

To-Do-List

1.0

Generated by Doxygen 1.14.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

TaskManager	??
TeamManager	??
Ui	??
UserManager	??

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

QDialog	
CreateUser	??
MainTasks	??
QMainWindow	
MainWindow	??
Task	??
Team	??
User	??

Chapter 3

Class Index

3.1 Class List

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

CreateUser	??
MainTasks	??
MainWindow	??
Task	??
Team	??
User	??

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

C:/Users/Krzysztof/CLionProjects/To-Do-List/ createuser.cpp	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ createuser.h	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ main.cpp	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ maintasks.cpp	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ maintasks.h	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ mainwindow.cpp	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ mainwindow.h	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ task.cpp	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ task.h	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ taskmanager.cpp	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ taskmanager.h	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ taskstatus.cpp	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ taskstatus.h	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ team.cpp	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ team.h	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ teammanager.cpp	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ teammanager.h	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ user.cpp	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ user.h	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ usermanager.cpp	??
C:/Users/Krzysztof/CLionProjects/To-Do-List/ usermanager.h	??

Chapter 5

Namespace Documentation

5.1 TaskManager Namespace Reference

Functions

- `bool createTask (const Task &task)`
Creates `Task` in a database.
- `std::vector< Task > getTasksForTeam (uint32_t teamId)`
Returns all `Team`'s `Task`.
- `std::vector< Task > getTasksForUser (uint32_t userId)`
Returns all `User`'s `Task`.
- `Task getTask (uint64_t id)`
Gets `Task` from a database identifying it by ID and returns as `Task`'s object.
- `bool updateTask (const Task &task)`
Updates `Task` in a database.
- `bool deleteTask (const Task &task)`
Deletes `Task` from a database.

5.1.1 Function Documentation

5.1.1.1 createTask()

```
bool TaskManager::createTask (  
    const Task & task)
```

Creates `Task` in a database.

Parameters

<code>task</code>	<code>Task</code> to create in a database
-------------------	---

Returns

True if success, false if there was any error

5.1.1.2 deleteTask()

```
bool TaskManager::deleteTask (  
    const Task & task)
```

Deletes `Task` from a database.

Parameters

task	Task to delete
------	----------------

Returns

True if success, false if Task doesn't exist or there was any error

5.1.1.3 getTask()

```
Task TaskManager::getTask (
    uint64_t id)
```

Gets Task from a database identifying it by ID and returns as Task's object.

Parameters

id	Task's ID
----	-----------

Returns

Task's object, if Task->id is 0, then Task doesn't exist, or there was an error

5.1.1.4 getTasksForTeam()

```
std::vector< Task > TaskManager::getTasksForTeam (
    uint32_t teamId)
```

Returns all Team's Task.

Parameters

teamId	Team's ID
--------	-----------

Returns

Vector of Team's Tasks

5.1.1.5 getTasksForUser()

```
std::vector< Task > TaskManager::getTasksForUser (
    uint32_t userId)
```

Returns all User's Task.

Parameters

<code>userId</code>	<code>User's ID</code>
---------------------	------------------------

Returns

Vector of `User's Tasks`

5.1.1.6 `updateTask()`

```
bool TaskManager::updateTask (
    const Task & task)
```

Updates `Task` in a database.

Parameters

<code>task</code>	<code>Task to update</code>
-------------------	-----------------------------

Returns

True if success, false if there was any error

5.2 TeamManager Namespace Reference

Functions

- `bool createTeam (const Team &team)`
Creates `Team` in a database.
- `Team getTeam (const std::string &name)`
Gets `Team` from a database identifying it by name and returns as `Team's` object.
- `Team getTeam (uint32_t id)`
Gets `Team` from a database identifying it by ID and returns as `Team's` object.
- `bool updateTeam (const Team &team)`
Updates `Team` in a database.
- `bool deleteTeam (const Team &team)`
Deletes `Team` from a database.
- `bool deleteTeam (uint32_t id)`
Deletes `Team` using his ID from a database.
- `std::vector< Team > getAllTeams ()`
Returns all Teams from a database.
- `std::vector< Team > getTeamsForUser (uint32_t userId)`
Returns all Teams that the `User` belongs to from a database.
- `Team getTeamForUser (uint32_t userId)`
Returns first `User's Team` from a database.

