

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 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗБЕРЕЖЕННЯ

ТА ПЕРЕГЛЯДУ ФОТОГРАФІЙ

SOFTWARE FOR STORING AND VIEWING THE PHOTOS

Виконав: студент(ка) 4 курсу, групи КНТ-119сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Лісанова Д.О.

(прізвище та ініціали)

Керівник Гофман Є.О.

(прізвище та ініціали)

Рецензент Коцур М.І.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

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

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Лісанової Данієли Олександрівни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для збереження та перегляду фотографій. Software for Storing and Viewing the Photos

керівник проєкту (роботи) Гофман Євгеній Олександрович, к.т.н.,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “27” квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) 7 червня 2022 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. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГофманЄ.О., доцент		
Нормоконтроль	БеловаА.В., асистент		

7. Дата видачі завдання “25”квітня 2022 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.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3,4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

ЛісановаД.О.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

ГофманЄ.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
101 с., 33 рис., 7 табл., 4 дод., 23 джерел.

ЗОБРАЖЕННЯ, ФОТОГРАФІЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ,
C#,MYSQL.

Об'єкт дослідження – процес розробки програмного забезпечення для збереження та перегляду фотографій.

Предмет дослідження – програмні засоби оброблення, збереження та перегляду фотографій.

Метою роботи є програмна реалізація процесу збереження та перегляду фотографій.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки, система керування базами даних MySQL.

Результати. Розроблено програмні засоби для збереження та перегляду фотографій.

Практична цінність роботи полягає у створенні програмних засобів для забезпечення процесу збереження та перегляду фотографій.

Висновки. Розроблено програмне забезпечення для автоматизації процесу збереження та перегляду фотографій.

Галузь використання – обробка зображень.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 101 pages, 33 figures, 7 tables, 4 appendixes, 23 sources.

IMAGE, PHOTOGRAPHY, SOFTWARE, C#, MYSQL.

The object of research is the process of software development for saving and viewing photos.

The subject of the research is software tools for saving and viewing photos.

The purpose of this work is the program implementation of the process of saving and viewing photos.

Materials, methods and technical tools: C# programming language, development environment, MySQL database management system.

Results. The software tools for saving and viewing photos have been developed.

The practical value of the work lies in the development of software to automate the process of saving and viewing photos.

Conclusions. The software for automation the process of saving and viewing photos is developed.

Scope of use – image processing.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Загальні поняття про програмні сервіси збереження та перегляду фотографій	11
1.2 Огляд програмних засобів оброблення, збереження та перегляду зображень.....	12
1.3 Висновки за розділом 1	22
2 Матеріали і методи	23
2.1 Вибір мови програмування	23
2.2 Вибір середовища розробки програмного забезпечення.....	24
2.3 Вибір системи керування базою даних	26
2.4 Висновки за розділом 2	28
3 Опис програми	29
3.1 Структурна схема програми	29
3.2 Функціонування програми збереження та перегляду фотографій	30
3.3 Опис функцій програмного забезпечення.....	33
3.4 База даних	35
3.5 Висновки за розділом 3	35
4 Експлуатація, тестування та експериментальне дослідження програми.....	37
4.1 Призначення й умови застосування програми	37
4.2 Характеристики програми	38
4.3 Інструкція по експлуатації програми.....	38
4.3.1 Звернення до програми.....	38
4.3.2 Вхідні й вихідні дані	39
4.3.3 Повідомлення.....	39
4.4 Тестування та результати роботи програмного забезпечення збереження та перегляду фотографій	40

4.5 Висновки за розділом 4	46
Висновки	47
Перелік джерел посилання	49
Додаток А Технічне завдання	51
Додаток Б Опис програми	55
Додаток В Текст програми	59
Додаток Д Слайди презентації	92

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

ОС – операційна система;

СКБД – система керування базами даних.

ВСТУП

Сьогодні існує багато видів програмних послуг і служб, що забезпечують зберігання та відображення зображень, кожна з яких має свої особливості, переваги та недоліки [1]-[6]. Так, деякі характеризуються дуже конкретними операціями, які звичайні користувачі не розуміють і не потребують. Усі сучасні засоби перегляду фотографій можна використовувати, як правило, у мережі Інтернет (наприклад, Instagram) або в звичайних автономних системах, таких як Microsoft.Picture.Manager [1]-[6]. Тому доцільно програмний продукт, який використовує комбінацію двох різних підходів, наприклад, дозволяє користувачам переглядати на будь-якому пристрої зображення, що зберігаються як в мережевих сервісах, так і на персональних комп'ютерах.

Таким чином, метою роботи є програмна реалізація процесу збереження та перегляду фотографій.

Для досягнення поставленої мети в роботі необхідно розв'язати такі задачі:

- виконати аналіз предметної області та процесу збереження та перегляду фотографій;
- здійснити проектування програмного забезпечення для збереження та перегляду фотографій;
- створити програмне забезпечення для збереження та перегляду фотографій;
- виконати тестування розробленого програмного забезпечення збереження та перегляду фотографій.

Розроблювана програма буде виконувати наступні функції:

- реєстрація користувачів;
- створення каталогів фотографій;
- додавання каталогів з фотографіями;
- додавання однієї чи декількох фотографій;

- видалення каталогів фотографій;
- видалення однієї чи декількох фотографій;
- синхронізація сховища з особистим персональним комп'ютером.

Всі необхідні дані будуть зберігатися у базі даних, що зробить підтримку програмного додатку збереження та перегляду фотографій більш простою та зручною.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні поняття про програмні сервіси збереження та перегляду фотографій

Служба перегляду та зберігання зображень — це програмний інструмент для відображення різноманітних зображень у цифровій формі. Такі служби повторюють основні функції простого альбому на персональний комп'ютер (ПК) [1]-[6].

Онлайн-альбоми є дуже популярним засобом в сучасному світі. Фотографія є важливою частиною спогадів, сцен, вражень і подій людей. З розвитком сучасних технологій кількість фотографій стрімко зростає, і сьогодні неможливо зробити 100 фотографій на годину, як це було 10-15 років тому [1]-[6]. Суттєве збільшення розміру вимагає термінового рішення зберегти зображення не тільки фізично, але й у цифровому вигляді [1]-[6].

Онлайн-фотоальбом дозволяє зберігати, переглядати та впорядковувати необмежену кількість фотографій у будь-який час на різних пристроях та інших пристроях [1]-[6]. Мета фотоальбому в Інтернеті — підтримувати необмежену кількість основних рутинних завдань, які виконують звичайні альбоми. Тому фотографії та альбоми вважаються об'єктами [1]-[6].

Кожен об'єкт має свої особливості: фото (назва, фото), альбом (назва, кількість фотографій, список фотографій) [1]-[6]. Таким чином, доцільно автоматизувати процес роботи з такими процесами:

- опис та характеристика зображень;
- завантаження об'єктів;
- перегляд об'єктів.

Це дійсно важливі та складні задачі, які займають багато часу у випадку ручного оброблення інформації. Найефективнішим способом вирішення цієї проблеми є автоматизація всього процесу у вигляді комп'ютерної підсистеми.

1.2 Огляд програмних засобів оброблення, збереження та перегляду зображень

Як відзначено вище, усі сучасні засоби перегляду фотографій можна використовувати, як правило, у мережі Інтернет або в звичайних автономних комп'ютерних системах [1]-[10]. Розглянемо деякі з них.

Microsoft Office Picture – це графічний редактор растрової версії, що вводитьися в Microsoft Office [7].

Основні функції редагування зображень включають правильний колір, обрізку, перевертання, зміну розміру та обертання. Щоб полегшити організацію зображень, Picture Manager включає в себе панель швидкого доступу, до якої користувачі можуть вручну або автоматично додавати ярлики командою Locate Pictures до папок ієрархічної макета файлової системи, що виключає необхідність створювати нові категорії для зображень або імпортувати їх до певного місця. Менеджер зображень (рис. 1.1) дозволяє користувачам обмінюватися фотографіями по електронній пошті [7].

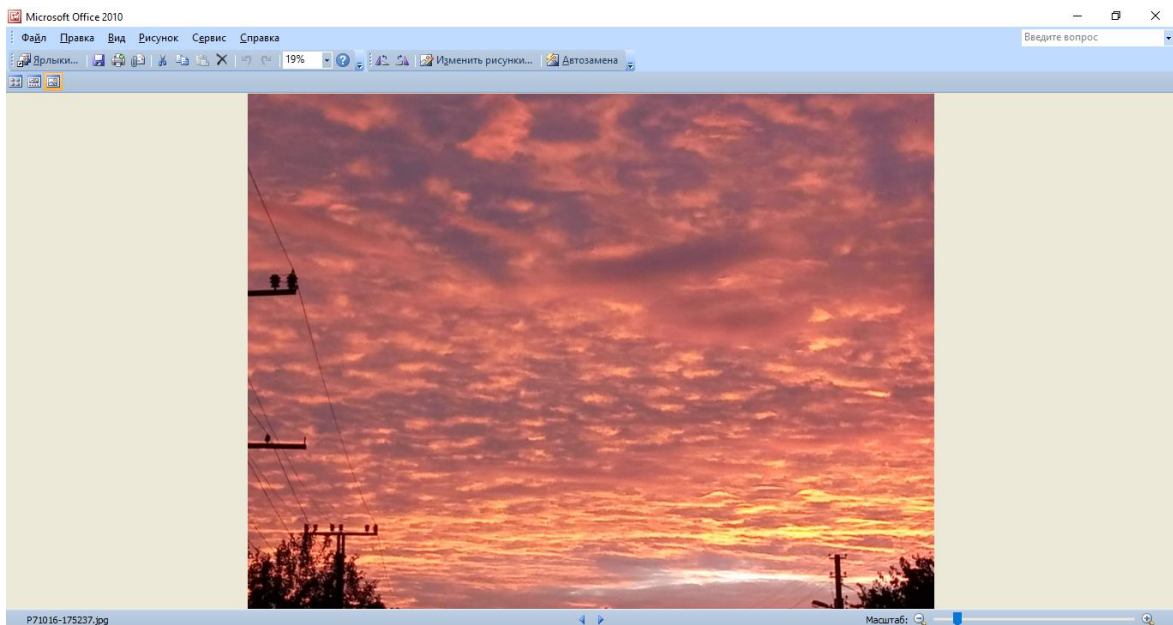


Рисунок 1.1 – Сервіс Microsoft.Picture.Manager

Менеджер – це класичний додаток від компанії Microsoft, який в цілому задовольняє невибагливого користувача, але він має свої недоліки. Microsoft Picture Manager є тільки офлайн системою і доступ до своїх фотографій можливий тільки з одного пристрою [7]. Також в додатку відсутня можливість структурування зображень за каталогами [7].

Instagram - це служба безкоштовного обміну та збереження фото та відео, що належить компанії Facebook, Inc [8]. Її створили Кевін Сістром та Майк Крігер, спочатку продукт було випущено для операційної системи iOS. Версія для пристроїв Android була пізніше, як і з обмежений інтерфейсом веб-сайт [10]. У 2016 році компанія створила також додатки для Windows 10 Mobile та Windows 10 [8].

Додаток дозволяє користувачам завантажувати фотографії та відео, які можна редагувати за допомогою різних фільтрів та організувати за допомогою тегів та інформації про місцезнаходження [8]. Публікації облікового запису можуть публікуватися публічно або з попередньо схваленими підписниками [8].

Користувачі можуть переглядати вміст інших користувачів за тегами та місцезнаходженнями, а також переглядати вміст із популярним вмістом. Користувачі можуть оцінювати фотографії та стежити за іншими користувачами [8].

Після запуску Instagram (рис. 1.2) швидко завоював популярність: мільйон зареєстрованих користувачів за два місяці, 10 мільйонів на рік, а в кінцевому підсумку - 800 мільйонів на вересень 2017 року [8]. У квітні 2012 року Facebook придбав послугу приблизно на 1 мільярд доларів США готівкою і запас. Станом на жовтень 2015 року до сервісу завантажено понад 40 мільярдів фотографій [8].

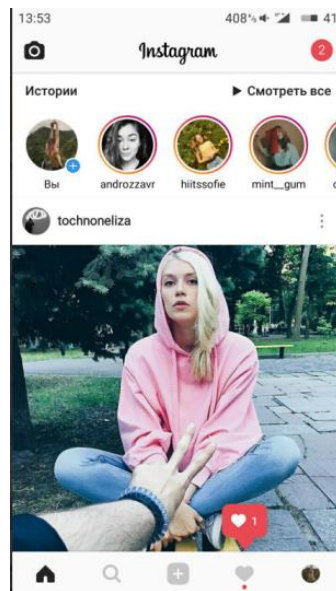


Рисунок 1.2 – Програмний продукт Instagram (головне вікно)

Зображення галереї додатка Instagram (рис. 1.3)

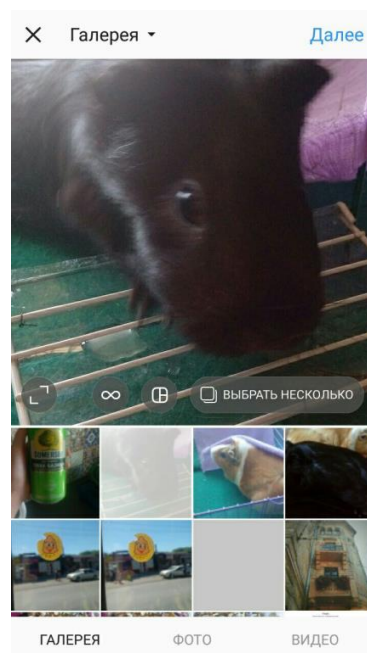


Рисунок 1.3 – Програмний продукт Instagram (галерея)

Незважаючи на фантастичну популярність даного додатку, він має два суттєвих недоліки [8]: перший – це неможливість користування додатком без мережі інтернет, тобто без підключення – актуальність даних втрачається, другий: недосконала версія додатку для ПК [8]. Ви маєте можливість

переглядати фотографії, але завантажувати не можете. Також, спільним недоліком з Microsoft.Picture.Manager є відсутність структурування зображень за каталогами.

Програмне забезпечення Canva [9] призначено для збереження, відображення та обробки зображень, та може використовуватися як професійними дизайнерами, так і для звичайними користувачами. Canva — це онлайн-платформа, яка полегшує розробку будь-чого для кожної особи та команди [9]. Завдяки тисячам професійно виготовлених шаблонів, які допоможуть розпочати роботу з зображеннями, можна створювати, налаштовувати та ділитися своїми дизайнами за кілька кліків [9]. Від наклейок до професійних брошур, приголомшливих відео до презентацій на робочому місці — все це дозволяє виконувати програмне забезпечення Canva (рис. 1.4).

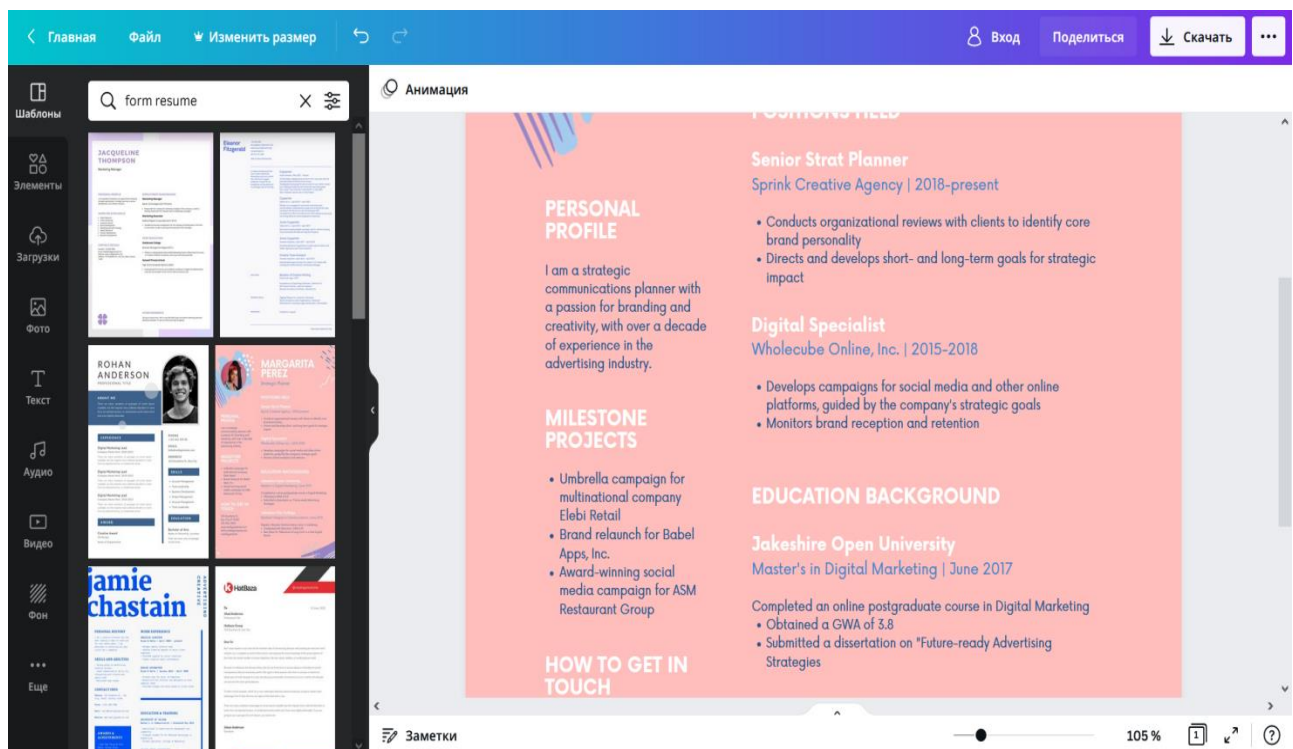


Рисунок 1.4 – Програмний продукт Canva [9]

Програмне забезпечення Canva має такі особливості та переваги[9]:

- можливості збереження, відображення та обробки зображень;
- представляє собою онлайн-платформу;

- наявність великої кількості професійно виготовлених шаблонів;
- можливість створення наклейок до професійних брошур, відео до презентацій, обробляти зображення;
- можливість додавання на фото графічних примітивів, графіки та інших зображень;
- засоби редагування текстів;
- наявність засобів роботи з аудіо та відео.

