**Student: Mitrache Stelian-Gabriel**

Programarea Calculatoarelor si Limbaje de Programare

**Anul I, Semestrul II**

**Titular Curs: Conf. Univ. Dr. Ing. Sandulescu Virginia**

**Student: Mitrache Stelian-Gabriel**

**Anul I, Semestrul II**

**Coordonatori:**

**Conf. Univ. Dr. Ing. Neculoiu Giorgian**

**Ing. Olteanu Gabriela**

Aplicație de Gestionare a Contactelor

**CUPRINS**

**Introducere**4

Motivare Alegere Tema5

Obiective5

**Tehnologii Utilizate**6

**Studiu de Caz – Aplicatie gestionare** **contacte**8

Implementare pas cu pas9

**Concluzie**14

**Bibliografie**15

**Anexe**16

**Introducere**

**Context**

În ultimii ani, volumul de informații personale și profesionale pe care le gestionăm zilnic a crescut exponențial. În acest context, metodele tradiționale – fie agende tipărite, fie tabele de calcul – devin insuficiente pentru nevoile moderne de accesibilitate, căutare și actualizare rapidă a datelor de contact. Lipsa unei soluții eficiente poate conduce la:

* Întârzieri în găsirea numărului de telefon sau adresei, costisitoare din punct de vedere al timpului și productivității;
* Erori de introducere atunci când numerele nu sunt validate automat;
* Inconsistență a formatelor (diferite convenții de scriere a numerelor, lipsa diacriticelor etc.);
* Risc de pierdere a datelor în cazul defectării mediilor fizice sau necontrolării versiunilor de fișiere.

**Scop**

Obiectivul principal al acestei aplicații este de a furniza un instrument simplu și robust pentru administrarea listei de contacte, care să îmbine următoarele atribute:

* **Accesibilitate** – interfață intuitivă, cu elemente standard Qt, ușor de înțeles pentru orice utilizator;
* **Reactivitate** – căutare live și actualizare instantanee a listei pe măsură ce utilizatorul tastează;
* **Siguranța datelor** – validări la introducerea numerelor de telefon și salvare automată pentru a preveni pierderile;
* **Portabilitate** – compatibilitate Windows, Linux și macOS, cu stocarea datelor în format JSON independant de platformă;

Prin îndeplinirea acestor obiective, aplicația servește drept exemplu didactic de implementare a principiilor de dezvoltare software orientată pe modele, design patterns și bune practici de UI.

**Motivare**

Motivația dezvoltării unei astfel de aplicații provine atât din necesitatea personală de a organiza contactele într-o manieră eficientă, cât și din dorința de a demonstra și aplica conceptele fundamentale din Qt și C++:

* **Validarea datelor:** utilizarea QIntValidator pentru a permite doar cifre în câmpul de telefon, reducând erorile de formatare;
* **Filtrare și sortare:** folosirea QSortFilterProxyModel arată puterea componentelor Qt pentru aplicațiile care implică volume mari de date;
* **Persistența JSON:** QJsonDocument și familiile sale (QJsonArray, QJsonObject) permit stocarea și schimbul facil de date cu alte aplicații sau servicii;
* **Design de UI modern:** generarea de avataruri dinamice pe baza inițialei numelui creează o experiență vizuală plăcută și profesională.

Mai mult, abordarea practică a provocărilor uzuale din aplicațiile de gestiune a contactelor oferă context real pentru implementarea conceptelor de inginerie software.

**Obiective**

Implementarea unui model ierarhic robust: Structurarea contactelor în grupuri alfabetice dinamice, utilizând QStandardItemModel și QSortFilterProxyModel pentru a permite extinderea și filtrarea recursivă.

Interfață intuitivă și reactivă: Dezvoltarea unui UI prietenos cu utilizatorul, prin QTreeView și QLineEdit cu filtrare live, astfel încât fiecare acțiune (adăugare, căutare, ștergere) să fie executată instantaneu.

Validarea și siguranța datelor: Implementarea de validatori (QIntValidator) și controale la nivel de dialog (QDialog, QFormLayout) pentru a asigura că numai date corecte (numere de telefon) sunt stocate, prevenind inconsistențele.

Persistență portabilă și ușor de integrat: Stocarea contactelor în format JSON (contacte.json) în directorul platform-independent returnat de QStandardPaths::AppDataLocation, garantând compatibilitate Windows, Linux și macOS.

**Tehnologii utilizate**

În dezvoltarea aplicației **“Agenda de Contacte”**, am utilizat un set bine definit de tehnologii și instrumente, care asigură performanță, portabilitate și ușurință în mentenanță. Fiecare componentă a fost aleasă în funcție de nevoile specifice ale proiectului, de la limbajul de bază până la sistemul de build și facilitățile de testare. În continuare prezentăm detaliat tehnologiile utilizate, motivația alegerii lor și modul în care se integrează în arhitectura generală.

