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"***

Виконала: Гаценко Марія Андріївна

Група: КП-02

Допущено до захисту

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

2 семестр 2020/2021

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 "\_\_"\_\_\_\_\_\_\_\_\_2р.  з оцінкою\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_/Гадиняк Р.А./ |

**Програмний додаток** **для** **керування електронною соціальної мережею**

Виконавець роботи

Гаценко Марія Андріївна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021р.

Зміст

[Вступ 4](#_Toc73573918)

[1. Аналіз предметної галузі 5](#_Toc73573919)

[1.1. Аналіз вимог до функціональності програмних засобів 5](#_Toc73573920)

[2. Аналіз мов програмування та технологій розроблення 6](#_Toc73573921)

[2.1. Мова програмування С# 6](#_Toc73573922)

[2.2. Фреймворк .NET 6](#_Toc73573923)

[2.3. База даних SQLite 6](#_Toc73573924)

[2.4. Бібліотека Terminal.Gui 6](#_Toc73573925)

[2.5. Бібліотека Microsoft.Data.Sqlite 7](#_Toc73573926)

[2.6. Бібліотека ScottPlot 7](#_Toc73573927)

[3. Опис розроблених програмних засобів 7](#_Toc73573928)

[3.1. Загальна структура системи 7](#_Toc73573929)

[3.2. Структура бази даних 9](#_Toc73573930)

[3.3. Модулі репозиторіїв даних 10](#_Toc73573931)

[3.4. TCP сервер і опис RPC протоколу 12](#_Toc73573932)

[3.5. Модуль експорту 12](#_Toc73573933)

[3.6. Модуль ... 12](#_Toc73573934)

[3.7. Бібліотека … (ваша бібліотека) 12](#_Toc73573935)

[4. Аналіз розроблених програмних засобів 12](#_Toc73573936)

[4.1. Особливості реалізації 12](#_Toc73573937)

[4.2. Інтерфейс користувача 13](#_Toc73573938)

[Висновки 20](#_Toc73573939)

# Вступ

В процесі нашого повсякденного життя часто виникає необхідність дізнатися відповідь на якесь питання, поділитися своїми думками або просто дізнатися щось нове. Інформації у сучасному світі іноді занадто багато, тому ми користуємося певними соціальними мережами для вищенаведених цілей.

Проблемами великої кількості соціальних мереж є складність у керуванні їх даними. Виникає необхідність створення програми, яка б була достатньо ефективною та простою у використанні. Саме тому пропонується розробка програмних засобів для керування електронною соціальної мережею.

Створення системи, що забезпечує доступ до даних, керування ними та надійне збереження їх та їх зв’язків, є актуальною задачею.

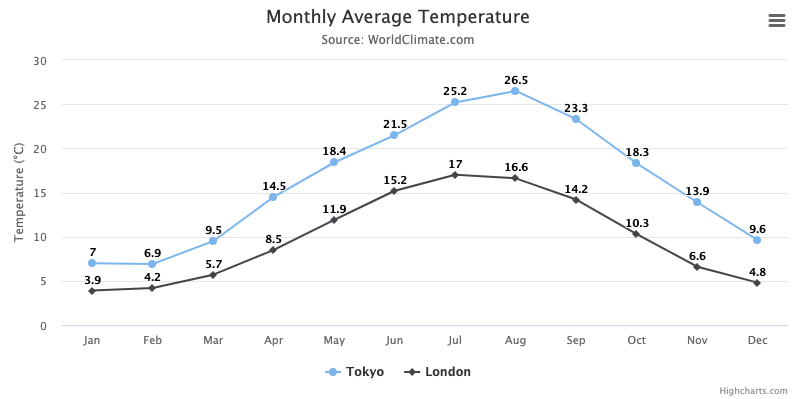
Даний проект присвячено розробленню програмного забезпечення, яке призначено для управління електронними соціальними мережами.

# 1. Аналіз предметної галузі

## 1.1. Аналіз вимог до функціональності програмних засобів

В процесі аналізу вимог до системи було виділено такі функціональні вимоги

1. Програма призначена для створення, керування вмістом бази даних, що містить такі дані:
2. Користувач-пост **one2many**
3. Користувач-коментар **one2many**
4. Пост-коментар **one2many**
5. Програма надає консольний інтерфейс користувача для зручного керування записами в базі даних.
6. Користувач є автором постів і коментарів
7. Користувачі ролі “Модератор” можуть видаляти будь-які питання та відповіді
8. У поста може бути ідентифікатор закріпленого коментаря. Тільки автор поста може закріпити чи відкріпити коментар.
9. Можна відфільтрувати пости по співпадінню тексту і експортувати всі знайдені пости з усіма їх коментарями у форматі XML
10. Можна імпортувати експортовані пости і коментарі.
11. Згенерувати **series** графік із середньою кількістю постів і коментарів протягом заданого періоду часу.



Series графік

# 2. Аналіз мов програмування та технологій розроблення

## 2.1. Мова програмування С#

C# - це високофункціональна об'єктно-орієнтована мова програмування, яка дозволяє зосередитися на використанні вже існуючих можливостей мови, замість того, щоб писати та тестувати все з нуля.