Програмне забезпечення Photo Booth Connected [10] представляє собою внутрішнє рішення, яке дозволяє операторам завантажувати фотографії або відео, додавати певні примітиви, видаляти зелені екрани тощо. Програмне забезпечення Photo Booth Connected не створює фотографії, а натомість імпортує їх з інших джерел, що робить його ідеальним супутником для існуючого програмного забезпечення [10]. Photo Booth Connected може допомогти підтримувати черги, надаючи гостям можливість увійти в систему та поділитися своїми фотографіями далеко від кабінки в іншому місці [10].

Photo Booth Connected також може працювати з наявним програмним забезпеченням як плагін для додавання соціальних мереж [10]. Фотографи також можуть використовувати програму Photo Booth Connected для iPad, Windows або Android, щоб додавати друк на місці та завантажувати соціальні мережі до своїх заходів. Photo Booth Connected також може автоматично додавати логотипи та водяні знаки, створювати фільтри, як-от чорно-білий, сепія та вінтаж, а також обробляти видалення зеленого екрана/хроматографа (рис. 1.5) [10]. Photo Booth Connected підходить для фотографів подій, щоб додати логотипи та брендинг клієнта до кожної фотографії перед тим, як вона пошириться в соціальних мережах [10]. Photo Booth Connected — це найпростіший спосіб додати соціальні мережі до наявних пропозицій продуктів [10].



Рисунок 1.5 – Програмний продукт Photo Booth Connected

Програмне забезпечення Photo Booth Connected має такі особливості та переваги [10]:

- можливості завантаження фотографії або відео;
- робота з наявним програмним забезпеченням як плагін;
- можливість роботи з різними операційними системами на різних платформах iPad, Windows або Android;
- можливість додавання логотипів та водяних знаків;
- можливість додавання на фото графічних примітивів, графіки та інших зображень;
- засоби роботи з фільтрами зображень;
- наявність засобів опитування користувачів і збору даних.

Програмне забезпечення inPixio Photo Studio [11] є засобом для роботи з зображеннями. Ця програма дозволяє повністю видаляти фон і небажані об'єкти з зображення, зокрема, знаки, дроти, люди або будь-яке зображення, яке відволікає від того, що є основним на зображенні [11]. Функція «Стерти» дозволяє видалити будь-який об'єкт або людину з зображення, зберігаючи його повну якість [11]. Інструмент вибору включає автоматичне визначення кольорів, що допомагає ретушувати фотографію. Функція Erase дозволяє

ретушувати портрети, усувати дрібні дефекти та зменшити появу зморшок. За допомогою Photo Studio можна додавати вирізи з фотографій до інших зображень для створення фотомонтажів або колажів (рис. 1.6) [11].

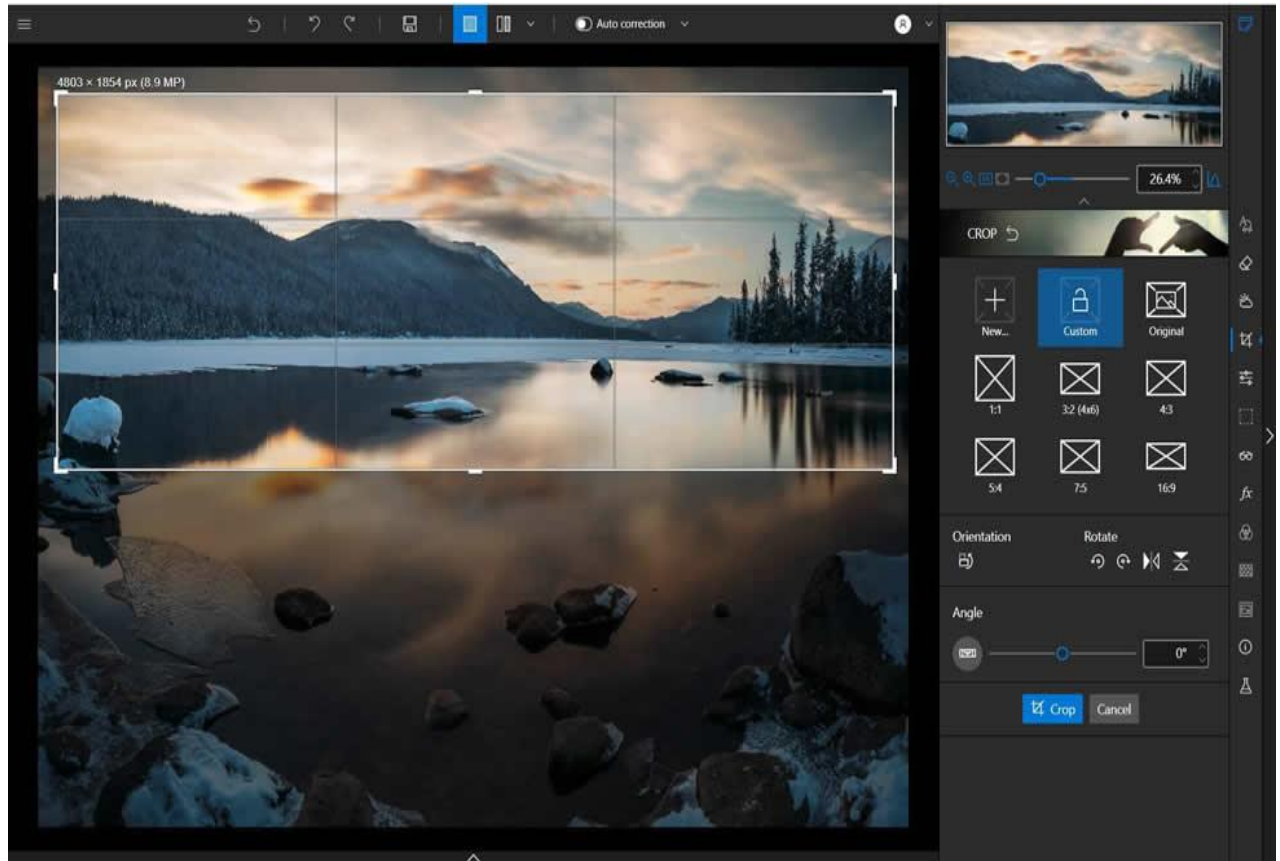


Рисунок 1.6 – Програмний продукт inPixio Photo Studio

Згладжування країв дозволяє розумно згладжувати контури для більш м'якого переходу між об'єктом і фоном для ще більш успішної інтеграції у ваш фотомонтаж [11]. Покращувати якість і кольори зображень можна досить швидко завдяки повністю автоматичному процесу та інструменту автоматичної корекції, що дозволяє вирівняти будівлі та споруди, які здаються нахиленими під час зйомки з низького кута [11].

Програмне забезпечення inPixio Photo Studio має такі особливості та переваги [11]:

- дозволяє повністю видаляти фон і небажані об'єкти з зображення;

- наявність інструментів для роботи з зображеннями, зокрема інструментів вибору;
- можливість додавання фотографій до інших зображень для створення фотомонтажів або колажів;
- можливість згладжування країв для більш м'якого переходу між об'єктом і фоном;
- можливість покращувати якість і кольори зображень;
- засоби роботи з текстовими фрагментами;
- автоматична корекція червоних очей.

Програмне забезпечення PicMonkey [12] дозволяє виконувати обробку фотографій у режимі онлайн-редактору. Засоби PicMonkey дозволяють обробляти фотографії, виконувати дизайни та публікації в соціальних мережах за допомогою фільтрів та інструментів редагування [12].

Програмне забезпечення PicMonkey може використовуватися як мобільний додаток або версія на комп'ютері [12]. Програма PicMonkey має прості у використанні інструменти для редагування фотографій [12]. Інструменти PicMonkey можуть бути застосовані для обрізання фотографій, зміни розміру, різкості, коригування кольорів та зміни експозиції [12]. Фотофільтри, які можна налаштувати для унікального вигляду, дозволяють додати до зображень певні фотоефекти [12]. Засоби PicMonkey дозволяють змінювати зовнішній вигляд камери, налаштовувати освітлення та додавати популярні ефекти лише кількома кліками [12].

Інструмент Smart Resize програми PicMonkey дозволяє швидко переформатувати зображення для різних соціальних розмірів. Інструменти графічного дизайну дозволяють створювати зображення для настільних і мобільних пристроїв [12]. Інструменти ретушування фотографій дозволяють видаляти ефекти червоних очей, виконувати аерографію, відбілювати зуби та багато іншого [12].

Програмне забезпечення PicMonkey має такі особливості та переваги [12]:

- дозволяє виконувати обробку фотографій у режимі онлайн-редактору;
- наявність інструментів для оброблення фотографій;
- можливість додавання фотографій до публікацій в соціальних мережах;
- можливість використання як мобільний додаток або версія на комп'ютері;
- можливість покращувати якість і кольори зображень;
- інструменти для редагування фотографій;
- наявність засобів обрізання фотографій, зміни розміру, різкості, коригування кольорів та зміни експозиції;
- інструменти для ретушування фотографій.

Таким чином, у наш час існує багато програмних засобів, які дозволяють зберігати, маніпулювати та переглядати фотографії. Порівняльний аналіз програмного забезпечення для збереження та перегляду фотографій наведено у табл. 1.1.

Таблиця 1.1 – Програмне забезпечення для збереження та перегляду фотографій

Критерії порівняння	Програмні засоби для збереження та перегляду фотографій				
	Microsoft Office Picture	Instagram	Canva	Photo Booth Connected	inPixio Photo Studio
1	2	3	4	5	6
Робота в онлайн-режимі	–	+	+	+	+
Наявність шаблонів	+	–	+	–	+
Робота з графічними примітивами	+	–	+	+	+

Продовження табл. 1.1

1	2	3	4	5	6
Засоби редагування текстів	+	+	+	–	+
Засіб роботи з аудіо та відео	–	+	+	+	–
Можливість роботи як плагін	–	+	–	+	–
Робота на різних ОС	–	+	–	+	+

Таким чином доцільно створити ПП, і який буде реалізовувати основні функції для комфортної роботи із переглядом зображень, це на самперед можливість реєстрації, перегляду фотографій та доступ з будь-якого пристрою. Також доцільно, щоб розроблювана програма надавала можливість створення каталогів для різних зображень.

Таким чином варто розробити програмне забезпечення збереження та перегляду фотографій, що дозволяє виконувати такі функції:

- можливість створення нових каталогів зображень;
- додавання каталогів зображень;
- реєстрація користувачів;
- синхронізація облікового запису на будь-якому ПК та отримати доступ до свої фотографій;
- видалення каталогів та зображень.

1.3 Висновки за розділом 1

Проаналізовано основні поняття про програмні сервіси збереження та перегляду фотографій. Визначено, що служба перегляду та зберігання зображень є програмним інструментом для відображення різноманітних зображень у цифровій формі, такі служби повторюють основні функції простого альбому на персональний комп'ютер. Створення програмного забезпечення для збереження та перегляду фотографій є досить актуальним завданням.

Досліджено та проаналізовано програмні засоби програмних засобів оброблення, збереження та перегляду зображень.

Визначено, що доцільно розробити програмне забезпечення, яке буде реалізовувати основні функції для комфортної роботи із переглядом зображень, це на самперед можливість створення нових каталогів зображень, додавання каталогів зображень, реєстрація користувачів, синхронізація облікового запису на будь-якому ПК та отримати доступ до своїх фотографій, видалення каталогів та зображень.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

У якості мови програмування для розробки програмного забезпечення для збереження та перегляду фотографій було обрано C# [13]-[20].

C# є об'єктно-орієнтована мова програмування із безпечною системою запису для платформи .NET. Синтаксис C# подібний до C++ і Java [13]-[20]. Мова підтримує фіксований статичний текст, поліморфізми, перевантаження операторів, функції-члени класу, властивості, події, атрибути, винятки та інтерпретацію формату XML [13]-[20]. Використання більшої кількості попередніх шаблонів, таких як C++, Delphi, Module та Smalltalk, C# виключає деякі моделі, які виявилися проблематичними при розробці програмного забезпечення на основі практики застосування, наприклад, мультигенетика (на відміну від C++) [13]-[20]. C# є дуже близьким варіантом мови програмування Java. Мова Java була розроблена компанією Sun Microsystems, коли метою глобальних обчислень було покращити Інтернет. На основі популярної мови C++ Java видала потенційно небезпечні елементи (наприклад, необмежену кількість неперевіраних покажчиків) [13]-[20].

Для розподілених обчислень була створена концепція віртуальної машини та машинно-незалежного байт-коду, який є своєрідним посередником між вихідним кодом програми та апаратними інструкціями комп'ютера чи іншого розумного пристрою [13]-[20].

Інновація C# — це можливість легше, ніж будь-коли, спілкуватися з кодом, написаним іншими мовами, що важливо для створення великих проєктів [13]-[20]. Якщо програми іншими мовами працюють на платформі .NET, .NET відповідає за сумісність програм (тобто типи даних) [13]-[20]. Сьогодні C# є флагманською мовою Microsoft, яка повністю використовує нові можливості .NET. Хоча інші мови програмування підтримуються, додатки .NET вважаються такими, що мають успадковані вразливості [13]-[20].

У таблиці 2.1 наведено порівняльну мов програмування для розробки програмного забезпечення збереження та перегляду фотографій.

Таблиця 2.1 – Порівняння мов програмування для створення ПЗ збереження та перегляду фотографій

Критерій порівняння мов програмування	Мова програмування		
	C#	C++	Python
Зручність для створення ПЗ для збереження та перегляду фотографій	+	+—	+—
Засоби обробки зображень	+	+—	+
Інструменти для роботи з файлами	+	+	+—
Обробка надзвичайних ситуацій	+	+—	+
Засоби роботи зБД	+	+—	+

Таким чином для розробки програмного забезпечення для збереження та перегляду фотографій було обрано C#.

2.2 Вибір середовища розробки програмного забезпечення

Як фреймворк-середовище розробки для створення програмного забезпечення для збереження та перегляду фотографій обрано Microsoft Visual Studio [14]-[19].

Microsoft Visual Studio — це набір продуктів Microsoft, що включає інтегроване середовище розробки програмного забезпечення та багато інших інструментів [14]-[19]. Ці продукти підтримують розробку як консольних, так і графічних програм, таких як підтримка технології Windows Forms, веб-сайтів, веб-додатків та веб-служб [14]-[19]. Він також працює з вихідним і керованим кодом для всіх платформ. Він підтримує Microsoft Windows, Windows Mobile, Windows Phone та Windows. CE — це набір продуктів

Microsoft, що включає .NET Framework, .NET Compact Framework, Microsoft Silverlight. Microsoft Visual Studio, інтегроване середовище розробки програмного забезпечення та багато інших інструментів [14]-[19]. Ці продукти дозволяють розробляти консольні та графічні програми, такі як Microsoft Windows, Windows Mobile, Windows Phone, WPF для всіх платформ, що підтримують Windows CE та новий UWP. , .NET Framework, .NET Compact Framework і Microsoft Silverlight [14]-[19].

Microsoft Visual Studio — це набір продуктів Microsoft, що включає інтегроване середовище розробки програмного забезпечення та багато інших інструментів [14]-[19]. Ці продукти дозволяють розробляти як консольні, так і графічні додатки, включаючи підтримку технології Windows Forms, веб-сайтів, веб-додатків і веб-сервісів із рідним і керованим кодом для всіх платформ [14]-[19]. Підтримується Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework і Microsoft Silverlight. Visual Studio 2017 була випущена 7 березня 2017 року. .NET Framework 4.6 включено [14]-[19]. Цей випуск вніс незначні зміни до інсталятора та інструментів розробки [14]-[19]. Його фокус зосереджений на підтримці та розробці нової кросплатформної платформи .Net Core та підтримці створення додатків на її основі [14]-[19]. Особливо це стосується створення веб-додатків бізнес-логіки на основі ASP.NET Core [14]-[19].

У таблиці 2.2 наведено результати аналізу різних середовищ розробки для створення програмного забезпечення збереження та перегляду фотографій.

Таблиця 2.2 – Порівняння середовищ розробки для створення програмного забезпечення збереження та перегляду фотографій

Критерій порівняння середовищ розробки	Середовище розробки		
	Microsoft Visual Studio	SharpDevelop	MonoDevelop
Зручність для створення ПЗ для збереження та перегляду фотографій	+	—	+—
Швидкість модифікації розроблюваного ПЗ	+	+—	+—
Можливість роботи на різних платформах	+—	—	+
Засоби роботи з репозиторіями	+	+—	—

Таким чином, як фреймворк-середовище розробки для створення програмного забезпечення для збереження та перегляду фотографій обрано Microsoft Visual Studio.

