



Universitatea Tehnică de Construcții București

Facultatea de Hidrotehnica

Programarea Calculatoarelor si Limbaje de Programare II

Profesor Coordonator:

Cosmin Fudulu Florentin

Gabriela Olteanu

Studenti:

Rădulescu Ioan-Radu



Universitatea Tehnică de Construcții București

Facultatea de Hidrotehnica

Aplicatie de gestionare a utilizatorilor dintr-o baza de date

Profesor Coordonator:

Cosmin Fudulu Florentin

Gabriela Olteanu

Studenti:

Radulescu Ioan-Radu

CUPRINS

1 - Prezentare Generala:	4
2 - Funcționalități principale:	4
2.1 - Ecranul principal:	4
2.2 - Vizualizarea	5
2.3 - Editarea	6
2.4 - Stergerea	6
2.5 - Adaugarea	7
3 - Tehnologii Utilizate	8
3.1 - Framework-ul Qt	8
3.2 - Limbajul de Programare C++	8
3.3 - Qt Creator IDE	9
3.4 - Baza de Date SQLite	10
3.5 - CMake	10
4 – Explicarea Codului	11
4.1 - Configurarea Bazei de Date	11
4.2 - Fereastra Principală	12
4.3 - Interacțiunile Utilizatorului	13
4.4 - Operațiunile cu Baza de Date	13
5 – Concluzie	14
6 – Bibliografie	15

1 - Prezentare Generala:

Acest proiect este o aplicație de tip GUI (Graphical User Interface / Interfață Grafică), menită să fie un intermediar între o bază de date de tip .db din aplicația SQLite și datele utilizatorului. Aplicația oferă o metodă ușoară de editare, adăugare, ștergere și vizualizare a datelor dintr-o bază de date, fără a fi necesară utilizarea codului SQL sau a altor aplicații mai greu de înțeles, cum ar fi SQLite Browser.

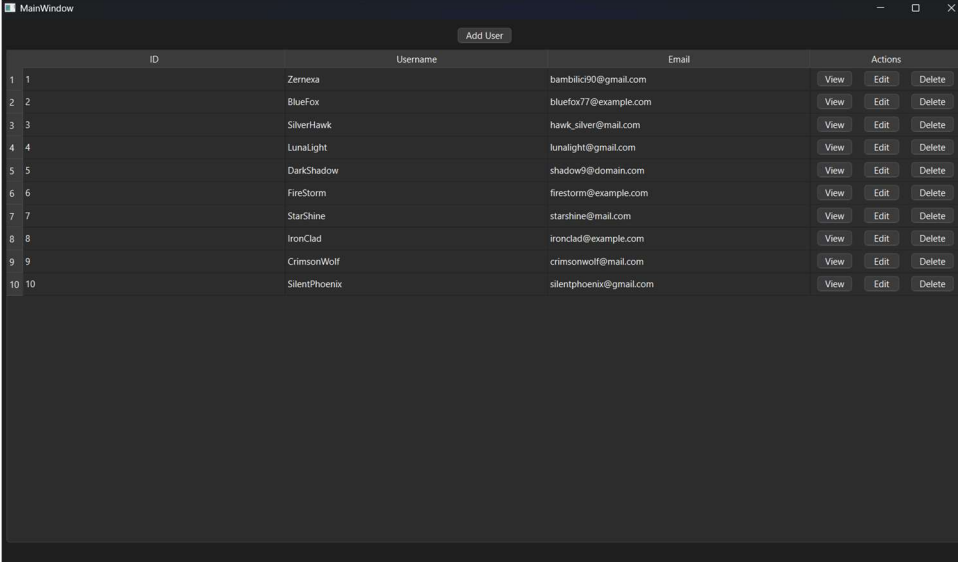
2 - Funcționalități principale:

Principala funcție a acestei aplicații este să simplifice munca unui gestionar de baze de date, astfel încât acesta să nu mai fie nevoit să lucreze direct cu cod SQL sau aplicații complexe. Aplicația are un aspect prietenos și ușor de înțeles, oferind funcționalități esențiale pentru administrarea unei baze de date.

