Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та
комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

другий (магістерський)
(рівень вищої освіти)

на тему: Інформаційна система автоматизованого документування стартапів

Виконав: студент _VI_ курсу групи _КН-61_
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Кравчук Н.В.
(прізвище та ініціали)

Керівник Мокрицька О.В.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Львів – 2021

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних технологій
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Крошний І. М.

"___" _____ 2020__ року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Кравчук Назар Вікторович
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна система автоматизованого документування стартапів

керівник роботи *Мокрицька Ольга Володимирівна, к. т. н., доцент*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "31"грудня 2020 року № С-593

2. Термін подання студентом роботи 10 грудня 2021

3. Вихідні дані до роботи *знання мови програмування C#, технологій UiAutomation та React, знання будови клієнт – серверної системи*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1. Створити систему, що може захоплювати дії користувача у вигляді активних елементів та скріншотів вікон. 2. Здійснити візуалізацію даних у вигляді списку кроків. 3. Перетворити отримані данні у формат Word.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Схема використання системи, функціонально – логічна схема, схема потоків даних: таблиці даних з полями та взаємозв'язок між ними, дерево цілей, клієнт-серверна архітектура, схема перетворення даних в Microsoft Word формат

6. Дата видачі завдання 18.12.2020

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Ознайомлення з предметною областю, дослідження	18.12.2020 – 02.05.2021	
	ринку та основних конкурентів, визначення цільової		
	аудиторії.		
2.	Проектування архітектури, визначення основних	02.05.2021 – 01.10.2021	
	функціональних модулів та їх призначення. Аналіз		
	зв’язків у системі.		
3.	Реалізація основних модулів системи	01.10.2021 – 20.10.2021	
4.	Реалізація користувацького інтерфейсу. Корегування	20.10.2021 – 07.11.2021	
	відображення кроків. Реалізація авторизації в системі.		
5.	Тестування системи. Знаходження та виправлення	07.11.2021 – 15.11.2021	
	помилок в системі.		
6.	Розроблення стартапу-проекту	15.11.2021 – 01.12.2021	
7.	Оформлення пояснювальної записки	01.12.2021 – 10.12.2021	

Студент _____ **Кравчук Н.В.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Мокрицька О.В.**
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить 62 сторінок. Пояснювальна записка містить 18 ілюстрацій, 1 таблицю, 17 використаних джерел та двох додатків, які містять код програмного продукту та 6 графічних матеріалів.

У результаті виконання даної дипломної роботи розроблено автоматизовану інформаційну систему для автоматизації документування процесів та формування звітів. Робота має практичне значення для забезпечення підвищення продуктивності та полегшення документування процесів в середині підприємства, що займається різними цифровими завданнями.

Метою дипломної роботи є дослідження наявних технологій для роботи з сучасним стеком десктоп-програмування в розрізі захоплення даних, налаштування середовища розробки програми в середовищі ОС Windows, а також відпрацювання навичок роботи із сучасними C# - бібліотеками для візуалізації даних і їх правильного поєднання в одній робочій системі.

Об’єктом дослідження даної роботи є настільна система для захоплення дій користувача та перетворення їх у документаційний файл.

Предмет дослідження – програмна реалізація системи у вигляді декількох компонетів:

- Модуль авторизації
- Модуль захоплення даних
- Модуль перегляду даних
- Модуль перетворення даних у Microsoft Word формат

Ключові слова: АВТОМАТИЗАЦІЯ, ДОКУМЕНТАЦІЯ, ПРОЦЕСИ, ВІЖУАЛ СТУДІО, КЛІЄНТ-СЕРВЕР.

ABSTRACT

Thesis contains 63 pages. The explanatory note contains 18 illustrations, 1 table, 17 sources used and two appendices, which contain the code of the software product and 6 graphic materials.

As a result of this thesis, an automated information system was developed to automate process documentation and report generation. The work is of practical importance to increase productivity and facilitate the documentation of processes within the enterprise engaged in various digital tasks.

The aim of the thesis is to study existing technologies for working with a modern desktop programming stack in terms of data capture, setting up a software development environment in Windows, as well as practice skills with modern C # - libraries to visualize data and combine them in one work system.

The object of research of this work is a desktop system for capturing user actions and converting them into a documentation file.

The subject of research - software implementation of the system in the form of several components:

• Authorization module

• Data capture module

• Data viewing module

• Module to convert data to Microsoft Word format

Keywords: AUTOMATION, DOCUMENTATION, PROCESSES, VISUAL STUDIO, CLIENT SERVER.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АІС – автоматизована інформаційна система;

АС – автоматизовані системи;

АСК – автоматизована система керування;

ІС – інформаційна система;

БД – база даних;

ОС – операційна система.

ЗМІСТ

ВСТУП

У теперішньому світі існування можливість мати потужної ресурсної бази стартапу не обіцяє належного рівня конкурентоспроможності на широкому ринку, коли увесь матеріальний та ресурсний потенціали не будуть раціонально та оптимально організовані та узгоджені. Усі потреби на сьогодні на глобальному, світовому чи регіональному ринку заставляють стартап підприємства весь час удосконалюватись та створювати сучасні продукти або покращувати уже існуючі щоб утримувати існуючі позиції та збільшення захоплення більшої частини ринку, а також постійні та стабільні прибутки для свого безпечного існування.

Вище згадані слова зобов'язують підприємство брати на озброєння сучасні більш оптимальні методи керування, базою яких звісно є знання. Для універсального використання навичок та відомостей на всіх стадіях виробничої діяльності та продажу готової продукції стартап організації необхідна система або інструменти для ефективного керування та збором знань і даних усіх учасників робочого процесу.

Метою дипломної роботи є дослідження наявних технологій для роботи з сучасним стеком десктоп-програмування в розрізі захоплення даних, налаштування середовища розробки програми в середовищі ОС Windows, а також відпрацювання навичок роботи із сучасними C# - бібліотеками для візуалізації даних і їх правильного поєднання в одній робочій системі.