Під час написання цієї роботи була використана велика кількість функцій, методів та класів цієї мови. З найбільш вживаних слід виділити System.Collections.Generic – універсальні колекції, зокрема списки; System.Xml.Serialization – компонент, що допомагає взаємодіяти з типом даних XML; System.IO – для взаємодію з файлами; System.Text – для використання кодування символів через ASCII тощо.

## 2.2. Фреймворк .NET

Платформа .NET дозволяє безпечно виконувати код програми, мінімізувати ймовірність конфліктів в процесі розгортання програмного забезпечення, прибирає проблему продуктивності середовища розробки та ін.

## 2.3. База даних SQLite

База даних SQLite дозволила швидко та зручно керувати таблицями даних. Вона підтримує мову SQL та оброблює запити на цій мові до таблиць даних, повертаючи потрібні значення або помилки.

## 2.4. Бібліотека Terminal.Gui

Бібліотека Terminal.Gui – це бібліотека коду, яка дозволяє створювати псевдографічний інтерфейс. Під час роботи над курсовою роботою, використовувалось багато функцій Terminal.Gui, зокрема: Button – графічна кнопка, яка дозволяє підписуватись на події; Label – мітка з текстом; TextField – поле для вводу тексту; ListView – вікно зі списком сутностей; FrameView – допомагає ініціалізувати екземпляр класу ListView; Dialog – діалогове вікно; MenuBar – рядок меню бару з MenuBarItem та MenuItem, які діють як кнопки тощо.

## 2.5. Бібліотека Microsoft.Data.Sqlite

Ця бібліотека коду дозволяє робити запити до бази даних через мову SQL. У сукупності з SQLite, вона дозволила швидко та зручно працювати з таблицями та даними у них, роблячи запити безпосередньо у самому коді та передаючи їх до SQLite.

Бібліотека Microsoft.Data.Sqlite дозволяє не тільки зручно робити запити на мові SQL, але і безпечно передавати потрібні дані у команди.

Мною були використані запити на мові SQL, підключення до бази даних, виконання запитів до неї тощо.

## 2.6. Бібліотека ScottPlot

Бібліотека ScottPlot – це бібліотека коду, що полегшує інтерактивне відображення великої кількості даних. Вона дозволяє генерувати зображення різних видів графіків, починаючи від лінійних функцій, закінчуючи складним розподіленням даних.

Під час роботи над курсовою роботою, були використані функції створення зображення, найменування вісей, генерація кривої по наперед визначеним точкам та ін.

# 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.

UI client

HTTP Server

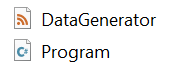
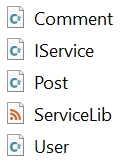
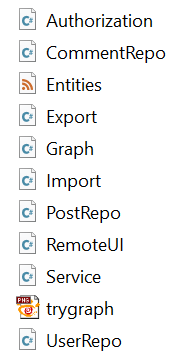
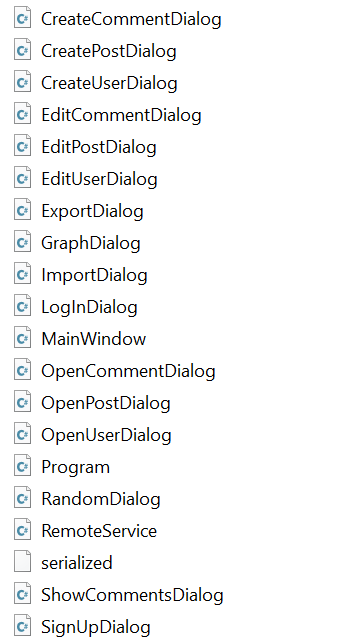
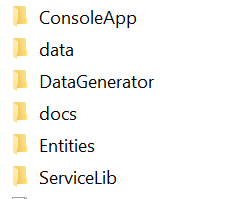
SQLite DB

Рис. Структурна схема системи

Сервер системи обробляє запити від користувачів та виконує такі функції:

1. Запит до бази даних через інтерфейс IService, який передає результат назад до сервера.
2. Серіалізація отриманого результату у XML файл та отримання з нього масиву байт.
3. Передача отриманого масиву клієнту, який зробив запит.
4. Обробка декількох клієнтів одночасно через потоки виконання.

Проект розбитий на такі модулі:



Основні модулі ConsoleApp Entities ServiceLib DataGenerator

Діаграма залежностей модулів IDE:

