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: Пецеля Артем Володимирович

Група: КП-01

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  "\_\_"\_\_\_\_\_\_\_\_\_2021р.  з оцінкою\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_/Гадиняк Р.А./ |

**Програмний додаток електронної соціальної мережі**

Виконавець роботи

Пецеля Артем Володимирович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021р.

Зміст

[**Вступ**](#_cr2eoqpdh4mm) **4**

[**1. Аналіз предметної галузі**](#_b73pd3816f8p) **5**

[1.1. Аналіз вимог до функціональності програмних засобів](#_qz6vreit6wm5) 5

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

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

[2.2. Фреймворк .NET](#_ayd32pcv0jlc) 7

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

[2.4. Бібліотека Terminal.Gui](#_6jqqk9eelxe) 7

[2.5. Бібліотека ScottPlot](#_xi7o2sr47j2z) 8

[2.6. Бібліотека Microsoft.Data.Sqlite](#_pt5i1swgedmp) 8

[**3. Опис розроблених програмних засобів**](#_kbi4mqs5pjyt) **8**

[3.1. Загальна структура системи](#_nlu4ckytvy2k) 8

[3.2. Структура бази даних](#_ny4p6qq0v8a9) 11

[3.3. Модулі репозиторіїв даних](#_p5yku72ilzdo) 12

[3.4. TCP сервер і опис RPC протоколу](#_ddryso9qa52o) 14

[3.5. Модуль експорту](#_bl0vwivvre05) 14

[3.6. Модуль імпорту](#_bl0vwivvre05) 15

[3.7. Модуль аутентифікації](#_vkb2811r6l9f) 15

[3.8. Модуль генерації звіту](#_94zb7kfdy3jm) 15

[3.9. Бібліотека RPC](#_kevj2vr3z9o8) 15

[3.10. Бібліотека Storage](#_r5dd51utdc1d) 15

[**4. Аналіз розроблених програмних засобів**](#_g0l8gqqu407h) **16**

[4.1. Особливості реалізації](#_5ajysoaxe2s5) 16

[4.2. Інтерфейс користувача](#_c8eyxdz8sp3u) 17

[**Висновки**](#_6dbqg2jsslr5) **23**

# Вступ

В процесі життєдіяльності людей в умовах карантину часто виникає необхідність соціалізації.

Проблемами соціалізації в умовах карантину є неможливість або обмежена можливість спілкування вживу. Виникає необхідність створення електронної соціальної мережі, в якій би люди могли ділитись своїми думками, коментувати ідеї інших, дискутувати на різні теми та спілкуватись незалежно від відстані. Саме тому пропонується розробка програмного додатку електронної соціальної мережі.

Створення системи, що забезпечує соціалізацію людей онлайн в умовах карантину, є актуальною задачею.

Даний проект присвячено розробленню соціальної мережі, який призначено для управління аккаунтами користувачів, постами, що були створені користувачами, та коментарями.