2.3 Вибір системи керування базою даних

Для створення програми збереження та перегляду фотографій як система керування базами даних обрана Microsoft SQL Server [21]-[23].

Microsoft SQL Server — це комерційна система керування базами даних, що поширюється компанією Microsoft [21]-[23]. В опитуванні використовується мова Transact-SQL, розроблена Microsoft і Sybase. Transact-SQL — це реалізація стандарту ANSI/ISO для мови структурованого пошуку SQL з розширеннями [21]-[23].

Використовується в базах даних малих і середніх, а також у базах даних великих підприємств [21]-[23]. Протягом багатьох років вона успішно конкурує з іншими системами управління базами даних [21]-[23]. Microsoft створила Entity Framework, щоб полегшити створення баз даних і керування ними. Entity Framework Core — це об'єктно-орієнтована, легка та гнучка технологія для доступу до даних Microsoft. EF Core дозволяє працювати з базами даних, але є більш високий рівень абстракції: EF Core дозволяє працювати з даними самої бази даних та її таблиць, незалежно від типу зберігання [21]-[23].

Якщо на фізичному рівні у цій системі керування базами даних іде робота з таблицями, індексами, первинними та зовнішніми ключами, то фактично виконується робота з об'єктами на рівні концепції, що рекомендовано Entity Framework [21]-[23]. За замовчуванням на даний момент Microsoft пропонує декілька встановлених постачальників послуг: MS SQL Server, SQLite та PostgreSQL [21]-[23]. Щоб зробити MS SQL Server простим у використанні, компанія розробила потужну програму під назвою SQL Server Management Studio (SSMS), що є першим програмним інструментом, випущеним у Microsoft SQL Server, який використовується для налаштування, керування та керування всіма компонентами Microsoft SQL Server [21]-[23]. Цей інструмент включає в себе об'єкт сервера, редактор сценаріїв і графічні інструменти, які працюють із можливостями [21]-[23].

У таблиці 2.3 наведено порівняння СКБД для розробки програмного забезпечення збереження та перегляду фотографій.

Таблиця 2.3 – Порівняння СКБД для розробки програмного забезпечення збереження та перегляду фотографій

Критерій порівняння СКБД	СКБД		
	Microsoft SQL Server	MySQL	MongoDB
Зручність для розробки ПЗ збереження та перегляду фотографій	+	+—	+—
Наявність літератури по використанню СКБД	+	+	+—
Засоби SSMS для налаштування БД	+	—	—
Можливість роботи за клієнт-серверною технологією	+	+	+—

Таким чином для створення програмного забезпечення збереження та перегляду фотографій обрано систему управління базами даних Microsoft SQL Server.

2.4 Висновки за розділом 2

Для створення програмного забезпечення збереження та перегляду фотографій було обрано мову програмування C#, яка є об'єктно-орієнтованою мовою програмування із безпечною системою запису для платформи .NET.

Як фреймворк-середовище розробки для створення програмного забезпечення для збереження та перегляду фотографій обрано Microsoft Visual Studio.

Для створення програми збереження та перегляду фотографій як СКБД обрано Microsoft SQL Server.

3 ОПИС ПРОГРАМИ

3.1 Структурна схема програми

Структуру розробленого програмного забезпечення для збереження та перегляду фотографій наведено на рис. 3.1.



Рисунок 3.1 – Структура програмного **забезпечення для збереження** та перегляду фотографій

Клієнтська частина ПЗ для збереження та перегляду фотографій відповідає за відображення даних та здійснення маніпулювань в доступному виді для користувача. Представляє собою графічний інтерфейс програм через, який користувач взаємодіє з нею. При здійсненні певних дій (додавання, видалення т.п), за допомогою запитів звертається до API програмного додатку. Після отримання відповіді, відображає результат користувачеві.

Web API відповідає за зв'язування запитів з клієнта до серверу ПЗ для збереження та перегляду фотографій. Роль API ПЗ для збереження та перегляду фотографій можливо порівняти з деяким мостом, який за певним запитом клієнта намагається отримати дані з серверу бази даних, отримавши дані про фотографії він повертає їх до клієнту.

Серверна частина ПЗ для збереження та перегляду фотографій представляє собою частину, на якій зберігається база даних про фотографії клієнтів та налаштування до неї. База даних про фотографії клієнтів зберігає інформацію користувачів, які при взаємодії з клієнтом програмного забезпечення можуть її отримати.

У таблиці 3.1 наведено опис компонентів реалізованого ПЗ для збереження та перегляду фотографій з визначенням їх основних функцій.

Таблиця 3.1 – Опис компонентів реалізованого ПЗ для збереження та перегляду фотографій

Назва компоненту	Призначення компоненту
Клієнтська частина	Відображення даних. Виведення на екран графіки (користувацькі форми, елементи керування, графічне оформлення)
Web API	Виконує зв'язну функцію між клієнтом та сервером. Оброблювання подій (запитів), видавання результату (відповідь)
Серверна частина	Містить базу даних. Робота з даними бази даних

3.2 Функціонування програми збереження та перегляду фотографій

Програма збереження та перегляду фотографій заснована на архітектурному підході API REST, найпоширенішим типом якого є програми на основі системи клієнт-сервер. Архітектура клієнт-сервер є однією з моделей архітектури програмного забезпечення і є домінуючою концепцією у розробці розподілених мережових додатків, які передбачають взаємодію та обмін даними між ними [21]-[23]. Він включає в себе наступні основні компоненти: набір серверів, які надають інформаційні та інші послуги програмному

забезпеченню, яке здійснює доступ до них; набір клієнтів, що користуються послугами сервера; мережа, яка забезпечує взаємодію клієнт-сервер [21]-[23].

Сервери незалежні один від одного. Клієнти працюють пліч-о-пліч незалежно один від одного. Клієнти не прив'язані до сервера [21]-[23]. Найчастіше сервер обробляє запити від різних клієнтів одночасно; з іншого боку, клієнт може отримати доступ до того чи іншого сервера [21]-[23]. Клієнти повинні знати про доступні сервери, але інші клієнти можуть не знати, що це таке [21]-[23].

Під час розробки програмного забезпечення для збереження та перегляду фотографій було визначено такі модулі системи: клієнтська частина – користувач виконує дії, які в свою чергу запитують систему API і отримують відповідь, виводячи інформацію на екран; Web API, завдяки якому клієнт підключається до сервера та може отримувати клієнтські запити, обробляти їх у зв'язку з базою даних та надсилати відповіді; сервер, що містить базу даних і отримує запити від Web API.

Схему функціонування клієнтської частини програмного забезпечення для збереження та перегляду фотографій зображено на рис. 3.2.

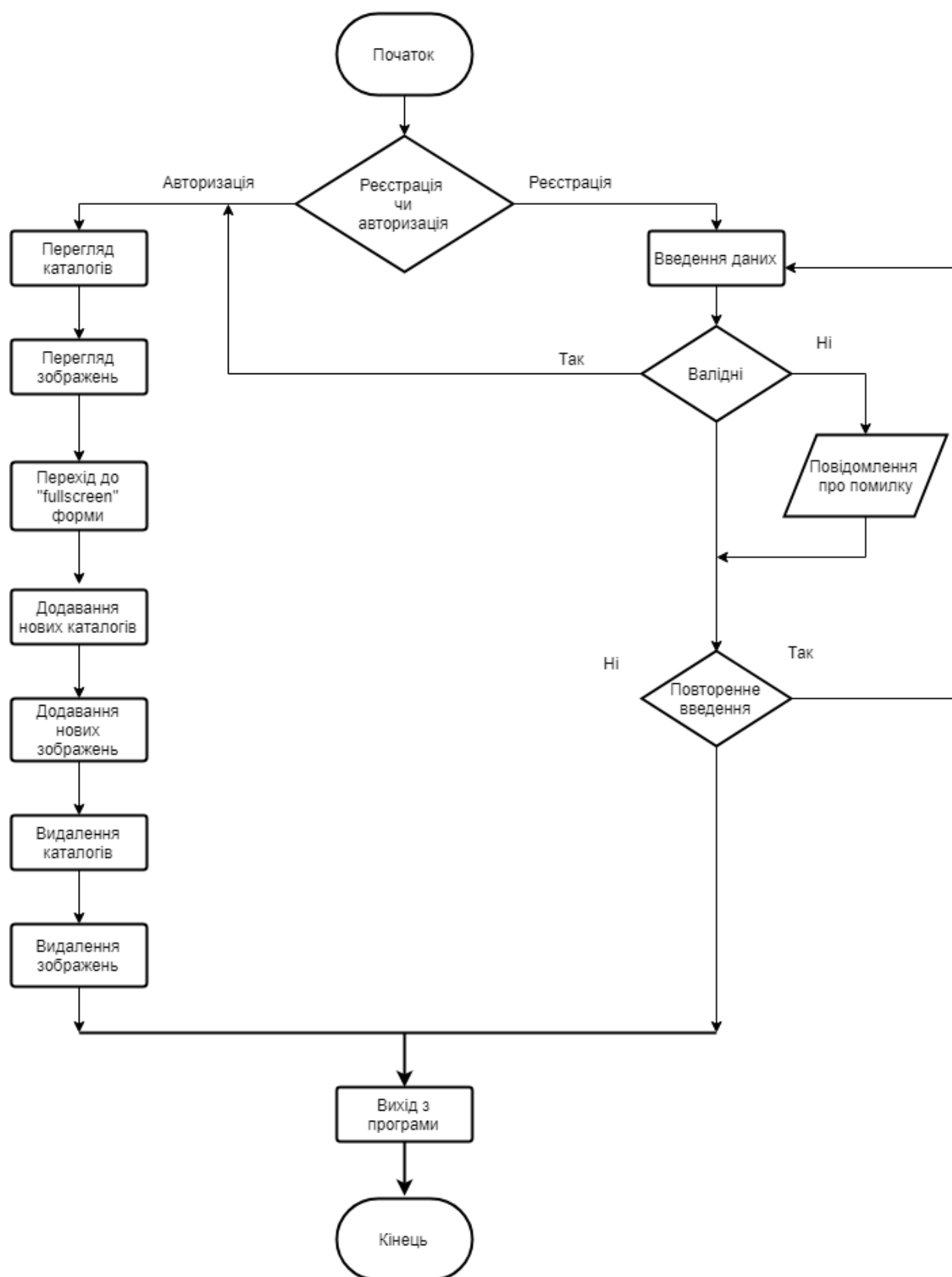


Рисунок 3.2 – Схема функціонування клієнтської частини програмного забезпечення для збереження та перегляду фотографій

3.3 Опис функцій програмного забезпечення

Авторизація відбувається за допомогою клієнту. Користувач заповнює поля email та пароль і при коректності даних заходить до облікового запису. В процесі прихованому від ока користувача, відбувається запит, який оброблює API, до серверу з веденими даними, на якому відбувається остаточна валідація та перевіряється у базі даних наявність користувача з такими параметрами. В разі помилок, користувачеві виводиться повідомлення про не валідність даних, тобто, введені ним дані були невірними.

Реєстрація винесана на сторінку у браузері з метою спрощення цього процесу (отримання даних з веб форм простіше оброблювати, ніж з Desktop-клієнту). Користувач вводить 3 необхідні поля (пошта, пароль та підтвердження паролю), в разі коректності даних, користувачеві повідомляється про успішну реєстрацію. В разі помилок при введенні необхідних даних, помилки відображається на формі реєстрації.

При вході у програму та наявності в ній вже створених каталогів, користувач має можливість перемикаючи кожен з каталогів переглядати зображення наявні в ньому, а також настигнувши двічі на зображень, відкрити форму «FullScreen», яка дозволяє переглядати фотографій в більшому розмірі по одинці в вибраному каталозі.

Додати каталог можливо двома способами.

Перший, це завантажити вже існуючу папку з ПК користувача. Під час завантаження, на сервері буде створено каталог з назвою цієї папки та завантажено до нього всі фотографії які містяться у цій папці на підрівні (subfile), це означає , якщо папка містить інші файли, чи ще якісь папки вони будуть проігноровані.

Другий спосіб, це наставши на навігаційному меню варіант «Addfolder», створити нову, чисту папку. Після натискання, для користувача з'явиться діалогове вікно для ведення назви папки. Після успішного ведення

та натиснення згоди на створення папки, папка будет створена на сервері, а також відображена на екрані.

Видалення каталогів відбувається за вибором користувача. Користувач обирає каталог та обирає пункт меню видалення. Каталог видаляється з бази даних та сховища на сервері, якщо каталог містить фотографії, вони видаляються разом з ним.

Видалення фотографій відбувається подібно видаленню каталогів, але з можливістю видалення декількох фотографій за один раз. Користувач має можливість обрати декілька зображень та натиснути видалення. Після цього, фотографії будуть видалені з бази даних та сховища на сервері.

Для більш зручного управління та користування розробленим ПП, більшість алгоритмів займають відповідні користувацькі форми, які описані у табл. 3.2.

Таблиця 3.2 – Користувацькі форми для роботи з програмою збереження та перегляду фотографій

Користувацька форма	Опис
SignUp.cs	Форма авторизації програми збереження та перегляду фотографій. Можливість авторизації, або перехід до реєстрації.
UserPhotos.cs	Форма користувача програми збереження та перегляду фотографій. Відображення каталогів та зображень у них, виконання маніпуляцій над ними.
FullScreen.cs	Перегляд каталогу програми збереження та перегляду фотографій у режимі “fullscreen”

3.4 База даних

Для повноцінної роботи програми збереження та перегляду фотографій також була розроблена БД, схема якої зображена на рис. 3.3.

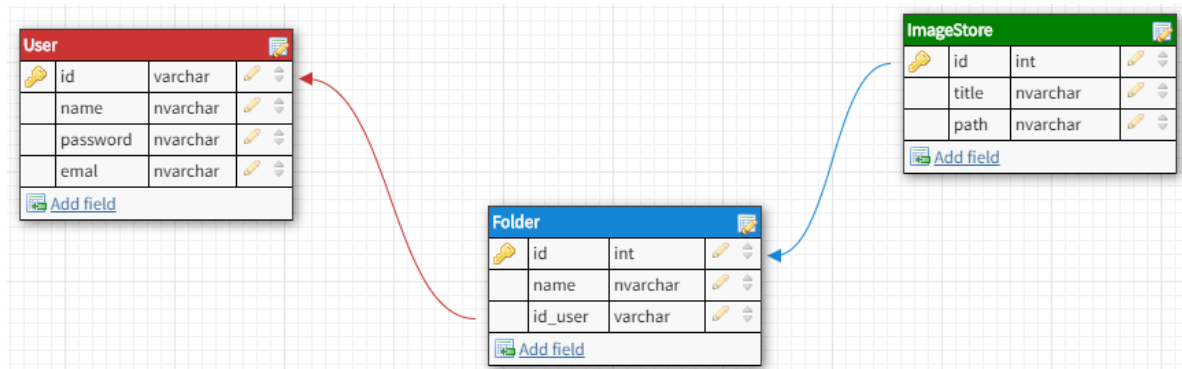


Рисунок 3.3 – Схема розробленої БД

База даних програмни налічує три таблиці:

- таблиця User відповідає за зберігання зареєстрованих користувачів у програмі. Таблиці містить такі поля (id, name, password, email);
- таблиця Folder відповідає за зберігання каталогів для користувача. Таблиця містить такі поля (id, name, id_user);
- таблиця ImageStore відповідає за зберігання зображень. Таблиця містить так і поля (id, title, path, id_folder).

3.5 Висновки за розділом 3

Розроблено структуру розробленого програмного забезпечення для збереження та перегляду фотографій. Визначено, що розроблене ПЗ складається з клієнтської частини, що відповідає за відображення даних та здійснення маніпулювань в доступному виді для користувача, Web API, яке відповідає за зв'язування запитів з клієнта до серверу ПЗ для збереження та перегляду фотографій, та серверної частини, на якій зберігається база даних про фотографії клієнтів та налаштування до неї.

Розроблено схему функціонування клієнтської частини програмного забезпечення для збереження та перегляду фотографій. Описано основні функції програмного забезпечення збереження та перегляду фотографій.

Розроблено базу даних для повноцінної роботи програми збереження та перегляду фотографій, яка складається з таблиць User, Folder та Image Store.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для перегляду зображень, через комп'ютер користувача та локальну мережу. Програма написана на мові C#. База даних розроблена в СКБД Microsoft SQL Server.

Програмні засоби обліку земельних ділянок забезпечують виконання функцій:

- реєстрація користувачів;
- створення каталогів фотографій;
- додавання каталогів з фотографіями;
- додавання однієї чи декількох фотографій;
- видалення каталогів фотографій;
- видалення однієї чи декількох фотографій;
- синхронізація сховища з особистим персональним комп'ютером.

Для нормального функціонування програми збереження та перегляду фотографій повинні виконуватися такі умови до апаратної частини:

- наявність персонального компютера (ПК), що виконує роль сервера;
- процесорі з тактовою частотою 2 GHz та більше;
- вільний простір на жорсткому диску не менше, ніж 1 ГБ;
- оперативна пам'ять 1Гб і вище;
- монітор, клавіатура та маніпулятор типу “миша”.

Умови до програмної частини, для нормального функціонування програми збереження та перегляду фотографій:

- наявність операційної системи(ОС) Windows;
- наявність на ПК .Net Framework;
- наявність на ПК Microsoft SQL Server.