5.2.1 Function Documentation

5.2.1.1 `createTeam()`

```
bool TeamManager::createTeam (
    const Team & team)
```

Creates `Team` in a database.

Creates `Team` in database.

Parameters

	team	Team to create
--	------	----------------

Returns

True if success, false if there was any error

5.2.1.2 deleteTeam() [1/2]

```
bool TeamManager::deleteTeam (
    const Team & team)
```

Deletes Team from a database.

Parameters

	team	Team to delete
--	------	----------------

Returns

True if success, false if Team doesn't exist or there was any error

5.2.1.3 deleteTeam() [2/2]

```
bool TeamManager::deleteTeam (
    uint32_t id)
```

Deletes Team using his ID from a database.

Parameters

	id	ID of a Team to delete
--	----	------------------------

Returns

True if success, false if Team doesn't exist or there was any error

5.2.1.4 getAllTeams()

```
std::vector< Team > TeamManager::getAllTeams ()
```

Returns all Teams from a database.

Returns

Vector of all Teams from a database

5.2.1.5 getTeam() [1/2]

```
Team TeamManager::getTeam (
    const std::string & name)
```

Gets Team from a database identifying it by name and returns as Team's object.

Parameters

name	Team's name
------	-------------

Returns

Team's object, if Team->id equals 0, then Team doesn't exist, or there was an error

5.2.1.6 getTeam() [2/2]

Team TeamManager::getTeam (
uint32_t id)

Gets Team from a database identifying it by ID and returns as Team's object.

Parameters

id	Team's ID
----	-----------

Returns

Team's object, if Team->id equals 0, then Team doesn't exist, or there was an error

5.2.1.7 getTeamForUser()

Team TeamManager::getTeamForUser (
uint32_t userId)

Returns first User's Team from a database.

Returns

First User's Team

5.2.1.8 getTeamsForUser()

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

Returns all Teams that the User belongs to from a database.

Returns

Vector of all Teams that the User belongs to

5.2.1.9 updateTeam()

bool TeamManager::updateTeam (
const Team & team)

Updates Team in a database.

Parameters

team	Team to update
------	----------------

Returns

True if success, false if there was any error

5.3 Ui Namespace Reference

5.4 UserManager Namespace Reference

Functions

- bool `createUser` (const `User` &user)
Creates `User` in a database.
- `User` `getUser` (const std::string &username)
Gets `User` from a database identifying him by username and returns as `User`'s object.
- `User` `getUser` (uint32_t id)
Gets `User` from a database identifying him by username and returns as `User`'s object.
- bool `updateUser` (const `User` &user)
Updates `User` in a database.
- bool `deleteUser` (const `User` &user)
Deletes `User` from a database.
- bool `deleteUser` (uint32_t id)
Deletes `User` using his ID from a database.
- std::vector< `User` > `getAllUsers` ()
Returns all Users from a database.

5.4.1 Function Documentation

5.4.1.1 createUser()

```
bool UserManager::createUser (
    const User & user)
```

Creates `User` in a database.

Parameters

user	User to create in a database
------	------------------------------

Returns

True if success, false if there was any error

5.4.1.2 deleteUser() [1/2]

```
bool UserManager::deleteUser (
    const User & user)
```

Deletes `User` from a database.

Parameters

	user	User to delete
--	------	----------------

Returns

True if success, false if User doesn't exist or there was any error

5.4.1.3 deleteUser() [2/2]

```
bool UserManager::deleteUser (
    uint32_t id)
```

Deletes User using his ID from a database.

Parameters

	id	ID of a User to delete
--	----	------------------------

Returns

True if success, false if User doesn't exist or there was any error

5.4.1.4 getAllUsers()

```
std::vector< User > UserManager::getAllUsers ()
```

Returns all Users from a database.

Returns

Vector of all Users from a database

5.4.1.5 getUser() [1/2]

```
User UserManager::getUser (
    const std::string & username)
```

Gets User from a database identifying him by username and returns as User's object.

Parameters

	username	User's name
--	----------	-------------

Returns

User's object, if User->id equals 0, then User doesn't exist, or there was an error

Parameters

	username	User's name
--	----------	-------------

Returns

User's object, if User->id is 0, then User doesn't exist or there was an error

5.4.1.6 getUser() [2/2]

```
User UserManager::getUser (
    uint32_t id)
```

Gets User from a database identifying him by username and returns as User's object.