**Limbajul de programare și standardul utilizat**

Am ales C++ deoarece oferă un echilibru optim între controlul performant al resurselor și facilitățile moderne de programare. Noile caracteristici introduse de standardul C++17—precum std::optional, if constexpr și expresiile lambda cu capturi îmbunătățite—permit reducerea codului boilerplate și îmbunătățirea siguranței la compilare. Pentru gestionarea dinamică a memoriei, folosim std::unique\_ptr și std::shared\_ptr, evitând astfel scurgerile de memorie și facilitând ownership-ul clar al obiectelor.

**Framework-ul GUI**

Framework-ul Qt este inima aplicației, oferind componente GUI, mecanisme de comunicare semnale/sloturi și suport nativ pentru JSON și fișiere.

* + **Qt Core**: elementele de bază, inclusiv QObject (suport pentru semnale și sloturi), QRegularExpression pentru filtrare avansată și QStandardPaths pentru localizarea directorului de date al aplicației (AppDataLocation).
  + **Qt Widgets**: widget-uri precum QMainWindow, QToolBar, QTreeView, QDialog, QFormLayout și QLineEdit formează interfața principală. Validarea numerică a câmpului “Telefon” se realizează prin QIntValidator.
  + **Qt Gui**: pentru generarea avatarurilor de contact, folosim QPainter pe un QPixmap transparent, aplicând anti-aliasing și desenând elipse colorate pe baza unei valori hash a numelui.
  + **Model–View–Controller**:
    - QStandardItemModel stochează contactele în grupuri după litera inițială.
    - QSortFilterProxyModel realizează căutarea și sortarea pe baza unui QRegularExpression, cu filtrare recursivă și case-insensitive.
    - QTreeView afișează structura modelului, cu suport pentru extinderea și navigarea în grupuri și sub-itemi.

**Gestionare fișiere**

Pentru persistența contactelor, se utilizează QJsonDocument, QJsonArray și QJsonObject. La închidere, aplicația parcurge QStandardItemModel, convertește fiecare contact într-un obiect JSON și scrie un array în contacte.json aflat în directorul AppDataLocation. La pornire, se încarcă fișierul JSON și se populează modelul.

Se verifică cu QFile::open() succesul deschiderii fișierului; în caz de eroare, se continuă cu modelul gol sau cu un set de contacte exemplu. QIntValidator garantează integritatea numerică a valorilor de telefon.

**Extensibilitate și versiuni viitoare**

Pentru sincronizare online sau operații de rețea, se pot integra modulele QtNetwork (HTTP/REST, WebSockets) și QtConcurrent (task-uri asincrone).

Dacă volumul de date crește, se poate migra la SQLite folosind Qt SQL pentru interogări SQL și tranzacții.

Componenta de model (ContactModel) poate fi extrasă într-o bibliotecă statică sau dinamică, separând logică de prezentare pentru re-folosire în alte proiecte.

**Studiu de caz**

Aplicatia „Contacte” este un program simplu făcut cu Qt și C++, pe care l-am creat ca să țin în ordine numele și numerele de telefon ale prietenilor și familiei. Când o deschizi, apare o fereastră verticală, destul de îngustă, cu un buton de „+” pentru adăugat contact și o lupă pentru căutare.

Deasupra, scrie mare şi îngroșat „Contacte”. Sub titlu este ascuns un câmp de căutare. Apăsând pe lupă, apare un dreptunghi în care poți scrie orice parte dintr-un nume și lista se filtrează automat. Dacă nu mai vrei să cauți, apeși din nou pe lupă și căsuța dispare.

Lista de contacte este împărțită pe litere, ca la dicționar. Dacă ai un prieten pe nume „Ana”, el intră sub litera „A”. Pentru fiecare primă literă există un rând îngroșat, care nu se poate selecta, arătând litera respectivă. Apoi, sub fiecare literă vin contactele cu numele și telefonul lor.

Când vrei să adaugi un contact nou, apeși pe „+” și se deschide o fereastră cu două câmpuri: Nume și Telefon. Pentru telefon există o regulă să fie doar cifre. Dacă apeși OK, contactul apare imediat în listă, la grupa potrivită. Dacă nu există încă grupa literei inițiale, programul o creează singur.

Dublu-click pe un contact deschide o fereastră unde poți schimba numele sau telefonul sau apăsa butonul de ștergere pentru a-l șterge. Dacă ștergi ultimul contact dintr-o literă, grupa aceea dispare automat.