Функціональні характеристики розробленого програмного забезпечення збереження та перегляду фотографій відповідають технічному завданню (додаток А). Опис програми збереження та перегляду фотографій та текст програми збереження та перегляду фотографій розташовано у додатках Б та В.

4.2 Характеристики програми

Програма збереження та перегляду фотографій виконана за допомогою мови програмування C# та середовища розробки Microsoft Visual Studio. В якості системи керування БД обрана система Microsoft SQL Server. Програма збереження та перегляду фотографій базується на комп'ютері. Працює програма збереження та перегляду фотографій постійно. Після оновлення функціоналу програми збереження та перегляду фотографій треба перезапустити сервер, де розташована база даних.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для завантаження програми необхідно завантажити інсталятор (архів AppPackages) та за допомогою PowerShell запустити файл Add-AppDevPackage.ps (рис. 4.1).

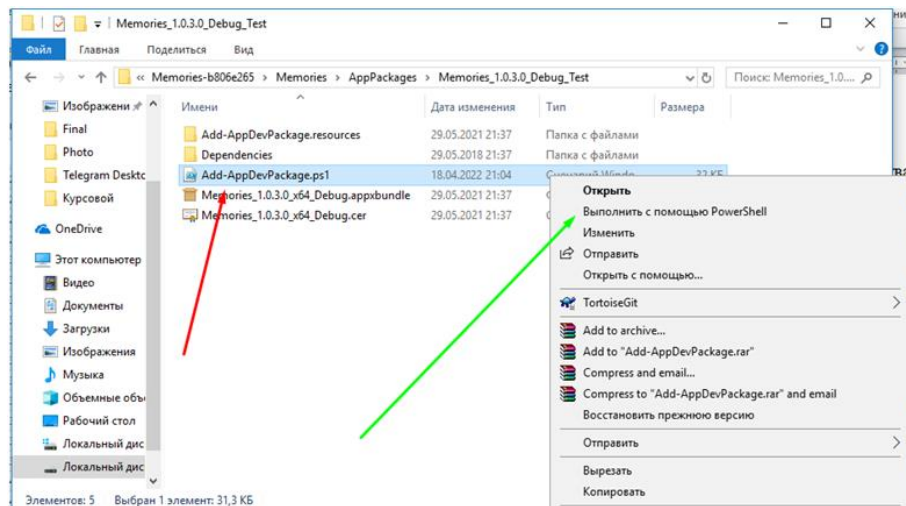


Рисунок 4.1 – Завантаження програми

4.3.2 Вхідні й вихідні дані

Вхідними даними до програми збереження та перегляду фотографій є інформація яку вводить користувач за допомогою графічного інтерфейсу програми. Вони можуть бути: текстові (пароль, логін, назва папки) та тип фотографії (bmp, jpg, png).

Вихідними даними програми збереження та перегляду фотографій є інформація яку отримує користувач перед очима на графічному інтерфейсі. Вихідними даними можуть бути текстові поля, контейнери із фотографіями та каталогами. Також вихідними даними можуть вважатися діалогові окна повідомлень програми.

4.3.3 Повідомлення

Розроблена програма збереження та перегляду фотографій забезпечує коректне функціонування на різних платформах. При некоректному заповненні даних у програмі збереження та перегляду фотографій відповідне повідомлення (табл. 4.1).

Таблиця 4.1 – Основні повідомлення при роботі з програмою збереження та перегляду фотографій

Повідомлення	Опис повідомлення
Info message: <i>Login or Password were not correct</i> Ok	При авторизації у програмі збереження та перегляду фотографій було не правильно введено логін або пароль.
Info message: <i>You load last sync data on your pc. If the folders and pictures are different, you can: 1) Sync select folder (its faster) 2) Sync all data (its takes long time)</i> Ok	Повідомлення про місце завантаження фотографій до програми збереження та перегляду фотографій
Info message: <i>Please, select folder for delete</i> Ok	Помилка видалення не обраного каталогу (папки) програмою збереження та перегляду фотографій
Info message: <i>Please select photo for delete.</i> Ok	Помилка видалення не обраного файлу програмою збереження та перегляду фотографій

4.4 Тестування та результати роботи програмного забезпечення збереження та перегляду фотографій

Тестування роботи програмного забезпечення збереження та перегляду фотографій буде проведене за таким сценарієм:

- незареєстрований користувач роботи програмного забезпечення збереження та перегляду фотографій виконує реєстрацію;
- якщо, програма збереження та перегляду фотографій працює коректно, то користувач отримає відповідне повідомлення та переде до процедури реєстрації;

- в ході процедури реєстрації, користувач програмного забезпечення збереження та перегляду фотографій зробить помилки;
- після процедури реєстрації користувач програмного забезпечення збереження та перегляду фотографій увійде до облікового запису та спробує створити каталог, додати до нього фотографії, видалити каталог чи фотографії, завантажити каталог;
- після цього користувач програмного забезпечення збереження та перегляду фотографій вийде з програми і знову виконає вхід;
- завантажений каталог повинен успішно відобразитись на екрані;
- натиснути двічі на одне із зображень, щоб переглянути його та інші у вікні «FullScreen».

На початку тестування користувач просто спробав увійти до програмного забезпечення збереження та перегляду фотографій, не вводячи логін та пароль. Виникла помилка, яка зображена на (рис. 4.2).

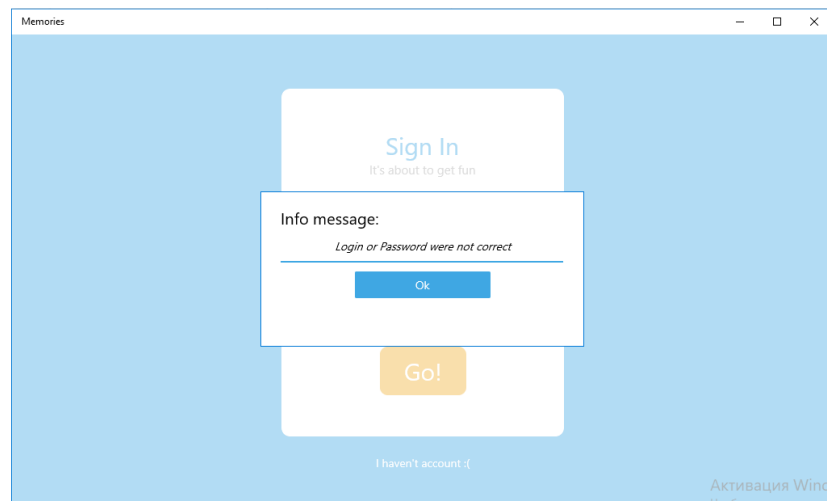
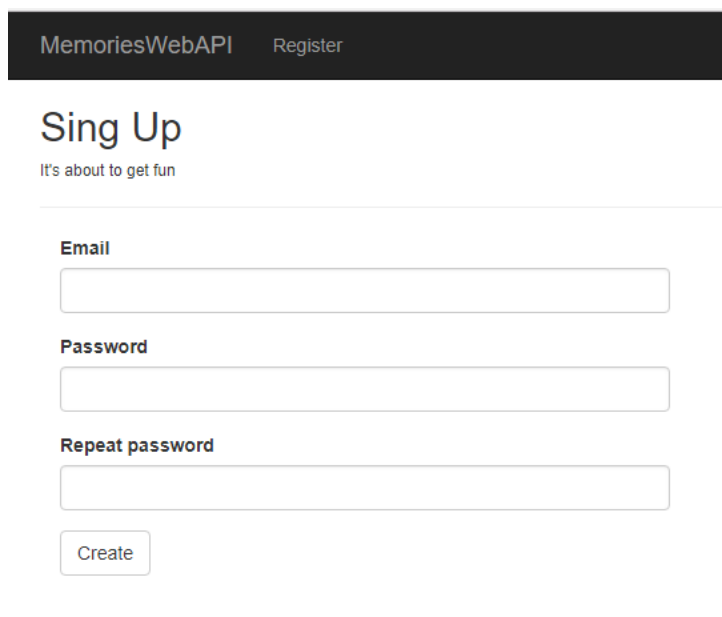


Рисунок 4.2 – Помилка при авторизації

Після цього користувач перейшов, до реєстрації, це зображено на рис. 4.3)



The screenshot shows a web browser window with a dark header bar containing the text 'MemoriesWebAPI' and a 'Register' link. Below the header, the main heading 'Sing Up' is displayed in a large, bold font, followed by the tagline 'It's about to get fun' in a smaller font. The registration form consists of three input fields labeled 'Email', 'Password', and 'Repeat password', each with a corresponding text box. Below these fields is a 'Create' button. The form is enclosed in a light gray border.

Рисунок 4.3 – Реєстрація

Після реєстрації користувач увійшов до аккаунту програмного забезпечення збереження та перегляду фотографій, це демонструється на **рисунку 4.4.**

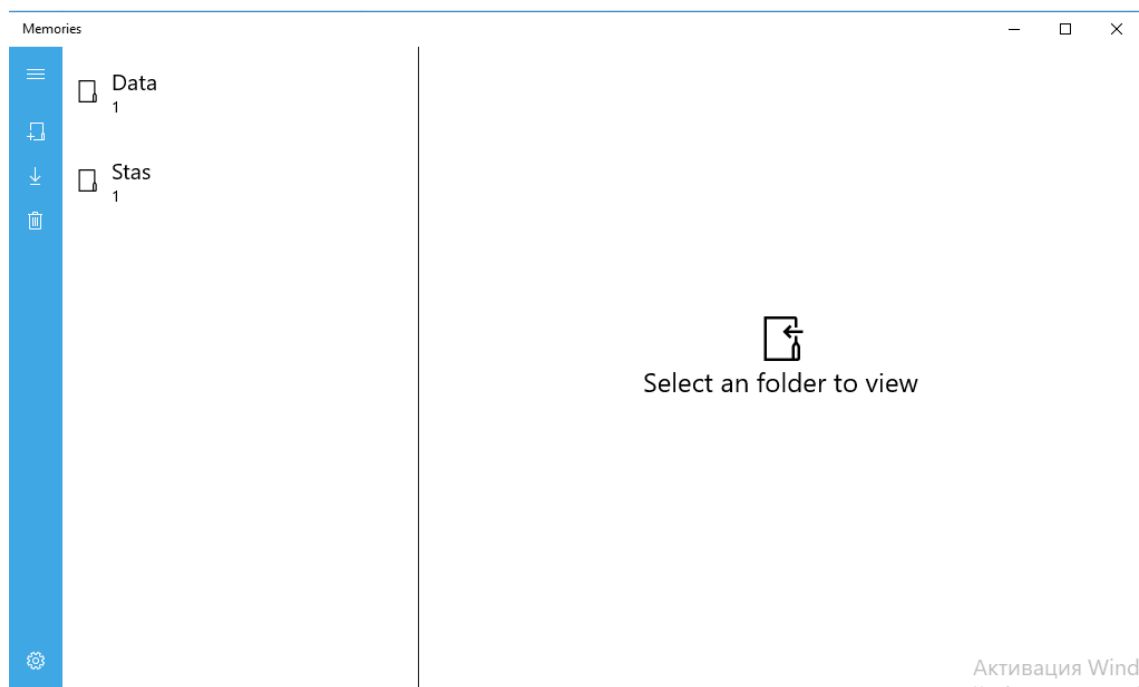


Рисунок 4.4 – Вхід авторизованого користувача програмного забезпечення збереження та перегляду фотографій

Створення нової папки, з назвою, наприклад, Birthday18, зображено на рис. 4.5.

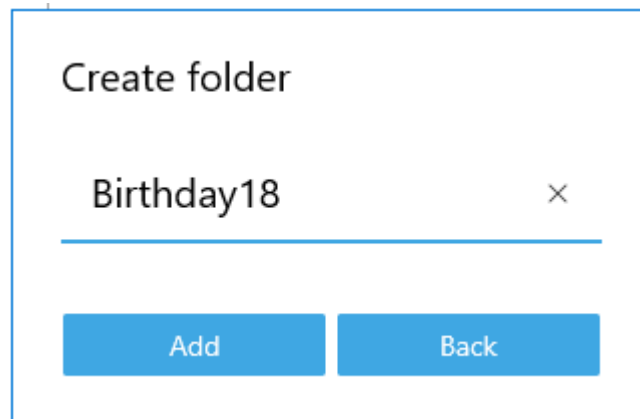


Рисунок 4.5 – Створення папки

Відображення папки у головному вікні наведено на рис. 4.6.

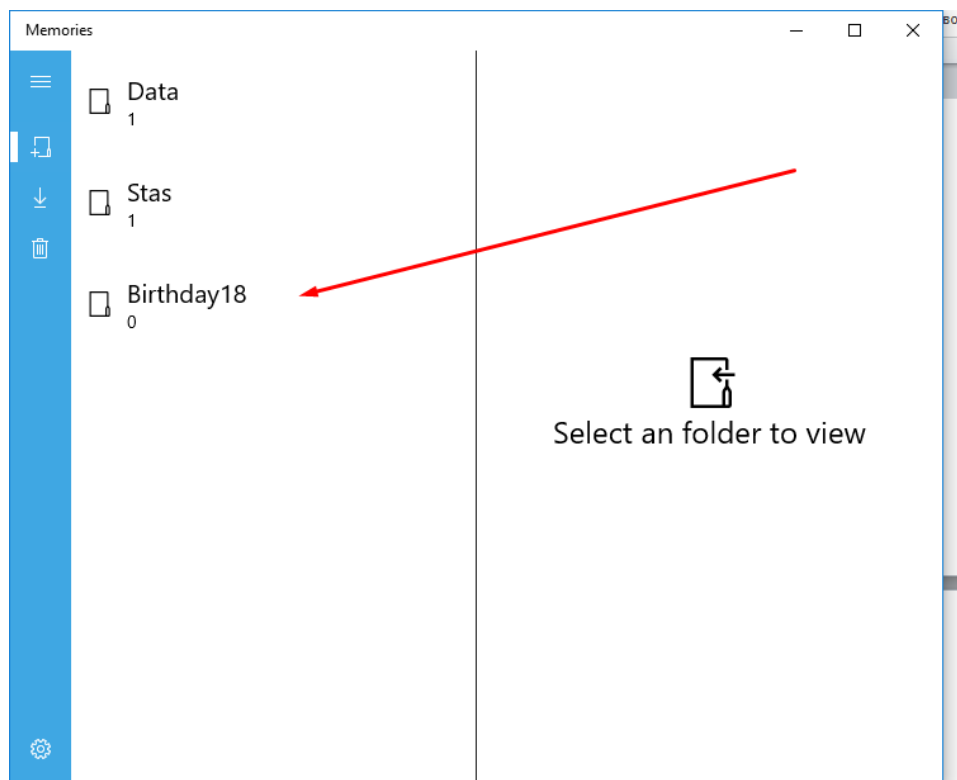


Рисунок 4.6 – Відображення створеної папки в аккаунті користувача

Виконаємо додавання деяких фотографій до створеного каталогу (результат наведено на рис. 4.7).

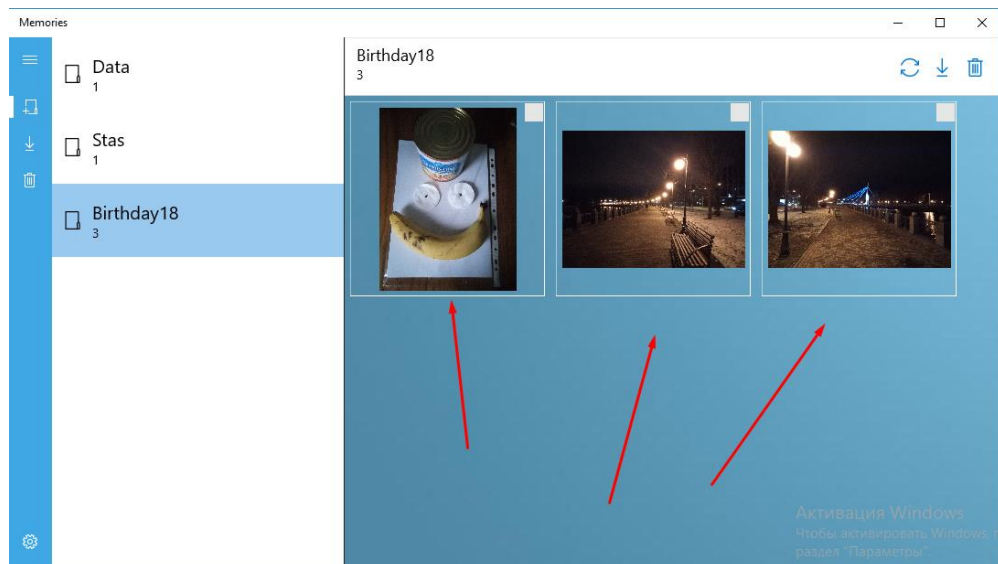


Рисунок 4.7 – Додані фотографії до папки

Попередні кроки були успішні, тому користувач вийшов з програми. Наступним кроком ми про симулюємо вхід користувача з іншого комп'ютера.

Правильною поведінкою програми буде така послідовність відображення та дій:

- спочатку користувач при вході побачить тільки папки (вони будуть відображати кількість фотографій, але самих фотографій в них не буде, це рішення було прийнято з точки зору оптимізації синхронізації фотографій), що зображено на рис. 4.8;

- після цього, користувач обере, наприклад, створену раніше папку Birthday18 та натисне синхронізувати (рис. 4.9);

- програма повинна завантажити та автоматично відобразити фотографії на екрані (рис. 4.10).

