LearNet

Sbârcea Ștefan-Vladimir

Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi
stefan.sbarcea@info.uaic.ro

1 Introduction

In this report we will discuss in detail about LearNet application. This application is a client-server application where clients can interact with each other through chat or find useful information about computer networks. Every client needs an existing account to enter the application or they can create one if they have an invite code. Every chat, account and informational data is saved in an SQLite database. The information about computer networks that is presented in the application is taken from Alboaie Lenuta and Panu Andrei Computer Networks courses from Faculty of Computer Science [3].

The structure of this paper is as follows: Section 2 describes the technologies used in the creation of the application. Section 4 describes the architecture of the application. In Section 5 we present some implementation details including parts of the source code. Finally we draw some conclusions.

2 Used technologies

The application source code is written in C++ programming language.

To achieve the client-server connection we used the concurrent TCP model (multi-processing), which allows the possibility of connecting several clients to the server and TCP's reliable data transfer service ensures that the data stream that a process reads out of its TCP receive buffer is uncorrupted, without gaps, without duplication, and in sequence; that is, the byte stream is exactly the same byte stream that was sent by the end system on the other side of the connection [2].

The client GUI was made with Qt Creator, a cross-platform integrated development environment (IDE) built for the maximum developer experience [4].

To save the information we used SQLite database because it is lightweight when it comes to setup complexity and resource usage [1].

3 Application architecture

As we said prior the connection between server and client is done with TCP protocol and we create different processes to be able to serve clients concurrently. The server gets requests from the client it processes them and send back the data resulted. We created a database to save every client account information (username, password, rank, chats and chat messages) and the information about computer networks. In figure 1 you can see how the connection between server-client was made and in figure 2 are presented the inside workings of the application.

