**Описание решения**.

Документ содержит описание реализации решения. Общее описание содержится в документе “Общее описание решения”.

**Архитектура решения**.

Состав исходного кода и схема зависимостей модулей показаны на рис. 1. Белым цветом показаны заголовочные файлы. Стрелкт сооветствуют директивам include.

Interface

Chat\_Client

chat\_interface.cpp

buffer.cpp

definitions.h

chat\_interface.h

Access

buffer.h

message\_codes.h

chat\_client.cpp

network.cpp

network.h

chat\_client.h

Transport

client\_net.cpp

Network

Chat\_Server

Transport

server\_net.cpp

network.h

chat\_server.h

network.cpp

Access

message\_codes.h

buffer.h

chat\_server.cpp

definitions.h

buffer.cpp

chat\_db.h

sha1.cpp

DBCS

chat\_db.cpp

sha1.h

Рис. 1. Схема зависимостей модулей.

Модуль chat\_interface.cpp реализует интерфейс приложения chat\_client. Модули chat\_client.cpp и chat\_server.cpp реализуют протокол обмена между клиентом и сервером.

Модули client\_net.cpp и server\_net.cpp реализуют цикл обмена сообщениями в процессе диалога. Организация диалога сама по себе не зависит от протокола обмена, реализуемого chat\_server и chat\_client.

Модуль chat\_db реализует базу данных, реализация chat\_server никак не зависит от реализации chat\_db.

**Реализация интерфейса**.

В каждый момент времени некоторый контекст являтся текущим. Каждый контекст представлен статическим экземпляром соответствующего класса. В каждый момент времени current\_context указывает на текущий контекст. Основные методы класса Context показаны в таблице 1.

Классы контекстов показаны на рис. 2. Белым цветом показаны абстрактные классы. Сплошная линия демонстрирует наследование, пунктирная - реализацию.

Auth\_Context

Context

Group\_Context

Contact\_Context

User\_Context

Transtional\_Context

Contact\_Control

Group\_Control

Control\_Context

Contact\_Choice

Group\_Choice

Choice\_Context

Рис. 2. Классы контекстов.

Класс Context выражает общие свойства контекста.

Класс User\_Context - прототип контекста пользователя.

Transitional\_Context - переходный контекст, реализует операцию перехода между контекстами, при отмене операции происходит возврат в исходный контекст.

Control\_Context - контекст, из которого возможен переход в контекст выбора.

Choice\_Context - контекст выбора из списка.

**Таблица 1**. Основные методы класса Context.

|  |  |
| --- | --- |
| Метод | Функциональность |
| void become\_Current() | Сделаться текущим контекстом |
| void show\_Menu() | Вывести меню контекста |
| void operation(int code) | Обработать введенный символ |

**Взаимодействие клиента и сервера**.

Логика работы клиента показана на рис. 3 (пунктиром показана передача сообщений через сеть). Интерфейс вызывает функции, описанные в chat\_interface.h и реализованные в chat\_client.cpp. Клиент формирует данные запроса на сервер и вызывает функцию send\_Request, которая после установления соединения с сервером отправляет запрос.

server

client\_net

chat\_client

chat\_interface

send\_Request

operation (data)

request

reply

client\_Processing

result

request

…

reply

client\_Processing

request (end)

success, finish

Завершение send\_Request, success

Завершение operation

Рис. 3. Логика работы клиента.

После получения ответа send\_Request вызывает функцию client\_Processing, которая обрабатывает ответ и формирует очередной запрос. Обмен продолжается до тех пор, пока client\_Processing не устанавливает флаг окончания. В этом случае send\_Request отправляет на сервер запрос окончания диалога и завершает работу. Функция client\_Processing может также вызывать функцию show\_Message, по окончании диалога может быть вызвана функция show\_Result, которая выводит информацию об ошибке. Функция client\_Processing также может устанавливать флаг success, если операция выполнена успешно.

chat\_logic

chat\_server

server\_net

client

request

server\_Processing

operation (data)

reply

…

…

server\_Processing

request (end)

finish

Рис. 4. Логика работы сервера.

На стороне сервера функция wait\_Request прослушивает порт, при установлении транспортного соединения начинает прием запросов и формирование ответов. Если запрос принят, вызывается функция server\_Processing, анализирующая запрос и формирующая ответ, при этом может вызываться функция chat\_db. После окончания диалога server\_Processing устанавливает флаг finish, и wait\_Request снова переходит к прослушиванию порта.

Модуль network.cpp реализует выполнение сетевых функций и делает реализацию основной части решения независимой от операционного окружения. Интерфейс модуля определяется заголовочным файлом network.h.

Время ожидания поступления очередного запроса от клиента ограничено величиной TIMEOUT (в мсек). По истечении таймера считается, что время ожидания пакета истекло, сервер закрывает транспортное соединение и вновь переходит к ппрослушиванию порта. Это позволяет сделать работу сервера независисимой от возможных сетевых сбоев или некоррекной работы клиента. Если запрос от клиента не поступает вовремя, сервер не “вешается”, а продолжает работу.