Об'єктом дослідження даної роботи є настільна система для захоплення дій користувача та перетворення їх у документаційний файл.

Предмет дослідження – програмна реалізація системи у вигляді декількох складових:

- Модуль авторизації
- Модуль захоплення даних
- Модуль перегляду даних

- Модуль перетворення даних у Microsoft Word формат

Основними завданнями є:

- Створити систему, що може захоплювати дії користувача у вигляді активних елементів та скріншотів вікон.
- Здійснити візуалізацію даних у вигляді списку кроків.
- Перетворити отримані данні у формат Word.

Робота має практичне значення для забезпечення підвищення продуктивності та полегшення документування процесів в середині підприємства, що займається різними цифровими завданнями.

Методики управління та керування знаннями все частіше стають основним фактором створення керівного потенціалу підприємства чи стартапу. У сучасному світі технології та працівники мають узгоджуватись за допомогою використання передових підходів до управління. Теорія керування накопичених інструкцій не є новою, та вона дозволяє нам осмислити нові технології керування та підточити їх до новітніх умов роботи підприємства [1].

Результативний менеджмент знань та інструкцій має за мету бути основним та найбільш пріоритетним вектором розвитку керівного та технічного аспекту на підприємствах нашої країни, та як співпраця нашої батьківщини на європейській та світовій аренах значною мірою означає конкурентні змагання та потребує знаходження більш сучасних методик керування.

Багато підприємств стикаються з проблемами розширення свого бізнесу, які тягнуть за собою питання пришвидшення або автоматизації певних бізнес процесів. Для цього насамперед потрібно ідентифікувати ці процеси – і автоматизована система захоплення дій у цьому випадку слугує швидким механізмом накопичення алгоритмів, чи покрокових дій, що виконуються співробітниками для досягнення їх щоденних цілей.

Питання документування постає дуже часто практично у всіх відомих сферах життя людей. І для ефективного збирання знань у певну систему, яка

постійно оновлюється, потрібен зручний інструмент, що дозволить пришвидшити цей так званий рутинний - проте дуже важливий процес.

Якщо брати до уваги роботу за комп'ютером, то переважно завжди користувач використовує мишку та клавіатуру для виконання своїх дій.

Усі ці переривання від периферійних пристроїв обробляє операційна система проте вона дозволяє також обробляти всі ці сигнали використовуючи ряд виконавців, викликаючи наступного обробника переривання.

Також домінуюча більшість операцій відбуваються з графічними інтерфейсами програм, які надають зручність під час виконання певних завдань. Усі графічні елементи поділяються на свої типи, такі як:

- Кнопки
- Текстові поля
- Випадаючі списки
- Радіо переключателі
- Галочки

За допомогою технології WinDomElements можна досліджувати графічні програми та графічні дерева елементів та визначати з якими елементами та якими програмами і процесами, взаємодіє користувач [2].

Паралельно відстежувати набір тексту з клавіатури, і в результаті ми зможемо захоплювати дії користувача реагувати на ці події створювати скріншоти та витягувати мета дані з активних елементів графічних вікон програм.

У результаті таких явних показників можна з сміливістю сказати, що інструмент дозволяє записувати покрокові інструкції без зайвої роботи та накопичувати велику систему знань в підприємстві, для підвищення конкурентоспроможності кожного працівника, що має **наукову новизну** у вирішення проблем документування.

Розділ 1. Стан проблемної області

1.1. Актуальність предметної області

Питання документації постає дуже часто практично у всіх відомих сферах життя людей. Дуже часто потрібно задокументувати певні процеси або кроки, які потрібно зробити щоб досягнути успіху.

Документування – це процес створення документів, збереження інформації про певний процес чи дію за допомогою інструментів на певному носію.

Метод документування – це метод або кілька методів, що дозволяють зберігати інформацію на інформаційному носію за допомогою таблиці символів.

Спосіб документування – це сукупність покрокових дій, які потрібно виконати, щоб збереження інформації на певному носії даних.

Номенклатура засобів складання, копіювання і розмноження документів, включає ручні, механічні, автоматизовані пристрої починаючи з гусячого пера і закінчується ЕОМ [3].

Система документування – це система (автоматизована), яка дозволяє накопичувати данні для певної компанії чи підприємства, надає зручні методи для пошуку необхідних даних та відображення документації.

Документування має в собі ряд процесів, а саме оформлення, складання, затвердження, узгодження, виготовлення та розповсюдження документів. Більшість з цих процесів контролюються актами законів, відомчими нормативними документами, стандартами, а також посібниками та довідниками, що мають рекомендаційний або нормативний характер дії.

Коли людина самостійно працює над певним процесом і виконує свою роботу, вона генерує певні нові алгоритми і кроки щоб виконати свою роботу швидше та краще.

Та для того щоб інші зацікавлені працівники чи особи що задіяні в схожій сфері теж мали змогу оцінити ці кроки, перша людина повинна

задокументувати свій підхід до вирішення поставленого типу завдань. Для цього використовуючи складні інструменти та витрачаючи додаткові години для написання детальної документації. Створюючи окремо скріншоти та детально описуючи кожен крок, навіть коли вони доволі прості, але їх кількість дуже велика і вимагає проведення так званої рутинної роботи [3].

Тут у пригоді явно стає програма, яка з легкістю дозволить практично не відволікаючи людину від основної роботи, захопити його дії, отримати результат та перетворити його в формат для поширення документації і в тому ж числі з економити години часу.

На сьогодні існує ряд дуже схожих програмних рішень. Але через непоширеність на нашому ринку, невеликий функціонал щодо документування та не дуже зручний інтерфейс дані програмні продукти не завоювали свого місця на ринку. І тому я дійшов висновку і можу стверджувати, що на даний момент розробки - розробка моєї програми та продукту в цілому є доцільне рішення.

Для того, щоб програма користувалося популярністю, додаток повинен бути представлений у вигляді фонового записувача, мати вікно контролю й зрозумілу структуру користування, забезпечувати швидке представлення захоплених дій у зручному форматі представлення інформації.