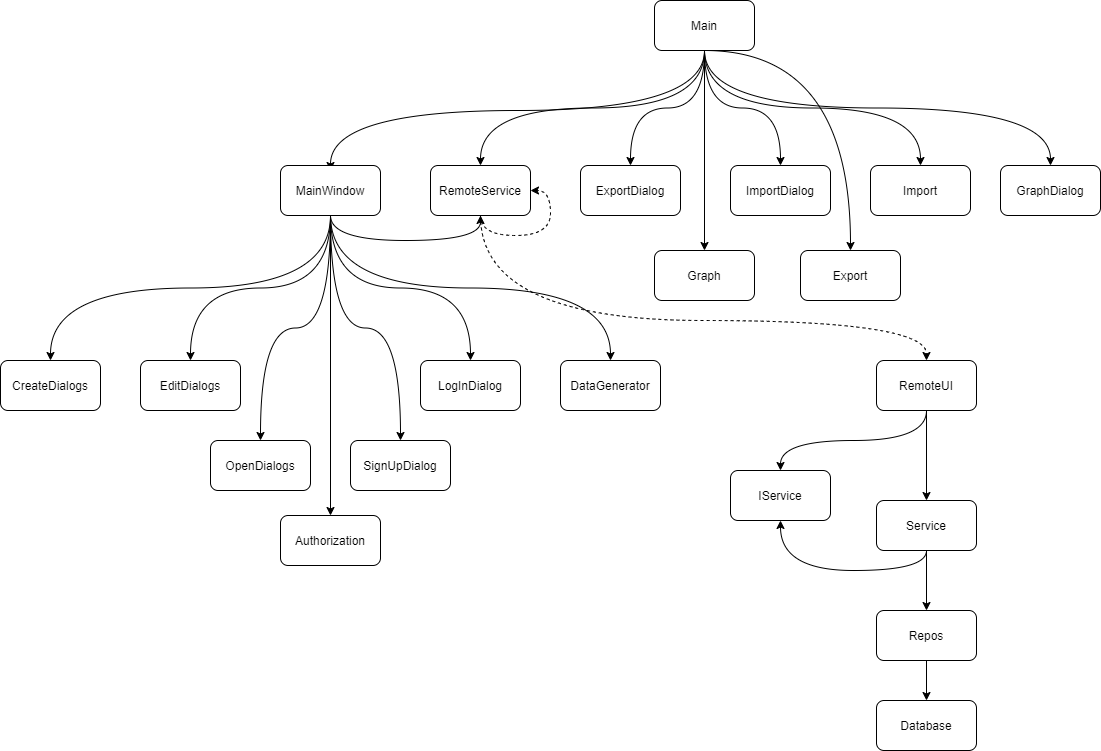


Рисунок 6 Діаграма залежностей

## 3.2. Структура бази даних

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

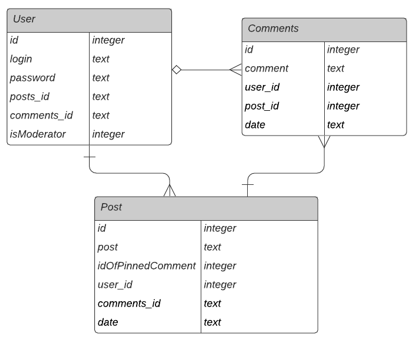


Рисунок 7 Структура бази даних

Таблиця Users:

Має ненормалізовані дані з наступними полями:

* id – унікальний ідентифікатор
* login – унікальний логін
* password – пароль у хеш форматі SHA256
* posts\_id – ідентифікатори постів, авторами яких є користувач
* comments\_id – ідентифікатори коментарів, автором яких є користувач
* isModerator – чи є користувач модератором (0 – ні, 1 – так)

Таблиця Comments:

Має нормалізовані дані з наступними полями:

* id – унікальний ідентифікатор
* comment – текст коментаря
* user\_id – ідентифікатор автора коментаря
* post\_id – ідентифікатор поста, до якого написаний коментар
* date – дата створення коментаря у форматі ISO8601

Таблиця Posts:

Має ненормалізовані дані з наступними полями:

* id – унікальний ідентифікатор поста
* post – текст поста
* user\_id – ідентифікатор автора поста
* comments\_id – ідентифікатори коментарів до поста
* idOfPinnedComment – ідентифікатор закріпленого коментаря
* date – дата створення поста

## 3.3. Модулі репозиторіїв даних

Інтерфейс модуля:

|  |
| --- |
| **UserRepo.сs** |
| private SqliteConnection connection;  private static string filePath;  public static User GetUser(SqliteDataReader reader)  public int DeleteById(int id)  public User GetById(int id)  private bool FindByLogin(string login)  public int Insert(User user)  public int InsertWithId(User user)  public Post[] GetAllPosts(int id)  public Comment[] GetAllComments(int id)  public List<User> GetPage(int pageNumber, int pageLenght)  public List<User> GetSearchPage(string searchValue, int pageNumber, int pageLenght)  public int GetSearchPagesCount(string searchValue, int pageLenght)  public int GetPagesCount(int pageLenght)  public bool Update(int userID, User updatedUser)  public bool UpdateEverythingButPassword(int userID, User updatedUser)  private static string GetHash(HashAlgorithm hashAlgorithm, string input)  public User FindUser(string userName)  public List<User> GetAll() |
| **PostRepo.сs** |
| public PostRepo(SqliteConnection connection, string file)  public static Post GetPost(SqliteDataReader reader)  public Post GetById(int id)  public int DeleteById(int id)  public int Insert(Post post)  public int InsertImported(Post post)  public Comment[] GetAllComments(int id)  public int GetDataForImage(System.DateTime date)  public int GetPagesCount(int pageLenght)  public int GetSearchPagesCount(string searchValue, int pageLenght)  public List<Post> GetPage(int pageNumber, int pageLenght)  public List<Post> GetSearchPage(string searchValue, int pageNumber, int pageLenght)  public bool Delete(int id)  public bool Update(int postId, Post updatedPost)  public List<Post> GetAll()  public int InsertWithId(Post post) |
| **CommentRepo.сs** |
| public CommentRepo(SqliteConnection connection, string file)  public static Comment GetComment(SqliteDataReader reader)  public int DeleteById(int id)  public Comment GetById(int id)  public int Insert(Comment comment)  public int GetDataForImage(System.DateTime date)  public int GetPagesCount(int pageLenght)  public int GetSearchPagesCount(string searchValue, int pageLenght)  public List<Comment> GetPage(int pageNumber, int pageLenght)  public List<Comment> GetSearchPage(string searchValue, int pageNumber, int pageLenght)  public bool Delete(int id)  public bool Update(int commentId, Comment updatedComment)  public List<Comment> GetAll()  public int InsertWithId(Comment comment)  public int GetCommentsForPostPagesCount(int pageLenght, Post post)  public int InsertImported(Comment comment)  public List<Comment> GetCommentsForPostPage(int pageNumber, int pageLenght, Post post)  public List<Comment> GetAllCommentsToThePost(Post post) |