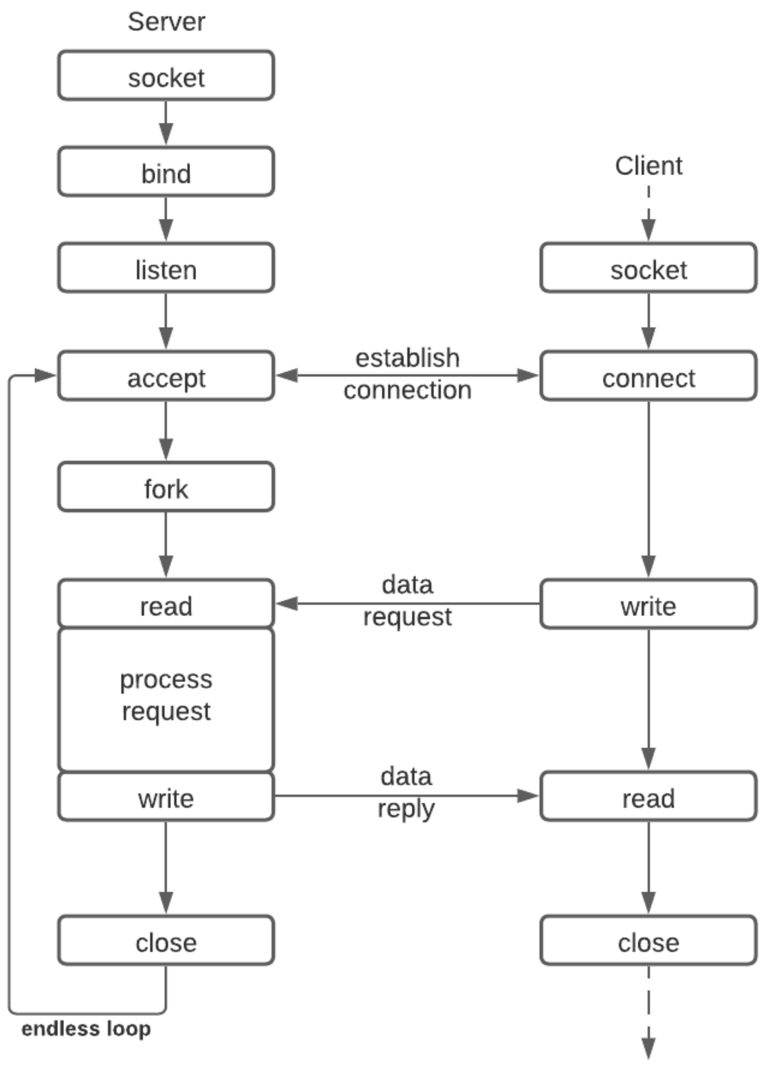


Fig. 1. Application diagram

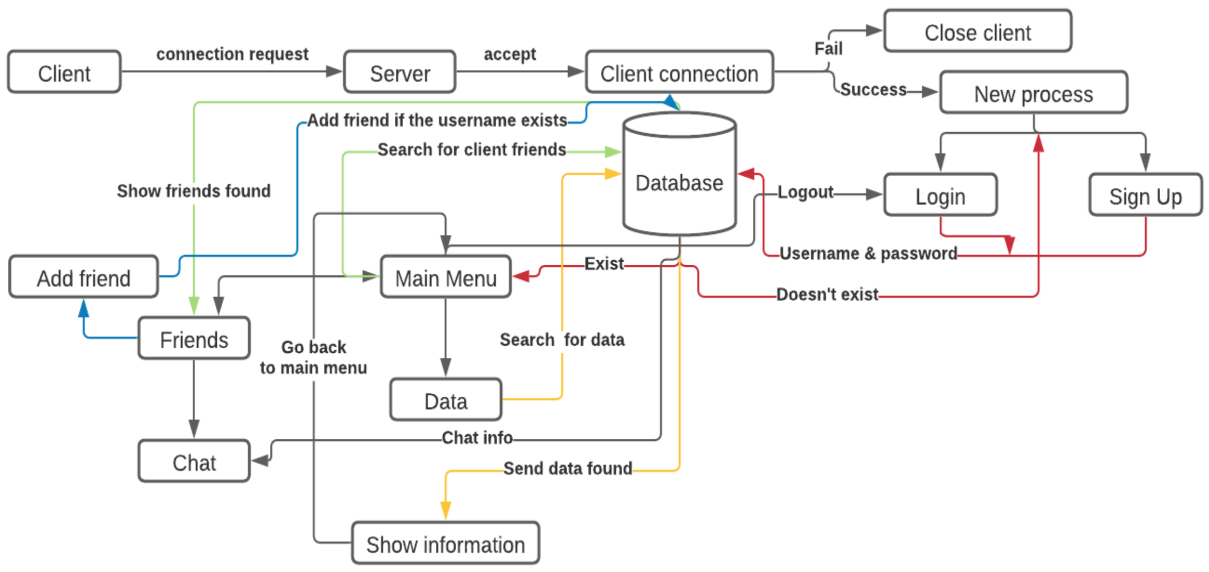


Fig. 2. Application diagram

4 Implementation details

Every byte stream, sent between clients and server, is prefixed by his length.

As you have seen in the application diagram (figure 2) the database stands in the center of the implementation, in figure 3 we show how the database was designed. The chats between individual users are saved as different tables for every two users, that's why only the table **courseschats** appears in the diagram, which is used to save messages from every global chat.