Gets User from a database identifying him by ID and returns as User's object.

Parameters

	username	User's name
--	----------	-------------

Returns

User's object, if User->id is 0, then User doesn't exist, or there was an error

Parameters

	id	User's ID
--	----	-----------

Returns

User's object, if User->id is 0, then User doesn't exist or there was an error

5.4.1.7 updateUser()

```
bool UserManager::updateUser (
    const User & user)
```

Updates User in a database.

Parameters

	user	User to update
--	------	----------------

Returns

True if success, false if there was any error

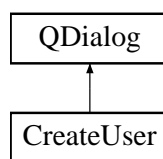
Chapter 6

Class Documentation

6.1 CreateUser Class Reference

```
#include <createuser.h>
```

Inheritance diagram for CreateUser:



Public Member Functions

- [CreateUser](#) (QWidget *parent=nullptr)
- [~CreateUser](#) ()

6.1.1 Constructor & Destructor Documentation

6.1.1.1 CreateUser()

```
CreateUser::CreateUser (  
    QWidget * parent = nullptr)
```

6.1.1.2 ~CreateUser()

```
CreateUser::~~CreateUser ()
```

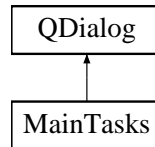
The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/[createuser.h](#)
- C:/Users/Krzysztof/CLionProjects/To-Do-List/[createuser.cpp](#)

6.2 MainTasks Class Reference

```
#include <maintasks.h>
```

Inheritance diagram for MainTasks:



Public Member Functions

- [MainTasks](#) (QWidget *parent=nullptr)
- [~MainTasks](#) ()

Protected Member Functions

- void [resizeEvent](#) (QResizeEvent *event) override
- void [showEvent](#) (QShowEvent *event) override

6.2.1 Constructor & Destructor Documentation

6.2.1.1 MainTasks()

```
MainTasks::MainTasks (  
    QWidget * parent = nullptr) [explicit]
```

6.2.1.2 ~MainTasks()

```
MainTasks::~MainTasks ()
```

6.2.2 Member Function Documentation

6.2.2.1 resizeEvent()

```
void MainTasks::resizeEvent (  
    QResizeEvent * event) [override], [protected]
```

6.2.2.2 showEvent()

```
void MainTasks::showEvent (  
    QShowEvent * event) [override], [protected]
```

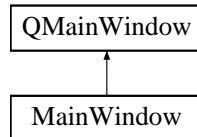
The documentation for this class was generated from the following files:

- C:/Users/Krzysztof/CLionProjects/To-Do-List/[maintasks.h](#)
- C:/Users/Krzysztof/CLionProjects/To-Do-List/[maintasks.cpp](#)

6.3 MainWindow Class Reference

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

Inheritance diagram for MainWindow:



Public Member Functions

- [MainWindow](#) (QWidget *parent=nullptr)
- [~MainWindow](#) ()

Static Public Attributes

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

6.3.1 Constructor & Destructor Documentation

6.3.1.1 MainWindow()

```
MainWindow::MainWindow (  
    QWidget * parent = nullptr)
```

6.3.1.2 ~MainWindow()

```
MainWindow::~MainWindow ()
```

6.3.2 Member Data Documentation

6.3.2.1 currentUser

```
User MainWindow::currentUser [static]
```

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](#)

6.4 Task Class Reference

```
#include <task.h>
```

Public Member Functions

- [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)
- 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

6.4.1 Constructor & Destructor Documentation

6.4.1.1 [Task\(\)](#) [1/4]

`Task::Task ()`

6.4.1.2 [Task\(\)](#) [2/4]

`Task::Task (`
 uint32_t id)

6.4.1.3 [Task\(\)](#) [3/4]

`Task::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)

6.4.1.4 Task() [4/4]

```
Task::Task (  
    const std::string & name,  
    const std::string & description,  
    uint8_t priority,  
    uint32_t teamId,  
    uint32_t userId,  
    TaskStatus status,  
    time_t deadline)
```