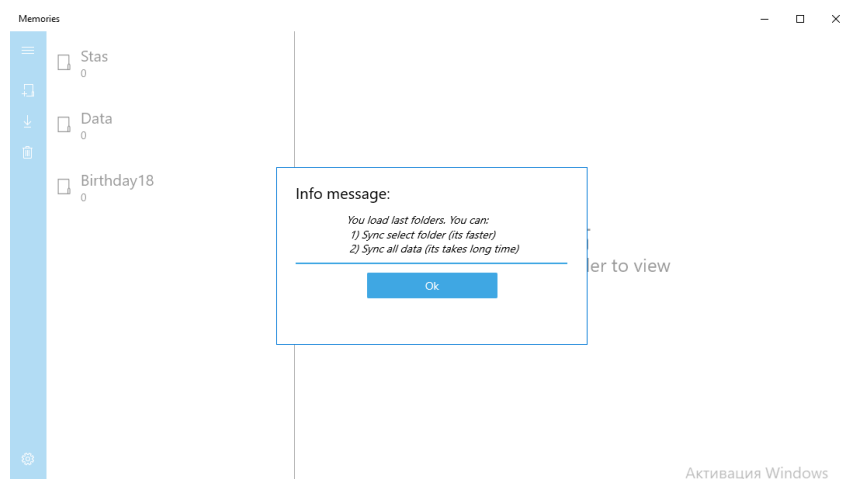


Рисунок 4.8 – Головна форма програми при вході з іншого комп'ютеру

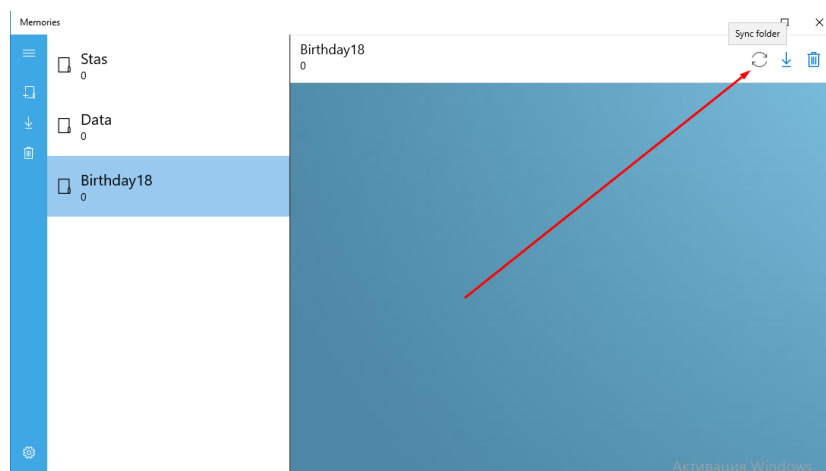


Рисунок 4.9 – Кнопка синхронізації фотографій

Відображення синхронізованих фотографій наведено на рис. 4.10.

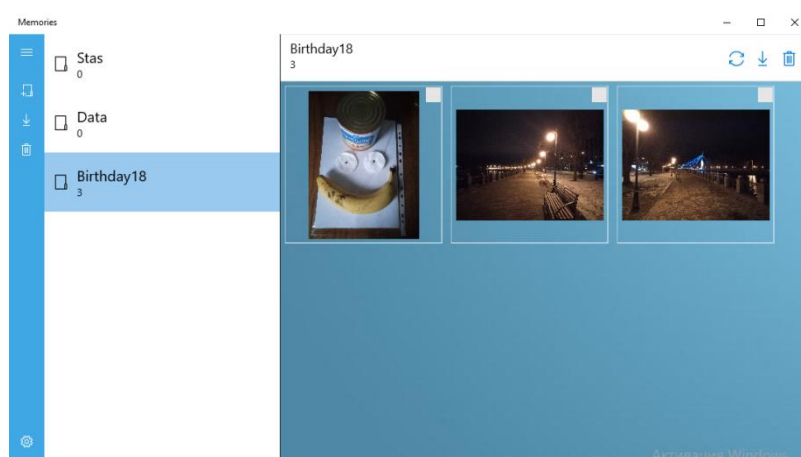


Рисунок 4.10 – Відображення синхронізованих фотографій

Результати тестування програмного забезпечення збереження та перегляду фотографій підтвердили його працездатність та можливість використання за призначенням.

4.5 Висновки за розділом 4

Описано програмне забезпечення збереження та перегляду фотографій. Виконано тестування розробленого програмного забезпечення обліку збереження та перегляду фотографій. Результати тестування програмного забезпечення збереження та перегляду фотографій підтвердили його працездатність та можливість використання за призначенням.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для збереження та перегляду фотографій.

Проаналізовано основні поняття про програмні сервіси збереження та перегляду фотографій. Визначено, що служба перегляду та зберігання зображень є програмним інструмент для відображення різноманітних зображень у цифровій формі, такі служби повторюють основні функції простого альбому на персональний комп'ютер. Створення програмного забезпечення для збереження та перегляду фотографій є досить актуальним завданням.

Визначено, що доцільно розробити програмне забезпечення, яке буде реалізовувати основні функції для комфортної роботи із переглядом зображень, це на самперед можливість створення нових каталогів зображень, додавання каталогів зображень, реєстрація користувачів, синхронізація облікового запису на будь-якому ПК та отримати доступ до свої фотографій, видалення каталогів та зображень.

Для створення програмного забезпечення збереження та перегляду фотографій було обрано мову програмування C#, яка є об'єктно-орієнтованою мовою програмування із безпечною системою запису для платформи .NET.

Як фреймворк-середовище розробки для створення програмного забезпечення для збереження та перегляду фотографій обрано Microsoft Visual Studio.

Для створення програми збереження та перегляду фотографій як СКБД обрано Microsoft SQL Server.

Розроблено структуру розробленого програмного забезпечення для збереження та перегляду фотографій. Визначено, що розроблене ПЗ складається з клієнтської частини, що відповідає за відображення даних та здійснення маніпулювань в доступному виді для користувача, Web API,

яке відповідає за зв'язування запитів з клієнта до серверу ПЗ для збереження та перегляду фотографій, та серверної частини, на якій зберігається база даних про фотографії клієнтів та налаштування до неї.

Розроблено схему функціонування клієнтської частини програмного забезпечення для збереження та перегляду фотографій. Описано основні функції програмного забезпечення збереження та перегляду фотографій.

Розроблено базу даних для повноцінної роботи програми збереження та перегляду фотографій, яка складається з таблиць User, Folder та ImageStore.

Виконано тестування розробленого програмного забезпечення обліку збереження та перегляду фотографій. Результати тестування програмного забезпечення збереження та перегляду фотографій підтвердили його працездатність та можливість використання за призначенням.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Графічні редактори зображень. Особливості роботи [Електрон. ресурс]. – Режим доступу : <https://hi-news.pp.ua/kompyuteri/181-grafchn-redaktori-zobrazhen-osoblivost-roboti.html>.
2. Програми для роботи із зображеннями [Електрон. ресурс]. – Режим доступу : <http://um.co.ua/3/3-7/3-78555.html>.
3. Редактори графічні та текстові [Електрон. ресурс]. – Режим доступу : <https://www.pharmencyclopedia.com.ua/article/1054/redaktori>.
4. Кращі фоторедактори для Windows, macOS, iOS та Android [Електрон. ресурс]. – Режим доступу : https://blog.allo.ua/ua/krashhi-fotoredaktori-dlya-windows-macos-ios-ta-android_2018-05-12/.
5. 10 застосунків для роботи з Instagram-stories [Електрон. ресурс]. – Режим доступу : <https://internews.ua/opportunity/instagram-stories-apps>.
6. Які є програми для перегляду картинок і фотографій? [Електрон. ресурс]. – Режим доступу : <http://pro-computer.pp.ua/4844-yak-ye-programi-dlya-pereglyadu-kartinok-fotografy.html>.
7. Microsoft.Picture.Manager [Electronic resource]. – Access mode : <https://support.office.com/en-us/article/where-is-picture-manager-58837c3e-34db-4904-95e8-4eca7b7c5730>.
8. Instagram [Electronic resource]. – Access mode : <https://www.instagram.com/>.
9. Canva [Electronic resource]. – Access mode : <https://www.canva.com>.
10. Photo Booth Connected [Electronic resource]. – Access mode : <https://www.photoboothsolutions.com>.
11. inPixio Photo Studio [Electronic resource]. – Access mode : <https://www.inpixio.com/>.
12. PicMonkey [Electronic resource]. – Access mode : <https://www.picmonkey.com/>.

13. Joseph A. C# inNutshell[Text] / Albahari Joseph. – O'Reilly Media, 2017. – 1133р.
14. Симан М. Внедрение зависимостей в .Net [Текст] / Марк Симан. – СПб.: Питер, 2013. – 663с.
15. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7: [Электрон. ресурс]. – Режим доступа : <https://metanit.com/sharp/tutorial/>.
16. Руководство по ASP.NETCore 2.0: [Электрон. ресурс]. – Режим доступа : <https://metanit.com/sharp/aspnet5/>.
17. Уроки програмування з нуля. Сі-шарп для чайників С: [Электрон. ресурс]. – Режим доступа : <http://mycsharp.ru/>.
18. Руководство по EntityFrameworkCore: [Электрон. ресурс]. – Режим доступа : <https://metanit.com/sharp/entityframeworkcore/>
19. Introduction to ASP.NET Core: [Электрон. ресурс]. – Режим доступа : <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.0>.
20. Universal Windows Platform documentation [Электрон. ресурс]. – Режим доступа : <https://docs.microsoft.com/en-us/windows/uwp/index>.
21. Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others” [Electronic resource]. – Access mode : <https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/>.
22. MySQL Documentation [Electronic resource]. – Access mode : <https://www.mysql.com>.
23. API MySQL [Electronic resource]. – Access mode : <https://www.php.net/manual/ru/book.mysql.php>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення збереження та перегляду фотографій може використовуватися для роботи з фотографіями.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для збереження та перегляду фотографій», затверджене наказом Національного університету «Запорізька політехніка» № 102 від 27 квітня 2022 р.

А.2 Призначення розробки

Програмний продукт для збереження та перегляду фотографій призначений для полегшення роботи користувачів різних професій при роботі з фотографічними зображеннями. Програмний продукт спрямований на зберігання, перегляд, завантаження фотографій як через комп'ютер користувача, так і через мережу.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для збереження та перегляду фотографій повинно забезпечувати виконання функцій:

- реєстрація користувачів;
- створення каталогів фотографій;
- додавання каталогів з фотографіями;
- додавання однієї чи декількох фотографій;
- видалення каталогів фотографій;

- видалення однієї чи декількох фотографій;
- синхронізація сховища з особистим персональним комп'ютером.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програми для збереження та перегляду фотографій повинен дозволяти користувачу взаємодіяти з програмою у зручний для користувачів спосіб.

A.3.3 Вимоги до надійності

Програмне забезпечення для збереження та перегляду фотографій повинно забезпечити обробку критичних ситуацій та забезпечити можливість надійної роботи у режимі 24/7.

A.3.4 Умови експлуатації

Програма для збереження та перегляду фотографій базується на комп'ютері. Програмне забезпечення для збереження та перегляду фотографій працює 24 години на добу, сім днів на тиждень. Після оновлення програмного забезпечення для збереження та перегляду фотографій необхідно перезапустити сервер.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для нормального функціонування програми збереження та перегляду фотографій повинні виконуватися такі умови до апаратної частини:

- наявність персонального компютера (ПК), що виконує роль сервера;
- процесор із тактовою частотою 2 GHz та більше;

- вільний простір на жорсткому диску не менше, ніж 1 ГБ;
- оперативна пам'ять 1Гб і вище;
- монітор, клавіатура та маніпулятор типу “миша”.

Умови до програмної частини, для нормального функціонування програми збереження та перегляду фотографій:

- наявність операційної системи(ОС) Windows;
- наявність на ПК .Net Framework;
- наявність на ПК Microsoft SQL Server.

А.3.6 Вимоги до маркування і пакування

Програма обліку збереження та перегляду фотографій може бути записана на носій інформації. На упаковці повинна бути назва програми – «Програмне забезпечення для збереження та перегляду фотографій».

А.4 Вхідні дані до роботи

Вхідними даними до програми збереження та перегляду фотографій є інформація яку вводить користувач за допомогою графічного інтерфейсу програми. Вони можуть бути: текстові (пароль, логін, назва папки) та тип фотографії (bmp, jpg, png).

Вихідними даними програми збереження та перегляду фотографій є інформація яку отримує користувач перед очима на графічному інтерфейсі. Вихідними даними можуть бути текстові поля, контейнери із фотографіями та каталогами. Також вихідними даними можуть вважатися діалогові окна повідомлень програми.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Програмне забезпечення для збереження та перегляду фотографій розроблено за допомогою мови програмування C#. Як фреймворк-середовище розробки для створення програмного забезпечення для збереження та перегляду фотографій обрано Microsoft Visual Studio. Для створення програми збереження та перегляду фотографій як СКБД обрано Microsoft SQL Server.

Б.2 Функціональне призначення

Програмний продукт для збереження та перегляду фотографій призначений для полегшення роботи користувачів різних професій при роботі з фотографічними зображеннями.

Програмне забезпечення для збереження та перегляду фотографій забезпечує виконання функцій:

- реєстрація користувачів;
- створення каталогів фотографій;
- додавання каталогів з фотографіями;
- додавання однієї чи декількох фотографій;
- видалення каталогів фотографій;
- видалення однієї чи декількох фотографій;
- синхронізація сховища з особистим персональним комп'ютером.

Б.3 Опис логічної структури

Програмне забезпечення для збереження та перегляду фотографій складається з трьох компонентів.

Клієнтська частина ПЗ для збереження та перегляду фотографій відповідає за відображення даних та здійснення маніпулювань в доступному виді для користувача. Представляє собою графічний інтерфейс програм через,

який користувач взаємодіє з нею. При здійсненні певних дій (додавання, видалення т.п), за допомогою запитів звертається до API програмного додатку. Після отримання відповіді, відображає результат користувачеві.

Web API відповідає за зв'язування запитів з клієнта до серверу ПЗ для збереження та перегляду фотографій. Роль API ПЗ для збереження та перегляду фотографій можливо порівняти з деяким мостом, який за певним запитом клієнта намагається отримати дані з серверу бази даних, отримавши дані про фотографії він повертає їх до клієнту.

Серверна частина ПЗ для збереження та перегляду фотографій представляє собою частину, на якій зберігається база даних про фотографії клієнтів та налаштування до неї. База даних про фотографії клієнтів зберігає інформацію користувачів, які при взаємодії з клієнтом програмного забезпечення можуть її отримати.

Б.4 Використані технічні засоби

Для нормального функціонування програми збереження та перегляду фотографій повинні виконуватися такі умови до апаратної частини:

- наявність персонального комп'ютера (ПК), що виконує роль сервера;
- процесор із тактовою частотою 2 GHz та більше;
- вільний простір на жорсткому диску не менше, ніж 1 ГБ;
- оперативна пам'ять 1Гб і вище;
- монітор, клавіатура та маніпулятор типу “миша”.

Умови до програмної частини, для нормального функціонування програми збереження та перегляду фотографій:

- наявність операційної системи(ОС) Windows;
- наявність на ПК .Net Framework;

– наявність на ПК Microsoft SQL Server.

Б.5 Виклик і завантаження

Для завантаження програми необхідно завантажити інсталятор (архів AppPackages) та за допомогою PowerShell запустити файл Add-AppDevPackage.ps.

Б.6 Вхідні та вихідні дані

Вхідними даними до програми збереження та перегляду фотографій є інформація яку вводить користувач за допомогою графічного інтерфейсу програми. Вони можуть бути: текстові (пароль, логін, назва папки) та тип фотографії (bmp, jpg, png).

Вихідними даними програми збереження та перегляду фотографій є інформація яку отримує користувач перед очима на графічному інтерфейсі. Вихідними даними можуть бути текстові поля, контейнери із фотографіями та каталогами. Також вихідними даними можуть вважатися діалогові окна повідомлень програми.