Зважаючи на те, що розробка програмного продукту ведеться в короткі терміни не визначені, потрібно включити ряд ризиків:

- Недостатнє покриття тестами систему.
- Недостатня оптимізація програми.
- Неправильний вибір шаблонів програми для цього програмного рішення.

Для мінімізації впливу цих побічних чинників розробки програмного рішення, було вирішено додати час для додаткового тестування.

Об’єктом дослідження даної роботи є настільна система для захоплення дій користувача та перетворення їх у документаційний файл.

Предмет дослідження – програмна реалізація системи у вигляді декількох складових:

- Модуль авторизації
- Модуль захоплення даних
- Модуль перегляду даних
- Модуль перетворення даних у Microsoft Word формат

1.2. Огляд існуючих систем документування ПЗ

В світі ринкових економік компанія майже завжди оточена конкурентами що присутні на цьому ринку. Ринок включає наявність змінного числа незалежних гравців і менеджерів з продажу, що можуть залишатись на ринку або покинути ринок при нагальній потребі.

З однієї сторони, більша кількість учасників на ринку сприяє добрій змагальній атмосфері що повинна позитивно впливати на весь сегмент ринку. Проте з іншої сторони, усі компанії та гравці на ринку хочуть досягнути кращих результатів за інших. Саме цей момент ринкового життя, спричиняє постійний моніторинг конкурентів, їхніх технологій, маркетингових стратегій та оцінку конкурентоспроможності щодо основних лідерів на ринку в даній сфері. Виділимо основні напрямки дослідження конкурентів.

Наступним кроком для аналізу конкурентів на ринку є дослідження головних чинників конкурентоспроможності компанії. Саме через це особливо важливою є роль маркетингових досліджень та інновацій, тому що через методи дослідження, ба більше методики опитування клієнтів, оцінюється та формується список ключових конкурентних показників: споживчих характеристик продукту, його якості, суми, унікальності та оригінальності. При оцінюванні показників конкурентоспроможності найбільшу цінність мають знання маркетингових досліджень, документи, отримані від різних каналів збуту товарів.

Під час дослідження ринку документування мені вдалось знайти потужні інструменти документування та більшість з них вимагають використовувати довгі та рутинні методи для створення документації.

Також є декілька дуже простих інструментів які дозволяють працювати з діями користувача та це практично захоплення поодиноких дії без змоги експорту в інші формати даних, серед них:

- SwipeGuide;
- Snagit;
- ScreenSteps;
- CloudApp.

Виходячи з аналізованої вище інформації можна сказати, що розробка подібної системи є надзвичайно актуальною в наш час, оскільки на ринку немає її аналогів. Основними споживачами цього програмного продукту є навчальні заклади, інтернет компанії, стартапи, що займаються накопичуванням та поширенням своїх знання.

Дане програмне рішення – це хороша можливість зробити цінний продукт з точки зори користування, що дозволить користувачам, економити години робочого часу для документування, створюючи покрокові інструкції базуючись на діях користувача.

У свою чергу програма потребує не значні затрати на розробку першої версії продукту, більшість коштів доцільно направити на маркетингові активності, для охоплення нових програмних ринків, та купівлю реклами для її конвертування в активних користувачів.

1.3. Класифікація документів

З ціллю організації та раціонального процесу створення документообігу всі документи можна поділити за потоками, наприклад, незареєстровані і зареєстровані документи; вхідні, вихідні та внутрішні документи; документи, що подаються до і надходять з організацій, або документи, що подаються до або надходять з підвідомчих організацій [4].

1.3.1. Класифікація за характером знакових засобів фіксації

Данні, що зберігаються в файлах здебільшого використовують різні таблиці кодування символі. Знаковість - обов'язкова характеристика документа, оскільки лише в закодованій знаками манері можна передати данні від реципієнта до комуніканта.

Ця властивість дозволяє нам класифікувати документи у наступній формі:

- Іконічні
- Звукові
- Цифрові
- Ідеографічні
- Матричні
- Текстові
- Комплексні

1.3.2. Класифікація за виміром запису інформації

За цією характеристикою можна розрізняти одно -, двох - і тривимірні документи (перфострічка, службовий лист, глобус).

1.3.3. Класифікація за призначеністю для сприйняття інформації

За цією характерною особливістю розрізняють документи, що можуть бути прочитані людиною, й ті, що можуть бути прочитані за допомогою технічних та електронних засобів (за допомогою магнітофону, телевізора, ЕОМ техніки, розумного телефону тощо).

1.3.4. Класифікація за каналом сприймання інформації

Інформація, що знаходиться в документі, може бути сприйнятою за використанням різних органів відчуття.

За цією характеристикою розрізняють наступні види документів:

- Візуальні – можна побачити

- Тактильні – можна відчути на дотик
- Аудіальні – можна прослухати
- Аудіовізуальні – відео документи

1.3.5. Класифікація за ступенем розповсюдженості інформації

За ступенем поширення інформації документи класифікують на:

- Опубліковані – доступні для читання.
- Неопубліковані – документи на стадії підготовки.
- Непубліковані – готові документи, та не відкриті для доступу.
- Електронні – представленні у вигляді файлів на ЕОМ.

