Клиент-серверное приложение

“Корпоративный планировщик задач сотрудников”.

Назначение проекта.

Когда штат некой организации вырастает до нескольких сотрудников, возникает необходимость централизованного контроля за исполнением поставленных им задач. Главная цель данного проекта – реализовать возможность в такой компании создавать и назначать задачи, а также контролировать выполнение поставленных задач сотрудниками. Данные о сотрудниках и задачах должны храниться в единой базе данных, управляемой сервером, доступ к которому есть из локальной сети организации. Доступ к данным в базе может быть получен только через сервер и должен быть разграничен в зависимости от уровня доступа сотрудника.

Реализация.

Проект реализован в виде комплекта из трёх приложений, каждое из которых выполняет свой комплекс задач. Два приложения предназначены для запуска из консоли и работе на компьютере, выполняющем роль сервера. Последнее приложение имеет графический интерфейс пользователя и предназначено для установки на компьютеры сотрудников.

Состав проекта:

1. “ db\_creator” – Консольное приложение для создания базы данных и занесения в неё начальных значений.
2. “task\_server“ – Консольное приложение для работы с базой данных, приёма и обработки запросов от клиентов, подключенных по сети.
3. “TaskClient” – Приложение с графическим интерфейсом для работы на компьютерах сотрудников.

Для успешной компиляции всех программ данного проекта, потребуется наличие на компьютере следующих библиотек:

1. boost
2. GoogleTest
3. Qt версии 5.15
4. sqlite
5. nlohmann/json

Библиотеки sqlite и nlohmann/json скачиваются и распаковываются автоматически при сборке проекта, если использовать cmake и файл CMakeLists.txt, входящий в состав проекта. Внутри данного файла указаны следующие пути для скачивания архивов:

1. <https://www.sqlite.org/snapshot/sqlite-snapshot-202401231504.tar.gz>
2. <https://github.com/nlohmann/json/releases/download/v3.11.2/json.tar.xz>

Перед компиляцией проекта желательно убедиться в доступности архивов по указанным ссылкам и в случае недоступности, заменить их в файле CMakeLists.txt на актуальные.

**Компиляция проекта:**

Исходный код обоих консольных приложений, работающих на сервере, находится в папке “otus-cpp-prof-hw-14-server”. Для компиляции этих приложений выполните следующие действия:

1. Создайте папку “build” внутри папки “otus-cpp-prof-hw-14-server”.
2. Откройте консоль.
3. Перейдите в созданную папку build.
4. Если сборка происходит под ОС Windows, проверьте куда установлены библиотеки boost и GoogleTest.
5. Далее необходимо выполнить команду, в которой подразумевается, что библиотека boost версии 1\_82 установлена в папку “D:\Boost”, а библиотека GoogleTest установлена в папку “D:\GoogleTest”.

Выполните команду:

cmake .. -DBOOST\_INCLUDEDIR=D:\Boost\Shared\include\boost-1\_82 -DBOOST\_LIBRARYDIR=D:\Boost\Shared\lib -DBOOST\_ROOT=D:\Boost\Shared\include\boost-1\_82\boost -DGTEST\_ROOT=D:\GoogleTest\Shared

Далее выполните:

cmake --build . --config Release

После успешной компиляции появятся два исполнимых файла:

1. db\_creator.exe
2. task\_server.exe

В папке “otus-cpp-prof-hw-14-client” находится исходный код клиентской части проекта.

Самый простой способ скомпилировать программу, это запустить среду разработки qt-creator, открыть там файл проекта “TaskClient/TaskClient.pro”, слева снизу выбрать тип сборки “Release” и выполнить следующие команды из меню Build:

1. Build -> Run qmake
2. Build -> Build Project "TaskClient"

Если сборка происходит под ОС Windows, появится исполнимый файл “TaskClient.exe”, расположенный в папке  
“build-TaskClient-Desktop\_Qt\_5\_15\_2\_MSVC2019\_64bit-Release/release”.

Если предпочтительна сборка из консоли, то под ОС Linux нужно выполнить следующее:

1. Рядом с папкой TaskClient создать папку build
2. Открыть консоль и перейти в папку build
3. Выполните команду: qmake-qt5 ../TaskClient/TaskClient.pro
4. Убедитесь, что появился файл Makefile
5. Выполните команду: make
6. Убедитесь, что появился исполнимый файл TaskClient

**Начало работы с проектом:**

Чтобы серверная часть могла начать работать с данными, необходимо создать начальную базу данных, описанную в файле “tables.xlsx”.

Для этого откройте консоль и выполните:  
db\_creator.exe tasks\_db

В результате выполнения программы, в той же папке появится файл с базой данных “tasks\_db.sqlite”.

Программа “db\_creator.exe” больше нам не потребуется. Если захотите создать ещё одну базу данных, запустите снова эту программу, указав другое имя файла.

Поскольку база данных подготовлена, настало время запустить серверную часть проекта.