# 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 консольний інтерфейс користувача. Команди використані відповідно до їхнього основного призначення.
   1. Наявно декілька станів інтерфейсу користувача.
   2. Дотримується єдиний стиль для всіх команд
2. Наявні стани для створення і редагування даних.
   1. Можна видаляти дані (з підтвердженням цієї дії).
   2. Наявний стан редагування даних має мати вже заповнені поля із попередніми значеннями. Є можливість відмінити оновлення сутності.
3. Використання списків та таблиць для показу колекцій даних.
   1. Дані в таблиці не є лише відображенням однієї таблиці БД, а містять склеєні чи змінені дані.
   2. Дані у списках та таблицях пагінуються, доступний пошук та фільтрація їх записів.
4. Всі вхідні дані від користувача та з файлів перевіряються, помилки обробляються, користувачу відображається інформація про помилки.
5. Програма має можливість завантаження зображень з файлової системи, їх збереження у сховище даних та їх показ у інтерфейсі користувача.
6. Програма має можливість експорту-імпорту даних у форматі даних XML. Є можливість зберігання даних у файлах, та передача даних до інших процесів через протокол TCP.
7. Програма взаємодіє з реляційною базою даних. SQL запити до бази даних не конкатенуються, а їх виконання винесено у спеціальний окремий модуль (вид репозиторія).
   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/або many-to-many зв'язки
8. Наявна реєстрація користувачів у системі. Паролі користувачів не зберігаються у відкритому виді, а хешуються обраним алгоритмом хешування рядків.
   1. Наявний стан аутентифікації. Пароль користувача при введені прихований. Цей стан має сповіщення про некоректність введених користувачем даних.
   2. Програма авторизує доступу до даних по користувачах. У базі є дані, що належать конкретному користувачу (наприклад, авторство сутностей).
9. Система дозволяє генерувати зображення та файлові звіти на основі даних, що зберігаються у БД.
10. Використання окремих потоків виконання для паралельних фонових дій.
11. Використання клієнт-серверної архітектури програм та TCP сокетів (розділення системи на клієнт і сервер з передачею даних у одному із стандартних форматів даних)
12. Частина проекту винесена у власні бібліотеки класів. Винесення саме цього коду обгрунтоване.

# 2. Аналіз мов програмування та технологій розроблення

## 2.1. Мова програмування С#

Мова C# є об’єктно-орієнтованою мовою. Це означає, що код можна розбивати на логічні структури, використовуючи динамічні та статичні класи, а також власні бібліотеки. В даній роботі використовується версія мови 9.0. Мова є високорівневою, що дозволяє писати код, що буде зрозумілий не тільки машині, а й людині.

## 2.2. Фреймворк .NET

Платформа .NET надає розробнику багато можливостей:

* компіляція коду на багатьох мовах (C#, F#, Visual Basic);
* влаштовані системні бібліотеки (System);
* власний збірник сміття (garbage collector), що видаляє з пам’яті пусті об’єкти;
* включає в себе інтерфейс командного рядка, що дозволяє легко створювати та зв’язувати проекти;

## 2.3. База даних SQLite

Було використано такі можливості бази даних SQLite:

* створення таблиць з полями різних типів
* додавання запису
* редагування запису
* видалення запису
* отримання запису
* отримання декартового добутку таблиць

## 2.4. Бібліотека Terminal.Gui

Дана бібліотека дозволяє розробити графічний інтерфейс користувача для зручності керування даними. Бібліотека надає елементи для вводу/виводу даних у програму, такі як:

* етикетка (Label);
* текстове поле (TextField);
* кнопка (Button);
* радіогрупа (RadioGroup)
* чекбокс (CheckBox)

## 2.5. Бібліотека ScottPlot

Дана бібліотека призначена для генерації графіків на основі масивів даних. В курсовій роботі було використано графік типу “вертикальні смуги” (Vertical Bars).

## 2.6. Бібліотека Microsoft.Data.Sqlite

Дана бібліотека призначена для доступу до бази даних SQLite. Бібліотека дозволяє робити безпечні (захищені від SQL Injection) SQL запити, такі як:

* вставити
* вибрати
* видалити
* оновити
* інші SQL команди

# 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.



Рис. Структурна схема системи

Сервер системи обробляє запити від користувачів та виконує такі функції:

1. Додати користувача / пост / коментар
2. Отримати користувача / пост / коментар за ідентифікатором
3. Редагувати користувача / пост / коментар
4. Видалити користувача / пост / коментар
5. Перевірка на існування користувача за ім’ям
6. Отримання користувача за ім’ям
7. Отримати кількість сторінок пагінованого списку постів / коментарів
8. Отримати список постів / коментарів для сторінки за номером сторінки
9. Отримати коментар за ідентифікатором поста
10. Отримати перший закріплений коментар за ідентифікатором поста
11. Отримати кількість коментарів за ідентифікатором поста та часовим проміжком, в якому коментар був створений