Всі репозиторії підтримують функції додавання, видалення, отримання сутності, отримання кількості сторінок для списку, отримання сутностей для списку.

## 

## 3.4. TCP сервер і опис RPC протоколу

TCP взаємодія відбувається наступним чином: клієнт відправляє запит у особливому форматі (дані запиту розділені між собою знаком долара «$»), сервер зчитує ці дані та відправляє потрібний запит до модуля Service, який вже викликає потрібну функцію певної бази даних. Після отримання результату, йде його серіалізація у формат XML, а масив байт відправляється назад до клієнта. Клієнт зчитує файл XML та десеріалізує результат.

## 3.5. Модуль експорту

|  |
| --- |
| **Export.cs** |
| public static void Serialize(Post[] posts, string filePath)  {  XmlSerializer ser = new XmlSerializer(typeof(Post[]));  System.IO.File.WriteAllText(filePath, "");      System.IO.StreamWriter writer = new System.IO.StreamWriter(filePath);      ser.Serialize(writer, posts);      writer.Close();  } |

## 3.6. Модуль імпорту

|  |
| --- |
| **CommentRepo.сs** |
| public static Post[] Deserialize(string filePath)  {  XmlSerializer ser = new XmlSerializer(typeof(Post[]));  StreamReader reader = new StreamReader(filePath);  Post[] value = (Post[])ser.Deserialize(reader);  reader.Close();  return value;  } |

## 3.7. Бібліотека ServiceLib

У цій бібліотеці знаходяться класи постів, коментарів та користувачі, а також інтерфейс IService, який реалізує Service, який доступається до репозиторіїв.

# 4. Аналіз розроблених програмних засобів

## 4.1. Особливості реалізації

Авторизація користувача відбувається за логіном та паролем. Логін є унікальним ключем, за яким йде пошук у базі даних, а пароль хешується за алгоритмом SHA256 та перевіряється на відповідність запису у базі даних. Якщо хешовані пароль однакові, користувачу надається доступ до основного інтерфейсу.

Під час регістрації відбувається перевірка унікальності логіну та хешування паролю. Якщо логін є унікальним – запис з новим користувачем додається у базу даних, інакше користувачеві повідомляється про те, що логін вже зайнятий.

Користувачі можуть бути модераторами. Модератори можуть видаляти будь-який коментар чи пост, робити інших користувачів модераторами, а також створювати нових користувачів. У звичайних користувачів таких можливостей немає.

Автори поста можуть закріпляти та відкріпляти коментар.

Сервер має можливість обробки багатьох клієнтів.

Якщо сервер не запущений, користувачеві повідомляється це.

## 4.2. Інтерфейс користувача

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

1. Діалогових вікон реєстрації та авторизації.
2. Діалогових вікон створення сутностей всіх типів
3. Діалогових вікон редагування сутностей всіх типів
4. Діалогових вікон підтвердження видалення сутностей всіх типів
5. Діалогових вікон для експорту та імпорту
6. Діалогових вікон вибору файлу
7. Пагінованого списку всіх сутностей
8. Вікон для заповнення бази даних випадково згенерованих сутностей
9. Вікон для перегляду сутностей всіх типів

