

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

## Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ МЕСЕНДЖЕРА З

БОТОМ-КОМПАНЬЙОНОМ

SOFTWARE IMPLEMENTATION OF A MESSENGER WITH

A COMPANION BOT

Виконала: студентка 4 курсу, групи КНТз-110\_2

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЮРАСОВА А.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ПОЛЯКОВ М. О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ТИМЕНКО А.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Факультет ФКНТ  
Кафедра програмних засобів  
Ступінь вищої освіти бакалавр  
Спеціальність 121 Інженерія програмного забезпечення  
(код і найменування)  
Освітня програма  
(спеціалізація) Інженерія програмного забезпечення  
(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**  
Завідувач кафедри ПЗ, д.т.н, проф.  
Сергій СУББОТІН  
«    »                      2023 року

**З А В Д А Н Н Я**  
**НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)**

ЮРАСОВА Анастасія Володимирівна  
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація месенджера з ботом-компаньйоном. Software Implementation of a Messenger with a Companion Bot

керівник проєкту (роботи) д.т.н, проф. Поляков Михайло Олексійович  
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «04» квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 1 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми.

3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми. 5. Текст програми

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

## 6. Консультанти розділів проєкту (роботи)

| Розділ          | ПРИЗВИЩЕ, ініціали та посада консультанта | Підпис, дата   |                           |
|-----------------|-------------------------------------------|----------------|---------------------------|
|                 |                                           | завдання видав | прийняв виконане завдання |
| Основна частина | Поляков М.О., професор                    |                |                           |
| Нормоконтроль   | Дейнега Л.Ю., ст. викладач                |                |                           |

7. Дата видачі завдання « 04 » квітня 2023 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проєкту (роботи)               | Строк виконання етапів проєкту (роботи) | Примітка     |
|-------|--------------------------------------------------------|-----------------------------------------|--------------|
| 1     | Постановка завдання роботи.                            | 1 тиждень                               | Завдання, ТЗ |
| 2     | Аналіз предметної області.                             | 2 тиждень                               | Розділ 1     |
| 3     | Вибір мови програмування та інших технологій.          | 3 тиждень                               | Розділ 2     |
| 4     | Розробка архітектури програми.                         | 3 тиждень                               | Розділ 2     |
| 5     | Розробка програми.                                     | 4–5 тижні                               | Розділ 3     |
| 6     | Тестування та експериментальне дослідження програми.   | 6 тиждень                               | Розділ 4     |
| 7     | Оформлення пояснювальної записки та документів до неї. | 7 тиждень                               | Додатки      |
| 8     | Нормоконтроль та рецензування.                         | 8 тиждень                               |              |
| 9     | Захист роботи.                                         | 9 тиждень                               |              |
|       |                                                        |                                         |              |
|       |                                                        |                                         |              |
|       |                                                        |                                         |              |
|       |                                                        |                                         |              |
|       |                                                        |                                         |              |
|       |                                                        |                                         |              |

Студент(ка)

\_\_\_\_\_ Анастасія ЮРАСОВА  
( підпис ) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

\_\_\_\_\_ Михайло ПОЛЯКОВ  
( підпис ) (Ім'я ПРИЗВИЩЕ)

## РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра:  
148 с., 2 табл., 53 рис., 2 дод., 22 джерел.

C#, MICROSOFT .NET FRAMEWORK, БАЗА ДАНИХ, БОТ, КЛІЄНТ-СЕРВЕРНИЙ ЗАСТОСУНОК, МЕСЕНДЖЕР, МОДЕЛЬ OSI.

Об'єкт дослідження – процес функціонування месенджерів та способи їх поєднання з ботом, який імітує віртуального улюбленця-компаньйона.

Предмет дослідження – програмна реалізація месенджера з ботом-компаньйоном.

Мета роботи – підвищення рівня діалогу месенджера з користувачем.

Матеріали, методи та технічні засоби: модульне, компонентно-орієнтоване та об'єктно-орієнтоване програмування, мова програмування C#, Microsoft .NET Framework, середовище розробки Visual Studio 2019, персональний комп'ютер з процесором Intel Core 5 під управлінням операційної системи Microsoft Windows 10.

Результати. Створено месенджер з інтерактивним улюбленцем ботом-компаньйоном.

Висновки. Розроблений месенджер містить інтерактивного улюбленця бота-компаньйона, який має власну історію, систему рейтингу відносин із користувачем, час від часу звертається до нього з питаннями та пропонує разом випити кави.

Галузь використання – комунікації та розваг.

## ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 148 pages, 2 tables, 53 figures, 2 appendixes, 22 sources.

DATABASE, BOT, CLIENT-SERVER APPLICATION, C#, MICROSOFT .NET FRAMEWORK, MESSENGER, OSI MODEL.

Object of study is the process of functioning of messengers and methods of their combination with a bot which imitates a virtual pet-companion.

Subject of study is the software implementation of a messenger with a bot-companion.

The purpose of the work is the is to increase the level of dialogue between the messenger and the user.

Materials, methods, and tools: modular, component-oriented and object-oriented programming, C# programming language, Microsoft .NET Framework, integrated development environments Visual Studio 2019, personal computer with an Intel Core 5 processor under the control of the Microsoft Windows 10 operating system.

Results. A messenger with an interactive pet companion bot has been created.

Conclusions. The developed messenger contains an interactive pet companion bot, which has own story, rating system of relationship with the user, from time to time asks him questions and offers to drink coffee together.

The field of use is communications and entertainment.

## ЗМІСТ

|                                                                           |    |
|---------------------------------------------------------------------------|----|
|                                                                           | С. |
| Перелік скорочень та умовних позначок.....                                | 8  |
| Вступ.....                                                                | 9  |
| 1 Аналіз предметної області.....                                          | 12 |
| 1.1 Аналіз існуючих месенджерів.....                                      | 13 |
| 1.2 Аналіз існуючих ботів у месенджерах.....                              | 14 |
| 1.3 Висновки до розділу 1.....                                            | 15 |
| 2 Розробка архітектури програми.....                                      | 16 |
| 2.1 Вибір мови програмування.....                                         | 16 |
| 2.2 Аналіз функціональних вимог.....                                      | 20 |
| 2.3 Проектування архітектури бази даних.....                              | 22 |
| 2.4 Проектування архітектури серверу.....                                 | 24 |
| 2.5 Проектування архітектури графічного інтерфейсу користувача.....       | 26 |
| 2.6 Проектування боту-компаньйону.....                                    | 27 |
| 2.7 Висновки до розділу 2.....                                            | 28 |
| 3 Основні рішення щодо реалізації компонентів системи.....                | 29 |
| 3.1 База даних.....                                                       | 29 |
| 3.2 Серверна частина.....                                                 | 33 |
| 3.2.1 DLL-бібліотека для роботи з базою даних.....                        | 34 |
| 3.2.2 Модулі служб.....                                                   | 36 |
| 3.2.3 Хост.....                                                           | 39 |
| 3.3 Частина клієнта.....                                                  | 40 |
| 3.3.1 Графічний інтерфейс користувача.....                                | 40 |
| 3.3.2 Бот-компаньйон.....                                                 | 43 |
| 3.4 Висновки до розділу 3.....                                            | 46 |
| 4 Експлуатація, тестування та експериментальне дослідження програми.....  | 47 |
| 4.1 Експлуатація, тестування та експериментальне дослідження хосту.....   | 47 |
| 4.2 Експлуатація, тестування та експериментальне дослідження клієнту..... | 48 |
| 4.3 Висновки до розділу 4.....                                            | 60 |

|                                   |     |
|-----------------------------------|-----|
| Висновки.....                     | 61  |
| Перелік джерел посилань.....      | 63  |
| Додаток А Текст програми.....     | 65  |
| Додаток Б Слайди презентації..... | 140 |

## **ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК**

- DLL – Dynamic Link Library;  
OSI – The Open Systems Interconnection model;  
WCF – Windows Communication Foundation;  
WPF – Windows Presentation Foundation;  
БД – база даних;  
ПЗ – програмне забезпечення;  
СУБД – система управління базами даних

## ВСТУП

Передача повідомлень між людьми, які знаходяться на відстані між одним з одним, походить своїми коріннями у далеке минуле. Люди з усього світу намагалися передати ту чи іншу інформацію різними методами, що свідчить про важливість дистанційної комунікації для людства.

Далеким праобразом сучасних застосунків для передачі повідомлень є система пошти, що дозволяла спілкуватися людям, що знаходяться у різних містах або країнах.

Винахід телеграфу зробив можливим доставку коротких повідомлень на великі відстані за короткий проміжок часу.

Поява всесвітнього павутиння Інтернет давала змогу трансляції повідомлень на широку аудиторію людей зі всього світу. На відміну від поштової системи, ці повідомлення доставлялися у хвилини чи секунди. А на відміну від телеграфу, вона робила можливим доставку великих повідомлень до будь-якої людини, яка використовує персональний комп'ютер. Згодом це призвело до появи віртуальної пошти, а ще пізніше – спеціалізованих програм обміну швидкими повідомленнями між людьми з будь-якої частини світу. Такі програми мають назву месенджер або, англійською, messenger – у перекладі: вісник, глашатай, посильний, кур'єр, той, хто розносить повідомлення [1].

Головною задачею месенджеру було тримати на зв'язку знайомих людей або знаходити нових співрозмовників. Проте з появою нових месенджерів-застосунків до них додавалися нові функції. Наприклад, обмін смайлами – графічними відображеннями емоцій, можливість передачі файлів та безкоштовні дзвінки.

Однак, феномен глобалізації не зміг повністю задовольнити потреби людства у спілкуванні. Про це свідчить поява нових месенджерів та популяризація їх у суспільстві. Нові застосунки не тільки отримують свою аудиторію, а й часто становляться більш популярними за попередні реалізації програм обміну миттєвих повідомлень.

З початком епідемії коронавірусу та переважно віддаленої роботи, зростає потреба людей у спілкуванні в віртуальному вимірі.

Проблематикою впливу віртуального виміру на людину займалося багато дослідників, тому це явище має багато ракурсів осмислення. Наприклад, дослідник Жан Бодрийяр дає переважно негативну оцінку. На його думку, віртуальна реальність – це лише імітація, яка спотворює уявлення про реальність. До того ж, людина стає беззахисною перед «розумними машинами», адже «машина робить лише те, що від неї потребує людина, проте людина також виконує те, на що запрограмована машина» [2, с.83].

Протилежної точки зору дотримувався дослідник Майкл Хайм, на думку якого, віртуальна реальність стимулює у людини потяг до творчості та розвиває уяву. До того ж, вона стимулює такий спектр органів відчуття, що відокремити штучну реальність від справжньої майже неможливо [3].

Ідея існування віртуальних вихованців належить розробникам гри «Тамагочі», популярній у 90-ті роки минулого століття та на початку нульових років нинішнього.

Сенс гри полягає в інтерактивному спостереженні за життям вихованця, від його народження і до смерті. Вихованця можна було годувати, грати з ним, стежити за здоров'ям, прибирати за ним, тобто електронна іграшка нагадувала взаємодію зі справжньою істотою.

Тобто, віртуальний вихованець є маленькою програмою, яка виконувала задані повторювані завдання (найчастіше за заданим розкладом). Такі програми мають назву ботів (скорочено від «робот»).

Сучасні роботи дедалі більше нагадують справжніх істот. Їх функції вже не обмежуються діями у часі, вони здатні до самонавчання [4].

Проте у сучасних месенджерах функції ботів дуже обмежені, вони не виступають повноцінними супутниками користувача, а лише виконують вузькі функції, наприклад, надають відповіді на найчастіші питання або збирають статистику.

Об'єктом дослідження є процес функціонування месенджерів та способи їх поєднання з ботом, який імітує віртуального улюбленця-компаньйона.

Предметом дослідження є програмна реалізація месенджеру з ботом-компаньйоном.

Наукова новизна роботи полягає у дослідженні та створенні месенджеру з віртуальною твариною-компаньйоном, який би збільшив задоволення людини від спілкування у тотожних застосунках. На відміну від звичайних чат-ботів, реалізація боту-компаньйону передбачає створення власної історії компаньйону, логіки його взаємодії з користувачем та начислення балів за відносини з ним.

Метою роботи є підвищення рівня діалогу месенджеру з користувачем.

Відповідно до мети були поставлені такі завдання:

- провести аналіз предметної області та існуючих аналогів;
- здійснити проектування месенджеру та інтерактивного боту-компаньйону;
- реалізувати спроектовану модель месенджеру з інтерактивним ботом-компаньйоном;
- виконати тестування розробленого месенджеру та бота-компаньйона.

## 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Програмне забезпечення для передачі даних між двома або більше користувачами реалізується завдяки використанню мережевої взаємодії, де є пара клієнт та сервер.

Клієнт – це модуль, призначений для формування та передачі повідомлень-запитів до ресурсів віддаленого комп'ютера від різних додатків з наступним прийомом результатів з мережі та передачею їх відповідним додаткам [5, с.46].

Сервер – це модуль, який постійно чекає на запит з мережі від клієнтів і, прийняв запит, намагається його обслужити. Один сервер може обслуговувати запити одразу кількох клієнтів (по черзі або одночасно) [5, с.46].

Таким чином, месенджери відносяться до клієнт-серверних застосунків. Це означає, що передача даних між користувачами проходить за одним із існуючих мережевих протоколів взаємодії.

Протоколи взаємодії визначені у моделі OSI, яка описує різні рівні взаємодії систем. Відповідно до неї, всього існує сім рівнів: фізичний (представляє передачу бітів по фізичному обладнанню), канальний (визначає фізичну адресацію), мережевий (визначає маршрут та логічну адресацію), транспортний (уявляє прямий зв'язок між кінцевими призначеннями та відповідає за надійність), сеансів (управляє сеансами), представлення (відповідає за представлення та шифрування даних) та прикладний рівень (відповідає за доступ до мережевих служб). Кожен рівень постачає свій окремий заголовок, який додається до даних, завдяки чому проходить взаємодія у мережі [6].

Сучасні месенджери мають зазвичай дві версії: мобільний та десктопний застосунок. Перші, як правило, використовують протокол HTTP, що належить до прикладного рівня. Другі зазвичай використовують протокол TCP/IP, який належить до транспортного рівню моделі OSI.

Однак, згідно моделі OSI, протокол HTTP, що належить до прикладного рівня, не має захищеного каналу, який би блокував потенційне викрадення даних злоумисниками. Цю проблему частково вирішує протокол HTTPS, який є поєднанням протоколів HTTP та SSL, що належить до рівню представлення. На відміну від цього, протоколи транспортного рівня, такі як TCP та UDP, які взаємодіють із сокетом, тобто, з'єднання йде за допомогою IP-адреси та порту на пристрої, додатково шифрують дані, які потрапляють у мережу, адже вони передаються як байти [5]-[8].

### **1.1 Аналіз існуючих месенджерів**

Ідея обміну миттєвими повідомленнями між людьми належить компанії Mirabilis, яка випустила програмне забезпечення ICQ у 1996 році.

ICQ (від англ. I seek you – «я шукаю тебе») був застосунком для персональних комп'ютерів, який через обмежений трафік мав змогу лише передавати повідомлення, не використовуючи картинок або передачі файлів. Окрім цього, історія повідомлень зберігалася не централізовано, а на пристрою клієнтів у текстовому файлі, що виключало надійність зберігання даних. Надалі застосунок постійно вдосконалювався, додаючи новий функціонал. На теперішній час воно має голосовий і відео зв'язок, а також підтримку передачі файлів методом Peer-to-peer (тобто в обхід серверу).

На зміну популярності ICQ прийшов застосунок Skype. Його нововведення полягало в тому, що воно спочатку орієнтувалося на підтримку голосових дзвінків. За своєю архітектурою, це була мережа без єдиного сервера (сервером виступали клієнтські персональні комп'ютери), а сховище сервера містило лише паролі і списки контактів користувачів. Як і його попередник, Skype використовував протокол TCP/IP.

Недоліком застосунку було використання портів 80 та 443, що є стандартними для локального хосту, що підтримувалися серверами apache, IIS, а також часто використовуються операційною системою за її потребами.

До додаткових функцій сучасного застосунку Skype відносяться: відео зв'язок, демонстрація екрану, мобільна версія з можливістю використання браузеру для входу в систему.

Viber став месенджером, який проектувався для використання мобільними пристроями. Він має свою особливість – можливість платних дзвінків на мобільні номери телефонів. Також він має змогу передачі фалів та аудіо дзвінків для користувачів застосунку.

Месенджер Telegram вважається дуже надійним з точки зору безпеки передачі даних, адже має власну систему шифрування, а його сервери зберігаються в різних країнах світу. Це зробило його популярним вже на першому етапі його запуску, коли його функціонал був обмежений виключно передачею повідомлень.

Сучасна версія Telegram має функцію передачі файлів, аудіо та відео дзвінків, демонстрації екрану.

До менш популярних месенджерів належать WhatsApp та Signal.

## **1.2 Аналіз існуючих ботів у месенджерах**

Окрім своєї власної системи шифрування повідомлень, Telegram перший став використовувати чат боти у месенджерах. Пізніше цю тенденцію перейняли інші месенджери. Однак, найчастіше їх використовують у маркетингових цілях масової розсилки.

Чат бот – це програмне забезпечення, що працює всередині месенджера. Така програма здатна відповідати на запитання, а також самостійно ставити їх. Чатботи використовуються у різних сферах для вирішення типових завдань.

Сучасні чат боти спрямовані на скорочення витрат, збільшення конверсії та покращення якості обслуговування користувачів. Вони не несуть жодної психологічної цінності: вони не імітують повноцінне спілкування та людські емоції (за винятком суму, що клієнт вирішив відмовитися від послуг). Проте вони мають практичну цінність, наприклад, Telegram має чат-бот прогнозу

погоди. Також месенджер надає змогу створювати власні чат-боти для збору інформації або інформування людей.

Відповідно до статистики за 2023 рік, 1,4 мільярда людей користуються чат-ботами, а 60% людей спілкувалися з чат-ботом протягом останніх 12 місяців [9]. Це свідчить про те, що ця технологія дедалі більше популяризується у суспільстві та має практичне значення.

Незважаючи на таку популярність, жоден месенджер не містить чат-бота помічника, який би не просто відповідав на запитання, а й був повноцінним постійним учасником процесу листування.

### **1.3 Висновки до розділу 1**

Отже, месенджери відносяться до клієнт-серверних застосунків, які використовують протоколи HTTP чи TCP/IP. У той час, як перший потребує додаткового забезпечення безпеки передачі даних, другий відноситься до безпечних протоколів передачі, який передає дані у байтах.

Історія створення першого месенджеру бере свій початок у 1996 році. А сучасний світ має декілька таких застосунків, які мають свої переваги та недоліки.

Поява чат ботів у месенджерах мала практичне значення: вони відповідали на питання, які часто задаються, рекламували товари, надавали цікаву інформацію користувачеві. Проте жоден месенджер не має свого боту, який би намагався задовольнити потяг користувача у спілкуванні, а також мав психологічну функцію підтримки та імітував будування людських відносин із користувачем.

## 2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

Реалізація завдання передбачає створення системи з трьох окремих компонентів: бази даних, де будуть зберігатися дані, серверної частини, яка буде координувати передавання даних між клієнтами застосунку а також працювати з даними із бази даних та клієнту служби, який є графічним інтерфейсом користувача.

Клієнт застосунку повинен містити програмну реалізацію бота-компаньйона.

Таким чином, взаємодія елементів системи відображена на рис. 2.1: клієнт відправляє та отримує повідомлення від серверу, який чекає на запити від клієнтів та взаємодіє із базою даних.

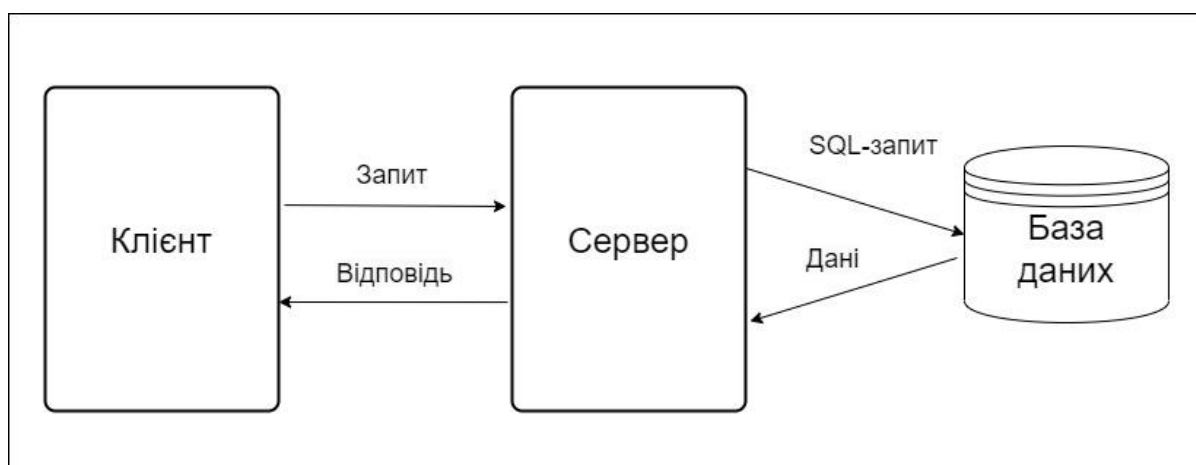


Рисунок 2.1 – Система роботи застосунку

### 2.1 Вибір мови програмування

Месенджери, які є мобільними застосунками, використовують протокол HTTP. Месенджери, які є десктопними застосунками, використовують протокол TCP/IP.

Аналізуючи мови програмування, які найчастіше використовуються для розробки веб-застосунків, такі як JavaScript, TypeScript та Python, додатково

можна зауважити, що вони містять лише умовно-приватні дані. Наприклад, у мові Python є інструменти, які перешкоджають зчитуванню даних за замовчуванням із модулю. Однак, якщо розробник або зловмисник захоче зчитати дані напряду за назвою змінних, це йому вдасться.

На відміну від цього, мови програмування, які компілюються, такі як C++, C# та Java, перетворюють дані у виконуваний бінарний файл, з якого неможливо зчитати конкретні класи чи функції. Тому приватні дані, які містять ці мови програмування, дійсно приховані від зовнішнього впливу.

Перший месенджер ICQ, який був розроблений компанією Mirabilis у 1996-1998 роках, написаний мовою C++, тобто тією, що компілюється та використовує протоколи транспортного рівня моделі OSI.

Наступні популярні месенджери, Skype від компанії Microsoft (рік створення – 2003) та Telegram від компанії Telegram FZ-LLC (який вийшов у світ у 2013 році), також використовували переважно мову C++. Але Telegram вже містив мову Java, котра виникла пізніше мови C++.

Мова програмування C++, розробником якої є Б'ярн Страуструп, є найпершим удосконаленням мови C. Вона містить модель об'єктно-орієнтованої парадигми програмування та може взаємодіяти як з користувацьким рівнем операційної системи, так і безпосередньо з ядром. Крім цього, вона дозволяє працювати з пам'яттю за допомогою її виділення та повернення, що дає великі можливості в галузі підвищення швидкодії застосунку [10].

Проте, такі переваги мови C++ стають одночасно її недоліками, адже розробнику доводиться самотійно постійно працювати з виділенням та вивільненням пам'яті, що може привести до додаткових помилок. До того ж, на думку самого Б'ярна Страуструпа, «краще взагалі не вивільняти пам'ять ніж допустити роботу деструктора двічі» [10, с.144]. Адже у першому випадку буде виток пам'яті, а у другому це може призвести до фатальної помилки.

Підводячи висновки, мова програмування C++ підходить у першу чергу застосункам, для яких на першому місці стоїть швидкодія або робота з ядром операційної системи.

Ще одним недоліком мови C++ є складність реалізації графічних інтерфейсів. Одними з найпоширеніших є бібліотека MFC (Microsoft Foundation Classes), яка використовує API Windows та фреймворк QT. Перша потребує написання детальних обробників подій, проте в її інструментах менше можливостей ніж в інших мовах програмування [11].

На відміну від неї, фреймворк QT уявляє собою простий інструмент створення графічного інтерфейсу користувача. Проте, він містить свої додаткові класи та методи. Оболонка фреймворку впливає на швидкодію коду, адже QT має власний препроцесор МОС, яки переводить специфічні класи та методи QT у чистий код C++, котрий потім також проходить компіляцію [11].

Перевагою QT є те, що розробнику не потрібно знати інші мови програмування для створення графічного інтерфейсу для застосунку, написаного мовою C++. Але мінуси, такі як знання спеціалізованих бібліотек та класів QT, зменшення швидкодії виконання коду, важка у налаштуванні середа розробки, робить її переваги умовними.

Через це в сучасному світі програмування поширеним явищем є написання основної частини застосунку мовою C++, а графічного інтерфейсу користувача – за допомогою мов програмування, які мають зручні та потужні інструменти для цього. Наприклад, мовою C#, яка має одночасно два зручних API-інтерфейсів для цього.

Мови програмування Java та C# з'явилися на десятиріччя пізніше C++. На відміну від останнього, вони не взаємодіють з ядром операційної системи та пам'яттю напряду. Так, у Java після компіляції код транслюється у віртуальну Java-машину, яка і виконує інструкції. Перевагою цієї мови програмування є те, що вона крос платформна, адже для виконання програми потрібна лише середа віртуальної машини. Проте ця мова має і недоліки. До них належать:

відсутність можливості передачі даних за посиланням, невелика кількість інструментів при створенні графічного інтерфейсу.

На відміну від Java, мова C# успадкувала від C можливість передачі даних за посиланням, а також має багато інструментів для створення графічних інтерфейсів користувача.

Мова програмування C# відноситься до C-подібних мов та успадкувала від останнього багато інструментів. Так, вона є об'єктно-орієнтованою та дозволяє множинне успадкування від інтерфейсів (також доступне успадкування від одного класу). Як і у Java, в неї є оболонка, яка взаємодіє з операційною системою, а тому у розробника немає необхідності виділяти та визволяти пам'ять під змінні. Проте, на відміну від Java, у мові C# є можливість використовувати покажчики, хоча це не рекомендовано робити [12].

Оскільки в C# дані є об'єктами, а робота з пам'яттю перекладається на збиральника сміття, вона відноситься до високорівневих мов програмування. Дані, які компілюються, транслуються у Common Language Infrastructure (CLI) – виконуючу середу Microsoft .NET Framework, що робить їх сумісними з іншими .NET-сумісними мовами (наприклад, C++, Visual Basic .NET, F# тощо). Наприклад, серіалізовані дані на C# можуть оброблюватися на Visual Basic .NET [12].

До переваг середи Microsoft .NET Framework також належать: ADO.NET (ActiveX Data Object для .NET) – технологія, яка містить додаткові інструменти для роботи із базою даних, WCF (Windows Communication Foundation) – фреймворк, який містить інструменти для розробки клієнт-сервісних застосунків та системи для конструювання графічних інтерфейсів користувача WPF (Windows Presentation Foundation) і WinForms. Важливою перевагою цих фреймворків є те, що в них використовується тільки мова C# на відміну від додаткового фреймворку QT для C++, яки потребує додаткового перетворення зі специфічних класів QT [13]-[17].

Сумуючи вищезазначене, можна порівняти мови програмування C++, Java та C# у табл. 2.1.

Таблиця 2.1 – Порівняння мов програмування

| Критерій порівняння                                              | C++ | C# | Java |
|------------------------------------------------------------------|-----|----|------|
| Додаткові інструменти для розробки клієнт-сервісних застосунків. | -   | +  | -    |
| Додаткові інструменти для створення графічних інтерфейсів.       | +   | +  | -    |
| Додаткові фреймворки для роботи з базою даних.                   | -   | +  | -    |
| Крос платформність.                                              | +   | +- | +    |
| Можливість прямої взаємодії з пам'яттю.                          | +   | +  | -    |
| Швидкість виконання за рахунок прямої роботи з пам'яттю.         | +   | -  | -    |
| Додаткові фреймворки для створення клієнт-серверних застосунків  | -   | +  | -    |

Таким чином, мова C++ найкраще підійде для розробки застосунків, де потрібна швидкість виконання. У той час, як мова Java є універсальною для розробки крос платформних застосунків. Для реалізації клієнт-серверних застосунків, а також графічних інтерфейсів користувача найбільше інструментів у мові C#. Тобто, вона є найкращим рішенням для реалізації поставленої мети дипломної роботи.

## 2.2 Аналіз функціональних вимог

Функціональні вимоги: робота із системою передбачає наявності двох сутностей: адміністратору, який керує запуском серверної частини та користувачу додатку.

Система повинна реалізовувати наступні вимоги:

- надавати адміністратору можливість запускати сервер;
- надавати адміністратору можливість зупиняти сервер;
- надавати адміністратору можливість вносити дані у базу даних;

- надавати адміністратору можливість стеження за сервером;
- надавати користувачу можливість реєстрації у системі;
- надавати користувачу можливість аутентифікації у системі;
- надавати користувачу можливість взаємодіяти із ботом-компаньйоном;
- надавати користувачу можливість відправляти повідомлення іншим користувачам;
- надавати користувачу можливість отримувати повідомлення від інших користувачів;
- надавати користувачу можливість дивитися опис-історію бота-компаньйона;
- надавати користувачу можливість редагувати особисті дані;
- надавати користувачу можливість дивитися дані інших користувачів.