6.4.2 Member Function Documentation

6.4.2.1 getDeadline()

```
time_t Task::getDeadline () const
```

6.4.2.2 getDescription()

```
std::string Task::getDescription () const
```

6.4.2.3 getId()

```
uint64_t Task::getId () const
```

6.4.2.4 getName()

```
std::string Task::getName () const
```

6.4.2.5 getPriority()

```
uint8_t Task::getPriority () const
```

6.4.2.6 getStatus()

```
TaskStatus Task::getStatus () const
```

6.4.2.7 getStatusAsInt()

```
int Task::getStatusAsInt () const
```

6.4.2.8 getTeamId()

```
uint32_t Task::getTeamId () const
```

6.4.2.9 getUserId()

```
uint32_t Task::getUserId () const
```

6.4.2.10 setDeadline()

```
void Task::setDeadline (  
    time_t deadline)
```

6.4.2.11 setDescription()

```
void Task::setDescription (  
    const std::string & description)
```

6.4.2.12 setId()

```
void Task::setId (  
    uint64_t id)
```

6.4.2.13 setName()

```
void Task::setName (  
    const std::string & name)
```

6.4.2.14 setPriority()

```
void Task::setPriority (  
    uint8_t priority)
```

6.4.2.15 setStatus() [1/2]

```
void Task::setStatus (  
    int status)
```

6.4.2.16 setStatus() [2/2]

```
void Task::setStatus (  
    TaskStatus status)
```

6.4.2.17 setTeamId()

```
void Task::setTeamId (  
    uint32_t teamId)
```


6.4.2.18 setId()

```
void Task::setId (
    uint32_t userId)
```

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](#)

6.5 Team Class Reference

```
#include <team.h>
```

Public Member Functions

- [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)
- 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)

6.5.1 Constructor & Destructor Documentation

6.5.1.1 Team() [1/7]

```
Team::Team ()
```

6.5.1.2 Team() [2/7]

```
Team::Team (  
    uint32_t id,  
    const std::string & name,  
    std::string & password,  
    const std::vector< uint32_t > & members)
```

6.5.1.3 Team() [3/7]

```
Team::Team (  
    uint32_t id,  
    const std::string & name,  
    const std::string & password,  
    const std::vector< uint32_t > & members)
```

6.5.1.4 Team() [4/7]

```
Team::Team (  
    uint32_t id,  
    const std::string & name,  
    std::string & password)
```

6.5.1.5 Team() [5/7]

```
Team::Team (  
    uint32_t id,  
    const std::string & name,  
    const std::string & password)
```

6.5.1.6 Team() [6/7]

```
Team::Team (  
    const std::string & name,  
    std::string & password,  
    const std::vector< uint32_t > & members)
```

6.5.1.7 Team() [7/7]

```
Team::Team (  
    const std::string & name,  
    const std::string & password,  
    const std::vector< uint32_t > & members)
```

6.5.2 Member Function Documentation

6.5.2.1 addMember() [1/2]

```
void Team::addMember (  
    const User & user)
```

6.5.2.2 addMember() [2/2]

```
void Team::addMember (  
    uint32_t userid)
```

6.5.2.3 containsUser() [1/2]

```
bool Team::containsUser (  
    const User & user) const
```

6.5.2.4 containsUser() [2/2]

```
bool Team::containsUser (  
    uint32_t userid) const
```

6.5.2.5 getId()

```
uint32_t Team::getId () const
```

6.5.2.6 getMembers()

```
std::vector< uint32_t > Team::getMembers () const
```

6.5.2.7 getMembersAsUsers()

```
std::vector< User > Team::getMembersAsUsers () const
```

6.5.2.8 getName()

```
std::string Team::getName () const
```

6.5.2.9 getPassword()

```
std::string Team::getPassword () const
```

6.5.2.10 removeMember() [1/2]

```
void Team::removeMember (  
    const User & user)
```

6.5.2.11 removeMember() [2/2]

```
void Team::removeMember (  
    uint32_t userid)
```

6.5.2.12 setId()

```
void Team::setId (
    uint32_t id)
```

6.5.2.13 setMembers()

```
void Team::setMembers (
    const std::vector< uint32_t > & members)
```

6.5.2.14 setName()

```
void Team::setName (
    const std::string & name)
```

6.5.2.15 setPassword() [1/2]

```
void Team::setPassword (
    const std::string & password)
```

6.5.2.16 setPassword() [2/2]

```
void Team::setPassword (
    std::string & password)
```

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](#)

6.6 User Class Reference

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

Public Member Functions

- [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)
- [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

