Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Створення веб-сервісу для завантаження та друку зображень із соціальних мереж

Виконав: студент IV курсу, групи СНзс-42
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Кріль М.І.
(прізвище та ініціали)

Керівник (підпис) Матійчук Л.П.
(прізвище та ініціали)

Нормоконтроль (підпис) Марценко С.В.
(прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О.
(прізвище та ініціали)

Рецензент (підпис) Жаровський Р.О.
(прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2023р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)
Студенту Кріль Михайло Іванович
(прізвище, ім'я, по батькові)
1. Тема роботи Створення веб-сервісу для завантаження та друку зображень із соціальних мереж

Керівник роботи Матійчук Любомир Павлович, к.е.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від « 07 » лютого 2023 року № 4/7-135.
2. Термін подання студентом завершеної роботи 12 червня 2023р.
3. Вихідні дані до роботи Наукові публікації, електронні ресурси, підручники , посібники тематики дослідження
4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Аналіз предметної області, проектування веб-сервісу завантаження та друку зображень із соціальних. 1.1 Опис предметної області, огляд і аналіз існуючих аналогів, що реалізують функції предметної області. 1.2 Специфікація вимог до системи. 1.3 Розроблення архітектури програмної системи та проектування структури бази даних. 1.4. API соціальних мереж використання протоколу OAuth. 1.5 Висновки до першого розділу. Розділ 2. Програмна реалізація веб-сервісу тестування та дослідна експлуатація. 2.1. Використовувані технології та стандарти. 2.2. Програмна реалізація отримання зображень, оплати, реалізація друку. 2.3. Тестування системи. 2.4. Особливості встановлення та налаштування продукту, інструкція користувача. Розділ 3. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема. 2. Актуальність роботи. 3. Аналіз відомих рішень. 4. Аналіз відомих рішень. 5. Загальна архітектура системи. 6. Загальна архітектура системи у вигляді діаграми пакетів. 7. Схема бази даних. 8. Діаграма послідовності для авторизації через протокол OAuth. 9. Діаграма послідовності для авторизації через протокол OAuth. 10. Ієрархія класів екрану веб-камери. 11. Діаграма класів для розрахунку ціни вибраних фотографій. 12. Тестування системи. 13. Сторінка вибору джерела фотографії. 14. Сторінка вибору фотографій. 15. Сторінка роботи з веб-камерою. Сторінка завантаження фотографій з телефону. 16. Сторінка редагування фотографій. Сторінка оплати та друку фото. 17. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О. Я., доцент кафедри МТ		

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	08.05.2023	*Виконано*
2.	Підбір джерел про програмно-алгоритмічний комплекс для обліку товарів і товарних операцій перукарського салону	09.05.2023-11.05.2023	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	12.05.2023-15.05.2023	*Виконано*
4.	Виконання дослідження щодо створення веб-сервісу для завантаження та друку зображень із соціальних мереж	16.05.2023-22.05.2023	*Виконано*
5.	Оформлення розділу «Аналіз предметної області, проектування веб-сервісу завантаження та друку зображень із соціальних»	23.05.2023-25.05.2023	*Виконано*
6.	Оформлення розділу «Програмна реалізація веб-сервісу тестування та дослідна експлуатація.»	26.05.2023-28.05.2023	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	29.05.2023-30.05.2023	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	31.05.2023-01.06.2023	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	02.06.2023-06.06.2023	*Виконано*
10.	Нормоконтроль	07.06.2023-08.06.2023	*Виконано*
11.	Перевірка на плагіат	09.06.2023	*Виконано*
12.	Попередній захист кваліфікаційної роботи	12.06.2023	*Виконано*
13.	Захист кваліфікаційної роботи	14.06.2023	

Студент _____ Кріль М.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Матійчук Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення веб-сервісу для завантаження та друку зображень із соціальних мереж// Кваліфікаційна робота освітнього рівня «Бакалавр» // Кріль Михайло Іванович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНзс-42 // Тернопіль, 2023 // С. 70, рис. –58, табл. – 3, бібліогр. – 31.

Ключові слова: мобільний застусунок, зображень із соціальних мереж,веб-сервіс, MySQL, база даних.

У першому розділі здійснено опис предметної області, напрями діяльності. Проведено аналіз відомих програмних систем. Здійснено аналіз вимог до програмної системи, розроблено архітектуру програмного додатку, що дозволить краще зрозуміти функції основних його частин. Створено та описано структурну схему, основними компонентами якої є: рівень клієнта, рівень бізнес-логіки та рівень даних. Описано функціональну структуру системи та її основних елементів – модулів обробки даних. Визначено основні елементи бази даних та встановлено зв'язки між ними. Спроектовано структуру бази даних

У другому розділі обґрунтовано технологію, мову програмування та розроблено програмну систему. Обґрунтовано засоби розробки бази даних та створено програмну реалізацію бази даних програмної системи за допомогою механізму збережених процедур. Здійснено опис процедур тестування та їхніх результатів, описані тест-вимоги до програмного забезпечення, а також виявлені дефекти. Розкрито питання встановлення та налаштування програмного забезпечення на сервері, а також вказані вимоги, дотримання яких необхідно для користування системою, описана інструкція користувача для роботи із системою.

ANNOTATION

Web Service Development for Social Networks Images Downloading and Printing // Qualification work of the educational level "Bachelor" // Mykhailo Ivanovich Kryl // Ivan Pulyuy Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer of computer sciences, group SNzs-42 // Ternopil, 2023 // C. 70, fig. -58, tab. – 3, bibliography -31.

Key words: mobile application, images from social networks, web service, MySQL, database.

In the first section, a description of the subject area, directions of activity was carried out. The analysis of known software systems was carried out. The analysis of the requirements for the software system was carried out, the architecture of the software application was developed, which will allow a better understanding of the functions of its main parts. A structural diagram has been created and described, the main components of which are: client level, business logic level, and data level. The functional structure of the system and its main elements - data processing modules - are described. The main elements of the database are defined and the connections between them are established. The structure of the database is designed

In the second chapter, the technology, the programming language and the software system are developed. The means of developing the database have been substantiated and the software implementation of the database of the software system has been created using the mechanism of stored procedures. Test procedures and their results are described, software test requirements are described, and defects are identified. The issue of installing and configuring the software on the server is disclosed, as well as the requirements that must be met in order to use the system, and the user manual for working with the system is described.

ЗМІСТ

ВСТУП

Актуальність теми. Соціальні мережі міцно увійшли в наше повсякденне життя, неможливо уявити звичний уклад речей, що не містить взаємодія з ними. Соціальна мережа - один з головних інструментів для комунікації, поширення користувацького контенту, для деяких вона стала навіть джерелом доходу. В такому середовищі одним зголовних видів інформації є зображення: фотографії, рисунки,скріншоти. Але, незважаючи на все прискорюється розвиток інформаційнихтехнологій, багато людей вважають за краще зберігати зображення не вцифровому, а у фізичному форматі, з ностальгічних або практичнихцілей. І, в такому випадку, перед людьми виростає проблема - яким чиномотримати потрібні фотографії, відредагувати їх і роздрукувати взадовільній якості, описана проблема стає ще гірше,якщо уявити, що користувач не має власного пристрою друкуабо навіть не має його в доступній близькості. Цю проблему покликані вирішитипринтери в вигляді фотобоксів, але на відміну від традиційнихфотобоксів, ці пристрої беруть зображення на друк з соціальнихмереж.

Апарати у вигляді фотобоксів, які отримують фотографії з соціальних мереж, останнім часом стають все популярнішими, і в зв'язку з цим замовник, який розробляє програмно-апаратні комплекси вендінгових пристроїв, проаналізував ринок і з'ясував неможливість придбання керуючого програмного забезпечення для такого апарату окремо від апаратної частини, тому, що по-перше не існує універсальних програмних рішень для різних апаратних конфігурацій, а по-друге виробники подібних фотобоксів не продають програмне забезпечення без апаратної частини. Тому була поставлена задача на розробку системи, описаної в даній роботі.

Мета і задачі дослідження.Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є розробка веб-застосунку. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

- Здійснити ґрунтовний аналіз предметної області,огляд і аналіз існуючих аналогів, що реалізують функції предметної області;
- Визначити специфікацію вимог до системи;
- Забезпечити розроблення архітектури програмної системи та проектування структури бази даних;
- Проектування веб-застосунку;
- Охарактеризувати технології та стандарти;
- Здійснити тестування розробленого веб-застосунку.

Практичне значення одержаних результатів.У даній роботі показана реалізація програми для скачування,оплати і друку зображень з соціальних мереж.Цей додаток розроблено для того, щоб використовуватися дляконтролю апарату-фотобокс, який надає послуги з друку фотографій.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПРОЕКТУВАННЯ ВЕБ-СЕРВІСУ ЗАВАНТАЖЕННЯ ТА ДРУКУ ЗОБРАЖЕНЬ ІЗ СОЦІАЛЬНИХ МЕРЕЖ

1.1Опис предметної області, огляд і аналіз існуючих аналогів, що реалізують функції предметної області

Фотобокс, фотоавтомат - пристрій для автоматичної зйомки і виготовлення фотографій на документи, різновид торгового автомата [12]. Установка монтується в відсіку спеціальної кабіни з регульованим стільцем. До появи цифрової фотографії фотоавтомати були найшвидшим способом отримання документних знімків, які були готові через кілька хвилин, тоді як у фотоательє на термінове замовлення витрачалося не менше години [1]. Завдяки тому, що зйомка та виготовлення фотографій відбуваються без участі фотографа, фотоавтомати отримали популярність в якості розваги створенням неформальних Селфі.

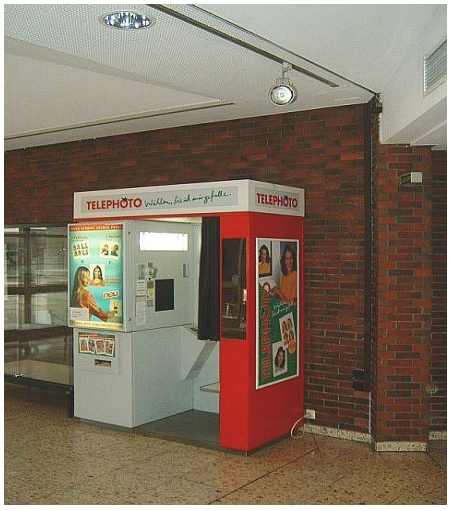

Рисунок 1.1 – Фотобокс

У сучасному значенні фотобоксами називаються також термінал (рисунок 1.2), призначений тільки для автоматичної роздруківки цифрових фотографій з носія клієнта. За ГОСТ це - вид побутової послуги з моментального отримання фотовідбитків в присутності замовника [3].

Рисунок 1.2 – Приклад сучасного терміналу друку фотографій (фотоавтомата) Photojet моделі Premium

Першу патентну заявку на фотобокс в 1888 році подали інженери з Балтімора Вільям Поуп і Едвард Пул [4]. Працездатний пристрій через рік запатентував і створив французький винахідник Теофіл Енгельберт. У 1889 році його фотоавтомат демонструвався на Всесвітній виставці в Парижі [5] [6]. У 1924 році на Бродвеї в Нью-Йорку була запущена перша фотокабіна, сконструйована американським винахідником російського походження Анатолем Марко Джозеф (Анатолій Йозефович) [7]. Фотобокси користувалися величезною популярністю і приносили такий прибуток, що вся компанія «Photomaton» через кілька років була продана за мільйон доларів. Комерційному успіху не змогли перешкодити навіть Велика депресія і Друга світова війна [8].

Фотобокс складається з двох відсіків, в одному з яких на стільці розташовується клієнт, а в іншому змонтовано фотообладнання [1]. Апаратний відсік перших фотокабін крім купюроприймача, фотоапарата і освітлювальних фотоспалахів містив фільм-процесор револьверного типу, який виробляв хіміко-фотографічну обробку знімків. Через це ранні моделі автоматів потребували підключенні до водопроводу і каналізації. Пізніші кабіни були автономними і могли обходитися без обслуговування до трьох тижнів [8].

Зйомка велася на звернутий фотопапір без проміжного негативу, завдяки чому час готовності фотографій не перевищувавв 3-5 хвилин [9]. В процесі

роботи автомата через певний інтервал часу послідовно робилися кілька знімків (найчастіше 4), які виходили у вигляді серії на смужці фотопаперу.

Сучасні фотоавтомати складаються з цифрового фотоапарата, найпростішого комп'ютера з контрольним монітором і кольорового фотопринтера. Багато з них збираються кустарним чином зі стандартних компонентів. Сучасна цифрова фотокабіна оснащується більш досконалим управлінням, що дозволяє вибирати формат і інші параметри знімка. Контроль зображення відбувається не по дзеркалу, як це було в аналогових пристроях, а по монітору, встановленому поряд з об'єктивом камери. Після зйомки на моніторі вибирається один знімок, який роздруковується в потрібній кількості примірників. Надалі фотокабіни стали називатися також пристрої, які виробляють фотозйомку, а також призначені для автоматичного друку цифрових знімків, зроблених клієнтом самостійно.

Найбільш розвинена мережа фотобоксів в Великобританії, Японії і Франції. Багато країн також мають зрілий ринок фотокабін, але з більш низьким рівнем. До них відносяться Німеччина, Італія, Іспанія, країни Бенілюксу і Скандинавії. В Україні більшість автоматів, встановлених в громадських місцях, були іноземного виробництва, головним чином моделі британської компанії «Photo-Me International» [8] .

З поширенням цифрової фотографії фотокабіни втратили своє розважальне значення, і використовуються тільки для документів. Ентузіасти міжнародного фото руху підтримують нечисленні аналогові фотоавтомати, деякі з яких продовжують працювати в якості атракціону в громадських місцях.

Розглянемо детальніше програмне забезпечення, яке зараз використовується для організації робоит фотобоксів.

Програмне забезпечення «Фотобокс» призначено для автоматизації роботи стаціонарних фотокабін. Під управлінням програми фотокабіна працює абсолютно автономно і не вимагає присутності обслуговуючого персоналу.

Все управління і настройка ПЗ виконується за допомогою сенсорного екрану. Таким чином, немає необхідності в постійному використанні миші, а в більшості випадків можна обійтися і без клавіатури.

ПЗ абсолютно несприйнятливі до різкого відключення живлення, інших нештатних ситуацій. Операційна система і керуюча програма відкриваються в режимі "тільки для читання". Це виключає знос носія інформації і значно підвищує надійність.

Рисунок 1.3 –Інтерфейс системи

ПЗ постійно вдосконалюється, розширюється набір доступних опцій. Установка оновлень ПЗ зводиться до простого копіювання файлів і займає всього кілька хвилин. При цьому всі налаштування, а так же статистика і фінансова інформація зберігаються незмінними.

Функціональні можливості ПЗ "Фотобокс":

- Друк кольорових і чорно-білих фото на документи різного формату.
- Друк одно-, двох-, трьох- і чотирьох кадрових фото.
- Використання барвистих шаблонів і брендування для фото, розміщення логотипів і т.д.
- Друк фотополосок 5x15 см. Функція доступна тільки при використанні нетипових принтерів. При використанні струйного принтера - потрібна ручна нарізка фотополосок.

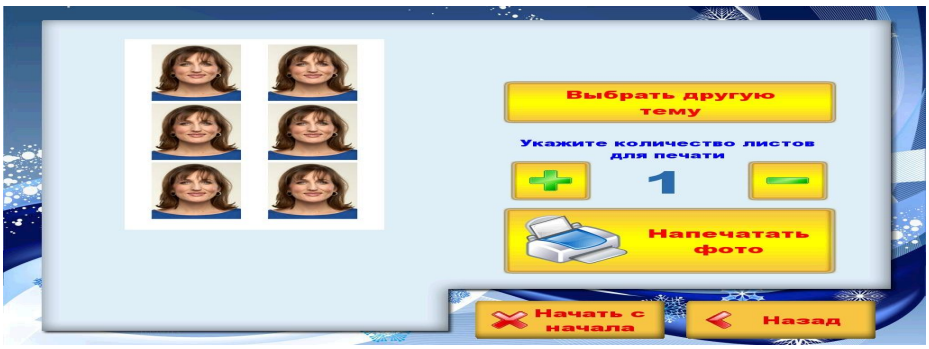

Рисунок 1.4 – Інтерфейс системи ПЗ "Фотобокс"

- Друк фото з флешок і карток пам'яті.
- Голосові підказки для спрощення орієнтування користувача в інтерфейсі фотокабіни. Можливість перемикання між різними мовами інтерфейсу і голосових підказок. В даний час доступні російська, українська, англійська мови інтерфейсу.
- Підтримка великої кількості різних купюроприймачів. Дозволяє фотокабіні працювати в вендінговому режимі абсолютно автономно. Можливість перемикання між оплатою після фотографування та передоплатою перед самим фотографуванням.
- Асинхронний друк фото (поки йде друк, інші користувачі можуть починати фотографуватися).
- Можливість роботи фотокабіни в повністю автономному режимі: автоматичне включення живлення камери та принтера після завантаження ПЗ, включення / відключення фотокабіни таймером.

Рисунок 1.5 – Інтерфейс системи ПЗ "Фотобокс"

Відправлення сервісних СМС-повідомлень з інформацією про статус фотокабіни, а також повідомлень про виникнення неполадок обладнання.

Основним недоліком даного програмного забезпечення є орієнтація на спеціальне апаратну частину, а також відсутність придбання окремо від апаратної частини.

Програмне забезпечення фото кабіни «МигФото» (рисунок 1.6).

Основними перевагами програми є:

1. Використання функції автоматичного вирівнювання фотографії відповідно до ГОСТу.

2. Гнучка настройка роботи і дизайну програми:

- Додавання нових і коректування існуючих типів фото на документи.

- Додавання нових шаблонів для розважальних фотографій і коригування існуючих (разом з основною програмою МігФото надається додаткова програма для швидкого і зручного додавання розважальних фотографій, щоб додати новий шаблон Вам буде потрібно всього пару хвилин).

- Зміна ціни для кожного типу фото індивідуально.

- Встановлено функція «стоп друк», яка стежить за кількістю паперу і не дозволить зробити ніяких дій клієнта, якщо папір закінчиться.

- У клієнта є п'ять спроб, щоб зробити фотографію. Кожну з п'яти спроб він може вибрати і відредагувати, як йому подобатися.

Рисунок 1.6 – Система "МигФото"

- Крім автоматичного вирівнювання клієнт може скористатися налаштуванням фотографії, виходячи з прикладу, який буде запропонований.

- Можливість програми працювати з трьома принтерами, тим самим збільшуючи автономність кабіни (читати розділ "ціни"), а також дозволяючи друкувати на різних типах паперу.

- Можливість підключення монетоприймача і чекового принтера.

- Можливість редагування фотознімків за рахунок панелі налаштувань: яскравість, контрастність, насиченість і т.д.

- Можливість підключення додаткової функції online-контроль.

- Можливість додавання своєї реклами на фотографії

- Всі дії в фотокабіні супроводжуються аудіо підказками.

- Якісна і стабільна робота програми.

3. Зрозумілий і зручний інтерфейс. З точки зору споживача, всі фотокабіни розглядаються як автомат моментального фотографування, тобто, простою мовою, швидко сфотографуватися і піти задоволеним, отримавши хороший знімок. Тому інтерфейс програми повинен бути настільки оптимізований і простий, щоб не викликати у потенційного клієнта уявлення, що це щось надприродне.

Підводячи підсумо проаналізованих програмні системи необхідно зробити наступні висновки:

- Відомі рішення досить дорого вартісні;
- Орієнтовані на поєднання з певним типом апаратного програмного забезпечення;
- Відсутність супутніх сервісів для друку фотографії із соціальних мереж, завантаження фотографій з мобільних пристроїв, та створення фото з камер.

1.2Специфікація вимог до системи

В результаті аналізу існуючих на ринку рішень були представлені наступні функціональні вимоги до системи:

- Пошук по користувачах та хештегах зображень і їх завантаження з соціальних мереж:, Instagram і Facebook.

- Формування та подальше редагування фотографій з завантажених зображень (можливість задати дату, місце, хештег і підпис для фотографії).

- Можливість зробити фотографію зі вбудованих пристроїв відеозапису і можливість сформувати з неї фотографію.

- Отримання зображення з призначених для користувача пристроїв на базі Android і IOS і сформувати з них фотографії.

- Оплата сформованих фотографій.

- Друк сформованих фотографій.

- Наявність засобів адміністрування для зміни параметрів система, моніторингу стану обладнання, тестування обладнання і перегляду статистики продажів.

Система повинна забезпечувати можливість комунікації з існуючим сервером для моніторингу стану пристроїв і відстеження статистики продажів.

Нефункціональні вимоги:

- Система повинна працювати на платформі Windows Embedded 7.

В результаті аналізу даних вимог були виявлені наступні типи користувачів:

- Клієнт. Тип користувача складається з людей, які здійснюють завантаження, покупку і друк фотографій.

- Адміністратор. До цього типу належать люди, які здійснюють налагодження даного ПЗ, відстеження стану обладнання і здійснюють моніторинг статистики.

В загальному випадку специфікація включає наступне:

- глосарій проекту;

- опис варіантів використання.

Наведемо список основних термінів та понять в області розробки web-сервісу для завантаження та друку зображень із соціальних мереж – глосарій. Глосарій – список понять в специфічній області знання з їх визначеннями [3]. Ці поняття та визначення подано у таблиці 1.1.

