НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВА РОБОТА**

***з дисципліни "Основи програмування"***

Виконав: Мовчан Максим Олександрович

Група: КП-92

Допущено до захисту

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2 семестр 2019/2020

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

|  |  |
| --- | --- |
| Узгоджено  Керівник роботи    \_\_\_\_\_\_\_\_\_\_/Гадиняк Р.А./ | ЗАХИЩЕНА  "\_\_"\_\_\_\_\_\_\_\_\_2020р.  з оцінкою\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_/Гадиняк Р.А./ |

**Програмний додаток з графічним інтерфейсом користувача для керування електронною базою письменників**

Виконавець роботи

Мовчан Максим Олександрович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020р.

Зміст

[**Вступ**](#_cr2eoqpdh4mm) **5**

[**1. Аналіз предметної галузі**](#_96dprzcmn4pe) **5**

[1.1. Аналіз вимог до функціональності програмних засобів](#_n2g7boooxclb) 5

[**2. Аналіз мов програмування та технологій розроблення**](#_ewtylcuh8bvc) **6**

[2.1. Мова програмування С++](#_fh0j2unhluf2) 6

[2.2. Фреймворк Qt](#_ayd32pcv0jlc) 6

[2.3. База даних SQLite](#_hul0g052l08h) 7

[**3. Опис розроблених програмних засобів**](#_kbi4mqs5pjyt) **7**

[3.1. Загальна структура системи](#_nlu4ckytvy2k) 7

[залежність від додаткової сутності](#_ny4p6qq0v8a9) 11

[залежність від певного користувача](#_g9g1shivuue7) 11

[3.2. Структура бази даних](#_flueaud3o047) 11

[3.3. Модуль сховища даних](#_dpq6kpdwm35k) 12

[3.4. Модуль для роботи з базою даних SQLite](#_5k98vrpspdu0) 14

[3.5. Модуль головного графічного вікна](#_9f375d5jm097) 15

[3.6. Модуль графічного інтерфейсу додавання ново](#_mgyyttz76lik)ї організації 17

[3.7. Модуль графічного інтерфейсу для оновлення інформації про організацію.](#_c43hm7dtpzpw) 18

[**4. Аналіз розроблених програмних засобів**](#_vi1byxrajut) **19**

[4.1. Особливості реалізації](#_5ajysoaxe2s5) 19

[4.2. Дизайн та вміст вікон](#_tyx0bqgpxyqc) 20

[**Висновки**](#_6dbqg2jsslr5) **24**

# Вступ

Взагалі, використовувати електронну базу організацій та засновників дуже можна використовувати по-різному. Наприклад, для державного обліку або для різних списків . Це може полегшити пошук деякої організацій певного засновника. Створення системи, що забезпечує керування електронною базою організацій , є актуальною задачею.

Електронна база може допомогти користувачеві знайти, додати або змінити інформацію про певну організацію або засновника

В житті, такий програмний засіб став би в нагоді для створення єдиної бази та отримання з неї інформації про організаціі і засновників.

У державних закладах також часто виникає необхідність створення єдиної бази, яка б давала можливість будь якому користувачу отримати швидкий доступ до інформації про необхідні організації або засновників а також для керування ними. Або ж людині, яка має мету швидко знайти інформацію про певну організації чи хто її засновник. Це дуже корисно для людей, які працюють в сферах, повʼязаних зі зберіганням інформації про організації адже не завжди є час довго шукати в інтернеті потрібну інформацію або про них або їх засновників.

Даний проект присвячено розробленню програмного додатку, який призначено для управління та використання бази засновників та організацій. Цей проект може бути з легкістю використаним в державних установах, на підприємствах, та у відкритих сховищах інформації про організації.

В подальшому, програмний додаток має можливість розвиток в цих сферах більш поглиблено.

# 1. Аналіз предметної галузі

## 1.1. Аналіз вимог до функціональності програмних засобів

В процесі аналізу вимог до системи було виділено такі функціональні вимоги :

1. Наявний графічний інтерфейс користувача. Графічні елементи використані відповідно до їхнього основного призначення.
   1. Наявно декілька вікон і діалогів. Для взаємодії із файловою системою використовуються стандартні діалоги
   2. Дотримується єдиний стиль для всіх вікон
2. Наявні форми для створення і редагування даних.
   1. Можна видаляти дані (з підтвердженням цієї дії).
   2. Наявна форма редагування даних має мати вже заповнені поля із попередніми значеннями. Є можливість відмінити оновлення сутності.
3. Всі вхідні дані від користувача та з файлів перевіряються, помилки обробляються, користувачу відображається інформація про помилки.
4. Програма взаємодіє з реляційною базою даних. SQL запити до бази даних не конкатенуються, а їх виконання винесено у спеціальний окремий модуль (вид сховища).
   1. База даних містить мінімум 3 таблиці. Таблиці в БД приведені до 1-ї та 2-ї нормальних форм.
   2. Між таблицями бази даних є one-to-many та many-to-many зв'язки
5. Наявна реєстрація користувачів у системі. Паролі користувачів не зберігаються у відкритому виді, а хешуються обраним алгоритмом хешування рядків.
   1. Наявна форма аутентифікації. Пароль користувача при введені у форму прихований (використано спеціальний елемент). Вікно цієї форми має сповіщення про некоректність введених користувачем даних.
   2. Програма авторизує доступу до даних по користувачах. У базі є дані, що належать конкретному користувачу (напр, авторство сутностей).
6. Використовуються можливості подійно-орієнтованого програмування (Qt signals-slots).

