ТЗ на диалоговую систему:
0.Переработать класс **Answer**: убрать из него поля name и connectionKey. Создать класс **AutherisationAnswer**, наследующийся от Answer`а и добавить в него поля name и connectionKey.
1. Клиент умеет отправлять запрос о поиске собеседника: Для этого он отправляет наследника класса Query **CollocutorQuery**, к которому добавляется одно новое поле: collocutor (с англ. собеседник, сам только узнал) в котором указан ник человека, с которым клиент пытается начать общение. Если ник присутствует в базе данных, то БД должна вернуть **CollocutorAnswer** с полями:
isOk=true, operation=collocation, String collocutor = имя собеседника
иначе isOk=false, operation=collocation, String collocutor = имя собеседника.

2.Клиент умеет отправлять запрос об отправке сообщения: Для этого он отправляет наследника CollocutorQuery **MessageQuery**,к которому добавляется одно новое поле: String message — сообщение,отправленное клиентом Если пользователь, которому отправлено сообщение, существует, то сервер записывает сообщение в свою базу данных, а пользователю отправляется экземпляр класса **MessageAnswer**, наследующийся от Answer и содержащий в себе новые поля:

string sender,
string receiver,
string message,
Calendar date.

Этот Answer отправляется в случае успешной записи сообщения в БД (а не в случае прочтения сообщения сервером — это важно).

3. Когда пользователь заходит в диалоги, он отправляет наследника класса Query - RequestMessagesQuery с полем operation=startGettingMessages и новым полем ArrayList<DateOfLastMessage> dates — список дат последних сообщений, где **DateOfLastMessage** — класс с двумя полями:
String nickname
Date dateOfLastMessage
В ответ ожидается получение всех новых сообщений.Определение новых сообщений осуществляется следующим образом. Для каждого диалога на сервере хранится дата последнего отправленного сообщения. Сервер отправляет все новые сообщения с помощью экземпляра класса **MessagesAnswer**, наследник Answer, который, помимо прочего из Answer, хранит в себе структуру данных ключ-значение, где ключом будет выступать String — имя пользователя, с кем ведётся диалог. Значением будет ArrayList<**Message**>;
Поля класса Message:
String sender;
String receiver;
String message;
Calendar date;

Позже, пока пользователь находится в диалогах, сервер отправляет экземпляры класса MessageAnswer. Сервер отправляет MessageAnswer даже в случае, когда отправителем выступает клиент — это нужно для того,чтобы в диалогах присутствовали лишь сообщения, которые хранятся на сервере.

4. Структуру данных ключ-значение заменить просто на ArrayList<**Message**> messages;