ДОДАТОК В
Текст програми

```

public class SignInViewModel : ViewModelBase
{
    private readonly IDataService dataService;
    private readonly INavigationService navigationService;

    /// <summary>
    /// Aggregation
    /// </summary>
    /// <param name="dataService"></param>
    /// <param name="navigationService"></param>
    public SignInViewModel(IDataService dataService, INavigationService
navigationService)
    {
        this.dataService = dataService;
        this.navigationService = navigationService;
    }

    private string email;
    public string Email
    {
        get { return email; }
        set
        {
            email = value;
            RaisePropertyChanged(nameof(Email));
        }
    }

    private string password;
    public string Password
    {
        get { return password; }
        set
        {
            password = value;
            RaisePropertyChanged(nameof(Password));
        }
    }
}

#region Command

private RelayCommand btnMoveToSignUp;

public RelayCommand BtnMoveToSignUpCommand
{
    get
    {

```

```

        return btnMoveToSignUp
            ?? (btnMoveToSignUp = new RelayCommand(async () =>
                {
                    MemoriesWebApi.Register();
                }

            ));
    }

    // Return navigationService
    internal INavigationService GetNavigationService() => navigationService;

    #endregion
}

public sealed partial class SignIn : Page
{
    public SignInViewModel Vm => (SignInViewModel)DataContext;
    public SignIn()
    {
        this.InitializeComponent();
    }

    private void GoBtn_Click(object sender, Windows.UI.Xaml.RoutedEventArgs e)
    {
        var response = MemoriesWebApi.UserLoginAsync(txtEmail.Text,
txtPassword.Password).Result;

        if (response.IsSuccessStatusCode)
        {
            INavigationService navigation = Vm.GetNavigationService();
            navigation.NavigateTo(ViewModelLocator.MyPhotosPageKey);
        }
        else
        {
            var dialog = new SystemDialog();
            dialog.InfoMessage = "Login or Password were not correct";
            dialog.ShowAsync();
        }
    }
}

public class UserPhotosViewModel : ViewModelBase
{
    private readonly IDataService dataService;
    private readonly INavigationService navigationService;

```

```

public List<AppImage> SelectedItems { get; set; }

/// <summary>
/// Aggregation
/// </summary>
/// <param name="dataService"></param>
/// <param name="navigationService"></param>
public UserPhotosViewModel(IDataService dataService, INavigationService
navigationService)
{
    this.dataService = dataService;
    this.navigationService = navigationService;

    Initialize();
}

private async Task Initialize()
{
    var item = new ObservableCollection<AppFolder>();
    try
    {
        // Check on pc
        item = await dataService.GetData();
        string message = "You load last sync data on your pc.\n" +
            "If the folders and pictures are different, you can:\n" +
            "1) Sync select folder (its faster)\n"+
            "2) Sync all data (its takes long time)";

        if (item == null)
        {
            item = await dataService.WebApiGetPholders();
            message = "You load last folders. You can:\n" +
                " 1) Sync select folder (its faster)\n" +
                " 2) Sync all data (its takes long time)";
        }

        SetBinding(item);
        ShowMessage(message);
    }
    catch (Exception ex)
    {
    }
}

private void SetBinding(ObservableCollection<AppFolder> item)
{
    foreach (var f in item)

```

```

    {
        f.DeleteCommand = new RelayCommand(DeletePhotos);
        f.UploadCommand = new RelayCommand(UploadPhotos);
        f.SyncCommand = new RelayCommand(SyncPhotos);
    }
    FolderList = item;
}

```

```

/// <summary>
/// Binding folder to view
/// </summary>
private ObservableCollection<AppFolder> folderList;
public ObservableCollection<AppFolder> FolderList
{
    get { return folderList; }
    set
    {
        folderList = value;
        RaisePropertyChanged(nameof(FolderList));
    }
}

```

```

/// <summary>
/// Check current selectFolder
/// </summary>
private AppFolder selectFolder;
public AppFolder SelectFolder
{
    get { return selectFolder; }
    set
    {
        selectFolder = value;
        RaisePropertyChanged(nameof(SelectFolder));
    }
}

```

#region Command

```

/// <summary>
/// Show settings
/// </summary>
private RelayCommand btnSettings;

public RelayCommand BtnSettingsCommand => btnSettings
    ?? (btnSettings = new RelayCommand(async () =>
    {
        SettingsDialog settingDialog = new SettingsDialog();
        ContentDialogResult result = await settingDialog.ShowAsync();
    }
    ));

```

```
    }));
```

```
/// <summary>
/// Create folder
/// </summary>
private RelayCommand btnCreateFolder;

public RelayCommand BtnCreateFolderCommand => btnCreateFolder
?? (btnCreateFolder = new RelayCommand(async () =>
{
    string nameFolder = "";

    FolderDialog folderDialog = new FolderDialog();
    ContentDialogResult result = await folderDialog.ShowAsync();

    if (result == ContentDialogResult.Primary)
        nameFolder = folderDialog.FolderName;
    else
        return;

    // Create folder on server
    MemoriesWebApi.CreateFolder(nameFolder);

    // Add folder to view
    AppFolder folder = new AppFolder(nameFolder, null);
    folder.DeleteCommand = new RelayCommand(DeletePhotos);
    folder.UploadCommand = new RelayCommand(UploadPhotos);
    folder.SyncCommand = new RelayCommand(SyncPhotos);

    folder.AppImages = new ObservableCollection<AppImage>();

    FolderList.Add(folder);

    // Create folder on user pc
    await FileHelper.UserPCCreateFolderAsync(nameFolder);
}));
```

```
/// <summary>
/// Delete folder
/// </summary>
private RelayCommand btnDeleteFolder;

public RelayCommand BtnDeleteFolderCommand => btnDeleteFolder
?? (btnDeleteFolder = new RelayCommand(() =>
{
    if (selectFolder == null)
    {
        ShowMessage("Please, select folder for delete");
        return;
    }
}
```

```

    }

    // Delete server
    MemoriesWebApi.DeleteFolder(selectFolder.Title);

    // Delete form pc
    FileHelper.UserDeleteFolder(selectFolder.Title);

    // Delete view
    FolderList.Remove(selectFolder);

  }));

```

```

/// <summary>
/// Delte photos
/// Method send as relaycommand parameter to AppFolder model
/// </summary>
private void DeletePhotos()
{
    var list = SelectedItems;

    if (list == null)
    {
        ShowMessage("Please select photo for delete.");
        return;
    }

    // Delete photo from view
    foreach (var img in list)
        selectFolder.AppImages.Remove(img);

    // Delete photo from system folder
    FileHelper.UserDeletePhotos(selectFolder.Title, list);

    // Delete photo from server
    foreach (var img in list)
        MemoriesWebApi.DeleteImage(selectFolder.Title, img.Title);

    SelectedItems = null;
}

/// <summary>
/// Upload photos
/// Method send as relaycommand parameter to AppFolder model
/// </summary>
private async void UploadPhotos()
{
    FileOpenPicker picker = GetFileOpnePicker();
    var loadFiles = await picker.PickMultipleFilesAsync();

```

```

        if (loadFiles != null)
        {
            foreach (var file in loadFiles)
            {
                var streamFile = await file.OpenStreamForReadAsync();

                var memStream = new MemoryStream();
                await streamFile.CopyToAsync(memStream);
                // Add image to view

                BitmapImage bitmapImage = new BitmapImage();
                memStream.Position = 0;
                bitmapImage.SetSource(memStream.AsRandomAccessStream());

                var loadImg = new AppImage(file.Name, bitmapImage);

                selectFolder.AppImages.Add(loadImg);

                // Create copy image on system folder
                streamFile.Position = 0;
                var bytes = memStream.ToArray();
                await FileHelper.UserPCCreateImage(bytes, selectFolder.Title, file.Name);

                // Upload image on server
                streamFile.Position = 0;
                await MemoriesWebApi.UploadImageAsync(streamFile, selectFolder.Title,
file.Name);
            }
        }

        /// <summary>
        /// Sync photos in folder with server
        /// Method send as relaycommand parameter to AppFolder model
        /// </summary>
        private async void SyncPhotos()
        {
            await dataService.WebApiSyncImages(selectFolder.Title);
            int index = FolderList.IndexOf(FolderList.Where(ph => ph.Title ==
selectFolder.Title).FirstOrDefault());
            FolderList[index].AppImages = await dataService.GetImages(selectFolder.Title);
        }

        /// <summary>
        /// Download folder
        /// Photos are included in this folder upload to server, also like pholder
        /// </summary>
        private RelayCommand btnDownloadFolderFromPC;

        public RelayCommand BtnDownloadFolderFromPC =>
        btnDownloadFolderFromPC

```

```

?? (btnDownloadFolderFromPC = new RelayCommand(async() =>
{
    var folderPicker = new FolderPicker();
    folderPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    folderPicker.FileTypeFilter.Add("*");

    StorageFolder folder = await folderPicker.PickSingleFolderAsync();

    if (folder != null)
    {
        // Add folder to view
        AppFolder createFolder = new AppFolder(folder.Name, null);
        createFolder.DeleteCommand = new RelayCommand(DeletePhotos);
        createFolder.UploadCommand = new
RelayCommand(UploadPhotos);
        createFolder.SyncCommand = new RelayCommand(SyncPhotos);

        FolderList.Add(createFolder);
        selectFolder = createFolder;

        var listImages = new ObservableCollection<AppImage>();

        // Create folder on server
        MemoriesWebApi.CreateFolder(folder.Name);

        // Create copy folder on system folder
        await FileHelper.UserPCCreateFolderAsync(folder.Name);

        // Download images

        var p = await folder.GetFilesAsync();
        foreach (var file in p)
        {
            // Check file -> Is image or not?
            if (IsImage(file.FileType))
            {
                var streamFile = await file.OpenStreamForReadAsync();

                var memStream = new MemoryStream();
                await streamFile.CopyToAsync(memStream);
                // Add image to view

                BitmapImage bitmapImage = new BitmapImage();
                memStream.Position = 0;

                bitmapImage.SetSource(memStream.AsRandomAccessStream());

                var loadImg = new AppImage(file.Name, bitmapImage);

                listImages.Add(loadImg);

                // Create copy image on system folder

```

```

        streamFile.Position = 0;
        var bytes = memStream.ToArray();
        await FileHelper.UserPCCreateImage(bytes, folder.Name,
file.Name);

        // Upload image on server
        streamFile.Position = 0;
        await MemoriesWebApi.UploadImageAsync(streamFile,
selectFolder.Title, file.Name);
    }
}

selectFolder.AppImages = listImages;
}

});

#endregion

#region Helper

/// <summary>
/// Initialization FilePicker
/// </summary>
/// <returns></returns>
private static FileOpenPicker GetFileOpnePicker()
{
    var picker = new FileOpenPicker();
    picker.ViewMode = PickerViewMode.Thumbnail;
    picker.SuggestedStartLocation = PickerLocationId.PicturesLibrary;
    picker.FileTypeFilter.Add(".jpg");
    picker.FileTypeFilter.Add(".jpeg");
    picker.FileTypeFilter.Add(".png");
    return picker;
}

/// <summary>
/// Check image by type
/// </summary>
/// <param name="type"></param>
/// <returns></returns>
private static bool IsImage(string type)
{
    string[] imgTypes = new string[] { ".png", ".jpg", ".BMP" };
    var result = imgTypes.FirstOrDefault(x => x.Contains(type));
    return (result==null) ? false : true;
}

/// <summary>
/// Convert Stream to bitmap
/// </summary>

```

```

/// <param name="streamFile"></param>
/// <returns></returns>
private BitmapImage StreamToBitmapImage(Stream streamFile)
{
    var memStream = new MemoryStream();
    streamFile.CopyToAsync(memStream);
    BitmapImage bitmapImage = new BitmapImage();
    memStream.Position = 0;
    bitmapImage.SetSource(memStream.AsRandomAccessStream());

    return bitmapImage;
}

/// <summary>
/// Convert Stream to byte[]
/// </summary>
/// <param name="streamFile"></param>
/// <returns></returns>
private byte[] StreamToByteArray(Stream streamFile)
{
    var memStream = new MemoryStream();
    streamFile.CopyToAsync(memStream);
    memStream.Position = 0;
    return memStream.ToArray();
}

private async void ShowMessage(string message)
{
    var dialog = new SystemDialog();
    dialog.InfoMessage = message;
    await dialog.ShowAsync();
}

// Return navigationService
internal INavigationService GetNavigationService() => navigationService;

#endregion
}

namespace Memories.ViewModels
{
    public class FullScreenViewModel : ViewModelBase
    {
        private readonly IDataService dataService;
        private readonly INavigationService navigationService;

        /// <summary>

```

```

    /// Aggregation
    /// </summary>
    /// <param name="dataService"></param>
    /// <param name="navigationService"></param>
    public FullScreenViewModel(IDataService dataService, INavigationService
navigationService)
    {
        this.dataService = dataService;
        this.navigationService = navigationService;
    }

    private AppFolder selectFolder;
    public AppFolder SelectFolder
    {
        get { return selectFolder; }
        set
        {
            selectFolder = value;
            RaisePropertyChanged(nameof(SelectFolder));
        }
    }

    private AppImage selectImage;
    public AppImage SelectImage
    {
        get { return selectImage; }
        set
        {
            selectImage = value;
            RaisePropertyChanged(nameof(SelectImage));
        }
    }

    private RelayCommand btnBack;

    public RelayCommand BtnBackCommand
    {
        get
        {
            return btnBack
                ?? (btnBack = new RelayCommand(() =>
navigationService.NavigateTo(ViewModelLocator.MyPhotosPageKey)));
        }
    }

    private RelayCommand btnDeletePhoto;

```

```

public RelayCommand BtnDeletePhotoCommand
{
    get
    {
        return btnDeletePhoto
            ?? (btnDeletePhoto = new RelayCommand(async () =>
            {
                if (selectImage == null)
                {
                    ShowMessage("Please select photo for delete.");
                    return;
                }

                var list = new List<AppImage>();
                list.Add(selectImage);

                // Delete photo from system folder
                await FileHelper.UserDeletePhotos(selectFolder.Title, list);

                // Delete photo from server
                MemoriesWebApi.DeleteImage(selectFolder.Title, selectImage.Title);

                selectFolder.AppImages.Remove(selectImage);
                if (selectFolder.AppImages.Count == 0)

navigationService.NavigateTo(ViewModelLocator.MyPhotosPageKey);
            }
            ));
    }
}

private async void ShowMessage(string message)
{
    var dialog = new MessageDialog(message);
    await dialog.ShowAsync();
}
}
}

```

```

namespace Memories.ViewModels
{
    /// <summary>
    /// This class contains properties that the SingUp View can data bind to.
    /// <para>
    /// See http://www.mvvmlight.net
    /// </para>
    /// </summary>
    public class ViewModelLocator
    {

```

```

public const string FullScreenPageKey = "FullScreen";
public const string MyPhotosPageKey = "MyPhotosPageKey";
public const string SignUpPageKey = "SignUpPageKey";
public const string SignInPageKey = "SignInPageKey";

static ViewModelLocator()
{
    ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);
    var nav = new NavigationService();
    nav.Configure(FullScreenPageKey, typeof(FullScreen));
    nav.Configure(MyPhotosPageKey, typeof(UserPhotos));
    nav.Configure(SignUpPageKey, typeof(SignUp));
    nav.Configure(SignInPageKey, typeof(SignIn));

    SimpleIoc.Default.Register<INavigationService>(() => nav);
    SimpleIoc.Default.Register<IDialogService, DialogService>();

    if (ViewModelBase.IsInDesignModeStatic)
    {
        SimpleIoc.Default.Register<IDataService, Design.FakeDataService>();
    }
    else
    {
        SimpleIoc.Default.Register<IDataService, DataService>();
    }

    SimpleIoc.Default.Register<SignInViewModel>();
    SimpleIoc.Default.Register<SignUpViewModel>();

    SimpleIoc.Default.Register<FullScreenViewModel>();
    SimpleIoc.Default.Register<UserPhotosViewModel>();
}

/// <summary>
/// Gets the Main property.
/// </summary>
[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Performance",
    "CA1822:MarkMembersAsStatic",
    Justification = "This non-static member is needed for data binding purposes.")]

public SignInViewModel SignInPage =>
ServiceLocator.Current.GetInstance<SignInViewModel>();

public SignUpViewModel SignUpPage =>
ServiceLocator.Current.GetInstance<SignUpViewModel>();

public FullScreenViewModel FullScreenPage =>
ServiceLocator.Current.GetInstance<FullScreenViewModel>();

public UserPhotosViewModel UserPhotosPage =>
ServiceLocator.Current.GetInstance<UserPhotosViewModel>();

```

```

    }
}

public class AppFolder : ViewModelBase
{
    private long id;
    private string title;

    private ObservableCollection<AppImage> appImages;

    public AppFolder() { }

    public AppFolder(string title, ObservableCollection<AppImage> appImages)
    {
        this.title = title;
        this.appImages = appImages;
    }

    public AppFolder(long id, string title, ObservableCollection<AppImage>
appImages)
    {
        Id = id;
        Title = title;
        AppImages = appImages;
    }

    public ObservableCollection<AppImage> AppImages { get => appImages;
set
    {
        appImages = value;
        RaisePropertyChanged();
    }
}

    public long Id { get => id; set => id = value; }
    public string Title { get => title; set => title = value; }

    public RelayCommand DeleteCommand { get; set; }

    public RelayCommand UploadCommand { get; set; }

    public RelayCommand SyncCommand { get; set; }
}

public class AppImage : ViewModelBase
{
    private int id;
    private string title;

```

```

private ImageSource appImageUrl;
private string path;

public AppImage() { }
public AppImage(string title, ImageSource imageSource)
{
    Title = title;
    AppImageUrl = imageSource;
}
public int Id { get => id; set => id = value; }
public string Title { get => title; set => title = value; }
public ImageSource AppImageUrl { get => appImageUrl; set => appImageUrl
= value; }
public string Path { get => path; set => path = value; }

}

```

Текст IDataService интерфейс данных (IDataService.cs)

```

public interface IDataService
{

    Task<ObservableCollection<AppFolder>> GetData();

    Task<ObservableCollection<AppImage>> GetImages(string folderName);

    // Web Api
    Task<ObservableCollection<AppFolder>> WebApiGetPholders();

    Task WebApiSyncImages(string folderName);

}

public class DataService : IDataService
{

    #region Load pc data
    public Task<ObservableCollection<AppFolder>> GetData()
    {
        return FileHelper.UserPCLoadData();
    }

    public Task<ObservableCollection<AppImage>> GetImages(string folderName)
    {
        return FileHelper.UserPCLoadImages(folderName);
    }

    #endregion
}