Проект розбитий на такі модулі:

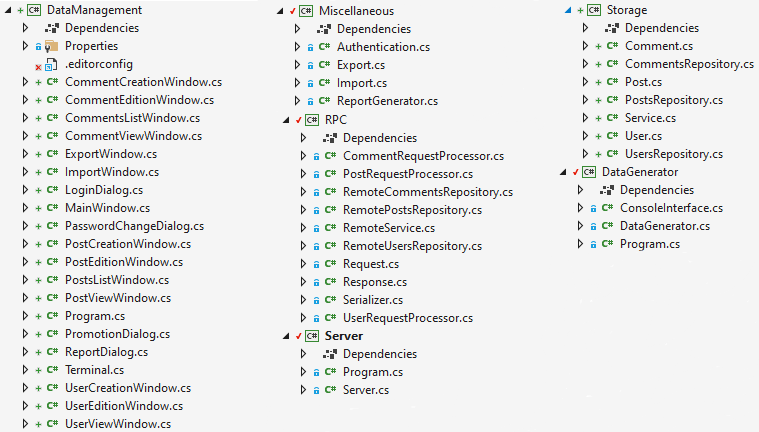
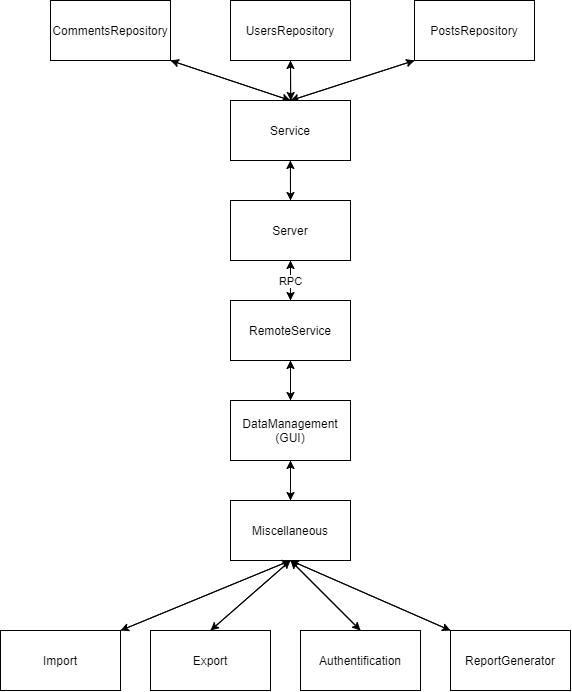


Рис. Модулі проекту

Діаграма залежностей модулів:



## 3.2. Структура бази даних

На рис. представлена діаграма структури бази даних системи.

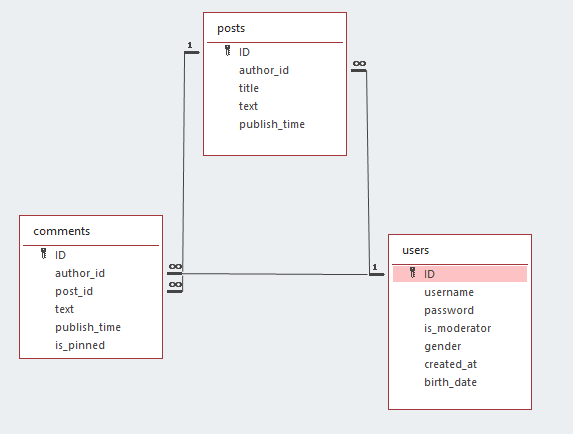


Рис. Структура бази даних

Таблиця users:

* Поле id - унікальний автоінкрементний ідентифікатор
* Поле username - унікальне користувацьке ім’я
* Поле password - пароль, зберігається в захешованому вигляді
* Поле is\_moderator - числове (булеве) поле, для позначення чи має користувач роль модератора
* Поле gender - числове поле, в якому зберігається код гендеру згідно з ISO 5218
* Поле created\_at - текстове поле, в якому зберігається час створення користувача згідно з ISO 8601
* Поле birth\_date - текстове поле, в якому зберігається дата народження користувача згідно з ISO 8601