# 2. Аналіз мов програмування та технологій розроблення

## 2.1. Мова програмування С++

В курсовій роботі були використані такі можливості мови С++, як:

1. Вбудований тип даних bool, який повертає значення true або false і для цього не потрібно підключати додаткових заголовочних файлів, як в мові Сі.
2. Простори імен. Вони визначають область дії локальної назви змінної, типу чи функції.
3. Віртуальні функції.Оскільки перевірка специфікаторів доступу відбувається при компіляції, можна доступитися до приватної функції класа-нащадка використовуючи віртуальні функції.
4. Наслідування класів. Наслідування дозволяє одному об’єкту переймати властивості іншого об’єкта.

Об’єкт може наслідувати основні властивості іншого об’єкта і додавати до них риси, що характерні тільки для нього.

## 2.2. Фреймворк Qt

Під час розробки цієї курсової роботи, були використані можливості подійно-орієнтованого програмування. В подієно-орієнтованому програмуванні, виконання програми визначається подіями - діями користувача, наприклад взаємодія з клавіатурою, повідомленнями інших програм і потоків, подіями операційної системи (наприклад, надходженням мережевого пакета).Для цього використовувалися слоти та сигнали , щоб обробити події. Були створені різні форми та елементи цих форм. Також в цій курсовій роботі були використані певні бібліотеки Qt , наприклад для роботи з базою даних, підключала бібліотеку QSql .

## 2.3. База даних SQLite

Тут були використані функції для роботи з базою даних, такі як створення таблиць, створення зв’язків між цими таблицями. Також були використані різні запити: select, remove, insert.

# 3. Опис розроблених програмних засобів

## 3.1. Загальна структура системи

Програмні засоби реалізовані у вигляді десктопного додатку з графічним інтерфейсом користувача. Структурна схема системи зображена на рис.

GUI client

SQLite DB

Рис. Структурна схема системи

GUI та SQlite BD зв’язані між собою так, що будь які зміни в першому елементі структури породжують зміни в другому елементі цієї структури.

GUI представляє собою вікна , та елементи які знаходяться на вікні, які й забезпечують роботу користувача з програмним додатком.

База даних SQlite DB забезпечує збереження всіх даних у вигляді реляційних таблиць, в яких є зв’язки між собою. Виконуються певні запити для отримання даних з таблиці та збереження всі зміни, які виконує користувач програмного додатку.

Проект розбитий на такі модулі:

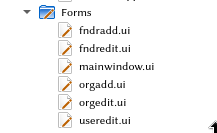
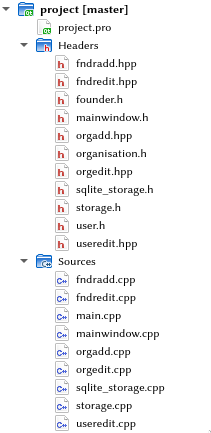
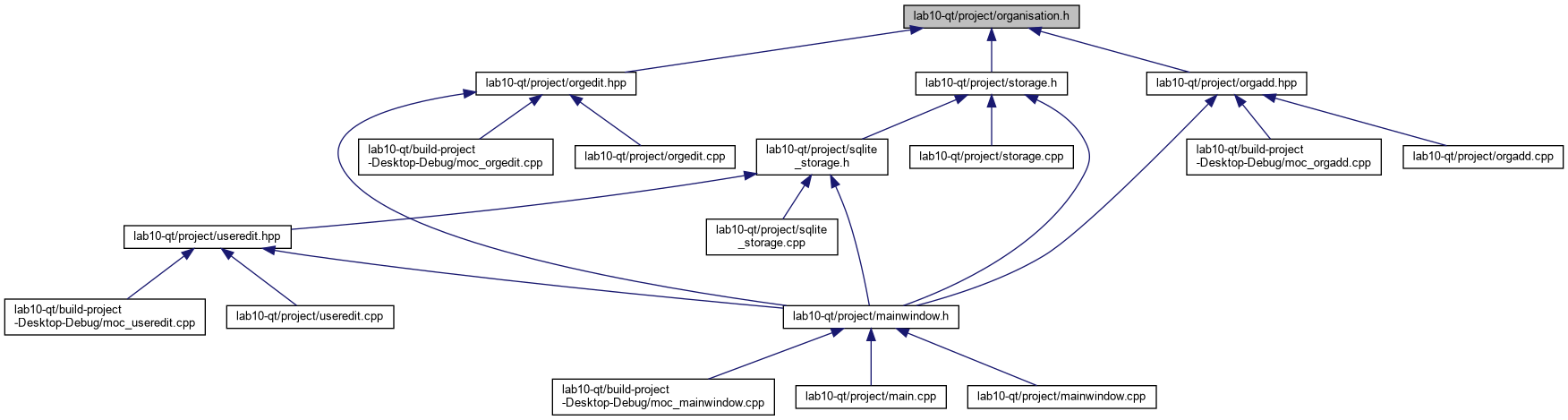


