Практическое тестовое задание

Цель:

Разработать две программы на Qt версии совместимой с Astra Linux 1.7

1) Агент — выполняет задачи локально, сообщает о статусах выполнения контроллеру.
2) Контроллер — управляет одним агентом, отображает задачи, позволяет создавать и отменять задачи.

Программа агент

Функциональные требования:

- хранит задачи в памяти
- создает, запускает и отменяет задачи, обновляет их статус и отправляет контроллеру;
- поддерживает 1 тип задач:
  - вычислительная – например, посчитать сумму от 1 до N.
- выполняет задачи в отдельном потоке (QThread);
- пользовательский интерфейс:
  - кнопка подключения к контроллеру.

Каждая задача имеет:

- уникальный ID (автогенерируемый)
- статус (новая, в работе, завершена, ошибка, отменена),
- прогресс (0–100%),
- результат (число - сумма).

Программа контроллер

Функциональные требования:

- хранит задачи в локальной базе данных (SQLite);
- подключается к одному агенту;
- отображает список задач в таблице (ID, статус, прогресс, результат);
- позволять создавать задачу (параметры задаются пользователем);
- позволять отменить задачу;
- обновляет таблицу при получении статусов.
- Пользовательский интерфейс:
  - таблица (QTableWidget) с задачами;
  - кнопка «Создать задачу» с полем ввода N;
  - кнопка «Отменить задачу».

Организация обмена

Между агентом и контроллером должны передаваться бинарные сообщения в формате protobuf.

Типы сообщений:

- Handshake содержит ID агента, фиксированная версия протокола «1.0»);
- TaskRequest содержит ID, тип ("sum"), параметр (N);
- TaskResponse содержит статус успех/ошибка;

- TaskStatus содержит ID, статус, прогресс, результат;
- CancelTask содержит ID.

На что обратить внимание при реализации:

- использовать Qt для сетевого взаимодействия (QTcpSocket) и потоков (QThread);
- задачи обрабатываются асинхронно, чтобы не блокировать пользовательский интерфейс;
- архитектура должна позволять добавить новый тип задачи;
- protobuf используется для сериализации сообщений, .proto-файл компилируется в C++ код.

По окончанию выполнения необходимо предоставить следующие материалы:

- исходный код обоих приложений;
- скрипт сборки;
- схему базы данных;
- файлы .proto;
- краткая инструкцию по запуску;
- краткое текстовое описание архитектуры (как устроено взаимодействие и как добавить новый тип задачи).

Возможный протокол (можно реализовать собственный)

```
syntax = "proto3";
package task;

message Handshake {
string agent_id = 1;
string protocol_version = 2;
}

message TaskRequest {
string task_id = 1;
string task_type = 2;
int32 param_n = 3;
}

message TaskResponse {
bool success = 1;
string error_message = 2;
}

message TaskStatus {
string task_id = 1;
string status = 2; // "new", "running", "completed" int32 progress = 3; // 0-100 int64
result = 4;        // Sum result
}

message CancelTask {
string task_id = 1;
}
```