Таблиця posts:

* Поле id - унікальний автоінкрементний ідентифікатор
* Поле author\_id - ідентифікатор користувача - автора поста
* Поле title - заголовок поста
* Поле text - текст поста
* Поле publish\_time - текстове поле, в якому зберігається дата створення поста згідно з ISO 8601

Таблиця comments:

* Поле id - унікальний автоінкрементний ідентифікатор
* Поле author\_id - ідентифікатор користувача - автора коментаря
* Поле post\_id - ідентифікатор поста, під яким коментар було залишено
* Поле text - текст коментаря
* Поле publish\_time - текстове поле, в якому зберігається дата створення коментаря згідно з ISO 8601
* Поле is\_pinned - числове (булеве) поле, для позначення чи закріплений коментар

## 3.3. Модулі репозиторіїв даних

Інтерфейс модуля:

|  |
| --- |
| **UsersRepository.сs** |
| public class UsersRepository  {  public long Insert(User user) {}  public User GetById(long id) {}  public int Edit(User editedUser) {}  public int DeleteById(long id) {}  public bool UserExists(string username) {}  public User GetByUsername(string username) {}  } |

|  |
| --- |
| **PostsRepository.сs** |
| public class PostsRepository  {  public long Insert(Post post) {}  public Post GetById(long id) {}  public int Edit(Post editedPost) {}  public int DeleteById(long id) {}  public int GetTotalPages(string searchKeyword, long userId) {}  public List<Post> GetPage(int pageNumber, string searchKeyword, long authorId) {}  } |

|  |
| --- |
| **CommentsRepository.сs** |
| public class CommentsRepository  {  public long Insert(Comment comment) {}  public Comment GetById(long id) {}  public int EditById(Comment editedComment) {}  public int DeleteById(long id) {}  public List<Comment> GetByPostId(long postId) {}  public int GetTotalPages(string searchKeyword, long id, bool isAuthor) {}  public List<Comment> GetPage(int pageNumber, string keyword, long id, bool isAuthor) {}  public Comment GetPinnedComment(long postId) {}  public int GetCommentCountBasedOnTimeSpan(long postId, DateTime dateFrom, DateTime dateTo) {}  } |

Модуль UsersRepository виконує такі функції:

* Додавання / видалення / редагування / отримання користувача
* Перевірка на існування користувача за ім’ям
* Отримання користувача за ім’ям

Модуль PostsRepository виконує такі функції:

* Додавання / видалення / редагування / отримання поста
* Отримання загальної кількості сторінок пагінованого списку
* Отримання списку постів для сторінки за номером сторінки

Модуль CommentsRepository виконує такі функції:

* Додавання / видалення / редагування / отримання коментаря
* Отримання списку коментарів за ідентифікатором поста
* Отримання загальної кількості сторінок пагінованого списку
* Отримання списку коментарів для сторінки за номером сторінки
* Отримання закріпленого коментаря за ідентифікатором поста
* Отримання кількості коментарів за ідентифікатором поста та часовим проміжком

## 3.4. TCP сервер і опис RPC протоколу

TCP сервер реалізований на сокетах, що надаються влаштованою бібліотекою System.Net.Sockets. Підключення серверу не виходить за рамки однієї машини, тобто відбувається локально. Віддалений виклик функцій було реалізовано на власній інтерпретації xmlRPC.

## 3.5. Модуль експорту

З бази даних дістається список всіх коментарів обраного поста. Список як структура серіалізується в рядок xml та записується в вказаний користувачем файл.

## 3.6. Модуль імпорту

З вказаного користувачем файлу зчитується рядок. Рядок десеріалізується, у разі невірно заданого рядка, програма критично не зупиняється, повідомляє про помилку. Отримані сутності з рядка записуються в базу даних.