Таблиця 1.1

Глосарій

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту	
Соціальна мережа	платформа для встановлення зв'язків між людьми на основі спільних інтересів, знайомства в реальному житті або якихось інших спільних рис.
Фреймворк	множина класів, які володіють певною функціональністю, вбудовані у великі програмні структури і які контролюють ці програмні структури (Inversion of Control).
Хештег	ключове слово для групування деякого множини об’єктів.
Фотографія	в контексті даної роботи, об’єкт, сформований з завантаженого зображення і підготовленого для друку.
API	набір готових класів, процедур, функцій, структур і констант, наданих додатком (бібліотекою, сервісом) або операційною системою для використання у зовнішніх програмних продуктах.
2. Користувачі системи	
Адміністратор	Користувач системи, який, окрім виконання функцій користувача, може здійснювати управління користувачами системи
Клієнт	Особа, яка здійснює операції, які пов’язані з завантаженням, оплатою та друком фотографій
1. Вхідні та вихідні документи	
Список фотографій	Список фотографій із зазначенням дати, підписів, формату та цін

Квитанція	документ, що видається і одержується в разі сплати за певні грошові суми, цінності, матеріали, устаткування або за виконану роботу

Сформовані функціональні вимоги представлені у вигляді наступних діаграм варіантів використання (рисунок 1.7-1.8).

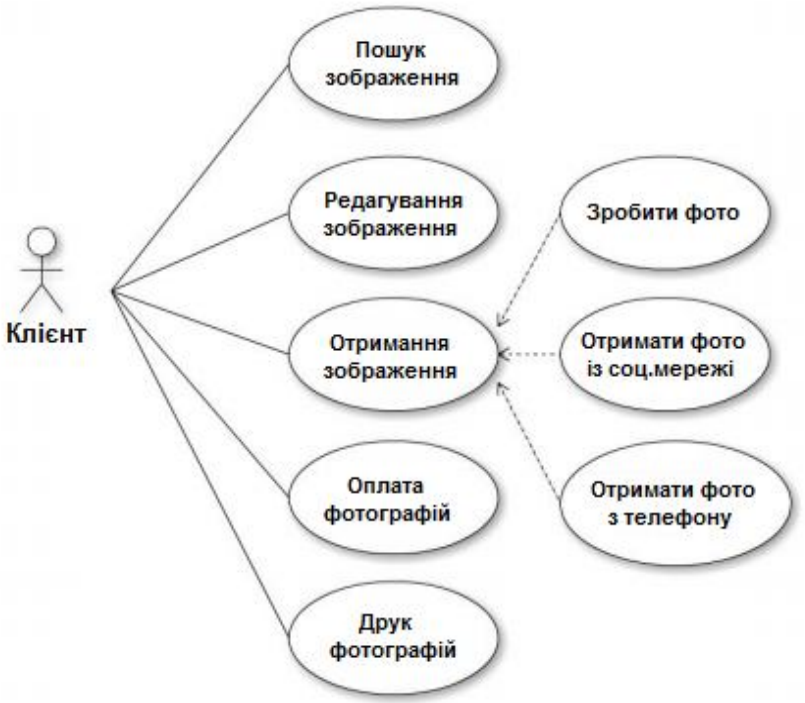

Рисунок 1.7 – Діаграма варіантів використання для клієнта

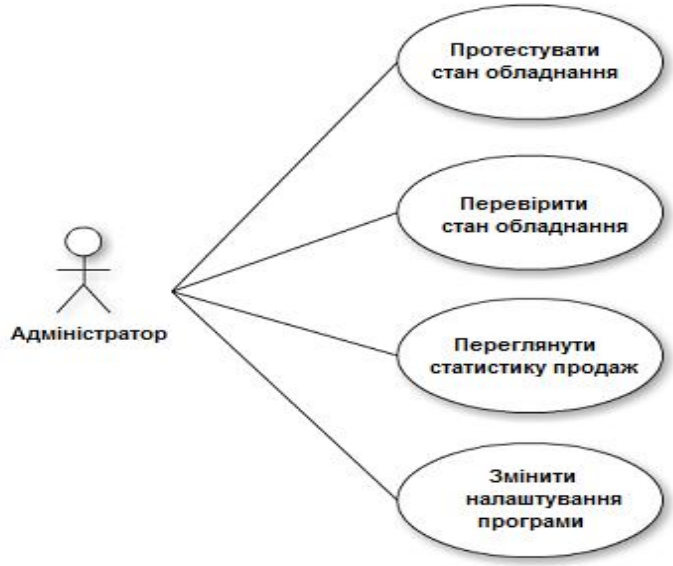

Рисунок 1.8 – Діаграма варіантів використання для адміністратора

Наведемо специфікацію суттєвих для проекту нефункціональних вимог:

1. Застосовність:

- мінімальний час для навчання звичайних і досвідчених користувачів;
- відповідність стандартам графічного інтерфейсу Web user interface.

2. Надійність:

- постійна безвідмовна робота;
- пропускна здатність каналу зв’язку 100 Mb/s;
- забезпечення можливості віддаленого доступу до комп’ютера, на якому буде встановлена система;
- доступність – 5%.

3. Робочі характеристики:

- швидкість завантаження інтернет-ресурсу: 0,1 – 1 с;
- число транзакцій: 100 / 1 с;
- використання ресурсів: від 1 Gb, в залежності від кількості операцій.

4. Проектні обмеження:

- Windows Embedded 7;
- веб-сервер IIS 7/8;
- Microsoft .NET Framework v4.0;
- SQlite;

5. Вимоги до документації

- наявність інтерактивної довідки.

6. Інтерфейси:

інтерфейс користувача – Web user interface.

1.3 Розроблення архітектури програмної системи та проектування структури бази даних

При проектуванні архітектури додатку застосовувався шаблон MVVM (рисунок 1.9), використання якого передбачається технологією WPF і фреймворком PRISM. Застосування шаблону дозволяє розділити представлення і бізнес-логіку додатку, тим самим зменшується зв'язаність додатку і з'являється можливість розробки і зміни представлення і бізнес логіки окремо один від одного.

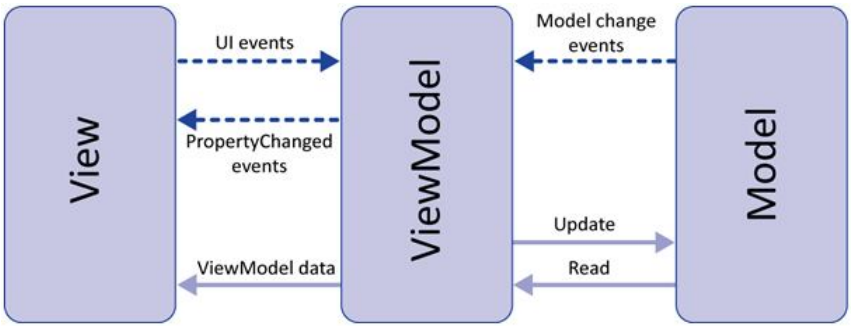

Рисунок 1.9 – Загальна архітектура системи

Модель (Model) - зазвичай мається на увазі частина програми, яка містить в собі функціональну бізнес-логіку. Модель повністю незалежна від інших частин програми і при цьому нічого не повина знати про елементи представлення. За допомогою цього досягається результат, який дозволяє змінювати подання даних, спосіб їх відображення, не змінюючи саму Модель.

- Пердставлення (View) - відповідає за представлення даних моделі, представляючи собою призначений для користувача інтерфейс. Нічого не знає про моделі і не взаємодіє з нею безпосередньо.

- Модель представлення (ViewModel) - інкапсулює логіку представлення і дані для відображення. Так само модель представлення перетворює дані з моделі в вид, специфічний для подання.

Як правило, модель представлення визначає команди для взаємодії з користувачем і координації взаємодії з моделлю, команди дозволяють інкапсулювати взаємодію з користувачем від представлення.

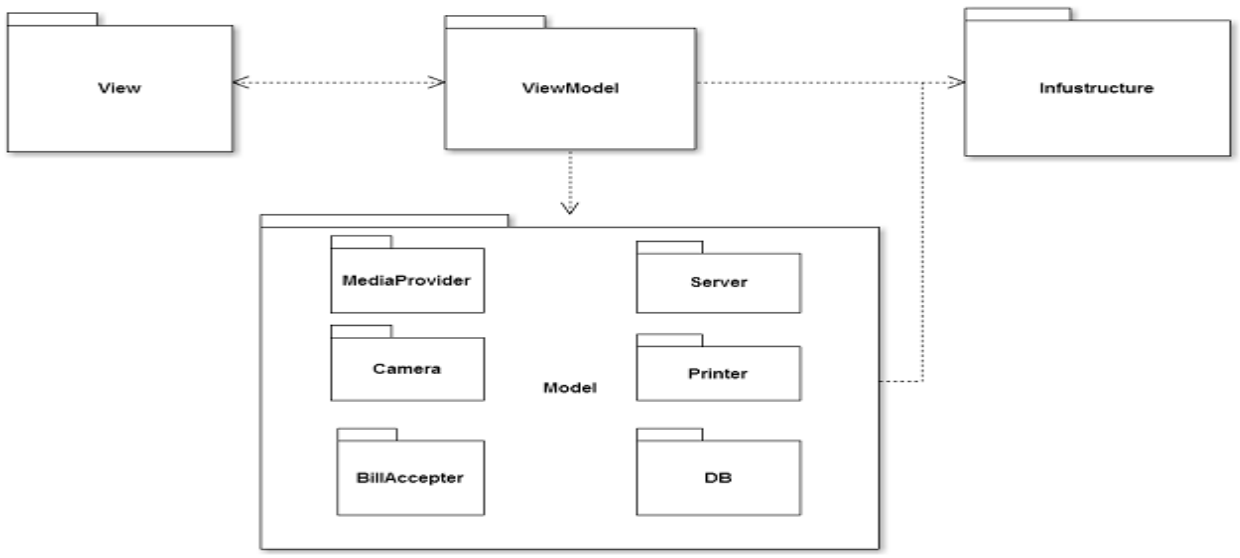

Рисунок 1.10 – Архітектура системи у вигляді діаграми пакетів

Навігація. Зміна екранів в додатку здійснюється через навігацію, процес зміни відображено на рисунку 1.11

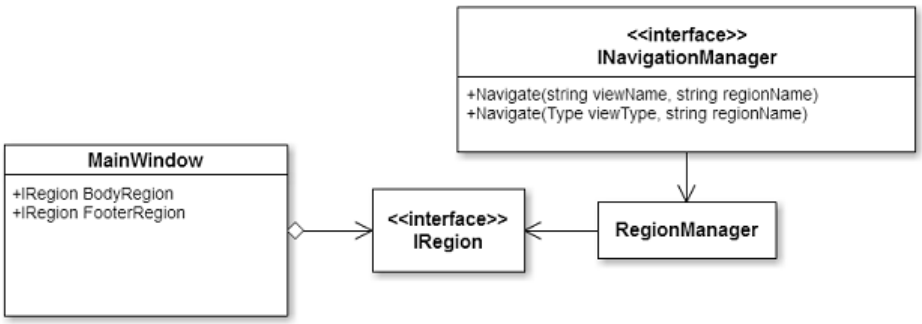

Рисунок 1.11 – Діаграма класів навігації

На діаграмі представлена архітектура навігації в додатку, де:

- MainWindow - головне вікно програми, є контейнером для різних представлень.

- IRegion - інтерфейс фреймворка PRISM, що дозволяє елементам представлення, які його реалізують взаємодіяти з системою навігації.

- RegionManager - клас фреймворка PRISM, що містить інформацію про регіони і представлення, що містяться в них. Так само містить методи, які дозволяють здійснювати навігацію в регіонах.

- INavigationManager - інтерфейс, що інкапсулює функціональність навігації в регіонах.

Більшість представлень повинні відображати деякі дані з Model, для зберігання, перетворення цих даних і оброблення дій користувача представлень, які містять модель представлення (ViewModel).

На рисунку 1.12 зображений клас ViewModelBase, який є батьком для всіх моделей представлень в програмі.

- INotifyPropertyChanged - інтерфейс містить подію PropertyChanged, що повідомляє всіх передплатників про зміну деякої властивості об’єкта.

- BindableBase - абстрактний клас, який реалізує роботу INotifyPropertyChanged.

- ViewModelBase - абстрактний клас моделі представлення.

Для відповіді на дії користувача з інтерфейсом WPF передбачає використання команд.

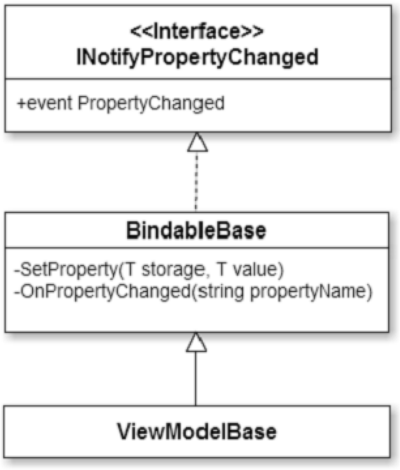

Рисунок 1.12 – Діаграма класів для ViewModelBase

Команда – об’єкт, який дозволяє розділити логіку і призначений для користувача інтерфейс, такий об'єкт реалізує інтерфейс ICommand і міститься в моделі представлення.

На рисунку 1.13 представлена реалізація ICommand: ICommand – інтерфейс, який визначає команду. RelayCommand - реалізація команди.

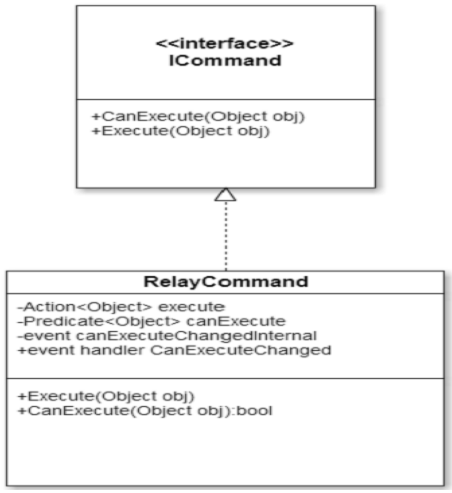

Рисунок 1.13 – Реалізація ICommand

Для зберігання даних про продажі та ціни для різних форматів через СУБД Sqlite була створена база даних, представлена на рисунку 1.14.

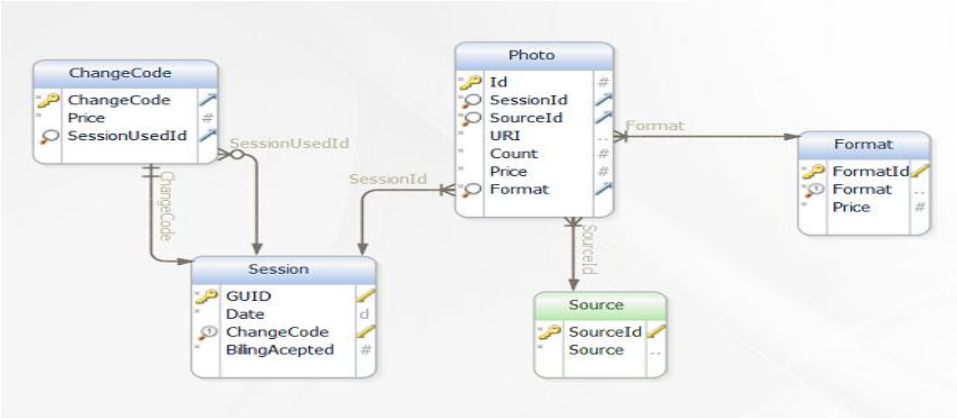

Рисунок 1.14 – Схема бази даних

Session - таблиця, що представляє собою факт продажу.

ChangeCode - таблиця з кодами здачі, отриманими користувачем після оплати, коди здачі не впроваджені в поточну роботу.

Photo - таблиця з роздрукованими за сесію фотографіями.

Format - таблиця, яка містить формат роздрукованих фотографій і їх ціну.

Source - таблиця із зазначенням джерела фотографій - ім'я конкретної соціальної мережі.

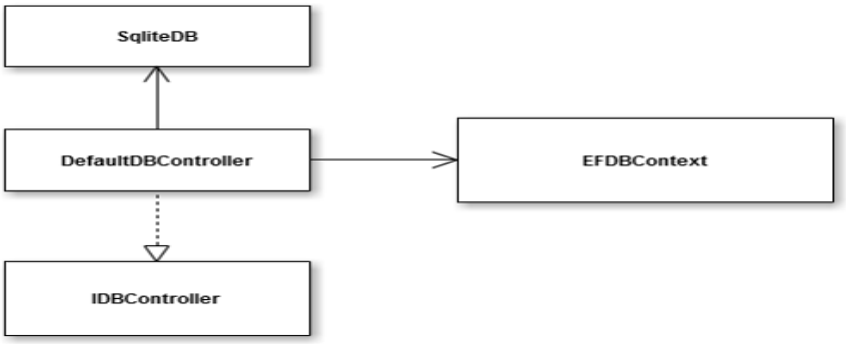

Рисунок 1.15 – Діаграма класів для взаємодії з базою даних

Взаємодія з базою даних описано на рисунку 1.15.

- IDBController - інтерфейс, що визначає поводження класів, працюючих з БД.

- DefaultDBController - клас для роботи з Sqlite базою даних.

- SqliteDB - драйвер для роботи з Sqlite СУБД.

- EFDBContext - являє собою агрегат з сутносей, які представляють собою записи таблиць БД.

1.4. API соціальних мереж використання протоколу OAuth

Взаємодія з соціальними мережами здійснюється за допомогою наданих сервісами API.

Сучасні соціальні мережі забезпечують взаємодії зі сторонніми додатками через REST API.

REST (Representational State Transfer - «передача стану представлення») - це набір принципів для побудови інтерфейсу взаємодії зі сторонніми сервісами. REST інтерфейс володіє такими принципами:

- Незалежність від стану - сервер не зберігає або відстежує контекстну інформацію про клієнта.

- Клієнт - серверний розподіл і єдиний інтерфейс – єдиний інтерфейс між сервером і клієнтом, цей поділ має на увазі відсутність зв'язку між клієнтами і серверним сховищем даних, що дозволяє повністю приховати деталі реалізації від клієнта.

- кешувальна і багаторівнева архітектура - відповідь сервера може кешуватися на певний період часу і використовується повторно без нових запитів до сервера.

Так само соціальні мережі використовують протокол OAuth 2.0 – протокол безпечної авторизації користувача сервісу на клієнті третьої сторони без надання даних третій стороні.

Instagram API. Instagram - соціальна мережа для обміну зображеннями і відео. Сервіс Instagram API для взаємодії передбачає реєстрацію додатку через діючий акаунт Instagram і подальше отримання ClientID і ClientSecret - поля, що прикріплюються до кожного запиту для ідентифікації додатку. Після реєстрації програма має пройти процедуру перевірки з боку сервісу для активації даних ідентифікації додатку.

Instagram API ділиться на кілька підсистем, іменовані Endpoints[5]:

- Users - підсистема для взаємодії з акантами користувачів: пошук, щоб отримати додаткову інформацію.

- Relationships - підсистема для відстеження та управління взаємозв'язками між користувачами.

- Comments - підсистема відстеження та управління коментарями до зображень і відео.

- Likes - підсистема відстеження та управління лайками (Like), дискретна величина схвалення користувачами зображення або відео

- Tags - підсистема, що здійснює взаємодію і управління зображеннями або відео через хештеги.

- Locations - підсистема, що здійснює взаємодію і управління зображеннями або відео через локацію, мітку зображення або відео, що представляє собою координати і адресу деякого місця.

Запити до сервісу здійснюються через URL: https://api.instagram.com/v1/. До кожного запиту повинні бути прикріплені поля ClientID і ClientSecret, що містять дані для ідентифікації додатку комунікації з сервісом. Так само деякі запити вимагають ACCESS_TOKEN - поля одержуваного при авторизації користувача через протокол OAuth2.

Кожна відповідь сервера на запит являє собою JSON об’єкт певної структури (рисунок 1.16).

Рисунок 1.16 – Структура відповіді від сервісу Instagram API

Відповідь має наступні поля:

- meta - поле містить поле code, що визначає стан запиту, може містити код певної помилки.

- data - поле з даними відповіді, специфічними для данного запиту.

- pagination - поле, що містять дані для переміщення через послідовні блоки даних, наприклад фотографії користувача або результат пошуку користувачів за певним ключу.

Facebook API. Facebook - одна з перших соціальн мереж в традиційному розумінні, має можливості створення профілю, поширення медіа контенту: фотографіями, відеозаписами і аудіозаписами, обміном повідомленнями: публічними і приватними.

Для запитів через GRAPH API потрібно отримати ClientID і ClientSecret для додатку, так само необхідно отримати ACCESS_TOKEN через авторизацію користувача в соціальній мережі.

Сучасні соціальні мережі для авторизації користувача на клієнті третьої сторони використовують протокол OAuth.

На рисунку 1.17 представлений алгоритм авторизації через протокол OAuth.

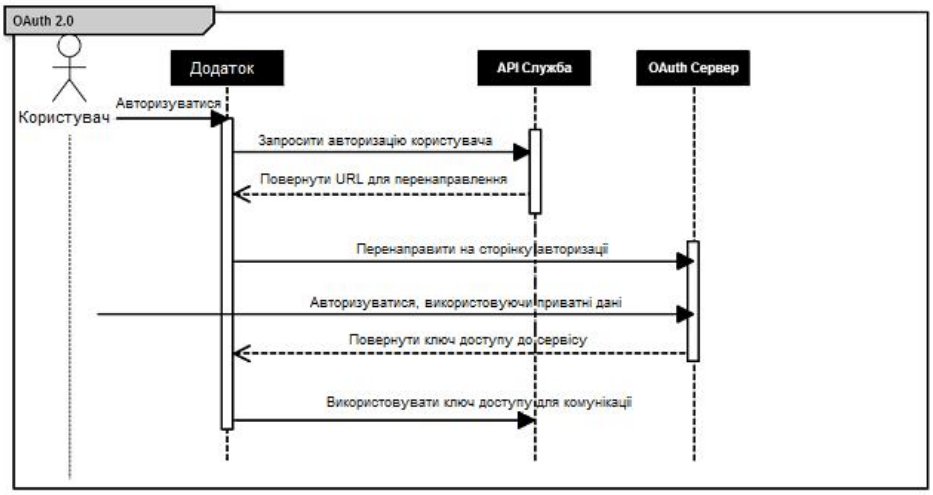

Рисунок 1.17 – Діаграма послідовності для авторизації через протокол OAuth

OAuth 2.0 - протокол безпечної авторизації, який дозволяє з додатком третьої сторони отримати доступ до HTTP-сервісу через організацію взаємодії між користувачем і сервісом авторизації без надання третій стороні даних авторизації користувача [6].