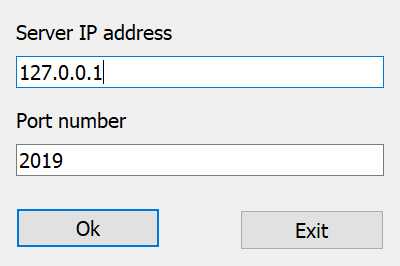
В той же консоли выполните:  
task\_server.exe 2019 tasks\_db.sqlite

В консоли будет видно, что сервер запущен и находится в ожидании нового подключения.

Число 2019 означает номер порта, на который должны приходить запросы от клиентов, чтобы сервер смог принять их и обработать.

После успешной компиляции клиентской части, появится исполнимый файл “TaskClient.exe”.

Это приложение имеет графический интерфейс и запускать его из консоли не обязательно. После запуска на экране появится окно, показанное ниже.



Если сервер и клиент запущены на одном компьютере, IP адрес сервера можно оставить “127.0.0.1”. Номер порта должен соответствовать тому, что использовали при запуске сервера.

Если подключение к серверу пройдёт успешно, на экране появится окно для ввода логина и пароля. В файле “tables.xlsx” содержатся несколько пользователей, которые создаются приложением “db\_creator” во время создания базы. Используйте столбец “user\_unique\_login” из таблицы “Users” чтобы выбрать логин и столбец “password” из таблицы “Security” чтобы узнать соответствующий пароль. Используйте эту пару в окне для ввода логина и пароля чтобы приступить к работе с базой.

**Подробнее о клиентской части**:

Интерфейс пользователя создается с применением фреймворка Qt 5.15.

Приложение поддерживает три типа пользователей:

1. Администратор.
2. Оператор базы данных (db\_operator).
3. Обычный сотрудник (user, пользователь).

Возможности администратора:

1. Запрос от сервера списка сотрудников.
2. Создание нового сотрудника.
3. Изменение типа пользователя.
4. Удаление пользователя.
5. Изменение пароля пользователя.
6. Запрос от сервера списка задач.
7. Запрос от сервера задач, назначенных только на себя.
8. Выключение сервера.

Возможности оператора базы данных:

1. Запрос от сервера списка сотрудников.
2. Запрос от сервера списка задач.
3. Запрос от сервера задач, назначенных только на себя.
4. Запрос от сервера задач, назначенных на выбранного пользователя.
5. Создание новой задачи с назначением на неё выбранного сотрудника.
6. Удаление задачи.
7. Изменение статуса выбранной задачи.
8. Изменение времени выполнения (дедлайна) выбранной задачи.
9. Назначение задачи на выбранного сотрудника.

Возможности сотрудника:

1. Запрос от сервера списка задач.
2. Запрос от сервера задач, назначенных только на себя.
3. Создание новой задачи с автоматическим назначением на себя.
4. Возможность взять на себя чужую задачу.
5. Изменение статуса своей задачи.
6. Изменение времени выполнения (дедлайна) своей задачи.

Ограничения:

1. Администратор, по сути, работает с пользователями, а задачи только смотрит.
2. Оператор базы данных может влиять на все задачи, а пользователей только просматривает.
3. Пользователь может просматривать все задачи в базе, но влиять может только на назначенные на себя. Запрашивать список сотрудников он не может.
4. Если потребуется добавить ещё один тип сотрудника, недостаточно будет изменить содержание базы данных. Потребуется добавить ещё один, соответствующий уровню доступа интерфейс в клиентскую часть. Т.е. нужно вносить изменения в исходный код и скомпилировать программу заново.

Общий алгоритм работы клиентской части:

1. После запуска приложения появляется окно для ввода ip адреса и порта для связи с сервером.
2. Если связь с сервером установлена, приложение запрашивает у сервера публичные данные – типы пользователей с их текстовым описанием и виды статусов задач, также с их текстовым описанием.
3. Запускается цикл, позволяющий вводить логин и пароль и далее приступать к работе с базой.
4. Если указанная пара логина и пароля проходит проверку на сервере, сервер отправляет обратно id данного пользователя и его тип, означающий уровень доступа к данным.
5. Далее на экране появится окно с интерфейсом, соответствующим полученному типу пользователя, после чего можно приступать к работе с базой.
6. Когда пользователь решит завершить работу, он нажимает кнопку Exit, после чего текущее окно будет закрыто и снова появится окно для ввода логина и пароля. Это сделано для возможности переключения между разными типами пользователей без завершения работы программы.
7. Если нажать Exit в окне для логина, работа программы будет завершена.

**Подробнее о серверной части**:

Графической части нет.

После включения работает в режиме асинхронной обработки запросов от клиента, создавая для каждого клиента свою сессию. Количество сессий ограничено для защиты от слишком большого количества запросов. Асинхронная работа по сети реализована при помощи библиотеки boost.

Управляет базой данных, в которой хранятся данные о сотрудниках и назначенных задачах. База данных является реляционной и создана на основе sqlite. Серверная часть может изменять содержимое базы данных, но не может создавать базу.

База данных содержит всего пять таблиц, две из которых имеют начальные неизменяемые значения, а остальные три изменяются по мере отработки запросов от клиентской части. Доступ к таблицам защищен / синхронизирован мьютексом.