Таким чином, діаграма бізнес-прецедентів для адміністратора зображена на рис. 2.2.



Рисунок 2.2 – Діаграма бізнес-прецедентів для адміністратора застосунку

Діаграма бізнес-прецедентів для користувача застосунку зображена на рис. 2.3.



Рисунок 2.3 – Діаграма бізнес-прецедентів для користувача застосунку

## 2.3 Проектування архітектури бази даних

База даних повинна містити інформацію про користувача: його особисті дані та пароль. Для більшої надійності захисту даних, логін та пароль користувача буде винесено в окрему таблицю. Для збереження історії переписки користувачів, потрібна таблиця із повідомленнями.

За відносини з ботом-компаньйоном користувачеві нараховуються бали. За отримання певної кількості балів надається новий статус відносин. Тобто, система потребує таблиці для збереження кількості балів користувача та окремої таблиці з описом існуючих статусів для відносин.

Таким чином, база даних повинна містити п'ять таблиць: таблицю логінів та паролів, таблицю даних о користувачах, таблицю зі збереженням даних переписок користувачів, таблицю з описом статусів відносин з ботом-компаньйоном та таблицю з балами користувачів за відносини.

Таблиці даних о користувачах та даних з повідомленнями будуть мати зовнішні ключі на таблицю з логінами та паролями для спрощення вибірки даних та уникнення дублювання даних, а також можливості додавання нових таблиць та зміни існуючих без втрати існуючих даних. Таблиця із балами-відносинами буде мати зовнішні ключі і на таблицю логінів та паролів, і на таблицю опису статусів відносин. Таким чином, діаграма пакетів бази даних відображена на рис. 2.4.

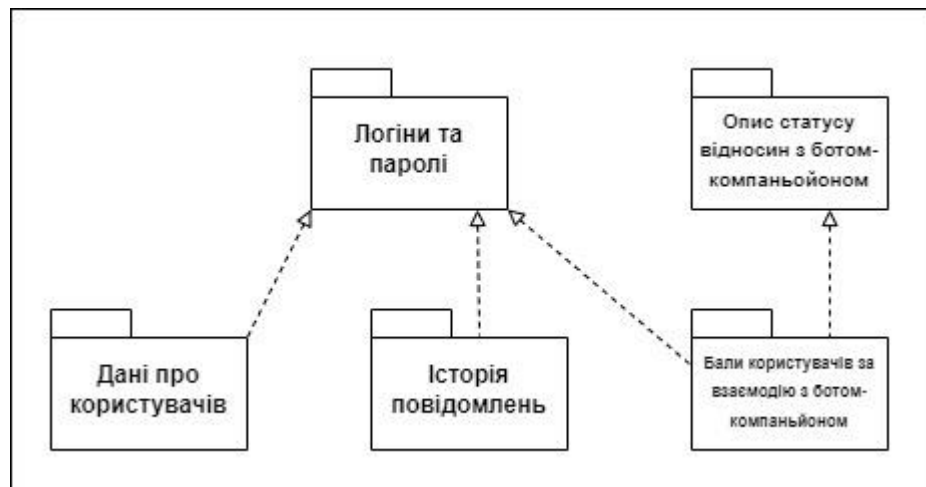


Рисунок 2.4 – Діаграма пакетів для бази даних

Найпоширенішими програмними рішеннями для створення, збереження та редагування даних є SQLite, MSSQL та MySQL.

База даних SQLite уявляє собою просте рішення для зберігання даних обмеженої кількості форматів. Фактично, це не СУБД, а файлова система збереження даних, що робить її більш вразливою.

Платформа MSSQL є потужним рішенням для збереження даних великого формату, а також надає велику кількість типів даних. Проте безкоштовна версія цієї СУБД містить обмежений функціонал. Також MSSQL має власну мову T-SQL, яка розширює спектр можливостей SQL [18].

На відміну від MSSQL, MySQL є повністю безкоштовною СУБД, яка містить достатню кількість форматів даних, зручна у використанні та підходить як для веб-розробки, так і для десктопних застосунків. Таким чином, вона є

оптимальним варіантом, якщо потрібне більш потужне рішення, ніж SQLite, але потребує не комерційного використання [18]-[19].

Сумуючи вищезазначене, порівняння систем керування базами даних наведене у табл. 2.2.

Таблиця 2.2 – Порівняння систем керування базами даних

| Критерій порівняння | SQLite  | MSSQL                       | MySQL           |
|---------------------|---------|-----------------------------|-----------------|
| Архітектура.        | файлова | клієнт-серверна             | клієнт-серверна |
| Мова запитів.       | SQL     | T-SQL                       | SQL             |
| Рівень безпеки.     | низький | високий                     | високий         |
| Безкоштовна.        | так     | тільки обмежений функціонал | так             |

## 2.4 Проєктування архітектури серверу

Для функціонування сервісу в мережі Інтернет, він потребує хосту. «Хост – це пристрій, який приєднується до мережі Інтернет, утворюючи вузол приймання та відправлення даних у ній» [15, с.11]. Для організації роботи служби застосунку, що уявляє собою клієнт-серверний додаток, який передає дані через протокол TCP/IP, Microsoft .NET Framework містить бібліотеку для створення автохостингу, тобто, дозволяє запускати хост-процес у консольному або графічному застосунку, розробленому мовою C#.

Також окремо слід зауважити, що WCF-фреймворк дозволяє створювати служби рівня виклику, сесії та одинарні (мають один екземпляр для всіх клієнтів) [15].

Для того, щоб передавати повідомлення за допомогою функції зворотного виклику клієнтам, які перебувають у мережі, використовується служба одного екземпляру.

Однак, подібні служби мають порівняно невеликий рівень масштабування. До того ж, повноваження передання повідомлень зворотнім викликом створює велике навантаження на швидкодію. Тож, потрібно виділити окремо службу для передачі повідомлень та збереження їх у базі даних, а роботу з будь-якими даними користувача надати окремо створеній службі рівню сесії. Тому серверна частина застосунку потребує двох служб: рівню сесії та рівню одного екземпляру.

Для зберігання принципу модульності та інкапсуляції окремих частин програми, потрібно створити проект-бібліотеку, яка буде містити класи для з'єднання з базою даних та функціями у них для створення SQL-запитів.

Тобто, система серверу містить чотири елементи: хост, який виступає вузлом з'єднання з мережею Інтернет, службу рівню одного екземпляру для приймання та відправлення повідомлень клієнтів, які підключені до мережі, службу рівня сесії для роботи із даними інформації про користувачів, а також пересилання архіву історії повідомлень та бібліотеки, що інкапсулює роботу з базою даних. Система зображена на рис. 2.5.

Зв'язок між службами та хостом повинен робитися динамічно, тобто, кінцеві точки прив'язки зазначаються у файлі `app.config`. Це дозволить змінювати адресу хосту чи служби динамічно, без повторної перекомпіляції застосунку.

Бібліотека для запитів до бази даних повинна бути сконструйована за допомогою патерну «Фабричний метод».

Фабричний метод – це патерн, який відноситься до породжуючих. Його також називають «віртуальним конструктором», адже він надає дочірнім класам інтерфейс для створення нового класу. В момент створення клас-нащадок визначає, який саме клас потрібно створити. Цей шаблон делегує створення об'єктів нащадкам батьківського класу [20].

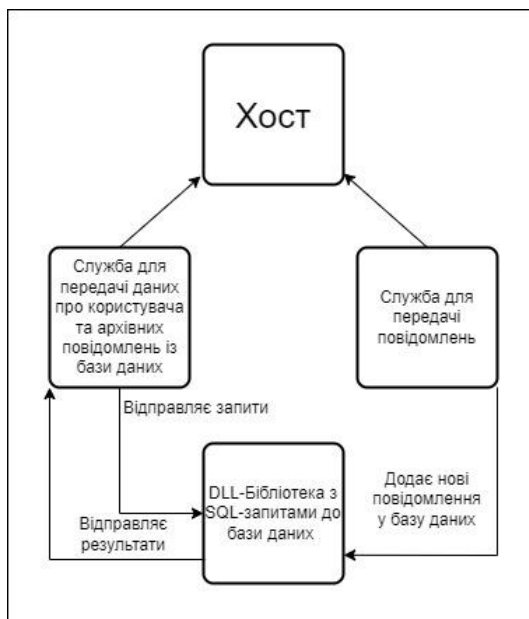


Рисунок 2.5 – Система роботи серверу

В мові C# відносно до взаємодії з базою даних, його суть полягає у реалізації існуючого інтерфейсу стандартної бібліотеки `ConfigurationManager`. При цьому, зв'язок із базою даних та вибір постачальника робиться динамічно у файлі `app.config`. Для роботи з базою даних, слід виділити окремий клас для роботи з таблицею логінів та паролів, даних про користувача, архіву повідомлень а також роботою з інформацією про відносини з ботом-компаньйоном. У подальшому це дозволить краще тестувати та розширювати систему.

## 2.5 Проектування архітектури графічного інтерфейсу користувача

Архітектура клієнту системи проектується за допомогою патерну MVVM.

Патерн MVVM (Model-View-ViewModel) дозволяє відокремити логіку застосунку від візуальної частини. Він складається з трьох компонентів: моделі (Model), моделі уявлення (ViewModel) та уявлення (View). Модель інкапсулює у собі сутність, яка відображається в уявленні, а модель уявлення керує зв'язком між візуальним відображенням та даними. Це робить архітектуру програми більш гнучкою та масштабованою [16].

Взаємодія між компонентами представлена на рис. 2.6.

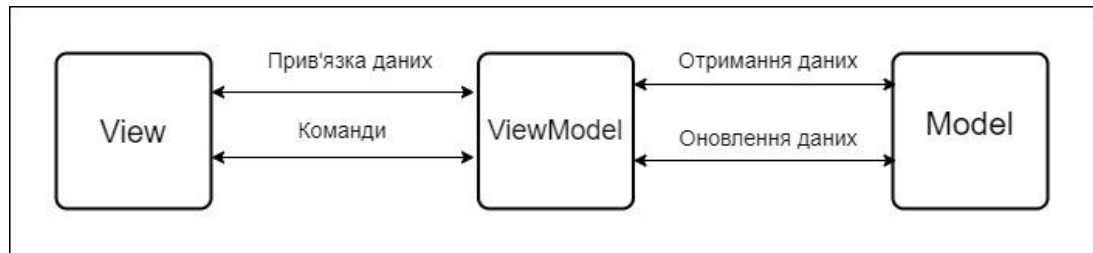


Рисунок 2.6 – Схема реалізації патерну Model-View-ViewModel

Окремою вимогою є зрозумілість інтерфейсу та його візуальна привабливість.

## 2.6 Проектування боту-компаньйону

Для створення боту-компаньйону було обрано образ голуба альбіноса, адже саме голуби є символами отримання добрих вістей та повідомлень.

Щоб надати компаньйону ознаки реальної істоти, у нього повинна бути власна історія та емоційні реакції, а також задана поведінка взаємодії: звернення до користувача, пропонування сумісних дій.

Для взаємодії з користувачем, зображення боту проектується на кнопки.

Реалізація боту-компаньйону передбачає зміну станів зображення компаньйону та його текстового звернення до користувача за таймером. Тобто, це найкраще реалізує патерн «Абстрактна фабрика».

«Абстрактна фабрика – це породжуючий патерн, який надає простий інтерфейс для створення об'єктів, які належать до того чи іншого сімейства» [21, с.11].

Архітектура боту-компаньйону складається з наступних станів:

- «прилітає» та вітає; слухає користувача;
- задає питання користувачеві;
- негативної реакції на відповідь;
- позитивної реакції на відповідь;

- сповіщає про нове повідомлення від іншого користувача;
- п'є каву (попередньо запропонував зробити це разом із користувачем).

Тобто, абстрактний інтерфейс дії компаньйону успадковується сьома конкретними класами дій або емоцій.

Окремо слід зазначити перший стан боту-компаньйону: вітання здійснюється шляхом показу декількох зображень компаньйону та звуку порхання крил птиці. Ця операція навантажує інтерфейс, та щоб уникнути цього, слід використовувати асинхронні методи програмування.

На платформі C# 5.0 асинхронні функції позначаються як `async`, а запуск паралельного процесу здійснюється за допомогою виразу `await` [22].

Для створення інтерактивності та ефекту присутності, бот-компаньйон при натиску на нього, видає звук воркотіння голубу. Також звук воркотіння супроводжує повідомлення про те, що надійшло повідомлення від іншого користувача.

## 2.7 Висновки до розділу 2

Проектування застосунку створюється з урахуванням подальшої масштабованості та зміни окремих частин без впливу на інші компоненти програмного забезпечення.

Це реалізується за допомогою модульності, переважно компонентно-орієнтованого підходу архітектури та частковим застосуванням породжуючих патернів програмування.

## 3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

### 3.1 База даних

Створена база даних має назву «siggi» за ім'ям боту-компаньйону. Вона містить п'ять таблиць: passwords, users, messages, relationship, reldescript.

Таблиця passwords містить дані про унікальний ідентифікатор користувача, його логін та пароль. Перевірка на унікальність логіну є у функціях сервісу, тому у базі даних в полі немає відзнаки unique. Таблиця містить наступні колонки:

- id: первинний ключ, чисельне поле, заповнюється автоматично, не може містити тип NULL;
- NickName: логін користувача, текстове поле, максимальна кількість символів – 200, не може бути NULL;
- Password: - пароль користувача, оскільки зберігається у вигляді хешу, може містити до 1000 символів, не може бути NULL;

Повний вигляд таблиці представлений на рис. 3. 1.



| # | Name     | Type          | Collation       | Attributes | Null | Default | Comments | Extra          | Action           |
|---|----------|---------------|-----------------|------------|------|---------|----------|----------------|------------------|
| 1 | id       | int(11)       |                 |            | No   | None    |          | AUTO_INCREMENT | Change Drop More |
| 2 | NickName | varchar(100)  | utf8_general_ci |            | No   | None    |          |                | Change Drop More |
| 3 | Password | varchar(1000) | utf8_general_ci |            | No   | None    |          |                | Change Drop More |

Рисунок 3. 1 – Таблиця passwords

Таблиця users містить дані про користувача: його ім'я, прізвище, країну, місто та фотографію. Також вона містить унікальний номер запису та зовнішній ключ-посилання на ідентифікатор таблиці passwords.

Колонки таблиці:

- **id**: первинний ключ, чисельне поле, заповнюється автоматично, не може містити тип NULL;
- **userId**: зовнішній ключ-посилання на поле **id** таблиці **passwords**. Чисельного типу. Не може містити тип NULL;
- **Name**: ім'я користувача, яке він може заповнювати та змінювати за бажанням. Текстове поле, максимальна кількість символів – 200. Може містити тип NULL;
- **LastName**: прізвище користувача, текстове поле, максимальна кількість символів – 200. Може заповнюватися та змінюватися користувачем за бажанням. Може містити тип NULL;
- **Country**: текстове поле для відображення країни користувача, яке заповнюється за бажанням, максимальна кількість символів – 200. Може містити тип NULL;
- **City**: назва міста, де живе користувач. Може заповнюватися за бажанням. Максимальна кількість символів текстового поля – 200. Може містити тип NULL;
- **Photo**: фотографія користувача, поле типу **longblob**. Оскільки має вимоги до розміру та формату, додається адміністратором за бажанням користувача. Може містити тип NULL;

Представлення таблиці відображене на рис. 3. 2.

| # | Name            | Type         | Collation       | Attributes | Null | Default | Comments | Extra          | Action           |
|---|-----------------|--------------|-----------------|------------|------|---------|----------|----------------|------------------|
| 1 | <b>id</b>       | int(11)      |                 |            | No   | None    |          | AUTO_INCREMENT | Change Drop More |
| 2 | <b>userId</b>   | int(11)      |                 |            | No   | None    |          |                | Change Drop More |
| 3 | <b>Name</b>     | varchar(200) | utf8_general_ci |            | Yes  | NULL    |          |                | Change Drop More |
| 4 | <b>LastName</b> | varchar(200) | utf8_general_ci |            | Yes  | NULL    |          |                | Change Drop More |
| 5 | <b>Country</b>  | varchar(200) | utf8_general_ci |            | Yes  | NULL    |          |                | Change Drop More |
| 6 | <b>City</b>     | varchar(200) | utf8_general_ci |            | Yes  | NULL    |          |                | Change Drop More |
| 7 | <b>Photo</b>    | longblob     |                 |            | Yes  | NULL    |          |                | Change Drop More |

Рисунок 3. 2 – Таблица users

Таблиця messages зберігає архів переписки між користувачами. Містить наступні дані: унікальний ідентифікатор запису, зовнішній ключ-посилання на id користувача у таблиці passwords, який посилав повідомлення, зовнішній ключ-посилання на id користувача у таблиці passwords, який отримував повідомлення, дату у чисельному вигляді та текст повідомлення.

Дата зберігається у чисельному типі, адже тоді можна уникнути конфлікту відображення даних, а також прискорити зчитування даних з таблиці.

Таблиця містить наступні колонки:

- id: первинний ключ, чисельне поле, заповнюється автоматично, не може містити тип NULL;
- UserFrom: зовнішній ключ-посилання на поле id таблиці passwords користувача, який відправляв повідомлення. Чисельного типу. Не може містити тип NULL;
- UserTo: зовнішній ключ-посилання на поле id таблиці passwords користувача, якому було призначене повідомлення. Чисельного типу. Не може містити тип NULL;
- Date: поле, яке містить дату отримання сервером повідомлення у чисельному еквіваленті.
- TextMessage: текстове поле для повідомлення.

Загалом таблиця виглядає наступним чином (рис. 3. 3):

| # | Name        | Type       | Collation       | Attributes | Null | Default | Comments | Extra          | Action           |
|---|-------------|------------|-----------------|------------|------|---------|----------|----------------|------------------|
| 1 | id          | int(11)    |                 |            | No   | None    |          | AUTO_INCREMENT | Change Drop More |
| 2 | UserFrom    | int(11)    |                 |            | No   | None    |          |                | Change Drop More |
| 3 | UserTo      | int(11)    |                 |            | No   | None    |          |                | Change Drop More |
| 4 | Date        | bigint(20) |                 |            | No   | None    |          |                | Change Drop More |
| 5 | TextMessage | text       | utf8_general_ci |            | No   | None    |          |                | Change Drop More |

Рисунок 3. 3 – Таблиця messages

Таблиця `relationship`, яка відображає дані балів за стосунки з ботом-компаньйоном. Містить наступні дані: унікальний ідентифікатор запису, зовнішній ключ-посилання на `id` користувача у таблиці `passwords`, дату останнього запису внесення отриманого балу у чисельному вигляді.

Колонки таблиці:

- `id`: первинний ключ, чисельне поле, заповнюється автоматично, не може містити тип `NULL`;
- `userId`: зовнішній ключ-посилання на поле `id` таблиці `passwords`. Чисельного типу. Не може містити тип `NULL`;
- `Points`: чисельне поле, яке відображає бали користувача за стосунки з ботом-компаньйоном. Не може містити тип `NULL`;
- `LastWisited`: поле, яке містить дату останньої зміни значення балів у чисельному еквіваленті.

Повний вигляд таблиці представлений на рис. 3. 4.

| # | Name        | Type       | Collation | Attributes | Null | Default | Comments | Extra          | Action           |
|---|-------------|------------|-----------|------------|------|---------|----------|----------------|------------------|
| 1 | id          | int(11)    |           |            | No   | None    |          | AUTO_INCREMENT | Change Drop More |
| 2 | userId      | int(11)    |           |            | No   | None    |          |                | Change Drop More |
| 3 | Points      | int(11)    |           |            | No   | 0       |          |                | Change Drop More |
| 4 | LastWisited | bigint(20) |           |            | Yes  | NULL    |          |                | Change Drop More |

Рисунок 3. 4 – Таблиця `relationship`

Таблиця `reldescript` зберігає текстовий опис статусу стосунків та поріг кількості балів, який для цього потрібен. Така архітектура таблиці дозволяє додавати нові статуси без втручання у дані накопичених балів користувачами. Містить наступні дані: унікальний ідентифікатор запису, текстовий опис статусу стосунків, кількість балів, яка потрібна для статусу.

Таблиця містить наступні колонки:

- `id`: первинний ключ, чисельне поле, заповнюється автоматично, не може містити тип `NULL`;

- **Description:** текстове поле опису статусу взаємовідносин з ботом-компаньйоном, максимальна кількість знаків – 100. Не може містити тип NULL;
- **Count:** чисельне поле потрібної кількості балів, щоб отримати статус.

На момент створення застосунку містить сім статусів: «Новий знайомий», «Знайомий», «Добрий знайомий», «Друг», «Найкращий друг», «Той, хто знає про Сіггі все» та «Супергерой:»).

Представлення таблиці відображене на рис. 3. 5.



| #                          | Name        | Type         | Collation       | Attributes | Null | Default | Comments | Extra          | Action             |
|----------------------------|-------------|--------------|-----------------|------------|------|---------|----------|----------------|--------------------|
| <input type="checkbox"/> 1 | id          | int(11)      |                 |            | No   | None    |          | AUTO_INCREMENT | Change  Drop  More |
| <input type="checkbox"/> 2 | Description | varchar(100) | utf8_general_ci |            | No   | None    |          |                | Change  Drop  More |
| <input type="checkbox"/> 3 | Count       | int(11)      |                 |            | No   | None    |          |                | Change  Drop  More |

Рисунок 3. 5 – Таблиця reldescript

Таким чином, спроектовано чотири таблиці, пов’язані між собою: таблиці users, messages та relationship мають зовнішні ключі-посилання на поле id таблиці passwords. Окремо розроблена таблиця опису статусів відносин з ботом-компаньйоном, котра логічно пов’язана з таблицею відображення даних балів користувачів.

Графічне відображення зв’язків між таблицями представлено на рис. 3. 6.

### 3.2 Серверна частина

Серверна частина реалізована за принципом модульності, де окремий модуль – це логічно відділена частина за стосунку, що може бути переробленою без впливу на інші частини.

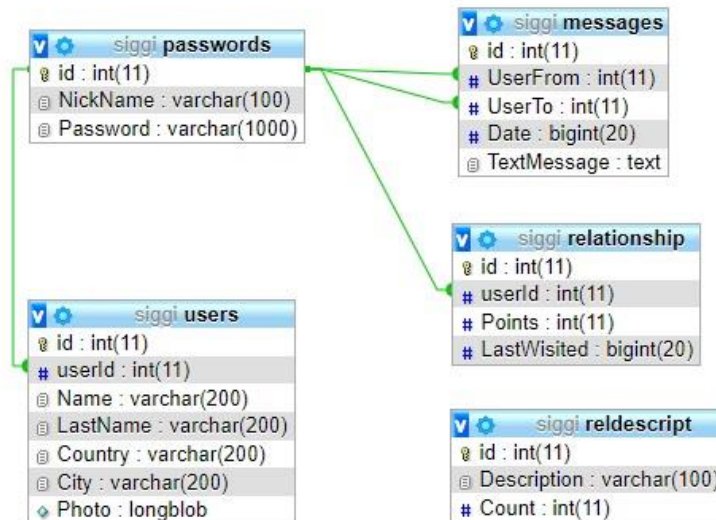


Рисунок 3. 6 – Структура бази даних

### 3.2.1. DLL-бібліотека для роботи з базою даних

Роботу з базою даних інкапсулює проект-бібліотека SiggiRepositoryMySQL. Це дозволяє відокремити роботу служби від запитів до бази даних. В подальшому, це дозволить змінювати класи роботи з даними або додавати нові без втручання у логіку роботи сервісів. Також окремий проект дозволить без впливу на роботу інших частин системи змінити СУБД, наприклад, на MSSQL, або продублювати у неї зберігання даних.

Підключення до даних відбувається у файлі App.config проекту, що також робить гнучкою можливість зміни постачальника. Назву самої бази даних приймає конструктор класів роботи із запитами, тобто, DLL-бібліотека не має статичного зв'язку з конкретною базою.

Для передавання рядку з'єднання, виконуваний модуль проекту повинен містити рядок параметрів connectionStrings у App.config проекту і цього буде достатньо, щоб бібліотека розпізнала об'єкт підключення.

Щоб працювати з сутностями таблиць бази даних, було створено дві моделі класів: UserInfoDB, який є представленням полів таблиці users та MessageForDB, який повторює колонки таблиці messages.

Для кожної таблиці, крім `reldescript`, було створено окремий клас для роботи із даними з неї.

Клас `LoginRepository` приймає у конструкторі назву бази даних та інкапсулює роботу з таблицею `passwords`, а саме: додавання та видалення нових користувачів, перевірку наявності логіну у базі, перевірку коректності паролю та його зміну. Також він передає сервісу список усіх логінів користувачів для передачі списку контактів клієнту застосунку.

Клас `UserRepository` також приймає у конструкторі дані для підключення до бази даних та інкапсулює роботу з таблицею `users`, а саме: додавання нового запису, зміну інформації про користувача, надає фотографію користувача, якщо вона є у базі.

До класу `MessageRepository` передається у конструкторі назва бази даних. Він відповідає за роботу з таблицею `messages`: формує SQL- запити для додавання нового повідомлення у базу та вибірки повідомлень між двома користувачами.

Для роботи з таблицями `relationship` та `reldescript`, які логічно пов'язані між собою, адже містять дані про відносини користувача з ботом-компаньйоном, створено клас `RelationshipRepository`. Він містить логіку додавання запису з новим користувачем у таблицю, зчитування та змінення балів за відносини та дати останньої зміни балів, а також надає текстовий статус відносин боту з користувачем.

Кожний клас для роботи із таблицями розроблений як окремий модуль, що може бути змінений без впливу на інший.

Всі класи бібліотеки відображаються на рис. 3.7, який є діаграмою класів DLL-бібліотеки.

Тобто, для реалізації бібліотеки використовувалася парадигма модульності та компонентно-орієнтованого програмування, а не об'єктно-орієнтованого для надання більшої гнучкості та масштабованості рішення.

Структура даних та організація класів відображена на рис. 3.8.

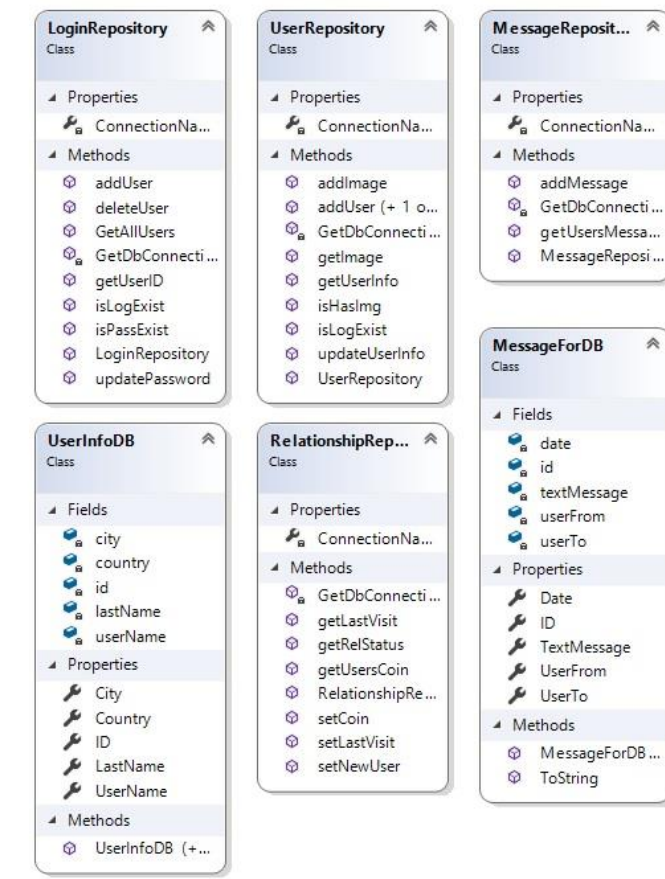


Рисунок 3. 7 – Діаграма класів DLL-бібліотеки для роботи з базою даних

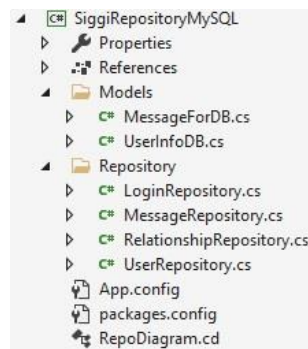


Рисунок 3.8 – Структура проекту DLL-бібліотеки

### 3.2.2. Модулі служб

Сервер має дві служби: рівню сесії для представлення даних користувачеві, їх зміни та рівню одного екземпляру для роботи з пересиланням повідомлень, а також додання їх у базу даних.

Для роботи з даними користувача було створено службу `IServiceUserInfo` рівня сесії. Вона відповідає за реєстрацію та аутентифікацію, зміну особистих даних, передачу списку користувачів (за винятком того, від кого був запит) та передачу історії повідомлень.

Створений інтерфейс містить не тільки функції, а й серіалізуємий клас `UserInfoServer`, який може використовувати як сама служба, так і її клієнти.