1.5 Висновки до першого розділу

Здійснено опис предметної області, напрями діяльності. Визначено склад функцій, що входять до бізнес-процесу на основі яких розроблено схему управління бізнес-процесом. Проведено аналіз відомих програмних систем. Здійснено аналіз вимог до програмної системи.

У цьому розділі розроблено архітектуру програмного додатку, що дозволить краще зрозуміти функції основних його частин. Створено та описано структурну схему, основними компонентами якої є: рівень клієнта, рівень бізнес-логіки та рівень даних. Описано функціональну структуру системи та її

основних елементів – модулів обробки даних. Визначено основні елементи бази даних та встановлено зв’язки між ними. Спроектовано структуру бази даних.

РОЗДІЛ 2. ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-СЕРВІСУ ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

2.1. Використовувані технології та стандарти

Для розробки даної системи були використані наступні технології:

- Мова програмування C# (рисунок 2.1).
- .NET (рисунок 2.2).
- Unity (рисунок 2.3).
- AForge.Net.Video.Directshow (рисунок 2.4).
- СУБД SQlite (рисунок 2.5).
- WPF (рисунок 2.6-2.7).
- PRISM (рисунок 2.8)

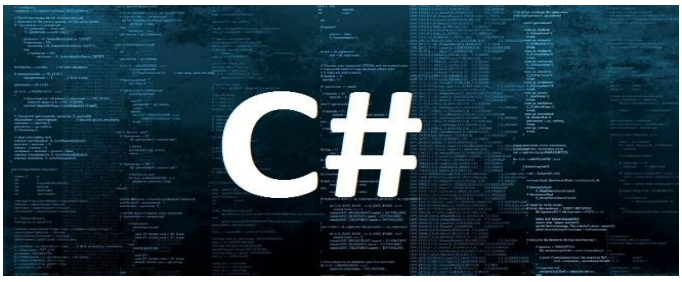

Рисунок 2.1 – Мова програмування C#

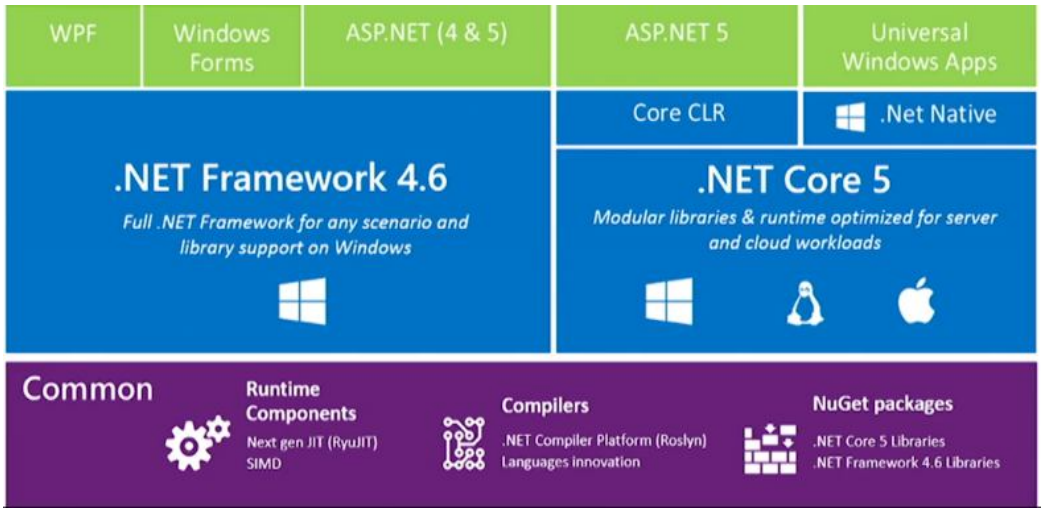

Рисунок 2.2 – NET Framework

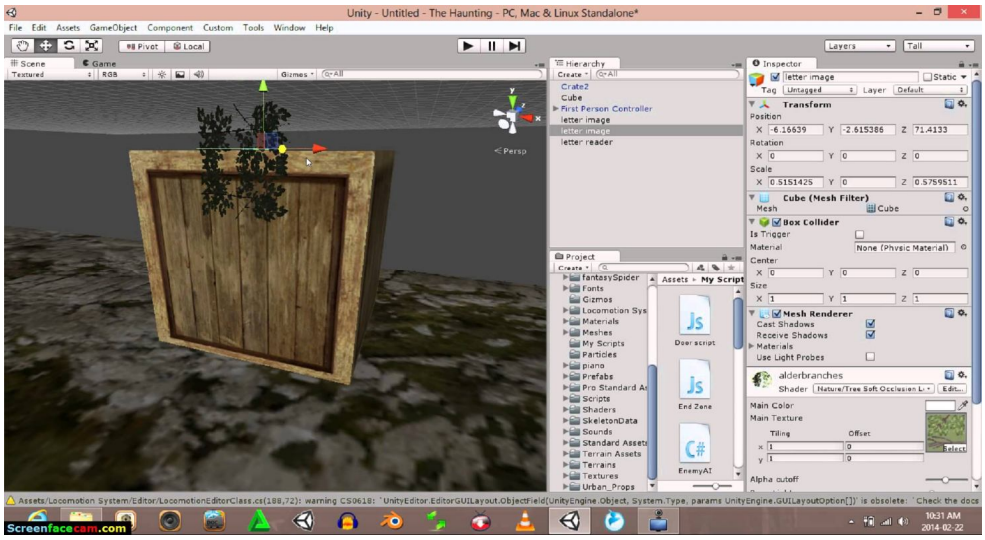

Рисунок 2.3 – Unity

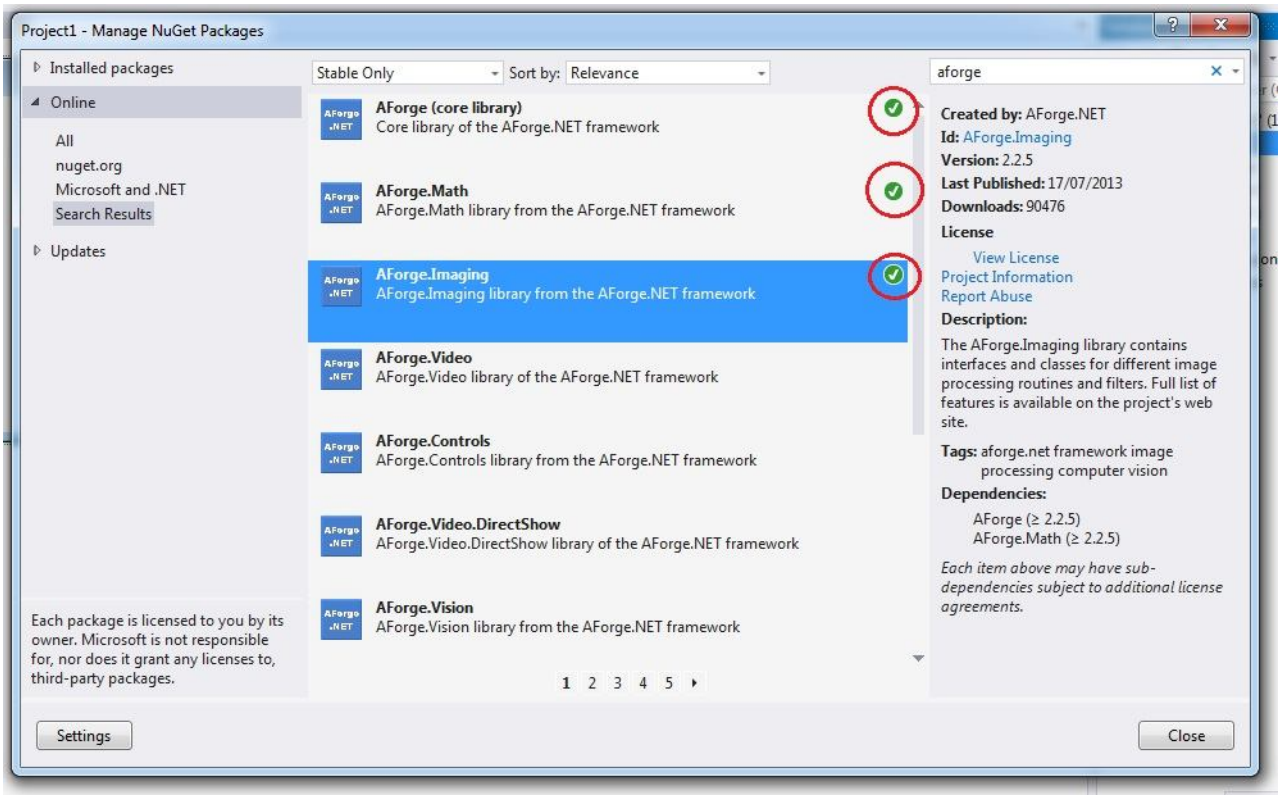

Рисунок 2.4 – AForge.Net.Video.Directshow

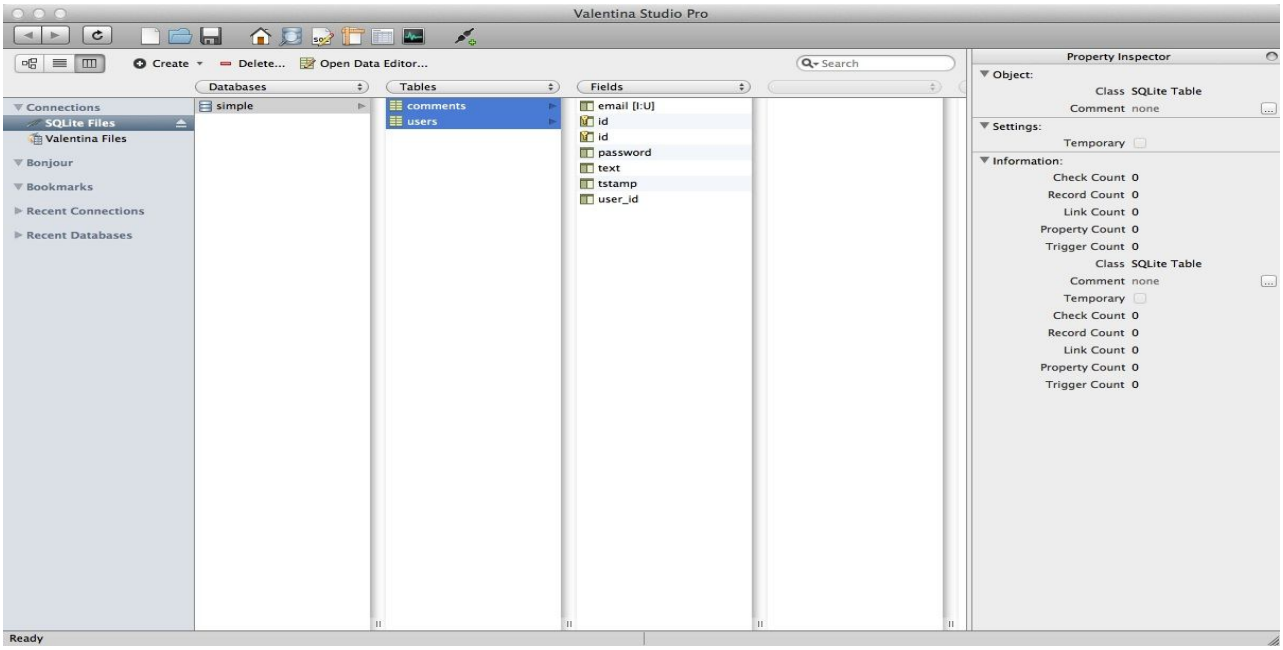

Рисунок 2.5 – СУБД SQlite

WPF або Windows Presentation Foundation представляє собою підсистему фреймворка .NET для створення графічного інтерфейсу [1].

Особливості WPF:

- XAML (Extensible Application Markup Language) - заснований на XML мові для декларативного опису призначеного для користувача інтерфейсу в різного роду додатках, зокрема, WPF додатках (рисунок2.7).

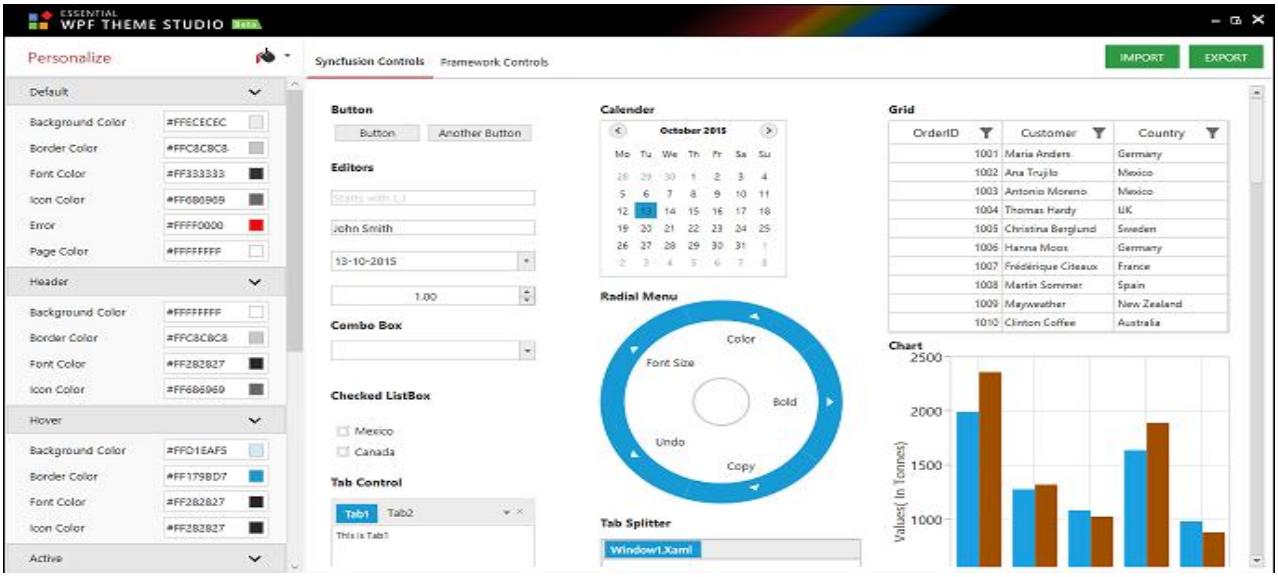

Рисунок 2.6 –WPF

```
<Window x:Class="PrinterSDKApplication.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:PrinterSDKApplication"
        DataContext="{Binding RelativeSource={RelativeSource Self}}"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <StackPanel Orientation="Vertical">
            <Button  Width="100" Height="100" Content="Print" Command="{Binding PrintCommand}"></Button>
            <TextBlock Name="TextBlock" Width="300"></TextBlock>
        </StackPanel>

    </Grid>
</Window>
```

Рисунок 2.7 – Приклад XAML опису інтерфейсу

Data Binding - механізм відображення і взаємодії даних з шару бізнес логіки в шарі UI. Даний механізм працює за допомогою повідомлення зміни прив'язаних даних через реалізацію інтерфейсу INotifyPropertyChanged. Повідомлення про зміни можуть відбуватися в будь-яких напрямках між шарами UI і бізнес логіки.

- Templates або шаблони, діляться на Control Templates і Data Templates. Кожен елемент WPF має Control Template – представлення елемента у вигляді візуального дерева, яке може бути змінено без зміни поведінки самого елемента. Data Template – механізм застосування шаблону для відображення прив'язаних до елементу візуального дерева даних.

- DataContext - властивість елемента візуального дерева, визначальна область видимості для Data Binding.

PRISM є фреймворк для створення композитних додатків на основі WPF або Silverlight з численними екранами UI, складним поданням і організацією даних і бізнес логіки. PRISM вирішує проблему організації складної бізнес-логіки і UI за допомогою поділу додатку на слабкозв'язані напівнезалежні частини, які інтегруються в головний додаток-оболонку (Shell) [4].

Особливості PRISM:

- Модулі (Modules) - пакети функціоналу, що мають роздільні цикли розробки.

- Оболонка (Shell) - головний додаток, в який модулі завантажуються і тестуються. Так само оболонкою називають головне вікно додатку, що розділяється на регіони.

- PRISM передбачає організацію додатку відповідно до шаблону MVVM (Model - ViewModel - View).

- DI Контейнер (Dependency Injection Container) – шаблон управліннями залежностями. PRISM надає розширення для наступних DI контейнерів: Unity, MEF або з контейнерами, які мають реалізацію Service Locator.

- Bootstrapper - підсистема використовувана для ініціалізації і реєстрації сервісів в DI (Dependency Injection) контейнері та ініціалізації Shell View.

- EventAggregator - клас реалізує шаблон "Спостерігач" [2]для забезпечення слабкої взаємодії різних частин програми.

- Регіони (Regions) - особливі елементи візуального дерева,які є контейнером для різних представлень. Дозволяютьстворити гнучку структуру програми, при якій зміни UI не тягнутьзміни в логіці програми. Зміна представлення здійснюєтьсяза допомогою навігації.

- Навігація (Navigation) - процес зміни представлень додатку. PRISM містить реалізації State-Based і View-Switchingнавігацій. Навігація здійснюється через регіони (Regions) і менеджеррегіонів (RegionManager), компонент системи, який здійснює контрольрегіонів, представлень в регіонах і їх зміну.

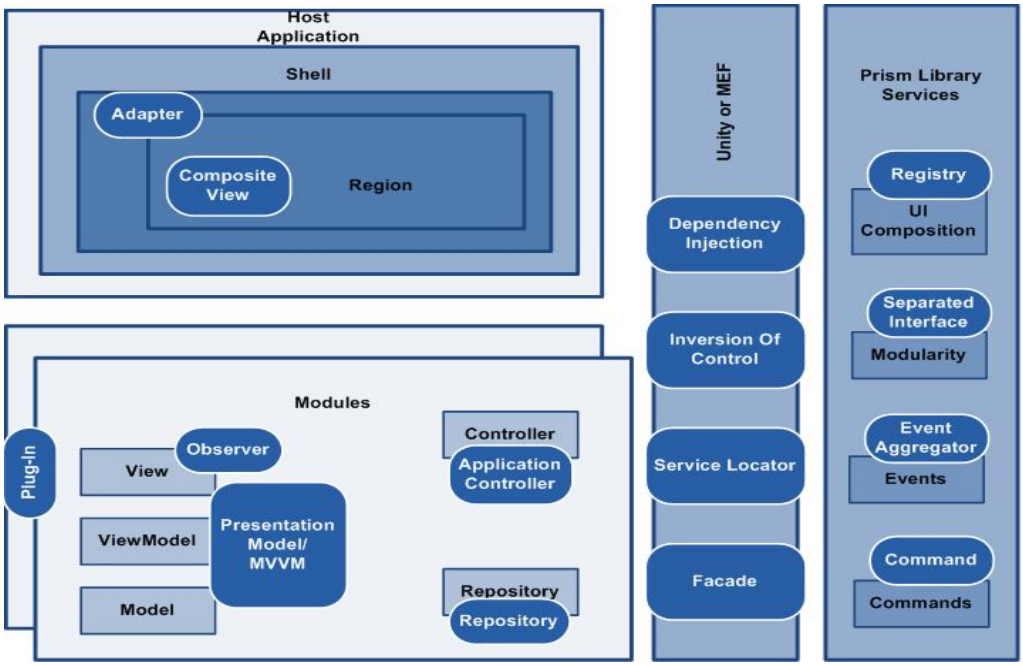

Рисунок 2.8 –. PRISM

2.2. Програмна реалізація отримання зображень, оплати, реалізація друку

Як джерело зображень можуть служити соціальні мережі, телефони користувачів на базі Android і IOS і веб-камера.

Вибір джерела зображень. Вибір джерела фотографій здійснюється через екран SourceSelectionViewModel.

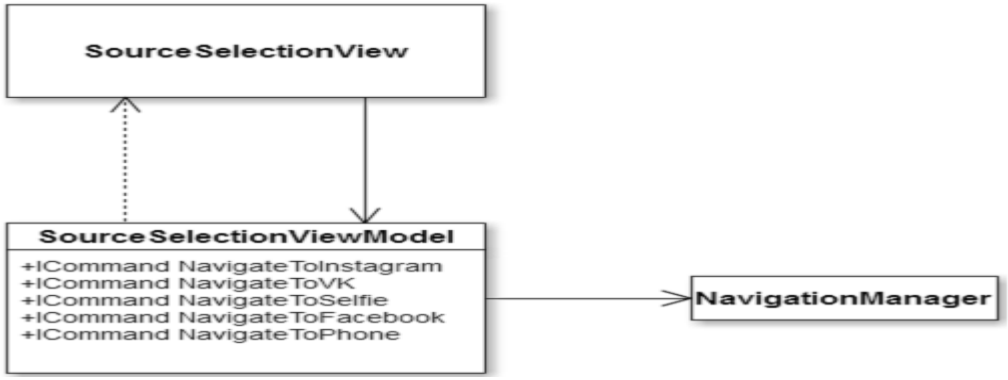

Рисунок 2.9 – Вибір джерела зображень

На рисунку 2.9 зображено ієрархію класів екрану вибору джерела зображень (рисунок 2.10).

- SourceSelectionView - екран вибору джерела.

- SourceSelectionViewModel - модель представлення екрану, яка містить команди, які здійснюють переходи на потрібний екран джерела.

- NavingationManager – класс, який здійснює навігацію між представленнями.

Пошук в соціальних мережах. В реалізованої роботі джерелом зображень можуть служити

соціальні мережі: Instagram, Facebook, пристрій користувача, який володіє WiFi модулем і веб-камера. Розглянемо архітектуру джерел за винятком камери (рисунок 2.11).