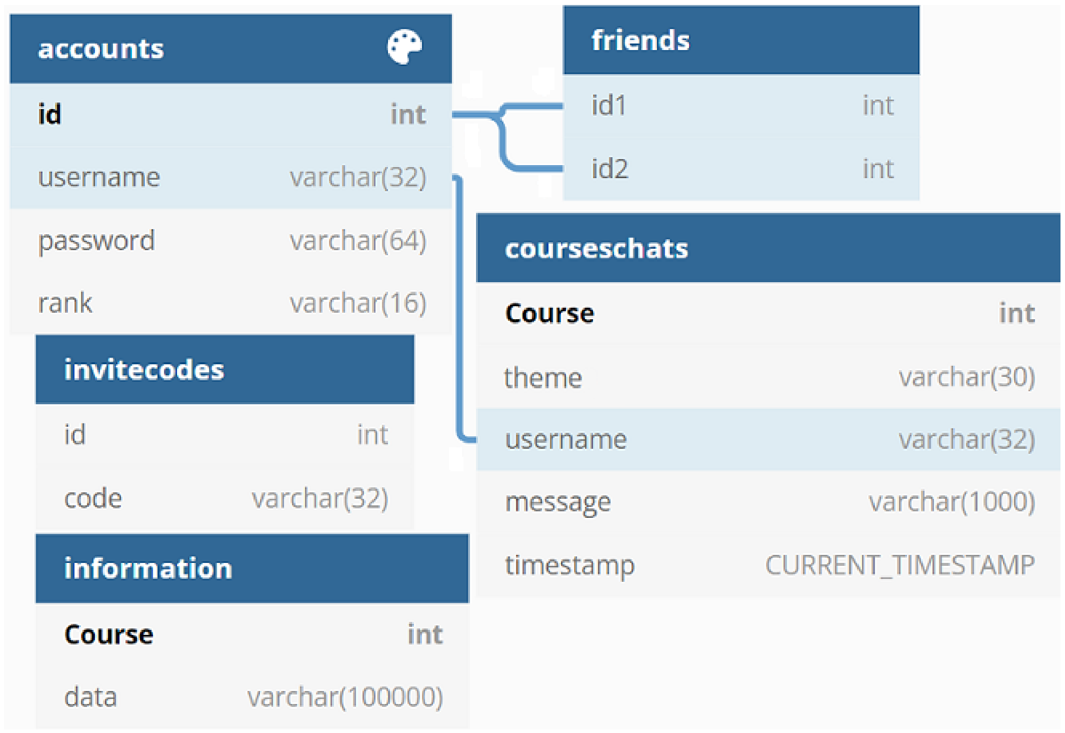
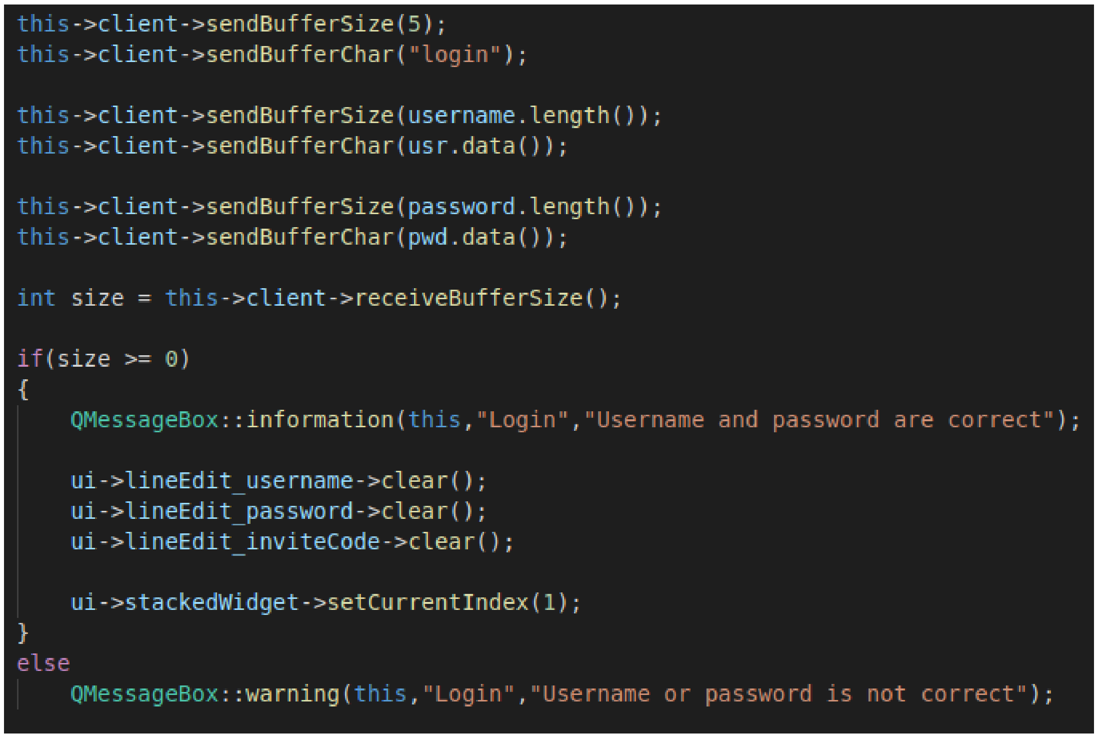


