**Общие сведения о протоколе**

В приложении используется **собственный текстовый протокол** обмена данными между клиентом и сервером, реализованный поверх сетевого соединения **TCP**.

Протокол обеспечивает обмен текстовыми командами и структурированными данными в формате **JSON** между клиентской и серверной частями приложения.

**Технические детали реализации**

* Серверная часть (прием и обработка данных):
  + Сервер прослушивает TCP-соединения на порту **33333** и принимает текстовые команды от клиента. Пример кода (сервер принимает подключение и читает данные):

// mytcpserver.cpp

mTcpServer = new QTcpServer(this);

connect(mTcpServer, &QTcpServer::newConnection, this, &MyTcpServer::slotNewConnection);

mTcpServer->listen(QHostAddress::Any, 33333);

// метод обработки подключения клиента

void MyTcpServer::slotNewConnection()

{

mTcpSocket = mTcpServer->nextPendingConnection();

connect(mTcpSocket, &QTcpSocket::readyRead, this, &MyTcpServer::slotServerRead);

}

// метод чтения и обработки команд от клиента

void MyTcpServer::slotServerRead()

{

QByteArray data = mTcpSocket->readAll();

QString input = QString(data).trimmed();

// обработка команд от клиента

if (input.startsWith("LOGIN")) {

// Авторизация

} else if (input.startsWith("REGISTER")) {

// Регистрация

} else if (input.startsWith("GET\_EXAMS")) {

// Получение экзаменов

} else if (input.startsWith("GET\_QUESTIONS")) {

// Получение вопросов

} else if (input.startsWith("SAVE\_RESULTS")) {

// Сохранение результатов

} else if (input.startsWith("GET\_STATISTICS")) {

// Получение статистики

} else {

mTcpSocket->write("ERROR\r\n");

}

}

**Клиентская часть (отправка команд на сервер):**

* Клиент отправляет текстовые команды серверу через TCP-соединение. Пример кода (подключение к серверу и отправка команд):

// Client.cpp

m\_socket = new QTcpSocket(this);

m\_socket->connectToHost("127.0.0.1", 33333);

// команда авторизации

void Client::loginUser(const QString &username, const QString &password)

{

QString request = QString("LOGIN %1 %2\r\n").arg(username).arg(password);

m\_socket->write(request.toUtf8());

}

// команда получения списка экзаменов

void Client::requestExamList()

{

QString request = "GET\_EXAMS\r\n";

m\_socket->write(request.toUtf8());

}

// команда сохранения результатов экзамена (JSON)

void Client::saveExamResults(const QString &jsonData)

{

QString request = QString("SAVE\_RESULTS %1\r\n").arg(jsonData);

m\_socket->write(request.toUtf8());

}

**Формат команд и данных протокола**

Примеры типичных запросов и ответов:

| **Действие** | **Запрос клиента** | **Ответ сервера** |
| --- | --- | --- |
| Авторизация | LOGIN user1 pass123 | OK или ERROR |
| Регистрация | REGISTER user2 pass456 | OK или ERROR |
| Запрос списка экзаменов | GET\_EXAMS | JSON-список экзаменов |
| Запрос вопросов экзамена №5 | GET\_QUESTIONS 5 | JSON-список вопросов |
| Сохранение результатов | SAVE\_RESULTS {"examId":5,"score":4} | OK или JSON-данные |
| Запрос статистики | GET\_STATISTICS | JSON-статистика |

**Краткий вывод и итоговая схема**

Протокол взаимодействия клиента и сервера в приложении **«**ExamSem**»**:

* Реализован на основе TCP-соединения.
* Использует простые текстовые команды.
* Использует формат JSON для передачи структурированных данных.
* Предназначен для операций: регистрации, авторизации, проведения экзаменов, сохранения результатов и получения статистики.

**Общая схема взаимодействия:**

Клиент (QTcpSocket)

|

| текстовые команды, JSON данные

|

Сервер (QTcpServer, QTcpSocket)

|

| SQL-запросы, работа с PostgreSQL

|

База данных PostgreSQL