Реалізація інтерфейсу викладена у класі-нащадку `ServiceUserInfo`. Також останній містить два додаткових приватних методи: перевірки дати останнього начислення балів відносин та додаткового хешуванню пароля перед відправкою до бази даних.

Друга служба, визначена інтерфейсом `IServiceMessage`, має методи приєднання користувача до сервісу, від'єднання його, надсилання повідомлення та функція зворотного виклику для доставки повідомлення іншому клієнту. Це служба одного екземпляру.

Для обробки користувачів служби, які знаходяться в мережі, було додатково створено клас користувача сервісу `UserService`, якій інкапсулює логін користувача, контекст виклику та присвоєний службою ідентифікатор.

Статичний клас `CountUsers` фіксує кількість користувачів та стежить за присвоєнням унікального ідентифікатору.

Робота з користувачами: весь їх список, додавання, видалення, за потребою – обнуління кількості, реалізована у класі `ListUsers`, який містить клас `ServiceMessage`, що реалізує інтерфейс служби.

Додатково служба містить серіалізуємий клас `MessageServer`, який є сутністю повідомлення користувача.

Діаграма класів служб представлена на рис. 3.9.

Структура організації класів представлена на рис. 3.10.

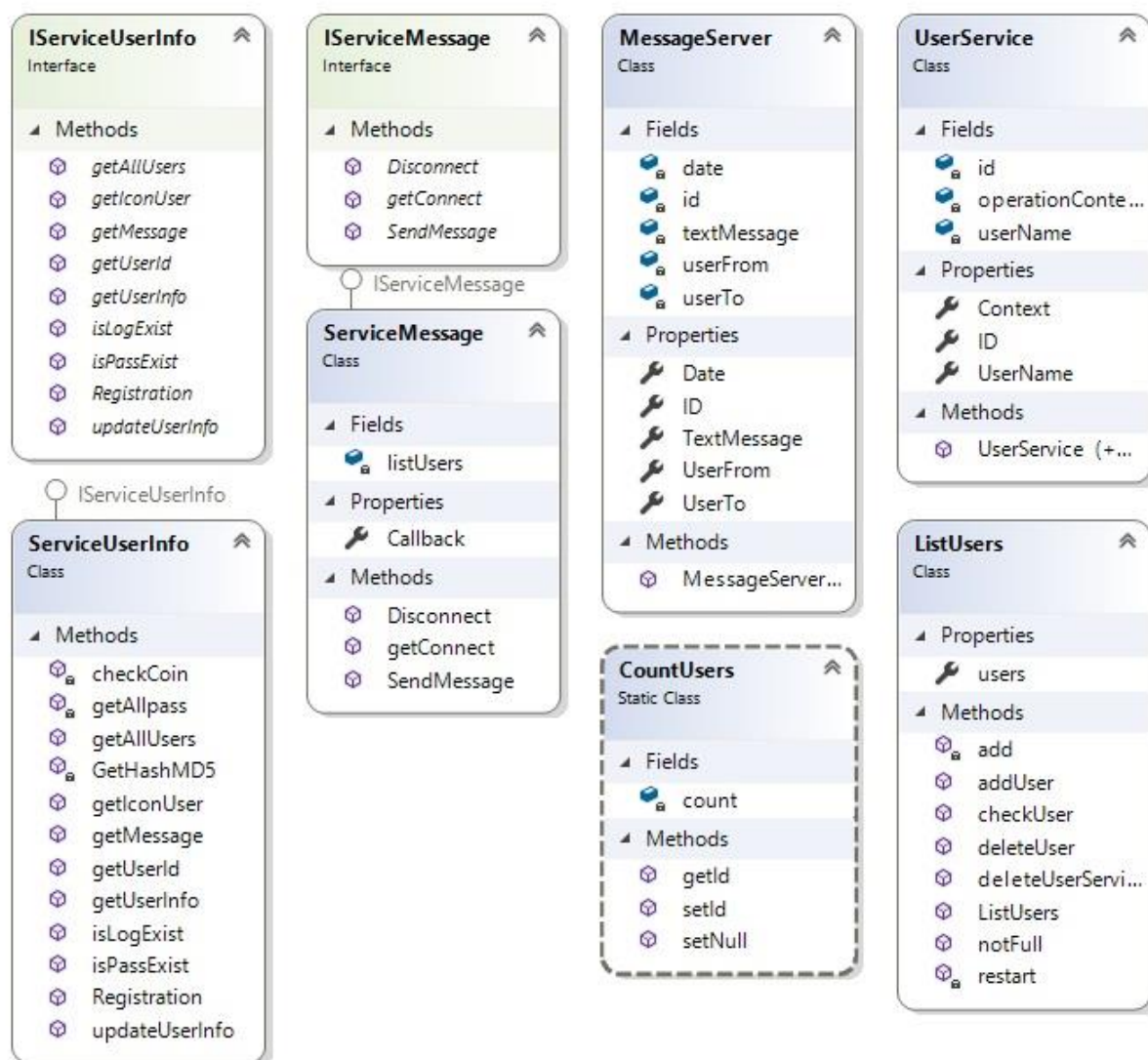


Рисунок 3.9 – Діаграма класів серверу

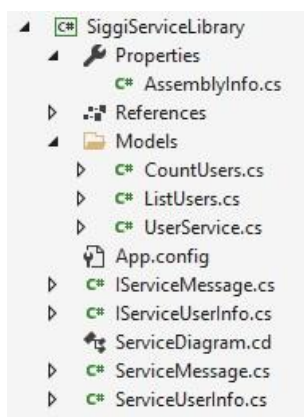


Рисунок 3.10 – Структура проекту серверу

### 3.2.3. Хост

Для того, щоб сервер функціонував у мережі Інтернет за протоколом TCP/IP, було створено консольний проекту хосту. Він містить класи HostUserInfo та HostMessage, які інкапсулюють створення, початок та завершення роботи служби, а також клас Program, де вони запускаються.

Для з'єднання з базою даних, у файлі App.config прописана строка підключення. Також він містить метадані служб, за допомогою яких проходить динамічна прив'язка з клієнтами додатку. Це дозволить у майбутньому змінювати кінцеві точки призначення або розташування хосту без повторної перекомпіляції застосунку.

Діаграма класів хосту представлена на рис. 3.11.



Рисунок 3.11 – Діаграма класів хосту

Структура організації класів представлена на рис. 3.12.

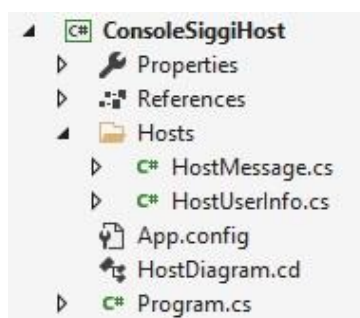


Рисунок 3.12 – Структура проекту хосту

### 3.3. Частина клієнта

Клієнтський застосунок умовно можна поділити на графічний інтерфейс користувача та реалізацію логіки боту-компаньйону.

З'єднання з сервером проводиться динамічно шляхом опису в файлі App.config метаданих.

При розробці клієнту повністю реалізовано патерн MVVM, який відокремлює візуальну частину від даних та обробки них.

#### 3.3.1. Графічний інтерфейс користувача

Дизайн графічного інтерфейсу виконаний у біло-голубих кольорах з акцентом у вигляді червоного та рожевого. Це зроблено для асоціації з кольорами голуба-альбіносу та неба.

Графічний інтерфейс користувача містить наступні вікна:

- MainWindow – вікно входу у додаток;
- Registration – вікно реєстрації нового користувача;
- AboutWindow – вікно з описам історії боту-компаньйону;
- UserInfoWindow – вікно для відображення даних користувача. Якщо воно викликається для відображення даних поточного користувача, він може їх змінити та відправити на сервер. Якщо для перегляду іншого користувача – текстові поля недоступні для зміни;
- SiggWindow – вікно для переписки з іншими користувачами а також взаємодії з ботом-компаньйоном.

Діаграма класів відображення зображена на рис. 3.13.

Для відображення сутностей даних було створено наступні класи-моделі:

- UserInvite для відображення даних сторінки реєстрації;
- UserInfo для відображення інформації про користувача;
- UserOfSigg для відображення даних користувача в вікні SiggWindow;

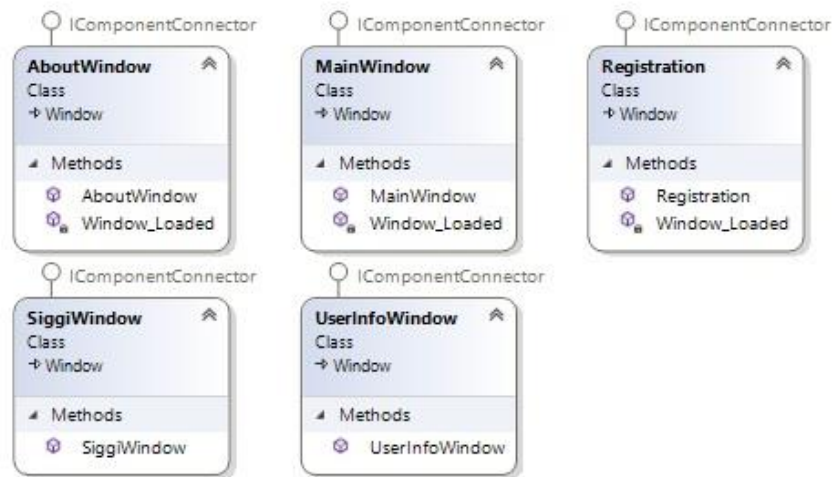


Рисунок 3.13 – Архітектура класів вікон за стосунку

- Contact для відображення контактів, доступних для переписки;
- Message для відображення повідомлень.
- AboutSiggi для відображення історії компаньйону.

Для того, щоб відокремити обробку кнопок від візуальної частини, було створено клас команди SimpleCommand, яка реалізує інтерфейс ICommand.

Всі класи наслідують інтерфейс INotifyPropertyChanged, що робить можливим їх відображення та зміну даних у вікнах.

Діаграма класів моделей зображена на рис. 3.14.

Для роботи із даними створено такі моделі представлення:

- LogWindowVM для обробки даних вікна входу у застосунок;
- RegistrationVM для роботи з даними вікна реєстрації;
- UserInfoWindowVM для відображення даних користувача;
- SiggiWindowVM та UsersMessagesVM для представлення даних сторінки переписки з користувачами та взаємодії з ботом-компаньйоном.

Щоб контролювати доступність кнопок та дій, вони містять команди, а для додавання, редагування та видалення даних, містять класи моделей сутностей.

Архітектура класів-моделей представлення показана на рис. 3.15.

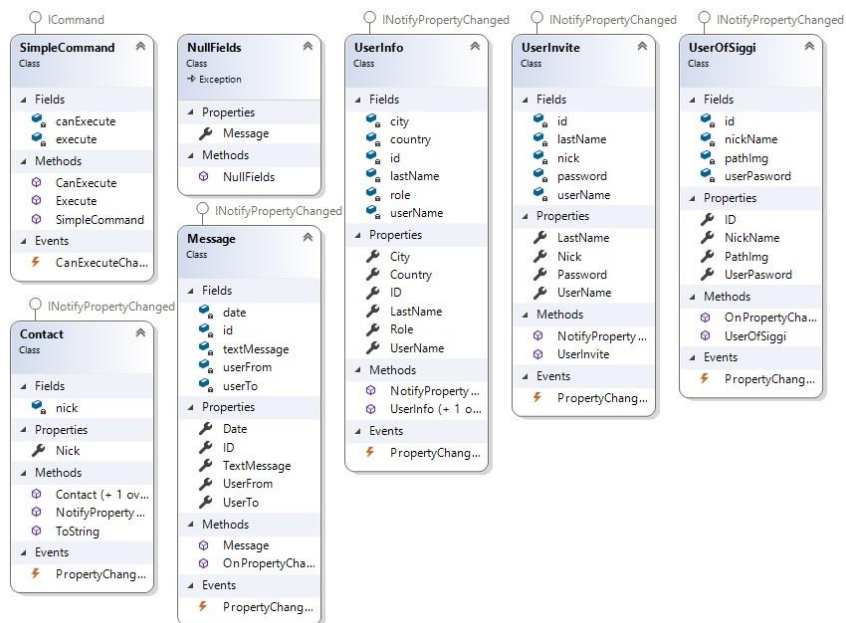


Рисунок 3.14 – Архітектура класів моделей застосунку

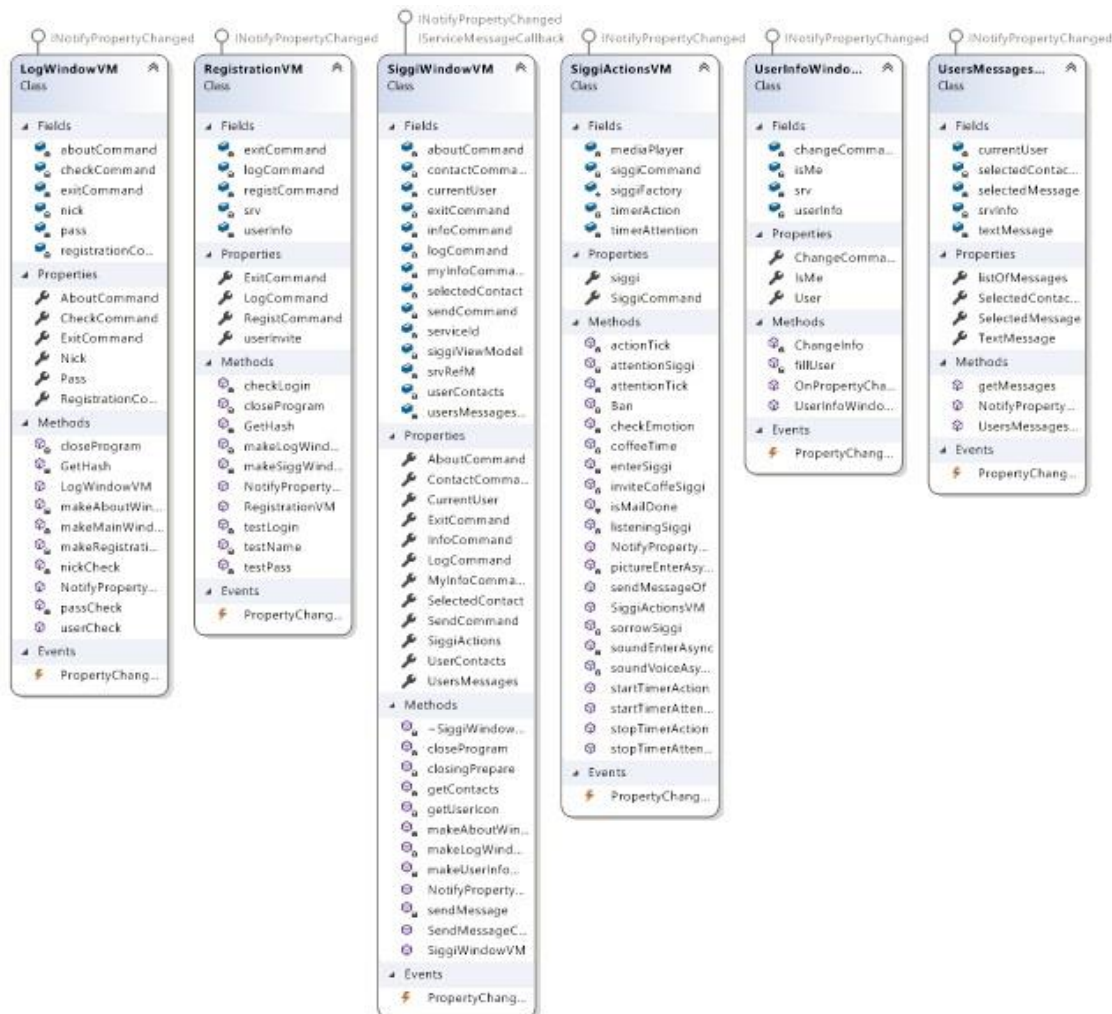


Рисунок 3.15 – Архітектура класів моделей-відображення застосунку

### 3.3.2. Бот-компаньйон

В проєкті поєдналася ідея віртуального компаньйону з реальним спілкуванням у месенджері.

Віртуальний компаньйон має свою історію та характер.

За легендою, це голубка-альбінос, яка має власне ім'я – Сіггі. Вона є високоінтелектуальною для свого виду особиною, через що у неї "не склалися стосунки" з її зграєю, тому вона вирішила "спробувати щастя з людьми", ставши для них корисною (передача повідомлень). Вона організувала свою власну поштову компанію «Siggi Post Corporation» (з анг. «Поштова корпорація Сіггі»). Крім цього, будучи комунікабельною, вона хоче знайти друзів серед людей. При цьому вона залишається птахом, що створює ґрунт для кумедних ситуацій та відповідей.

Залежно від статусу користувача по відношенню до неї, вона поступово розповідатиме про себе все більше і навіть поділиться своєю заповітною мрією.

Якщо ж відповідати грубо, весь час ігнорувати її, то Сіггі буде "засмучуватися" і навіть може тимчасово ігнорувати користувача.

При цьому максимальною санкцією є тимчасове ігнорування користувача за його поведінку (при збереженні функції передачі повідомлень).

Вона є вже дорослим птахом і існує стільки, скільки існуватиме сам месенджер, будучи його віртуальною власницею. Це виключить можливу психологічну травму у дітей, спричинену смертю компаньйону.

Візуально голуб-альбінос носить фірмову сумку, кепку листоноші та прикрасу – червоне намисто.

Знайомство користувача з ботом-компаньйоном починається з вибору у пункті меню «Трошки про Сіггі», де розкривається історія пташки.

Наступна взаємодія проходить у вікні SiggiWindow, де відображається картинка компаньйону.

Модель уявлення компаньйону відображена у наступних класах:

– Siggibird, який містить поля текстового звернення та картини голуба.

Для створення відображення станів пташки, застосований патерн «Абстрактна фабрика», тож було створено абстрактні класи SiggispeakText (для рядку повідомлення від боту) та SiggispeakPath для з шляху розташування потрібної картини.

Клас SiggispeakText наслідують такі класи:

- EnterSpeakText для привітання користувача;
- ListeningSpeakText для спостереження за користувачем;
- MailSpeakText для доставки повідомлення;
- InviteCoffeeSpeakText для прохання випити каву разом;
- CoffeeSpeakText для тексту під час пиття кави;
- AttentionSpeakText для прохання приділити увагу;
- SorrowSpeakText для вираження жалю або незадоволення;
- BanSpeakText для ігнорування користувача.

У свою чергу, абстрактний клас SiggispeakPath наслідують такі класи:

- ListeningSpeakPath для спостереження за користувачем;
- AttentionSpeakPath для прохання приділити увагу;
- MailSpeakPath для доставки повідомлення;
- CoffeeSpeakPath для картини під час пиття кави;
- SorrowSpeakPath для вираження жалю або незадоволення;
- BanSpeakPath для ігнорування користувача.

Слід зазначити, що класів з картинками менше, адже прохання приділити увагу та випити разом каву відрізняється лише текстом повідомлення від компаньйону.

Візуально зв'язок між абстрактними та конкретними класами відображає рис. 3.16.

Для компоновки тексту та картини у єдиний клас-представлення пташки, створено інтерфейс ISiggieMotionFactory.

Таким чином, стани боту-компаньйону наступні:

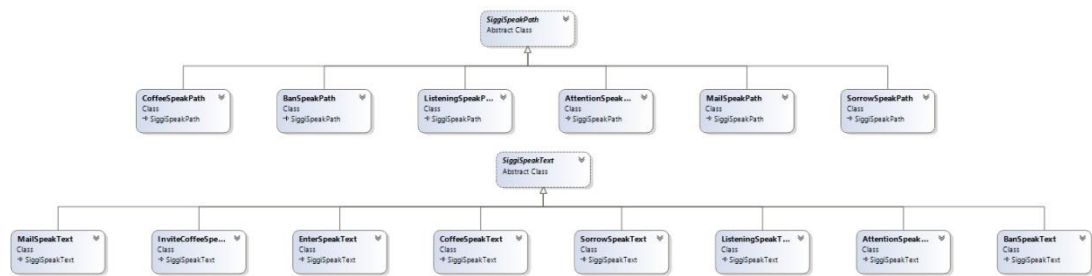


Рисунок 3.16 – Реалізація патерну «Абстрактна фабрика» через створення абстрактних класів SiggiSpeakText та SiggiSpeakPath

- EnterSpeakFactory для привітання користувача після входу у застосунок;
- ListeningSpeakFactory для спостереження за користувачем;
- MailSpeakFactory для сповіщення про нове повідомлення;
- AttentionSpeakFactory для привернення уваги;
- InviteCoffeeSpeakFactory для прохання випити разом кави;
- SorrowSpeakFactory для вираження жалю або незадоволення;
- BanSpeakFactory для ігнорування користувача.

Архітектура наслідування інтерфейсу класами-нащадками зображено на рис. 3.17.

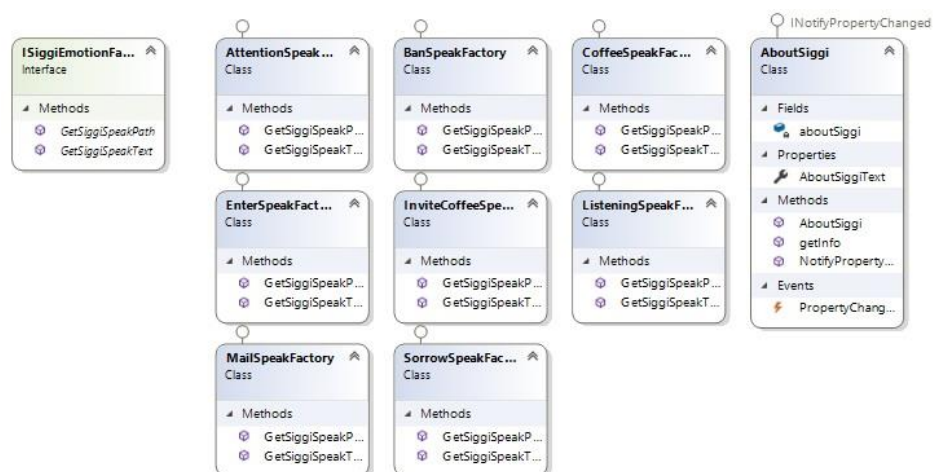


Рисунок 3.17 – Реалізація патерну «Абстрактна фабрика» через створення інтерфейсу, який успадковують класи, що повертають стан боту-компаньйону

Логіка дії компаньйону реалізована у класі `SiggiActionsVM`, який містить два таймери: для привернення уваги та прохання випити разом кави. Вони спрацьовують за заданою вірогідністю через певний проміжок часу. Вірогідність задається з метою імітації особистої волі компаньйона. Наприклад, якщо довго не спрацьовує дія за таймером, це значить, що компаньйон був не у настрою спілкуватися.

### **3.4 Висновки до розділу 3**

Програмна реалізація створеного застосунку має серверну та клієнтську частину. Для налаштування їх взаємодії використано фреймворк WCF та реалізовано динамічне зв'язування у файлах конфігурацій.

Клієнт реалізований за архітектурним патерном MVVM фреймворку WPF.

Бот-компаньйон спроектований за породжуючим патерном «Абстрактна фабрика».

## **4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ**

### **4.1 Експлуатація, тестування та експериментальне дослідження хосту**

Хост сервісу – це консольний застосунок, який після запуску видає інформацію про служби та сповіщає адміністратора про приєднання та від'єднання користувачів, а також показує сповіщення від бази даних, помилки та іншу інформацію, яку генерують служби під час виконання.

Оскільки застосунок хосту використовує порти операційної системи, вона повинна блокувати його запуск, якщо він проводиться не від імені адміністратора.

Тест запуску хоста від імені адміністратора.

Кроки відтворення:

- запустити консольний застосунок хосту від імені адміністратора операційної системи;
- спостерігати за інформацією від служб.

Очікуваний результат: хост запустився, вивив інформацію про запуснені служби та очікує з'єднання з користувачами. Виводить коректну інформацію щодо запитів та відповідей від служб.

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

Графічна ілюстрація тесту приведена на рис. 4.1.

Тест запуску хоста від імені користувача операційної системи без прав адміністратора.

Кроки відтворення:

- запустити консольний застосунок хосту від імені користувача операційної системи без прав адміністратора;
- отримати повідомлення про неможливість запуску серверу.

```

D:\DiplomZNTU\SiggiService\ConsoleSiggiHost\bin\Debug\ConsoleSiggiHost.exe
Address: http://localhost:8090/mex
Binding: MetadataExchangeHttpBinding
Contract: IMetadataExchange
Contract operation: System.ServiceModel.Description.IMetadataExchange
Contract operation:

**** HOST USER INFO ****

***** Host info: *****
Address: net.tcp://localhost:8311/
Binding: NetTcpBinding
Contract: IServiceUserInfo
Contract operation: SiggiServiceLibrary.IServiceUserInfo
Contract operation:

Address: http://localhost:8091/mex
Binding: MetadataExchangeHttpBinding
Contract: IMetadataExchange
Contract operation: System.ServiceModel.Description.IMetadataExchange
Contract operation:

Wcf host is running... press any key to stop
The user with id - 1 and login Freya has connected
The user with id - 2 and login Bishop has connected
The user with id - 2 disconnected
***NO***
***NO***
The user with id - 3 and login Bishop has connected

```

Рисунок 4.1 – Тест запуску хоста від імені адміністратора

Очікуваний результат: застосунок не запускається. Аналіз помилки видає неможливість приєднання до служби.

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

## 4.2 Експлуатація, тестування та експериментальне дослідження клієнту

Проектування графічного інтерфейсу користувача реалізоване за допомогою патерну MVVM, тому доступність тих чи інших дій можливе за виконання умови команди.

Так, при вході у застосунок, можливість логування заблокована доки не заповняться поля вікна.

Тест запуску клієнтського застосунку.

Кроки відтворення:

- запустити клієнтський застосунок;
- звернути увагу на доступність дії логування за відсутності заповнення полів вікна.

Очікуваний результат: кнопка «Увійти» та пункт меню «Увійти в обліковий запис» недоступні.

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

Графічна ілюстрація тесту приведена на рис. 4.2.



Рисунок 4.2 – Тест запуску клієнтського застосунку

Тест введення некоректних даних у поля вікна входу у застосунок.

Кроки відтворення:

- запустити клієнтський застосунок;
- заповнити поля не існуючим користувачем та його паролем;
- натиснути кнопку входу;
- звернути увагу на поля для заповнення даних.

Очікуваний результат: поля заповнені повідомленнями про помилку.

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

Графічна ілюстрація тесту приведена на рис. 4.3.



Рисунок 4.3 – Тест введення некоректних даних у поля вікна входу у застосунок

У разі переходу у вікно реєстрації, кнопка запуску основного вікна також є недоступною поки не будуть заповнені обов'язкові поля.

Тест кнопки входу у застосунок при не заповнених полях вікна реєстрації.

Кроки відтворення:

- запустити клієнтський застосунок;
- перейти у вікно реєстрації нового користувача;
- звернути увагу на доступність кнопки відправлення даних.

Очікуваний результат: кнопка «Полетіли» недоступна.

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

Графічна ілюстрація тесту приведена на рис. 4.4.

Тест реєстрації користувача, який вже є у базі.

Кроки відтворення:

- запустити клієнтський застосунок;

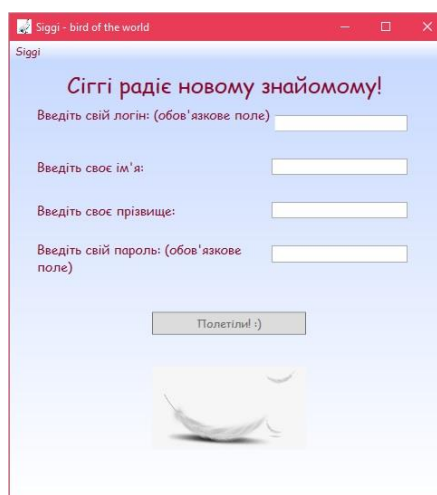


Рисунок 4.4 – Тест кнопки входу у за стосунок при не заповнених полях вікна реєстрації

- перейти у вікно реєстрації;
- заповнити поле логіну існуючим користувачем;
- звернути увагу на поля для заповнення даних.

Очікуваний результат: поле логіну заповнено повідомленням про помилку.

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

Графічна ілюстрація тесту приведена на рис. 4.5.

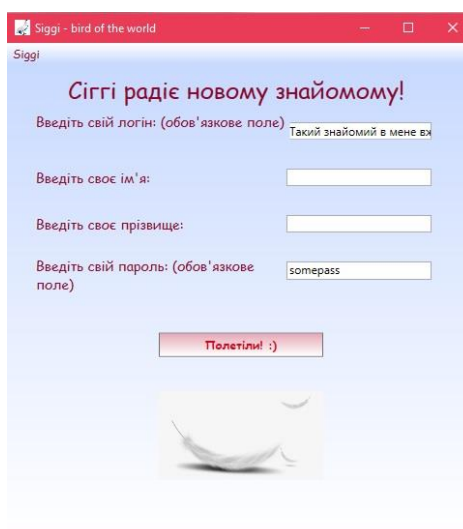


Рисунок 4.5 – Тест реєстрації користувача, який вже є у базі

Для ознайомлення з історією бота-компаньйону створено окреме вікно. На ньому зображена птаха та її розповідь про заснування власної поштової корпорації.