Рисунок 2.10 –. Сторінка вибору джерела зображень

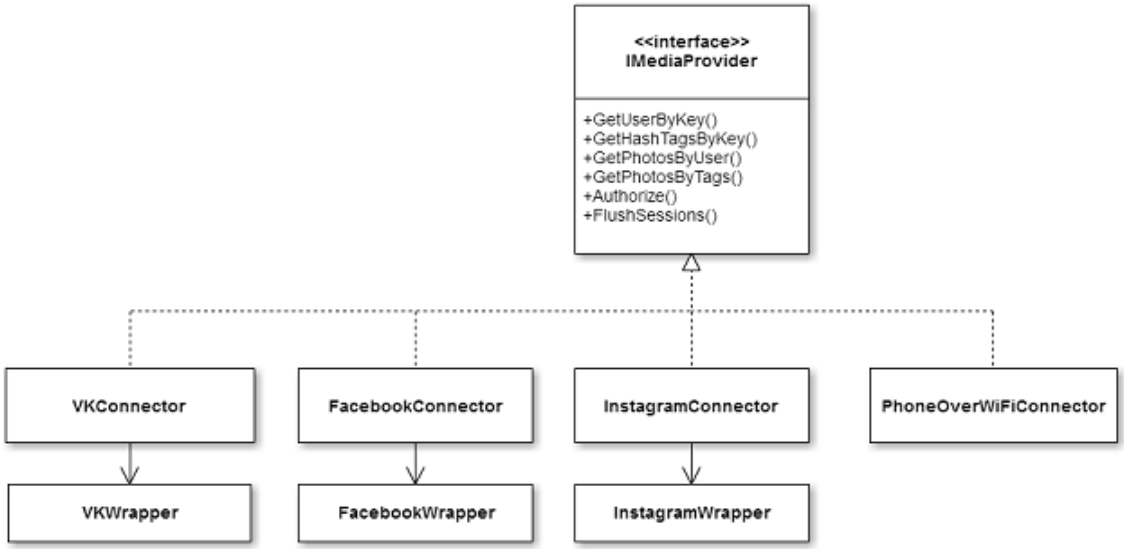

Рисунок 2.11 – Діаграма класів пакета MediaProvider

IMediaProvider - інтерфейс для взаємодії з джерелами зображень, що дозволяє працювати з користувачами, фотографіями і системами авторизації соціальних мереж.

- FacebookConnector - реалізація IMediaProvider для взаємодії з соціальною мережею Facebook.

- InstagramConnector - реалізація IMediaProvider для взаємодії з соціальною мережею Instagram.

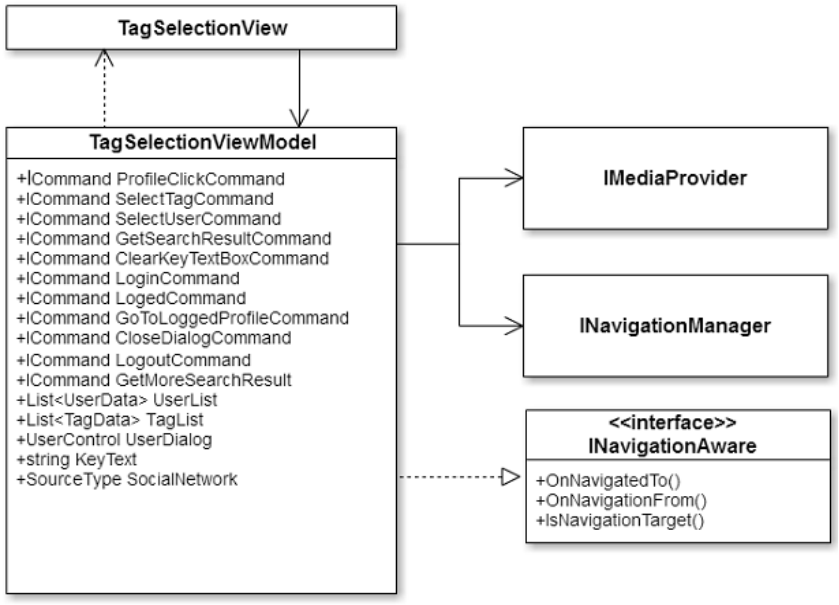

Рисунок 2.12 –Діаграма класів сторінки пошуку

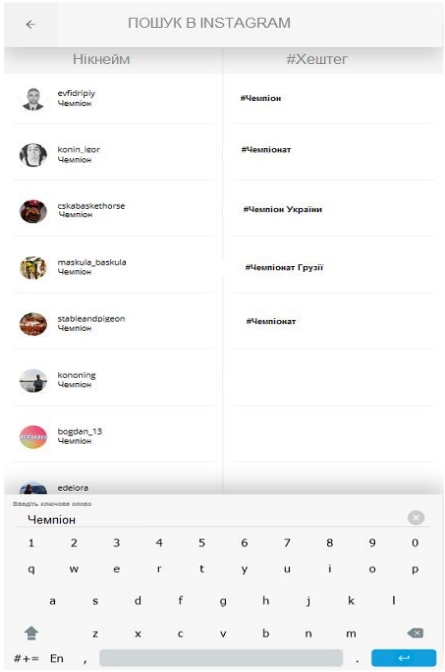

Рисунок 2.13 – Сторінка пошуку

Також на екрані пошуку користувач може провести авторизацію в соціальній мережі (рисунок 2.14) по протоколу OAuth 2.0, підтримуваний соціальними мережами.

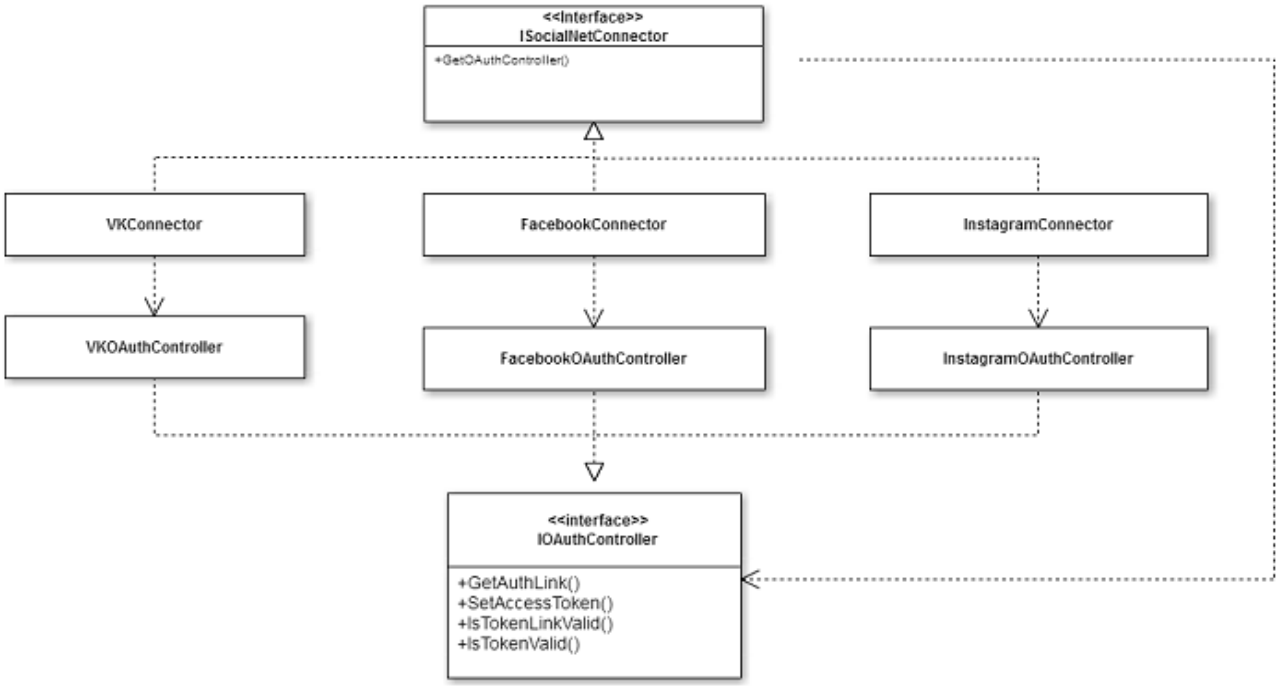

Рисунок 2.14 – Ієрархія класів для авторизації по протоколу OAuth 2.0

На рисунку 2.14 зображено ієрархію класів для авторизації в соціальних мережах:

- ISocialNetConnector - інтерфейс для отримання IOAuthController.

- IOAuthController - клас для отримання токена (ACCESS_TOKEN), що дозволяє працювати за протоколом OAuth.

- FacebookOAuthController, VKOAuthController, InstamatOAuthController - конкретні реалізація IOAuthController для соціальних мереж.

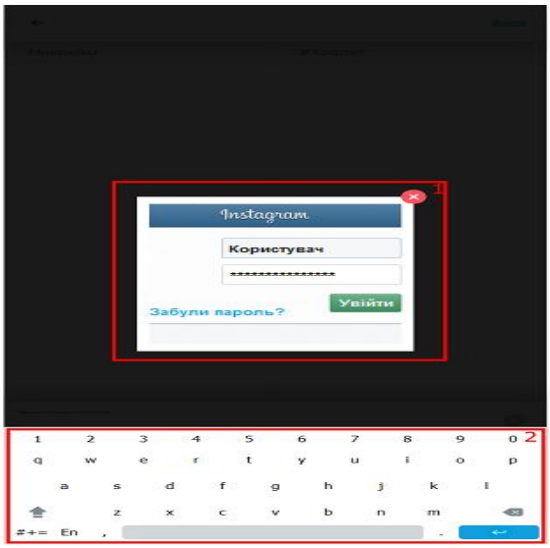

Рисунок 2.15 – Процес авторизації користувача на екрані пошуку

Отримання фотографій через WiFi. У роботі був реалізований сервер, що дозволяє завантажити фотографії (рисунок 2.16) через сайт, розгорнутий на локальному сервері, отримані фотографії потрапляють в директорію на комп'ютері з встановленим ПЗ.

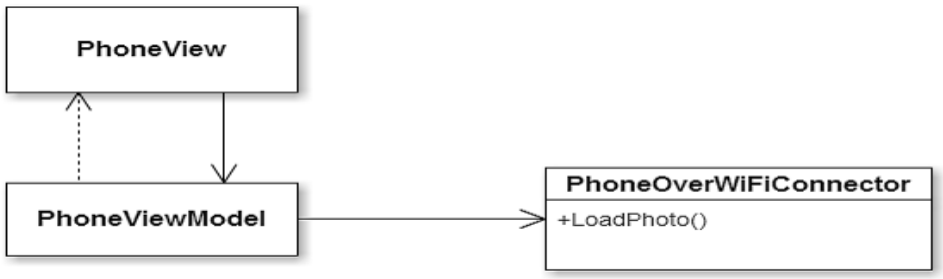

Рисунок 2.16 – Ієрархія класів екрану завантаження фотографій з телефонів

На діаграмі класів на рисунку 2.17 зображені:

- PhoneView - екран завантаження фотографій.

- PhoneViewModel - модель представлення екрана завантаження фотографій, отримує повідомлення про завантажених фотографій і завантажує їх в PhoneOverWiFiConnector.

- PhoneOverWiFiConnector - джерело фотографій, що містить методи для завантаження фотографій, реалізує IMediaProvider.

Рисунок 2.17 – Сторінка завантаження фотографій з телефону

Завантаження і вибір фотографій. Завантаження зображень здійснюється на екрані вибору фотографій (рисунок 2.18).

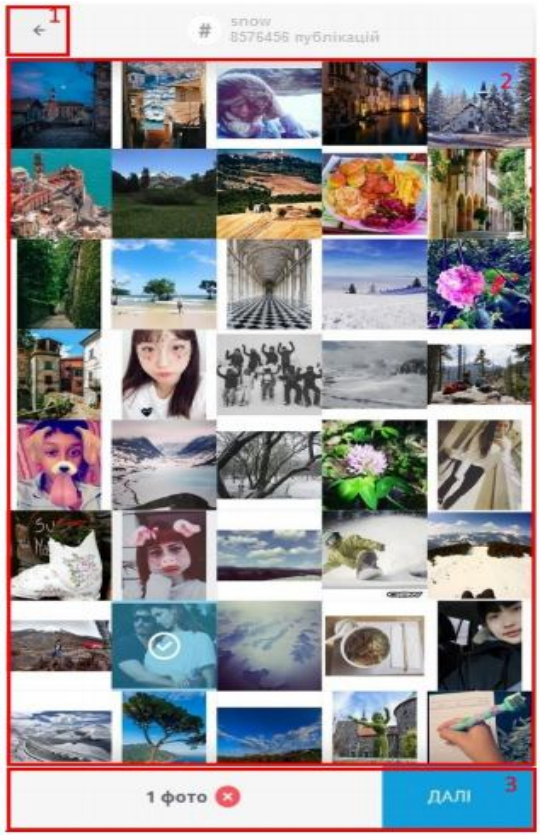

Рисунок 2.18 – Сторінка вибору фотографій

На рисунку 2.19-2.20 зображені класи, які здійснюють завантаження і відображення фотографій.

- PhotoWallView - екран вибору фотографій.

- PhotoWallViewModel - модель представлення екрана вибору зображень, яка ініціалізує завантаження зображень, збереження завантажених зображень, і збереження обраних зображень.

- PhotoDownloader - клас, який відповідає за завантаження зображень, який використовує IMediaProvider для отримання інформації про зображення і ImageHelper для їх завантаження.

- ImageHelper - клас для операцій над зображеннями.

- IPhotoDownloaderProvider - фабрика, яка містить логіку ініціалізації PhotoDownloader.

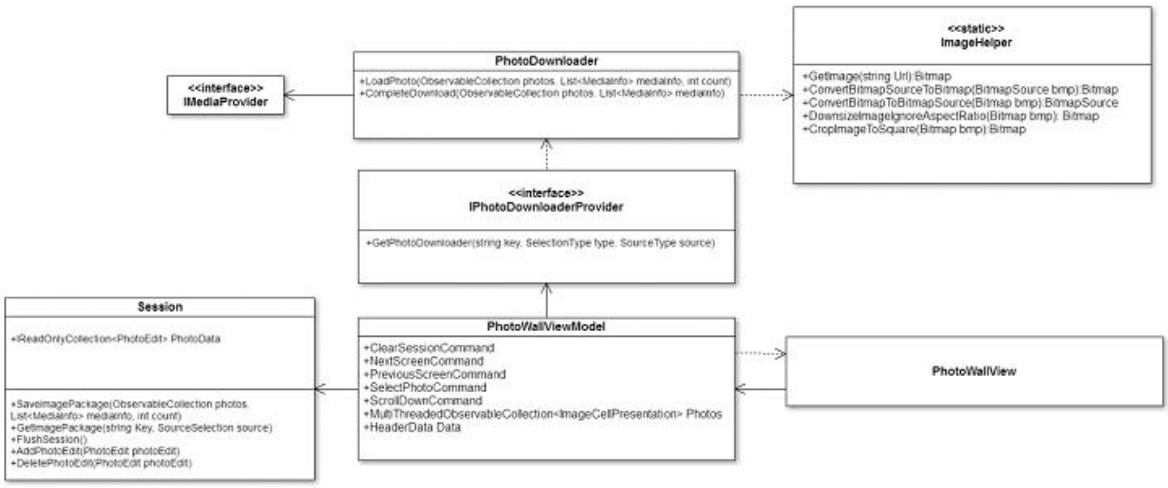

Рисунок 2.19 – Ієрархія класу вибору фотографій

Session - здійснює кешування завантажених фотографій, зберігання інформації по обраних фотографіях. Кеш зберігається на зовнішній пам’яті.

- IMediaProvider - містить методи для отримання інформації про фотографії.

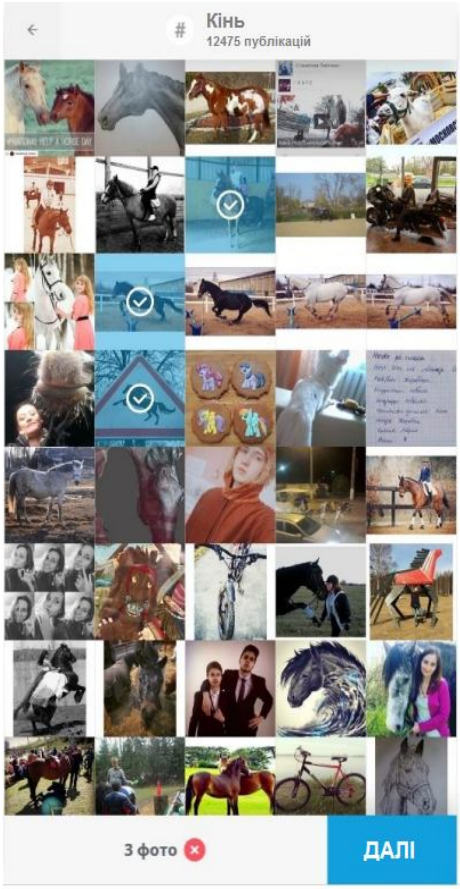

Рисунок 2.20 – Сторінка вибору фотографій

Отримання зображення з веб-камери. Одним із способів отримання зображення є веб-камера (рисунок 2.21-2.22).

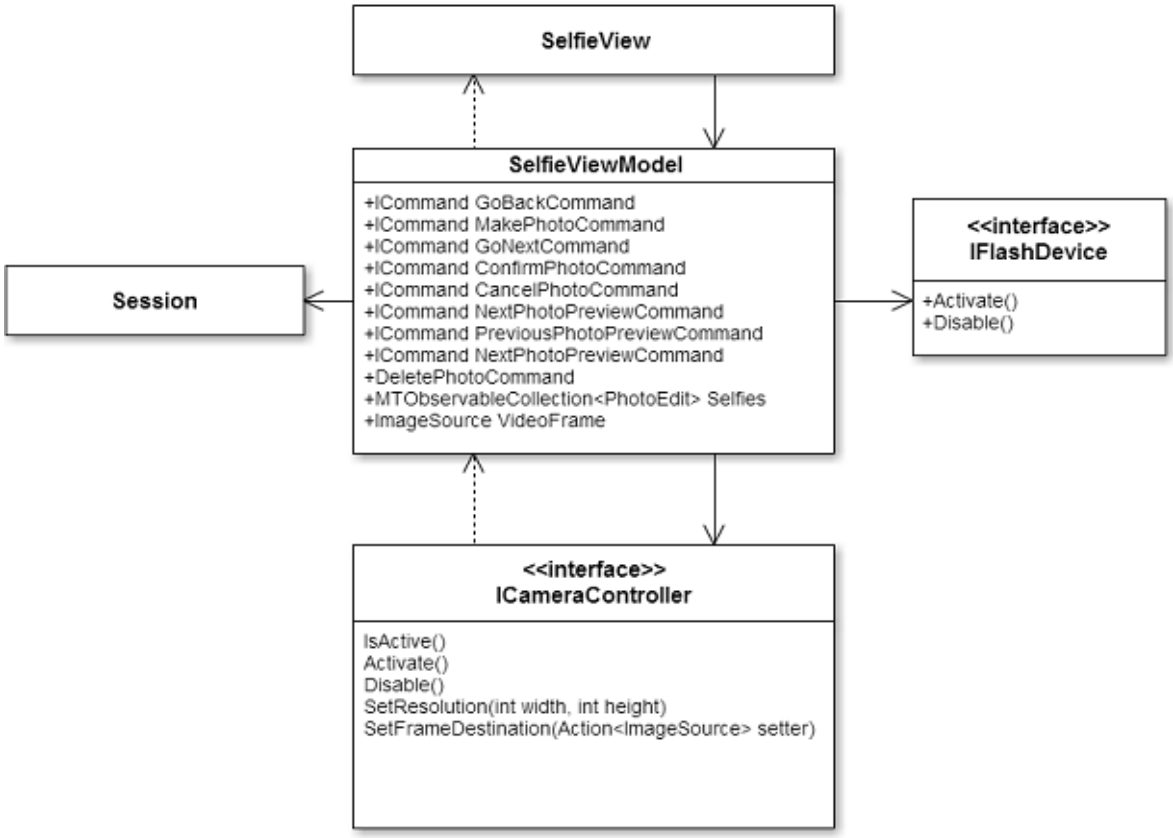

Рисунок 2.21 – Ієрархія класів екрану веб-камери

SelfieView - екран роботи з веб-камерою.

- SelfieViewModel - модель представлення екрана роботи з веб-камерою, управляє процесом відображення даних з веб-камери і збереженням, видаленням і відображенням зроблених фотографій.

- Session - репозиторій обраних зображень, при виході з екрану всі вибрані фотографії зберігаються в цьому об’єкті.

- ICameraController - оголошує інтерфейс для роботи з веб-камерою.

- IFlashDevice - оголошує інтерфейс для управління спалахом, включається при фотографуванні.