```

```

#region WebApi

/// <summary>
/// Get pholders without images
/// </summary>
/// <returns></returns>
public async Task<ObservableCollection<AppFolder>> WebApiGetPholders()
{
    var list = MemoriesWebApi.ServerGetFolders();
    return await Task.FromResult(list);
}

/// <summary>
/// Get images from folder (sync)
/// </summary>
/// <param name="titleFolder"></param>
/// <returns></returns>
public async Task WebApiSyncImages(string folderName)
{
    var folder = MemoriesWebApi.GetSyncSelectedFolder(folderName);
    if (folder.AppImages.Count > 0)
    {
        await FileHelper.UserPCCreateProgramFolderAsync();
        await FileHelper.UserPCCreateUserFolder();
        await FileHelper.UserPCCreateFolderAsync(folderName);

        foreach (var b in folder.AppImages)
        {
            byte[] bytes = MemoriesWebApi.LoadImage(b.Path);
            await FileHelper.UserPCCreateImage(bytes, folderName, b.Title);
        }
    }
}

#endregion

}

public static class MemoriesWebApi
{
    public static string APP_PATH = "http://localhost:49979";
    public static HttpClient client = new HttpClient { BaseAddress = new
Uri(APP_PATH) };
    public static string userfolder;

    #region Authorization

    public static Task<HttpResponseMessage> UserLoginAsync(string username,
string password)

```

```

{
    client = new HttpClient { BaseAddress = new Uri(APP_PATH) };
    var json = JsonConvert.SerializeObject(username);

    string url = $"/api/account/login?email={username}&password={password}";

    var response = client.PostAsync(url, null).Result;
    if (response.IsSuccessStatusCode)
        userfolder = username.Split("@")[0];

    return Task.FromResult(response);
}

public async static void Register()
{
    string path = "/api/account/register";

    string uriToLaunch = APP_PATH + path;
    var uri = new Uri(@uriToLaunch);

    var success = await Windows.System.Launcher.LaunchUriAsync(uri);
}

#endregion

#region Get-Query

/// <summary>
/// Get folder from Server
/// </summary>
/// <returns></returns>
public static ObservableCollection<AppFolder> ServerGetFolders()
{
    HttpResponseMessage response = client.GetAsync("/api/appfolder").Result;
    var list = new ObservableCollection<AppFolder>();
    if (response.IsSuccessStatusCode)
    {
        var jsonData = response.Content.ReadAsStringAsync().Result;
        try
        {
            {
                list =
JsonConvert.DeserializeObject<ObservableCollection<AppFolder>>(jsonData.ToString());
            }
            catch (Exception ex) { }
        }
    }
}

```

```

        return list;
    }

    public static AppFolder GetSyncSelectedFolder(string title)
    {
        AppFolder folder = GetFolder(title);
        return folder;
    }

    public static AppFolder GetFolder(string title)
    {
        string path = "/api/appfolder/" + title;
        HttpResponseMessage response = client.GetAsync(path).Result;
        var folder = new AppFolder();
        if (response.IsSuccessStatusCode)
        {
            var jsonData = response.Content.ReadAsStringAsync().Result;
            try
            {
                folder = JsonConvert.DeserializeObject<AppFolder>(jsonData.ToString());
            }
            catch (Exception ex) { }
        }

        return folder;
    }

    public static byte[] LoadImage(string path)
    {
        path = APP_PATH + "/" + path.Replace("\\", "/");
        WebClient wc = new WebClient();
        byte[] bytes = wc.DownloadData(path);

        return bytes;
    }

#endregion

#region Post-Query

/// <summary>
///     Создание и удаление папки, необходимо сделать возвращающее значение
/// </summary>
/// <param name="title"></param>
public static void CreateFolder(string title)
{
    AppFolder item = new AppFolder();
    item.Title = title;

```

```

        var json = JsonConvert.SerializeObject(item);

        var stringContent = new StringContent(json, UnicodeEncoding.UTF8,
"application/json");

        HttpResponseMessage response = client.PostAsync("/api/appfolder",
stringContent).Result;
        if (response.IsSuccessStatusCode)
            Debug.WriteLine("{0} ({1})", (int)response.StatusCode,
response.ReasonPhrase);
        else
            Debug.WriteLine("{0} ({1})", (int)response.StatusCode,
response.ReasonPhrase);
    }

    public static void DeleteFolder(string title)
    {

        string path = "/api/appfolder/" + title;
        HttpResponseMessage response = client.DeleteAsync(path).Result;
        if (response.IsSuccessStatusCode)
            Debug.WriteLine("{0} ({1})", (int)response.StatusCode,
response.ReasonPhrase);
        else
            Debug.WriteLine("{0} ({1})", (int)response.StatusCode,
response.ReasonPhrase);
    }

    public async static Task<HttpResponseMessage> UploadImageAsync(Stream
image, string folderName, string imageName)
    {
        string url = "/api/appimage/" + folderName;

        HttpContent fileStreamContent = new StreamContent(image);
        fileStreamContent.Headers.ContentDisposition = new
ContentDispositionHeaderValue("form-data") { Name = "imageFile", FileName = imageName };
        fileStreamContent.Headers.ContentType = new
MediaTypeHeaderValue("application/octet-stream");

        using (var formData = new MultipartFormDataContent())
        {
            formData.Add(fileStreamContent);
            var response = await client.PostAsync(url, formData);
            if (response.IsSuccessStatusCode)
            {
                Debug.WriteLine("Success");
            }
            else
            {
                Debug.WriteLine("Doesn't success");
            }
        }
    }

```

```

        }
        return response;
    }
}

// Должно возвращать значение
public static void DeleteImage(string folderName, string imageName)
{
    string baseUrl = String.Concat(APP_PATH, "/api/appimage/");
    var builder = new UriBuilder(baseUrl);

    var query = HttpUtility.ParseQueryString(builder.Query);
    query["folderTitle"] = folderName;
    query["imageTitle"] = imageName;
    builder.Query = query.ToString();
    string url = builder.ToString();

    HttpResponseMessage response = client.DeleteAsync(url).Result;
    if (response.IsSuccessStatusCode)
        Debug.WriteLine("{0} ({1})", (int)response.StatusCode,
response.ReasonPhrase);
    else
        Debug.WriteLine("{0} ({1})", (int)response.StatusCode,
response.ReasonPhrase);
    }

    #endregion
}

```

Частина Web API

```

public class ApplicationUser : IdentityUser
{
    // This is a mapped property
    protected virtual ICollection<AppFolder> _appFolders { get; set; }

    // This is an un-mapped property that just wraps _photos
    public IEnumerable<AppFolder> AppFolders
    {
        get { return _appFolders; }
    }

    public ApplicationUser() { }

    public ApplicationUser(string userName) : base(userName)
    {
        _appFolders = new List<AppFolder>();
    }
}

```

```

public class AppFolder
{
    public long Id { get; set; }
    public string Title { get; set; }

    public AppFolder()
    {
        _appImages = new List<AppImage>();
    }

    // This is a mapped property
    protected virtual ICollection<AppImage> _appImages { get; set; }

    // This is an un-mapped property that just wraps _photos
    public IEnumerable<AppImage> AppImages
    {
        get { return _appImages; }
    }
    public string ApplicationUserId { get; set; }
    public ApplicationUser User { get; set; }
}

```

```

public class AppImage
{
    public long Id { get; set; }
    public string Title { get; set; }
    public string Path { get; set; }

    public long FolderId { get; set; }
    public AppFolder Folder { get; set; }
}

```

```

public class RegisterViewModel
{
    [Required]
    [Display(Name = "Email")]
    public string Email { get; set; }

    [Required]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [Required]
    [Compare("Password", ErrorMessage = "Password isn't equal")]
    [DataType(DataType.Password)]

```

```

        [Display(Name = "Repeat password")]
        public string PasswordConfirm { get; set; }
    }

```

```
@model MemoriesWebAPI.Models.AccountViewModels.RegisterViewModel
```

```

@{
    ViewData["Title"] = "Register";
}

```

```

<h2>Sing Up</h2>
<h6>It's about to get fun</h6>
<hr />
<div class="container">
<div class="row">
<div class="col-md-5">
<formasp-action="Register">
<divasp-validation-summary="ModelOnly" class="text-danger"></div>
<div class="form-group">
<labelasp-for="Email" class="control-label"></label>
<inputasp-for="Email" class="form-control" />
<spanasp-validation-for="Email" class="text-danger"></span>
</div>
<div class="form-group">
<labelasp-for="Password" class="control-label"></label>
<inputasp-for="Password" class="form-control" />
<spanasp-validation-for="Password" class="text-danger"></span>
</div>
<div class="form-group">
<labelasp-for="PasswordConfirm" class="control-label"></label>
<inputasp-for="PasswordConfirm" class="form-control" />
<spanasp-validation-for="PasswordConfirm" class="text-danger"></span>
</div>
<div class="form-group">
<input type="submit" value="Create" class="btn btn-default" />
</div>
</div>
</form>
</div>
</div>
</div>

```

```

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

```

```

[Route("api/[controller]/[action]")]
public class AccountController : Controller
{

```

```

private readonly UserManager<ApplicationUser> _userManager;
private readonly SignInManager<ApplicationUser> _signInManager;
private readonly IHostingEnvironment _hosting;
private readonly ApplicationDbContext _context;

public AccountController(UserManager<ApplicationUser> userManager,
SignInManager<ApplicationUser> signInManager,
    IHostingEnvironment hosting, ApplicationDbContext context)
{
    _userManager = userManager;
    _signInManager = signInManager;
    _hosting = hosting;
    _context = context;
}

[HttpGet]
public IActionResult Register()
{
    return View();
}

private Task<ApplicationUser> GetCurrentUserAsync() =>
_userManager.GetUserAsync(HttpContext.User);

[HttpPost]
public async Task<object> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        ApplicationUser user = new ApplicationUser { Email = model.Email,
UserName = model.Email };
        // добавляемпользователя
        var result = await _userManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            // установкакуки
            await _signInManager.SignInAsync(user, false);

            FileHelper.ServerCreateUserFolder(user.Id, _hosting.WebRootPath);

            return Content($"Registered was succeeded! Please, back to desktop client
and authorize!");
        }
        else
        {
            foreach (var error in result.Errors)
            {
                ModelState.AddModelError(string.Empty, error.Description);
            }
        }
    }
    return View(model);
}

```

```
}
```

```
[HttpPost]
public async Task<IActionResult> Login(string email, string password)
{
    if (ModelState.IsValid)
    {
        var result =
            await _signInManager.PasswordSignInAsync(email, password, true, false);

        if (result.Succeeded)
        {
            return Ok(result);
        }
        else
        {
            return NotFound();
        }
    }
    else
    {
        return NotFound();
    }
}
```

```
[HttpDelete]
public async Task<IActionResult> DeleteUser([FromBody] ApplicationUser user)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);

    var userId = await _userManager.FindByEmailAsync(user.Email);
    try
    {
        {
            await _userManager.DeleteAsync(userId);
        }
        catch(Exception ex)
        {
        }

        string path = Path.Combine(_hosting.WebRootPath, "AppStore", userId.Id);

        FileHelper.ServerDeleteUser(path);
        _context.SaveChanges();

        return Ok(user);
    }
}
```

```
}
```

```
[Produces("application/json")]
[Route("api/[controller]")]
public class AppFolderController : Controller
{
    private readonly ApplicationDbContext _context;
    private readonly IHostingEnvironment _hosting;

    private readonly UserManager<ApplicationUser> _userManager;
    public AppFolderController(ApplicationDbContext context, IHostingEnvironment
hosting, UserManager<ApplicationUser> userManager)
    {
        _context = context;
        _hosting = hosting;
        _userManager = userManager;

        System.Security.Claims.ClaimsPrincipal currentUser = this.User;
    }

    /// <summary>
    /// GET: api/AppFolder
    /// Without images
    /// </summary>
    /// <returns></returns>
    [HttpGet]

    public IEnumerable<AppFolder> GetFolders()
    {

        var id = _userManager.GetUserId(User);

        var list = _context.AppFolders.Where(f => f.ApplicationUserId == id);

        return list;
    }

    /// <summary>
    /// GET: api/AppFolder/4
    /// Get images from folder by id
    /// </summary>
    /// <param name="id"></param>
    /// <returns></returns>
    [HttpGet("{Title}")]
    public AppFolder GetImagesInFolder(string title)
    {
        // Check sql query
```

```

        var folder = _context.AppFolders.Include(p => p.AppImages).FirstOrDefault(f =>
f.Title == title);
        return folder;

    }

    /// <summary>
    /// Simple create folder
    /// POST: api/AppFolder
    /// </summary>
    /// <param name="appFolder"></param>
    /// <returns></returns>
    [HttpPost]
    public async Task<IActionResult> CreateFolder([FromBody] AppFolder appFolder)
    {
        if (!ModelState.IsValid)
        {
            return BadRequest(ModelState);
        }

        appFolder.ApplicationUserId = _userManager.GetUserId(User);

        _context.AppFolders.Add(appFolder);
        await _context.SaveChangesAsync();

        FileHelper.ServerCreateUserFolder(appFolder.ApplicationUserId,
_hosting.WebRootPath);
        FileHelper.ServerCreateFolder(appFolder.ApplicationUserId, appFolder.Title);

        return CreatedAtAction("GetAppFolder", new { id = appFolder.Id }, appFolder);
    }

    /// <summary>
    /// Update title folder
    /// PUT: api/AppFolder/5
    /// </summary>
    /// <param name="id"></param>
    /// <param name="appFolder"></param>
    /// <returns></returns>
    [HttpPut("{id}")]
    public async Task<IActionResult> PutAppFolder(long id, [FromBody] AppFolder
appFolder)
    {
        if (!ModelState.IsValid)
            return BadRequest(ModelState);

        var folder = _context.AppFolders.Find(id);
        if (folder == null)
            return NotFound();
    }

```

```

        folder.Title = appFolder.Title;

        _context.AppFolders.Update(folder);

        //Need change title folder in server store

        try
        {
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!AppFolderExists(id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }

        return NoContent();
    }

    /// <summary>
    /// DELETE: api/AppFoldersTest/5
    /// Delete folder
    /// </summary>
    /// <param name="id"></param>
    /// <returns></returns>
    [HttpDelete("{Title}")]
    public async Task<IActionResult> DeleteAppFolder([FromRoute] string title)
    {
        var appFolder = _context.AppFolders.FirstOrDefault(ph => ph.Title == title);

        if (!ModelState.IsValid)
            return BadRequest(ModelState);

        if (appFolder == null)
            return NotFound();

        string path = Path.Combine(_hosting.WebRootPath, "AppStore",
appFolder.ApplicationUserId, appFolder.Title);

        FileHelper.ServerDeletePholder(path);

        _context.AppFolders.Remove(appFolder);
        await _context.SaveChangesAsync();

        return Ok(appFolder);
    }

```

```

    }

    private bool AppFolderExists(long id) => _context.AppFolders.Any(e => e.Id ==
id);

[Produces("application/json")]
[Route("api/[controller]")]
public class AppImageController : Controller
{
    private readonly ApplicationDbContext _context;
    private readonly IHostingEnvironment _hosting;
    private readonly UserManager<ApplicationUser> _userManager;

    public AppImageController(IHostingEnvironment hosting, ApplicationDbContext
context, UserManager<ApplicationUser> userManager)
    {
        _context = context;
        _hosting = hosting;
        _userManager = userManager;

        System.Security.Claims.ClaimsPrincipal currentUser = this.User;
    }

    /// <summary>
    /// Upload image on server by folder title
    /// </summary>
    /// <param name="imageFile"></param>
    /// <param name="title"></param>
    /// <returns></returns>
    [HttpPost("{Title}")]
    public async Task<IActionResult> UploadFile(IFormFile imageFile, string title)
    {
        var folder = _context.AppFolders.FirstOrDefault(i => i.Title == title);

        if (imageFile == null || imageFile.Length == 0)
            return Content("file not selected");

        //2) Create user folder
        var idUser = _userManager.GetUserId(User);

        FileHelper.ServerCreateUserFolder(idUser, _hosting.WebRootPath);

        //3) Create folder
        FileHelper.ServerCreateFolder(idUser, folder.Title);

        var path = Path.Combine("AppStore", idUser, folder.Title, imageFile.FileName);

        AppImage appImage = new AppImage();
        appImage.Title = imageFile.FileName;
        appImage.Path = path;
    }
}

```

```

appImage.FolderId = folder.Id;

//4) Create image

path = Path.Combine(_hosting.WebRootPath, path);

await FileHelper.ServerCreateImageAsync(path, imageFile);

// Write database
_context.AppImages.Add(appImage);
await _context.SaveChangesAsync();

return CreatedAtAction("GetAppImage", new { id = appImage.Id }, appImage);
}

[HttpDelete]
public async Task<IActionResult> Delete(string folderTitle, string imageTitle)
{
    if (folderTitle == null || imageTitle == null)
        return NotFound();

    if (!ModelState.IsValid)
        return BadRequest(ModelState);

    var image = _context.AppFolders.Include(p => p.AppImages).FirstOrDefault(f =>
f.Title == folderTitle)
        .AppImages.SingleOrDefault(i => i.Title == imageTitle);

    if (image == null)
        return NotFound();

    string path = Path.Combine(_hosting.WebRootPath, image.Path);

    _context.AppImages.Remove(image);
    FileHelper.ServerDeleteImage(path);

    await _context.SaveChangesAsync();

    return Ok(image);
}

}

public static class FileHelper
{
    private static string pathDownload =

Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.UserProfile),
"Downloads");