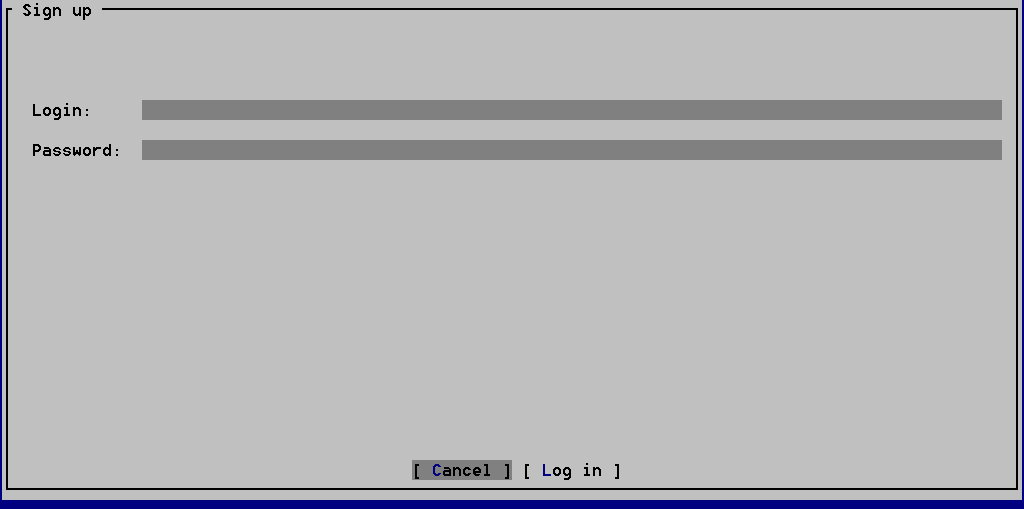


Рисунок 8 Вікно реєстрації



Рисунок 9 Вікно авторизації

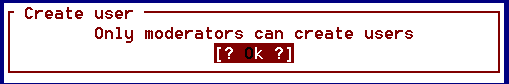


Рисунок 25 Помилки

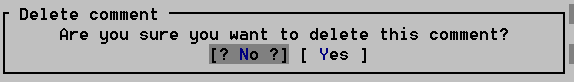
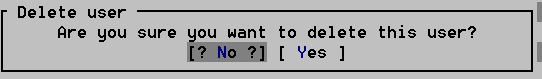
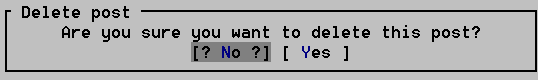