În acest exemplu, lucrăm cu o bază de date ce conține informațiile utilizatorilor unui site de tip „forum” (un site pentru încărcarea de texte, imagini și altele, exemplu: Reddit, 4chan).

2.1 - Ecranul principal:

Ecranul principal servește ca un mic rezumat al datelor fiecărui utilizator, afișând doar informațiile esențiale, utile pentru diferențierea rapidă a acestora. Sunt afișate date unice, precum ID-ul și adresa de e-mail, dar și numele de utilizator, folosit pentru acces rapid în cazul postărilor. (Nu există postări în aplicație, aceasta fiind doar baza de date accesată de site-ul nostru imaginar de tip „forum”).



	ID	Username	Email	Actions
1	1	Zernea	bambico90@gmail.com	View Edit Delete
2	2	BlueFox	bluefox77@example.com	View Edit Delete
3	3	SilverHawk	hawk_silver@mail.com	View Edit Delete
4	4	LunaLight	lunalight@gmail.com	View Edit Delete
5	5	DarkShadow	shadow9@domain.com	View Edit Delete
6	6	FireStorm	firestorm@example.com	View Edit Delete
7	7	StarShine	starshine@mail.com	View Edit Delete
8	8	IronClad	ironclad@example.com	View Edit Delete
9	9	CrimsonWolf	crimsonwolf@mail.com	View Edit Delete
10	10	SilentPhoenix	silentphoenix@gmail.com	View Edit Delete

Fig 1 – Ecranul principal

2.2 - Vizualizarea

Vizualizarea este funcția care permite administratorului bazei de date să consulte toate datele aferente unui utilizator, pentru revizie, verificare sau corectarea eventualelor erori de gestionare.

Pe lângă cele trei date afișate în ecranul principal, acest ecran prezintă toate informațiile utilizatorului, afișate clar, pe rânduri separate.

Butonul de vizualizare este unic pentru fiecare utilizator și este legat prin cod direct de acel utilizator. Astfel, nu mai este necesară introducerea manuală a ID-ului pentru a-l căuta și afișa.

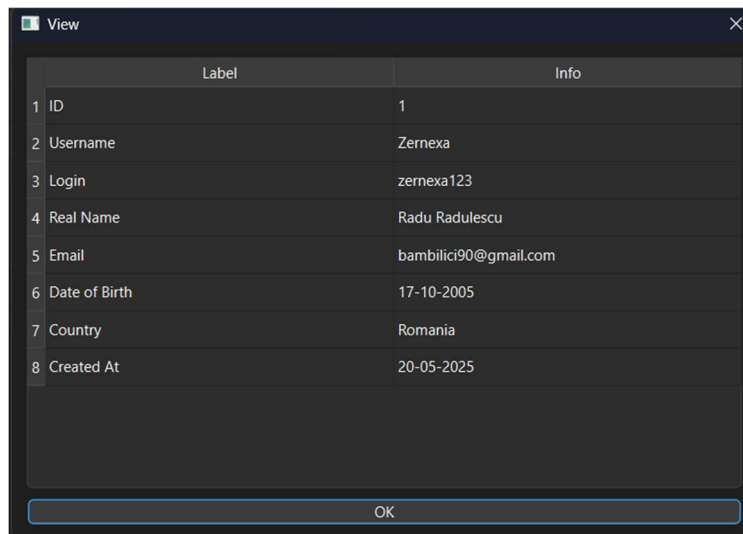
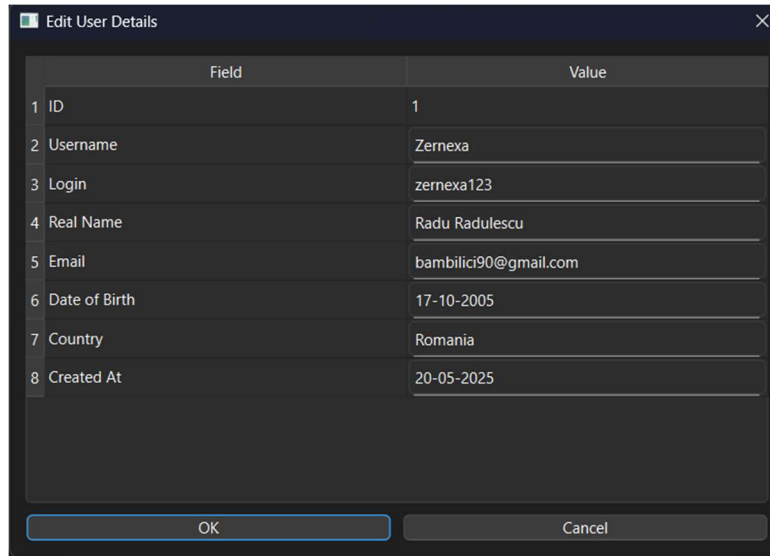