Fig. 3. Database diagram

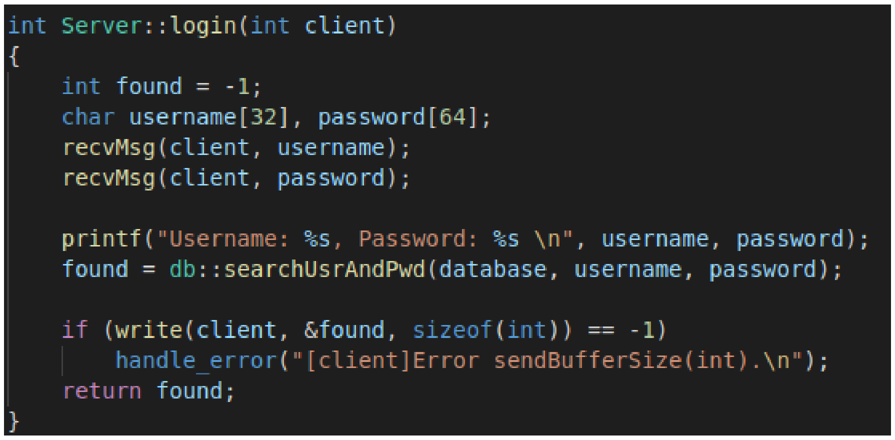
The **invitecodes** table is used for sign up, only clients with an invite code are able to create a new account and every invite code is generated by an user with the specific rank.

We decided to explain and show how the login works. The client sends the command ”login”, the username and the password to the server and waits for the server to respond (figure 4). The server checks the command and calls function **login(int)** where it reads the username and password and searches for them in the database to see if there is a match in **accounts** table the function **searchUsrAndPwd(database, username, password)** returns -1 on failure and the user id on success (0≤id) (figure 5).



```
this->client->sendBufferSize(5);
this->client->sendBufferChar("login");

this->client->sendBufferSize(username.length());
this->client->sendBufferChar(usr.data());

this->client->sendBufferSize(password.length());
this->client->sendBufferChar(pwd.data());

int size = this->client->receiveBufferSize();

if(size >= 0)
{
    QMessageBox::information(this,"Login","Username and password are correct");

    ui->lineEdit_username->clear();
    ui->lineEdit_password->clear();
    ui->lineEdit_inviteCode->clear();

    ui->stackedWidget->setCurrentIndex(1);
}
else
    QMessageBox::warning(this,"Login","Username or password is not correct");
```

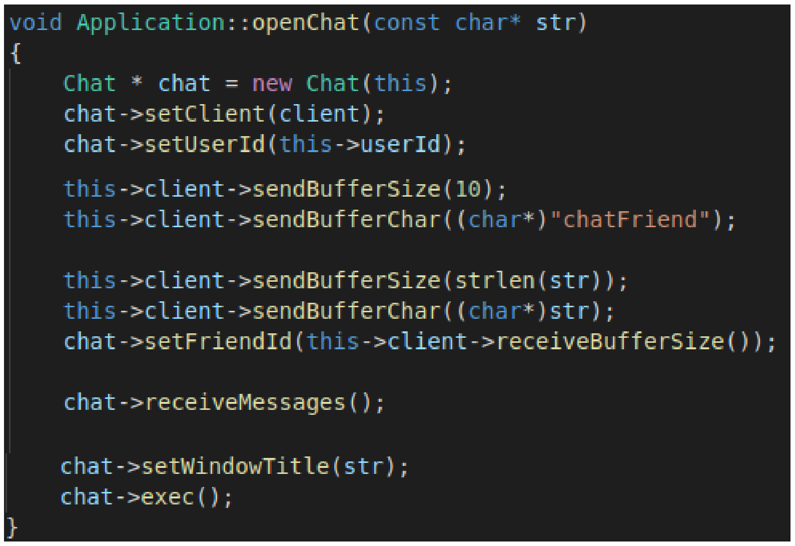
Fig. 4. Client-login



```
int Server::login(int client)
{
    int found = -1;
    char username[32], password[64];
    recvMsg(client, username);
    recvMsg(client, password);

    printf("Username: %s, Password: %s \n", username, password);
    found = db::searchUsrAndPwd(database, username, password);

    if (write(client, &found, sizeof(int)) == -1)
        handle_error("[client]Error sendBufferSize(int).\n");
    return found;
}
```