Toate datele se salvează singure când închizi aplicația. Contactele se scriu într-un fișier JSON, așa că la următoarea pornire, programul le încarcă și le vede exact cum erai lăsat. Nu trebuie să apeși tu „Save”.

În plus, fiecare contact are un mic cerc colorat cu prima literă din nume. Culoarea se alege automat după un algoritm simplu, ca să fie mereu aceeași pentru același nume. Astfel, găsești mai ușor pe cine cauți.

Pe scurt, „Agenda de contacte” e rapidă, simplă și te ajută să ții minte toate numerele de telefon fără bătăi de cap. Totul se face din câteva click-uri și se salvează automat.

**Implementare pas cu pas**

Pentru realizarea acestui proiect am realizat 3 fisiere, un header in care am declarat toate functiile si structurile de date, iar celelalte 2 pentru functionarea propriu-zisa.

Header:

Header-ul contine o singura clasa „FereastraContacte” care reprezinta programul in sine. Clasa contine toate argumentele specifice, „public”, „protected” si „private”.

Functia pentru a salva datele aplicatie la finalizarea executiei

A close up of a black background

AI-generated content may be incorrect.

Pentru a putea sa adaugam, ascundem/afisam o categorie si pentru a afisa detaliile unui contact declaram urmatoarele functii:

A screen shot of a computer code

AI-generated content may be incorrect.

Keyword-ul „slots” este o functionalitate speficia a framework-ului Qt, prin care putem atribui automat o fuctie unui element atunci cand este activat.

Aspectul vizual (UI) este cat si functionarea propriu-zisa a aplicatie este realizata de functiile definite mai jos

A screen shot of a computer code

AI-generated content may be incorrect.

Primele 4 functii au rolul de a realiza in ordinea in care urmeaza urmatoarele:

afisarea contactelor alfabetic, afisarea butoanelor de adaugare si cautare, afisarea campului de cautare la activare, restabilirea datelor salvate.

Qicon realizeaza creearea avatarului care contine prima litera a numelui. Ulterior, datele sunt stocate in modelContacte, proxyContacte acceseaza contactele pentru a executa diferite operatiuni, viewContacte realizeaza afisarea contactelor in ordine alfabetica iar campCautare reprezinta proprietatea propriu zisa pentru cautare.

A screen shot of a computer program

AI-generated content may be incorrect.

**Functionalitate**

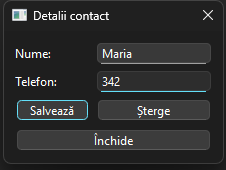
A screenshot of a computer

AI-generated content may be incorrect.In momentul executari aplicatiei se afiseaza urmatorul meniu care reprezinta aplicatia propriu zisa, contactele sunt automat ascunse si ordonte alfabetic.

A screenshot of a computer

AI-generated content may be incorrect.Pentru a afisa o sectiune trebuie apasat dublu click pe categoria respectiva.

Pentru a vizualiza detaliile unui anumit contact se actioneaza dublu click si va aparea o fereastra in care sunt prezente Numele si Tefelfonul. Daca intentionam sa editam contactul, dupa modificari apasam Salveaza iar apoi inchide. Pentru a sterge contactul apasam Sterge si se va sterge direct.



Campul pentru Nume accepta orice caractere, dar campul Telefon accepta doar cifre, la tastarea altor caractere ele pur si simplu nu vor aparea

In bara de sus, in partea dreapta sunt prezente doua iconite, un plus si o lupa

Adaugarea unui contact nou necesita actionarea iconitei +, la actionarea acesteia se va deschide un meniu asemanator cu cel de detalii, avand doar doua campuri pentru Nume si Telefon, si doua butoane, pentru salvare sau anulare, daca meniul este inchis inchis direct nu se va salva nimic.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.Necesitatea de a economisi timp si pentru o eficienta sporita a fost implementat buton de caturare care are simbolul de lupa. Actionarea acestui buton va face aparitia unui meniu de cautare in care putem tasta numele contactului cautat, daca acesta nu exista nu se va afisa nimic. Cautarea se realizeaza exclusiv dupa nume. Campul de cautare va disparea automat la inca o acctionare a butonului de cautare.

**Concluzie**

Proiectul **„Agenda de contacte”** construit cu Qt/C++ pornește de la o idee simplă, dar esențială pentru orice utilizator: să ai mereu la îndemână numele și numerele celor dragi, în format digital, organizat şi uşor de navigat. La prima pornire, fereastra principală salută utilizatorul cu un titlu ferm, „Contacte”, şi două pictograme intuitive – un plus pentru a adăuga un prieten nou şi o lupă pentru a căuta rapid în listă. În spatele acestei simplităţi vizuale, însă, se ascunde o arhitectură gândită să asigure performanţă, modularitate şi fiabilitate.