Fig 2 – Ecranul de vizionare a utilizatorului ales.

2.3 - Editarea

Editarea este cea de-a doua funcție specifică fiecărui utilizator. Fiecare utilizator afișat în ecranul principal are un buton unic de editare, legat direct de el. Pagina de editare, deschisă la apăsarea butonului, prezintă aceleași câmpuri ca în pagina de vizualizare, însă de această dată câmpurile sunt editabile.

Utilizatorul poate modifica datele respective, apoi trebuie să aleagă între butonul „OK” (pentru a salva modificările) sau „Cancel” (pentru a anula modificările și a reveni la forma inițială).



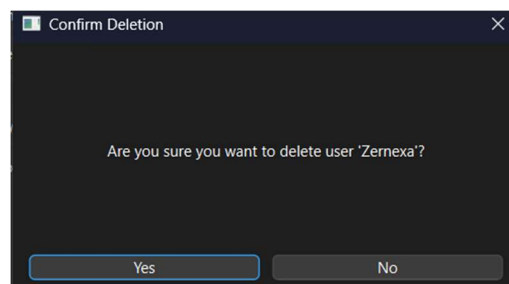
	Field	Value
1	ID	1
2	Username	Zernexa
3	Login	zernexa123
4	Real Name	Radu Radulescu
5	Email	bambilici90@gmail.com
6	Date of Birth	17-10-2005
7	Country	Romania
8	Created At	20-05-2025

OK Cancel

Fig 3 – Ecranul de editare.

2.4 - Stergerea

Ștergerea este cea de-a treia și ultima funcție specifică unui utilizator. Aceasta nu deschide un ecran separat, ci afișează un dialog de confirmare de tip „Yes or No”, pentru a preveni ștergerea accidentală a unui utilizator. Acest strat suplimentar de protecție ajută la evitarea greșelilor.



Confirm Deletion

Are you sure you want to delete user 'Zernexa'?

Yes No

Fig 4 – Ecranul de confirmare a ștergerii utilizatorului.

2.5 - Adaugarea

Adăugarea este o funcție generală pentru baza de date, iar butonul de adăugare nu este legat de un utilizator existent. Acesta servește pentru introducerea de noi utilizatori în baza de date.

La apăsarea butonului, se deschide o fereastră cu aceleași câmpuri ca la editare, dar de această dată câmpurile sunt goale, așteptând inputul utilizatorului.

Câmpurile nu pot fi lăsate necompletate — dacă se încearcă acest lucru, utilizatorul va fi întâmpinat de un mesaj de eroare („Warning Dialog”) care solicită completarea informațiilor minime necesare (Username și Email).