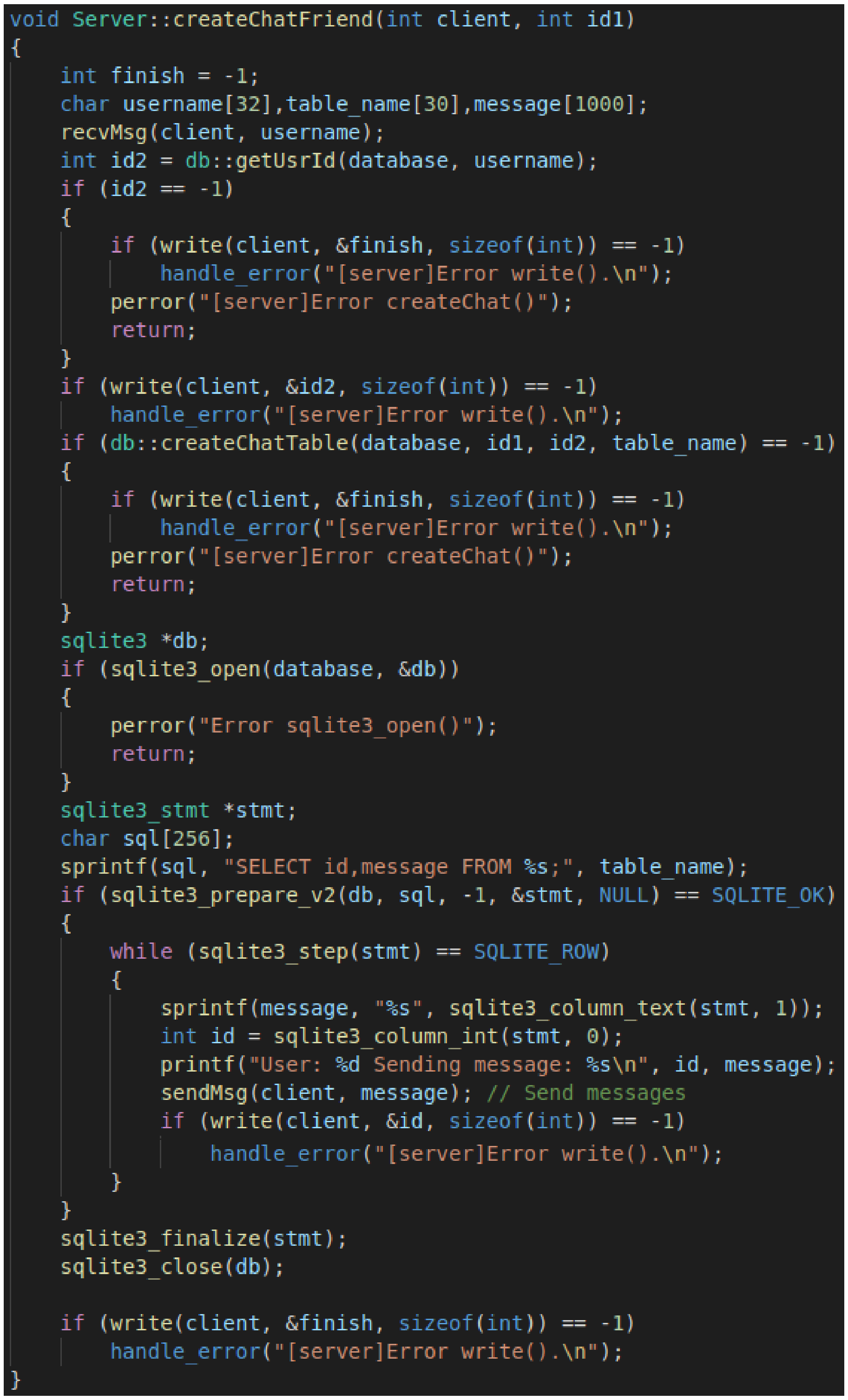
Fig. 5. Server-login

Another important part of the source code is the generation of the chats between users that is shown in the following figures (figure 6 and 7). The client creates a new object **Chat** and fills it with the messages the server provides from the database, if there are any.



```
void Application::openChat(const char* str)
{
    Chat * chat = new Chat(this);
    chat->setClient(client);
    chat->setUserId(this->userId);

    this->client->sendBufferSize(10);
    this->client->sendBufferChar((char*)"chatFriend");

    this->client->sendBufferSize(strlen(str));
    this->client->sendBufferChar((char*)str);
    chat->setFriendId(this->client->receiveBufferSize());

    chat->receiveMessages();

    chat->setWindowTitle(str);
    chat->exec();
}
```

Fig. 6. Client-chat



```
void Server::createChatFriend(int client, int id1)
{
    int finish = -1;
    char username[32],table_name[30],message[1000];
    recvMsg(client, username);
    int id2 = db::getUsrId(database, username);
    if (id2 == -1)
    {
        if (write(client, &finish, sizeof(int)) == -1)
            handle_error("[server]Error write().\n");
        perror("[server]Error createChat()");
        return;
    }
    if (write(client, &id2, sizeof(int)) == -1)
        handle_error("[server]Error write().\n");
    if (db::createChatTable(database, id1, id2, table_name) == -1)
    {
        if (write(client, &finish, sizeof(int)) == -1)
            handle_error("[server]Error write().\n");
        perror("[server]Error createChat()");
        return;
    }
    sqlite3 *db;
    if (sqlite3_open(database, &db))
    {
        perror("Error sqlite3_open()");
        return;
    }
    sqlite3_stmt *stmt;