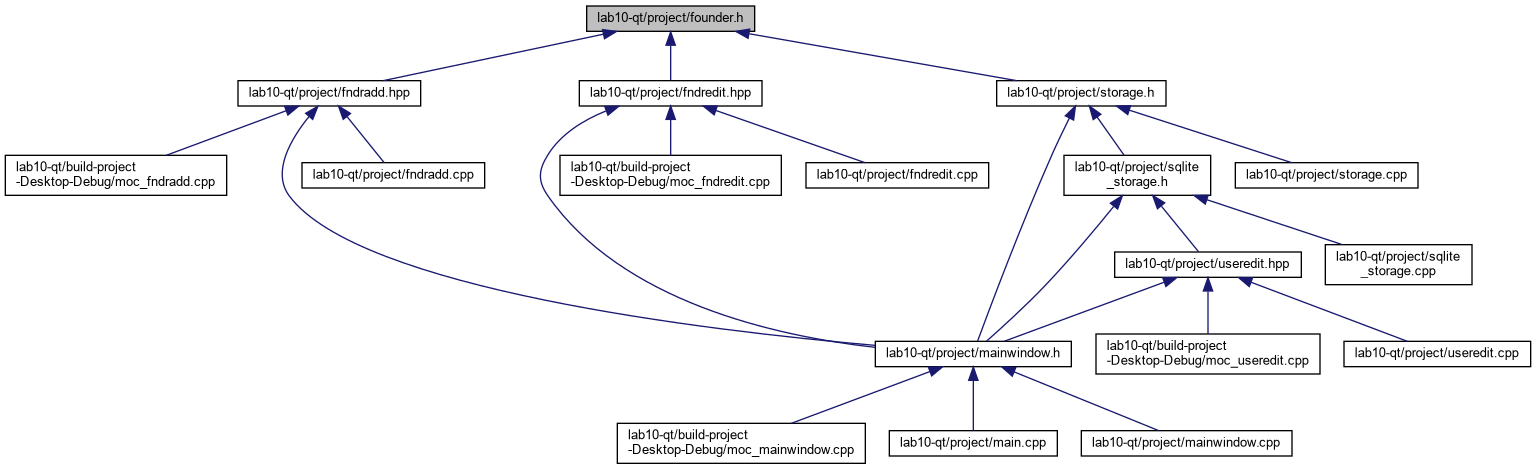
Рис. Модулі проекту

Діаграма залежностей модулів, побудована у Visual Studio IDE:

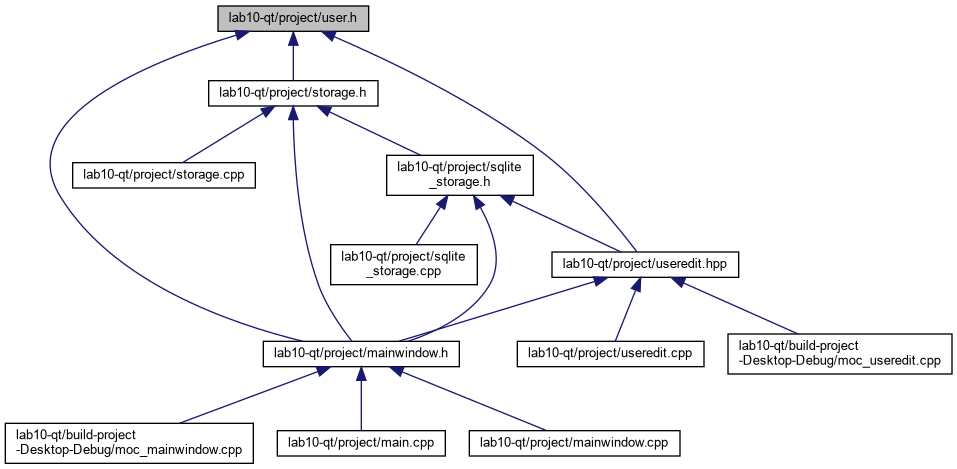
1. залежності від основної сутності



## залежність від додаткової сутності



## залежність від певного користувача



## 3.2. Структура бази даних

На рис. представлена ERD-діаграма структури бази даних системи.