Зазначені згори основні види і характеристики документів за їх основними складовими для кращого візуального сприйняття можна переглянути на рис. 1.1

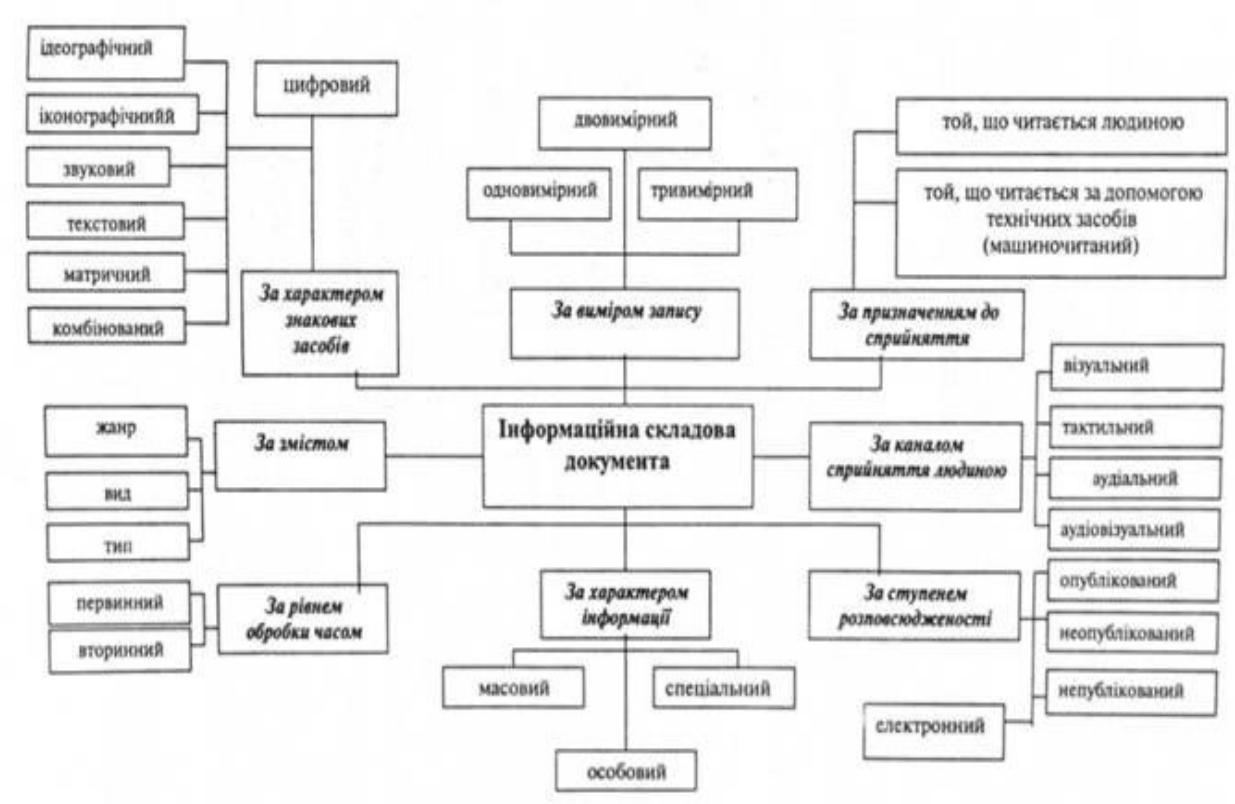


Рис. 1.1. Класифікація документів за характером соціальної інформації

1.4. Висновки до розділу

В даному розділі ви ознайомились з темою та метою моєї магістерської кваліфікаційної роботи, дізнались більше про набір сучасних технологій використовуваних під час розробки, а також оцінили вимоги до даного програмного продукту.

Особливу увагу під час написання цього розділу було приділено аспектам актуальності предметної області.

Також були розглянуто та складено ряд вимог до настільної програми, її функціоналу, вигляду та швидкодії.

Розділ 2. Інформаційне забезпечення

В цьому розділі ми розглянемо основні технології, що були вибрані для програмної реалізації даної системи. Також визначимо основні переваги та принципи роботи цих технологій, платформ, алгоритмів та підходів.

Основними моментами, що ми повинні вибрати та дослідити являються:

- Платформа, яку ми будемо використовувати для написання основного додатка
- Мова програмування
- Технологія створення користувацького інтерфейсу
- Технологія дослідження користувацького інтерфейсу та пошуку активних елементів взаємодії.
- База даних

2.1. Підвищення ефективності діяльності підприємства

Удосконалення системи знань на підприємстві - це ряд заходів, що дозволяє допомогти керуючій еліті підприємства ефективніше планувати та використовувати ресурси, в свій час і пришвидшити знаходження, та виправлення недоліків в поширенню знань.

Якщо новий персонал на підприємстві працює не дуже ефективно або не так настільки ефективно, як хотілося б еліті чи процес навчання відбувається дуже довго чи взагалі немає чіткого плану по “онбордінгу”, потрібно поставити питання створення системи знань на підприємстві та найпростішими методами задокументувати основні процеси в кожній критичній сфері? Якщо виправити проблеми з цим в керуючій системі, то більше існуючих проблем на підприємстві зможуть зникнути самі собою. Таким чином, гроші, що витрачені на створення бази знань, в майбутньому неодноразово окупляться, бо не ефективність системи знань на фірмі - це

майже "смертний вирок" цій компанії в найближчому світлому майбутньому [6].

У сьогоднішніх умовах поряд з усіма іншими ресурсами такими як, фінанси, матеріали, людський персонал та інші ресурси, ефективне керування являє собою дуже цінний потенціал компанії. А отже, підвищення ефективності розвитку нових кадрів та персоналу стає одним з напрямків заохочення діяльності підприємства в цілому. Найбільш очікуваним методом підвищення ефективності є створення документів та його автоматизація.

Управлінська робота відрізняється складністю і різноманіттям, наявністю великого числа форм і видів, багатосторонніми залученнями і взаємодією з різними явищами і процесами. Це насамперед праця творча та інтелектуальна, в основному пов'язаний з IT. Тому автоматизація управлінської роботи напочатку зосереджена з автоматизацією деяких додаткових, так званих рутинних операцій. Але бурхливий розвиток інформаційних та комп'ютерних технологій, вдосконалення технічної платформи розробки і поява принципово нових парадигм програмних продуктів привело в наші дні до редагування підходів в автоматизації управління не тільки виробництвом, а й іншими процесами [4].

Головним шляхом автоматизаційної роботи та її радикального вдосконалення, і заточення її до сучасних умов, стає масове використання електронної комп'ютерного і телекомунікаційного обладнання, формування на його стороні високоефективних IT підрозділів. Засоби та методи прикладних інформаційних технологій використовуються для управління та у маркетингу.