6.6.1 Constructor & Destructor Documentation

6.6.1.1 User() [1/10]

```
User::User ()
```

6.6.1.2 User() [2/10]

```
User::User (  
    uint32_t id,  
    const std::string & username,  
    const std::string & password)
```

6.6.1.3 User() [3/10]

```
User::User (  
    uint32_t id,  
    const std::string & username,  
    const std::string & password,  
    const QDate & creationDate)
```

6.6.1.4 User() [4/10]

```
User::User (  
    uint32_t id,  
    const std::string & username,  
    std::string & password)
```

6.6.1.5 User() [5/10]

```
User::User (  
    uint32_t id,  
    const std::string & username,  
    std::string & password,  
    const QDate & creationDate)
```

6.6.1.6 User() [6/10]

```
User::User (  
    const std::string & username,  
    const std::string & password)
```

6.6.1.7 User() [7/10]

```
User::User (  
    const std::string & username,  
    const std::string & password,  
    const QDate & creationDate)
```

6.6.1.8 User() [8/10]

```
User::User (  
    const std::string & username,  
    std::string & password)
```

6.6.1.9 User() [9/10]

```
User::User (  
    const std::string & username,  
    std::string & password,  
    const QDate & creationDate)
```

6.6.1.10 User() [10/10]

```
User::User (  
    const std::string & username) [explicit]
```

6.6.2 Member Function Documentation

6.6.2.1 getCreationDate()

```
QDate User::getCreationDate () const
```

6.6.2.2 getId()

```
uint32_t User::getId () const
```

6.6.2.3 getPassword()

```
std::string User::getPassword () const
```

6.6.2.4 getUsername()

```
std::string User::getUsername () const
```

6.6.2.5 setCreationDate()

```
void User::setCreationDate (  
    const QDate & creationDate)
```

6.6.2.6 setId()

```
void User::setId (  
    uint32_t id)
```

6.6.2.7 setPassword() [1/2]

```
void User::setPassword (  
    const std::string & password)
```

6.6.2.8 setPassword() [2/2]

```
void User::setPassword (  
    std::string & password)
```

6.6.2.9 setUsername()

```
void User::setUsername (  
    const std::string & username)
```

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 7

File Documentation

7.1 C:/Users/Krzysztof/CLionProjects/To-Do-List/createuser.cpp File Reference

```
#include "createuser.h"  
#include "ui_createuser.h"
```

7.2 C:/Users/Krzysztof/CLionProjects/To-Do-List/createuser.h File Reference

```
#include "user.h"  
#include "userManager.h"
```

Classes

- class [CreateUser](#)

Namespaces

- namespace [Ui](#)

7.3 createuser.h

[Go to the documentation of this file.](#)

```

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

```

7.4 C:/Users/Krzysztof/CLionProjects/To-Do-List/main.cpp File Reference

```

#include "mainwindow.h"
#include <QApplication>
#include <QIcon>
#include <QFile>
#include <QSqlDatabase>
#include <QSqlError>
#include <QSqlQuery>

```

Functions

- int [main](#) (int argc, char *argv[])

7.4.1 Function Documentation

7.4.1.1 main()

```

int main (
    int argc,
    char * argv[])

```

7.5 C:/Users/Krzysztof/CLionProjects/To-Do-List/maintasks.cpp File Reference

```
#include "maintasks.h"
#include "ui_maintasks.h"
#include "mainwindow.h"
#include "taskmanager.h"
#include <QListWidgetItem>
#include <QSqlQuery>
#include <QSqlError>
#include "teammanager.h"
#include <QStringListModel>
#include <QInputDialog>
```

7.6 C:/Users/Krzysztof/CLionProjects/To-Do-List/maintasks.h File Reference

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

Classes

- class [MainTasks](#)

Namespaces

- namespace [Ui](#)

7.7 maintasks.h

[Go to the documentation of this file.](#)

```
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_addMemberConfimButton_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

```

7.8 C:/Users/Krzysztof/CLionProjects/To-Do-List/mainwindow.cpp File Reference

```

#include "mainwindow.h"
#include "../ui_mainwindow.h"
#include "maintasks.h"
#include <QPixmap>
#include "createuser.h"

```

7.9 C:/Users/Krzysztof/CLionProjects/To-Do-List/mainwindow.h File Reference

```

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

```

Classes

- class [MainWindow](#)

Namespaces

- namespace [Ui](#)