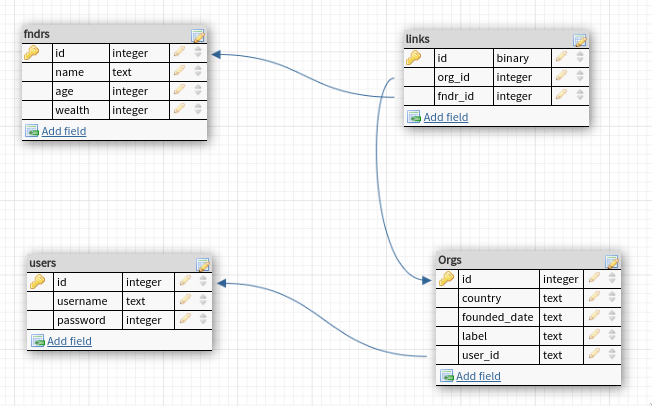


Рис. Структура бази даних

Таблиця  **Orgs,** яка представляє собоютакі поля, слугує для збереження основної інформації про організацію:

* id - ціле число, первинний ключ, унікальний, автоінкрементний
* country - країна заснування
* founded\_date - рік заснування, ціле число
* label – назва
* user\_id – користувач що створив

Таблиця **Fndrs**  містить такі поля та слугує для збереження короткої інформації про певного засновника :

* id - ціле число, первиний ключ, унікальний, автоінкрементний
* name – імʼя
* age - вік
* wealth – статок, число

Таблиця **Users** слугує для ідентифікації та авторизації користувача в системі та для доступу до його інформації.

* id - ціле число, первиний ключ, унікальний, автоінкрементний
* password - пароля користувача для входу в систему, захешований, текст
* username - логін користувача, текст

Таблиця **links,** слугує для зв’язку між таблицями **Orgs** i **Fndrs** типу **many-to-many.** Містить такі поля:

* id - ціле число, первиний ключ, унікальний, автоінкрементний
* orgs\_id - вториний ключ, який взятий з таблиці **Orgs**, ціле число
* fndrs\_id - вториний ключ, взятий з таблиці **Fndrs** , ціле число

## 3.3. Модуль сховища даних

Інтерфейс модуля:

|  |
| --- |
| **storage.h** |
| #pragma once  #include <QString>  #include <vector>  #include <optional>  #include "founder.h"  #include "organisation.h"  #include "user.h"  using namespace std;  class Storage  {  private:  QString dir\_name\_;  public:  Storage();  explicit Storage(const QString &);  virtual ~Storage() {}  void setName(const QString &);  QString name() const;  virtual bool isOpen() const = 0;  virtual bool open() = 0;  virtual void close() = 0;  virtual bool exists() const = 0;  virtual bool newStorage() = 0;  // students  virtual QVector<Org> getAllOrgs() = 0;  virtual optional<Org> getOrgById(int) = 0;  virtual bool updateOrg(const Org &) = 0;  virtual bool removeOrg(int) = 0;  virtual int insertOrg(const Org &) = 0;  // courses  virtual QVector<Fndr> getAllFndrs() = 0;  virtual optional<Fndr> getFndrById(int) = 0;  virtual bool updateFndr(const Fndr &) = 0;  virtual bool removeFndr(int) = 0;  virtual int insertFndr(const Fndr &) = 0;  // users  virtual bool registration\_user(const QString &, const QString &) = 0;  virtual bool change\_user\_options(int &, const QString &, const QString &) = 0;  virtual optional<User> getUserAuth( const QString &, const QString &) = 0;  virtual QVector<Org> getAllUserOrgs(int) = 0;  // links  virtual QVector<Fndr> getAllOrgsFndrs(int) = 0;  virtual bool insertOrgFndr(int, int) = 0;  virtual bool removeOrgFndr(int, int) = 0;  virtual bool fndrOrgExec(int, int) = 0;  }; |

# Цей модуль призначений для роботи програми з даними. Він є абстрактним та від нього наслідується клас SqliteStorage. Це є ієрархічним наслідуванням.

Тут передбачені такі функції як відкривання бази даних, отримання даних основної та додаткової сутності, додавання нових елементів в сховище, видалення елементів, оновлення інформації про основну чи додаткову сутність, закриття цього сховища та аутентифікація користувача в системі, для отримання даних, які належать лише йому.

## 3.4. Модуль для роботи з базою даних SQLite

Інтерфейс модуля:

|  |
| --- |
| **sqlite\_storage.h** |
| #pragma once  #include <QDir>  #include <QSqlDatabase>  #include <QtSql>  #include "storage.h"  using namespace std;  class **SqliteStorage** : public Storage {  protected:  QSqlDatabase database\_;  public:  explicit **SqliteStorage**(const QString &dir\_name = "");  bool ***isOpen***() const;  bool ***exists***() const;  bool ***newStorage***();  bool ***open***();  void ***close***();  QVector<Org> ***getAllOrgs***();  optional<Org> ***getOrgById***(int);  bool ***updateOrg***(const Org &);  bool ***removeOrg***(int);  int ***insertOrg***(const Org &);  // fndrs  QVector<Fndr> ***getAllFndrs***(void);  optional<Fndr> ***getFndrById***(int);  bool ***updateFndr***(const Fndr &);  bool ***removeFndr***(int);  int ***insertFndr***(const Fndr &);  // users  bool ***registration\_user***(const QString &, const QString &);  bool ***change\_user\_options***(int &, const QString &, const QString &);  optional<User> ***getUserAuth***(const QString &, const QString &);  QVector<Org> ***getAllUserOrgs***(int);  // links  QVector<Fndr> ***getAllOrgsFndrs***(int );  bool ***insertOrgFndr***(int, int);  bool ***removeOrgFndr***(int, int);  bool ***fndrOrgExec***(int, int);  };  QString **hashPassword**(QString const &); |