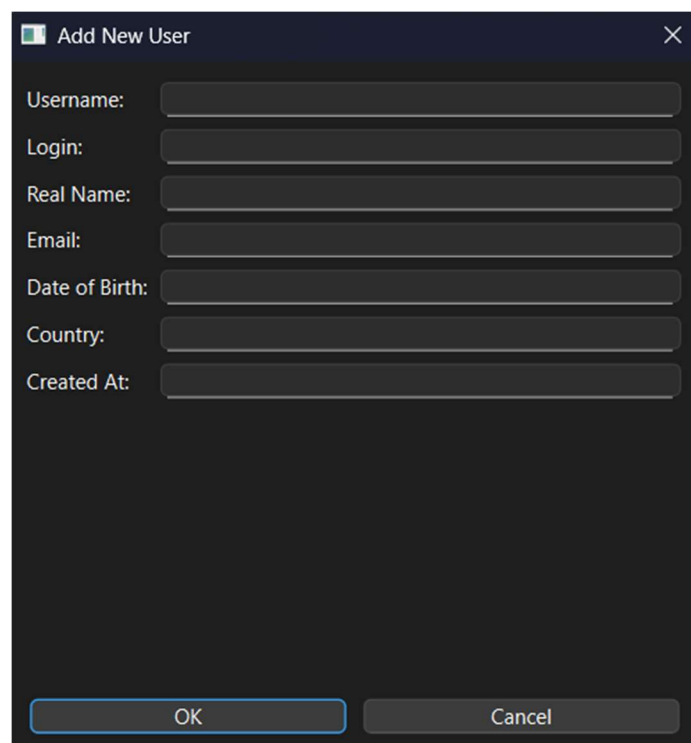
A screenshot of a dark-themed dialog box titled "Add New User" with a close button (X) in the top right corner. The dialog contains seven text input fields stacked vertically, each with a label to its left: "Username:", "Login:", "Real Name:", "Email:", "Date of Birth:", "Country:", and "Created At:". At the bottom of the dialog, there are two buttons: "OK" and "Cancel".

Fig 5 – Ecranul de adaugare a unui nou utilizator.

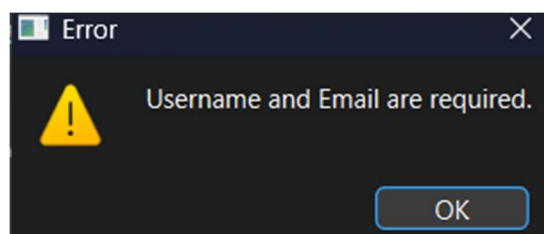


Fig 6 – Eroarea prezentata cand nu sunt adaugate minimul de detalii.

3 - Tehnologii Utilizate

Aplicatie de gestionare a utilizatorilor dintr-o baza de date

- Framework-ul Qt: Folosit pentru a crea o interfață grafică cross-platform pentru sistemul de gestionare a utilizatorilor, permițând o tabelă responsivă și interacțiuni bazate pe dialoguri pe Windows, macOS și Linux.
- C++: Limbajul de bază pentru implementarea logicii aplicației, asigurând performanță ridicată pentru operațiunile cu baza de date și actualizările interfeței.
- Qt Creator IDE: Servește ca mediu de dezvoltare, simplificând designul interfeței, codarea și depanarea pentru proiect.
- SQLite: Folosit ca bază de date ușoară pentru stocarea datelor utilizatorilor, integrat cu Qt pentru operațiuni CRUD eficiente.

3.1 - Framework-ul Qt

Framework-ul Qt este baza interfeței grafice a aplicației, oferind un set robust de instrumente pentru construirea aplicațiilor desktop cross-platform. Acesta permite crearea ferestrei principale, care include o tabelă ce afișează datele utilizatorilor și butoane pentru acțiuni precum adăugarea, vizualizarea, editarea și ștergerea utilizatorilor. Widget-urile Qt, cum ar fi QTableWidgetItem și QDialog, creează o interfață responsivă și intuitivă. Modulul SQL al framework-ului se integrează perfect cu SQLite, permițând aplicației să efectueze operațiuni eficiente cu baza de date. Mecanismul de semnale și sloturi al Qt conectează clicurile pe butoane la funcții precum deschiderea dialogurilor sau actualizarea bazei de date. Natura sa cross-platform asigură că aplicația rulează pe mai multe sisteme de operare fără modificări, făcând-o versatilă pentru implementare.

3.2 - Limbajul de Programare C++

C++ este limbajul principal de programare pentru proiect, integrându-se nativ cu Qt. Acesta gestionează logica aplicației, inclusiv interacțiunile cu baza de date (de ex., Database::addUser, Database::updateUser) și funcționalitatea interfeței (de ex., MainWindow::onAddUserClicked). Performanța ridicată a C++ asigură operațiuni rapide și eficiente precum încărcarea datelor utilizatorilor în tabelă sau actualizarea bazei de date. Natura orientată pe obiecte a C++ organizează codul în clase precum MainWindow și Database, făcându-l modular și ușor de întreținut. Această alegere valorifică eficiența și controlul C++, critice pentru un sistem care gestionează datele utilizatorilor cu actualizări frecvente ale interfeței.