Нові технології, що були створені на комп'ютерній техніці, вимагають відчутних змін організаційних структур управління, його регламенту, кадрового потенціалу, системи документації, фіксації переробки і передачі інформації. Особливе значення має впровадження IT, значно розширювальної можливості використання компаніями інформаційних ресурсів. Розвиток IT зв'язано з створенням системи обробки та збереження

даних і знань, їх послідовного розвитку до ступеня інтегрованих інформаційних автоматизованих систем на підприємствах, що можуть охоплювати по вертикалі всі гілки і дерева виробництва [7].

Згідно визначення, прийнятого ЮНЕСКО, IT - це комплекс що включає ряд наукових, технологічних, інженерних сфер та дисциплін, що вивчають ефективні методи організації колективної роботи людей, залучених опрацюванням і зберіганням інформації; обчислювальну електронну техніку і методи організації і взаємодії з людьми і виробничим устаткуванням, практичні додатки, а також пов'язані з усім цим соціальні, економічні та культурні проблеми. Самі IT потребують складної підготовки, інвестицій у вигляді великих первинних витрат і наукомісткої, складної техніки [6].

2.2. Дерево цілей

Дерево цілей - це візуальне представлення взаємозв'язків і постановці цілей, що показує розподіл місії і мети та завдань на цілі, під даною цілю, завдання та окремі дії.

Дерево цілей можна назвати, як основний компас компанії, продукту чи діяльності. Загальна форма дерева цілей зображено на рис. 1.2.

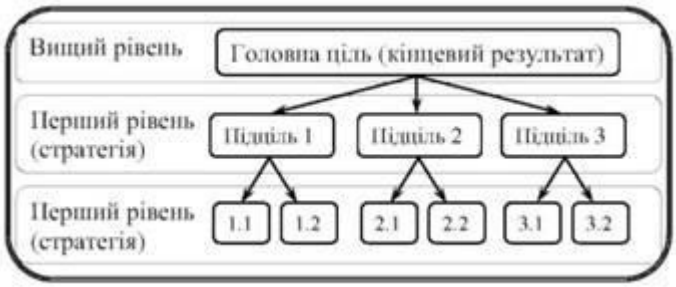


Рис. 2.1. Послідовність побудови дерева цілей

Дерево цілей з явними показниками, які використовують в якості головного із засобів при розгляді та прийнятті рішень, і носить додаткове ім’я дерево рішень. Головна виграшна перевага дерева рішень над іншими

рішеннями - можливість поєднати поставленні цілі з явними діями, що підлягають роботі вже сьогодні.

Основна думка щодо створення дерева цілей являється декомпозиція.

Декомпозиція (розбиття) - це метод розбиття складної системи, при якому за однією характеристикою або властивістю її розділяють на декілька складових.

Декомпозиція часто використовується для створення дерева цілей, щоб поєднати головну мету зі методами її досягнення, що показані у вигляді завдань окремим співробітникам.

Розглянемо технологічні методи створення дерева цілей. Не існує повністю універсальних шляхів створення дерева цілей. Методи його побудови в певній мірі залежать від характеру поставленої цілі, обраної методології, а також від того, хто створює дерево цілей, як він уявляє собі отримані перед ним завдання, як він бачить можливість їх поєднати.

Основна інструкція створення дерева цілей - це повнота редукції - процес приведення складного явища та процесу або системи до простих елементів і частинок.

Для дотримання цієї інструкції використовують такий структурний підхід, як ціль вищого рівня є вказівником, метою для розробки (декомпозиції) завдань більш нижчого рівня. Для автоматизованої інформаційної системи по обслуговуванню клієнтів підприємством, що надає послуги по ремонту мобільних пристроїв також розроблене дерево цілей, що зображене на рис.1.3.