Тест вікна з історією бота-компаньйона.

Кроки відтворення:

- запустити клієнтський застосунок;
- у пункті меню обрати «Про застосунок»;
- обрати підпункт «Трошки про Сіггі»;
- звернути увагу на вікно, що з'явилося.

Очікуваний результат: вікно з історією бота-компаньйону відображається окремо, не закриваючи основне вікно, через яке було викликано дочірнє.

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

Графічна ілюстрація тесту приведена на рис. 4.6.

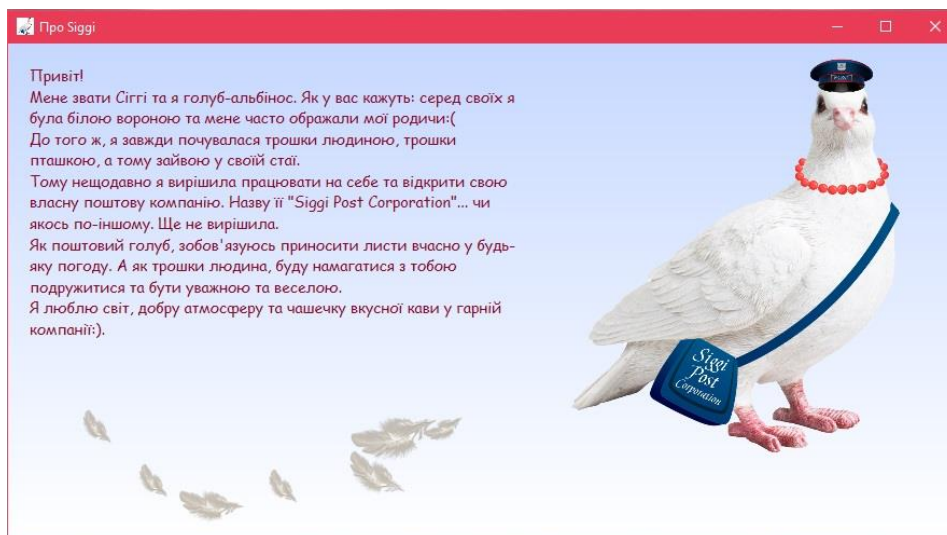


Рисунок 4.6 – Тест вікна з історією бота-компаньйона

Після проходження процесу реєстрації або аутентифікації, користувач потрапляє до вікна переписки з іншими користувачами та взаємодії з ботом-компаньйоном. Кнопка для виведення інформації про користувачів не активна,

тому що не обраний контакт зі списку. Також не активна кнопка надсилання повідомлення.

Тест входу у вікно для переписки з іншими користувачами та взаємодією з ботом-компаньйоном.

Кроки відтворення:

- запустити клієнтський застосунок;
- успішно пройти аутентифікацію;
- звернути увагу на вікно, що з'явилося.

Очікуваний результат: вікно відображається з не активними кнопками для надсилання повідомлення та перегляду профілю користувача, адже ще не обрано контакт.

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

Графічна ілюстрація тесту приведена на рис. 4.7.

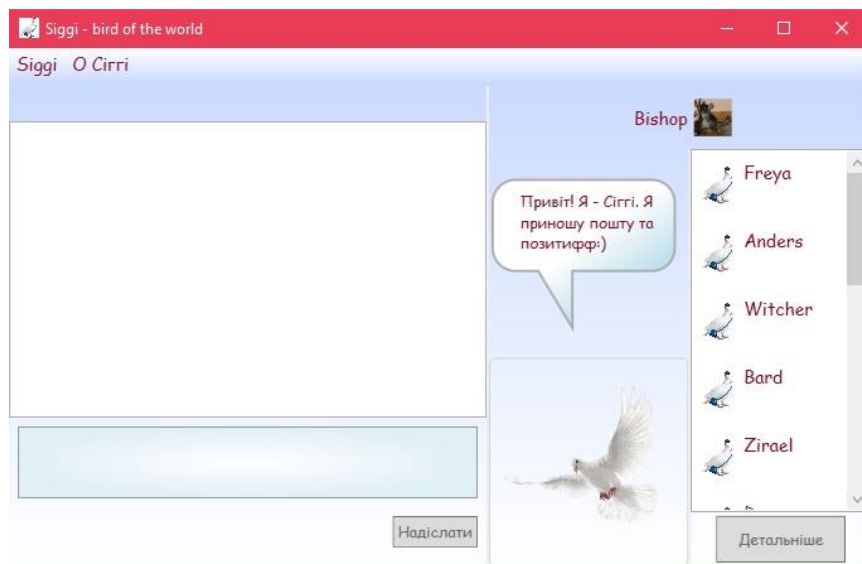


Рисунок 4.7 – Тест входу у вікно для переписки з іншими користувачами та взаємодією з ботом-компаньйоном

Коли користувачеві приходить повідомлення від іншого користувача та переписка з ним не активна, про це сповіщає компаньйон, уточнюючі, від кого саме надійшло повідомлення.

Тест поведінки бота-компаньйона у разі надходження повідомлення від іншого користувача, переписка з яким не активна.

Кроки відтворення:

- запустити клієнтський застосунок;
- успішно пройти аутентифікацію;
- чекати на повідомлення від іншого користувача, переписка з яким не є активною;
- звернути увагу на компаньйона при надходженні повідомлення.

Очікуваний результат: бот-компаньйон повідомляє про надходження нового повідомлення зміною картинки, текстовим написом та воркотінням.

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

Графічна ілюстрація тесту приведена на рис. 4.8.

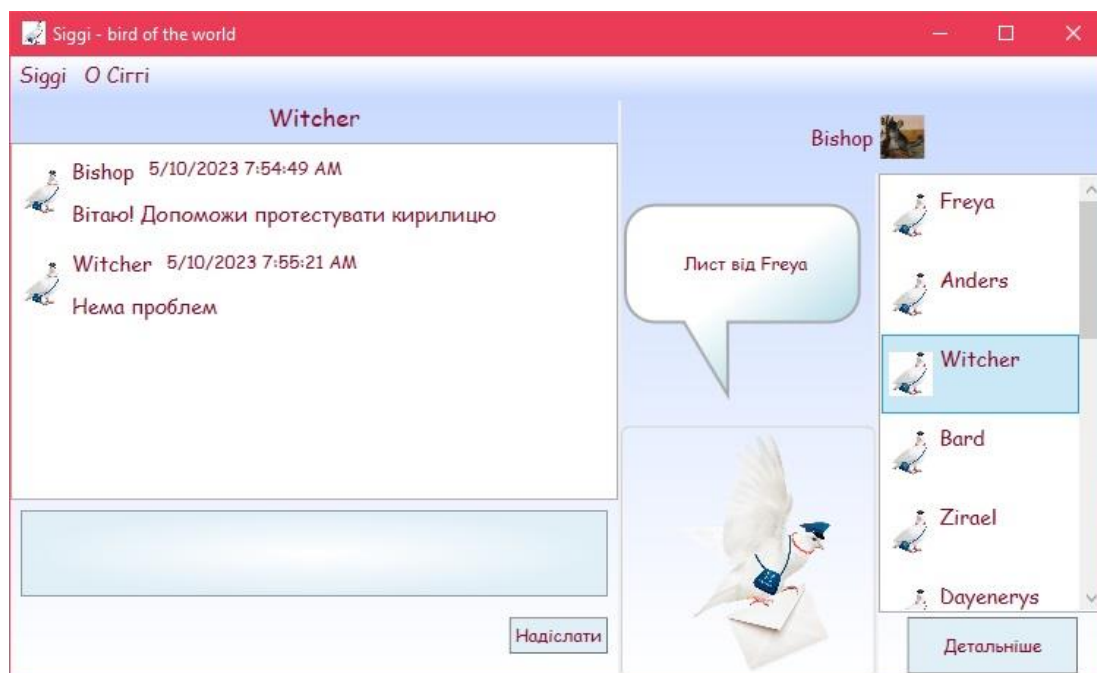


Рисунок 4.8 – Тест поведінки бота-компаньйона у разі надходження повідомлення від іншого користувача, переписка з яким не активна

Час від часу компаньйон просить у користувача приділити йому увагу та задає питання.

Тест поведінки бота-компаньйона у разі спрацювання таймеру для функції запиту від користувача.

Кроки відтворення:

- запустити клієнтський застосунок;
- успішно пройти аутентифікацію;
- чекати на спрацювання таймеру запуску функції запиту від боту.

Функція визивається з урахуванням вірогідності, тому немає точного часу, коли саме це станеться;

- звернути увагу на компаньйона.

Очікуваний результат: бот-компаньйон просить приділити йому увагу воркотінням, зміною картинки та текстовим написом. При натисканні на компаньйона, відкривається вікно запиту. Компаньйона можна підтримати або ні.

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

Графічна ілюстрація тесту приведена на рис. 4.9.

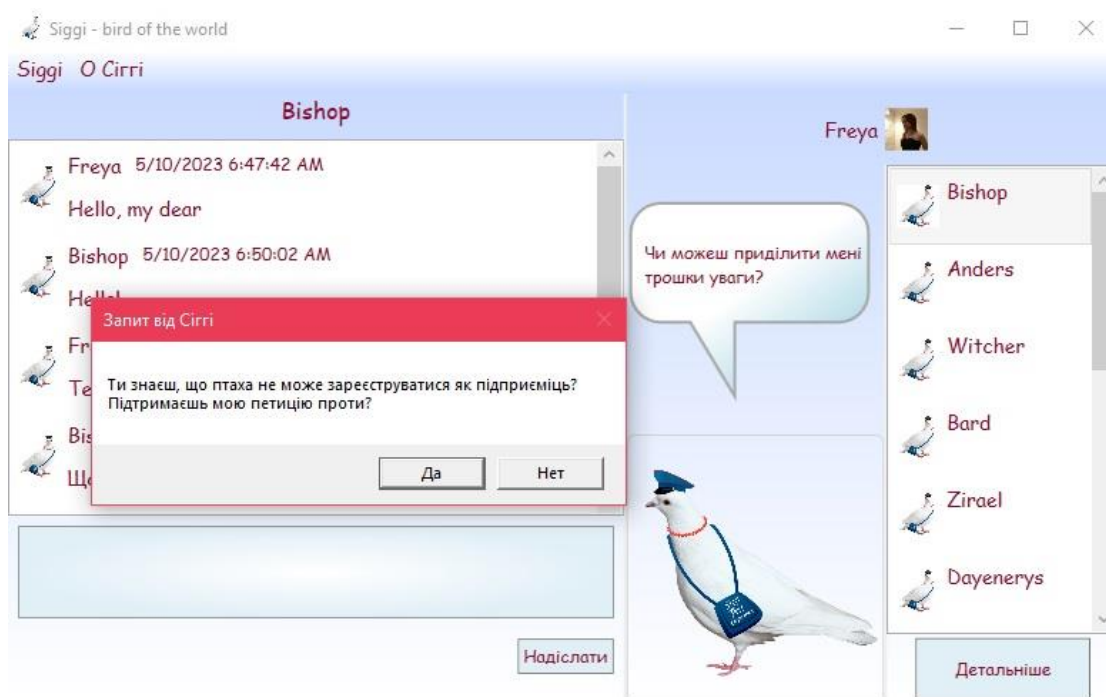


Рисунок 4.9 – Тест поведінки бота-компаньйона у разі спрацювання таймеру для функції запиту від користувача

У разі позитивної відповіді на питання, птаха задоволення (рис. 4.10).



Рисунок 4.10 – Поведінка боту-компаньйону у разі його підтримки

В разі негативної відповіді, компаньйон має інший стан (рис. 4.11):

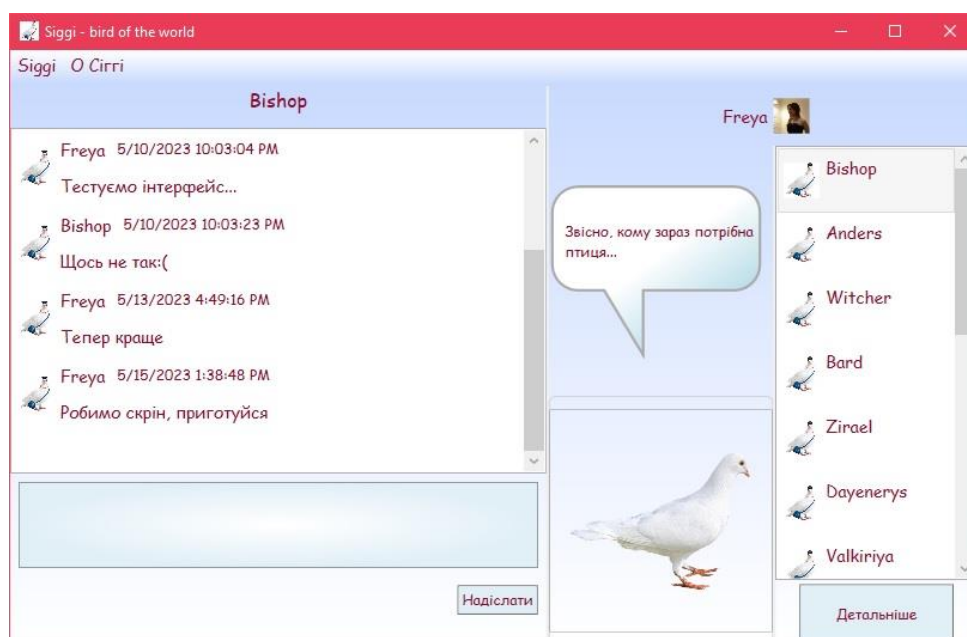


Рисунок 4.11 – Поведінка боту-компаньйону у разі негативної відповіді на запитання

Час від часу спрацьовує таймер запрошення на каву від Сіггі. Про це сповіщає воркотіння та текстове запрошення випити кави разом. Користувач

може натиснути на компаньйона, щоб підтвердити згоду. Після запрошення на каву з компаньйоном, він змінює стан.

Тест поведінки бота-компаньйона у разі спрацювання таймеру для функції запрошення на каву та позитивної відповіді користувача.

Кроки відтворення:

- запустити клієнтський застосунок;
- успішно пройти аутентифікацію;
- чекати на спрацювання таймеру запуску функції запрошення на каву. Функція визивається з урахуванням вірогідності, тому немає точного часу, коли саме це станеться;
- підтвердити згоду натисканням на компаньйона;
- звернути увагу на реакцію компаньйона.

Очікуваний результат: бот-компаньйон просить приділити йому увагу воркотінням, зміною картинки та текстовим написом. При натисканні на компаньйона, змінюється його відображення на голуба, який п'є каву та розмовляє з користувачем.

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

Графічна ілюстрація тесту приведена на рис. 4.12.

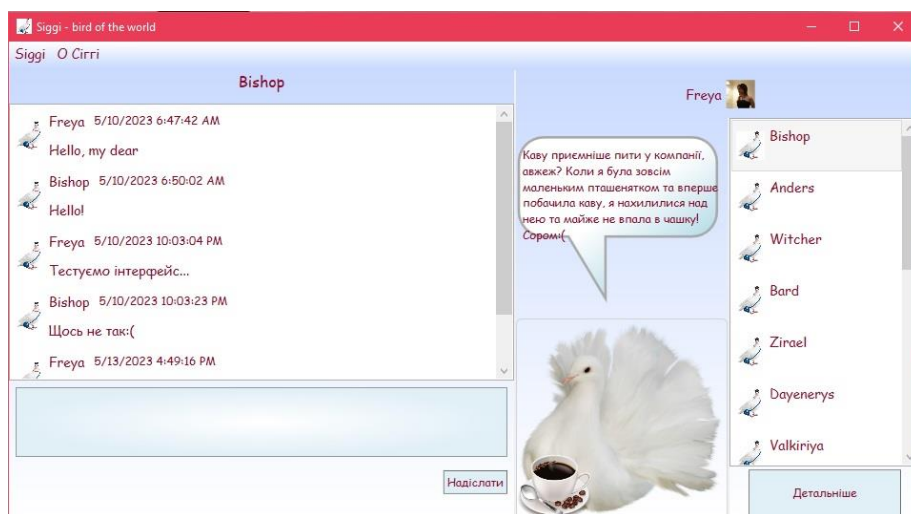


Рисунок 4.12 – Тест поведінки бота-компаньйона у разі згоди випити каву разом

Для редагування інформації про себе, користувач може обрати розділ у пункті меню, після чого з'являється вікно з інформацією. За логікою проектування вікна, воно отримує булевий параметр чи визвано воно командою зміни інформації про поточного користувача.

Тест відображення вікна з інформацією, якщо воно викликається командою зміни інформації про поточного користувача.

Кроки відтворення:

- запустити клієнтський застосунок;
- успішно пройти аутентифікацію;
- у пункті меню «Siggi» обрати підпункт «Персональні дані»;
- звернути увагу на поля вікна.

Очікуваний результат: поля вікна заповнені інформацією про поточного користувача із бази даних. При цьому, всі поля, окрім статусу відносин з ботом-компаньйоном є активними (останнє заповнюється автоматично за досягненням певної границі балів).

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

Графічна ілюстрація тесту приведена на рис. 4.13.

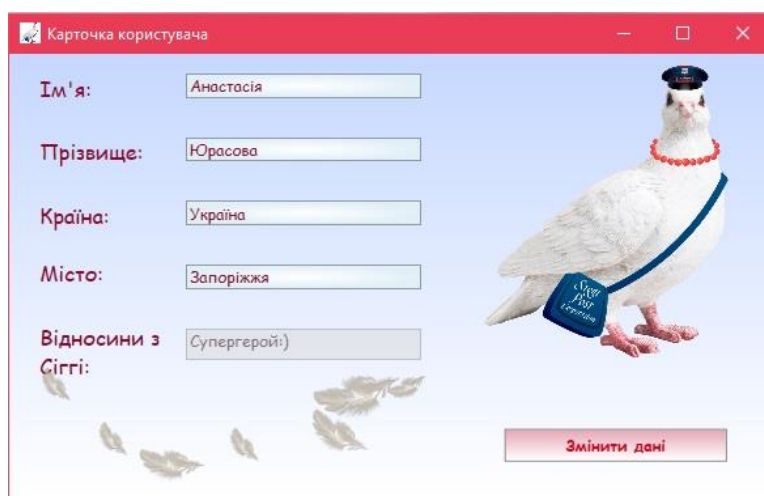


Рисунок 4.13 – Тест відображення вікна з інформацією, якщо воно викликається командою зміни інформації про поточного користувача

У тому разі, коли користувач натискає кнопку відображення інформації про обраного користувача у списку контактів, вікно для показу даних отримує булевий параметр брехні, тож поля вікна не є активними.

Тест відображення вікна з інформацією про обраний контакт зі списку користувачів застосунку.

Кроки відтворення:

- запустити клієнтський застосунок;
- успішно пройти аутентифікацію;
- обрати контакт зі списку праворуч;
- натиснути кнопку «Детальніше»;
- звернути увагу на поля вікна.

Очікуваний результат: поля вікна заповнені інформацією про обраного користувача із бази даних. Всі поля не доступні для редагування.

Фактичний результат: тестування пройшло успішно, очікувані дані підтверджені.

Графічна ілюстрація тесту приведена на рис. 4.14.

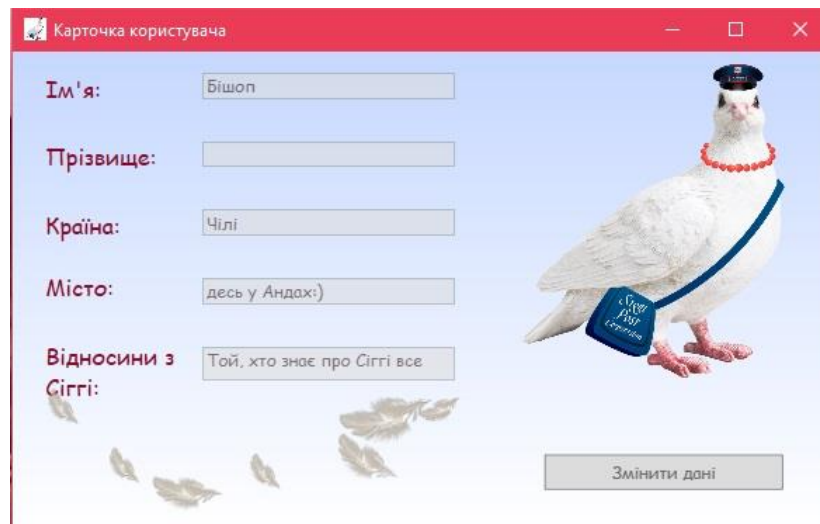


Рисунок 4.14 – Тест відображення вікна з інформацією про обраний контакт зі списку користувачів застосунку

### **4.3 Висновки до розділу 4**

Проведені тести показали коректну роботу застосунку при різних вхідних даних. Вірогідність помилок (наприклад, заповнення некоректними даними користувачем) була проробленою та обробленою. Нові помилки в роботі застосунку не були виявлені. Отже, програмне забезпечення працює коректно.

## ВИСНОВКИ

Популярність месенджерів обумовлена глибиною потребою людства у спілкуванні. Вони роблять можливим процес комунікації на далекій відстані.

Інтерес до застосунків для обміну миттєвими повідомленнями підтверджується популярністю нових рішень, які не завжди суттєво відрізняються від існуючих.

Сучасні чат боти, які містяться у месенджерах, виконують практичні функції відповіді на часті питання, інформування населення, рекламують послуги або товари. Хоча вони є популярними, в них немає функцій спрямованих на психологічне задоволення користувачів.

Для реалізації завдання була обрана мова програмування C# та .NET Framework, а системою управління базами даних обрано MySQL. Для реалізації клієнт-серверної взаємодії використано фреймворк WCF та реалізовано динамічне зв'язування у файлах конфігурацій.

Серверна частина включає у собі DLL-бібліотеку з SQL-запитами до бази даних, дві служби та консольний проект-хост. Служба рівню сесії інкапсулює логіку надання даних з бази про користувачів, їх редагування, а також архіву повідомлень. Крім цього, вона здійснює роботу з базами відносин із ботом-компаньйоном.

Служба рівню одного екземпляру створена для роботи з клієнтами, підключеними до неї. Вона регулює коректне надання ідентифікатору новому користувачу, слідкує за тим, хто зараз підключений до мережі та за допомогою функції зворотного виклику доставляє повідомлення. Також вона відповідає за додання нових повідомлень у базу.

Частина клієнту виконана за архітектурним патерном MVVM фреймворку WPF. Користувач має змогу реєструватися та входити у систему, обмінюватися повідомленнями з іншими користувачами, проглядати інформацію про них та редагувати свої особисті дані. Також він має змогу

прочитати історію компаньйону, реагувати на його запитання та пити з ним каву.

Бот-компаньйон реалізований за допомогою породжуючого патерна “Абстрактна фабрика”.

Створена архітектура застосунку є масштабованою та слабо зв’язаною, тобто, рішення дозволяє змінювати будь-який компонент або додавати новий без впливу на іншу частину програми.

Наукова новизна роботи полягає у створенні бота-компаньйона. Це персоналізована птаха-альбінос, яка має ім’я, свою історію, а саме, що вона вирішила піти від інших голубів та організувала власну поштову службу «Siggi Post Corporation». Вона любить людей та хоче з ними подружитися. Це відображається у тому, що бот-компаньйон періодично звертається до користувача з питаннями та проханням попиту разом кави. Бот-компаньйон виражає емоції та накопичує бали відносин із користувачем. За досягненням певної кількості балів, користувач отримує новий статус відносин із компаньйоном.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Красильнікова О. І. Що таке месенджери, які у них можливості – топ популярних месенджерів [Електроний ресурс] / О. І. Красильнікова. – Режим доступу: <https://vseosvita.ua/library/so-take-mesendzeri-aki-u-nih-mozlivosti-top-popularnih-mesendzeriv-281047.html>.
2. Baudrillard J. The Transparency of Evil: Essays on Extreme Phenomena (Radical Thinkers) / J. Baudrillard. – New York: Verso, 1993. – 174 p.
3. Heim M. The Metaphysics of Virtual Reality / M. Heim. – Oxford: Oxford University Press, 1994. – 208 p.
4. Abubshait A. You Look Human, But Act Like a Machine: Agent Appearance and Behavior Modulate Different Aspects of Human–Robot Interaction [Electronic resource] / A. Abubshait, E. Wiese // National Library of Medicine. – Access mode: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5572356>.
5. Olifer N. Computer Networks, Principles, Technologies, and Protocols for Network Design / N. Olifer V. Olifer. – Hoboken: Wiley, 2006. – 973 p.
6. Sivakumar S. Professional .NET Network Programming / S. Sivakumar, A. Mungale, A. Krowczyk, V. Kumar, C. Nagel, N. Laghari, T. Parker. – London: Peer Information Inc., 2002. – 800 p.
7. Tanenbaum A. Computer Networks / A. Tanenbaum. – Boston: Pearson, 2010. – 960 p.
8. Washburn K. TCP/IP (Computer network protocol), Computer networks / K. Washburn. – Boston: Addison-Wesley, 1993. – 568 p.
9. Max D. 38 Incredible Chatbot Statistics (2023) [Electronic resource] / D. Max. – Access mode: <https://www.netomi.com/chatbot-statistics>.
10. Stroustrup B. The C++ Programming Language, 4th Edition / B. Stroustrup. – Boston: Addison-Wesley Professional, 2013. – 1376 p.
11. Blanchette J. C++ GUI Programming with Qt 4 / J. Blanchette, M. Summerfield. – Hoboken: Prentice Hall, 2008. – 718 p.

12. Troelsen A. Pro C# 5.0 and the .NET Framework 4.5 / A. Troelsen. – New York: Apress, 2012. – 1487 p.
13. Hamilton B. ADO.NET 2.0 Cookbook / B. Hamilton. – Sebastopol: O'Reilly Media, Inc., 2008. – 980 p.
14. Mueller J. Microsoft ADO.NET Entity Framework Step by Step (Step by Step Developer) / J. Mueller. – Unterschleissheim: Microsoft Press, 2013. – 448 p.
15. Lowy J. Programming WCF Services / J. Lowy. – Sebastopol: O'Reilly Media, Inc., 2010. – 910 p.
16. MacDonald M. Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5 / M. MacDonald. – New York: Apress, 2012. – 1111 p.
17. Nathan A., WPF 4.5 / A. Nathan. – Carmel: Sams Publishing, 2013. – 864 p.
18. Forta B. Teach Yourself SQL in 10 Minutes / B. Forta. – Carmel: Sams, 2019. – 306 p.
19. Forta B. MariaDB Crash Course / B. Forta. – Boston: Addison Wesley, 2011. – 290 p.
20. Freeman E. Head First Design Patterns / E. Freeman, B. Bates, K. Sierra, E. Robson. – Sebastopol: O'Reilly Media, Inc., 2004. – 694 p.
21. Будаї А. Дизайн-патерни – просто, як двері [Електроний ресурс] / А. Будаї. – 2012. – 90 с. – Режим доступу: <https://andriybuday.com/book>.
22. Davies A., Async in C# 5.0: Unleash the Power of Async / A. Davies. – Sebastopol: O'Reilly Media, Inc., 2012. – 108 p.