## 3.7. Модуль аутентифікації

Цей модуль реєструє та ідентифікує користувачів. Під час реєстрації перевіряється низка інформації, такі як довжина паролю, унікальність ім’я, вік користувача та інше. Під час ідентифікації перевіряється співпадіння введеного ім’я та паролю з іменем та паролем у базі даних. Паролі зберігаються в хешованому вигляді.

## 3.8. Модуль генерації звіту

Цей модуль на основі введених даних генерує звіт до поста та графік до нього. Генерація docx файлу відбувається з використанням раніше створеного шаблону, шляхом підміни даних та зображення.

## 3.9. Бібліотека RPC

Ця бібліотека містить інтерфейс репозиторіїв, сутності запиту та відповідей, обробник запитів та серіалізатор.

## 3.10. Бібліотека Storage

Ця бібліотека містить основні сутності типу користувач, пост та коментар. Також бібліотека містить класи репозиторіїв.

# 4. Аналіз розроблених програмних засобів

## 4.1. Особливості реалізації

* Авторизація користувачів поділяється на дві частини: реєстрацію та автентифікацію. Реєстрація користувачів неможлива, якщо ім’я нового користувача вже зайнято, довжина паролю менше 6 символів, вік користувача менше 12 років. Ідентифікація можлива лише за умови наявності ім’я користувача в базі даних та співпадінню захешованих паролів.
* Паролі зберігаються в базі даних в захешованому вигляді, що робить базу даних користувачів більш безпечною. Хешування відбувалося за допомогою алгоритма SHA256.
* Користувачам може бути надана роль модератора. Вона дає право користувачам редагувати та видаляти будь-які пости чи коментарі, давати цю роль іншим користувачам.
* TCP взаємодія:

У авторів постів є можливість закріпити коментарі. TCP сервер реалізовано на сокетах, що є влаштованими в стандартну бібліотеку System. Сокети між собою обмінюються байтами, що робить можливим передавати між клієнтом і сервером рядки символів, записані в масив байтів кодуванням UTF-8, що дозволяє передавати не тільки латинські літери, а й кирилицю.

RPC протокол було реалізовано власноруч, без використання сторонніх бібліотек. Клієнт і сервер обмінюються між собою рядками, що представляють собою серіалізовані в xml структури запиту (Request) та відповіді (Response). В структурі запиту є поле для назви методу та поле списку параметрів, що необхідні для виклику функції. В структурі відповіді є поле типу дженеріка, що дозволяє відправляти серіалізовані в xml сутності різного типу.

Обмін повідомленнями відбувається за такою схемою:

1. Клієнт формує запит з назвою методу та параметрами
2. Запит серіалізується в xml рядок, рядок записується в масив байтів та передається на сервер
3. Сервер приймає масив байтів, записує в рядок, рядок десеріалізується в структуру запиту.
4. Запит обробляється, викликається відповідна функція
5. Формується відповідь, результат функції записується у відповідь
6. Відповідь серіалізується в xml рядок, рядок записується в масив байтів та передається до клієнта
7. Клієнт отримує масив байтів, записує в рядок, рядок десеріалізується в структуру відповіді.
8. З структури відповіді клієнт отримує результат функції

* За допомогою влаштованої бібліотеки System.XML відбувається серіалізація/десеріалізація даних.

## 4.2. Інтерфейс користувача

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

1. Головне вікно
2. Діалог ідентифікації
3. Вікно реєстрації
4. Вікно перегляду користувача
5. Вікно редагування користувача
6. Вікно створення посту
7. Вікно перегляду посту
8. Вікно редагування посту
9. Вікно створення коментарю
10. Вікно перегляду коментарю
11. Вікно редагування коментарю
12. Вікно пагінованого списку постів
13. Вікно пагінованого списку коментарів
14. Діалог експорту
15. Діалог імпорту
16. Діалог генерації звіту