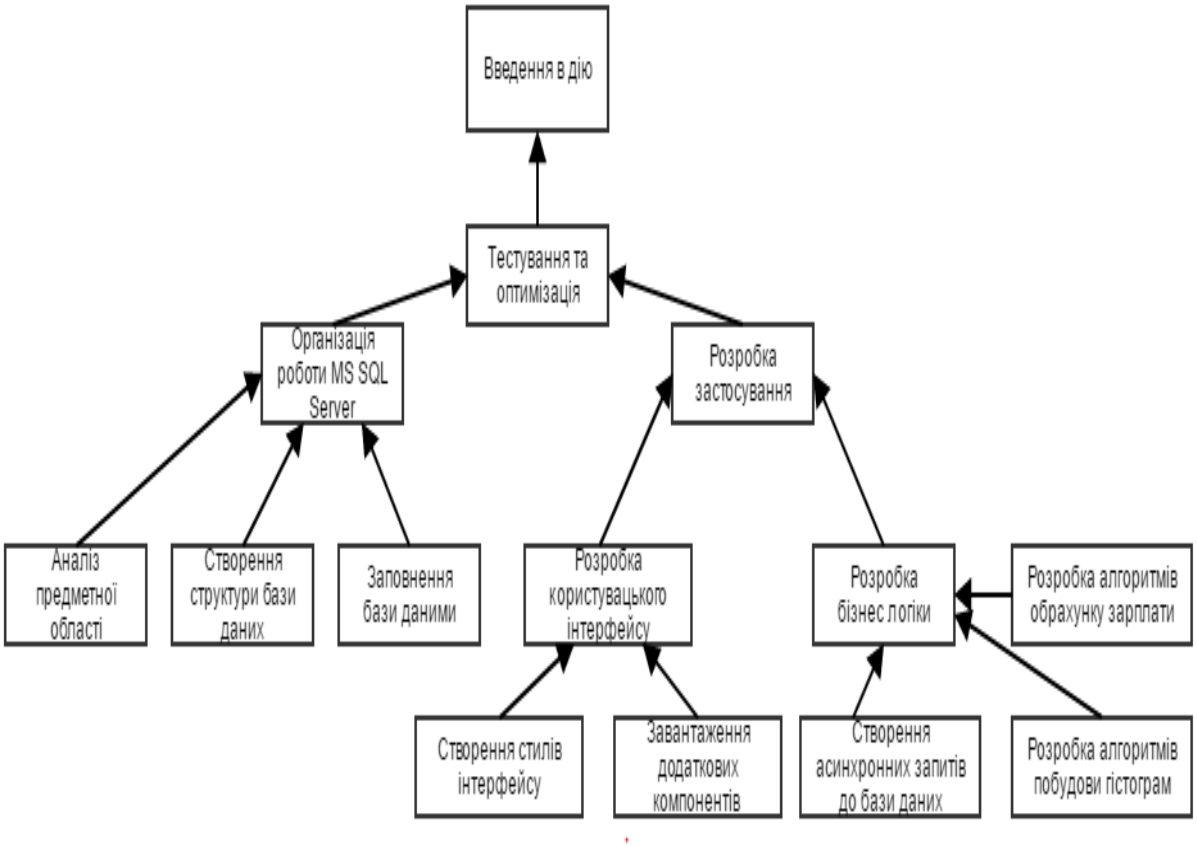


Рис. 2.2. Дерево цілей

2.3. Технологія WPF

Дана технологія WPF (Windows Presentation Foundation) є частина платформи .NET і являє собою підсистему для побудови графічних інтерфейсів для користувачів.

При створенні звичайних додатків на базі WinForms за прорисовку елементів керування і комп'ютерної графіки використовувались графічні частини операційної системи Windows, як бібліотеки User32 і GDI +, то програми на основі WPF побудовані на використанні DirectX. Це основна та ключова відмінність прорисовки графіки на базі WPF: при використанні WPF, левова частка графічної роботи по прорисовці графіки, як нескладних компонентів, так і не простих 3D-моделей, передається на графічний процесор на відеокарті, що також дає переваги по використанню апаратних прискорень графічного відображення [15].

Однією з зручних особливостей побудови графічного інтерфейсу є можливість використання мови розмітки у вигляді HTML інтерфейсу XAML, виготовленого на основі XML: ви можете маніпулювати графічним інтерфейс, за допомогою оголошення інтерфейсу, або з використанням коду на керованих мовах C # і VB.NET, або поєднувати ці дві можливості.

Схематично архітектура WPF зображена на рис. 4.1

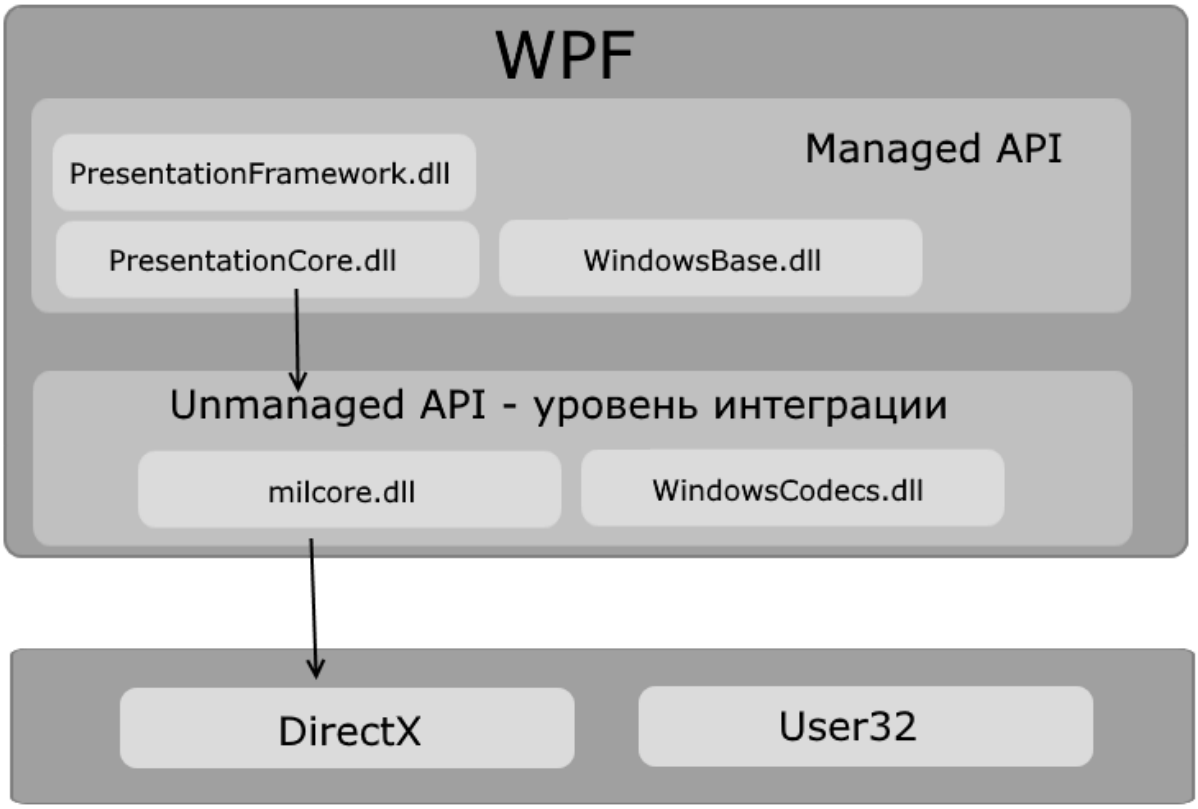


Рис. 4.3. Архітектура WPF

Як зображено на схемі, WPF розкладається на 2 рівня: managed API і unmanaged API (рівень інтеграції з DirectX). Managed API (керований API-інтерфейс) містить код, що виконується під управлінням загальномовного середовища виконання .NET - Common Language Runtime. Цей API описує основний функціонал платформи WPF.

Внизу безпосередньо містяться елементи операційної системи і DirectX, що представляють візуальні компоненти програмного рішення, або дозволяють виконувати додаткову обробку на нижчому рівні. В основному,

за допомогою низько рівневого інтерфейсу відділена Direct3D, що включена у залежності DirectX, та відбувається комунікація.