**ДОДАТОК А**  
**Текст програми**

## A.1 Проект DLL-Бібліотеки SiggiRepositoryMySQL

Клас UserInfoDB для роботи з інформацією про користувача.

```
using System;
namespace SiggiRepositoryMySQL.Models
{
    public class UserInfoDB
    {
        private int id;
        private string userName;
        private string lastName;
        private string country;
        private string city;
        public UserInfoDB() { }
        public UserInfoDB(int id, string userName, string lastName, string country, string
city)
        {
            this.id = id;
            this.userName = userName;
            this.lastName = lastName;
            this.country = country;
            this.city = city;
        }
        public int ID
        {
            get { return id; }
            set { id = value; }
        }
        public string UserName
        {
            get { return userName; }
            set { userName = value; }
        }
        public string LastName
        {
            get { return lastName; }
            set { lastName = value; }
        }
        public string Country
        {
            get { return country; }
            set { country = value; }
        }
        public string City
        {
            get { return city; }
            set { city = value; }
        }
    }
}
```

Клас MessageForDB для роботи з моделлю повідомлення.

```
using System;
namespace SiggiRepositoryMySQL.Models
{
    public class MessageForDB
    {
        private int id;
        private string userFrom;
        private string userTo;
```

```

        private long date;
        private string textMessage;
        public MessageForDB() { }
        public MessageForDB(int _id, string _userFrom, string _userTo, long _date, string
        _textMessage)
        {
            id = _id;
            userFrom = _userFrom;
            userTo = _userTo;
            date = _date;
            textMessage = _textMessage;
        }
        public int ID
        {
            get { return id; }
            set { id = value; }
        }
        public string UserFrom
        {
            get { return userFrom; }
            set { userFrom = value; }
        }
        public string UserTo
        {
            get { return userTo; }
            set { userTo = value; }
        }
        public long Date
        {
            get { return date; }
            set { date = value; }
        }
        public string TextMessage
        {
            get { return textMessage; }
            set { textMessage = value; }
        }

        public override string ToString()
        {
            return $"{ID} {UserFrom} {UserTo} {Date} {TextMessage}";
        }
    }
}

```

Клас LoginRepository для роботи з SQL-запитами до таблиці passwords.

```

using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.Common;

namespace SiggiRepositoryMySQL.Repository
{
    public class LoginRepository
    {
        string ConnectionName { get; set; }
        public LoginRepository(string connectionName)
        {
            ConnectionName = connectionName;
        }
        private MySqlConnection GetDbConnection()
        {
            var settings = ConfigurationManager.ConnectionStrings[ConnectionName];

```

```

        var factory = MySqlClientFactory.Instance;
        var connection = factory.CreateConnection();
        connection.ConnectionString = settings.ConnectionString;
        return (MySqlConnection)connection;
    }
    public bool isLogExist(string login)
    {
        bool result;
        string query = $"SELECT * FROM passwords WHERE NickName = '{login}'";
        using (var connection = GetDbConnection())
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;
            var reader = cmd.ExecuteReader();
            if (reader.HasRows)
            {
                result = true;
            }
            else
            {
                result = false;
            }
            reader.Close();
        }
        return result;
    }
    public bool isPassExist(string userName, string password)
    {
        bool result;
        string query = $"SELECT * FROM passwords WHERE NickName = '{userName}' AND
Password = '{password}'";

        using (var connection = GetDbConnection())
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;
            var reader = cmd.ExecuteReader();
            if (reader.HasRows)
            {
                result = true;
            }
            else
            {
                result = false;
            }
            reader.Close();
        }

        return result;
    }
    public int getUserID(string userName, string password)
    {
        int ID = -1;
        string query = $"SELECT id FROM passwords WHERE NickName = '{userName}' AND
Password = '{password}'";

        using (var connection = GetDbConnection())
        {
            connection.Open();
            var cmd = connection.CreateCommand();

```

```

        cmd.CommandText = query;
        cmd.Connection = connection;

        var reader = cmd.ExecuteReader();
        if (reader.HasRows)
        {
            foreach (DbDataRecord row in reader)
            {
                ID = Int32.Parse(row["id"].ToString());
            }
        }
        reader.Close();
    }
    return ID;
}
public int addUser(string name, string password)
{
    string query = $"INSERT INTO passwords (NickName, Password) VALUES('{name}',
'{password}')";
    int number = -1;
    using (var connection = GetDbConnection())
    {
        try
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;
            number = cmd.ExecuteNonQuery();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
    return number;
}
public int deleteUser(string name)
{
    string query = $"Delete FROM passwords WHERE NickName = '{name}'";
    int number = -1;
    using (var connection = GetDbConnection())
    {
        try
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;
            number = cmd.ExecuteNonQuery();
            Console.WriteLine("Успішно видалено: {0}", number);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
    return number;
}
public bool updatePassword(string userName, string password)
{
    int result;
    string query = $"UPDATE passwords SET Password = '{password}' WHERE NickName =
'{userName}'";

```

```

        using (var connection = GetDbConnection())
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;
            result = cmd.ExecuteNonQuery();
        }
        if (result == 1)
        {
            return true;
        }
        else return false;
    }
    public List<string> GetAllUsers(int id)
    {
        int ID = -1;
        List<string> users = new List<string>();
        string query = $"SELECT * FROM passwords";
        using (var connection = GetDbConnection())
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;

            var reader = cmd.ExecuteReader();
            if (reader.HasRows)
            {
                foreach (DbDataRecord row in reader)
                {
                    ID = Int32.Parse(row["id"].ToString());
                    if (ID != 7 && ID != id)
                    {
                        users.Add(row["NickName"].ToString());
                    }
                }
            }
            reader.Close();
        }
        return users;
    }
}
}

```

Клас UserRepository для роботи з SQL-запитами до таблиці users.

```

using MySql.Data.MySqlClient;
using SiggiRepositoryMySQL.Models;
using System;
using System.Configuration;
using System.Data.Common;
namespace SiggiRepositoryMySQL.Repository
{
    public class UserRepository
    {
        string ConnectionName { get; set; }

        public UserRepository(string connectionName)
        {
            ConnectionName = connectionName;
        }
        private MySqlConnection GetDbConnection()
        {
            var settings = ConfigurationManager.ConnectionStrings[ConnectionName];

```

```

var factory = MySqlConnectionFactory.Instance;
var connection = factory.CreateConnection();
connection.ConnectionString = settings.ConnectionString;
return (MySqlConnection)connection;
}
public bool isLogExist(string login)
{
    bool result;
    string query = $"SELECT * FROM users WHERE NickName = '{login}'";

    using (var connection = GetDbConnection())
    {
        connection.Open();
        var cmd = connection.CreateCommand();
        cmd.CommandText = query;
        cmd.Connection = connection;
        var reader = cmd.ExecuteReader();
        if (reader.HasRows)
        {
            result = true;
        }
        else
        {
            result = false;
        }
        reader.Close();
    }
    return result;
}
public int addUser(int userId)
{
    string query = $"INSERT INTO users (userId) VALUES('{userId}')";
    int number = -1;
    using (var connection = GetDbConnection())
    {
        try
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;
            number = cmd.ExecuteNonQuery();
            Console.WriteLine("Успішно додано: {0}", number);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
    return number;
}
public int addUser(int userId, string name, string l1stName)
{
    string query = $"INSERT INTO users (userId, Name, LastName, Country, City)
VALUES('{userId}', '{name}', '{l1stName}', ' ', ' ');";
    int number = -1;
    using (var connection = GetDbConnection())
    {
        try
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;

```

```

        number = cmd.ExecuteNonQuery();
        Console.WriteLine("Успішно додано: {0}", number);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
return number;
}
public short isHasImg(int nickName)
{
    short result = 0;
    string query = $"SELECT Photo FROM users WHERE id = '{nickName}' AND Photo IS
NULL";
    using (var connection = GetDbConnection())
    {
        connection.Open();
        var cmd = connection.CreateCommand();
        cmd.CommandText = query;
        cmd.Connection = connection;

        var reader = cmd.ExecuteReader();
        if (reader.HasRows)
        {
            result++;
        }
        reader.Close();
    }
    return result;
}
public void addImage(byte[] imgByte, string nickName)
{
    string imgString = BitConverter.ToString(imgByte);
    imgString = imgString.Replace("-", "");
    int number = -1;

    string query = $"UPDATE users SET Photo = x'{imgString}' WHERE NickName =
'{nickName}'";
    using (var connection = GetDbConnection())
    {
        try
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;
            number = cmd.ExecuteNonQuery();
            Console.WriteLine("Успішно додано: {0}", number);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}
public string getImage(int nickName)
{
    string imgString = "";
    string query = $"SELECT Photo FROM users WHERE id = '{nickName}'";
    using (var connection = GetDbConnection())
    {
        connection.Open();
        var cmd = connection.CreateCommand();

```

```

        cmd.CommandText = query;
        cmd.Connection = connection;

        var reader = cmd.ExecuteReader();
        if (reader.HasRows)
        {
            foreach (DbDataRecord row in reader)
            {
                byte[] bt = (byte[])row["Photo"];
                imgString = Convert.ToBase64String(bt);
            }
            reader.Close();
        }
        return imgString;
    }
    public UserInfoDB getUserInfo(string login)
    {
        string query = $"SELECT * FROM users WHERE userId = (SELECT id FROM passwords
WHERE NickName = '{login}')";
        UserInfoDB u = null;
        using (var connection = GetDbConnection())
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;

            var reader = cmd.ExecuteReader();
            if (reader.HasRows)
            {
                foreach (DbDataRecord row in reader)
                {
                    u = new UserInfoDB()
                    {
                        ID = int.Parse(row["id"].ToString()),
                        UserName = row["Name"].ToString(),
                        LastName = row["LastName"].ToString(),
                        Country = row["Country"].ToString(),
                        City = row["City"].ToString()
                    };
                }
            }
            reader.Close();
        }
        return u;
    }
    public bool updateUserInfo(UserInfoDB user)
    {
        int result;
        string query = $"UPDATE users SET Name = '{user.UserName}', LastName =
'{user.LastName}', Country = '{user.Country}', City = '{user.City}' WHERE userId =
'{user.ID}'";
        using (var connection = GetDbConnection())
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;
            result = cmd.ExecuteNonQuery();
        }
        if (result == 1)
        {
            return true;
        }
    }

```

```

    }
    else return false;
}
}
}

```

Клас MessageRepository для роботи з SQL-запитами до таблиці messages.

```

using MySql.Data.MySqlClient;
using SiggiRepositoryMySQL.Models;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.Common;

namespace SiggiRepositoryMySQL.Repository
{
    public class MessageRepository
    {
        string ConnectionName { get; set; }
        public MessageRepository(string connectionName)
        {
            ConnectionName = connectionName;
        }
        private MySqlConnection GetDbConnection()
        {
            var settings = ConfigurationManager.ConnectionStrings[ConnectionName];
            var factory = MySqlConnectionFactory.Instance;
            var connection = factory.CreateConnection();
            connection.ConnectionString = settings.ConnectionString;
            return (MySqlConnection)connection;
        }
        public int addMessage(string from, string to, string message)
        {
            DateTime tm = new DateTime();
            tm = DateTime.Now;
            long Date = tm.Ticks;
            string query = $"INSERT INTO messages SET UserFrom = (SELECT id FROM passwords
WHERE NickName = '{from}'), " +
                $"UserTo = (SELECT id FROM passwords WHERE NickName = '{to}'), Date =
'{Date}', TextMessage = '{message}';";
            int number = -1;
            using (var connection = GetDbConnection())
            {
                try
                {
                    connection.Open();
                    var cmd = connection.CreateCommand();
                    cmd.CommandText = query;
                    cmd.Connection = connection;
                    number = cmd.ExecuteNonQuery();
                }
                catch (Exception ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }
            return number;
        }
        public List<MessageForDB> getUsersMessage(string userOwner, string userTo)
        {
            string query = $"SELECT m.id, p1.NickName as p1, p2.NickName as p2, Date,
TextMessage FROM messages m " +
                $"INNER JOIN passwords p1 ON m.UserFrom = p1.id " +
                $"INNER JOIN passwords p2 ON m.UserTo = p2.id " +

```

```

        $"WHERE m.UserFrom = (SELECT id FROM passwords WHERE NickName =
'{userOwner}') " +
        $"AND m.UserTo = (SELECT id FROM passwords WHERE NickName = '{userTo}') " +
        $"OR m.UserTo = (SELECT id FROM passwords WHERE NickName = '{userOwner}') "
+
        $"AND m.UserFrom = (SELECT id FROM passwords WHERE NickName = '{userTo}')";

List<MessageForDB> messages = null;
using (var connection = GetDbConnection())
{
    connection.Open();
    var cmd = connection.CreateCommand();
    cmd.CommandText = query;
    cmd.Connection = connection;
    var reader = cmd.ExecuteReader();
    messages = new List<MessageForDB>();
    if (reader.HasRows)
    {
        foreach (DbDataRecord row in reader)
        {
            messages.Add(new MessageForDB()
            {
                ID = int.Parse(row["id"].ToString()),
                UserFrom = row["p1"].ToString(),
                UserTo = row["p2"].ToString(),
                Date = long.Parse((row["Date"].ToString())),
                TextMessage = row["TextMessage"].ToString(),
            });
        }
    }
    else
    {
        Console.WriteLine("***NO***");
    }
    reader.Close();
}
return messages;
}
}
}

```

Клас RelationshipRepository для роботи з SQL-запитами до таблиць relationship та reldescript.

```

using MySql.Data.MySqlClient;
using System;
using System.Configuration;
using System.Data.Common;
namespace SiggiRepositoryMySQL.Repository
{
    public class RelationshipRepository
    {
        string ConnectionName { get; set; }
        public RelationshipRepository(string connectionName)
        {
            ConnectionName = connectionName;
        }
        private MySqlConnection GetDbConnection()
        {
            var settings = ConfigurationManager.ConnectionStrings[ConnectionName];
            var factory = MySqlConnectionFactory.Instance;
            var connection = factory.CreateConnection();
            connection.ConnectionString = settings.ConnectionString;
            return (MySqlConnection)connection;
        }
    }
}

```

```

    }
    public int setNewUser(string login)
    {
        DateTime tm = new DateTime();
        tm = DateTime.Now;
        long Date = tm.Ticks;
        string query = $"INSERT INTO relationship SET userId = (SELECT id FROM passwords
WHERE NickName = '{login}'), Points = '1', LastWisited = '{Date}';";
        int number = -1;
        using (var connection = GetDbConnection())
        {
            try
            {
                connection.Open();
                var cmd = connection.CreateCommand();
                cmd.CommandText = query;
                cmd.Connection = connection;
                number = cmd.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
        return number;
    }
    public int getUsersCoin(string login)
    {
        string query = $"SELECT Points FROM relationship WHERE userId = (SELECT id FROM
passwords WHERE NickName = '{login}')";
        int result = -1;
        using (var connection = GetDbConnection())
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;

            var reader = cmd.ExecuteReader();
            if (reader.HasRows)
            {
                foreach (DbDataRecord row in reader)
                {
                    result = int.Parse(row["Points"].ToString());
                }
            }
            reader.Close();
        }
        return result;
    }
    public string getRelStatus(int coin)
    {
        int coins = coin / 100 + 1;
        string query = $"SELECT Description FROM reldescript WHERE id = '{coins}'";
        string result = "";

        using (var connection = GetDbConnection())
        {
            connection.Open();
            var cmd = connection.CreateCommand();
            cmd.CommandText = query;
            cmd.Connection = connection;
            var reader = cmd.ExecuteReader();
            if (reader.HasRows)

```

```

        {
            foreach (DbDataRecord row in reader)
            {
                result = row["Description"].ToString();
            }
        }
        reader.Close();
    }
    return result;
}
public long getLastVisit(int id)
{
    string query = $"SELECT LastWisited FROM relationship WHERE userId = '{id}'";
    long result = -1;
    using (var connection = GetDbConnection())
    {
        connection.Open();
        var cmd = connection.CreateCommand();
        cmd.CommandText = query;
        cmd.Connection = connection;

        var reader = cmd.ExecuteReader();
        if (reader.HasRows)
        {
            foreach (DbDataRecord row in reader)
            {
                result = long.Parse((row["LastWisited"].ToString()));
            }
        }
        reader.Close();
    }
    return result;
}
public bool setLastVisit(int id)
{
    int result;
    DateTime tm = new DateTime();
    tm = DateTime.Now;
    long Date = tm.Ticks;
    string query = $"UPDATE relationship SET LastWisited = '{Date}' WHERE userId = '{id}'";

    using (var connection = GetDbConnection())
    {
        connection.Open();
        var cmd = connection.CreateCommand();
        cmd.CommandText = query;
        cmd.Connection = connection;
        result = cmd.ExecuteNonQuery();
    }
    if (result == 1)
    {
        return true;
    }
    else return false;
}

public bool setCoin(int id, int new_coin)
{
    int result;
    string query = $"UPDATE relationship SET Points = '{new_coin}' WHERE userId = '{id}'";

    using (var connection = GetDbConnection())

```

```

    {
        connection.Open();
        var cmd = connection.CreateCommand();
        cmd.CommandText = query;
        cmd.Connection = connection;
        result = cmd.ExecuteNonQuery();
    }
    if (result == 1)
    {
        return true;
    }
    else return false;
}
}
}

```

Зміст файлу App.config для налаштування доступу до бази даних.

```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false"
/>
  </configSections>
  <entityFramework>
    <providers>
      <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
      <provider invariantName="MySQL.Data.MySqlClient"
type="MySQL.Data.MySqlClient.MySqlProviderServices, MySQL.Data.EntityFramework,
Version=8.0.33.0, Culture=neutral, PublicKeyToken=c5687fc88969c44d">
    </provider></providers>
    </entityFramework>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="System.Runtime.CompilerServices.Unsafe"
publicKeyToken="b03f5f7f11d50a3a" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-6.0.0.0" newVersion="6.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Memory" publicKeyToken="cc7b13ffcd2ddd51"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-4.0.1.2" newVersion="4.0.1.2" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>

```

## A.2 Проект Сервісу SiggisServiceLibrary

Інтерфейс служби IServiceUserInfo та серіалізуємий клас UserInfoServer.

```

[ServiceContract]
public interface IServiceUserInfo
{
    [OperationContract]
    bool isLogExist(string login);
}

```

```

[OperationContract]
bool isPassExist(string login, string pass);

[OperationContract]
int getUserId(string login, string pass);

[OperationContract]
bool Registration(string login, string pass, string name, string lstName);

[OperationContract]
string getIconUser(int user);

[OperationContract]
UserInfoServer getUserInfo(string login);

[OperationContract]
bool updateUserInfo(UserInfoServer user);

[OperationContract]
List<string> getAllUsers(int id);

[OperationContract]
List<MessageServer> getMessage(string userOwner, string userTo);
}

[DataContract]
public class UserInfoServer
{
    private int id;
    private string userName;
    private string lastName;
    private string country;
    private string city;
    private string role;
    public UserInfoServer() {}
    public UserInfoServer(int _id, string _userName, string _lastName, string _country,
string _city, string _role)
    {
        id = _id;
        userName = _userName;
        lastName = _lastName;
        country = _country;
        city = _city;
        role = _role;
    }

    [DataMember]
    public int ID
    {
        get { return id; }
        set { id = value; }
    }

    [DataMember]
    public string UserName
    {
        get { return userName; }
        set { userName = value; }
    }

    [DataMember]
    public string LastName
    {
        get { return lastName; }
        set { lastName = value; }
    }
}

```

```

[DataMember]
public string Country
{
    get { return country; }
    set { country = value; }
}
[DataMember]
public string City
{
    get { return city; }
    set { city = value; }
}
[DataMember]
public string Role
{
    get { return role; }
    set { role = value; }
}
}
}

```

Клас ServiceUserInfo, який реалізує інтерфейс служби IServiceUserInfo.

```

public class ServiceUserInfo : IServiceUserInfo
{
    public bool isLogExist(string login)
    {
        bool result = false;
        try
        {
            LoginRepository loginRepo = new LoginRepository("siggi_local");
            result = loginRepo.isLogExist(login);
        }
        catch(Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        return result;
    }
    public bool isPassExist(string login, string pass)
    {
        bool result = false;
        pass = getAllpass(login, pass);
        try
        {
            LoginRepository loginRepo = new LoginRepository("siggi_local");
            result = loginRepo.isPassExist(login, pass);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        return result;
    }
    public int getUserId(string login, string pass)
    {
        int id = -1;
        pass = getAllpass(login, pass);
        try
        {
            LoginRepository loginRepo = new LoginRepository("siggi_local");
            id = loginRepo.getUserId(login, pass);
            bool coin = checkCoin(id, login);
        }
        catch (Exception ex)
    }
}

```

```

        {
            Console.WriteLine(ex.Message);
        }
        return id;
    }
    public bool Registration(string login, string pass, string name, string lstName)
    {
        bool r = false;
        int result = -1;
        string new_pass = getAllpass(login, pass);
        UserRepository u = new UserRepository("siggi_local");
        LoginRepository l = new LoginRepository("siggi_local");
        RelationshipRepository relationshipRepository = new
RelationshipRepository("siggi_local");
        result = l.addUser(login, new_pass);
        if (result != -1)
        {
            int id = l.getUserID(login, new_pass);
            if (id != -1)
            {
                Console.WriteLine("name: {0} {1}", name, lstName);
                r = true;
                result = u.addUser(id, name, lstName);
                int rel = relationshipRepository.setNewUser(login);
            }
        }
        else
        {
            result = l.deleteUser(login);
        }

        return r;
    }
    public string getIconUser(int user)
    {
        string imgString = "";
        short result = 0;

        var userRepo = new UserRepository("siggi_local");

        result = userRepo.isHasImg(user);
        if (result == 1)
        {
            return imgString;
        }
        else
        {
            imgString = userRepo.getImage(user);
        }
        return imgString;
    }
    public UserInfoServer getUserInfo(string login)
    {
        UserInfoServer userInfoServer = new UserInfoServer();
        UserRepository userRepo = new UserRepository("siggi_local");
        RelationshipRepository relationshipRepository = new
RelationshipRepository("siggi_local");
        try
        {
            UserInfoDB u = userRepo.getUserInfo(login);
            if (u != null)
            {
                userInfoServer.ID = u.ID;
                userInfoServer.UserName = u.UserName;
            }
        }
    }

```

```

        userInfoServer.LastName = u.LastName;
        userInfoServer.Country = u.Country;
        userInfoServer.City = u.City;
    }
}
catch(Exception ex)
{
    Console.WriteLine(ex.Message);
}
try
{
    int count = relationshipRepository.getUsersCoin(login);
    string result = relationshipRepository.getRelStatus(count);
    if (result != "")
    {
        userInfoServer.Role = result;
    }
}
catch(Exception ex)
{
    Console.WriteLine(ex.Message);
}
return userInfoServer;
}
public bool updateUserInfo(UserInfoServer user)
{
    bool result = false;
    UserRepository userRepo = new UserRepository("siggi_local");
    UserInfoDB u = new UserInfoDB(user.ID, user.UserName, user.LastName,
user.Country, user.City);
    try
    {
        result = userRepo.updateUserInfo(u);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return result;
}
public List<string> getAllUsers(int id)
{
    List<string> users = new List<string>();
    LoginRepository loginRepo = new LoginRepository("siggi_local");
    try
    {
        users = loginRepo.GetAllUsers(id);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return users;
}
public List<MessageServer> getMessage(string userOwner, string userTo)
{
    List<MessageServer> messageServers = new List<MessageServer>();
    MessageRepository m = new MessageRepository("siggi_local");

    try
    {
        List<MessageForDB> messages = m.getUsersMessage(userOwner, userTo);
        foreach (var item in messages)
        {

```

```

        messageServers.Add(new MessageServer()
        {
            ID = item.ID,
            UserFrom = item.UserFrom,
            UserTo = item.UserTo,
            Date = new DateTime(item.Date),
            TextMessage = item.TextMessage
        });
    }
}
catch(Exception ex)
{
    Console.WriteLine(ex.Message);
}
return messageServers;
}
private bool checkCoin(int id, string login)
{
    bool result = false;
    RelationshipRepository relationshipRepository = new
RelationshipRepository("siggi_local");
    try
    {
        long lastDate = relationshipRepository.getLastVisit(id);
        DateTime last = new DateTime(lastDate);
        DateTime current = new DateTime();
        current = DateTime.Now;

        TimeSpan interval = current - last;

        if (interval.Days > 0)
        {
            int new_coin = relationshipRepository.getUsersCoin(login);
            new_coin++;
            bool r1 = relationshipRepository.setCoin(id, new_coin);
            bool r2 = relationshipRepository.setLastVisit(id);
        }
    }
    catch(Exception ex)
    {
        Console.WriteLine(ex.Message);
    }

    return result;
}
private string getAllpass(string login, string pass)
{
    return pass += GetHashMD5(login);
}

private static string GetHashMD5(string s)
{
    byte[] bytes = Encoding.Unicode.GetBytes(s);
    MD5CryptoServiceProvider CSP = new MD5CryptoServiceProvider();
    byte[] byteHash = CSP.ComputeHash(bytes);
    string hash = string.Empty;
    foreach (byte b in byteHash)
        hash += string.Format("{0:x2}", b);
    return hash;
}
}
}

```

Клас `UserService`, який представляє модель користувача служби `IServiceMessage`.

```
using System;
using System.ServiceModel;
namespace SiggiServiceLibrary.Models
{
    public class UserService
    {
        private long id;
        private string userName;
        private OperationContext operationContext;
        public UserService() { }
        public UserService(long _id, string _userName, OperationContext contex)
        {
            id = _id;
            userName = _userName;
            operationContext = contex;
        }
        public long ID
        {
            get { return id; }
            set { id = value; }
        }
        public string UserName
        {
            get { return userName; }
            set { userName = value; }
        }
        public OperationContext Context
        {
            get { return operationContext; }
            set { operationContext = value; }
        }
    }
}
```

Статичний клас `CountUsers`, який відповідає за коректність надання ідентифікатору користувачеві служби.

```
using System;

namespace SiggiServiceLibrary.Models
{
    public static class CountUsers
    {
        private static long count = 0;

        public static long getId()
        {
            return count;
        }
        public static void setId()
        {
            count++;
        }

        public static void setNull()
        {
            count = 0;
        }
    }
}
```

}

Клас ListUsers, який містить список користувачів служби та роботу з НИМИ.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.ServiceModel;
namespace SiggiServiceLibrary.Models
{
    public class ListUsers
    {
        public List<UserService> users { get; set; }
        public ListUsers()
        {
            users = new List<UserService>();
        }
        private void add(long id, string nickName, OperationContext contex)
        {
            users.Add(new UserService(id, nickName, contex));
        }
        private void restart()
        {
            users.Clear();
            CountUsers.setNull();
        }
        public bool checkUser(string nickName)
        {
            bool result = true;
            short count = 0;
            foreach (var user in users)
            {
                if (count == 1)
                {
                    result = false;
                    break;
                }
                if (user.UserName == nickName)
                    count++;
            }
            return result;
        }
        public long addUser(string nickName)
        {
            long id = -1;
            bool result = checkUser(nickName);
            if (result == true)
            {
                CountUsers.setId();
                id = CountUsers.getId();
                add(id, nickName, OperationContext.Current);
            }

            return id;
        }
        public bool deleteUser(long id)
        {
            bool result = false;
            UserService user = users.FirstOrDefault(u => u.ID == id);
            if (user != null)
            {
                users.Remove(user);
                result = true;
            }
        }
    }
}
```

```

    }
    return result;
}
public bool notFull()
{
    bool result = true;
    long id = CountUsers.getId();

    if (id + 50 > long.MaxValue)
    {
        result = false;
        restart();
    }

    return result;
}
public void deleteUserService(long id)
{
    UserService user = users.FirstOrDefault(u => u.ID == id);
    if (user != null)
    {
        users.Remove(user);
    }
}
}
}

```

Інтерфейс служби IServiceMessage, інтерфейс функції зворотного виклику

IServiceMessageCallback та серіалізуємий клас MessageServer.

```

using System;
using System.Runtime.Serialization;
using System.ServiceModel;
namespace SiggiServiceLibrary
{
    [ServiceContract(CallbackContract = typeof(IServiceMessageCallback))]
    public interface IServiceMessage
    {
        [OperationContract]
        long getConnect(string login);

        [OperationContract]
        void Disconnect(long id);

        [OperationContract(IsOneWay = true)]
        void SendMessage(MessageServer ms, string nick);
    }

    public interface IServiceMessageCallback
    {
        [OperationContract(IsOneWay = true)]
        void SendMessageCallback(MessageServer m);
    }

    [DataContract]
    public class MessageServer
    {
        private int id;
        private string userFrom;
        private string userTo;
        private DateTime date;
        private string textMessage;

        public MessageServer() {

```

```

        date = new DateTime();
    }
    public MessageServer(int _id, string _userFrom, string _userTo, DateTime _date,
string _textMessage)
    {
        id = _id;
        userFrom = _userFrom;
        userTo = _userTo;
        date = _date;
        textMessage = _textMessage;
    }
    [DataMember]
    public int ID
    {
        get { return id; }
        set { id = value; }
    }
    [DataMember]
    public string UserFrom
    {
        get { return userFrom; }
        set { userFrom = value; }
    }
    [DataMember]
    public string UserTo
    {
        get { return userTo; }
        set { userTo = value; }
    }
    [DataMember]
    public DateTime Date
    {
        get { return date; }
        set { date = value; }
    }
    [DataMember]
    public string TextMessage
    {
        get { return textMessage; }
        set { textMessage = value; }
    }
    }
}

```

Клас ServiceMessage, який реалізує інтерфейс служби IServiceMessage.

```

using SiggiRepositoryMySQL.Repository;
using SiggiServiceLibrary.Models;
using System;
using System.Linq;
using System.ServiceModel;
namespace SiggiServiceLibrary
{
    [ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]
    public class ServiceMessage : IServiceMessage
    {
        private ListUsers listUsers = new ListUsers();

        public long getConnect(string login)
        {
            long id = -1;
            id = listUsers.addUser(login);
            Console.WriteLine($"The user with id - {id} and login {login} has connected");

            return id;
        }
    }
}

```

```

    }

    public void Disconnect(long id)
    {
        listUsers.deleteUserService(id);
        Console.WriteLine($"The user with id - {id} disconnected");
    }

    public void SendMessage(MessageServer ms, string nick)
    {
        UserService user = listUsers.users.FirstOrDefault(u => u.UserName == nick);

        if (user != null)
        {
            foreach (var item in listUsers.users)
            {
                if (item.UserName == nick)
                {
                    try
                    {
                        item.Context.GetCallbackChannel<IServiceMessageCallback>().SendMessageCallback(ms);
                    }
                    catch (Exception ex) { Console.WriteLine(ex.ToString()); }
                }
            }
        }
        int r = 0;
        MessageRepository mRepo = new MessageRepository("siggi_local");
        r = mRepo.addMessage(ms.UserFrom, ms.UserTo, ms.TextMessage);
    }

    public IServiceMessageCallback Callback
    {
        get
        {
            return
                OperationContext.Current.GetCallbackChannel<IServiceMessageCallback>();
        }
    }
}

```

Зміст файлу App.config для налаштування доступу до служби.