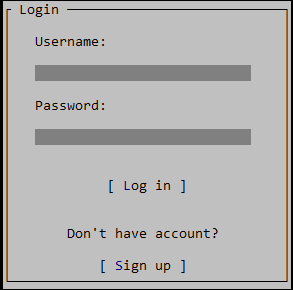


Рис. Діалог ідентифікації

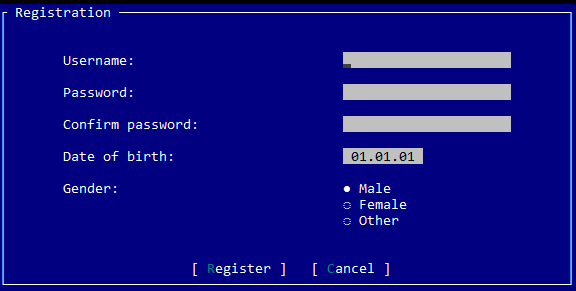


Рис. Вікно реєстрації

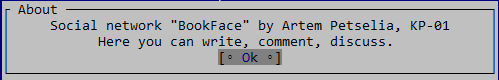


Рис. Діалог “Про програму”

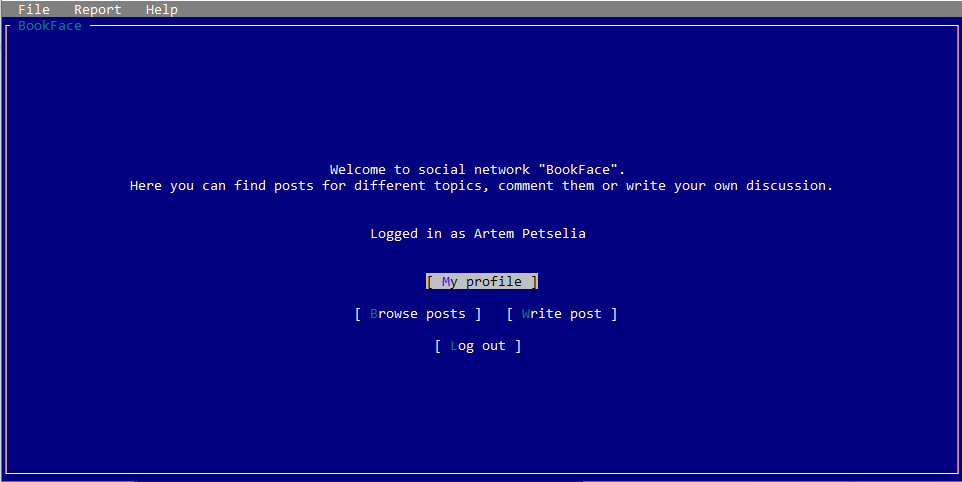


Рис. Головне вікно

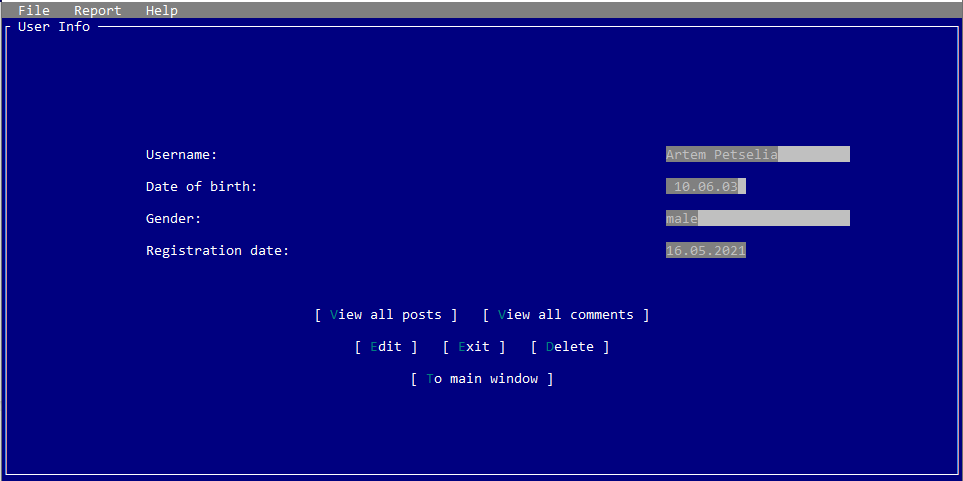


Рис. Вікно перегляду користувача



Рис. Вікно редагування користувача

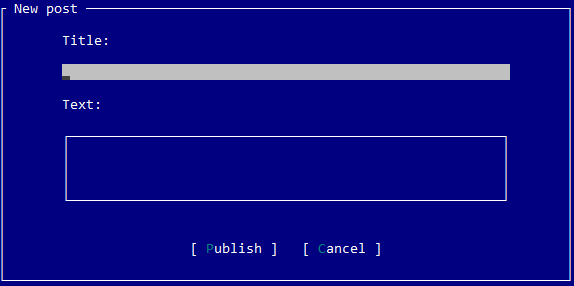


Рис. Вікно створення посту

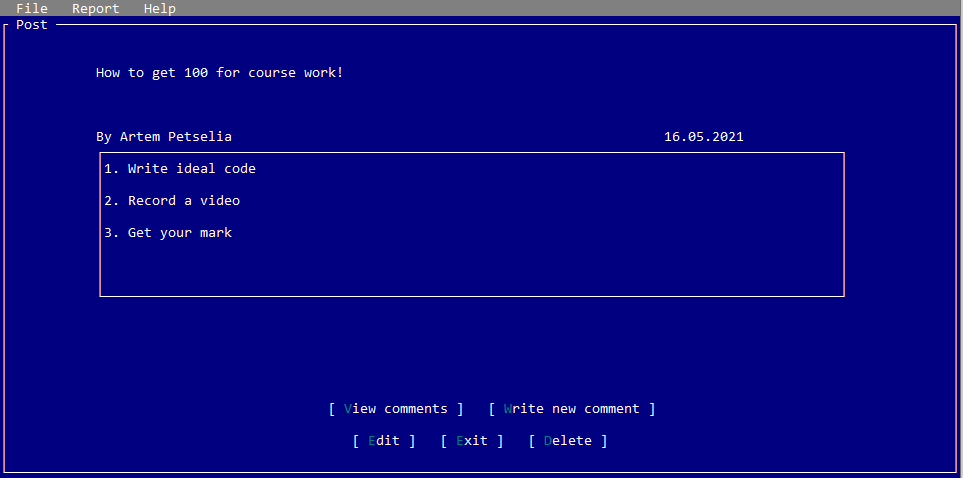


Рис. Вікно перегляду посту

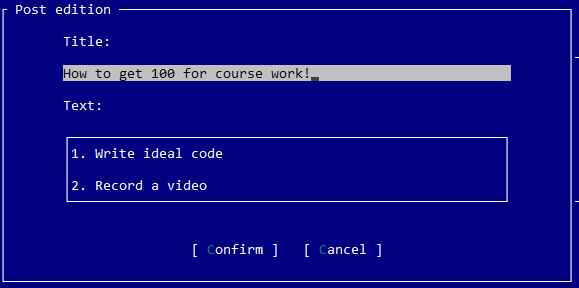