Рисунок 16 Підтвердження видалення сутностей

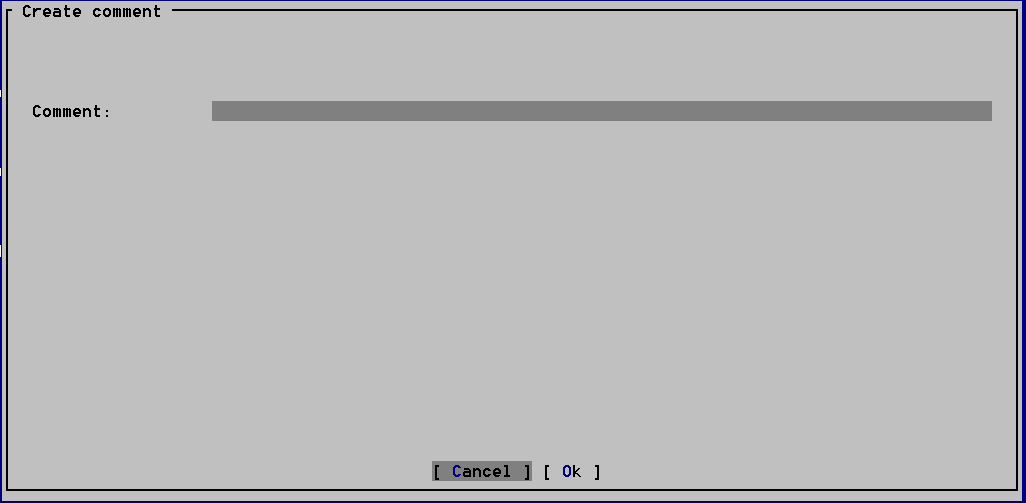


Рисунок 10 Вікно створення коментаря

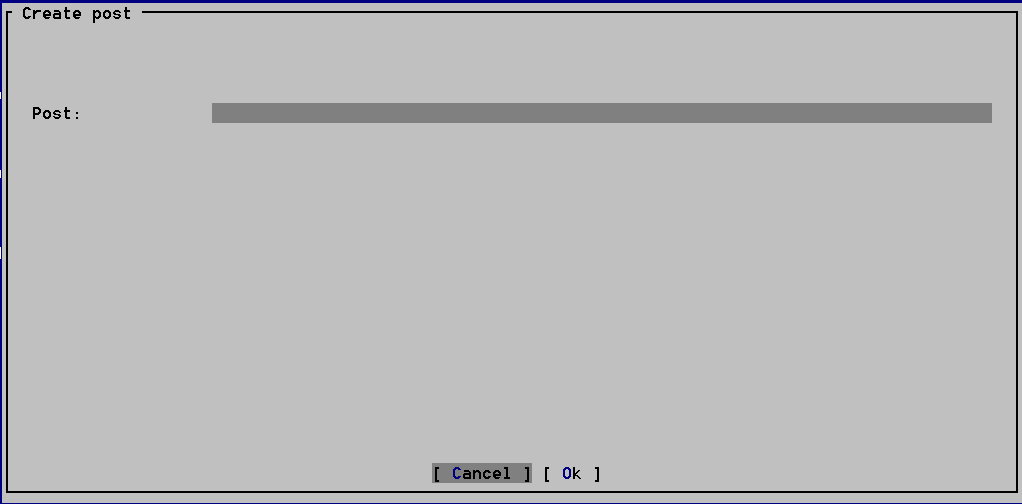


Рисунок 11 Вікно створення поста



Рисунок 12 Пагінований список користувачів

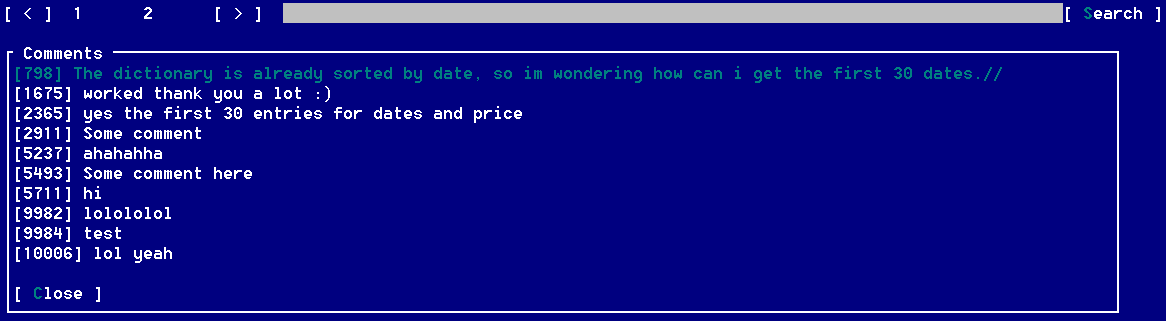


Рисунок 13 Пагінований список коментарів

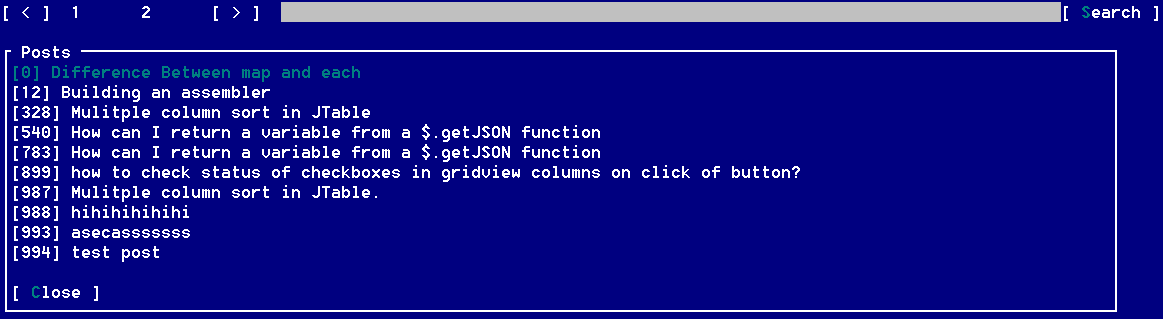


Рисунок 14 Пагінований список постів

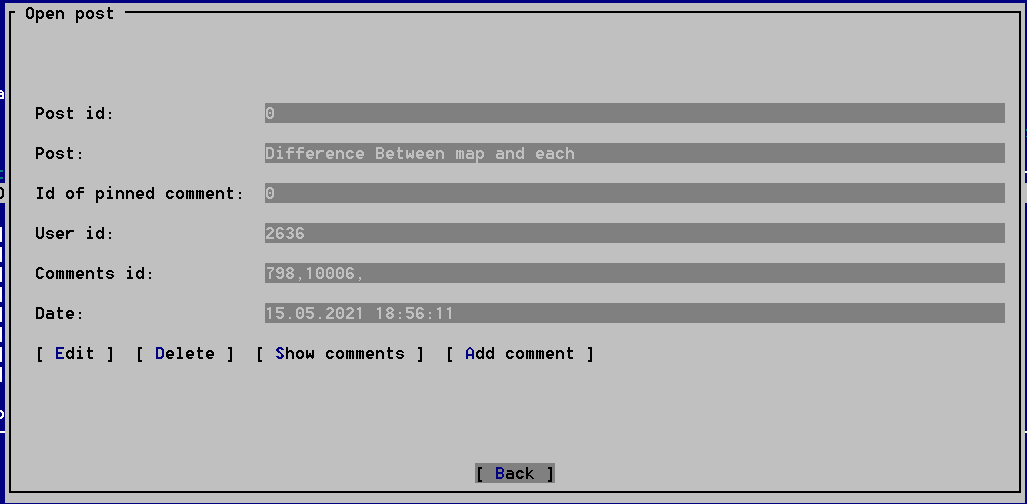


Рисунок 15 Перегляд інформації про пост



Рисунок 17 Перегляд інформації про користувача

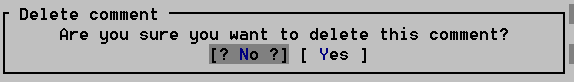
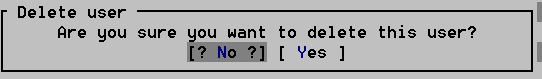
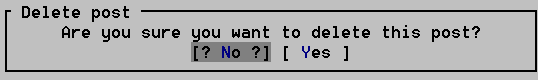