    char sql[256];
    sprintf(sql, "SELECT id,message FROM %s;", table_name);
    if (sqlite3_prepare_v2(db, sql, -1, &stmt, NULL) == SQLITE_OK)
    {
        while (sqlite3_step(stmt) == SQLITE_ROW)
        {
            sprintf(message, "%s", sqlite3_column_text(stmt, 1));
            int id = sqlite3_column_int(stmt, 0);
            printf("User: %d Sending message: %s\n", id, message);
            sendMsg(client, message); // Send messages
            if (write(client, &id, sizeof(int)) == -1)
                handle_error("[server]Error write().\n");
        }
    }
    sqlite3_finalize(stmt);
    sqlite3_close(db);

    if (write(client, &finish, sizeof(int)) == -1)
        handle_error("[server]Error write().\n");
}
```

Fig. 7. Server-chat

5 Conclusions

This application can be very useful for people passionate about computer networks, because it offers a decent amount of information about the subject and it gives you the option to discuss about the subject with other people interested in this topic. An upgrade that can be done to this application would be the possibility to up vote useful messages or user profiles to show how involved they are in the community or how trusted they can be when asking for an advice or help in global chats.

References

1. Kreibich, J.: Using SQLite. " O'Reilly Media, Inc." (2010)
2. Kurose, J.F., Ross, K.W.: Computer networking. A Top-Down (1986), https://bit.ly/31oJF1q
3. Lenuta, A., Andrei, P.: Computer networks (2021), https://profs.info.uaic.ro/~computernetworks/, [Online; accessed 1-December-2021]
4. Nord, H., Chambe-Eng, E.: Embedded software development tools — cross platform ide — qt creator (2021), https://www.qt.io/product/development-tools, [Online; accessed 1-December-2021]