## 3.5. Модуль головного графічного вікна

Інтерфейс модуля:

Воно використовується для обробки сигналів і слотів графічного інтерфейсу головного вікна додатку

|  |
| --- |
| **mainwindow.h** |
| #ifndef MAINWINDOW\_H  #define MAINWINDOW\_H  #include <QMainWindow>  #include <QCloseEvent>  #include <QListWidgetItem>  #include <QMessageBox>  #include <QListWidget>  #include "storage.h"  #include "user.h"  #include "useredit.hpp"  #include "orgedit.hpp"  #include "fndradd.hpp"  #include "fndredit.hpp"  #include "orgadd.hpp"  #include "sqlite\_storage.h"  #include <QGraphicsBlurEffect>  #include "ui\_mainwindow.h"  #include <QFileDialog>  #include <QMetaType>  #include <QDebug>  namespace Ui {  class MainWindow;  }  class MainWindow : public QMainWindow  {  Q\_OBJECT  public:  explicit MainWindow(QWidget \*parent = 0);  ~MainWindow();  signals:  void user\_to\_edit(optional<User>\*);  void org\_to\_update(QListWidgetItem\*);  void fndr\_to\_update(QListWidgetItem\*);  protected:  void closeEvent(QCloseEvent \*);  private slots:  void addOrg(const QVector<Org> &);  void addFndr(const QVector<Fndr> &);  void user\_update(User\*);  void org\_update(Org\*);  void org\_push(Org\*);  void fndr\_push(Fndr\*);  void fndr\_update(Fndr\*);  void dialogDisabling();  void on\_Continue\_clicked();  void on\_open\_button\_clicked();  void on\_password\_textChanged(const QString &);  void on\_login\_textChanged(const QString &);  void on\_new\_button\_clicked();  void on\_confirm\_textChanged(const QString &);  void on\_org\_list\_itemClicked(QListWidgetItem \*);  void on\_logout\_clicked();  void on\_exit\_clicked();  void on\_close\_db\_clicked();  void on\_edit\_user\_clicked();  void on\_org\_edit\_clicked();  void on\_org\_add\_clicked();  void on\_founders\_button\_clicked();  void on\_fndr\_list\_itemClicked(QListWidgetItem\*);  void on\_fndr\_add\_clicked();  void on\_org\_del\_clicked();  void on\_back\_clicked();  void on\_fndr\_del\_clicked();  void on\_fndr\_edit\_clicked();  private:  Ui::MainWindow \*ui;  FndrEdit\* fndredit;  FndrAdd\* fndradd;  OrgEdit\* orgedit;  OrgAdd\* orgadd;  UserEdit\* useredit;  Storage \* storage\_=nullptr;  optional <User> auth\_user;  };  #endif // MAINWINDOW\_H  //Q\_DECLARE\_METATYPE(Operator); |

## 3.7. Модуль графічного інтерфейсу додавання нової організації

Цей модуль використовується для графічного діалогу додавання нового елементу основної сутності, тобто організації.

Основною функцією є отримання інформації, що вводить користувач

Інтерфейс модуля:

|  |
| --- |
| **orgadd.h** |
| #pragma once  #include "organisation.h"  #include <QDebug>  #include <QListWidget>  #include <QMainWindow>  #include <QMessageBox>  #include <QDialog>  namespace Ui {  class OrgAdd;  }  class OrgAdd : public QDialog  {  Q\_OBJECT  signals:  void org\_add(Org\*);  void disableToggle();  public:  explicit OrgAdd(QWidget \*parent = nullptr);  ~OrgAdd();  protected:  void closeEvent(QCloseEvent\*);  private slots:  void on\_okay\_button\_clicked();  void on\_cancer\_button\_clicked();  private:  Ui::OrgAdd \*ui;  }; |

## 

## 3.8. Модуль графічного інтерфейсу для оновлення інформації про організацію.

## В цьому модулі є оновлення інформації про додаткову сутність, тобто функціонал роботи з організацією.

Інтерфейс модуля:

|  |
| --- |
| **editdialog.h** |
| #pragma once  #include "organisation.h"  #include <QDebug>  #include <QDialog>  #include <QListWidget>  #include <QMainWindow>  #include <QMessageBox>  namespace Ui {  class OrgEdit;  }  class OrgEdit : public QDialog  {  Q\_OBJECT  signals:  void upd\_org(Org\*);  void disableToggle();  protected:  void closeEvent(QCloseEvent\*);  private slots:  void on\_okay\_button\_clicked();  void on\_cancel\_button\_clicked();  void edit\_org(QListWidgetItem\*);  public:  explicit OrgEdit(QWidget \*parent = nullptr);  ~OrgEdit();  private:  Ui::OrgEdit \*ui;  }; |

# 4. Аналіз розроблених програмних засобів

## 4.1. Особливості реалізації

При реєстрації користувач має задати username, який має бути унікальним, так як в системі виконується перевірка на існування користувача з таким іменем. Якщо такий користувач вже існує, програма попереджає про це. Також є подвійна перевірка на коректність вводу паролю при реєстрації, інакше кажучи підтвердження паролю. Якщо все введено правильно, то користувач успішно зареєстрований і може використовувати програмний додаток в своїх цілях. Якщо ж ні, потрібно заново ввести дані реєстрації.