3.3 - Qt Creator IDE

Qt Creator este mediul integrat de dezvoltare (IDE) folosit pentru proiect. Acesta facilitează designul interfeței ferestrei principale prin designerul vizual, unde mainwindow.ui definește tabela și butoanele. Editorul de cod al IDE-ului, cu funcții precum evidențierea sintaxei și completarea codului, simplifică codarea în C++. Instrumentele sale de „debug” rezolvă probleme, cum ar fi eșecul conectării butonului "Add User", permițând inspecția variabilelor și parcurgerea pas cu pas a codului. Integrarea Qt Creator cu CMake asigură că proiectul se compilează și rulează corect pe diverse platforme.

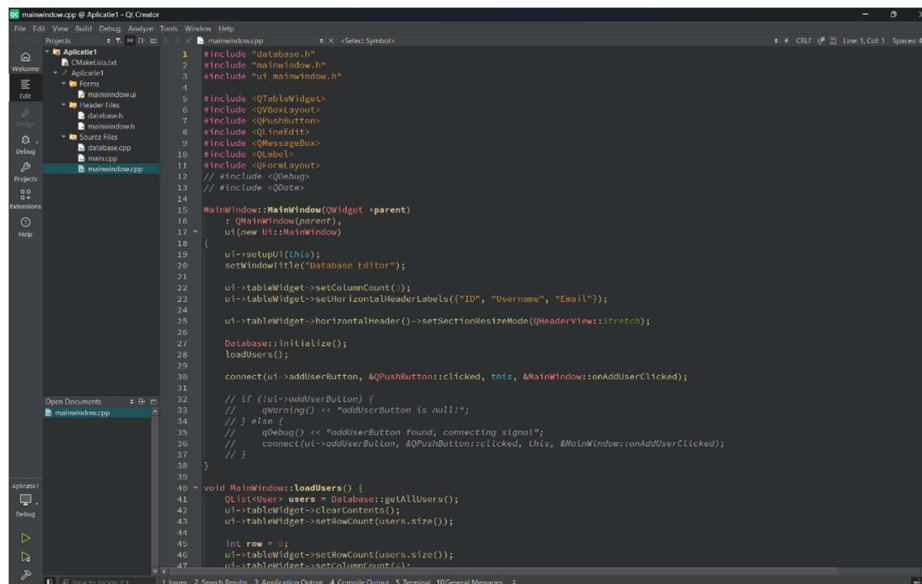


Fig 7 – Interfața Qt Creator IDE.

3.4 - Baza de Date SQLite

SQLite este baza de date utilizată pentru stocarea datelor utilizatorilor într-un fișier numit users.db. Acesta conține o tabelă users cu câmpuri precum id, username, login, realname, email, dob, country și createdAt. Natura ușoară și bazată pe fișiere a SQLite nu necesită un server separat, simplificând implementarea. Modulul SQL al Qt permite interacțiunea ușoară cu SQLite, permițând aplicației să efectueze operațiuni CRUD prin funcții precum Database::getAllUsers și Database::addUser. Această configurare asigură gestionarea eficientă a datelor pentru un sistem de gestionare a utilizatorilor de dimensiuni mici până la medii.