На цьому рівні також міститься стандарна бібліотека user32.dll. Ми згадували вище, що WPF не містить залежностей на цю бібліотеку для відображення та візуалізації даних, проте все ще для ряду обчислювальних операцій (візуалізація не включається в список цих задач) дана бібліотека все ще широко може використовуватися.

2.4. Мова програмування C#

Мова C# — це багато-парадигмова об'єктно-орієнтована та мова програмування що є таких компонентно-орієнтована зі суворою типизацією, виготовлена для великої платформи .NET Framework. Робота даної мови визначене стандартами ECMA-3344 [14].та ISO/IEC 23270:20065 [15]. Данна мова програмування була виготовлена командою Microsoft Research під наглядом Андерса Гейлсберга. Основний синтаксис мови програмування близький до мов C++ та Java.

Деякі риси (наприклад, сувора статична типизація), наближують її структуру до Delphi (Object Pascal).Цілі, поставлені при розробці мови C#, були такими;[9].

- C# має бути зручно, сучасною, об`єктно-орієнтованою мовою програмування;
- мова має підтримувати безпечні принципи програмування, такі як строга перевірка типів, перевірка меж масиву, виявлення спроб використання не ініціалізованих змінних, і автоматичне прибирання сміття;
- можливість розробки програмних компонентів для розподілених систем;
- мова має підтримувати переносимість коду;
- підтримка національних мовних та інших особливостей має бути простою.

Базовий синтаксис C# схожий до інших C-подібних мов, таких як C, C++ та Java, зокрема:

- крапку з комою використовують для позначення кінця інструкції;
- фігурні дужки використовують для формування програмних блоків;
- значення змінним присвоюють за допомогою знаку "=", а для їх порівняння використовують "= =";
- квадратні дужки використовують для індексації масивів.

Проте, C# має суттєві відмінності від розглянутих мов- попередників, у тому числі: [8].

- краща переносимість між різними платформами, пов`язана з тим, що мова відповідає специфікації CLI;
- строга типизація робить мову безпечнішою. C# підтримує логічний тип Boolean;
- мова забезпечує використання властивостей у класах, роблячи роботу з об`єктами зручнішою та безпечнішою;
- програма на C# краще структурована завдяки групуванню коду в простори імен;
- обмежене використання вказівників робить безпечнішою роботу з пам`яттю.

Версія перша C# 1.0 вийшла разом з Microsoft Visual Studio .NET у лютому 2002 року. В липні 2015 року випущено версію C# 6.0. Порівняно із першою версією, мову C# суттєво доповнили та розширили її числом вагомих рис, що показують її сучасною, простою, гнучкою та перспективною для вирішення великого кола завдань [1]. Існує кілька середовищ програмування, які використовують мову C#, зокрема: Microsoft Visual Studio, MonoDevelop, SharpDevelop тошо.

Найчастіше з них програмісти використовують середовище розробки Microsoft Visual Studio, яке крім C# мови програмування, також підтримує і число інших сучасних мов програмування, що забезпечує великий набір готових класів для розв`язування різного типу завдань.

2.5. Висновки до розділу

В цьому розділі ми розглянули основні технології, що були вибрані для програмної реалізації даної системи. Також визначили основні переваги та принципи роботи цих технологій, платформ, алгоритмів та підходів.

Основними моментами, що ми дослідили та вибрали являються:

- Мова програмування - C#
- Технологія створення користувацького інтерфейсу - WPF
- База даних – MS SQL

Розділ 3. Математичне забезпечення

3.1. Комп`ютерна платформа .Net Framework

.NET Framework - це платформа для програмування. В загальному, комп`ютерна платформа - це ряд програмних застосувань та бібліотек, що служить базою для різних програмних рішень та систем. Яскравим прикладом може бути платформа для програмування - операційна система Windows. Об’єктно орієнтована мова програмування C# була створена для написання програм на платформі .NET [8].

Розробка програмного рішень або (ПЗ) на платформах операційних систем (ОС) класу Windows мала на меті використання мови програмування C у поєднанні із специфічними інструментами ОС Windows, які програмісти називають скорочено API. API - скорочення інтерпретується, як Application Programming Interface - інтерфейс взаємодії програм в програмуванні.

Більшість нових рішень зазвичай дуже не протестовані і не перевірені часом, і дороблюються шляхом довгих випробувань, експлуатації. Можливість відійти від безпосереднього використання API дозволила створити більш стабільні системи для програмування наприклад, наприклад, Borland C++ Builder, що значно дозволили спростити та зробили складну роботу розробника витонченою. Але робота не стоїть на місці а рухається, і мова C на певному етапі свого життя не мала достатньо функцій щоб забезпечувати потреби в програмуванні. І тоді з`явилася концепція що ми всі знаємо і пам’ятаємо це об`єктно-орієнтованого програмування (ООП), що дозволяє розглядати сам процес розробки програмного рішення практично зовсім з іншої точки зору, даючи можливість програмістам та більш ширші методи для автоматизації його роботи і створення більш стабільного та якіснішого програмного продукту [9].

Основою та ядром ООП стали визначення класів та об`єктів. Програмісти мови C пішли обхідним курсом додавання до C поняття "клас". Утворилася мова C++. Цей метод насправді став настільки складним, що думаю, свого моменту розробники напевно сильно пошкодували, що

прийняли саме таку методологію та бути на рівні нових цілей та завдань до процесу виготовлення програмного рішення. У гонитві за швидкодією та можливістю швидше оброблювати даних і за потрібною надійністю і безпекою роботи застосувань розробникам довелося організовувати два види пам`яті при обробці даних: некеровану і керовану та створити спеціальний і досить складний апарат статистик [10].

Врешті решт, мабуть, у розробників терпець увірвався, і вони створили нову мову під назвою C#, яка враховує нові віяння в програмуванні (ООП) і вільну від недоліків C++. Проте і C++ не виявилася покинутою з причини, відміченої раніше (сумісність і підтримка вже працюючих у світі програм).

