clicked();

    void on_addMemberCancelButton_clicked();

    void on_leaveJoinTeamButton_clicked();

    void on_createTeamConfirmButton_clicked();

    void on_allTeamsComboBox_currentIndexChanged(int index);

    void on_addMemberConfimButton_clicked();

    void on_sortTasksComboBox_currentIndexChanged(int index);

    // Others

    void updateDisplay();

    void setDisplay(int time);

    void setTimer(int time);

    void refreshTaskList();

    void updateProfileStats();

    void moveAddTaskButton();

private:
    Ui::MainTasks *ui;

    //Timer
    QTimer *pomodoroTimer;
    int remainingTime;
    int startingTime;
    bool isRunning = false;
    QSoundEffect *timerEndSound;

    //Add task button
    QPushButton *addTaskButton;

    void loadAllTeamsToComboBox();

    // Task Sorting
    enum TaskSortCriteria { // «< Updated enum
        SortByDueDateAsc,
        SortByDueDateDesc,
        SortByNameAsc,
        SortByNameDesc
    };
    TaskSortCriteria currentTaskSortCriteria;

protected:
    void resizeEvent(QResizeEvent *event) override;
    void showEvent(QShowEvent *event) override;
};

#endif // MAINTASKS_H
```

5.3 mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "maintasks.h"
#include "createuser.h"
#include "user.h"

QT_BEGIN_NAMESPACE
namespace Ui {
    class MainWindow;
}
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    static User currentUser; // Static member to store current logged user

private slots:

    // Event handlers

    // Click handlers

    void on_loginButton_clicked();
    void on_registerButton_clicked();

private:
    Ui::MainWindow *ui;
    MainTasks *taskWindow;
    CreateUser *createUserWindow;

    bool authenticateUser(const QString& username, const QString& password);

    User getCurrentUser() const;


};

#endif // MAINWINDOW_H
```

5.4 task.h

```
#ifndef TASK_H
#define TASK_H

#include <cstdint>
#include <string>

#include "taskstatus.h"

class Task
{
private:
    uint64_t id;
    std::string name;
    std::string description;
    uint8_t priority;
```

```cpp
    uint32_t teamId; // If teamId is 0, that means it’s a Task specified only for one User
    uint32_t userId; // If userId is 0, that means it’s Task specified for Team
    TaskStatus status;
    time_t deadline; // If 0 - Task with unlimited time.
public:
    // Constructors
    Task();
    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);

    // Setters
    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);

    // Getters
    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;

};

#endif // TASK_H
```

5.5 taskmanager.h

```cpp
#ifndef TASKMANAGER_H
#define TASKMANAGER_H
#include "task.h"
#include <vector>

namespace TaskManager {

    bool createTask(const Task& task);

    std::vector<Task> getTasksForTeam(uint32_t teamId);

    std::vector<Task> getTasksForUser(uint32_t userId);

    Task getTask(uint64_t id);

    bool updateTask(const Task& task);

    bool deleteTask(const Task& task);

}

#endif // TASKMANAGER_H
```

5.6 taskstatus.h

```cpp
#ifndef TASKSTATUS_H
#define TASKSTATUS_H

enum TaskStatus {
    DONE,
    IN_PROGRESS,
    NOT_DONE
};

TaskStatus getTaskStatus(int status);

int getTaskStatusInt(TaskStatus status);

#endif // TASKSTATUS_H
```

5.7 team.h

```
#ifndef TEAM_H
#define TEAM_H

#include <cstdint>
#include <string>
#include <vector>
#include <QCryptographicHash>

#include "user.h"
#include "usermanager.h"

class Team
{
private:
    uint32_t id;
    std::string name;
    std::string password;
    std::vector<uint32_t> members;
public:
    // Constructors
    Team();
    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);

    // Setters
    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);

    // Getters
    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;

    // Adding members
    void addMember(const User &user);
    void addMember(uint32_t userid);
    void removeMember(const User &user);
    void removeMember(uint32_t userid);

};

#endif // TEAM_H
```

5.8 teammanager.h

```
#ifndef TEAMMANAGER_H
#define TEAMMANAGER_H
#include "team.h"

#include <string>
#include <QMessageBox>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>

namespace TeamManager {

    bool createTeam(const Team& team);

    Team getTeam(const std::string& name);

    Team getTeam(uint32_t id);

    bool updateTeam(const Team& team);

    bool deleteTeam(const Team& team);

    bool deleteTeam(uint32_t id);



    std::vector<Team> getAllTeams();

    std::vector<Team> getTeamsForUser(uint32_t userId);

    Team getTeamForUser(uint32_t userId);



}

#endif // TEAMMANAGER_H
```

5.9 user.h

```
#ifndef USER_H
#define USER_H

#include <cstdint>
#include <string>
#include <QDate>
#include <QCryptographicHash>

class User
{
private:
    uint32_t id;
    std::string username;
    std::string password;
    QDate creationDate;
public:
    // Constructors
    User();
    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);
    explicit User(const std::string& username);

    // Setters
    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);

    // Getters
    uint32_t getId() const;
    std::string getUsername() const; // Make const
    std::string getPassword() const;
    QDate getCreationDate() const;
};

#endif // USER_H
```

5.10 usermanager.h

```
#ifndef USERMANAGER_H
#define USERMANAGER_H
#include <QMessageBox>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>

#include "user.h"

namespace UserManager {
    bool createUser(const User& user);

    User getUser(const std::string& username);

    User getUser(uint32_t id);

    bool updateUser(const User& user);

    bool deleteUser(const User& user);

    bool deleteUser(uint32_t id);

    std::vector<User> getAllUsers();


}

#endif // USERMANAGER_H
```

Index