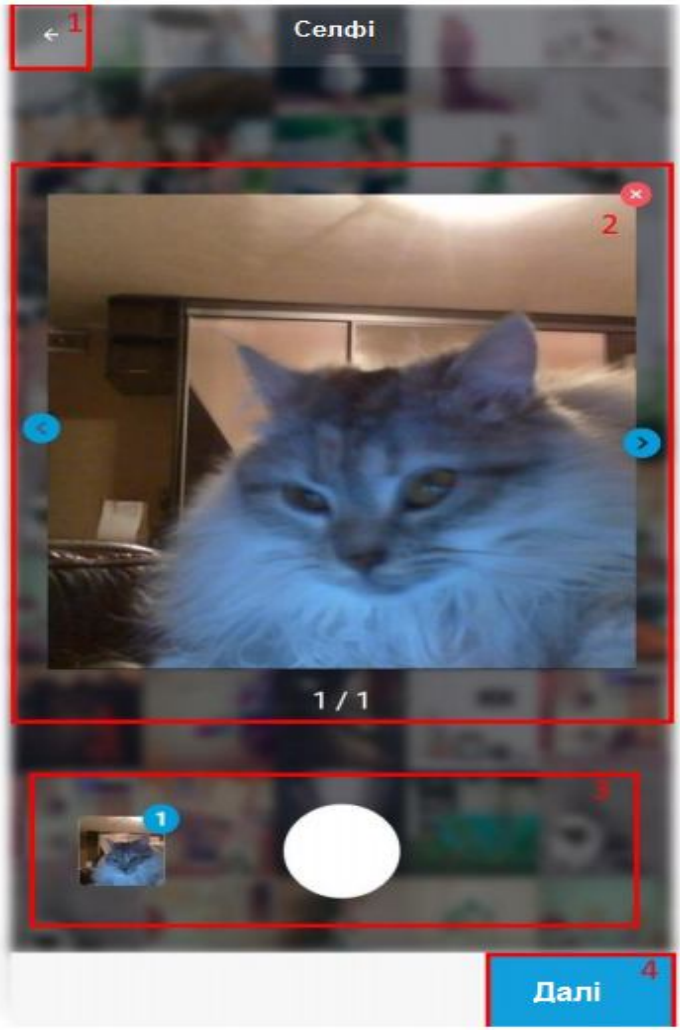

Рисунок 2.22 – Сторінка роботи з веб-камерою

Редагування вибраних зображень. З вибраних зображень для друку формуються фотографії з декількома редагованими полями. Перед друком користувач може переглянути і відредагувати сформовані фотографії (рисунок 2.23, 2.24).

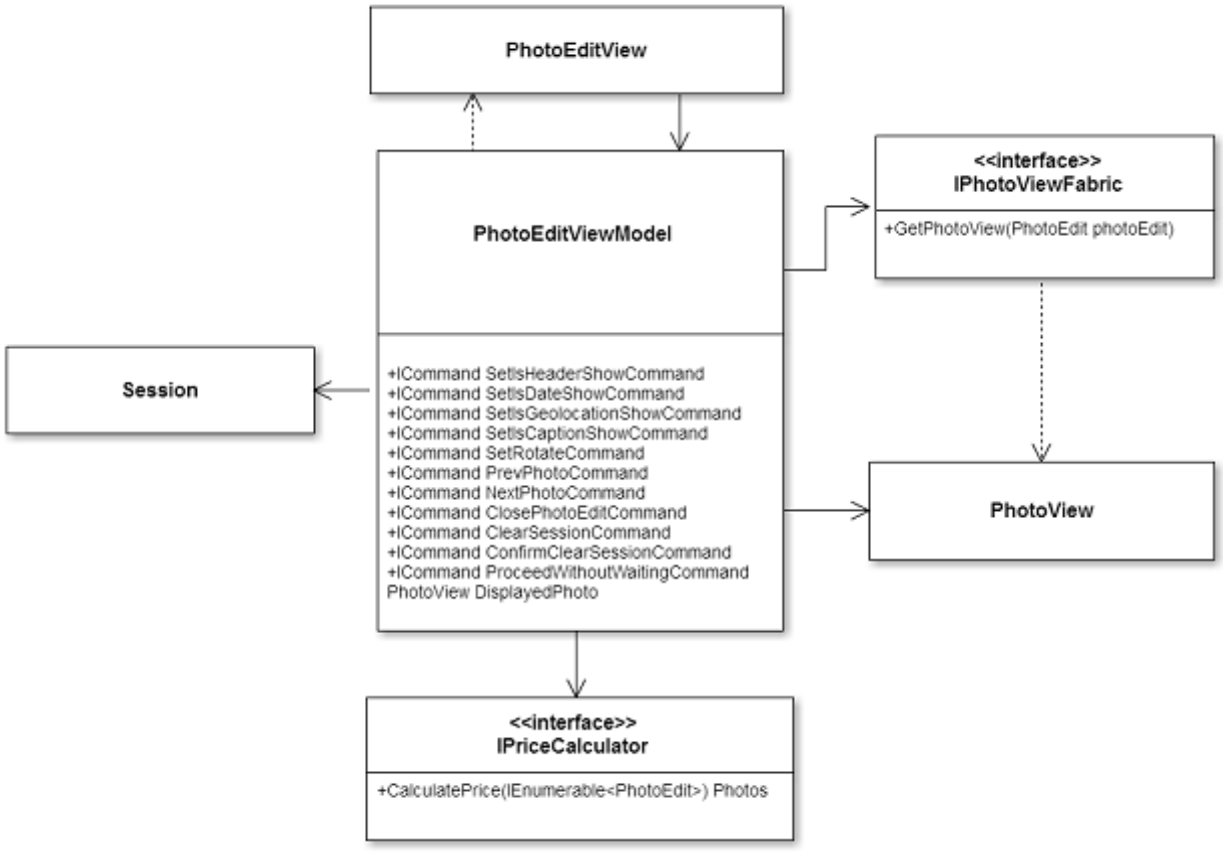

Рисунок 2.23 – Ієрархія класів екрану редагування фотографій

PhotoEditView - екран редагування фотографій.

- PhotoEditViewModel - модель представлення екрану редагування фотографій, здійснює процес редагування і отримання фотографій через IPhotoViewFabric.

- IPriceCalculator - оголошує інтерфейс для розрахунку ціни поточних фотографій.

- PhotoView - візуальний елемент, який представляє собою фотографію.

- Session - репозиторій фотографій.

- IPhotoViewFabric - оголошує інтерфейс для створення PhotoView, інкапсулює логіку створення елемента.

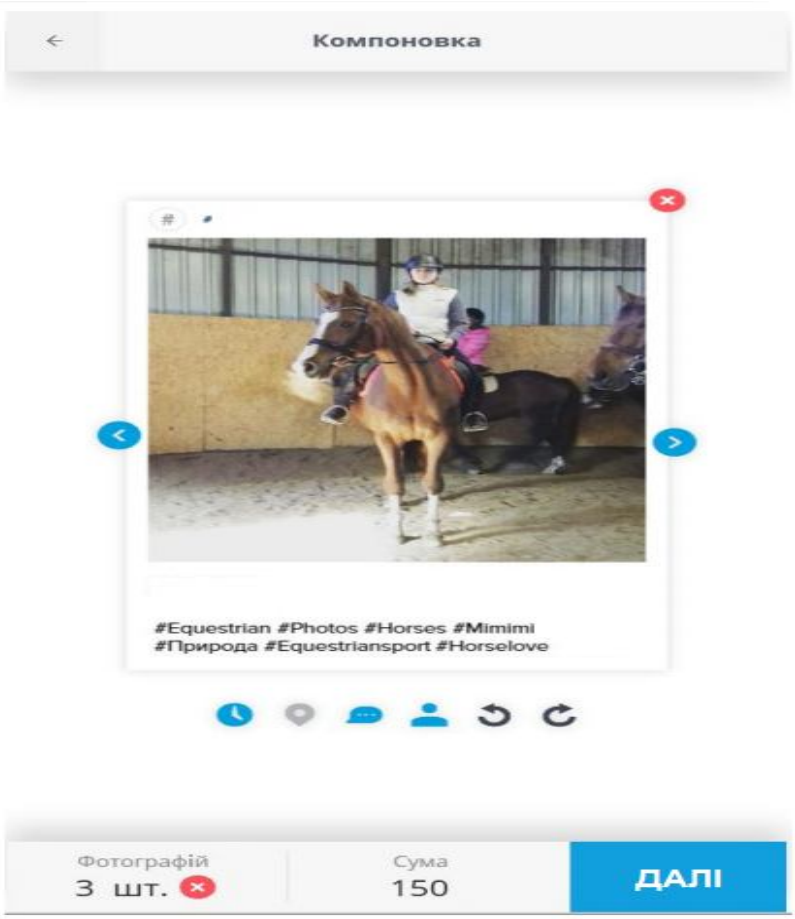

Рисунок 2.24 – Сторінка редагування фотографій

Розрахунок ціни фотографій. Перед оплатою програма повинна розрахувати ціну обраних фотографій, при цьому потрібно враховувати різні ціни різних форматів і спеціальні пропозиції (Offer) (рисунок 2.25).

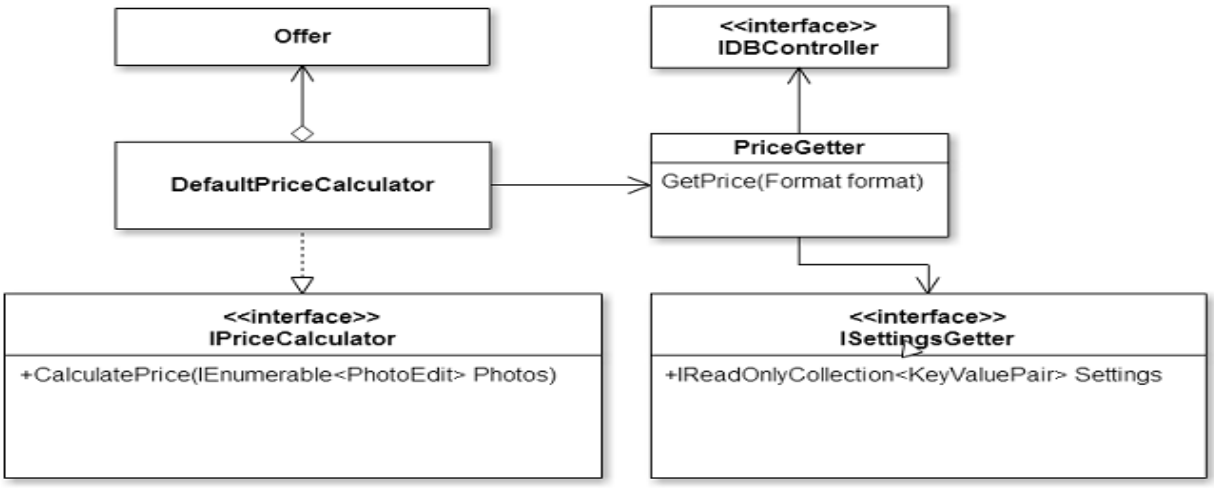

Рисунок 2.25 – Діаграма класів для розрахунку ціни вибраних фотографій

- IPricCalculator - оголошує інтерфейс для розрахунку цін.

- DefaultPriceCalculator - реалізація IPriceCalculator.

- PriceGetter - клас для отримання актуальної ціни фотографій, для отримання ціни намагається звернутися до бази даних, що містить ціни, якщо база даних недоступна отримує ціни з установок програми.

- ISettingsGetter - інтерфейс для отримання установок програми.

- IDBController - інтерфейс для комунікації з базою даних.

- Offer – об’єкт інкапсулює логіку для отримання знижки, знижка може залежати від кількості фотографій, формату, значень полів фотографії.

Оплата фотографій. Оплата вибраних фотографій здійснюється через пристрій прийому готівки - купюроприймач. В якості такого купюроприймача використовується ICT V7. Комунікація відбувається по протоколу ICT (рисунок 2.26).

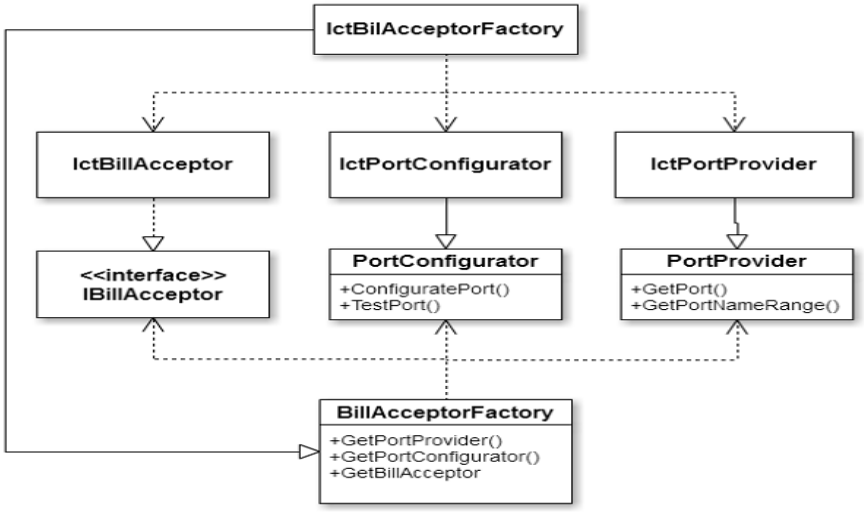

Рисунок 2.26 – Діаграма класів для роботи з купюроприймачем

BillAcceptorFactory - фабрика для створення класів для роботи з купюроприймачем.

- IctBillAcceptorFactory - фабрика для створення класів для роботи з ICT V7.

- IBillAcceptor – клас, який здійснює комунікацію з купюроприймачем.

- PortProvider - клас для створення і ініціалізації об'єкта, представляє собою інкапсуляцію логіки взаємодії з фізичним пристроєм - портом.

- PortConfigurator - клас для настройки і тестування об’єкта порту.

Вищеописані класи являють собою фізичний аспект оплати, після реєстрації купюроприймач купюри необхідно зареєструвати і перетворити дані про неї в потрібну форму і сповістити зацікавлені в її реєстрації сторони (рисунок 2.27).

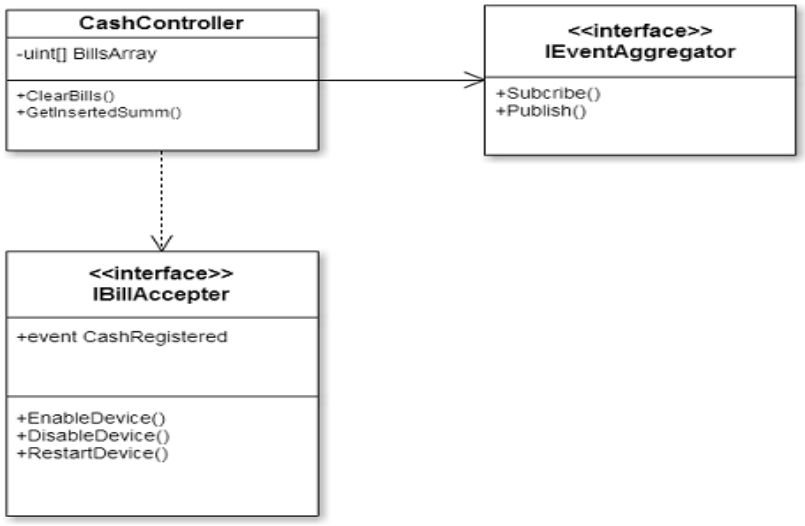

Рисунок 2.27 – Діаграма класів для реєстрації банкнот

IBillAccepter - клас для комунікації з купюроприймачем.

- CashController - клас, який здійснює перетворення і зберігання даних отриманих від купюроприймача про реєстрацію купюр, так само сповіщає про це інші частини програми.

- IEventAggregator - інтерфейс для підписки на події та публікації подій. Реалізовано через шаблон проектування "Спостерігач", є частиною фреймворка PRISM.

Після завантаження зображень, формування фотографій і оплати, необхідно їх роздрукувати (рисунок 2.28).

Для взаємодії з пристроєм друку були використані стандартні класи бібліотеки .Net з пакета System.Printing, дані класи являють собою обгортку над архітектурою друку Windows і класи SDK для роботи з принтером, надані компанією Mitsubishi Electronics.

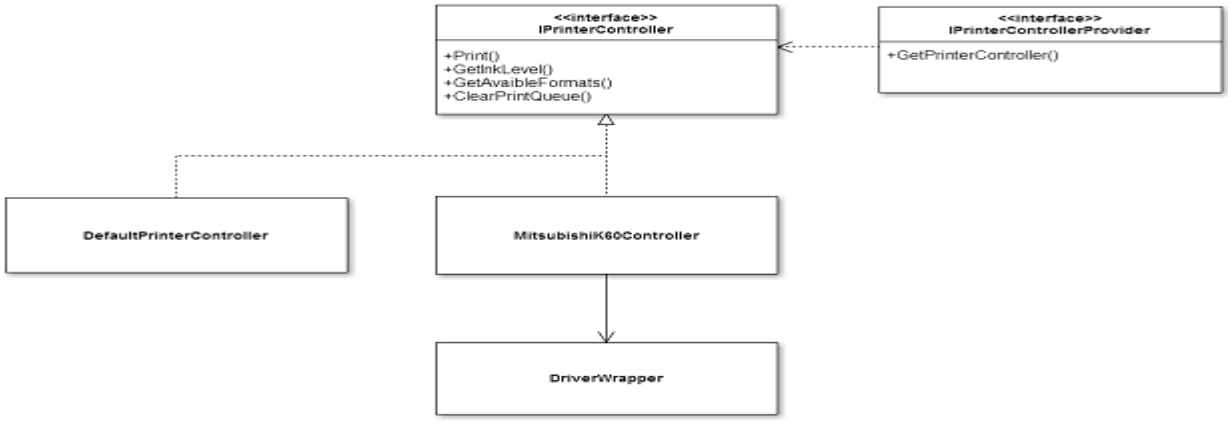

Рисунок 2.28 – Діаграма класів пакета Printer

IPrinterController - інтерфейс, що визначає поведінку класів для роботи з друком.

- DefaultPrinterController - універсальний клас для друку з будь-якого пристрою, але не на всіх пристроях клас має передбачувану поведінку через некоректні взаємодії драйверів деяких пристроїв і архітектури друку Windows.

- MitsubishiK60PrinterController - клас для роботи з певним пристроєм друку - Mitsubishi K60DW-S.

- MitsubishiK60DriverWrapper - обгортка для SDK принтера у вигляді DLL з некерованого коду, написаного на C++. За допомогою пакета System.InterlopSevices була забезпечена сумісність з C# кодом. Друк фотографій здійснюється на екрані оплати (рисунок 2.29, 2.30).

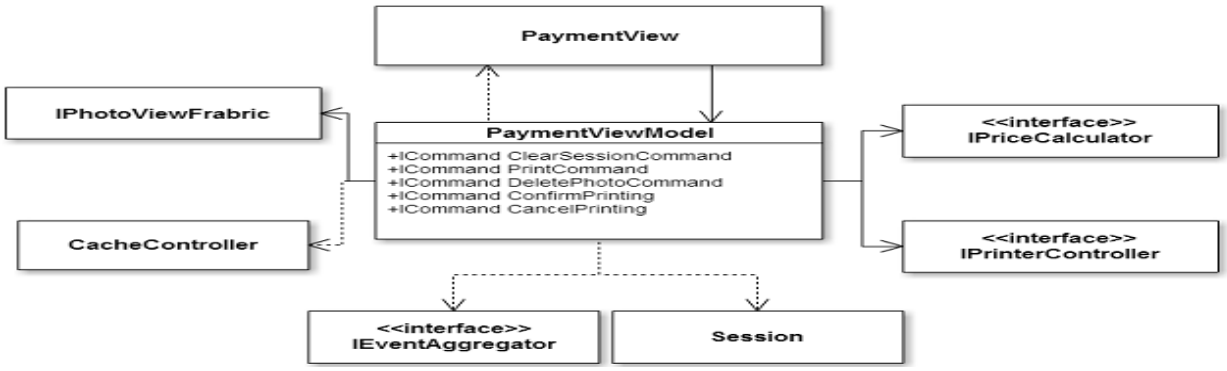

Рисунок 2.29 – Діаграма класів екрана оплати

PaymentView - екран оплати.

- PaymetViewMode - модель представлення екрана оплати, отримує дані по обраних фотографіях і формує з них фотографії. Після успішної оплати посилає сформовані фотографії на друк.

- CacheController - містить дані про внесені гроші.

- IPhotoViewFabric - фабрика для створення фотографій.

- IEventAggregator - дозволяє підписатися на події про внесення грошей.

- Session - дозволяє отримати інформацію про обрані фотографії.

- IPrinterController - містить метод для друку сформованих фотографій.

- IPriceCalculator - дозволяє порахувати ціну обраних фотографій.

Рисунок 2.30 – Екран оплати

Сервер був реалізований до розробки програми для моніторингу стану різних пристроїв, що представляють собою вендингові апарати.

Сервер для взаємодії використовує інтерфейс, розроблений за принципом, описаним вище - REST (рисунок 2.31).

Сервер використовує такі команди:

- 200 - "Порожній" відгук клієнта, вказує серверу що автомат працює;

- 210 (id, money, date, time) - внесення грошової суми.

- 211 (id, date, time) - виготовлення сувеніра.

- 420 (id, date, time) - несправність терміналу.

- 421 (id, date, time) - помилка підрахунку суми.

- 422 (id, date, time) - застрягання купюри.

- 423 (id, date, time) - видалення купюри.

- 424 (id, date, time) - укладальник відкритий.

- 425 (id, date, time) - помилка сенсора.

- 427 (id, date, time) - спроба злому купюроприймача.

- 428 (id, date, time) - помилка укладальника.

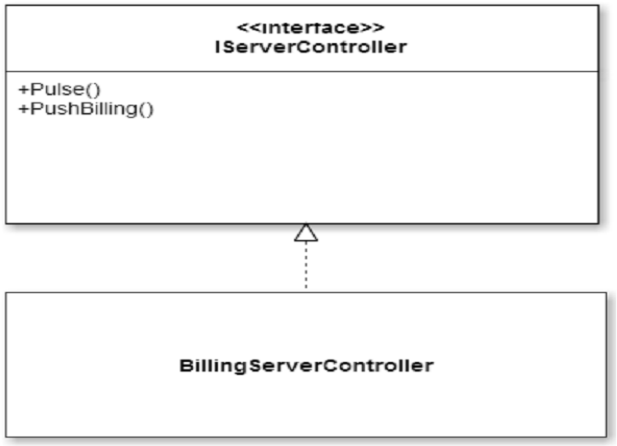

Рисунок 2.31 – Діаграма класів пакету Server

Параметри запитів:

- id - унікальний ідентифікатор терміналу.

- date - дата в форматі dd.MM.YY.