Время ожидания поступления очередного ответа от сервера также ограничено величиной TIMEOUT, по истечении таймера клиент фиксирует ошибку обмена. Это позволяет сделать работу клиента назависимой от сетевых сбоев и некорретной работы сервера.

**Описание протокола**.

Все сообщения представлены в текстовой форме. Сообщение содержит код и, возможно, параметры. Коды запросов, ответов и параметры представлены строками.

**Таблица 2**. Запросы клиента.

|  |  |  |  |
| --- | --- | --- | --- |
| Код | Запрос | Параметры | Возможные ответы |
|  | SIGN\_IN | username, pаssword | AUTHENTICATED, USER\_NOT\_FOUND, WRONG\_PASSWORD, DATABASE\_ERROR |
|  | SIGN\_UP | username, pаssword | AUTHENTICATED, USER\_NOT\_FOUND, EXISTING\_USER, DATABASE\_ERROR |
|  | CHANGE\_PASSWORD | username,  oldpassword, newpassword | AUTHENTICATED, USER\_NOT\_FOUND, WRONG\_PASSWORD, DATABASE\_ERROR |
|  | FIND\_USER | username, pаssword,  peername | OK, USER\_NOT\_FOUND, WRONG\_PASSWORD, DATABASE\_ERROR |
|  | CONTACT\_LIST | username, pаssword | LINE\_LIST, USER\_NOT\_FOUND, WRONG\_PASSWORD, DATABASE\_ERROR |
|  | CONTACT\_MESSAGES | username, pаssword,  peername | MESSAGE, END, USER\_NOT\_FOUND, WRONG\_PASSWORD, DATABASE\_ERROR |
|  | TO\_CONTACT | username, pаssword,  peername, text | OK, USER\_NOT\_FOUND, WRONG\_PASSWORD, DATABASE\_ERROR |
|  | GROUP\_LIST | username, pаssword | LINE\_LIST, USER\_NOT\_FOUND, WRONG\_PASSWORD, DATABASE\_ERROR |
|  | GROUP\_MEMBERS | username, pаssword  groupname | LINE\_LIST, USER\_NOT\_FOUND, WRONG\_PASSWORD, GROUP\_NOT\_FOUND, DATABASE\_ERROR |
|  | JOIN\_GROUP | username, pаssword  groupname | OK, USER\_NOT\_FOUND, WRONG\_PASSWORD, GROUP\_NOT\_FOUND, DATABASE\_ERROR |
|  | LEAVE\_GROUP | username, pаssword  groupname | OK, USER\_NOT\_FOUND, WRONG\_PASSWORD, GROUP\_NOT\_FOUND, DATABASE\_ERROR |
|  | NEW\_GROUP | username, pаssword  groupname | OK, USER\_NOT\_FOUND, WRONG\_PASSWORD, EXISTING\_GROUP, DATABASE\_ERROR |
|  | GROUP\_MESSAGES | username, pаssword  groupname | MESSAGE, END, USER\_NOT\_FOUND, WRONG\_PASSWORD, GROUP\_NOT\_FOUND, DATABASE\_ERROR |
|  | TO\_GROUP | username, pаssword  groupname, text | OK, USER\_NOT\_FOUND, WRONG\_PASSWORD, GROUP\_NOT\_FOUND, DATABASE\_ERROR |
|  | NEXT\_MESSAGE |  | MESSAGE, END, DATABASE\_ERROR |

Каждый запрос клиента (за исключением запроса NEXT\_MESSAGE) содержит имя пользователя и пароль. После успешной аутентификации выполняется операция пользователя, сервер отправляет клиенту ответ.

Каждый диалог за исключением запроса сообщений состоит из одного запроса и одного ответа. Запросы клиента показаны в таблице 2.

Ответ LINE\_LIST содержит последовательность строк. Ответ MESSAGE содержит следующие параметры, каждый из который также представлен текстовой строкой.

timestamp текстовое представление значения time\_t.

sender отправитель

recipient получатель

text текст сообщения

request

server

client

reply

Рис. 5. Простой диалог.

Каждый диалог, за исключением запроса сообщений состоит из одного запроса и одного ответа (см. рис.5).

server

client

request

Отправка запроса

Представление результата

Данные сообщения

MESSAGE

NEXT\_MESSAGE

MESSAGE

Представление результата

Данные сообщения

…

NEXT\_MESSAGE

END

Рис. 8. Выборка сообщений.

Запрос сообщений может включать серию последующих запросов и ответов (см. рис. 6), при этом в ответ на каждый запрос передаются данные одного сообщения.

**Схема данных.**

Графическое представление схемы данных приведено на рис. 9. Каждое сообщение отправлено определенным поьззователем либо контакту, либо группе.

id

person\_id

mutual\_id

person\_contact

contact\_id

person

id

id

timestamp

sender\_id

message

contact\_id

community\_id

text

name

pass

id

id1

id2

contact

id

person\_id

community\_id

person\_community

id

name

community

Таблица person\_community представляет связь между пользователями и группами. Таблица contact представляет пары пользователей. Таблица person\_contact делает удобным нахождение связи между пользоваьелем и контактом.