```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <appSettings>
    <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" />
  </appSettings>
  <system.web>
    <compilation debug="true" />
  </system.web>
  <system.serviceModel>
    <services>
      <service name="SiggiServiceLibrary.ServiceUserInfo">
        <endpoint address="" binding="wsHttpBinding"
contract="SiggiServiceLibrary.IServiceUserInfo">
          <identity>
            <dns value="localhost"/>
          </identity>
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange"/>
      </service>
    </services>
  </system.serviceModel>
</configuration>

```

```

        <baseAddresses>
            <add
baseAddress="http://localhost:8733/Design_Time_Addresses/SiggiServiceLibrary/ServiceUserInfo
"/>
            </baseAddresses>
        </host>
    </service>
    <service name="SiggiServiceLibrary.ServiceMessage">
        <endpoint address="" binding="wsDualHttpBinding"
contract="SiggiServiceLibrary.IServiceMessage">
            <identity>
                <dns value="localhost"/>
            </identity>
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange"/>
    </host>
    <baseAddresses>
        <add
baseAddress="http://localhost:8734/Design_Time_Addresses/SiggiServiceLibrary/ServiceMessage/
"/>
    </baseAddresses>
</host>
</service>
</services>
<behaviors>
    <serviceBehaviors>
        <behavior>
            <serviceMetadata httpGetEnabled="True" httpsGetEnabled="True" />
            <serviceDebug includeExceptionDetailInFaults="True" />
        </behavior>
    </serviceBehaviors>
</behaviors>
</system.serviceModel>
<runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
        <dependentAssembly>
            <assemblyIdentity name="System.Runtime.CompilerServices.Unsafe"
publicKeyToken="b03f5f7f11d50a3a" culture="neutral" />
            <bindingRedirect oldVersion="0.0.0.0-6.0.0.0" newVersion="6.0.0.0" />
        </dependentAssembly>
        <dependentAssembly>
            <assemblyIdentity name="System.Memory" publicKeyToken="cc7b13ffcd2ddd51"
culture="neutral" />
            <bindingRedirect oldVersion="0.0.0.0-4.0.1.2" newVersion="4.0.1.2" />
        </dependentAssembly>
    </assemblyBinding>
</runtime>
</configuration>

```

### A.3 Проект хосту ConsoleSiggiHost

Клас HostMessage, який реалізує хост служби IServiceMessage.

```

using System;
using System.ServiceModel;

namespace ConsoleSiggiHost.Hosts
{
    class HostMessage
    {

```

```

internal static ServiceHost serviceHost = null;
internal static void StartService()
{
    serviceHost = new ServiceHost(typeof(SiggiServiceLibrary.ServiceMessage));
    serviceHost.Open();
}
internal static void StopService()
{
    if (serviceHost.State != CommunicationState.Closed)
    {
        serviceHost.Close();
    }
}
internal static void InfoService()
{
    Console.WriteLine();
    Console.WriteLine("***** Host info: *****");

    foreach (System.ServiceModel.Description.ServiceEndpoint se in
serviceHost.Description.Endpoints)
    {
        Console.WriteLine("Adress: {0}", se.Address);
        Console.WriteLine("Binding: {0}", se.Binding.Name);
        Console.WriteLine("Contract: {0}", se.Contract.Name);
        Console.WriteLine("Contract operation: {0}", se.Contract.ContractType);
        Console.WriteLine("Contract operation: {0}",
se.Contract.CallbackContractType);
        Console.WriteLine();
    }
}
}
}

```

Клас HostUserInfo, який реалізує хост служби IServiceUserInfo.

```

using System;
using System.ServiceModel;
namespace ConsoleSiggiHost.Hosts
{
    class HostUserInfo
    {
        internal static ServiceHost serviceHost = null;
        internal static void StartService()
        {
            serviceHost = new ServiceHost(typeof(SiggiServiceLibrary.ServiceUserInfo));
            serviceHost.Open();
        }
        internal static void StopService()
        {
            if (serviceHost.State != CommunicationState.Closed)
            {
                serviceHost.Close();
            }
        }

        internal static void InfoService()
        {
            Console.WriteLine();
            Console.WriteLine("***** Host info: *****");
            foreach (System.ServiceModel.Description.ServiceEndpoint se in
serviceHost.Description.Endpoints)
            {
                Console.WriteLine("Adress: {0}", se.Address);
                Console.WriteLine("Binding: {0}", se.Binding.Name);
                Console.WriteLine("Contract: {0}", se.Contract.Name);
            }
        }
    }
}

```

```

        Console.WriteLine("Contract operation: {0}", se.Contract.ContractType);
        Console.WriteLine("Contract operation: {0}",
se.Contract.CallbackContractType);
        Console.WriteLine();
    }
}
}
}

```

### Клас Program хосту.

```

using ConsoleSiggiHost.Hosts;
using System;
using System.Text;
namespace ConsoleSiggiHost
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("**** SIGGI! ****");

            try
            {
                Console.OutputEncoding = UTF8Encoding.UTF8;
                Console.WriteLine(new string('*', 20));
                Console.WriteLine("**** HOST MESSAGE ****");
                HostMessage.StartService();
                HostMessage.InfoService();
                Console.WriteLine("**** HOST USER INFO ****");
                HostUserInfo.StartService();
                HostUserInfo.InfoService();
                Console.WriteLine();
                Console.WriteLine("Wcf host is running... press any key to stop");
                Console.ReadLine();
                HostMessage.StopService();
                HostUserInfo.StopService();
            }
            catch(Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
}

```

Зміст файлу App.config який містить метадані для підключення клієнтів та строку підключення до бази даних.

```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
  <connectionStrings>
    <add name="siggi_local"
connectionString="datasource=127.0.0.1;port=3306;username=root;password=;charset=utf8;databa
se=siggi;Convert Zero Datetime=True" providerName="MySql.Data.MySqlClient"/>
  </connectionStrings>
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior name="mexBehSiggi">
          <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true"/>

```

```

        <serviceDebug includeExceptionDetailInFaults="true"/>
    </behavior>
</serviceBehaviors>
</behaviors>
<services>
    <service name="SiggiServiceLibrary.ServiceMessage"
behaviorConfiguration="mexBehSiggi">
        <endpoint address="" binding="netTcpBinding"
contract="SiggiServiceLibrary.IServiceMessage"/>
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>
        <host>
            <baseAddresses>
                <add baseAddress="http://localhost:8090/" />
                <add baseAddress="net.tcp://localhost:8310/" />
            </baseAddresses>
        </host>
    </service>
    <service name="SiggiServiceLibrary.ServiceUserInfo"
behaviorConfiguration="mexBehSiggi">
        <endpoint address="" binding="netTcpBinding"
contract="SiggiServiceLibrary.IServiceUserInfo"/>
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>
        <host>
            <baseAddresses>
                <add baseAddress="http://localhost:8091/" />
                <add baseAddress="net.tcp://localhost:8311/" />
            </baseAddresses>
        </host>
    </service>
</services>
</system.serviceModel>
</configuration>

```

## A.4 Проект клієнтського застосунку Siggi

Клас UserInvite, модель-представлення користувача при реєстрації.

```

using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;
namespace Siggi.Models
{
    public class UserInvite : INotifyPropertyChanged
    {
        private int id;
        private string nick;
        private string userName;
        private string lastName;
        private string password;
        public UserInvite() { }
        public string Nick
        {
            get { return nick; }
            set
            {
                nick = value;
                NotifyPropertyChanged("Nick");
            }
        }
        public string UserName

```

```

{
    get { return userName; }
    set
    {
        userName = value;
        NotifyPropertyChanged("UserName");
    }
}
public string LastName
{
    get { return lastName; }
    set
    {
        lastName = value;
        NotifyPropertyChanged("LastName");
    }
}
public string Password
{
    get { return password; }
    set
    {
        password = value;
        NotifyPropertyChanged("Password");
    }
}
public event PropertyChangedEventHandler PropertyChanged;
public void NotifyPropertyChanged([CallerMemberName] string propertyName = "")
{
    if (PropertyChanged != null)
        PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
}
}
}

```

Клас UserInfo, модель-представлення інформації про користувача.

```

using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;
namespace Siggi.Models
{
    public class UserInfo : INotifyPropertyChanged
    {
        private int id;
        private string userName;
        private string lastName;
        private string country;
        private string city;
        private string role;
        public UserInfo() { }
        public UserInfo(int id)
        {
            this.id = id;
        }
        public int ID
        {
            get { return id; }
            set { id = value; }
        }
        public string UserName
        {
            get { return userName; }
            set
            {

```

```

        userName = value;
        NotifyPropertyChanged("UserName");
    }
}
public string LastName
{
    get { return lastName; }
    set
    {
        lastName = value;
        NotifyPropertyChanged("LastName");
    }
}
public string Country
{
    get { return country; }
    set
    {
        country = value;
        NotifyPropertyChanged("Country");
    }
}
public string City
{
    get { return city; }
    set
    {
        city = value;
        NotifyPropertyChanged("City");
    }
}
public string Role
{
    get { return role; }
    set
    {
        role = value;
        NotifyPropertyChanged("Role");
    }
}
public event PropertyChangedEventHandler PropertyChanged;
public void NotifyPropertyChanged([CallerMemberName] string propertyName = "")
{
    if (PropertyChanged != null)
        PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
}
}
}

```

Клас UserOfSiggi, модель-представлення користувача у вікні SiggiWindow.

```

using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows.Media.Imaging;
namespace Siggi.Models
{
    public class UserOfSiggi : INotifyPropertyChanged
    {
        private int id;
        private string userPasword;
        private string nickName;
    }
}

```

```

private BitmapImage pathImg;

public UserOfSiggi()
{
    id = -1;
    userPasword = "";
    nickName = "";
}
public int ID
{
    get { return id; }
    set { id = value; }
}
public string UserPasword
{
    get { return userPasword; }
    set { userPasword = value; }
}
public string NickName
{
    get { return nickName; }
    set
    {
        nickName = value;
        OnPropertyChanged("NickName");
    }
}
public BitmapImage PathImg
{
    get { return pathImg; }
    set
    {
        pathImg = value;
        OnPropertyChanged("PathImg");
    }
}
public event PropertyChangedEventHandler PropertyChanged;
public void OnPropertyChanged([CallerMemberName] string propertyName = "")
{
    if (PropertyChanged != null)
        PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
}
}
}

```

Клас Message, модель-представлення повідомлення.

```

using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;
namespace Siggi.Models
{
    public class Message : INotifyPropertyChanged
    {
        private int id;
        private string userFrom;
        private string userTo;
        private DateTime date;
        private string textMessage;

        public Message()
        {
            date = new DateTime();
        }
        public int ID

```

```

    {
        get { return id; }
        set
        {
            id = value;
        }
    }
    public string UserFrom
    {
        get { return userFrom; }
        set
        {
            userFrom = value;
            OnPropertyChanged("UserFrom");
        }
    }
    public string UserTo
    {
        get { return userTo; }
        set { userTo = value; }
    }
    public DateTime Date
    {
        get { return date; }
        set { date = value; }
    }
    public string TextMessage
    {
        get { return textMessage; }
        set
        {
            textMessage = value;
            OnPropertyChanged("TextMessage");
        }
    }
    public event PropertyChangedEventHandler PropertyChanged;
    public void OnPropertyChanged([CallerMemberName] string propertyName = "")
    {
        if (PropertyChanged != null)
            PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
    }
}
}

```

Клас Contact, модель-представлення контакту.

```

using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;
namespace Siggi.Models
{
    public class Contact : INotifyPropertyChanged
    {
        private string nick;

        public Contact() {}
        public Contact(string nick)
        {
            this.nick = nick;
        }

        public string Nick
        {
            get { return nick; }
            set

```

```

        {
            nick = value;
            NotifyPropertyChanged("Nick");
        }
    }
    public override string ToString()
    {
        return Nick;
    }

    public event PropertyChangedEventHandler PropertyChanged;
    public void NotifyPropertyChanged([CallerMemberName] string propertyName = "")
    {
        if (PropertyChanged != null)
            PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
    }
}

```

Клас AboutSiggi, модель-представлення вікна з історією компаньйону.

```

using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;
namespace Siggi.Models.SiggiModels
{
    public class AboutSiggi : INotifyPropertyChanged
    {
        private string aboutSiggi;

        public AboutSiggi()
        {
            aboutSiggi = getInfo();
        }

        public string getInfo()
        {
            aboutSiggi = "Привіт!\n" +
                "Мене звали Сіггі та я голуб-альбінос. Як у вас кажуть: серед своїх я була білою вороною та мене часто ображали мої родичи:\n" +
                "До того ж, я завжди почувалася трошки людиною, трошки пташкою, а тому зайвою у своїй стаї.\n" +
                "Тому нещодавно я вирішила працювати на себе та відкрити свою власну поштову компанію. Назву її " +
                "\"Siggi Post Corporation\"... чи якось по-іншому. Ще не вирішила.\n" +
                "Як поштовий голуб, зобов'язуюсь приносити листи вчасно у будь-яку погоду. "
            +
                "А як трошки людина, буду намагатися з тобою подружитися та бути уважною та веселою.\n" +
                "Я люблю світ, добру атмосферу та чашечку вкусної кави у гарній компанії:).\n";

            return aboutSiggi;
        }
        public string AboutSiggiText
        {
            get { return aboutSiggi; }
            set
            {
                NotifyPropertyChanged("AboutSiggiText");
            }
        }

        public event PropertyChangedEventHandler PropertyChanged;
        public void NotifyPropertyChanged([CallerMemberName] string propertyName = "")
        {

```

```

        if (PropertyChanged != null)
            PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
    }
}

```

Клас `SiggiBird`, модель-представлення компаньйону.

```

using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows.Media.Imaging;
namespace Siggi.Models
{
    public class SiggiBird : INotifyPropertyChanged
    {
        private string messageFromSiggi;
        private string pathForDialog;
        private BitmapImage pathImg;
        public string MessageFromSiggi
        {
            get { return messageFromSiggi; }
            set
            {
                messageFromSiggi = value;
                NotifyPropertyChanged("MessageFromSiggi");
            }
        }
        public string PathForDialog
        {
            get { return pathForDialog; }
            set
            {
                pathForDialog = value;
                NotifyPropertyChanged("PathForDialog");
            }
        }
        public BitmapImage PathImg
        {
            get { return pathImg; }
            set
            {
                pathImg = value;
                NotifyPropertyChanged("PathImg");
            }
        }
        public event PropertyChangedEventHandler PropertyChanged;
        public void NotifyPropertyChanged([CallerMemberName] string propertyName = "")
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}

```

Абстрактний клас `SiggiSpeakText` для тексту повідомлення від боту-компаньйону та його класи-нащадки.

```

using System;
namespace Siggi.Models
{
    public abstract class SiggiSpeakText
    {
        protected string speakText;
    }
}

```

```

        protected SiggiSpeakText(string text)
        {
            speakText = text;
        }
        public override string ToString()
        {
            return speakText;
        }
    }
    public class EnterSpeakText : SiggiSpeakText
    {
        public EnterSpeakText(string txt) : base(txt) { }
    }
    public class ListeningSpeakText : SiggiSpeakText
    {
        public ListeningSpeakText(string txt) : base(txt) { }
    }
    public class MailSpeakText : SiggiSpeakText
    {
        public MailSpeakText(string txt) : base(txt) { }
    }
    public class CoffeeSpeakText : SiggiSpeakText
    {
        public CoffeeSpeakText(string txt) : base(txt) { }
    }
    public class AttentionSpeakText : SiggiSpeakText
    {
        public AttentionSpeakText(string txt) : base(txt) { }
    }
    public class InviteCoffeeSpeakText : SiggiSpeakText
    {
        public InviteCoffeeSpeakText(string txt) : base(txt) { }
    }
    public class SorrowSpeakText : SiggiSpeakText
    {
        public SorrowSpeakText(string txt) : base(txt) { }
    }
    public class BanSpeakText : SiggiSpeakText
    {
        public BanSpeakText(string txt) : base(txt) { }
    }
}

```

Абстрактний клас `SiggiSpeakPath` для картинки боту-компаньйону та його класи-нащадки.

```

using System;

namespace Siggi.Models
{
    public abstract class SiggiSpeakPath
    {
        protected string speakPath;

        protected SiggiSpeakPath(string text)
        {
            speakPath = text;
        }
        public override string ToString()
        {
            return speakPath;
        }
    }
    public class ListeningSpeakPath : SiggiSpeakPath

```

```

    {
        public ListeningSpeakPath() : base($"..\..\Images\SiggiEmotions\listening.png")
    }
}
public class MailSpeakPath : SiggiSpeakPath
{
    public MailSpeakPath() : base($"..\..\Images\SiggiEmotions\mail.png") { }
}

public class CoffeeSpeakPath : SiggiSpeakPath
{
    public CoffeeSpeakPath() : base($"..\..\Images\SiggiEmotions\coffee.png") { }
}
public class AttentionSpeakPath : SiggiSpeakPath
{
    public AttentionSpeakPath() : base($"..\..\Images\SiggiEmotions\attention.png")
}
}
public class SorrowSpeakPath : SiggiSpeakPath
{
    public SorrowSpeakPath() : base($"..\..\Images\SiggiEmotions\heartsore.png") { }
}
public class BanSpeakPath : SiggiSpeakPath
{
    public BanSpeakPath() : base($"..\..\Images\SiggiEmotions\resentment.png") { }
}
}

```

Абстрактний інтерфейс `ISiggiEmotionFactory` для формування повного стану компаньйона.

```

using System;

namespace Siggi.Models.SiggiModels
{
    public interface ISiggiEmotionFactory
    {
        SiggiSpeakText GetSiggiSpeakText();
        SiggiSpeakPath GetSiggiSpeakPath();
    }

    public class EnterSpeakFactory : ISiggiEmotionFactory
    {
        public SiggiSpeakText GetSiggiSpeakText()
        {
            return new EnterSpeakText("Привіт! Я - Сіггі. Я приношу пошту та позитив!");
        }
        public SiggiSpeakPath GetSiggiSpeakPath()
        {
            return new ListeningSpeakPath();
        }
    }

    public class ListeningSpeakFactory : ISiggiEmotionFactory
    {
        public SiggiSpeakText GetSiggiSpeakText()
        {
            return new ListeningSpeakText("Сподіваюсь, ми з тобою подружимося...");
        }
        public SiggiSpeakPath GetSiggiSpeakPath()
        {
            return new ListeningSpeakPath();
        }
    }

    public class MailSpeakFactory : ISiggiEmotionFactory

```

```

{
    public SiggiSpeakText GetSiggiSpeakText()
    {
        return new MailSpeakText("Лист від ");
    }
    public SiggiSpeakPath GetSiggiSpeakPath()
    {
        return new MailSpeakPath();
    }
}

public class CoffeeSpeakFactory : ISiggiEmotionFactory
{
    public SiggiSpeakText GetSiggiSpeakText()
    {
        return new CoffeeSpeakText("Каву приємніше пити у компанії, авжеж? Коли я була
зовсім маленьким пташенятком та вперше побачила каву, я нахилилася над нею та майже не впала
в чашку! Сором:");
    }
    public SiggiSpeakPath GetSiggiSpeakPath()
    {
        return new CoffeeSpeakPath();
    }
}

public class AttentionSpeakFactory : ISiggiEmotionFactory
{
    public SiggiSpeakText GetSiggiSpeakText()
    {
        return new AttentionSpeakText("Чи можеш приділити мені трошки уваги?");
    }
    public SiggiSpeakPath GetSiggiSpeakPath()
    {
        return new AttentionSpeakPath();
    }
}

public class InviteCoffeeSpeakFactory : ISiggiEmotionFactory
{
    public SiggiSpeakText GetSiggiSpeakText()
    {
        return new InviteCoffeeSpeakText("Хочеться чогось гарячого, затишного,
ароматного... Наприклад, кави:). Складеш мені компанію?");
    }
    public SiggiSpeakPath GetSiggiSpeakPath()
    {
        return new AttentionSpeakPath();
    }
}

public class SorrowSpeakFactory : ISiggiEmotionFactory
{
    public SiggiSpeakText GetSiggiSpeakText()
    {
        return new SorrowSpeakText("Звісно, кому зараз потрібна птиця...");
    }
    public SiggiSpeakPath GetSiggiSpeakPath()
    {
        return new SorrowSpeakPath();
    }
}

public class BanSpeakFactory : ISiggiEmotionFactory
{
    public SiggiSpeakText GetSiggiSpeakText()
    {
        return new BanSpeakText("***Сігги вас ігнорує***");
    }
}

```

```

        public SiggiSpeakPath GetSiggiSpeakPath()
        {
            return new BanSpeakPath();
        }
    }
}

```

Клас SimpleCommand, що реалізує стандартний інтерфес команди ICommand.

```

using System;
using System.Windows.Input;
namespace Siggi.Commands
{
    public class SimpleCommand : ICommand
    {
        private Action<object> execute;
        private Func<object, bool> canExecute;
        public event EventHandler CanExecuteChanged
        {
            add { CommandManager.RequerySuggested += value; }
            remove { CommandManager.RequerySuggested -= value; }
        }
        public SimpleCommand(Action<object> execute, Func<object, bool> canExecute = null)
        {
            this.execute = execute;
            this.canExecute = canExecute;
        }
        public bool CanExecute(object parameter)
        {
            return this.canExecute == null || this.canExecute(parameter);
        }
        public void Execute(object parameter)
        {
            this.execute(parameter);
        }
    }
}

```

Xaml-розмітка вікна MainWindow.

```

<Window x:Class="Siggi.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Siggi"
        mc:Ignorable="d"
        Title="Siggi - bird of the world" Height="530" Width="350"
        Icon="/Siggi;component/Images/SiggiIcon.ico" Style="{StaticResource WindowStyleLogIn}"
        Loaded="Window_Loaded">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="0.07*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="0.07*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="0.07*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <DockPanel Grid.Row="0" Grid.ColumnSpan="3">
            <Menu MinWidth="130" MaxHeight="25" VerticalAlignment="Top"
            Style="{StaticResource MenuStyleMain}">

```

```

        <MenuItem Header="Siggi">
            <MenuItem Header="Увійти в обліковий запис" Command="{Binding
CheckCommand}" Style="{StaticResource MenuItemStyleMain}"/>
            <MenuItem Header="Зареєструватися" Command="{Binding
RegistrationCommand}" Style="{StaticResource MenuItemStyleMain}"/>
            <MenuItem Header="Вийти з застосунку" Command="{Binding ExitCommand}"
Style="{StaticResource MenuItemStyleMain}"/>
        </MenuItem>
        <MenuItem Header="Про застосунок">
            <MenuItem Header="Трошки про Cirri" Command="{Binding AboutCommand}"
Style="{StaticResource MenuItemStyleMain}"/>
        </MenuItem>
    </Menu>
</DockPanel>
<StackPanel Grid.Row="1" Orientation="Vertical" HorizontalAlignment="Center"
Grid.Column="1" >
    <TextBlock Text="Ласкаво прошу" HorizontalAlignment="Center" FontSize="24"
Style="{StaticResource TextBlockGeneral}" />
    <TextBlock Text="до" HorizontalAlignment="Center" FontSize="24"
Style="{StaticResource TextBlockGeneral}" />
    <TextBlock Text="Cirri!" HorizontalAlignment="Center" FontSize="28"
Style="{StaticResource TextBlockGeneral}" />

    <Image Source="/Siggi;component/Images/SiggiIcon.ico" MaxWidth="70"
MaxHeight="70" Margin="0,10" />

    <TextBlock Text="Введіть свій логін:" TextWrapping="Wrap"
HorizontalAlignment="Center" FontSize="14" Style="{StaticResource TextBlockGeneral}"/>
    <TextBox Text="{Binding Nick}" Margin="15" Width="150"/>
    <TextBlock Text="Введіть свій пароль:" TextWrapping="Wrap"
HorizontalAlignment="Center" FontSize="14" Style="{StaticResource TextBlockGeneral}"/>
    <TextBox Text="{Binding Pass}" Margin="15" Width="150"/>
    <Button Content="Увійти" Command="{Binding CheckCommand}" Style="{StaticResource
ButtonLogIn}" Template="{StaticResource ButtonGeneralTemplate}"/>

    <TextBlock Text="Немає облікового запису?" TextWrapping="Wrap"
HorizontalAlignment="Center" FontSize="14" Margin="0,15,0,0" Style="{DynamicResource
TextBlockGeneral}"/>
    <Button Content="Я хочу подружитися з Cirri!" Command="{Binding
RegistrationCommand}" Margin="0,10,0,0" Style="{StaticResource ButtonLogIn}"
Template="{StaticResource ButtonGeneralTemplate}"/>
</StackPanel>
</Grid>
</Window>

```

### Клас вікна MainWindow.

```

using Siggi.ViewModels;
using System;
using System.Windows;
namespace Siggi
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            DataContext = new LogWindowVM();
        }
    }
}

```



```

        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
},
(obj) => Nick != null && Pass != null));
}
}
private SimpleCommand registrationCommand;
public SimpleCommand RegistrationCommand
{
    get
    {
        return registrationCommand ??
            (registrationCommand = new SimpleCommand(obj =>
            {
                makeRegistrationWindow();
            }));
    }
}
private SimpleCommand exitCommand;
public SimpleCommand ExitCommand
{
    get
    {
        return exitCommand ??
            (exitCommand = new SimpleCommand(obj =>
            {
                closeProgram();
            }));
    }
}

public void userCheck()
{
    bool result = true;
    using (var srv = new SrvRefinfo.ServiceUserInfoClient())
    {
        if (!passCheck())
        {
            result = false;
            Pass = "Ой, щось не так:";
        }

        if (!nickCheck())
        {
            result = false;
            Nick = "Не можу пригадати такого знайомого...";
        }

        if (result == true)
        {
            string p = GetHash(Pass);
            int id = srv.getUserId(Nick, p);
            string name = Nick;
            makeMainWindow(id, name);
        }
    }
}

private bool nickCheck()
{
    bool result = false;

```

```

        using (var srv = new SrvRefinfo.ServiceUserInfoClient())
        {
            result = srv.isLogExist(Nick);
        }

        return result;
    }
    private bool passCheck()
    {
        bool result = true;
        string p = GetHash(Pass);
        using (var srv = new SrvRefinfo.ServiceUserInfoClient())
        {
            result = srv.isPassExist(Nick, p);
        }

        return result;
    }
    private void makeAboutWindow()
    {
        AboutWindow aboutWindow = new AboutWindow();
        aboutWindow.Owner = Application.Current.Windows[0];
        aboutWindow.Show();
    }
    private void makeMainWindow(int ID, string Nick)
    {
        SiggiWindow mainWindow = new SiggiWindow(ID, Nick);
        mainWindow.Show();
        Application.Current.Windows[0].Close();
    }
    private void makeRegistrationWindow()
    {
        Registration regWindow = new Registration();
        regWindow.Show();
        Application.Current.Windows[0].Close();
    }

    private void closeProgram()
    {
        System.Windows.Application.Current.Shutdown();
    }

    private static string GetHash(string input)
    {
        //SHA512
        var bytes = System.Text.Encoding.UTF8.GetBytes(input);
        using (var hash = System.Security.Cryptography.SHA512.Create())
        {
            var hashedInputBytes = hash.ComputeHash(bytes);

            var hashedInputStringBuilder = new System.Text.StringBuilder(128);
            foreach (var b in hashedInputBytes)
                hashedInputStringBuilder.Append(b.ToString("X2"));
            return hashedInputStringBuilder.ToString();
        }
    }
    public event PropertyChangedEventHandler PropertyChanged;
    public void NotifyPropertyChanged([CallerMemberName] string propertyName = "")
    {
        if (PropertyChanged != null)
            PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
    }
}

```

## Xaml-розмітка вікна Registration.