- time - час в форматі hh: mm: ss.

- money - грошова сума, отримана автоматом.

Як відповідь сервер повертає: 200 - запит пройшов без помилокабо 400 - помилка.

- IServerController - інтерфейс, який містить методи длявзаємодії з сервером.

- DefaultServerController - реалізація інтерфейсу, що дозволяєвзаємодіяти з сервером моніторингу станів.

- BillingInserter - клас, що інкапсулює взаємодію з RESTAPI сервера.

2.3. Тестування системи

Етап тестування є одним із найважливіших етапів у розробці програмного забезпечення, адже помилки в роботі програми можуть призвести до великих матеріальних витрат. За способом виконання тестів розрізняють автоматизоване та ручне тестування. Основна відмінність між цим підходами полягає у самому способі тестування програмного продукту: при автоматизованому підході використовуються спеціальні програмні продукти типу Silenium та Ranorex. Написання автоматизованих тестів є клопіткою справою та потребує багато часу та грошей, адже потрібно найти спеціалістів із правильними навичками, або самому потратити час на дослідження даної області.

На відміну від автоматизованого тестування, ручне тестування скоротить суттєво витрати на етапі тестування і зекономить час. Проте в майбутньому при розширенні програмного продукту ми стикнемось із проблемою повторного тестування, що призведе то повторного виконання тієї ж роботи, яка при автоматизованому тестуванні займе декілька хвилин. Але при кардинально сильних змінах представлення програми або функціонал буде потребувати супровід тих тестів.

На сьогодні тестування програмного забезпечення – один з найдорожчих етапів життєвого циклу програмного забезпечення, на нього відводиться від 45% до 55% загальних витрат [5]. Зазвичай, для проведення тестування застосовуються методи структурного («білий ящик») та функціонального («чорний ящик») тестування [7]. Можна виділити наступні методи тестування:- модульне тестування;- функціональне тестування;- тестування графічного інтерфейсу;тестування безпеки.

Першим етапом тестування було модульне. Суть модульного тестування полягає в окремому тестуванні кожного модуля коду програми. Для проведення модульного тестування було використано програмний продукт Microsoft Visual Studio, в якому було створено тестовий проект консольного типу, до якого підключено усі модулі та вручну було проведено тестування, суть якого у написанні програмного коду та перевірки отриманого результату.

При функціональному тестуванні вихідний код програми недоступний. Суть полягає в перевірці відповідності роботи програми до її специфікації. Критерієм повноти слід вважати перебір всіх можливих вхідних даних, що практично здійснити надзвичайно важко.

Метою тестування веб-сервісу є пошук його слабких місць у безпеці та недоліків відображення. На даний момент є велика кількість браузерів, котрі мають різні функції та по-різному трактують вже існуючі елементи, тому слід перевірити правильність відображення програми на різних браузерах, тобто провести кросбраузерне тестування.

Після проведення функціонального тестування було виправлено ряд суттєвих дефектів, які впливали на функціональність системи в цілому.

У таблиці 2.1 наведено опис дефектів, що були виявлені під час функціонального тестування.

Таблиця 2.1

Дефекти, які виявлені під час функціонального тестування

Дефект	Опис дефекту	Вирішення дефекту
При введенні некоректних параметрів для друку фотографій, система опрацьовує ті дані і зберігає для подальшого друку	Дефект полягає у тому що система повинна проводити перевірку даних, щоб не призвести до пошкодження цілісності даних	Додавання перевірки даних перед збереженням.

Продовження таблиці 2.1

Користувачам видно функції адміністратора	Дефект полягає у тому що є проблеми при перевірці ролі користувача	Виправлення атрибуту перевірки адміністратора
Можна зареєструвати користувача із вже існуючим логіном	Дефект полягає у тому, що немає перевірки на існування вже існуючого користувача	Потрібно добавити перевірку при додаванні на існування користувача із такою інформацією

Також надзвичайно важливу роль в розробці таких систем відіграє безпека, адже при дефектах у безпеці можна понести великі матеріальні збитки. Тому даний етап тестування потребує певних навичок у галузі інформаційної безпеки, що дасть змогу виявити потенціальні загрози.

Для тестування безпеки ресурсу було використано програмний продукт Fiddler.

Для тестування безпеки ми модифікуємо протоколи, котрі приходять до користувача, обходячи систему перевірки на браузері (рисунок 2.32). Результат тестування програмної системи є позитивним.

Наступним етапом тестування була перевірка правильності відображення інтерфейсу користувача. Через існування великої кількості браузерів, це є досить важкою роботою, адже потрібно переглядати правильність відображення у браузерах за даними при формуванні вимог. Також деякі браузери не мають можливостей як інші, тому потрібно розробляти програмний продукт, котрий буде враховувати усі ці аспекти.

Під час тестування відображення було виявлено ряд дефектів представлених у таблиці 2.2.

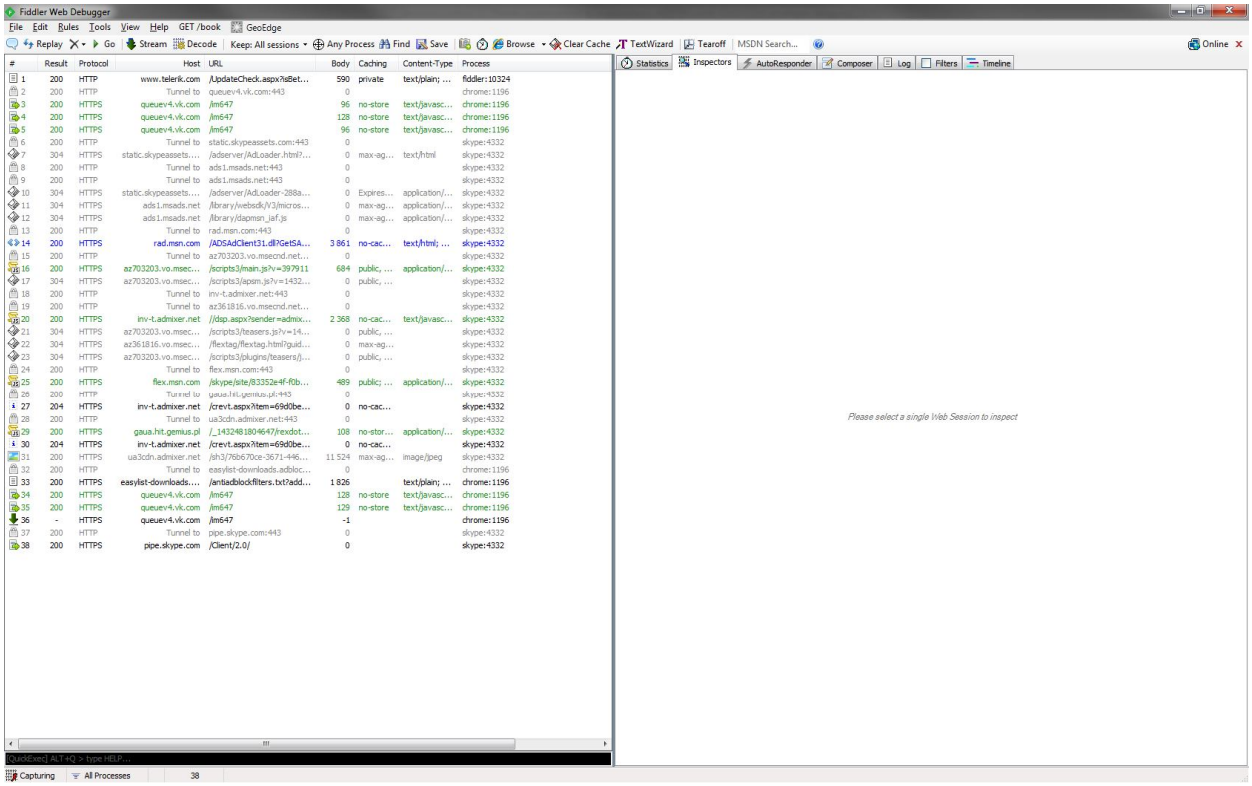

Рисунок 2.32 – Основне вікно роботи із Fiddler

Таблиця 2.2

Дефекти, які виявлені під час тестування інтерфейсу

Дефект	Опис дефекту	Вирішення дефекту
При відображені фотографії рамка верхнього контейнера налазить на фотографію	Дефект полягає у неправильному формуванні пріоритетів відображення	Потрібно видалити пріоритети відображення та встановити правильну стратегію відображення
При відображенні всього списку знайдених фотографій, нижні елементи налазять на кнопки	Дефект полягає у тому що відсутні обмеження на відображення.	Потрібно винести кнопки на вищий рівень ніж контент

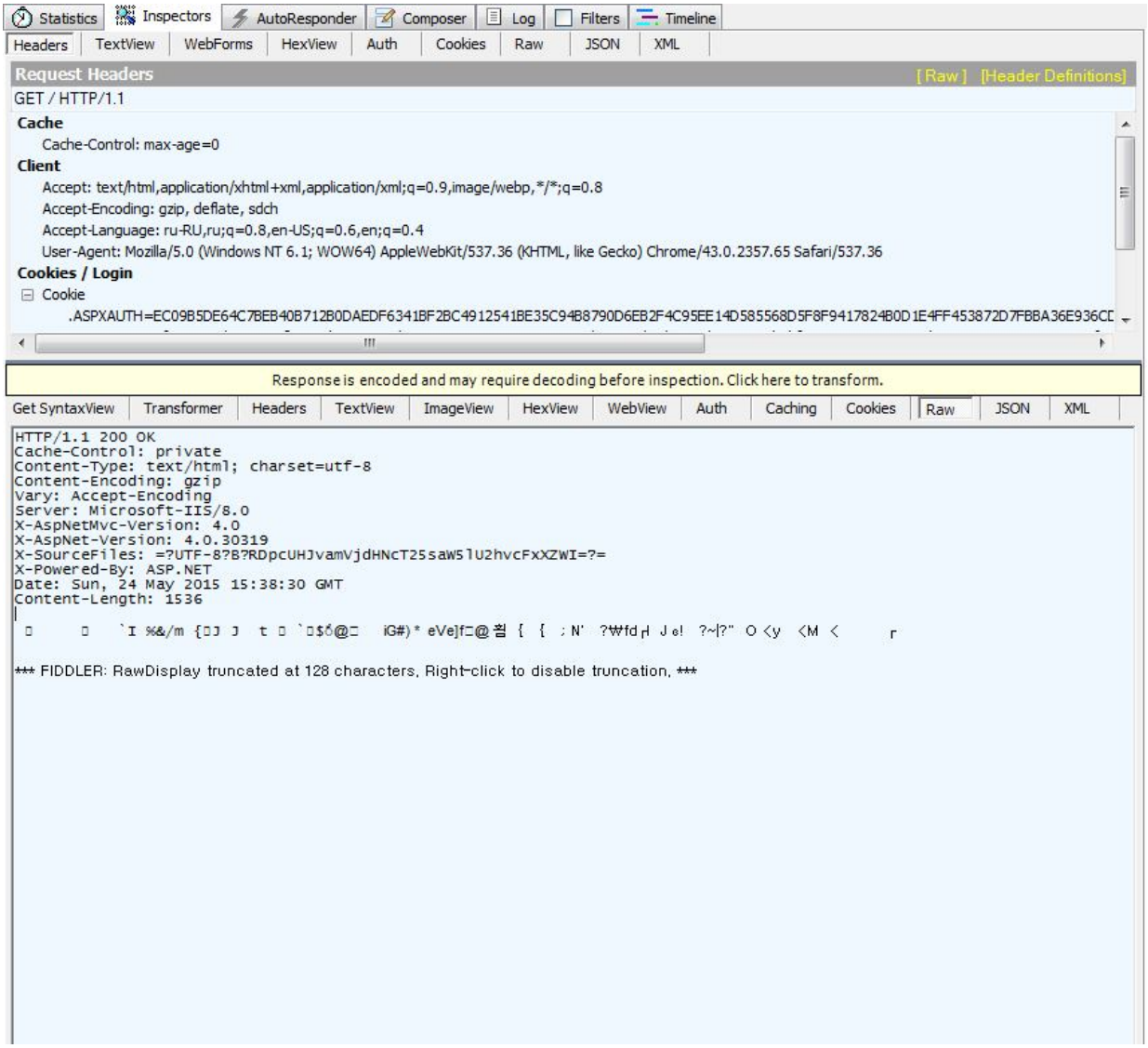

Рисунок 2.33. – Вікно модифікації протоколу.

Після проведення тестування інтерфейсу було виправлено ряд суттєвих дефектів, які впливали на відображення системи.

2.4. Особливості встановлення та налаштування продукту, інструкція користувача

Для розгортання програмної системи необхідно задовольнити вимоги, що наведені нижче:

- операційна система Windows Embedded 7;
- SQlite;

- web-сервер IIS 7/8;
- Microsoft .NET Framework 4.0/4.5.

Для формування апаратних вимого потрібно провести аналіз кількості користувачів, котрі будуть використовувати розроблену систему. Для невеликої кількості користувачів достатньо наступної конфігурації для апаратних ресурсів:

- процесор з частотою не менше 2 Ghz;
- оперативна пам’ять в межах 4 Gb;
- система потребує великого об’єму пам’яті на жорсткому диску для зберігання бази даних та зображень. Тому об’єм жорсткого диску має бути не менше 512 Gb;
- рекомендована швидкість з’єднання становить від 100 Mb/s.

Дана програмна система є клієнт-серверною, тому слід визначити вимоги до програмного забезпечення на стороні кінцевих користувачів. Так як програма є веб-орієнтованою, для її роботи потрібна наявність одного з наступного переліку

- Internet Explorer 9/10;
- Opera 16;
- Chrome 37.

Для розгортання системи на сервері, необхідно попередньо її збільшити та перенести усі модулі програмного продукту у відповідну папку на серверів. Після перенесення всіх елементів системи слід вказати, що ця папка є джерелом веб-сайту, клікнувши правою клавішею миші і вибравши функцію «Add web-site». Після встановлення папки як веб-сайт, появиться діалогове вікно, де потрібно встановити шлях до папки із контентом програми. Після проходження усіх попередніх кроків необхідно опублікувати веб-сайт за допомогою функції «Publish» в Microsoft Visual Studio.для публікації.

При запуску програми клієнт переходить на екран вибору джерела фотографії, що представлено на рисунку 2.34.

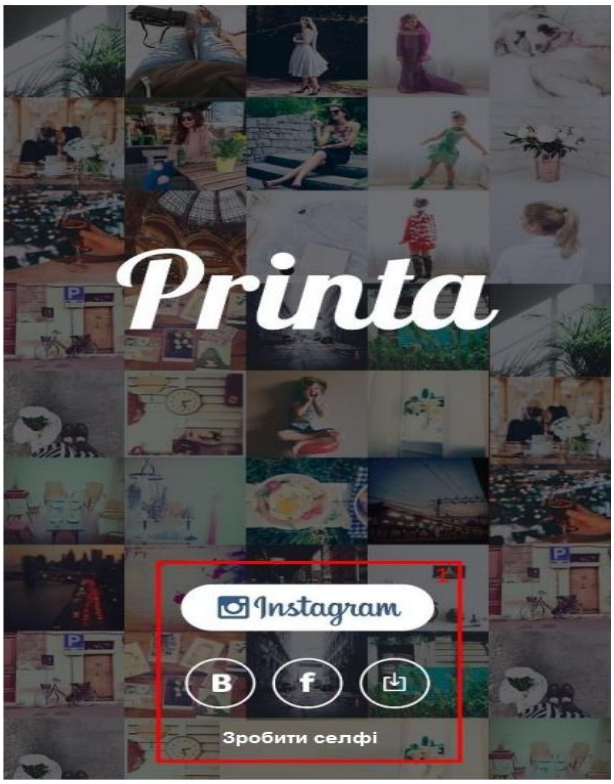

Рисунок 2.34. – Сторінка вибору джерела фотографії

На інтерфейсі екрану вибору джерела фотографій, показаного на рисунку 4.4, можна виділити ключову область, яка містить кнопки для переходу на екрани різних джерел фотографій. В якості джерел доступні:

- Instagram;
- Facebook;
- Завантаження фотографій з телефону;
- Отримання фотографій з вбудованою в апарат камери.

При виборі на екрані вибору джерела фотографій соціальної мережі, користувач переходить на екран пошуку фотографій в соціальних мережах, зображений на рисунку 4.4.

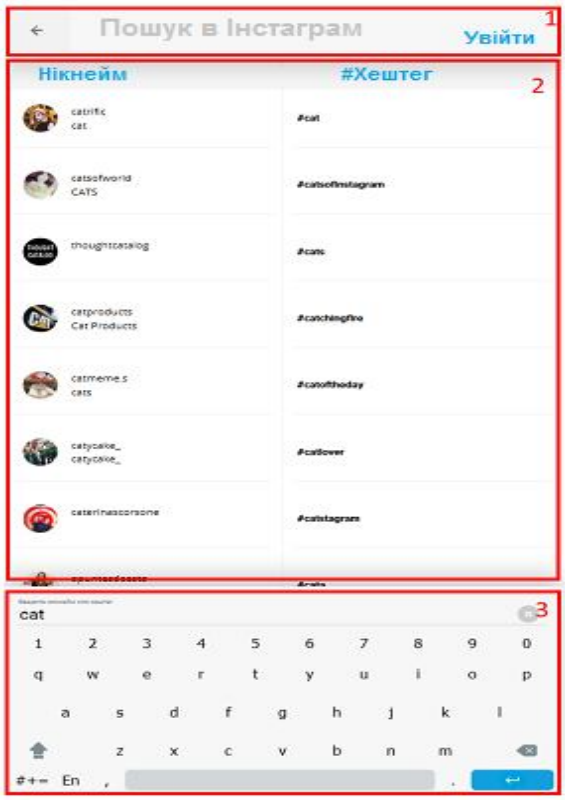

Рисунок 2.35. – Сторінка вибору джерела фотографії

В інтерфейсі екрану можна виділити кілька ключових областей:

1. Область навігації. Область містить кнопки для переходу на екран вибору джерела фотографії та авторизації в соціальній мережі.

2. Область результату пошуку. Область містить результати пошуку в соціальній мережі, розділені на дві групи: хештеги і користувачі, при натисканні на який-небудь результат відбувається перехід на екран вибору фотографій.

3. Область набору ключового слова. Область містить клавіатуру і область для відображення набраного ключа пошуку.

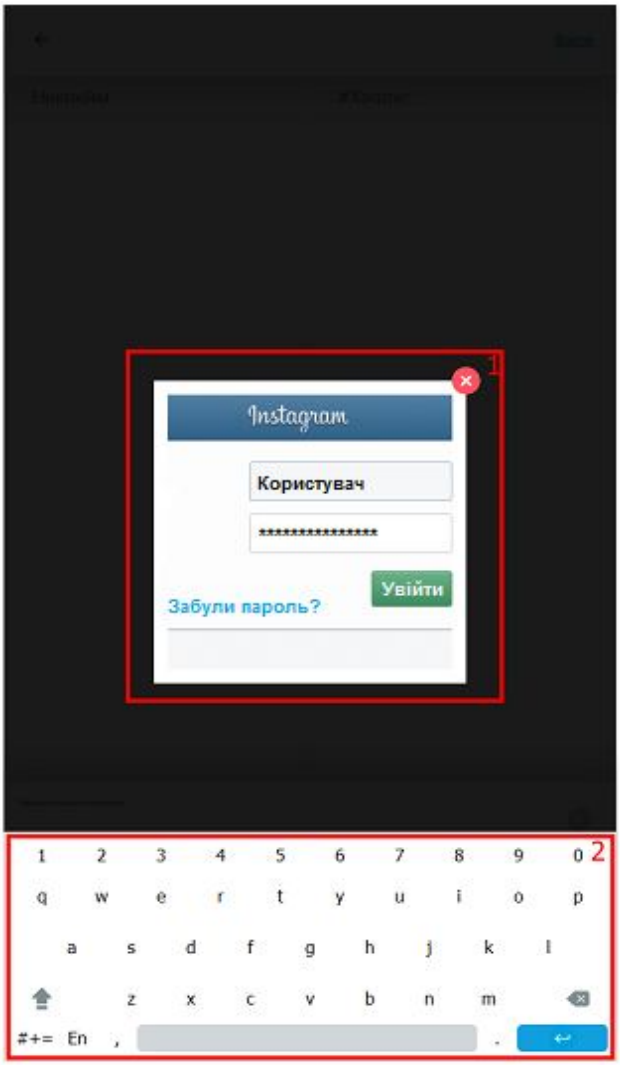

Рисунок 2.36. – Сторінка авторизації у соціальній мережі

При натисканні кнопки "Увійти" на екрані пошуку фотографії користувач переходить на екран авторизації в соціальній мережі, представлений на рисунку 2.36.

На екрані можна виділити наступні ключові області:

1. Область авторизації. Область містить вікно для авторизації в соціальної мережі, авторизація проходить аналогічно авторизації в браузері.

2. Область клавіатури. Містить клавіатуру. Після вибору результату пошуку в соціальній мережі, користувач переходить на екран вибору фотографій, показаний на рисунку 2.37.