Таблицы в базе разделены на таблицы с константными данными и на таблицы с изменяемыми данными.

Неизменяемые таблицы:

1. UserTypes – содержит возможные типы пользователей.
2. TaskStatuses – содержит возможные статусы задач.

Изменяемые таблицы:

1. Users – таблица с пользователями.
2. Security – реализует связь между пользователем, его типом и паролем.
3. Tasks – содержит задачи, статусы задач и связь с пользователями.

**Подробнее о таблицах**:

Номер столбца, наименование, тип данных, описание.

*UserTypes*

1, user\_type\_number, TINYINT, уникальное число-ключ для типа пользователя.  
2, user\_type\_description, CHAR(16), текстовое название типа пользователя.

*TaskStatuses*

1, status\_type\_number, TINYINT, уникальное число - ключ для статуса задачи.  
2, status\_type\_description, CHAR(16), текстовое название статуса задачи.

*Users*

1, user\_unique\_id, SMALLINT, уникальное число – идентификатор пользователя.  
2, user\_unique\_login, CHAR(16), уникальный логин пользователя.  
3, first\_name, CHAR(24), фамилия.  
4, second\_name, CHAR(24), имя.  
5, patronymic, CHAR(24), отчество.

*Security*

1, user\_unique\_id, SMALLINT, уникальное число – идентификатор пользователя.  
2, user\_type\_number, TINYINT, число – тип пользователя.  
3, password, CHAR(32), пароль.

*Tasks*

1, task\_unique\_id, INTEGER, уникальное число – идентификатор задачи.  
2, user\_unique\_id, SMALLINT, число – идентификатор пользователя.  
3, status\_type\_number, TINYINT, число – статус задачи.  
4, deadline, DATE, крайний срок выполнения.  
5, task\_name, CHAR(64), название задачи.  
6, task\_description, VARCHAR(1000), описание, что нужно сделать.

**Подробнее о консольном приложении для создания базы.**

Графической части нет.

При запуске требует указания имени файла для новой базы данных. Если в переданном имени отсутствует расширение “.sqlite”, добавляет автоматически. Если файл с таким именем уже существует, программа завершается с ошибкой, т.к. можно создавать новую базу, но нельзя перезаписывать старую. Программа создает новый файл с переданным именем и в нём новую базу данных. Далее программа использует заранее подготовленные sql скрипты для следующих действий:

1. Создаются пять таблиц, описанные выше.
2. Таблицы UserTypes и TaskStatuses заполняются константными данными.
3. Таблицы Users, Security и Tasks заполняются начальными данными.
4. Производится вывод в консоль содержимого всех таблиц.

На этом работа консольного приложения завершается.

**Виды сетевых запросов от клиентской части к серверу.**

Запросы от клиента к серверу разделены на 9 типов:

1. Тестовый запрос для проверки связи с сервером.
2. Набор запросов для получения общих (публичных) данных, не требующих логина на сервере.
3. Запрос совершения логина на сервере – создание новой сессии пользователя.
4. Запрос на закрытие сессии пользователя.
5. Запрос на выключение сервера.
6. Набор запросов для получения данных, доступных только после успешного логина на сервере.
7. Набор запросов для добавления данных в базу.
8. Набор запросов для изменения данных в базе.
9. Набор запросов для удаления данных из базы.

Следующие запросы входят в набор для получения общих (публичных) данных:

1. Запрос на получение списка возможных типов пользователей, который состоит из числовых значений и соответствующих им текстовых описаний.
2. Запрос на получение списка возможных статусов задач, который состоит из числовых значений и соответствующих им текстовых описаний.

Следующие запросы входят в набор для получения данных, доступных только после успешного логина на сервере:

1. Запрос на получение списка уникальных идентификаторов всех пользователей.
2. Запрос на получение списка уникальных идентификаторов всех задач.
3. Запрос на получение пары из логина и типа пользователя по идентификатору пользователя.
4. Запрос на получение ФИО пользователя по идентификатору пользователя.
5. Запрос на получение данных одной задачи по идентификатору задачи.

Следующие запросы входят в набор для добавления данных в базу:

1. Запрос на создание нового пользователя.
2. Запрос на создание новой задачи.

Следующие запросы входят в набор для изменения данных в базе:

1. Запрос на изменение типа пользователя.
2. Запрос на изменение статуса задачи.
3. Запрос на изменение пароля пользователя.
4. Запрос на изменение deadline задачи.
5. Запроса на назначение другого пользователя исполнителем задачи.

Следующие запросы входят в набор для удаления данных из базы:

1. Запрос на удаление пользователя.
2. Запрос на удаление задачи.

**Визуальный контроль выполнения сетевых запросов на серверной стороне.**

Т.к. серверная часть лишена графического интерфейса, для неё реализован вывод в консоль следующих событий:

1. Начало прослушивания порта в ожидании новых подключений.
2. Подключение нового пользователя.
3. Завершение сессии пользователя.
4. Получение сетевого запроса.
5. Реакция на сетевой запрос.
6. Выключение сервера с прекращением ожидания новых подключений.