```

<Window x:Class="Siggi.View.Registration"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Siggi.View"
        mc:Ignorable="d"
        Title="Siggi - bird of the world" Height="550" Width="500"
        Icon="/Siggi;component/Images/SiggiIcon.ico" Style="{StaticResource
WindowStyleRegistration}" Loaded="Window_Loaded">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="0.07*" />
            <RowDefinition Height="0.08*" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <DockPanel Grid.Row="0">
            <Menu MinWidth="130" MaxHeight="25" VerticalAlignment="Top"
Style="{StaticResource MenuStyleMain}">
                <MenuItem Header="Siggi">
                    <MenuItem Header="Увійти існуючому користувачу" Command="{Binding
LogCommand}" Style="{StaticResource MenuItemStyleMain}" />
                    <MenuItem Header="Вихід із застосунку" Command="{Binding ExitCommand}"
Style="{StaticResource MenuItemStyleMain}" />
                </MenuItem>
            </Menu>
        </DockPanel>

        <TextBlock Grid.Row="1" Text="Сігри радіє новому знайомому!" TextWrapping="Wrap"
HorizontalAlignment="Center" FontSize="24" Margin="0,0,10,0" Style="{StaticResource
TextBlockGeneral}" />

        <StackPanel Grid.Row="2" Orientation="Vertical" HorizontalAlignment="Center">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="0.5*" />
                </Grid.ColumnDefinitions>
                <TextBlock Grid.Column="0" Text="Введіть свій логін: (обов'язкове поле)"
TextWrapping="Wrap" HorizontalAlignment="Left" FontSize="14" Margin="0,5,6,0"
Style="{StaticResource TextBlockGeneral}" />
                <TextBox Grid.Column="1" Text="{Binding Path=userInvite.Nick}"
Margin="0,15,15,15" Width="150" HorizontalAlignment="Right" Grid.ColumnSpan="2" />
            </Grid>
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>
                <TextBlock Grid.Column="0" Text="Введіть своє ім'я:" TextWrapping="Wrap"
HorizontalAlignment="Left" FontSize="14" Margin="0,15,0,0" Style="{StaticResource
TextBlockGeneral}" />
                <TextBox Grid.Column="1" Text="{Binding Path=userInvite.UserName}"
Margin="15" Width="150" HorizontalAlignment="Right" />
            </Grid>
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>
            </Grid>
        </StackPanel>
    </Grid>

```

```

        <TextBlock Grid.Column="0" Text="Введіть своє прізвище:" TextWrapping="Wrap"
HorizontalAlignment="Left" FontSize="14" Margin="0,15,0,0" Style="{StaticResource
TextBlockGeneral}"/>
        <TextBox Grid.Column="1" Text="{Binding Path=userInvite.LastName}"
Margin="15" Width="150" HorizontalAlignment="Right"/>
    </Grid>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <TextBlock Grid.Column="0" Text="Введіть свій пароль: (обов'язкове поле)"
TextWrapping="Wrap" HorizontalAlignment="Left" FontSize="14" Margin="0,10,5,0"
Style="{StaticResource TextBlockGeneral}"/>
        <TextBox Grid.Column="1" Text="{Binding Path=userInvite.Password}"
Margin="15" Width="150" HorizontalAlignment="Right"/>
    </Grid>

    <Button Content="Полетіли! :)" Command="{Binding RegistCommand}"
Margin="0,40,0,0" Style="{StaticResource ButtonRegistration}" Template="{StaticResource
ButtonGeneralTemplate}"/>
    <Image Source="/Siggi;component/Images/FeatherMain.jpg" MaxWidth="170"
MaxHeight="170" Margin="0,35,0,0"/>
</StackPanel>
</Grid>
</Window>

```

### Клас вікна Registration.

```

using Siggi.ViewModels;
using System;
using System.Windows;
namespace Siggi.View
{
    public partial class Registration : Window
    {
        public Registration()
        {
            InitializeComponent();
        }
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            DataContext = new RegistrationVM();
        }
    }
}

```

### Клас RegistrationVM, моделі-представлення вікна MainWindow.

```

using Siggi.Commands;
using Siggi.Models;
using Siggi.View;
using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows;
namespace Siggi.ViewModels
{
    public class RegistrationVM : INotifyPropertyChanged
    {
        private UserInvite userInfo;
        private SrvRefinfo.ServiceUserInfoClient srv;

        public RegistrationVM()
        {

```

```

        userInfo = new UserInvite();
    }
    public UserInvite userInvite
    {
        get { return userInfo; }
        set
        {
            userInfo = value;
            NotifyPropertyChanged("userInvite");
        }
    }
    private SimpleCommand registCommand;
    public SimpleCommand RegistCommand
    {
        get
        {
            return registCommand ??
                (registCommand = new SimpleCommand(obj =>
                {
                    if (!testLogin())
                    {
                        userInvite.Nick = "Від 3 до 20 символів без пробілів";
                        return;
                    }
                    if (!testPass())
                    {
                        userInvite.Password = "Від 6 до 30 символів без пробілів";
                        return;
                    }

                    if (checkLogin())
                    {
                        userInvite.Nick = "Такий знайомий в мене вже є";
                        return;
                    }

                    bool result = false;
                    testName();
                    string password = GetHash(userInvite.Password);
                    using (srv = new SrvRefinfo.ServiceUserInfoClient())
                    {
                        result = srv.Registration(userInvite.Nick, password,
userInvite.UserName, userInvite.LastName);
                    }
                    if (!result)
                    {
                        MessageBox.Show("Ой, щось пішло не так:(\nСпробуйте ще раз\n");
                    }
                    else
                    {
                        try
                        {
                            srv = new SrvRefinfo.ServiceUserInfoClient();
                            int id = srv.getUserId(userInvite.Nick, password);
                            makeSiggWindow(id, userInvite.Nick);
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show(ex.Message);
                        }
                    }
                }
                },
                (obj) => userInvite.Nick != null && userInvite.Password != null));
        }
    }
}

```

```

}
private SimpleCommand logCommand;
public SimpleCommand LogCommand
{
    get
    {
        return logCommand ??
            (logCommand = new SimpleCommand(obj =>
            {
                makeLogWindow();
            }));
    }
}
private SimpleCommand exitCommand;
public SimpleCommand ExitCommand
{
    get
    {
        return exitCommand ??
            (exitCommand = new SimpleCommand(obj =>
            {
                closeProgram();
            }));
    }
}
private bool testLogin()
{
    if (userInvite.Nick == null)
        return false;
    string test = userInvite.Nick.Replace(" ", "");
    if (test != userInvite.Nick)
        return false;
    if (userInvite.Nick.Length < 3 || userInvite.Nick.Length > 20)
        return false;

    return true;
}
private bool testPass()
{
    if (userInvite.Password == null)
        return false;
    string test = userInvite.Password.Replace(" ", "");
    if (test != userInvite.Password)
        return false;
    if (userInvite.Password.Length < 6 || userInvite.Password.Length > 20)
        return false;

    return true;
}
private void testName()
{
    if (userInvite.UserName == null)
        userInvite.UserName = "";

    if (userInvite.LastName == null)
        userInvite.LastName = "";
}
private bool checkLogin()
{
    bool result = true;
    using (srv = new SrvRefinfo.ServiceUserInfoClient())
    {
        result = srv.isLogExist(userInvite.Nick);
    }
}

```

```

        return result;
    }
    private void makeSiggWindow(int ID, string Nick)
    {
        SiggiWindow mainWindow = new SiggiWindow(ID, Nick);
        mainWindow.Show();
        Application.Current.Windows[0].Close();
    }
    private void makeLogWindow()
    {
        MainWindow logWindow = new MainWindow();
        logWindow.Show();
        Application.Current.Windows[0].Close();
    }
    private void closeProgram()
    {
        System.Windows.Application.Current.Shutdown();
    }
    public event PropertyChangedEventHandler PropertyChanged;
    public void NotifyPropertyChanged([CallerMemberName] string propertyName = "")
    {
        if (PropertyChanged != null)
            PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
    }
    private static string GetHash(string input)
    {
        //SHA512
        var bytes = System.Text.Encoding.UTF8.GetBytes(input);
        using (var hash = System.Security.Cryptography.SHA512.Create())
        {
            var hashedInputBytes = hash.ComputeHash(bytes);

            var hashedInputStringBuilder = new System.Text.StringBuilder(128);
            foreach (var b in hashedInputBytes)
                hashedInputStringBuilder.Append(b.ToString("X2"));
            return hashedInputStringBuilder.ToString();
        }
    }
}

```

### Xaml-розмітка вікна AboutWindow.

```

<Window x:Class="Siggi.View.AboutWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Siggi.View"
        mc:Ignorable="d"
        Title="Про Siggi" HorizontalAlignment="Center"
        Icon="/Siggi;component/Images/SiggiIcon.ico" Style="{StaticResource WindowStyleAbout}"
        Loaded="Window_Loaded" >
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition MinWidth="100" Width="*" />
            <ColumnDefinition MinWidth="100" Width="0.8*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="0.3*" />
        </Grid.RowDefinitions>
        <TextBlock Grid.Column="0" Grid.Row="0" Text="{Binding Path=AboutSiggiText}"
        FontSize="14" Margin="20" TextWrapping="Wrap" Style="{StaticResource TextBlockGeneral}" />
        <Image Grid.Column="0" Grid.RowSpan="2"
        Source="/Siggi;component/Images/FeathersProfile.png" VerticalAlignment="Top"
        Margin="0,316,0,0" />
        <Image Grid.Column="1" Grid.Row="0" Grid.RowSpan="2"
        Source="/Siggi;component/Images/SiggiEmotions/SiggiIcon.png" VerticalAlignment="Top" />
    </Grid>

```

```
</Grid>
</Window>
```

### Клас вікна AboutWindow.

```
using Siggi.Models.SiggiModels;
using System;
using System.Windows;
namespace Siggi.View
{
    public partial class AboutWindow : Window
    {
        public AboutWindow()
        {
            InitializeComponent();
        }
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            DataContext = new AboutSiggi();
        }
    }
}
```

### Xaml-розмітка вікна UserInfoWindow.

```
<Window x:Class="Siggi.View.UserInfoWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Siggi.View"
        mc:Ignorable="d"
        Title="Карточка користувача" HorizontalAlignment="Center"
        Icon="/Siggi;component/Images/SiggiIcon.ico" Style="{StaticResource WindowUserInfo}">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition MinWidth="100" Width="*" />
            <ColumnDefinition MinWidth="100" Width="0.7*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="0.3*" />
        </Grid.RowDefinitions>
        <StackPanel Grid.Row="0" Grid.Column="0" Orientation="Vertical"
            HorizontalAlignment="Center" Margin="10,0,0,0">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="0.5*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <TextBlock Grid.Column="0" Text="Ім'я:" TextWrapping="Wrap"
                    HorizontalAlignment="Left" FontSize="16" Margin="0,15,0,0" Style="{StaticResource
                    TextBlockGeneral}" />
                <TextBox Grid.Column="1" Text="{Binding Path=User.UserName,
                    UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}"
                    IsEnabled="{Binding IsMe}" Style="{StaticResource TextBoxInfo}"
                    HorizontalAlignment="Right" Grid.ColumnSpan="2" />
            </Grid>
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="0.5*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <TextBlock Grid.Column="0" Text="Прізвище:" TextWrapping="Wrap"
                    HorizontalAlignment="Left" FontSize="16" Margin="0,15,0,0" Style="{StaticResource
                    TextBlockGeneral}" />
                <TextBox Grid.Column="1" Text="{Binding Path=User.LastName,
                    UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}"
```

```

        IsEnabled="{Binding IsMe}" Style="{StaticResource TextBoxInfo}"
HorizontalAlignment="Right"/>
    </Grid>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="0.5*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <TextBlock Grid.Column="0" Text="Країна:" TextWrapping="Wrap"
HorizontalAlignment="Left" FontSize="16" Margin="0,15,0,0" Style="{StaticResource
TextBlockGeneral}"/>
        <TextBox Grid.Column="1" Text="{Binding Path=User.Country,
UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}"
        IsEnabled="{Binding IsMe}" Style="{StaticResource TextBoxInfo}"
HorizontalAlignment="Right"/>
    </Grid>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="0.5*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <TextBlock Grid.Column="0" Text="Місто:" TextWrapping="Wrap"
HorizontalAlignment="Left" FontSize="16" Margin="0,10,5,0" Style="{StaticResource
TextBlockGeneral}"/>
        <TextBox Grid.Column="1" Text="{Binding Path=User.City,
UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}"
        IsEnabled="{Binding IsMe}" Style="{StaticResource TextBoxInfo}"
HorizontalAlignment="Right"/>
    </Grid>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="0.5*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <TextBlock Grid.Column="0" Text="Відносини з Сіггі:" TextWrapping="Wrap"
HorizontalAlignment="Left" FontSize="16" Margin="0,10,5,0" Style="{StaticResource
TextBlockGeneral}"/>
        <TextBox Grid.Column="1" Text="{Binding Path=User.Role}" IsEnabled="False"
Style="{StaticResource TextBoxInfo}" HorizontalAlignment="Right"/>
    </Grid>
</StackPanel>
<Image Grid.Column="0" Grid.RowSpan="2"
Source="/Siggi;component/Images/FeathersProfile.png" VerticalAlignment="Top"
Margin="0,220,0,0" />
<Image Grid.Column="1" Grid.Row="0" Grid.RowSpan="2"
Source="/Siggi;component/Images/SiggiEmotions/SiggiIcon.png" VerticalAlignment="Top" />

    <Button Grid.Column="1" Grid.Row="1" Grid.ColumnSpan="2"
        Content="Змінити дані"
        Command="{Binding ChangeCommand}"
        Style="{StaticResource ButtonRegistration}"
        Template="{StaticResource ButtonGeneralTemplate}" Margin="0,0,0,5"/>
</Grid>
</Window>

```

### Клас вікна UserInfoWindow.

```

using Siggi.ViewModels;
using System;
using System.Windows;
namespace Siggi.View
{
    public partial class UserInfoWindow : Window
    {
        public UserInfoWindow(string name, bool isme)

```



```

        }
        },
        (obj) => IsMe == true));
    }
}
private UserInfo fillUser(string name)
{
    UserInfo u = new UserInfo();
    srv = new SrvRefinfo.ServiceUserInfoClient();
    try
    {
        UserInfoServer infoServer = srv.getUserInfo(name);

        u.ID = infoServer.ID;
        u.UserName = infoServer.UserName;
        u.LastName = infoServer.LastName;
        u.Country = infoServer.Country;
        u.City = infoServer.City;
        u.Role = infoServer.Role;
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        srv.Close();
    }

    return u;
}
private void ChangeInfo()
{
    bool isSuccess = false;
    srv = new SrvRefinfo.ServiceUserInfoClient();
    UserInfoServer infoServer = new UserInfoServer();
    infoServer.ID = User.ID;
    infoServer.UserName = User.UserName;
    infoServer.LastName = User.LastName;
    infoServer.Country = User.Country;
    infoServer.City = User.City;
    infoServer.Role = User.Role;
    try
    {
        {
            isSuccess = srv.updateUserInfo(infoServer);
            if (isSuccess == true)
            {
                MessageBox.Show("Дані успішно змінені");
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        srv.Close();
    }
}
}
public event PropertyChangedEventHandler PropertyChanged;
public void OnPropertyChanged([CallerMemberName] string prop = "")
{
    if (PropertyChanged != null)

```

```

        PropertyChanged(this, new PropertyChangedEventArgs(prop));
    }
}

```

### Xaml-розмітка вікна SiggiWindow.

```

<Window x:Class="Siggi.View.SiggiWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Siggi.View"
        xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"
        mc:Ignorable="d"
        Title="Siggi - bird of the world" Height="450" Width="700"
        HorizontalAlignment="Center" Icon="/Siggi;component/Images/SiggiIcon.ico"
        Style="{StaticResource WindowStyleMain}">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition MinWidth="300" Width="*/>
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition MinWidth="180" Width="0.8*/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="30"/>
            <RowDefinition Height="*/>
        </Grid.RowDefinitions>
        <DockPanel Grid.ColumnSpan="3" Grid.Row="0">
            <Menu MinWidth="130" Height="30" VerticalAlignment="Center"
                Style="{StaticResource MenuStyleMain}" >
                <MenuItem Header="Siggi" FontSize="14">
                    <MenuItem Header="Персональні дані"
                        Command="{Binding MyInfoCommand}"
                        Style="{StaticResource MenuItemStyleMain}"/>
                    <MenuItem Header="Вихід з облікового запису"
                        Command="{Binding LogCommand}"
                        Style="{StaticResource MenuItemStyleMain}"/>
                    <MenuItem Header="Вихід з додатку"
                        Command="{Binding ExitCommand}"
                        Style="{StaticResource MenuItemStyleMain}" />
                </MenuItem>
                <MenuItem Header="O Cirri" FontSize="14">
                    <MenuItem Header="Трошки про Cirri"
                        Command="{Binding AboutCommand}"
                        Style="{StaticResource MenuItemStyleMain}"/>
                </MenuItem>
            </Menu>
        </DockPanel>
    </Grid>

```

```

        </Menu>
    </DockPanel>
    <Grid Grid.Column="0" Grid.Row="1">
        <Grid.RowDefinitions>
            <RowDefinition Height="0.12*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="0.3*"/>
            <RowDefinition Height="0.2*"/>
        </Grid.RowDefinitions>
        <TextBlock Grid.Row="0" Text="{Binding
Path=UsersMessages.SelectedContactNick,
UpdateSourceTrigger=PropertyChanged}"
HorizontalAlignment="Center" FontSize="16" Style="{StaticResource TextBlockGeneral}"/>
        <TextBox Grid.Row="2" Text="{Binding Path=UsersMessages.TextMessage}"
HorizontalAlignment="Stretch" FontSize="14" TextWrapping="Wrap" Style="{StaticResource
TextBoxInput}"/>
        <StackPanel Grid.Row="3" HorizontalAlignment="Right" Margin="7" >
            <Button Content="Надіслати"
                Padding="3"
                Command="{Binding SendCommand}"
                Style="{StaticResource ButtonMain}"
                Template="{StaticResource ButtonGeneralTemplate}"/>
        </StackPanel>
        <ListBox Grid.Row="1" ItemsSource="{Binding
Path=UsersMessages.listOfMessages}"
SelectedItem="{Binding
Path=UsersMessages.SelectedMessage}">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <StackPanel Orientation="Horizontal">
                        <Image Source="/Siggi;component/Images/SiggiIcon.ico"
Width="30" Height="30" Margin="0,10" />
                        <StackPanel Orientation="Vertical">
                            <StackPanel Orientation="Horizontal">
                                <TextBlock Text="{Binding UserFrom}"
FontSize="14" Margin ="5" Style="{StaticResource TextBlockGeneral}"/>
                                <TextBlock Text="{Binding Date}" FontSize="12"
Margin ="5" Style="{StaticResource TextBlockGeneral}"/>
                            </StackPanel>
                            <TextBlock Text="{Binding TextMessage}" FontSize="14"
Margin ="5" Style="{StaticResource TextBlockGeneral}"/>
                        </StackPanel>
                    </StackPanel>
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>
    </Grid>

```

```

        </ListBox>
    </Grid>
    <GridSplitter Grid.Column="1" Grid.Row="1" ShowsPreview="True" Width="2"
HorizontalAlignment="Center" VerticalAlignment="Stretch" />
    <Grid Grid.Column="2" Grid.Row="1">
        <Grid.RowDefinitions>
            <RowDefinition Height="50"/>
            <RowDefinition Height="*/>
        </Grid.RowDefinitions>
        <StackPanel Grid.Row="0" Orientation="Horizontal"
HorizontalAlignment="Center">
            <TextBlock Text="{Binding Path=CurrentUser.NickName}" FontSize="14"
Margin = "5" HorizontalAlignment="Center" VerticalAlignment="Center" Style="{StaticResource
TextBlockGeneral}"/>
            <!--<Image Name="ImgUser"
Source="/Siggi;component/Images/SiggiIcon.ico" Width="30" Height="30" Margin="0,10" />-->
            <Image Source="{Binding Path=CurrentUser.PathImg}" Width="30"
Height="30" Margin="0,10" />
        </StackPanel>
    <Grid Grid.Row="1">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*/>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="0.9*/>
        </Grid.ColumnDefinitions>
        <Grid Grid.Column="0">
            <Grid.RowDefinitions>
                <RowDefinition/>
                <RowDefinition/>
            </Grid.RowDefinitions>
            <Viewbox Grid.Row="0" HorizontalAlignment="Right"
Margin="3,7,10,3">
                <Path Stroke="DarkGray" StrokeThickness="2">
                    <Path.Fill>
                        <LinearGradientBrush StartPoint="0.2,0"
EndPoint="0.8,1" >
                            <LinearGradientBrush.GradientStops>
                                <GradientStop Color="White"
Offset="0"></GradientStop>
                                <GradientStop Color="White"
Offset="0.45"></GradientStop>
                                <GradientStop Color="LightBlue"
Offset="0.9"></GradientStop>

```

```

                                <GradientStop                                Color="Gray"
Offset="1"></GradientStop>

                                </LinearGradientBrush.GradientStops>
                                </LinearGradientBrush>
                                </Path.Fill>
                                <Path.Data>
                                    <PathGeometry>
                                        <PathGeometry.Figures>
                                            <PathFigure                                StartPoint="20,0"
IsClosed="True">

                                                <LineSegment Point="140,0"/>
                                                <ArcSegment Point="160,20" Size="20,20"
SweepDirection="Clockwise"/>

                                                <LineSegment Point="160,60"/>
                                                <ArcSegment Point="140,80" Size="20,20"
SweepDirection="Clockwise"/>

                                                <LineSegment Point="70,80"/>
                                                <LineSegment Point="70,130"/>
                                                <LineSegment Point="40,80"/>
                                                <LineSegment Point="20,80"/>
                                                <ArcSegment Point="0,60" Size="20,20"
SweepDirection="Clockwise"/>

                                                <LineSegment Point="0, 20"/>
                                                <ArcSegment Point="20,0" Size="20,20"
SweepDirection="Clockwise"/>

                                            </PathFigure>
                                        </PathGeometry.Figures>
                                    </PathGeometry>
                                </Path.Data>
                            </Path>
                        </Viewbox>
                    <TextBlock                                Grid.Row="0"                                Text="{Binding
Path=SiggiActions.siggi.MessageFromSiggi}"                                TextWrapping="Wrap"                                Margin="7,30,10,80"
Style="{StaticResource                                TextBlockGeneral}"                                HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                    <Button                                Grid.Row="1"                                Command="{Binding
Path=SiggiActions.SiggiCommand}"                                Style="{StaticResource                                ButtonSiggi}"
Template="{StaticResource ButtonSiggiTemplate}">
                        <Button.Background>
                            <ImageBrush                                ImageSource="{Binding
Path=SiggiActions.siggi.PathImg}" Stretch="Uniform"/>
                        </Button.Background>
                    </Button>

```

```

        </Grid>
        <GridSplitter Grid.Column="1" ShowsPreview="False" Width="2"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch"/>

        <Grid Grid.Column="2">
            <Grid.RowDefinitions>
                <RowDefinition Height="*"/>
                <RowDefinition Height="0.15*"/>
            </Grid.RowDefinitions>
            <ListBox Grid.Row="0" ItemsSource="{Binding UserContacts}"
SelectedItem="{Binding SelectedContact}">
                <ListBox.ItemTemplate>
                    <DataTemplate>
                        <StackPanel Orientation="Horizontal">
                            <Image
Source="/Siggi;component/Images/SiggiIcon.ico" Width="30" Height="30" Margin="0,10" />
                            <TextBlock Text="{Binding Nick}" FontSize="14"
Margin = "5" Style="{StaticResource TextBlockGeneral}"/>
                        </StackPanel>
                    </DataTemplate>
                </ListBox.ItemTemplate>
                <i:Interaction.Triggers>
                    <i:EventTrigger EventName="MouseDoubleClick">
                        <i:InvokeCommandAction
                                Command="{Binding
DataContext.ContactCommand, RelativeSource={RelativeSource AncestorType=ListBox}}"
                                CommandParameter="{Binding
SelectedItem}" />
                    </i:EventTrigger>
                </i:Interaction.Triggers>
            </ListBox>
            <Button Grid.Row="1" Content="Детальніше"
Margin="20,3"
Command="{Binding InfoCommand}"
Style="{StaticResource ButtonMain}"
Template="{StaticResource ButtonGeneralTemplate}"/>
        </Grid>
    </Grid>
</Grid>
</Window>

Клас вікна SiggiWindow.
using Siggi.ViewModels;

```

```

using System;
using System.Windows;
namespace Siggi.View
{
    public partial class SiggiWindow : Window
    {
        public SiggiWindow(int ID, string Nick)
        {
            InitializeComponent();
            DataContext = new SiggiWindowVM(ID, Nick);}}}

```

Клас UsersMessagesVM, моделі-представлення частини вікна

SiggiWindow.

```

using Siggi.SrvRefinfo;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows;
namespace Siggi.ViewModels
{
    public class UsersMessagesVM : INotifyPropertyChanged
    {
        private SrvRefinfo.ServiceUserInfoClient srvInfo;
        public ObservableCollection<Siggi.Models.Message> listOfMessages { get; set; }
        private string selectedContactNick;
        private Siggi.Models.Message selectedMessage;
        private string textMessage;
        private string currentUser;
        public UsersMessagesVM(string userName, string nameContact)
        {
            currentUser = userName;
            SelectedContactNick = nameContact;
            listOfMessages = new ObservableCollection<Siggi.Models.Message>();
            getMessages();
        }
        public string SelectedContactNick
        {
            get { return selectedContactNick; }
            set {
                selectedContactNick = value;
                NotifyPropertyChanged("SelectedContactNick");
            }
        }
        public Siggi.Models.Message SelectedMessage
        {
            get { return selectedMessage; }
            set
            {
                selectedMessage = value;
                NotifyPropertyChanged("SelectedMessage");
            }
        }
        public string TextMessage
        {
            get { return textMessage; }
            set
            {
                textMessage = value;
                NotifyPropertyChanged("TextMessage");
            }
        }
    }
}

```

```

public void getMessages()
{
    try
    {
        using (srvInfo = new ServiceUserInfoClient())
        {
            List<MessageServer> messageServers = new List<MessageServer>();
            messageServers = srvInfo.getMessage(currentUser, SelectedContactNick);

            foreach (var item in messageServers)
            {
                listOfMessages.Add(new Siggi.Models.Message()
                {
                    ID = item.ID,
                    UserFrom = item.UserFrom,
                    UserTo = item.UserTo,
                    Date = item.Date,
                    TextMessage = item.TextMessage
                });
            }
            srvInfo.Close();
        }
    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}

public event PropertyChangedEventHandler PropertyChanged;
public void NotifyPropertyChanged([CallerMemberName] string propertyName = "")
{
    if (PropertyChanged != null)
        PropertyChanged(this, new PropertyChangedEventArgs(propertyName));}}}

```

Клас SiggiActionsVM, моделі-представлення частини вікна SiggiWindow.