7.10 mainwindow.h

[Go to the documentation of this file.](#)

```

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
00045     bool authenticateUser(const QString& username, const QString& password);
00046
00052     User getCurrentUser() const;
00053
00054
00055 };
00056
00057 #endif // MAINWINDOW_H

```

7.11 C:/Users/Krzysztof/CLionProjects/To-Do-List/task.cpp File Reference

```
#include "task.h"
```

7.12 C:/Users/Krzysztof/CLionProjects/To-Do-List/task.h File Reference

```

#include <cstdint>
#include <string>
#include "taskstatus.h"

```

Classes

- class [Task](#)

7.13 task.h

[Go to the documentation of this file.](#)

```

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

```

7.14 C:/Users/Krzysztof/CLionProjects/To-Do-List/taskmanager.cpp File Reference

```

#include "taskmanager.h"
#include <QStringQuery>
#include <QStringError>
#include <QMessageBox>
#include <QDateTime>
#include <vector>

```

7.15 C:/Users/Krzysztof/CLionProjects/To-Do-List/taskmanager.h File Reference

```

#include "task.h"
#include <vector>

```

Namespaces

- namespace [TaskManager](#)

Functions

- `bool TaskManager::createTask (const Task &task)`
Creates `Task` in a database.
- `std::vector< Task > TaskManager::getTasksForTeam (uint32_t teamId)`
Returns all `Team`'s `Task`.
- `std::vector< Task > TaskManager::getTasksForUser (uint32_t userId)`
Returns all `User`'s `Task`.
- `Task TaskManager::getTask (uint64_t id)`
Gets `Task` from a database identifying it by ID and returns as `Task`'s object.
- `bool TaskManager::updateTask (const Task &task)`
Updates `Task` in a database.
- `bool TaskManager::deleteTask (const Task &task)`
Deletes `Task` from a database.

7.16 taskmanager.h

[Go to the documentation of this file.](#)

```
00001 #ifndef TASKMANAGER_H
00002 #define TASKMANAGER_H
00003 #include "task.h"
00004 #include <vector>
00005
00006 namespace TaskManager {
00007
00014     bool createTask(const Task& task);
00015
00022     std::vector<Task> getTasksForTeam(uint32_t teamId);
00023
00030     std::vector<Task> getTasksForUser(uint32_t userId);
00031
00038     Task getTask(uint64_t id);
00039
00046     bool updateTask(const Task& task);
00047
00054     bool deleteTask(const Task& task);
00055
00056 }
00057
00058 #endif // TASKMANAGER_H
```

7.17 C:/Users/Krzysztof/CLionProjects/To-Do-List/taskstatus.cpp File Reference

```
#include "taskstatus.h"
```

Functions

- `TaskStatus getTaskStatus (int status)`
- `int getTaskStatusInt (TaskStatus status)`

7.17.1 Function Documentation

7.17.1.1 `getTaskStatus()`

`TaskStatus` `getTaskStatus` (
 int status)

7.17.1.2 `getTaskStatusInt()`

int `getTaskStatusInt` (
 [TaskStatus](#) status)

7.18 C:/Users/Krzysztof/CLionProjects/To-Do-List/taskstatus.h File Reference

Enumerations

- enum `TaskStatus` { `DONE` , `IN_PROGRESS` , `NOT_DONE` }

Functions

- `TaskStatus` `getTaskStatus` (int status)
- int `getTaskStatusInt` (`TaskStatus` status)

7.18.1 Enumeration Type Documentation

7.18.1.1 `TaskStatus`

enum `TaskStatus`

Enumerator

	<code>DONE</code>
<code>IN_PROGRESS</code>	
	<code>NOT_DONE</code>

7.18.2 Function Documentation

7.18.2.1 `getTaskStatus()`

`TaskStatus` `getTaskStatus` (
 int status)

7.18.2.2 getTaskStatusInt()

```
int getTaskStatusInt (  
    TaskStatus status)
```

7.19 taskstatus.h

[Go to the documentation of this file.](#)

```
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
```

7.20 C:/Users/Krzysztof/CLionProjects/To-Do-List/team.cpp File Reference

```
#include "team.h"
```

7.21 C:/Users/Krzysztof/CLionProjects/To-Do-List/team.h File Reference

```
#include <cstdint>  
#include <string>  
#include <vector>  
#include <QCryptographicHash>  
#include "user.h"  
#include "userManager.h"
```