Рис. Вікно редагування посту

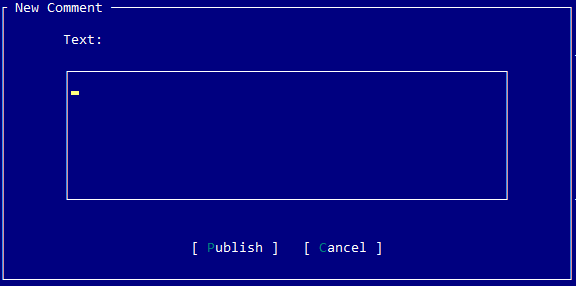


Рис. Вікно створення коментарю

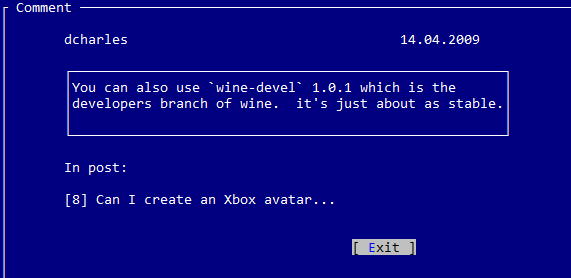


Рис. Вікно перегляду коментарю

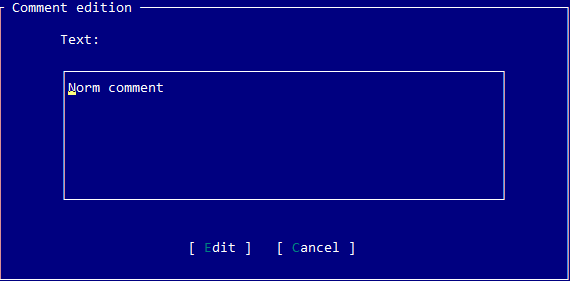


Рис. Вікно редагування коментарю

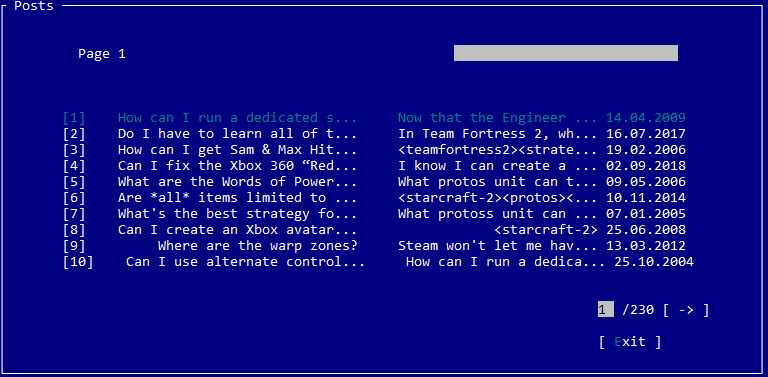


Рис. Вікно пагінованого списку постів

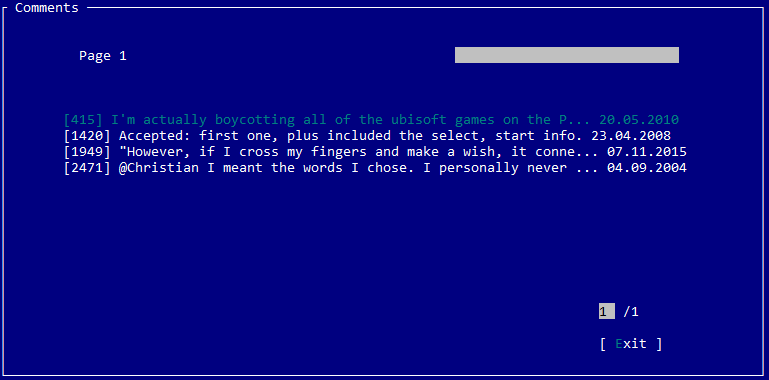


Рис. Вікно пагінованого списку коментарів

# Висновки

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

Розроблена система:

* забезпечує авторизований доступ до системи;
* дозволяє керувати постами та коментарями користувачам - авторам та користувачам - модераторам;
* дає можливість переглядати інформацію про користувачів, пости та коментарі;

Особливу увагу під час розроблення даного програмного продукту було приділено графічному інтерфейсу користувача, безпеці доступу до бази даних та користувацької інформації, написанню власного протоколу віддаленого виклику процедур, уникненню критичного завершення роботи програми.

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

Використання розробленого додатку дозволить користувачам проводити час спілкуючись онлайн.