```

using Siggi.Commands;
using Siggi.Models;
using Siggi.Models.SiggiModels;
using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Threading;
namespace Siggi.ViewModels
{
    public class SiggiActionsVM : INotifyPropertyChanged
    {
        DispatcherTimer timerAttention;
        DispatcherTimer timerAction;
        public SiggiBird siggi { get; set; }
    }
}

```

```

private MediaPlayer mediaPlayer;
protected ISiggiEmotionFactory siggiFactory;
public SiggiActionsVM()
{
    try
    {
        siggi = new SiggiBird();
        enterSiggi();
        startTimerAttention();
        startTimerAction();
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
private SimpleCommand siggiCommand;
public SimpleCommand SiggiCommand
{
    get
    {
        return siggiCommand ??
            (siggiCommand = new SimpleCommand(obj =>
            {
                checkEmotion();
            }));
    }
}
public void startTimerAttention()
{
    timerAttention = new DispatcherTimer();
    timerAttention.Tick += new EventHandler(attentionTick);
    timerAttention.Interval = new TimeSpan(0, 0, 30);
    timerAttention.Start();
}
public void startTimerAction()
{
    timerAction = new DispatcherTimer();
    timerAction.Tick += new EventHandler(actionTick);
    timerAction.Interval = new TimeSpan(0, 0, 30);
}
public void stopTimerAttention()
{

```

```

        timerAttention.Stop();
    }
    public void stopTimerAction()
    {
        timerAction.Stop();
    }
    internal void isMailDone()
    {
        if (siggiFactory is MailSpeakFactory)
        {
            listeningSiggi();
        }
        else return;
    }
    private void checkEmotion()
    {
        if (siggiFactory is ListeningSpeakFactory)
        {
            soundVoiceAsync();
        }
        else if (siggiFactory is MailSpeakFactory)
        {
            listeningSiggi();
        }
        else if (siggiFactory is InviteCoffeeSpeakFactory)
        {
            coffeeTime();
        }
        else if (siggiFactory is AttentionSpeakFactory)
        {
            string message = "Ти знаєш, що птаха не може зареєструватися як  

підприєміць? Підтримаєшь мою петицію проти?";
            string title = "Запит від Cirri";
            MessageBoxButtons buttons = MessageBoxButtons.YesNo;
            DialogResult result = System.Windows.Forms.MessageBox.Show(message,
title, buttons);

            if (result == DialogResult.No)
            {
                sorrowSiggi();
            }
            else
            {
                siggi.MessageFromSiggi = "Дуже, дуже дякую!:)";
            }
        }
    }

```

```

        }
    }
    else return;
}
void attentionTick(object sender2, EventArgs e)
{
    Random rnd = new Random();
    int value = rnd.Next(0, 100);

    if (value < 20)
    {
        if (timerAttention.IsEnabled == true)
        {
            timerAttention.Stop();
        }
        attentionSiggi();
        timerAction.Start();
    }
    if (value > 20 && value < 50)
    {
        if (timerAttention.IsEnabled == true)
        {
            timerAttention.Stop();
        }
        inviteCoffeSiggi();
        timerAction.Start();
    }
}
void actionTick(object sender2, EventArgs e)
{
    listeningSiggi();
    if (timerAttention.IsEnabled == false)
        timerAttention.Start();
    timerAction.Stop();
}
private void enterSiggi()
{
    siggi.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory,
$"..\..\Images\SiggiEmotions\Enter1.png")));
    siggiFactory = new EnterSpeakFactory();
    SiggiSpeakText siggiSpeakText = siggiFactory.GetSiggiSpeakText();
    siggi.MessageFromSiggi = siggiSpeakText.ToString();
}

```

```

        pictureEnterAsync();
        soundEnterAsync();
    }
    public void sendMessageOf(string name)
    {
        soundVoiceAsync();
        siggiFactory = new MailSpeakFactory();
        SiggispeakText siggispeakText = siggiFactory.GetSiggispeakText();
        SiggispeakPath siggispeakPath = siggiFactory.GetSiggispeakPath();
        siggi.MessageFromSiggi = $"{siggispeakText}{name}";
        siggi.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, siggispeakPath.ToString())));
    }
    private void listeningSiggi()
    {
        soundVoiceAsync();
        siggiFactory = new ListeningSpeakFactory();
        SiggispeakText siggispeakText = siggiFactory.GetSiggispeakText();
        SiggispeakPath siggispeakPath = siggiFactory.GetSiggispeakPath();
        siggi.MessageFromSiggi = siggispeakText.ToString();
        siggi.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, siggispeakPath.ToString())));
    }
    private void coffeeTime()
    {
        soundVoiceAsync();
        siggiFactory = new CoffeeSpeakFactory();
        SiggispeakText siggispeakText = siggiFactory.GetSiggispeakText();
        SiggispeakPath siggispeakPath = siggiFactory.GetSiggispeakPath();
        siggi.MessageFromSiggi = siggispeakText.ToString();
        siggi.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, siggispeakPath.ToString())));
    }
    private void attentionSiggi()
    {
        soundVoiceAsync();
        siggiFactory = new AttentionSpeakFactory();
        SiggispeakText siggispeakText = siggiFactory.GetSiggispeakText();
        SiggispeakPath siggispeakPath = siggiFactory.GetSiggispeakPath();
        siggi.MessageFromSiggi = siggispeakText.ToString();
        siggi.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, siggispeakPath.ToString())));
    }
}

```

```

private void inviteCoffeSiggi()
{
    soundVoiceAsync();
    siggiFactory = new InviteCoffeeSpeakFactory();
    SiggiSpeakText siggiSpeakText = siggiFactory.GetSiggiSpeakText();
    SiggiSpeakPath siggiSpeakPath = siggiFactory.GetSiggiSpeakPath();
    siggi.MessageFromSiggi = siggiSpeakText.ToString();
    siggi.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, siggiSpeakPath.ToString())));
}
private void sorrowSiggi()
{
    soundVoiceAsync();
    siggiFactory = new SorrowSpeakFactory();
    SiggiSpeakText siggiSpeakText = siggiFactory.GetSiggiSpeakText();
    SiggiSpeakPath siggiSpeakPath = siggiFactory.GetSiggiSpeakPath();
    siggi.MessageFromSiggi = siggiSpeakText.ToString();
    siggi.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, siggiSpeakPath.ToString())));
}
private void Ban()
{
    soundVoiceAsync();
    siggiFactory = new BanSpeakFactory();
    SiggiSpeakText siggiSpeakText = siggiFactory.GetSiggiSpeakText();
    SiggiSpeakPath siggiSpeakPath = siggiFactory.GetSiggiSpeakPath();
    siggi.MessageFromSiggi = siggiSpeakText.ToString();
    siggi.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, siggiSpeakPath.ToString())));
}
private async void pictureEnterAsync()
{
    string[] path = {
        $"..\\..\\Images\\SiggiEmotions\\Enter1.png",
        $"..\\..\\Images\\SiggiEmotions\\Enter2.png",
        $"..\\..\\Images\\SiggiEmotions\\Enter3.png"
    };
    foreach (var item in path)
    {
        try
        {
            siggi.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, item)));

```

```

        }
        catch
        {
            siggi.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory,
$"..\\..\\Images\\SiggiEmotions\\listening.png"))));
        }
        await Task.Run(() =>
        {
            Thread.Sleep(1000);
        });
    }
    siggi.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory,
$"..\\..\\Images\\SiggiEmotions\\listening.png"))));
}
private async void soundEnterAsync()
{
    await Task.Run(() =>
    {
        mediaPlayer = new MediaPlayer();
        try
        {
            mediaPlayer.Open(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, $"..\\..\\Sound\\Enter.mp3")));
            mediaPlayer.Play();
        }
        catch (Exception ex)
        {
            System.Windows.MessageBox.Show(ex.ToString());
        }
    });
}
private async void soundVoiceAsync()
{
    await Task.Run(() =>
    {
        mediaPlayer = new MediaPlayer();
        try
        {
            mediaPlayer.Open(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, $"..\\..\\Sound\\Voice.mp3")));
            mediaPlayer.Play();
        }
        catch (Exception ex)
        {
            System.Windows.MessageBox.Show(ex.ToString());
        }
    });
}

```

```

        }
        catch (Exception ex)
        {
            System.Windows.MessageBox.Show(ex.ToString());
        }
    });
}
public event PropertyChangedEventHandler PropertyChanged;
public void NotifyPropertyChanged([CallerMemberName] string propertyName =
    "")
{
    if (PropertyChanged != null)
        PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
}
}
}

```

Клас `SiggiWindowVM`, моделі-представлення вікна `SiggiWindow`.

```

using Siggi.Commands;
using Siggi.Models;
using Siggi.SrvRefM;
using Siggi.View;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.IO;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Media.Imaging;
namespace Siggi.ViewModels
{
    public class SiggiWindowVM : INotifyPropertyChanged,
        SrvRefM.IServiceMessageCallback
    {
        private UserOfSiggi currentUser;
        private Contact selectedContact;
        private ObservableCollection<Contact> userContacts;
        private SiggiActionsVM siggiViewModel;
        private UsersMessagesVM usersMessagesVM;
        private long serviceId;
    }
}

```

```

private SrvRefM.ServiceMessageClient srvRefM;
public SiggWindowVM(int ID, string Nick)
{
    currentUser = new UserOfSiggi();
    currentUser.ID = ID;
    currentUser.NickName = Nick;
    siggiViewModel = new SiggActionsVM();
    userContacts = getContacts(ID);
    getUserIcon();
    serviceId = -1;
    try
    {
        srvRefM = new SrvRefM.ServiceMessageClient(new
System.ServiceModel.InstanceContext(this));
        serviceId = srvRefM.getConnect(currentUser.NickName);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}
public SiggActionsVM SiggActions
{
    get { return siggiViewModel; }
    set
    {
        siggiViewModel = value;
        NotifyPropertyChanged("SiggActions");
    }
}
public UsersMessagesVM UsersMessages
{
    get { return usersMessagesVM; }
    set
    {
        usersMessagesVM = value;
        NotifyPropertyChanged("UsersMessages");
    }
}
public UserOfSiggi CurrentUser
{
    get { return currentUser; }
    set

```

```

        {
            currentUser = value;
            NotifyPropertyChanged("CurrentUser");
        }
    }
    public Contact SelectedContact
    {
        get { return selectedContact; }
        set
        {
            selectedContact = value;
            NotifyPropertyChanged("SelectedContact");
        }
    }
    public ObservableCollection<Contact> UserContacts
    {
        get { return userContacts; }
        set
        {
            userContacts = value;
            NotifyPropertyChanged("UserContacts");
        }
    }
    private SimpleCommand contactCommand;
    public SimpleCommand ContactCommand
    {
        get
        {
            return contactCommand ??
                (contactCommand = new SimpleCommand(obj =>
                {
                    siggiViewModel.isMailDone();
                    UsersMessages = new UsersMessagesVM(currentUser.NickName,
SelectedContact.Nick);
                }));
        }
    }
    private SimpleCommand sendCommand;
    public SimpleCommand SendCommand
    {
        get
        {
            return sendCommand ??

```

```

        (sendCommand = new SimpleCommand(obj =>
        {
            sendMessage();
        },
        (obj) => UsersMessages != null));
    }
}
private SimpleCommand logCommand;
public SimpleCommand LogCommand
{
    get
    {
        return logCommand ??
            (logCommand = new SimpleCommand(obj =>
            {
                makeLogWindow();
            }));
    }
}
private SimpleCommand aboutCommand;
public SimpleCommand AboutCommand
{
    get
    {
        return aboutCommand ??
            (aboutCommand = new SimpleCommand(obj =>
            {
                makeAboutWindow();
            }));
    }
}
private SimpleCommand myInfoCommand;
public SimpleCommand MyInfoCommand
{
    get
    {
        return myInfoCommand ??
            (myInfoCommand = new SimpleCommand(obj =>
            {
                makeUserInfowindow(currentUser.NickName, true);
            }));
    }
}
}

```

```

private SimpleCommand infoCommand;
public SimpleCommand InfoCommand
{
    get
    {
        return infoCommand ??
            (infoCommand = new SimpleCommand(obj =>
            {
                makeUserInfowindow(SelectedContact.Nick, false);
            },
            (obj) => SelectedContact != null));
    }
}
private SimpleCommand exitCommand;
public SimpleCommand ExitCommand
{
    get
    {
        return exitCommand ??
            (exitCommand = new SimpleCommand(obj =>
            {
                closeProgram();
            }));
    }
}
private ObservableCollection<Contact> getContacts(int ID)
{
    ObservableCollection<Contact> newContacts = new
ObservableCollection<Contact>();
    List<string> contactNames = new List<string>();
    SrvRefinfo.ServiceUserInfoClient srv = null;
    try
    {
        srv = new SrvRefinfo.ServiceUserInfoClient();
        contactNames = srv.getAllUsers(ID);
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        if (srv != null)

```

```

        {
            srv.Close();
        }
    }
    if (contactNames.Count > 0)
    {
        foreach(var item in contactNames)
        {
            newContacts.Add(
                new Contact(item));
        }
    }
    return newContacts;
}
private void getUserIcon()
{
    string path;
    path = $"..\..\Images\UsersImages\" + CurrentUser.NickName + ".jpg";
    SrvRefinfo.ServiceUserInfoClient srv = null;
    try
    {
        if (File.Exists(path))
        {
            CurrentUser.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, path)));
        }
        else
        {
            using (srv = new SrvRefinfo.ServiceUserInfoClient())
            {
                string imgString = srv.getIconUser(CurrentUser.ID);
                //string imgString = "";
                if (imgString == "")
                {
                    path = $"..\..\Images\SiggiIcon.ico";
                    CurrentUser.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, path)));
                }
                else
                {
                    byte[] bt = Convert.FromBase64String(imgString);

```

```

        using (FileStream fs = new
FileStream($"..\..\Images\UsersImages\\" + CurrentUser.NickName + ".jpg",
FileMode.OpenOrCreate, FileAccess.Write))
        {
            fs.Write(bt, 0, bt.Length);
            fs.Close();
        }
        if (File.Exists(path))
        {
            CurrentUser.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, path))); ;
        }

        else
        {
            path = $"..\..\Images\SiggiIcon.ico";
            CurrentUser.PathImg = new BitmapImage(new
Uri(System.IO.Path.Combine(Environment.CurrentDirectory, path)));
        }
    }
}
if (srv != null)
{
    srv.Close();
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}
private void sendMessage()
{
    int count = UsersMessages.listOfMessages.Count;
    try
    {
        if (UsersMessages != null)
        {
            string txt;
            if (UsersMessages.TextMessage != null)
            {
                txt = UsersMessages.TextMessage;
            }
        }
    }
}

```

```

        Siggis.Models.Message m = new Siggis.Models.Message()
        {
            ID = count + 1,
            UserFrom = CurrentUser.NickName,
            UserTo = UsersMessages.SelectedContactNick,
            Date = DateTime.Now,
            TextMessage = txt
        };
        UsersMessages.listOfMessages.Add(m);
        UsersMessages.TextMessage = "";
        try
        {
            SrvRefM.MessageServer ms = new SrvRefM.MessageServer()
            {
                ID = count + 1,
                UserFrom = m.UserFrom,
                UserTo = m.UserTo,
                Date = m.Date,
                TextMessage = m.TextMessage
            };
            SrvRefM.ServiceModelClient srvRefM = new
            SrvRefM.ServiceModelClient(new System.ServiceModel.InstanceContext(this));
            srvRefM.SendMessage(ms,
            UsersMessages.SelectedContactNick);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
    }

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}

private void closingPrepare()
{
    siggiViewModel.stopTimerAttention();
    siggiViewModel.stopTimerAction();
    if (serviceId != -1)
    {

```

```

        try
        {
            srvRefM.Disconnect(serviceId);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
    }
}

private void makeUserInfowindow(string name, bool isme)
{
    UserInfowindow infoWindow = new UserInfowindow(name, isme);
    infoWindow.Owner = Application.Current.Windows[0];
    infoWindow.Show();
}

private void makeAboutWindow()
{
    AboutWindow aboutWindow = new AboutWindow();
    aboutWindow.Owner = Application.Current.Windows[0];
    aboutWindow.Show();
}

private void makeLogWindow()
{
    closingPrepare();
    MainWindow logWindow = new MainWindow();
    logWindow.Show();
    Application.Current.Windows[0].Close();
}

private void closeProgram()
{
    closingPrepare();
    System.Windows.Application.Current.Shutdown();
}

public void SendMessageCallback(SrvRefM.MessageServer ms)
{
    try
    {
        if (UsersMessages != null)
        {
            if (ms.UserFrom == UsersMessages.SelectedContactNick)
            {
                UsersMessages.listOfMessages.Add(new Siggii.Models.Message()

```



```

        <transport sslProtocols="None" />
    </security>
</binding>
<binding name="NetTcpBinding_IServiceMessage">
    <security>
        <transport sslProtocols="None" />
    </security>
</binding>
</netTcpBinding>
</bindings>
<client>
    <endpoint address="net.tcp://localhost:8311/" binding="netTcpBinding"
        bindingConfiguration="NetTcpBinding_IServiceUserInfo"
contract="SrvRefinfo.IServiceUserInfo"
        name="NetTcpBinding_IServiceUserInfo">
        <identity>
            <userPrincipalName value="FREYASOLEN\DELL" />
        </identity>
    </endpoint>
    <endpoint address="net.tcp://localhost:8310/" binding="netTcpBinding"
        bindingConfiguration="NetTcpBinding_IServiceMessage"
contract="SrvRefM.IServiceMessage"
        name="NetTcpBinding_IServiceMessage">
        <identity>
            <userPrincipalName value="FREYASOLEN\DELL" />
        </identity>
    </endpoint>
</client>
</system.serviceModel>
</configuration>

```

**ДОДАТОК Б**  
**Слайди презентації**

Національний університет «Запорізька політехніка»  
Кафедра програмних засобів

Випускна робота бакалавра на тему:

## Програмна реалізація месенджера з ботом-компаньйоном

Виконала:

ст. гр. КНТз-110\_2

Анастасія ЮРАСОВА

Керівник:

доктор тех. наук, професор

Михайло ПОЛЯКОВ

2023

Рисунок Б.1 – Слайд № 1

## Об'єкт, предмет та мета роботи

- Об'єкт дослідження – процес функціонування месенджерів та способи їх поєднання з ботом, який імітує віртуального улюбленця-компаньйона.
- Предмет дослідження – програмна реалізація месенджеру з ботом-компаньйоном.
- Мета роботи – підвищення рівня діалогу месенджеру з користувачем.

Рисунок Б.2 – Слайд № 2



## Завдання роботи

- Провести аналіз предметної області та існуючих аналогів.
- Здійснити проектування месенджеру та інтерактивного бота-компаньйону.
- Реалізувати спроектовану модель месенджеру з інтерактивним ботом-компаньйоном.
- Виконати тестування розробленого месенджеру та бота-компаньйона.

3

Рисунок Б.3 – Слайд № 3



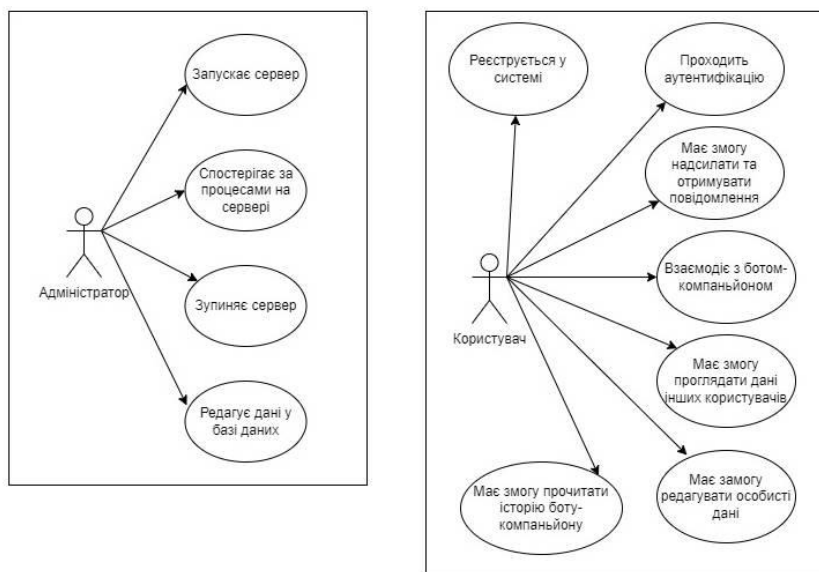
## Існуючі аналоги вирішення завдання

- ICQ
- Skype
- Viber
- Telegram
- WhatsApp
- Signal

4

Рисунок Б.4 – Слайд № 4

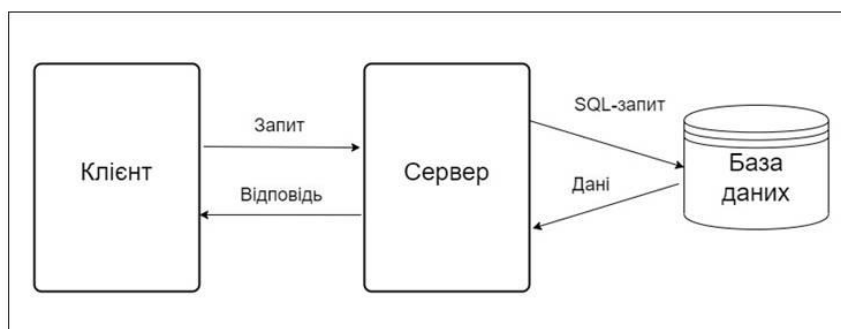
## Діаграма бізнес-прецедентів



5

Рисунок Б.5 – Слайд № 5

## Схема роботи програмного забезпечення



6

Рисунок Б.6 – Слайд № 6

## Вибір мови програмування

| Критерій порівняння                                              | C++ | C# | Java |
|------------------------------------------------------------------|-----|----|------|
| Додаткові інструменти для розробки клієнт-сервісних застосунків. | -   | +  | -    |
| Додаткові інструменти для створення графічних інтерфейсів.       | +   | +  | -    |
| Додаткові фреймворки для роботи з базою даних.                   | -   | +  | -    |
| Крос платформність.                                              | +   | +- | +    |
| Можливість прямої взаємодії з пам'яттю.                          | +   | +  | -    |
| Швидкість виконання за рахунок прямої роботи з пам'яттю.         | +   | -  | -    |
| Додаткові фреймворки для створення клієнт-серверних застосунків  | -   | +  | -    |

7

Рисунок Б.7 – Слайд № 7

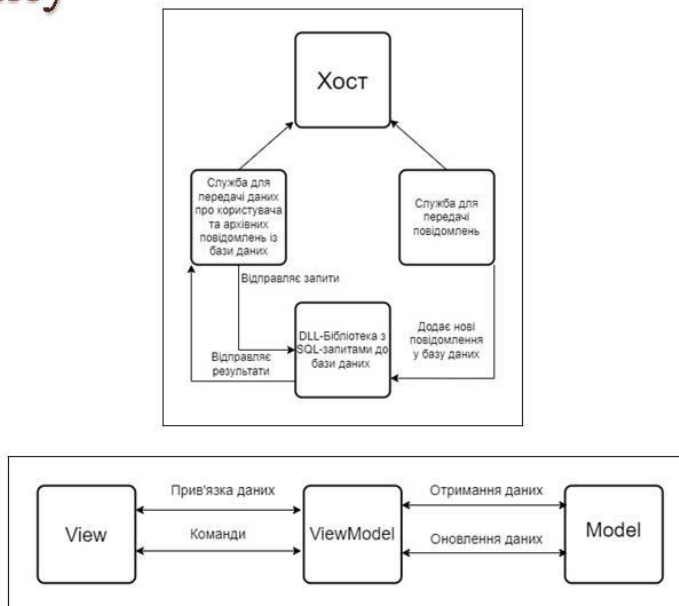
## Вибір системи керування базами даних

| Критерій порівняння | SQLite  | MSSQL                       | MySQL           |
|---------------------|---------|-----------------------------|-----------------|
| Архітектура.        | файлова | клієнт-серверна             | клієнт-серверна |
| Мова запитів.       | SQL     | T-SQL                       | SQL             |
| Рівень безпеки.     | низький | високий                     | високий         |
| Безкоштовна.        | так     | тільки обмежений функціонал | так             |

8

Рисунок Б.8 – Слайд № 8

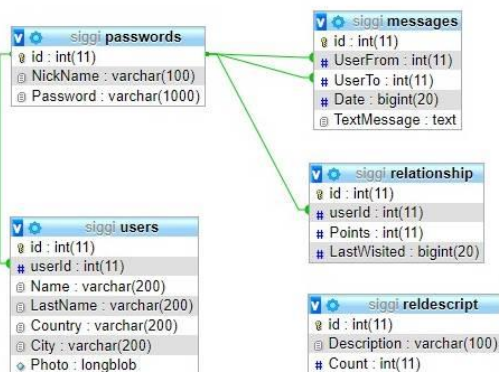
## Архітектура проєктування серверу та клієнту



9

Рисунок Б.9 – Слайд № 9

## Структура бази даних

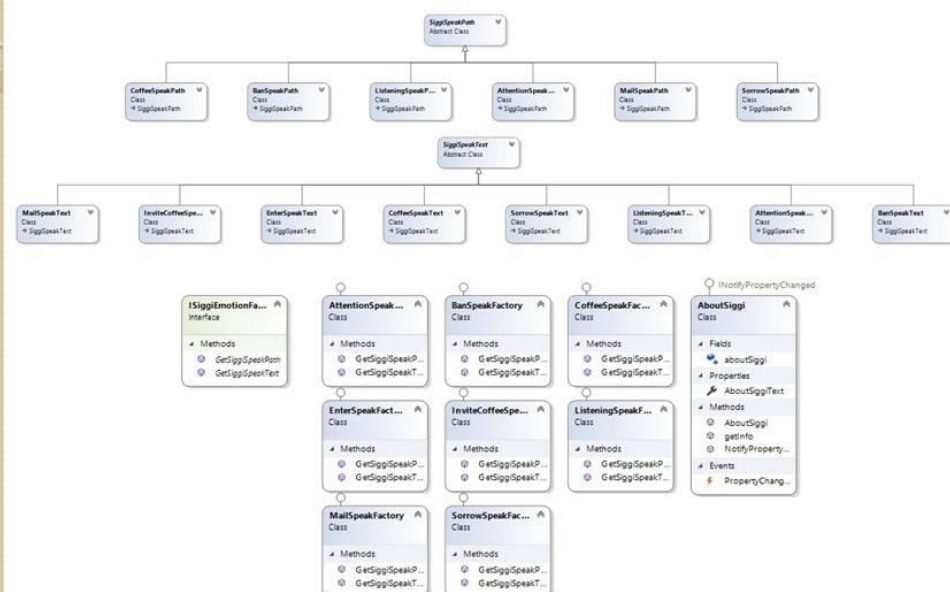


10

Рисунок Б.10 – Слайд № 10



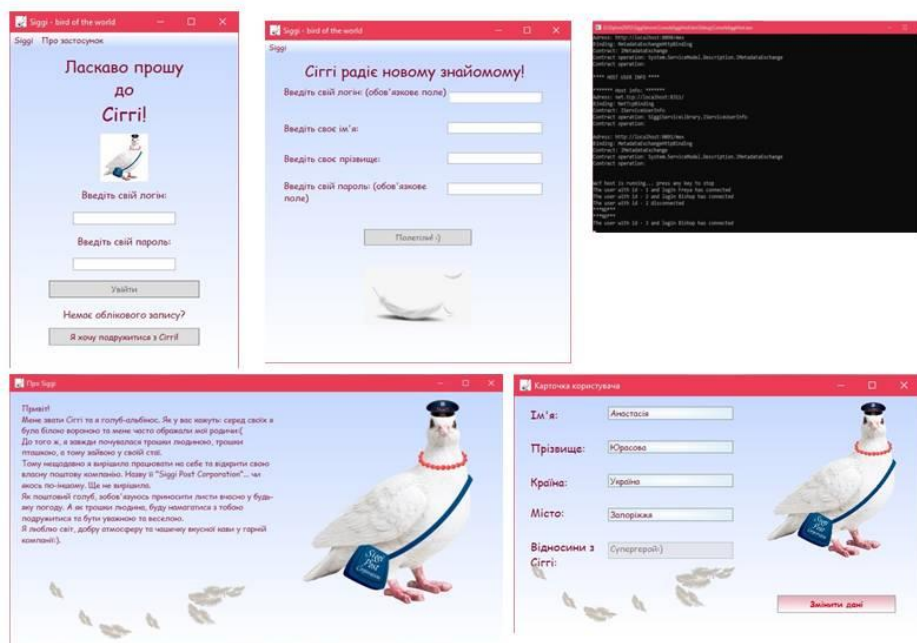
## Діаграма класів боту-компаньйону



13

Рисунок Б.13 – Слайд № 13

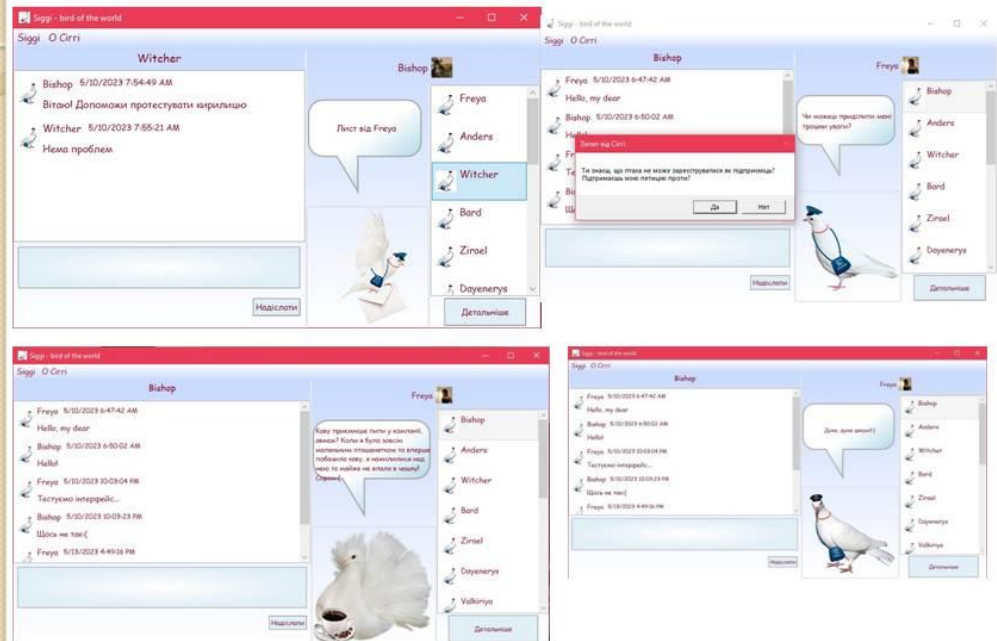
## Інтерфейс користувача



14

Рисунок Б.14 – Слайд № 14

## Приклад прояву боту-компаньйону



15

Рисунок Б.15 – Слайд № 15

## Висновки

- В результаті виконання дислоюної кваліфікаційної роботи був проведений аналіз предметної області та існуючих аналогів.
- Для реалізації ПЗ була обрана мова програмування C# та .NET Framework, а системою управління базами даних обрано MySQL. Для реалізації клієнт-серверної взаємодії використано WCF та реалізовано динамічне зв'язування у файлах конфігурацій.
- Запити до БД інкапсульовані у створеній DLL-бібліотеці, яку використовує сервер.
- Сам сервер складається з двох служб (рівню сесії для отримання чи редагування даних та рівню одного екземпляру для контролю отримання та розсилання повідомлень між клієнтами) та консольного застосунку-хосту.
- Частина клієнту виконана за архітектурним патерном MVVM фреймворку WPF. Користувач має змогу реєструватися та входити у систему, обмінюватися повідомленнями з іншими користувачами, переглядати інформацію про них та редагувати свої особисті дані. Також він має змогу прочитати історію компаньйону, реагувати на його запитання та пити з ним каву.
- Бот-компаньйон реалізований за допомогою пораджуючого патерну "Абстрактна фабрика".
- Створена архітектура застосунку є масштабованою та слабо зв'язаною, тобто, рішення дозволяє змінювати будь-який компонент або додавати новий без впливу на іншу частину програми.
- Наукова новизна роботи полягає у створенні бота-компаньйона. Це персоналізована іптаха-альбінос, яка має ім'я, свою історію, а саме, що вона вирішила піти від інших голубів та організувала власну поштову службу «Siggi Post Corporation». Вона любить людей та хоче з ними подружитися. Це відображається у тому, що бот-компаньйон періодично звертається до користувача з питаннями та проханнями. Бот-компаньйон виражає емоції та накопичує бали відносин із користувачем. За досягненням певної кількості балів, користувач отримує новий статус відносин із компаньйonom.

16

Рисунок Б.16 – Слайд № 16