Classes

- class [Team](#)

7.22 team.h

[Go to the documentation of this file.](#)

```

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

```

7.23 C:/Users/Krzysztof/CLionProjects/To-Do-List/teammanager.cpp

File Reference

```
#include "teammanager.h"
```

7.24 C:/Users/Krzysztof/CLionProjects/To-Do-List/teammanager.h

File Reference

```

#include "team.h"
#include <string>
#include <QMessageBox>

```

```
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>
```

Namespaces

- namespace [TeamManager](#)

Functions

- bool [TeamManager::createTeam](#) (const [Team](#) &team)
Creates [Team](#) in a database.
- [Team](#) [TeamManager::getTeam](#) (const std::string &name)
Gets [Team](#) from a database identifying it by name and returns as [Team](#)'s object.
- [Team](#) [TeamManager::getTeam](#) (uint32_t id)
Gets [Team](#) from a database identifying it by ID and returns as [Team](#)'s object.
- bool [TeamManager::updateTeam](#) (const [Team](#) &team)
Updates [Team](#) in a database.
- bool [TeamManager::deleteTeam](#) (const [Team](#) &team)
Deletes [Team](#) from a database.
- bool [TeamManager::deleteTeam](#) (uint32_t id)
Deletes [Team](#) using his ID from a database.
- std::vector< [Team](#) > [TeamManager::getAllTeams](#) ()
Returns all Teams from a database.
- std::vector< [Team](#) > [TeamManager::getTeamsForUser](#) (uint32_t userId)
Returns all Teams that the [User](#) belongs to from a database.
- [Team](#) [TeamManager::getTeamForUser](#) (uint32_t userId)
Returns first [User](#)'s [Team](#) from a database.

7.25 teammanager.h

[Go to the documentation of this file.](#)

```
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     bool deleteTeam(uint32_t id);
00024
00025     std::vector<Team> getAllTeams();
00026
00027 }
```

```

00074     std::vector<Team> getTeamsForUser(uint32_t userId);
00075
00081     Team getTeamForUser(uint32_t userId);
00082
00083
00084
00085 }
00086
00087 #endif // TEAMMANAGER_H

```

7.26 C:/Users/Krzysztof/CLionProjects/To-Do-List/user.cpp File Reference

```
#include "user.h"
```

7.27 C:/Users/Krzysztof/CLionProjects/To-Do-List/user.h File Reference

```

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

```

Classes

- class [User](#)

7.28 user.h

[Go to the documentation of this file.](#)

```

00001 #ifndef USER_H
00002 #define USER_H
00003
00004 #include <cstdint>
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

```

7.29 C:/Users/Krzysztof/CLionProjects/To-Do-List/usermanager.cpp File Reference

```

#include "usermanager.h"
#include <vector>

```

7.30 C:/Users/Krzysztof/CLionProjects/To-Do-List/usermanager.h File Reference

```

#include <QMessageBox>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>
#include "user.h"

```

Namespaces

- namespace [UserManager](#)

Functions

- bool [UserManager::createUser](#) (const [User](#) &user)
Creates [User](#) in a database.
- [User](#) [UserManager::getUser](#) (const std::string &username)
Gets [User](#) from a database identifying him by username and returns as [User](#)'s object.
- [User](#) [UserManager::getUser](#) (uint32_t id)
Gets [User](#) from a database identifying him by username and returns as [User](#)'s object.
- bool [UserManager::updateUser](#) (const [User](#) &user)
Updates [User](#) in a database.
- bool [UserManager::deleteUser](#) (const [User](#) &user)
Deletes [User](#) from a database.
- bool [UserManager::deleteUser](#) (uint32_t id)
Deletes [User](#) using his ID from a database.
- std::vector< [User](#) > [UserManager::getAllUsers](#) ()
Returns all Users from a database.

7.31 usermanager.h

[Go to the documentation of this file.](#)

```
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 {
00017     bool createUser(const User& user);
00018
00025     User getUser(const std::string& username);
00026
00033     User getUser(uint32_t id);
00034
00041     bool updateUser(const User& user);
00042
00049     bool deleteUser(const User& user);
00050
00057     bool deleteUser(uint32_t id);
00058
00064     std::vector<User> getAllUsers();
00065
00066
00067 }
00068
00069 #endif // USERMANAGER_H
```