Всі паролі повноцінно не зберігаються в системі, база даних має спеціальний збережений хеш, який згенерований за алгоритмом SHA3. При введенні паролю, він перетворюється на хеш пароль, який вже потім порівнюється з хеш-паролем який знаходиться в системі. Якщо паролі пройшли порівняння успішно, то користувач авторизований.

Надалі йому виводяться лише ті дані, які належать лише йому, тобто тільки ті організації, які зв’язані з його ідентифікатором. Користувач може вільно додавати організації, ввівши інформацію про них, додавати засновників, видаляти, та оновлювати інформацію.

## 4.2. Дизайн та вміст вікон

Для показу даних система має графічний інтерфейс користувача, що складається з:

1. З головного вікна на якому є
2. Поля аутентифікації
3. Поля реєстрування користувача
4. Вікно додавання головної сутності
5. Вікно змінення інформації про головну сутність та додаткову сутність
6. Створення нового запису про додаткову сутність
7. Поле закриття бази і програми



Рис. Головного вікна перед авторизації або реєстрації користувача

Для входу потрібно вибрати пункт меню File, а далі вибрати чи відкрити вже існуючу базу даних, чи створити нову базу даних

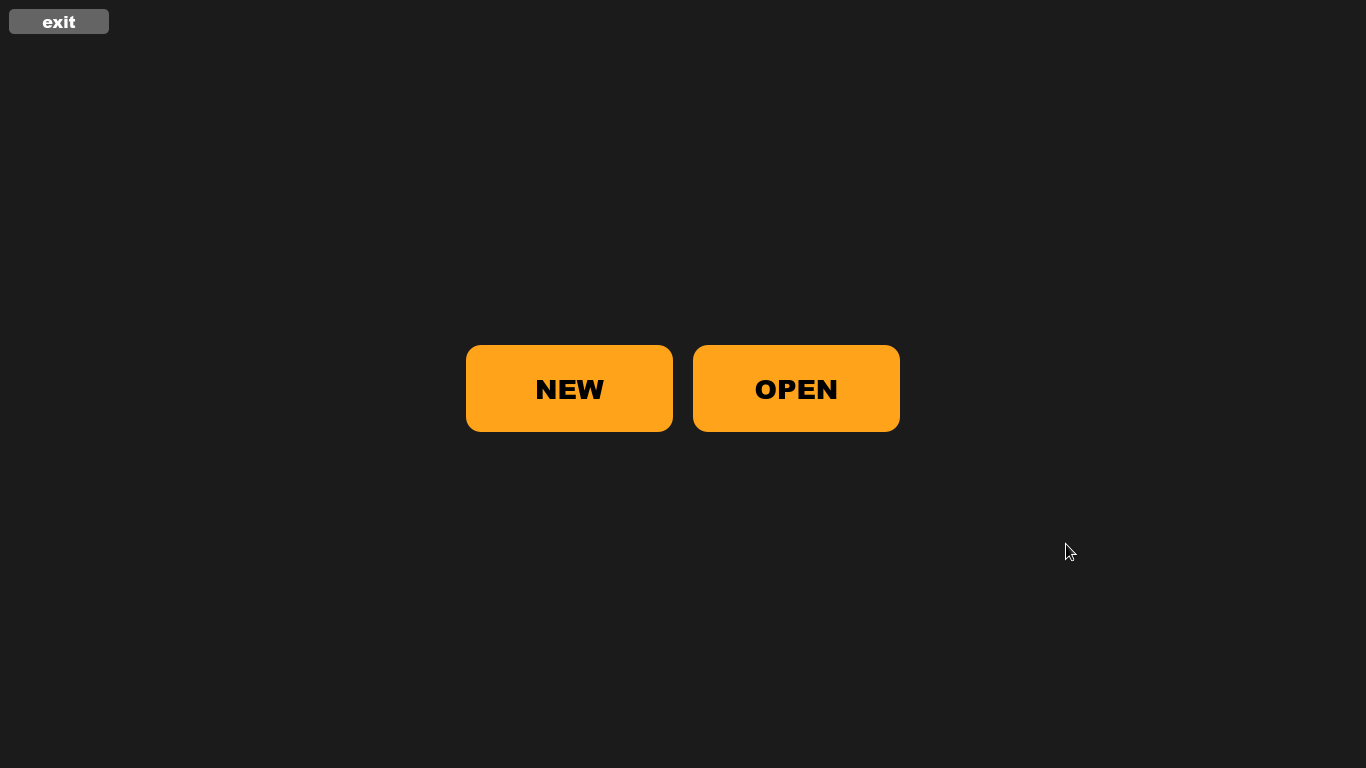
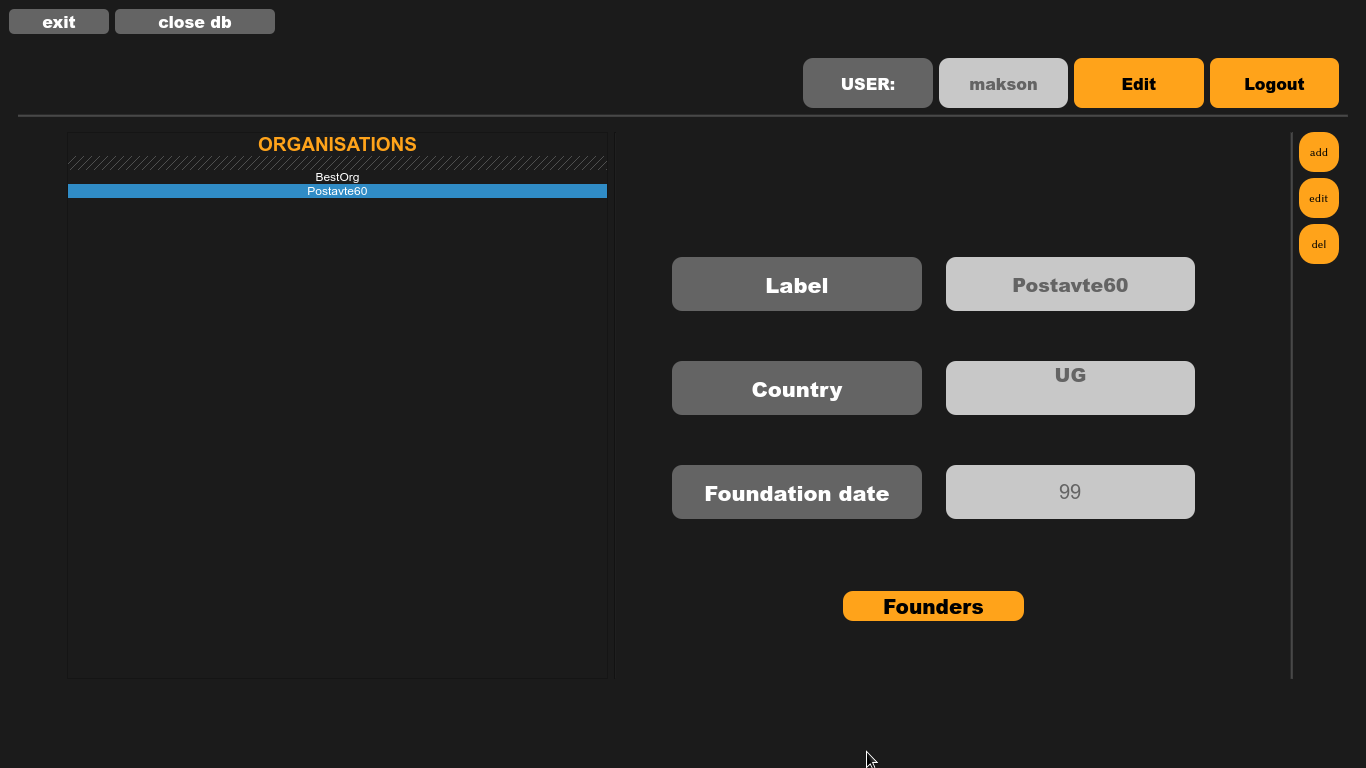


Рис. Вікно авторизації

Правильне введення даних, дозволяє користувачеві увійти в систему.