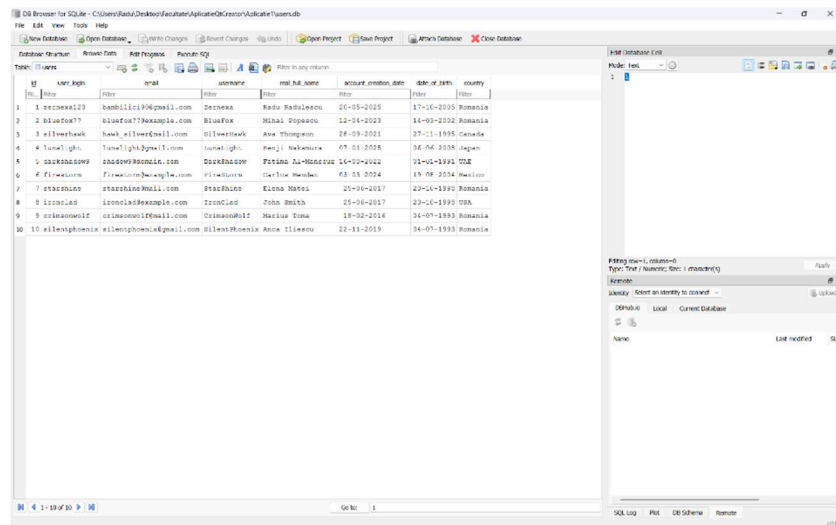


Fig 8 – Interfața SQLite.

3.5 - CMake

CMake este un generator de sisteme de compilare cross-platform care gestionează procesul de compilare pentru acest proiect, asigurând compatibilitatea cu diverse platforme și sisteme de compilare. Configurarea se realizează prin fișierul CMakeLists.txt, care îndeplinește următoarele roluri esențiale:

4 – Eplicarea Codului

4.1 - Configurarea Bazei de Date

Baza de date este gestionată utilizând **SQLite**, o soluție ușoară și eficientă pentru stocarea datelor utilizatorilor. Clasa Database din fișierul database.cpp este responsabilă pentru inițializarea și conectarea la baza de date.

Cum funcționează:

- Se creează o conexiune la un fișier SQLite numit users.db.
- Se verifică dacă baza de date poate fi deschisă.
- Se creează o tabelă numită users cu coloane pentru id (cheie primară care se incrementează automat), username și email, dar doar dacă aceasta nu există deja.

```
bool Database::initialize() {
    QSqlDatabase db = QSqlDatabase::addDatabase("SQLITE");

    QString dbPath = QApplication::applicationDirPath() + "/users.db";
    db.setDatabaseName(dbPath);

    if (!db.open()) {
        qWarning() << "Failed to open database:" << db.lastError().text();
        return false;
    }

    QSqlQuery query;
    if (!query.exec("CREATE TABLE IF NOT EXISTS users ( "
        "id INTEGER PRIMARY KEY, "
        "username TEXT, "
        "user_login TEXT, "
        "real_full_name TEXT, "
        "email TEXT, "
        "date_of_birth TEXT, "
        "country TEXT, "
        "account_creation_date TEXT)")) {
        qWarning() << "Failed to create table:" << query.lastError().text();
        return false;
    }

    qDebug() << "Database opened successfully.";
    qDebug() << "Database path used:" << dbPath;
    return true;
}
```

Fig. 9 – Exemplu cod (Inițializarea bazei de date).

Acest cod asigură că aplicația are o bază de date funcțională în care să stocheze informațiile despre utilizatori.

4.2 - Fereastra Principală

Fereastra principală a aplicației este definită în fișierele `mainwindow.cpp` și `mainwindow.ui`. Aceasta conține o tabelă (`QTableWidget`) care afișează datele utilizatorilor și un buton „Add User” pentru a adăuga noi utilizatori.

Cum funcționează:

- Tabela este populată cu datele utilizatorilor stocate în baza de date.
- Fiecare rând din tabel afișează id, username și email pentru un utilizator.
- Butoanele pentru vizualizare, editare și ștergere sunt adăugate pentru a permite interacțiuni suplimentare.

```
void MainWindow::loadUsers() {
    QList<User> users = Database::getAllUsers();
    ui->tableWidget->clearContents();
    ui->tableWidget->setRowCount(users.size());

    int row = 0;
    ui->tableWidget->setRowCount(users.size());
    ui->tableWidget->setColumnCount(4);
    ui->tableWidget->setHorizontalHeaderLabels({"ID", "Username", "Email", "Actions"});

    ui->tableWidget->horizontalHeader()->setSectionResizeMode(0, QHeaderView::Stretch);
    ui->tableWidget->horizontalHeader()->setSectionResizeMode(1, QHeaderView::Stretch);
    ui->tableWidget->horizontalHeader()->setSectionResizeMode(2, QHeaderView::Stretch);
    ui->tableWidget->horizontalHeader()->setSectionResizeMode(3, QHeaderView::Fixed);

    ui->tableWidget->setColumnWidth(3, 200);
}
```

Fig. 10 – Exemplu de cod (Încărcarea utilizatorilor în aplicație).

