Клиент-серверное приложение

“Корпоративный планировщик задач сотрудников”.

Назначение:

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.

**Быстрый запуск (Quick start):**

После успешной компиляции серверной части появятся два исполнимых файла:

1. db\_creator.exe

2. task\_server.exe

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.

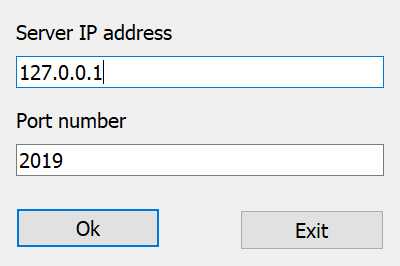
В той же консоли выполните:  
task\_server.exe 2019 tasks\_db.sqlite

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.

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 пересобирать программу.

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.

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.**

1. Проверка пары логин/пароль. Ответ сервера либо содержит ответ “false”, означающий, что с такой комбинацией авторизация не возможна, либо содержит ответ “true” и поле с номером типа пользователя user\_type\_number.
2. Запрос описания типа пользователя. Отправляем user\_type\_number и получаем строку с описанием типа: administrator, db\_operator, user.
3. Получение списка всех пользователей. Запрос массива со всеми user\_unique\_id пользователей.
4. Получение данных пользователя по его id. Отправить user\_unique\_id пользователя и получить значения полей user\_unique\_login, first\_name, second\_name и patronymic.
5. Изменение пароля пользователя. Отправляем user\_unique\_id и новый пароль. В ответе может быть либо true, либо false, если по какой-то причине не удалось сохранить в базе новый пароль.
6. Изменение типа пользователя. Отправляем user\_unique\_id и новый user\_type\_number. В ответе может быть либо true, либо false, если по какой-то причине не удалось записать в базу новый тип пользователя.
7. Создание нового пользователя. Отправляем логин, ФИО, тип пользователя и пароль. Новый user\_unique\_id формируется на сервере автоматически. В ответе от сервера получаем false, если произошла ошибка, например пользователь с таким логином уже есть, либо получаем true и строку с новым user\_unique\_id.
8. Получение списка всех задач. Запрос массива со всеми task\_unique\_id задач.
9. Запрос данных задачи. Отправляем task\_unique\_id. Получаем значения полей user\_unique\_id, status\_type\_number, deadline, task\_name, task\_description.
10. Получение описания статуса задачи. Отправляем status\_type\_number, а получаем строку со значением поля status\_type\_description.
11. Получение списка задач, назначенных на конкретного пользователя. Отправляем id пользователя user\_unique\_id, а получаем массив с task\_unique\_id, привязанных именно к нему.
12. Изменение статуса задачи. Отправляем task\_unique\_id и новый status\_type\_number. Получаем либо true, либо false в случае ошибки.
13. Изменение крайнего срока выполнения задачи. Отправляем task\_unique\_id и новый deadline. Получаем либо true, либо false в случае ошибки.
14. Назначение нового пользователя на выполнение задачи. Отправляем task\_unique\_id и новый user\_unique\_id. Получаем либо true, либо false в случае ошибки.
15. Создание новой задачи. Отправляем user\_unique\_id, deadline, task\_name и task\_description. Сервер сформирует новый task\_unique\_id и назначит задаче статус appointed. В ответе от сервера получаем false, если произошла ошибка при добавлении задачи в базу, либо получаем true и строку с новым task\_unique\_id.
16. Получение списка всех задач, имеющих определённый статус. Отправляем status\_type\_number, а получаем массив с соответствующими task\_unique\_id.
17. Получение списка всех задач, которые необходимо выполнить до определённой даты. Отправляем deadline, а получаем массив с соответствующими task\_unique\_id.
18. Получение списка всех задач, крайний срок исполнения для которых вышел. Отправляем deadline, а получаем массив с соответствующими task\_unique\_id.
19. Проверка, имеет ли пользователь доступ к задаче. Отправляем user\_unique\_id и task\_unique\_id. Получаем либо true, либо false, если пользователь с задачей не связан. Может понадобиться для сохранения результата редактирования задачи.