Рисунок 16 Підтвердження видалення сутностей

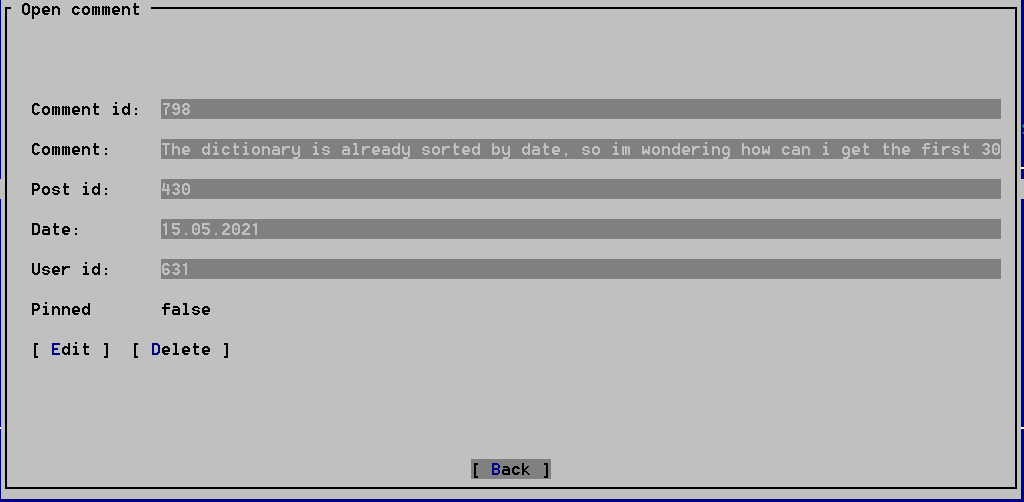


Рисунок 18 Перегляд інформації про коментар

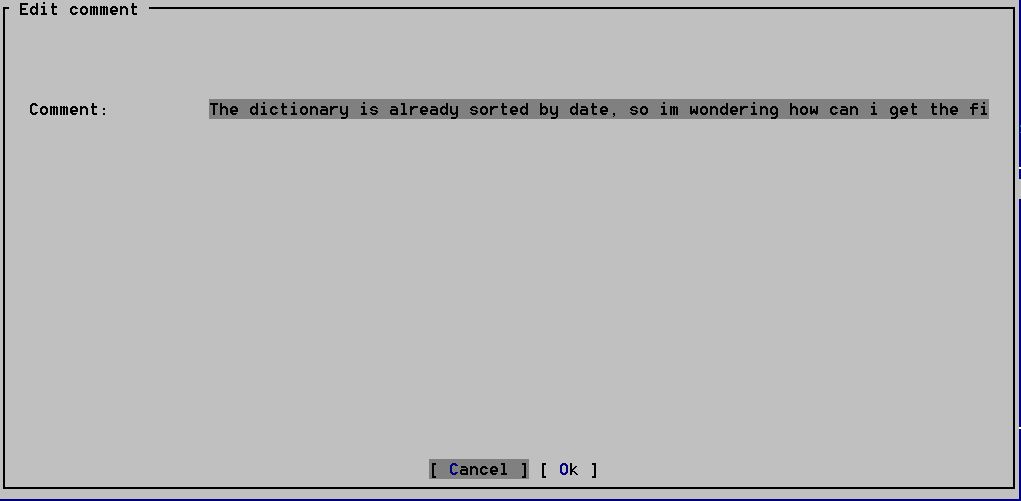


Рисунок 19 Редагування коментаря

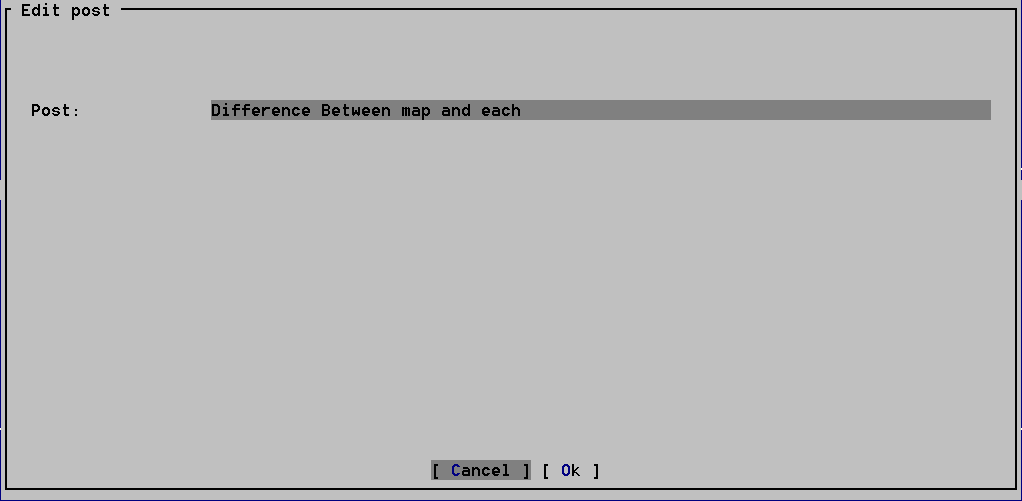


Рисунок 20 Редагування поста

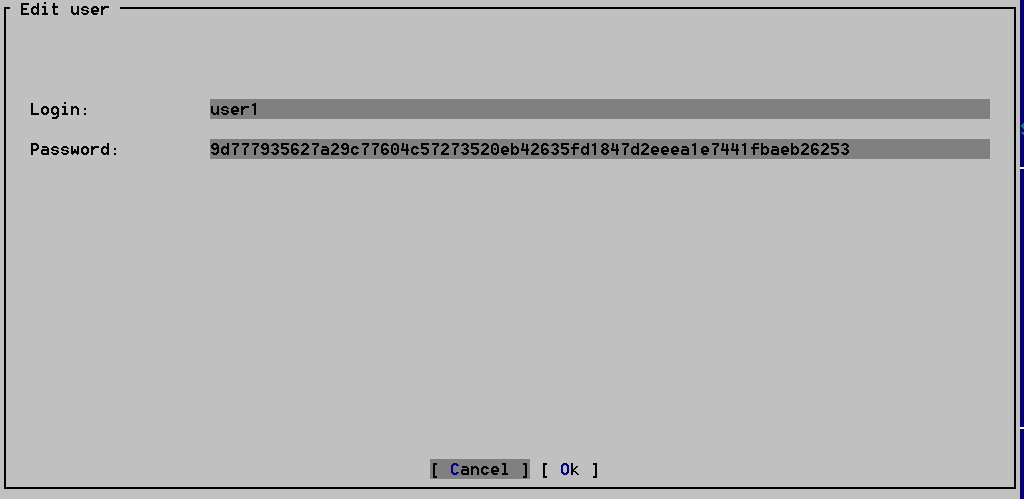


Рисунок 21 Редагування користувача



Рисунок 22 Експорт

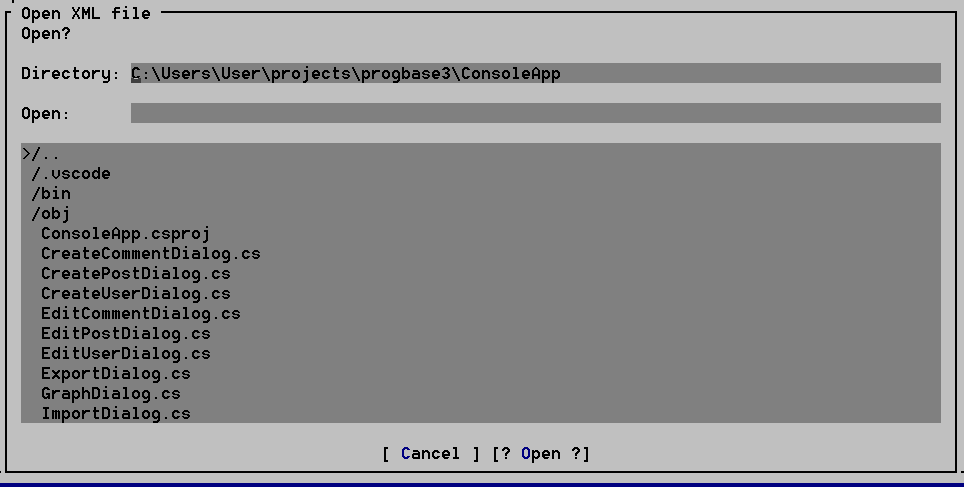


Рисунок 23 Вибір файлу

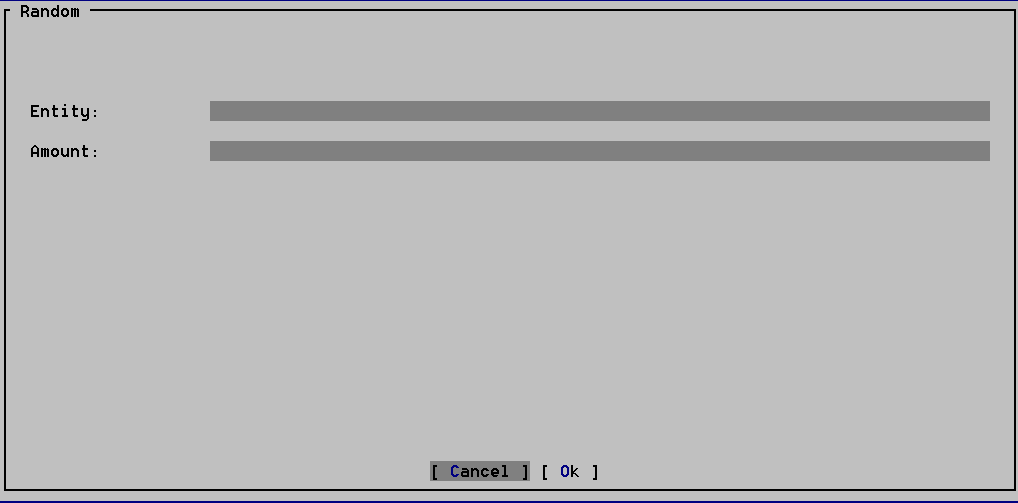


Рисунок 24 Випадкові сутності

# Висновки

Метою даного курсового проекту було розроблення десктопного додатку з подійно-орієнтованим інтерфейсом користувача для керування електронною соціальної мережею.

Розроблена система:

* забезпечує авторизований доступ до бази даних;
* дозволяє додавати та видаляти зв’язки між сутностям, створювати нові сутності, редагувати та видаляти їх;
* дає можливість швидко та зрозуміло працювати з базою даних типу електронної мережі;

Особливу увагу під час розроблення даного програмного продукту було приділено зручності та зрозумілості інтерфейсу та швидкому доступу до бази даних.

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

Використання розробленого додатку дозволить легко та зручно працювати з базою даних типу електронної мережі.