 Рис. Інтерфейс основної сутності

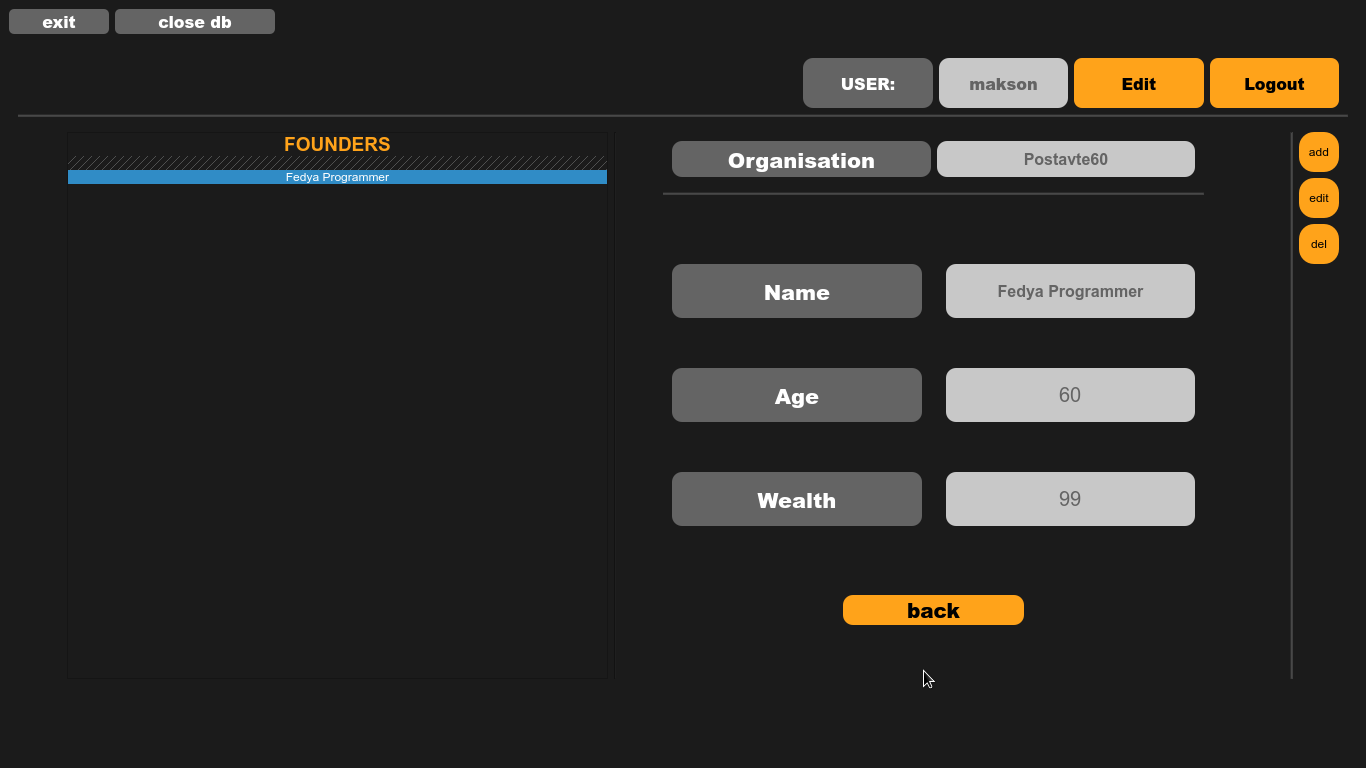
Рис. Інтерфейс дод сутності

Рис. Вікно редагування інформації

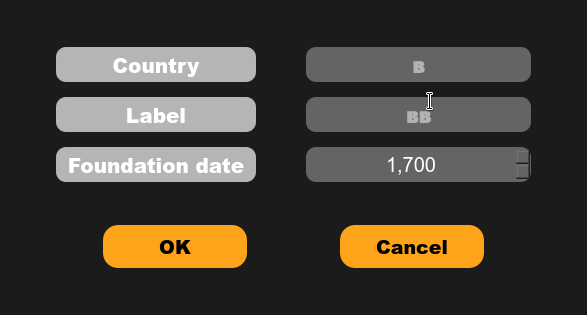


Рис. Вікно додавання

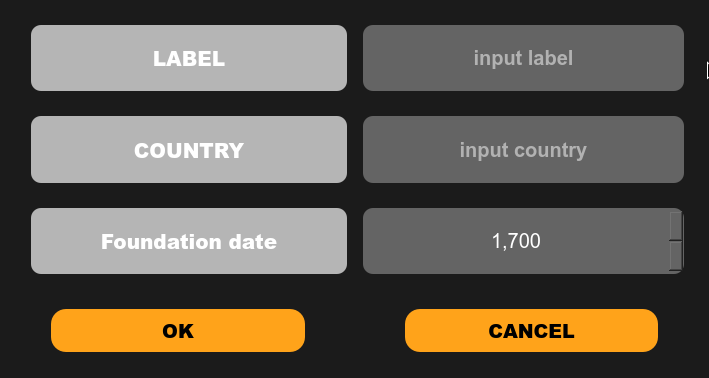


Рис. Вікно редагування інформації

# Висновки

Метою даного курсового проекту було розроблення десктопного додатку з графічним інтерфейсом користувача для керування електронною базою організацій.

Розроблена система:

* забезпечує авторизований доступ до системи;
* дозволяє управляти даними організацій та засновників
* дає можливість змінювати дані, додавати та видаляти певні організації або засновників
* має зрозумілий та зручний інтерфейс користувача.

Особливу увагу під час розроблення даного програмного продукту було приділено взаємодії користувача та програмного додатку, роботі з користувачами та доступом до їхніх даних. А саме, кожен користувач програми має можливість доступитися до даних, створених цією людиною в рамках можливостей графічного додатку.

Редагування цих даних та видалення їх.

Також, при некоректних вводах даних, або будь-якій іншій помилці, программа попереджає користувача, для подальшого виправлення помилок.

Розробка виконана у повному обсязі, всі вимоги враховані, продукт протестовано.

Використання розробленої системи дозволить покращити та спростити роботу у галузях, яким може знадобитися такий функціонал.