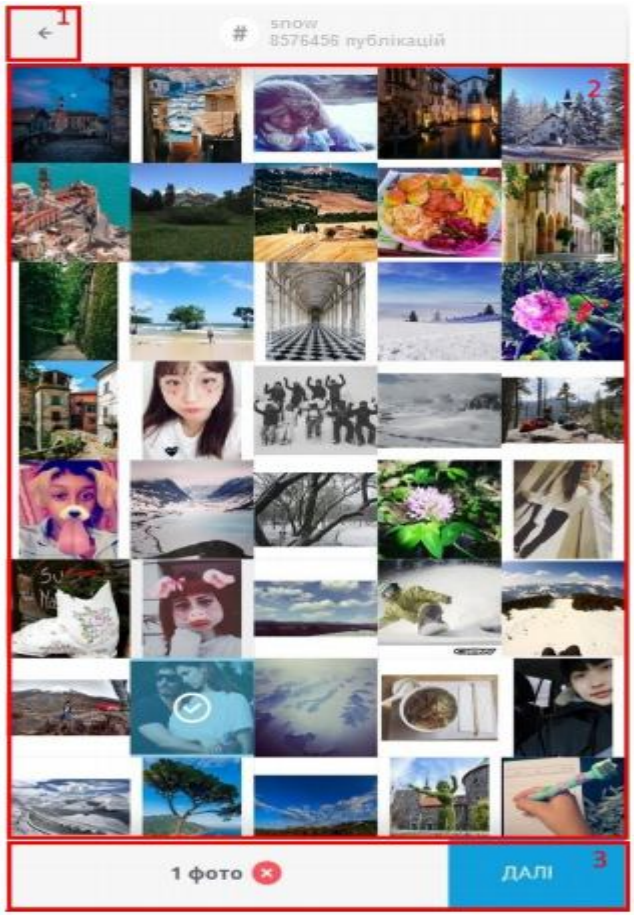

Рисунок 2.37. – Сторінка вибору фотографій

На екрані вибору фотографій можна виділити наступні ключові області:

1. Область навігації на попередній екран.

2. Область відображення фотографій з джерела. Містить завантажені фотографії, при натисканні на фотографію, фотографія виділяється і додається в колекцію фотографій на друк, при повторному натисканні на виділену фотографію виділення знімається і фотографія видаляється з колекції.

3. Область містить кількість вибраних фотографій, кнопку очищення колекції вибраних фотографій, при якій відбувається навігація на екран вибору джерела і кнопка для навігації на екран редагування фотографії.

При виборі на екрані джерела веб-камери, користувач переходить на екран роботи з веб-камерою, показаний на рисунку 2.38.

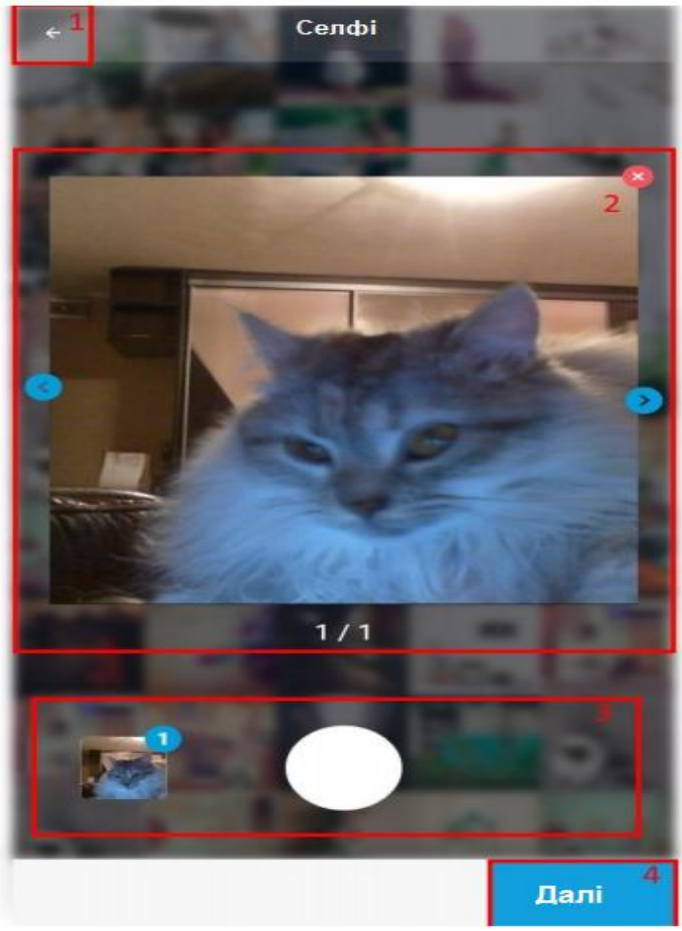

Рисунок 2.38 – Сторінка роботи з веб-камерою

На екрані можна виділити наступні ключові області:

1. Область навігації. Містить кнопку навігації на екран вибору джерела фотографії.

2. Область перегляду зображень з веб-камери, працює в двох режимах: режим відображення відео з веб-камери і перегляд зроблених фотографій.

3. Область управління. Містить кнопку, яка робить фотографію, якщо область перегляду зображень знаходиться в режимі відображення відео, після чого переходить в режим перегляду зроблених фотографій, якщо екран вже знаходиться в режимі перегляду, перемикає режим на режим відображення відео.

4. Область навігації на екран редагування фотографій. Якщо користувач на екрані вибору джерела вибирає в якості джерела телефон, він переходить на екран завантаження фотографій з телефонів, показаний на рисунку 2.39.

Рисунок 2.39. – Сторінка завантаження фотографій з телефону

На екрані можна виділити кілька основних областей:

1. Область навігації на екран навігації вибору джерела фотографії.

2. Область інструкцій. Містить інструкції для завантаження фотографій з телефонів.

3. Область навігації на екран вибору фотографій.

Після вибору фотографій з будь-яких джерел, перед друком користувач переходить на екран редагування фотографій, представлений на рисунку 2.40.

Рисунок 2.40. – Сторінка редагування фотографій

На екрані можна виділити наступні ключові області:

1. Область навігації на попередній екран.

2. Область перегляду фотографії, сформованої відповідно з заданими параметрами. Так само містить кнопки для видалення обраної фотографії та перемикання між обраними фотографіями.

3. Область редагування підпису до фотографії.

4. Область задання параметрів фотографії. Може задавати наявність або відсутність заголовка фотогр афії, місця розташування,підписи фотографії та дати. Так само можливий поворот фотографії на 90 градусів.

Після вибору фотографій і їх редагування користувач переходить на екран оплати і друку, показаний на рисунку 2.41.

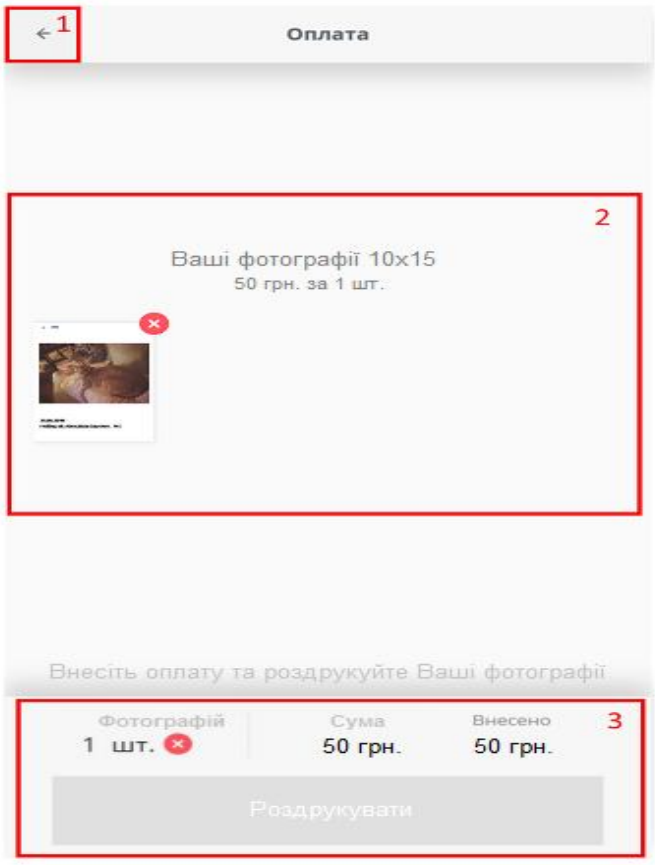

Рисунок 2.41. – Сторінка оплати та друку фото

На екрані можна виділити наступні ключові області:

1. Область навігації на екран редагування фотографій.

2. Область перегляду і видалення вибраних фотографій.

3. Область друку. Містить інформацію про необхідну суму і внесену суму, як тільки необхідна сума перестане перевищувати внесену, активується кнопка друку, при натисканні на яку відбувається друк. Після друку користувач знову переходить на екран вибору джерела фотографій.

2.5 Висновки до другого розділу

У даному розділі обґрунтовано технологію, мову програмування та розроблено програмну систему. Обґрунтовано засоби розробки бази даних та створено програмну реалізацію бази даних програмної системи за допомогою механізму збережених процедур.

Здійснено опис процедур тестування та їхніх результатів, описані тест-вимоги до програмного забезпечення, а також виявлені дефекти. Розкрито питання встановлення та налаштування програмного забезпечення на сервері, а також вказані вимоги, дотримання яких необхідно для користування системою, описана інструкція користувача для роботи із системою.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Методи підвищення мотивації безпеки праці

Потрібно знайти такі способи впливу на людей, щоб вони усвідомили необхідність працювати безпечно, створити такі «правила гри», в межах яких людині було б вигідно дотримуватися встановлених норм.

Безпечна поведінка на виробництві залежить не тільки від професійних знань, навичок і здібностей, а й значною мірою від мотивів поведінки працівника. Відповідно управляти діями людини можна тільки за допомогою управління її мотивами.

В обмін за свою працю працівники очікують не тільки високої оплати, а й створення умов для особистісного росту, отримання задоволення від власної роботи, інших компенсацій, які адекватні професійному рівню та відповідають особистим інтересам. Ефективна праця допомагає швидшому розвиванню підприємства.

Для заохочення працівників потрібно підбадьорення та підтримка з сторони начальства. Стимулювати ефективну роботу можуть матеріальні методи, наприклад премії, винагороди, безкоштовне харчування, додатковий дохід та інше. Мотиваційний комплекс взагалі й безпечної поведінки людини зокрема носить полімотивований характер, містить у собі широкий спектр мотиваційних регуляторів як матеріального, так і нематеріального характеру та має певну ієрархічність. На особистісному рівні працювати продуктивно та безпечно вигідно самій людині; від цього залежить успіх роботи її підрозділу; і нарешті - це потрібно підприємству (компанії). Тобто для вирішення охоронних проблем у праці потрібно зацікавити працівників трудитися безпечно не тільки для себе, а й для оточуючих.

Практично будь-якого працівника можна зацікавити будь-яким мотивом, оскільки абсолютно ні на що не мотивованих людей немає! Очевидно, тільки закликами, зверненнями, деклараціями, пропагандою ці проблеми навряд чи вдасться вирішити.

Потрібно знайти такі способи впливу на людей, щоб вони усвідомили необхідність працювати безпечно, створити такі «правила гри», в межах яких людині було б вигідно дотримуватися встановлених регламентів. І цей вплив вона повинна відчувати безпосередньо в процесі всієї трудової діяльності.

Проаналізувавши загальні методи мотивації для підвищення роботи працівників можемо охарактеризувати методи які потрібні для підвищення мотивації безпеки праці осіб на різних підприємствах. Найголовнішим чинником для будь-якого підприємства має бути на першому місці створення безпечних умов праці та дотримання всіх необхідних безпечних заходів для своїх працівників. Важливим показником охорони праці на підприємстві є внутрішнє стимулювання для безпечного ведення робіт. При можливості на об’єкті можуть працівники долучитись до охорони праці та запропонувати свої варіанти, підвівши підсумки можуть скласти договір. У цій угоді, яку склали колективно можуть вказати свої матеріальні та нематеріальні вимоги. При цьому не може бути системи стимулювання, яка мотивує всіх співробітників однаково.

Система стимулів має бути персоніфікованою, ретельно дозованою та розроблятися для кожної людини або певної групи людей з подібними домінуючими потребами, або загальна система має індивідуалізуватися. Тому моніторинг домінуючих потреб персоналу - необхідна умова функціонування.

3.2 Забезпечення захисту працівників суб’єкта господарювання від іонізуючих випромінювань

Іонізуюче випромінювання або радіоактивність є небезпечним явищем для людського організму. При взаємодії впливу іонізаційних випромінювань у

навколишнє середовище можуть відбутись різні утворення зарядів . Існують два різновиди випромінювання – «альфа» та «бета».

В залежності від носія та енергії, вони мають різну проникаючу здатність. Альфа це випромінювання яке проявляється важкими частинами складеними з протонів і нейтронів.

В свою чергу бета випромінювання являє собою ланцюг електронів та позитронів які є більшу здатність проникати у середовище. Працюючи на таких територіях, де існує радіаційна атмосфера можуть виникнути різні випадки.

На підприємстві можуть виникнути інциденти при користуванні ядерними матеріалами , зберіганні радіоактивних відходів в наслідок чого працівники можуть отримати травму у вигляді дози опромінення, використання іонізуючих джерел випромінювання.

Також у випадку такої радіаційної аварії забруднюється навколишнє середовище, люди можуть отримати травму у вигляді потужної дози опромінення. Призвести аварію на підприємстві може також якщо активна реакційна речовина знаходиться у роботі та це відбувається незаконно.

Це може привезти до опромінення жителів та перевищити межу дози опромінення. Частинки з цього випромінювання можуть залишати сліди на дихальній системі на травній системі людського організму. Також ці елементи можуть бути у водних каналах, які постачають питну воду людям.

На підприємстві де проводяться роботи з радіаційними речовинами обов’язково мають вживатись заходи проти радіації. Протирадіаційні захисти це така система правових, організаційних норм та санітарної гігієни.

До переліку таких захистів можна включити медичні заходи для забезпечення радіаційної безпеки персоналу та проектно-конструкторські. Для організації заходів проти іонізації опромінювання підприємство має ввести обов’язкові методи щоб подбати про безпеку працюючого персоналу. До таких методів можуть належати заходи які обмежують допуск працівників до джерел які випромінюють радіацію.

До таких працівників можемо віднести таких, які не підходять за віком, за статтю та працівники які вже отримали дозу випромінення. Підприємство мусить створити сприятливі умови що дотримуються встановлених норм та вимог для працівників та застосовувати індивідуальні засоби для захисту працівника цього підприємства.

Організація повинна контролювати рівні опромінювання та вести інформаційну систему про стан радіації на підприємстві та призначених місць для праці.

На підприємстві повинні бути проведені заходи щодо організації безпеки для робіт які проводяться у радіаційних ділянках а саме:

-організація роботи нарядів та розпоряджень;

-організація та перевірка пропусків до робочих місць;

-оформлення контролю за процесом виконання роботи;

-введення примусового часу на перерву та вчасне закінчення робочого процесу.

До фізичних норм захисту проти радіації існують перешкоди поширення іонізації опромінень. Для поширення дози випромінювання може бути ряд перешкод, залежать вони від кількості годин, перешкоджати може дистанція , також перешкодою може бути чисельність.

Реалізувати заходи проти радіації за певний відрізок часу можливо, тим що працівники , які працюють з іонізованими випромінюваннями можуть виконувати вчасно свою роботу ,відповідно керівництво може за якісну роботу зменшити кількість робочих днів у тижні.

Цим самим вони застереженням вони зменшать знаходження працівників у зоні випромінювання та відповідно буде менше контактування з радіаційними приладами. Захистити працівників за допомогою відстані підприємство може шляхом доцільного розміщення приміщення, правильно розставити та розрахувати робочі місця для працівників а також забезпечити приладами, які зможуть контактувати, керувати робочим процесом з технікою яка має радіаційний вплив на відстані.

Слугувати захистом може покриття свинцем меблів які присутні у приміщенні (двері, вікна, робочі столи), створення перекриття між поверхами та перегородки. Працівникам обов’язково має бути виданий спеціальний одяг ,такі як фартухи, шапочки та рукавиці зшиті з просвинцевої тканини.

Розміщення робочих місць повинно мати правильний розрахунок на загальну кімнату, не робити перенабір та забезпечити відповідним та необхідним обладнанням робочі кабінети. При користуванні відкритими приладами іонізованого опромінення провести герметизації цих систем, при можливості використовувати роботу техніки. Підприємство повинне вжити усіх санітарно-гігієнічних заходів та соціальних, а також важливо необхідний є медичний захист робочих на об’єкті.

3.3 Висновок до третього розділу

В даному розділі описано заходи та методи із забезпечення радіаційних впливів та іонізації опромінювання на підприємствах. Описані вимоги для керівництва та підлеглих працюючих на об’єктах щодо їхніх дій в разі виникнення радіації .

Також описані вимоги для мотивації робітників щодо дотримання правил охорони праці на підприємстві. Мотивація - одна з центральних функцій управління як персоналом, так і охороною праці. Вона може відігравати важливу роль як фактор спонукання персоналу діяти адекватним способом у власних і корпоративних інтересах.

Для цього потрібно, щоб мета підприємства збігалася з метою працівників. Однак мотивація одночасно є не тільки рушійним механізмом, а й фактором залучення, наприклад, до охорони праці, високопрофесійних спеціалістів. Це механізм, що спонукає вдосконалювати систему управління. Крім того, рівень мотивації працівників відіграє важливу роль у загальному успіху підприємства (компанії).

ВИСНОВКИ

В ході виконання бакалаврської роботи, були проаналізовані вимоги, отримані від замовника, виявлені ролі акторів і варіанти використання для відповідних ролей.

Був обраний і вивчений інструментарій: бібліотека створення призначеного для користувача інтерфейсу. Також були вивчені API соціальних мереж (Instagram, Facebook) і робота обладнання, необхідного для реалізації поставленої мети.

В результаті проаналізованих вимог були використані навички об’єктно-орієнтованого проектування та сформовані діаграми класів і пакетів, що описують систему.

На основі артефактів, отриманих на етапі проектування, за допомогою вивчених інструментів було реалізовано і впроваджено програмне забезпечення для скачування, оплати і друку зображень з соціальної мережі.

Надалі в систему планується додати такі можливості:

- коди здачі.

- можливість вибору способу оплати.

Розширені можливості отримання фотографії через веб-камеру: фотографія через хмари, технології суміщення двох і більше зображень чи кадрів в одній композиції для суміщення фотографії та деякого заздалегідь заготовленого зображення, розпізнавання осіб на фотографії.

ПЕРЕЛІК ДЖЕРЕЛ

1. Sotnik, S. Developing the information search system for selecting the moulds forming elements / S. Sotnik, V. Nevliudova, I. Malaya // Innovative technologies and scientific solutions for industries (ITSSI). – 2(2). – 2017. – p. 86–92.

2. Al-Sherrawi, Mohannad H. Information model of plastic products formation process duration by injection molding method / Al-Sherrawi, Mohannad H. Saadoon, Ali Malik Sotnik, S. Lyashenko, V. // International Journal of Mechanical Engineering and Technology. – 2018. – Vol. 9(3). – P. 357–366.

3. Lyashenko, V. Recognition of Voice Commands Based on Neural Network / V. Lyashenko, F. Laariedh, S. Sotnik, M. Ayaz Ahmad // TEM Journal. – 2021. – Volume 10. – Issue 2. – P. 583 ‑ 591.

4. Abu-Jassar, A.T. Some Features of Classifiers Implementation for Object Recognition in Specialized Computer systems / A.T. Abu-Jassar, Y.M. Al-Sharo, V. Lyashenko, S. Sotnik // TEM Journalthis link is disabled. – 2021. – Vol. 10. – Issue 4. – P. 1645–1654.

5. Deineko, Zh. Confidentiality of Information when Using QR-Coding / Zh. Deineko, S. Sotnik, V. Lyashenko // International Journal of Academic Information Systems Research (IJAISR).– 2022. – Vol. 6. – Issue 9. – P. 10-15.

6. Al-Sharo, Y. M. et al. Neural Networks As A Tool For Pattern Recognition of Fasteners / Y.M. Al-Sharo, A.T. Abu-Jassar, S. Sotnik, V. Lyashenko // studies. – 2021. – Т. 4. – №. 11. – С. 13.

7. Scholz, S.S. Contemporary scientometric analyses using a novel web application: the science performance evaluation (SciPE) approach / S.S. Scholz, M. Dillmann, A. Flohr, C. Backes et al.// Clinical Research in Cardiology. – 2020. – Т. 109. – №. 7. – С. 810-818.

8. Fredj, O.B. An OWASP top ten driven survey on web application protection methods / O.B. Fredj, O. Cheikhrouhou et al. // International Conference on Risks and Security of Internet and Systems. – Springer, Cham, 2020. – С. 235-252.

9. Mobile operating systems' market share worldwide from January 2012 to October 2020. – Електрон. дан. – Режим доступу: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operatingsystems-since-2009/

10. Конспект лекцій з дисципліни «Програмування для мобільних пристроїв» для студентів денної форми навчання спеціальності 126 «Інформаційні системи та технології» / Укладачі: Готович В.А., Михайлович Т.В. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2020. – 216 с.

11. Офіційний сайт компанії Apple. – Електрон. дан. – Режим доступу: https://www.apple.com/ru/ios/what-is/

12. App Store. – Електрон. дан. – Режим доступу: https:// developer.apple.com/support/appstore/

13. iOS Human Interface. – Електрон. дан. – Режим доступу: https://developer.apple.com/library/ios/documentation/userexperience/conceptual/Mobile HIG/index.html

14. Офіційний сайт SQLite. – Електрон. дан. – Режим доступу: http://www.sqlite.org

15. iOS Technology Overview. – Електрон. дан. – Режим доступу: https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneos techoverview/Introduction/Introduction.html

16. R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. Hypertext Transfer Protocol -- HTTP/1. – The Internet Society, 1999. – 114 с.

17. Коноваленко І. В. Платформа .NET та мова програмування C# 8.0 : навчальний посібник / І. В. Коноваленко, П. О. Марущак. – Тернопіль : ФОП Паляниця В. А., 2020. – 320 с.

18. Комп’ютерні мережі: [навчальний посібник] / А. Г. Микитишин, М. М. Митник, П. Д. Стухляк, В. В. Пасічник. — Львів: «Магнолія 2006», 2013. — 256 с. ISBN 978-617-574-087-3

19. Петрик М.Р. Моделювання програмного забезпечення : науково методичний посібник / М.Р. Петрик, О.Ю. Петрик – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2015. – 200 с.