Acest cod preia lista de utilizatori din baza de date și o afișează în tabel, oferind utilizatorului o vedere de ansamblu asupra datelor.

4.3 - Interacțiunile Utilizatorului

Utilizatorii interacționează cu aplicația prin dialoguri declanșate de butoane. De exemplu, butonul „Add User” deschide un dialog cu câmpuri de intrare pentru username și email.

Cum funcționează:

- Dialogul conține câmpuri de text pentru introducerea datelor și un buton „OK”.
- Când utilizatorul apasă „OK”, datele sunt salvate în baza de date, iar tabela din fereastra principală este actualizată.

Acest cod creează un dialog simplu, preia datele introduse de utilizator și le salvează, actualizând apoi tabela.

4.4 - Operațiunile cu Baza de Date

Clasa Database oferă funcții pentru operațiunile CRUD (Create, Read, Update, Delete), care permit gestionarea datelor utilizatorilor.

Cum funcționează:

- Operațiunea „Create” adaugă un utilizator nou în tabela users.
- Se utilizează interogări pregătite pentru a preveni erorile și a asigura securitatea.

```
bool Database::addUser(const User &user)
{
    QSqlDatabase db = QSqlDatabase::database();
    if (!db.isOpen()) {
        qWarning() << "Database not open in addUser!";
        return false;
    }

    QSqlQuery query(db);
    query.prepare("INSERT INTO users (username, user_login, real_full_name, email, date_of_birth, country, account_creation_date) "
        "VALUES (:username, :user_login, :real_full_name, :email, :date_of_birth, :country, :account_creation_date)");
    query.bindValue(":username", user.username);
    query.bindValue(":user_login", user.login);
    query.bindValue(":real_full_name", user.realname);
    query.bindValue(":email", user.email);
    query.bindValue(":date_of_birth", user.dob);
    query.bindValue(":country", user.country);
    query.bindValue(":account_creation_date", user.createdAt);

    if (!query.exec()) {
        qWarning() << "Failed to add user:" << query.lastError().text();
        return false;
    }

    int rowsAffected = query.numRowsAffected();
    qDebug() << "User added: Username" << user.username << ", Rows affected:" << rowsAffected;
    return rowsAffected > 0;
}
```

Fig. 11 – Exemplu de cod (Initializarea funcției de adaugare a unui utilizator, legata mai apoi la butonul respectiv in/mainwindow.cpp)

Acest cod inserează un utilizator nou în baza de date și returnează true dacă operațiunea reușește sau false dacă apare o eroare.

5 – Concluzie

Acest proiect a fost pentru mine un exercițiu de imaginație și inovație. Prin realizarea acestuia am fost nevoit să învăț și să înțeleg cum să înțeleg documentații pentru viitor, cum să gestionez erori care parca apăreau din senin și cum să inventez soluții ingenioase la probleme care la prima vedere par ușor de rezolvat, dar greu de pus în practică.

Dacă ar fi să continui acest proiect, aș vrea să adaug abilitatea de a selecta baza de date de pe calculatorul utilizat cu o funcție de genul „File > Open > Select file” și care să dea „fetch” la toate câmpurile bazei de date și să le metamorfozeze într-o interfață grafică uniformă.

6 – Bibliografie

1. <https://www.qt.io/>
2. <https://doc.qt.io/qt-6/>
3. <https://isocpp.org/>
4. <https://en.cppreference.com/w/>
5. <https://www.qt.io/product/development-tools>
6. <https://doc.qt.io/qtcreator/>
7. <https://www.sqlite.org/index.html>
8. <https://www.sqlite.org/docs.html>
9. <https://cmake.org/documentation>
10. <https://github.com/qt>
11. https://www.youtube.com/watch?v=y9Zx_FJBC1U&t=8s
12. https://www.youtube.com/watch?v=6_eLY8O20I8
13. <https://www.youtube.com/watch?v=VigUMAfE2q4>