Pe de o parte, modelul de date (QStandardItemModel) păstrează fiecare contact grupat alfabetic, creând sub fiecare literă „capitole” ne-selectabile care mențin ordinea și claritatea. Pe de altă parte, un QSortFilterProxyModel interceptează interogările din câmpul de căutare și, folosind expresii regulate insensibile la majuscule, filtrează în timp real lista astfel încât orice nume parțial tastat afișează instantaneu rezultatele potrivite. Între aceste două componente „stă” QTreeView, care îmbină estetica plăcută cu eficienţa: un arbore cu rânduri uniforme, fără antet, ce răspunde la dublu-click prin deschiderea unui dialog de editare sau ștergere.

Un detaliu inedit al interfeței constă în avatarurile colorate ale fiecărui contact: un cerc generat automat prin QPainter, cu nuanţa determinată din hash-ul numelui și cu inițiala poziționată central. În felul acesta, recunoașterea vizuală devine mai rapidă, iar lista capătă un aspect prietenos şi modern.

Sub capotă, întregul set de contacte se încarcă la pornire dintr-un fișier JSON plasat în directorul propriu al aplicației, iar la închidere se salvează automat, fără ca utilizatorul să-și bată capul cu butonul tradițional „Save”. Această persistență automată, combinată cu validarea numerică a telefonului și cu mecanismul de semnale-sloturi al Qt, asigură o experienţă fluidă, lipsită de griji și întreruperi inutile.

Privind înainte, „Agenda de contacte” are deja o fundaţie solidă pentru a crește: de la import-export CSV și criptare locală a datelor, până la sincronizare cloud sau categorii tematice personalizate (familie, colegi, prieteni).

**Bibliografie**

The Qt Company. (2024). Qt Documentation. Disponibil online: https://doc.qt.io

Bjarne Stroustrup. (2013). The C++ Programming Language (4th ed.). Addison-Wesley.

Johan Thelin. (2012). Game Programming using Qt 5 Beginner’s Guide. Packt Publishing.

Blanchette, J., & Summerfield, M. (2008). C++ GUI Programming with Qt 4 (2nd ed.). Prentice Hall.

JSON.org. (2024). Introducing JSON. Disponibil online: https://www.json.org/json-en.html

Qt Project. (2024). QStandardItemModel Class Reference. Disponibil online: https://doc.qt.io/qt-5/qstandarditemmodel.html

Qt Project. (2024). QSortFilterProxyModel Class Reference. Disponibil online: https://doc.qt.io/qt-5/qsortfilterproxymodel.html

Qt Project. (2024). QPainter Class Reference. Disponibil online: https://doc.qt.io/qt-5/qpainter.html

Qt Project. (2024). QStandardPaths Class Reference. Disponibil online: <https://doc.qt.io/qt-5/qstandardpaths.html>

**Anexa**

**Anexa A – Fișierul de header FereastraPrincipala.h**

#ifndef FEREASTRAPRINCIPALA\_H

#define FEREASTRAPRINCIPALA\_H

#include <QMainWindow>

#include <QTreeView>

#include <QLineEdit>

#include <QStandardItemModel>

#include <QSortFilterProxyModel>

#include <QIcon>

#include <QCloseEvent>

#include <QModelIndex>

class FereastraContacte : public QmainWindow{

Q\_OBJECT

public:

explicit FereastraContacte(QWidget \*parent = nullptr);

protected:

void closeEvent(QCloseEvent \*event) override;

private slots:

void adaugaContact();

void comutaCautare();

void actualizeazaFiltru(const QString &text);

void afiseazaDetaliiContact(const QModelIndex &index);

private:

void initializareModel();

void initializareToolbar();

void initializareVizualizare();

void incarcareContacteDinFisier();

void salvareContacteInFisier() const;

void adaugaContactInModel(const QString &nume, const QString &telefon);

QIcon creareAvatarDinNume(const QString &nume) const;

QStandardItemModel \*modelContacte;

QSortFilterProxyModel \*proxyContacte;

QTreeView \*viewContacte;

QLineEdit \*campCautare;

};

#endif // FEREASTRAPRINCIPALA\_H

**Anexa B – Fișierul de implementare FereastraPrincipala.cpp**

**…**

**Anexa C – Funcția main()**

#include <QApplication>

#include "fereastraPrincipala.h"

int main(int argc, char \*argv[])

{

QApplication aplicatie(argc, argv);

FereastraContacte fereastra;

fereastra.show();

return aplicatie.exec();

}