20. Філімончук Т.В., Чепелєв Є.О. Модель мобільного застосунку з використанням фреймворку Flutter. Проблеми інформатизації: Тези доповідей дев’ятої міжнародної науково-технічної конференції. Том 2, секція 4. Черкаси – Харків – Баку – Бельсько-Бяла. 2021. С. 99.

21. App Store Resource Center. – Електрон. дан. – Режим доступу: https://developer.apple.com/appstore/index.html

22. Google Maps SDK for iOS. – Електрон. дан. – Режим доступу: https://developers.google.com/maps/documentation/ios/?hl=ru

23. Map Kit Framework Reference. – Електрон. дан. – Режим доступу: https://developer.apple.com/library/ios/documentation/MapKit/Reference/MapKit_Fr ame work_Reference/_index.html

24. Cocoa Application Competencies for iOS. – Електрон. дан. – Режим доступу: https://developer.apple.com/library/ios/documentation/general/conceptual/DevpediaC ocoaApp/Storyboard.html

25. Instruments User Guide. – Електрон. дан. – Режим доступу: https://developer.apple.com/library/mac/documentation/developertools/conceptual/ instrumentsuserguide/Introduction/Introduction.html

26. React: Up & Running, 2nd Edition / S. Stefanov - O'Reilly, 2021 – 222c.

27. Components and Props [Електронний ресурс] – Режим доступу: https://reactjs.org/docs/components-and-props.html/.

28. React in Action, 1st Edition / M. T. Thomas – Manning Shelter Island, 2019 – 366c.

29. Electron vs PWA: The Pros And Cons Of Both Approaches [Електронний ресурс] – Режим доступу: https://javascript.plainenglish.io/electron-vspwa-the-pros-and-cons-of-both-approaches-b4ce172f0022.

30. Electron in Action, 1st Edition / S. Kinney - Manning Publications, 2018 – 376c.

31. What Is i18n? [Електронний ресурс] – Режим доступу: https://lingoport.com/what-is-i18n/.

ДОДАТКИ

ДОДАТОК А

ЛІСТИНГ ВСТАНОВЛЕННЯ ЗВ'ЯЗКУ З SQLITE

```
private SQLiteConnection sql_con;
private SQLiteCommand sql_cmd;
private SQLiteDataAdapter DB;
private DataSet DS = new DataSet();
private DataTable DT = new DataTable();
private void SetConnection()
        {
                sql_con = new SQLiteConnection
                ("Data Source=DemoT.db;Version=3;New=False;Compress=True;");
        }
 private void ExecuteQuery(string txtQuery)
        {
        SetConnection();
        sql_con.Open();
        sql_cmd = sql_con.CreateCommand();
        sql_cmd.CommandText=txtQuery;
        sql_cmd.ExecuteNonQuery();
        sql_con.Close();
        }
private void LoadData()
        {
        SetConnection();
        sql_con.Open();
        sql_cmd = sql_con.CreateCommand();
        string CommandText = "select id, desc from foto";
        DB = new SQLiteDataAdapter(CommandText,sql_con);
        DS.Reset();
        DB.Fill(DS);
        DT= DS.Tables[0];
        Grid.DataSource = DT;
```

```
            sql_con.Close();
            }
private void Add()
            {
string txtSQLQuery = "insert into  foto (desc) values ('"+txtDesc.Text+"')";
            ExecuteQuery(txtSQLQuery);
}
```

ДОДАТОК Б

ЛІСТИНГ ПРОЦЕДУРИ НАЛАШТУВАННЯ СИСТЕМИ ТА ОБЛАДНАННЯ

```
using System;
using System.IO.Ports;
using System.Threading;
using System.Windows.Forms; // temp use during debug

   public class HyperTerminalAdapter {

   SerialPort oSerialPort = new SerialPort();

   // Allow the user to set the appropriate properties.
   public int BaudRate = 9600;
   public int DataBits = 8;
   public int ReadTimeout = 500;
   public int WriteTimeout = 500;
   public string PortName = "COM4";
   public string Handshake = "";
   public string Name = "user";
   public string DataReceived = "";
   public string sParity = "none";
   public int iStopBits = 1;



   public HyperTerminalAdapter() {
      this.Configure();
   }

   public void Configure() {
      oSerialPort.PortName = this.PortName;
      oSerialPort.BaudRate = this.BaudRate;
```

```
    oSerialPort.DataBits = this.DataBits;
    oSerialPort.ReadTimeout = this.ReadTimeout;
    oSerialPort.WriteTimeout = this.WriteTimeout;

    oSerialPort.Handshake = System.IO.Ports.Handshake.None;

    if(this.sParity == "even"){
       oSerialPort.Parity = Parity.Even;
    }else if(this.sParity == "odd"){
       oSerialPort.Parity = Parity.Odd;
    }else if(this.sParity == "mark"){
       oSerialPort.Parity = Parity.Mark;
    }else if(this.sParity == "space"){
       oSerialPort.Parity = Parity.Space;
    } else {
       oSerialPort.Parity = Parity.None;
    }

    if(this.iStopBits == ){
       oSerialPort.StopBits = StopBits.None;
    }else if(this.iStopBits == .){
       oSerialPort.StopBits = StopBits.OnePointFive;
    }else if(this.iStopBits == ){
       oSerialPort.StopBits = StopBits.Two;
    } else {
       oSerialPort.StopBits = StopBits.One;
    }

    MessageBox.Show("Configured");
}

public void Connect() {
    try {
       if (!oSerialPort.IsOpen) {
          oSerialPort.Open();
```

```
            MessageBox.Show("Connected");
        } // else already open
    } catch (Exception e) {
            MessageBox.Show("Error: Connection is in use or is not available: \n\n" + e);
    }
}

public void Disconnect() {
    try{
        if (oSerialPort.IsOpen) {
            oSerialPort.Close();
            MessageBox.Show("Disconnected");
        } //else not open
    } catch { }
}

public void Write(string sData /* string data to write to the port */ ) {
    if (oSerialPort.IsOpen) {
        try {
            oSerialPort.WriteLine(sData);
            MessageBox.Show("Command Issued: " + sData);
        } catch { }
    }
}

public string Read() {
    try {
        this.DataReceived = oSerialPort.ReadLine().ToString();
        MessageBox.Show(this.DataReceived);
        return (this.DataReceived);
    } catch {
        return "";
    }
}
```

```
        } // exit class

HyperTerminalAdapter oHyperTerminalAdapter = new HyperTerminalAdapter();
        oHyperTerminalAdapter.Connect();
        oHyperTerminalAdapter.Write("AT");
  string x = oHyperTerminalAdapter.Read();
        oHyperTerminalAdapter.Disconnect();
        MessageBox.Show("Result of the command was: " + x);
```

ДОДАТОК В

ЛІСТИНГ ОСНОВНИХ МОДУЛІВ СИСТЕМИ

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Media;
using System.Windows.Shapes;
using System.Windows.Media.Imaging;

namespace SDKSamples.ImageSample
{

   public partial class PhotoView : Window
   {
      Photo _photo;

      public PhotoView()
      {
         InitializeComponent();
      }

      public Photo SelectedPhoto
      {
         get { return _photo; }
         set { _photo = value; }
      }

      private void WindowLoaded(object sender, RoutedEventArgs e)
      {
         ViewedPhoto.Source = _photo.Image;
         ViewedCaption.Content = _photo.Source;
      }

      private void Rotate(object sender, RoutedEventArgs e)
      {

         BitmapSource img = (BitmapSource)(_photo.Image);

         CachedBitmap cache = new CachedBitmap(img, BitmapCreateOptions.None,
BitmapCacheOption.OnLoad);
         _photo.Image = BitmapFrame.Create(new TransformedBitmap(cache, new
RotateTransform(90.0)));

         ViewedPhoto.Source = _photo.Image;
      }
```

```
        private void Crop(object sender, RoutedEventArgs e)
        {
            BitmapSource img = (BitmapSource)(_photo.Image);

            int halfWidth = img.PixelWidth / 2;
            int halfHeight = img.PixelHeight / 2;
            _photo.Image = BitmapFrame.Create(new CroppedBitmap(img, new Int32Rect((halfWidth -
(halfWidth / 2)), (halfHeight - (halfHeight / 2)), halfWidth, halfHeight)));

            ViewedPhoto.Source = _photo.Image;
        }

        private void BlackAndWhite(object sender, RoutedEventArgs e)
        {
            BitmapSource img = (BitmapSource)(_photo.Image);
            _photo.Image = BitmapFrame.Create(new FormatConvertedBitmap(img,
PixelFormats.Gray8, BitmapPalettes.Gray256, 1.0));

            ViewedPhoto.Source = _photo.Image;
        }
    }
}

using System;
using System.Globalization;
using System.Windows.Data;

namespace SDKSamples.ImageSample
{
    /// <summary>
    /// Converts an exposure time from a decimal (e.g. 0.0125) into a string (e.g. 1/80)
    /// </summary>
    public class ExposureTimeConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

        {
            try
            {
                decimal exposure = (decimal)value;

                exposure = Decimal.Round(1 / exposure);
                return String.Format("1/{0}", exposure.ToString());
            }
            catch (NullReferenceException)
            {
                return null;
            }
        }

        public object ConvertBack(object value, Type targetTypes, object parameter, CultureInfo
culture)
```

```
        {
            string exposure = ((string) value).Substring(2);
            return (1 / Decimal.Parse(exposure));
        }
    }

    /// <summary>
    /// Converts an exposure mode from an enum into a human-readable string (e.g. AperturePriority
    /// becomes "Aperture Priority")
    /// </summary>
    public class ExposureModeConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
        {
            ExposureMode exposureMode = (ExposureMode)value;

            switch (exposureMode)
            {
                case ExposureMode.AperturePriority:
                    return "Aperture Priority";

                case ExposureMode.HighSpeedMode:
                    return "High Speed Mode";

                case ExposureMode.LandscapeMode:
                    return "Landscape Mode";

                case ExposureMode.LowSpeedMode:
                    return "Low Speed Mode";

                case ExposureMode.Manual:
                    return "Manual";

                case ExposureMode.NormalProgram:
                    return "Normal";

                case ExposureMode.PortraitMode:
                    return "Portrait";

                case ExposureMode.ShutterPriority:
                    return "Shutter Priority";

                default:
                    return "Unknown";
            }
        }

        public object ConvertBack(object value, Type targetTypes, object parameter, CultureInfo
culture)
        {
            throw new NotSupportedException();
        }
```

```
    }

    /// <summary>
    /// Converts a lens aperture from a decimal into a human-preferred string (e.g. 1.8 becomes F1.8)
    /// </summary>
    public class LensApertureConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
        {
            if (value != null)
            {
                return String.Format("F{0:##.0}", value);
            }
            else
            {
                return String.Empty;
            }
        }

        public object ConvertBack(object value, Type targetTypes, object parameter, CultureInfo
culture)
        {
            if (!String.IsNullOrEmpty((string)value))
            {
                return Decimal.Parse(((string)value).Substring(1));
            }
            else
            {
                return null;
            }
        }
    }

    /// <summary>
    /// Converts a focal length from a decimal into a human-preferred string (e.g. 85 becomes 85mm)
    /// </summary>
    public class FocalLengthConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
        {
            if (value != null)
            {
                return String.Format("{0}mm", value);
            }
            else
            {
                return String.Empty;
            }
        }

        public object ConvertBack(object value, Type targetTypes, object parameter, CultureInfo
culture)
```

```
        {
            throw new NotSupportedException();
        }
    }

    /// <summary>
    /// Converts an x,y size pair into a string value (e.g. 1600x1200)
    /// </summary>
    public class PhotoSizeConverter : IMultiValueConverter
    {
        public object Convert(object[] values, Type targetType, object parameter, CultureInfo culture)
        {
            if ((values[0] == null) || (values[1] == null))
            {
                return String.Empty;
            }
            else
            {
                return String.Format("{0}x{1}", values[0], values[1]);
            }
        }

        public object[] ConvertBack(object value, Type[] targetTypes, object parameter, CultureInfo
culture)
        {
            if ((string) value == String.Empty)
            {
                return new object[2];
            }
            else
            {
                string[] sSize = new string[2];
                sSize = ((string)value).Split('x');

                object[] size = new object[2];
                size[0] = UInt32.Parse(sSize[0]);
                size[1] = UInt32.Parse(sSize[1]);
                return size;
            }
        }
    }
}
using System;
using System.Collections.ObjectModel;
using System.IO;
using System.Windows.Media.Imaging;

namespace SDKSamples.ImageSample
{
    /// <summary>
    /// This class describes a single photo - its location, the image and
    /// the metadata extracted from the image.
```

```
/// </summary>
public class Photo
{
    public Photo(string path)
    {
        _path = path;
        _source = new Uri(path);
        _image = BitmapFrame.Create(_source);
        _metadata = new ExifMetadata(_source);
    }

    public override string ToString()
    {
        return _source.ToString();
    }

    private string _path;

    private Uri _source;
    public string Source { get { return _path; } }

    private BitmapFrame _image;
    public BitmapFrame Image { get { return _image; } set { _image = value; } }

    private ExifMetadata _metadata;
    public ExifMetadata Metadata { get { return _metadata; } }

}

/// <summary>
/// This class represents a collection of photos in a directory.
/// </summary>
public class PhotoCollection : ObservableCollection<Photo>
{
    public PhotoCollection() { }

    public PhotoCollection(string path) : this(new DirectoryInfo(path)) { }

    public PhotoCollection(DirectoryInfo directory)
    {
        _directory = directory;
        Update();
    }

    public string Path
    {
        set
        {
            _directory = new DirectoryInfo(value);
            Update();
        }
        get { return _directory.FullName; }
```

```
    }

    public DirectoryInfo Directory
    {
        set
        {
            _directory = value;
            Update();
        }
        get { return _directory; }
    }
    private void Update()
    {
        this.Clear();
        try
        {
            foreach (FileInfo f in _directory.GetFiles("*.jpg"))
                Add(new Photo(f.FullName));

        }
        catch (DirectoryNotFoundException)
        {
            System.Windows.MessageBox.Show("No Such Directory");
        }
    }

    DirectoryInfo _directory;
}


public enum ColorRepresentation
{
    sRGB,
    Uncalibrated
}

public enum FlashMode
{
    FlashFired,
    FlashDidNotFire
}

public enum ExposureMode
{
    Manual,
    NormalProgram,
    AperturePriority,
    ShutterPriority,
    LowSpeedMode,
    HighSpeedMode,
    PortraitMode,
    LandscapeMode,
```

```
        Unknown
    }

    public enum WhiteBalanceMode
    {
        Daylight,
        Fluorescent,
        Tungsten,
        Flash,
        StandardLightA,
        StandardLightB,
        StandardLightC,
        D55,
        D65,
        D75,
        Other,
        Unknown
    }

    public class ExifMetadata
    {
        BitmapMetadata _metadata;

        public ExifMetadata(Uri imageUri)
        {
            BitmapFrame frame = BitmapFrame.Create(imageUri,
BitmapCreateOptions.DelayCreation, BitmapCacheOption.None);
            _metadata = (BitmapMetadata)frame.Metadata;

        }

        private decimal ParseUnsignedRational(ulong exifValue)
        {
                return (decimal)(exifValue & 0xFFFFFFFFL) / (decimal)((exifValue &
0xFFFFFFFF00000000L) >> 32);
        }

        private decimal ParseSignedRational(long exifValue)
        {
            return (decimal)(exifValue & 0xFFFFFFFFL) / (decimal)((exifValue &
0x7FFFFFFF00000000L) >> 32);
        }

        private object QueryMetadata(string query)
        {
            if (_metadata.ContainsQuery(query))
                return _metadata.GetQuery(query);
            else
                return null;
        }

        public uint? Width
```

```
{
   get
   {
      object val = QueryMetadata("/app1/ifd/exif/subifd:{uint=40962}");
      if (val == null)
      {
         return null;
      }
      else
      {
         if (val.GetType() == typeof(UInt32))
         {
            return (uint?)val;
         }
         else
         {
            return System.Convert.ToUInt32(val);
         }
      }
   }
}

public uint? Height
{
   get
   {
      object val = QueryMetadata("/app1/ifd/exif/subifd:{uint=40963}");
      if (val == null)
      {
         return null;
      }
      else
      {
         if (val.GetType() == typeof(UInt32))
         {
            return (uint?)val;
         }
         else
         {
            return System.Convert.ToUInt32(val);
         }
      }
   }
}

public decimal? HorizontalResolution
{
   get
   {
      object val = QueryMetadata("/app1/ifd/exif:{uint=282}");
      if (val != null)
      {
```

```
            return ParseUnsignedRational((ulong)val);
        }
        else
        {
            return null;
        }
    }
}

public decimal? VerticalResolution
{
    get
    {
        object val = QueryMetadata("/app1/ifd/exif:{uint=283}");
        if (val != null)
        {
            return ParseUnsignedRational((ulong)val);
        }
        else
        {
            return null;
        }
    }
}

public string EquipmentManufacturer
{
    get
    {
        object val = QueryMetadata("/app1/ifd/exif:{uint=271}");
        return (val != null ? (string)val : String.Empty);
    }
}

public string CameraModel
{
    get
    {
        object val = QueryMetadata("/app1/ifd/exif:{uint=272}");
        return (val != null ? (string)val : String.Empty);
    }
}

public string CreationSoftware
{
    get
    {
        object val = QueryMetadata("/app1/ifd/exif:{uint=305}");
        return (val != null ? (string)val : String.Empty);
    }
}
```

```
public ColorRepresentation ColorRepresentation
{
   get
   {
      if ((ushort)QueryMetadata("/app1/ifd/exif/subifd:{uint=40961}") == 1)
         return ColorRepresentation.sRGB;
      else
         return ColorRepresentation.Uncalibrated;
   }
}

public decimal? ExposureTime
{
   get
   {
      object val = QueryMetadata("/app1/ifd/exif/subifd:{uint=33434}");
      if (val != null)
      {
         return ParseUnsignedRational((ulong)val);
      }
      else
      {
         return null;
      }
   }
}

public decimal? ExposureCompensation
{
   get
   {
      object val = QueryMetadata("/app1/ifd/exif/subifd:{uint=37380}");
      if (val != null)
      {
         return ParseSignedRational((long)val);
      }
      else
      {
         return null;
      }
   }
}

public decimal? LensAperture
{
   get
   {
      object val = QueryMetadata("/app1/ifd/exif/subifd:{uint=33437}");
      if (val != null)
      {
         return ParseUnsignedRational((ulong)val);
      }
```

```
        else
        {
            return null;
        }
    }
}

public decimal? FocalLength
{
    get
    {
        object val = QueryMetadata("/app1/ifd/exif/subifd:{uint=37386}");
        if (val != null)
        {
            return ParseUnsignedRational((ulong)val);
        }
        else
        {
            return null;
        }
    }
}

public ushort? IsoSpeed
{
    get
    {
        return (ushort?)QueryMetadata("/app1/ifd/exif/subifd:{uint=34855}");
    }
}

public FlashMode FlashMode
{
    get
    {
        if ((ushort)QueryMetadata("/app1/ifd/exif/subifd:{uint=37385}") % 2 == 1)
            return FlashMode.FlashFired;
        else
            return FlashMode.FlashDidNotFire;
    }
}

public ExposureMode ExposureMode
{
    get
    {
        ushort? mode = (ushort?)QueryMetadata("/app1/ifd/exif/subifd:{uint=34850}");

        if (mode == null)
        {
            return ExposureMode.Unknown;
        }
```

```
        else
        {
            switch ((int)mode)
            {
                case 1: return ExposureMode.Manual;
                case 2: return ExposureMode.NormalProgram;
                case 3: return ExposureMode.AperturePriority;
                case 4: return ExposureMode.ShutterPriority;
                case 5: return ExposureMode.LowSpeedMode;
                case 6: return ExposureMode.HighSpeedMode;
                case 7: return ExposureMode.PortraitMode;
                case 8: return ExposureMode.LandscapeMode;
                default: return ExposureMode.Unknown;
            }
        }
    }
}

public WhiteBalanceMode WhiteBalanceMode
{
    get
    {
        ushort? mode = (ushort?)QueryMetadata("/app1/ifd/exif/subifd:{uint=37384}");

        if (mode == null)
        {
            return WhiteBalanceMode.Unknown;
        }
        else
        {
            switch ((int)mode)
            {
                case 1: return WhiteBalanceMode.Daylight;
                case 2: return WhiteBalanceMode.Fluorescent;
                case 3: return WhiteBalanceMode.Tungsten;
                case 10: return WhiteBalanceMode.Flash;
                case 17: return WhiteBalanceMode.StandardLightA;
                case 18: return WhiteBalanceMode.StandardLightB;
                case 19: return WhiteBalanceMode.StandardLightC;
                case 20: return WhiteBalanceMode.D55;
                case 21: return WhiteBalanceMode.D65;
                case 22: return WhiteBalanceMode.D75;
                case 255: return WhiteBalanceMode.Other;
                default: return WhiteBalanceMode.Unknown;
            }
        }
    }
}

public DateTime? DateImageTaken
{
    get
```

```
            {
                object val = QueryMetadata("/app1/ifd/exif/subifd:{uint=36867}");
                if (val == null)
                {
                    return null;
                }
                else
                {
                    string date = (string)val;
                    try
                    {
                        return new DateTime(
                            int.Parse(date.Substring(0, 4)),    // year
                            int.Parse(date.Substring(5, 2)),    // month
                            int.Parse(date.Substring(8, 2)),    // day
                            int.Parse(date.Substring(11, 2)),   // hour
                            int.Parse(date.Substring(14, 2)),   // minute
                            int.Parse(date.Substring(17, 2))    // second
                        );
                    }
                    catch (FormatException)
                    {
                        return null;
                    }
                    catch (OverflowException)
                    {
                        return null;
                    }
                    catch (ArgumentNullException)
                    {
                        return null;
                    }
                    catch (NullReferenceException)
                    {
                        return null;
                    }
                }
            }

        }
    }
}
```