```

```

// We get path from controller IHostingEnvironment
// Path initialize in method: ServerCreateSystemFolder
private static string pathUpload = "";

private const string programName = "Memories Desktop";
private const string systemFolder = "AppStore";

// For create path in controoler (AppImage -> method: Upload File)
public static string SystemFolder { get; private set; }

/// <summary>
/// File system on Server
/// </summary>
#region server

// Create system folder on server
public static void ServerCreateSystemFolder(string hostingPath)
{

    SystemFolder = systemFolder;

    string path = Path.Combine(pathUpload, SystemFolder);
    if (!Directory.Exists(path))
    {
        Directory.CreateDirectory(path);
    }
}

// Create user folder on Server Store
public static void ServerCreateUserFolder(string idUser, string hostingPath)
{
    pathUpload = hostingPath;
    string path = Path.Combine(pathUpload, systemFolder, idUser);
    if (!Directory.Exists(path))
    {
        Directory.CreateDirectory(path);
    }
}

// Create folder on server store
public static void ServerCreateFolder(string idUser, string nameFolder)
{
    string path = Path.Combine(pathUpload, systemFolder, idUser, nameFolder);
    if (!Directory.Exists(path))
    {
        Directory.CreateDirectory(path);
    }
}

// Create image on server store
public static async Task ServerCreateImageAsync(string path, IFormFile file)

```

```

{
    using (var stream = new FileStream(path, FileMode.Create))
    {
        await file.CopyToAsync(stream);
    }
}

// Delete user
public static void ServerDeleteUser(string path)
{
    if (Directory.Exists(path))
        Directory.Delete(path, true);
}

// Delete folder
public static void ServerDeletePholder(string path)
{
    if (Directory.Exists(path))
        Directory.Delete(path, true);
}

// Delete image
public static void ServerDeleteImage(string path)
{
    if (File.Exists(path))
        File.Delete(path);
}

#endregion

/// <summary>
/// File system on User PC
/// </summary>
#region user pc
// Create program folder on user pc when he sign up in client
public static void UserPCCreateProgramFolder()
{
    string path = Path.Combine(pathDownload, programName);
    if (!Directory.Exists(path))
    {
        Directory.CreateDirectory(path);
    }
}

// Create folder on user pc
public static void UserPCCreateFolder(string folderName)
{
    string path = Path.Combine(pathDownload, programName, folderName);
    if (!Directory.Exists(path))

```

```
        {
            Directory.CreateDirectory(path);
        }
    }

    // Create image on user pc
    public static void UserPCCreateImage(byte[] imageArray, string pathImage)
    {
        string path = Path.Combine(pathDownload, programName, pathImage);
        if (!File.Exists(path))
        {
            File.WriteAllBytes(path, imageArray);
        }
    }

    #endregion
}
```

ДОДАТОК Д
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗБЕРЕЖЕННЯ
ТА ПЕРЕГЛЯДУ ФОТОГРАФІЙ**

Виконала
ст. гр. КНТ-119сп

Д. О. Лісанова

Керівник
к.т.н., доцент

Є. О. Гофман

Рисунок Д.1 – Слайд 1

Об’єктом дослідження є процес розробки програмного забезпечення для збереження та перегляду фотографій.

Предмет дослідження – програмні засоби оброблення, збереження та перегляду фотографій.

Мета роботи – програмна реалізація процесу збереження та перегляду фотографій.

Рисунок Д.2 – Слайд 2

Завдання роботи:

- виконати аналіз предметної області та процесу збереження та перегляду фотографій;
- здійснити проєктування програмного забезпечення для збереження та перегляду фотографій;
- створити програмне забезпечення для збереження та перегляду фотографій;
- виконати тестування розробленого програмного забезпечення збереження та перегляду фотографій.

3

Рисунок Д.3 – Слайд 3

Дослідження програмних засобів для збереження та перегляду фотографій

Критерії порівняння	Програмні засоби для збереження та перегляду фотографій				
	Microsoft Office Picture	Instagram	Canva	Photo Booth Connected	inPixio Photo Studio
Робота в онлайн-режимі	–	+	+	+	+
Наявність шаблонів	+	–	+	–	+
Робота з графічними примітивами	+	–	+	+	+
Засоби редагування текстів	+	+	+	–	+
Засоби роботи з аудіо та відео	–	+	+	+	–
Можливість роботи як плагін	–	+	–	+	–
Робота на різних ОС	–	+	–	+	+

4

Рисунок Д.4 – Слайд 4

Вибір мови програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	C++	Python
Зручність для створення ПЗ для збереження та перегляду фотографій	+	+–	+–
Засоби обробки зображень	+	+–	+
Інструменти для роботи з файлами	+	+	+–
Обробка надзвичайних ситуацій	+	+–	+
Засоби роботи з БД	+	+–	+

5

Рисунок Д.5 – Слайд 5

Вибір середовища розробки

Критерій порівняння середовищ розробки	Середовище розробки		
	Microsoft Visual Studio	SharpDevelop	MonoDevelop
Зручність для створення ПЗ для збереження та перегляду фотографій	+	–	+–
Швидкість модифікації розроблюваного ПЗ	+	+–	+–
Можливість роботи на різних платформах	+–	–	+
Засоби роботи з репозиторіями	+	+–	–

6

Рисунок Д.6 – Слайд 6

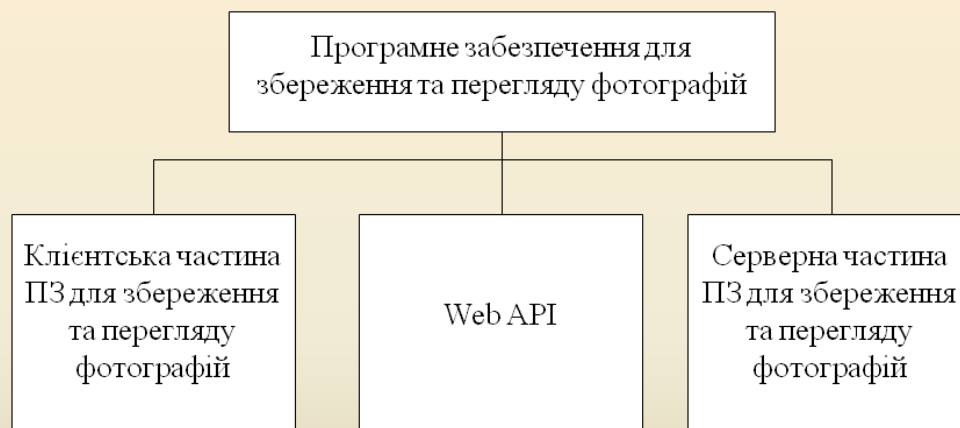
Вибір системи керування базою даних

Критерій порівняння СКБД	СКБД		
	Microsoft SQL Server	MySQL	MongoDB
Зручність для розробки ПЗ збереження та перегляду фотографій	+	+-	+-
Наявність літератури по використанню СКБД	+	+	+-
Засоби SSMS для налаштування БД	+	-	-
Можливість роботи за клієнт-серверною технологією	+	+	+-

7

Рисунок Д.7 – Слайд 7

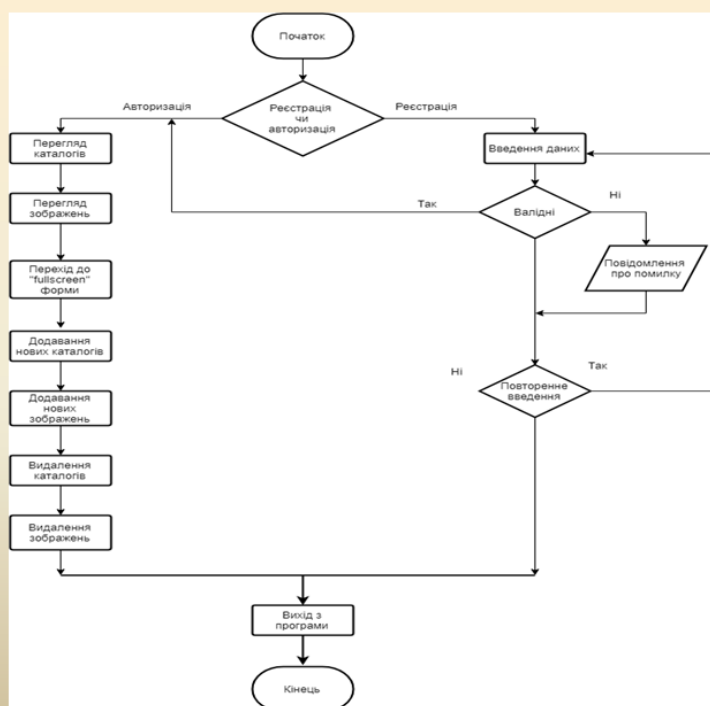
Структура програмного забезпечення для збереження та перегляду фотографій



8

Рисунок Д.8 – Слайд 8

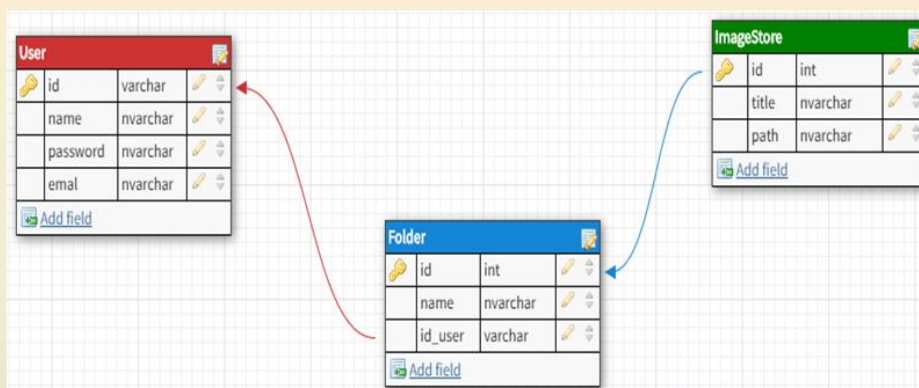
Узагальнена схема роботи клієнтської частини програми



9

Рисунок Д.9 – Слайд 9

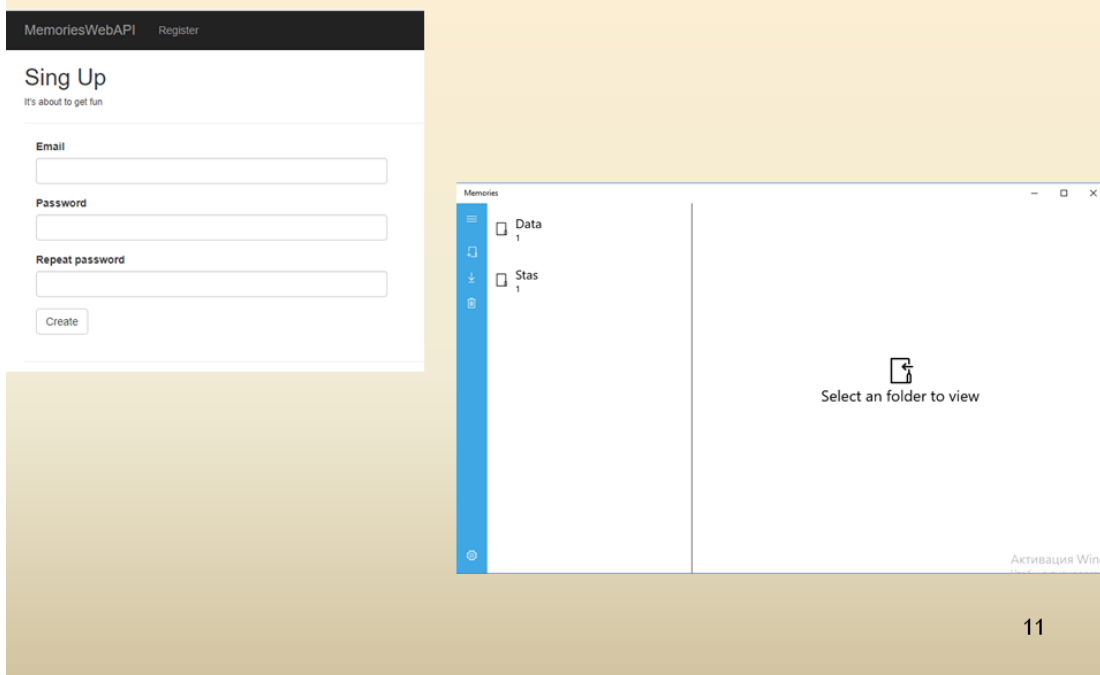
Схема бази даних



10

Рисунок Д.10 – Слайд 10

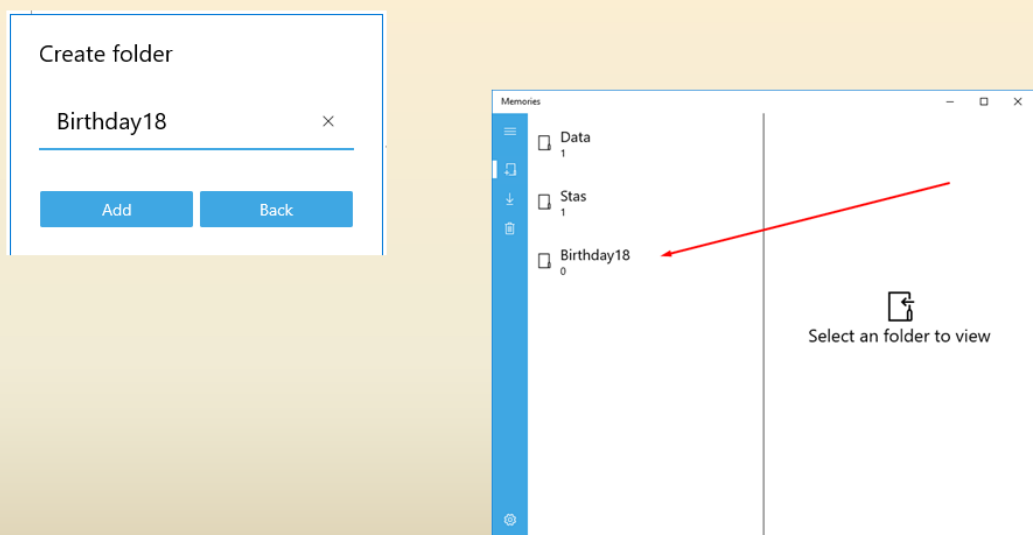
Робота з програмою. Реєстрація. Вхід зареєстрованого користувача



11

Рисунок Д.11 – Слайд 11

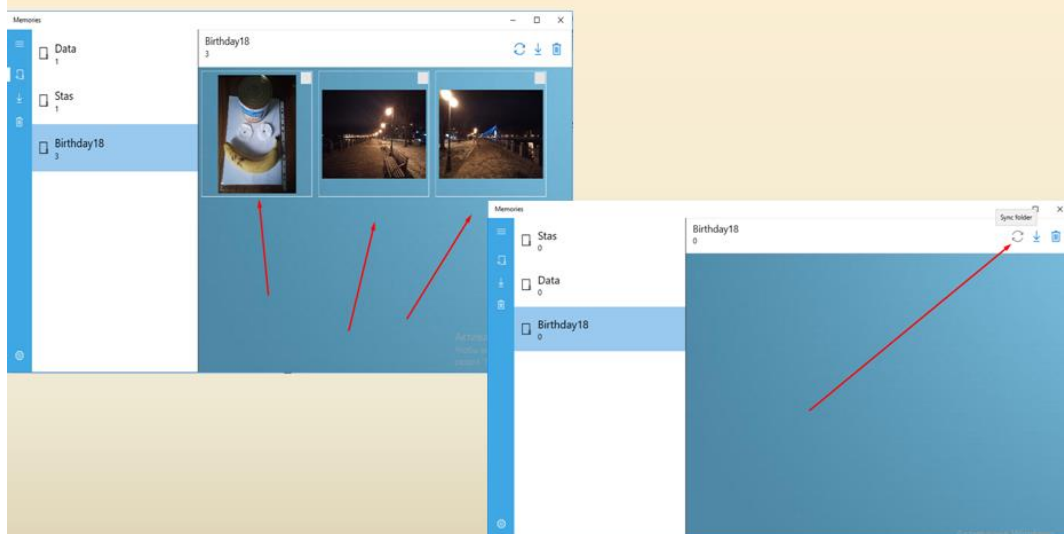
Робота з програмою. Створення нової папки



12

Рисунок Д.12 – Слайд 12

Робота з програмою. Додавання фотографій. Можливість синхронізації



13

Рисунок Д.13 – Слайд 13

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для збереження та перегляду фотографій.

Проаналізовано основні поняття про програмні сервіси збереження та перегляду фотографій.

Для створення програмного забезпечення збереження та перегляду фотографій було обрано мову програмування C#, яка є об'єктно-орієнтованою мовою програмування із безпечною системою запису для платформи .NET.

Розроблено схему функціонування клієнтської частини програмного забезпечення для збереження та перегляду фотографій. Описано основні функції програмного забезпечення збереження та перегляду фотографій.

Розроблено базу даних для повноцінної роботи програми збереження та перегляду фотографій, яка складається з таблиць User, Folder та ImageStore.

Виконано тестування розробленого програмного забезпечення обліку збереження та перегляду фотографій. Результати тестування програмного забезпечення збереження та перегляду фотографій підтвердили його працездатність та можливість використання за призначенням.

14

Рисунок Д.14 – Слайд 14