В основному, платформа система .NET Framework – це суб'єктивно середовище виконання з можливістю керування, що надає ряд різноманітних служб виконуючим в ньому програмам. Вона має в собі два основних компонентів: виконавчого середовища спільної мови (Common Language Runtime, CLR), яка є механізмом, управляючим застосуванням, яке виконуються, і бібліотеки класів .NET Framework, яка представляє список бібліотек з тестованим та перевіреним кодом, що призначені для повторного використання, що програміст може використовувати зі своїх застосувань.

Послуги та сервіси, які платформа .NET Framework надає виконуючим програмним рішенням:

- Управління пам`яттю.
- Система спільного типу. Усі типи даних в цій систему успадковують функції та методи від батьківського класу Object.
- Широка бібліотека класів та реалізацій. Цей список включає готові бібліотеки для роботи з най різноманітнішими задачами, такими як: робота з файлами, робота з передачею даних через мережу інтернету, графічне представлення користувацького інтерфейсу.
- Платформи і шаблони розробки. Дана платформа включає в себе так би мовити менші під платформи та інструменти. Для прикладу

велика система та фреймворк для розробки додатків в інтернеті – ASP .Net Framework.

- Взаємодія програмних мов та компонентів. Мовні компілятори та перетворювачі вихідного коду у платформі .NET Framework перетворють додаток не у виконуваний код відразу, а в проміжний код, що ми називаємо мовою CIL (Common Intermediate Language), який згодом перетворюється під час без посереднього виконання середовищем CLR.
- Сумісність з попередніми версіями. За неймовірно рідкісними виключеннями, додатки, які створювались за допомогою платформи .NET Framework одної версії, можуть виконуватися без змін значних на більш новішій версії;
- Паралельне виконання. Дана платформа надає ряд інструментів та класів для паралельної роботи програмних продуктів. Це включає контроль над кількістю паралельних операцій, що виконуються. Також методи що дозволяють синхронізувати доступ до обмежених ресурсів.
- Налаштування для різних версій.

Якщо ви не розробляєте застосування .NET Framework, але використовуєте їх, ви не повинні володіти якимись спеціальними знаннями про платформу .NET Framework або її роботу.

Якщо використовується ОС Windows, платформа .NET Framework може бути вже встановлена на комп`ютері. Крім того якщо встановлюється додаток, що вимагає платформу .NET Framework, програма встановлення застосування може інсталювати конкретну версію .NET Framework на вашому комп`ютері. Інколи можна побачити діалогове вікно, яке просить встановити платформу .NET Framework [11].

Зазвичай, не вимагається видаляти які-небудь версії .NET Framework вже встановлені на вашому комп`ютері, тому що використовуваний додаток може залежати від конкретної версії. У разі видалення якої-небудь версії

його виконання може завершитися помилкою. Зверніть увагу, що на одному комп`ютері може бути одночасно завантажено декілька версій платформи .NET Framework. Це означає, що не треба видаляти попередні версії для встановлення пізнішої версії.

Розробник може вибрати будь-яку мову програмування, яка підтримує платформу .NET Framework, для створення застосування. Через те, що платформа .NET Framework забезпечує незалежність і взаємодію мов, можна взаємодіяти з іншими застосуваннями компонентами платформи .NET Framework незалежно від мови, за допомогою якої вони були розроблені.

3.2. Ui Automation framework

Microsoft UI Automation дозволяє допоміжним технологіям та автоматизованим інструментам тестування взаємодіяти з елементами управління інтерфейсом інших програм. У цьому розділі пояснюються основні поняття, на яких базується UI Automation.

API автоматичного інтерфейсу складається з двох частин. Одна частина використовується програмами постачальника послуг UI Automation, а інша використовується клієнтськими програмами UI Automation. API провайдера дозволяє розробникам спеціальних програм управління Microsoft Win32 та інших систем управління виставляти ці елементи управління автоматичним інтерфейсом та робити їх видимими для клієнтських програм. API клієнта дозволяє програмам взаємодіяти з елементами управління в інших додатках та отримувати інформацію про них.

Microsoft UI Automation - це система доступності для Windows. Він забезпечує програмний доступ до більшості елементів інтерфейсу на робочому столі. Це дає можливість допоміжним технологічним продуктам, таким як зчитувачі екрану, надавати інформацію про користувальницький інтерфейс кінцевим користувачам та маніпулювати користувальницьким інтерфейсом іншими способами, ніж стандартне введення даних.

Автоматизація інтерфейсу також дозволяє автоматизованим тестовим сценаріям взаємодіяти з інтерфейсом користувача [12].

Автоматизація інтерфейсу користувача вперше була доступна в Windows XP як частина Microsoft .NET Framework. Хоча в той час був також опублікований некерований API C ++, корисність функцій клієнта була обмежена через проблеми взаємодії. Для Windows 7 API було переписано в модель об'єктної компоненти (COM).

Клієнтські програми UI Automation можна писати із впевненістю, що вони працюватимуть у декількох рамках управління Microsoft Windows. Ядро UI Automation маскує будь-які відмінності в рамках, що лежать в основі різних фрагментів інтерфейсу. Наприклад, властивість вмісту кнопки Windows Presentation Foundation (WPF), властивість Caption кнопки Microsoft Win32 та властивість ALT у зображенні HTML всі відображаються на одне властивість, Name, у вікні автоматизації інтерфейсу користувача.

UI Automation забезпечує повну функціональність у Windows XP, Windows Server 2003 та пізніших операційних системах.

Постачальники UI Automation - це компоненти, що реалізують підтримку UI Automation на елементах управління та пропонують певну підтримку клієнтських програм Microsoft Active Accessibility через вбудовану послугу мостів.

Типи керування Microsoft UI Automation - це властивості, які служать відомими ідентифікаторами, які вказують на тип управління, який представляє певний елемент інтерфейсу, наприклад комбінований ящик або кнопку. Клієнтські програми використовують тип для виявлення можливостей управління та визначення способу взаємодії з ним.

Для кожного типу керування опис включає набір умов, які повинен підтримувати контроль даного типу. Ось список найбільш відомих компонентів взаємодії:

- AppBar Control
- Button Control

- Calendar Control
- CheckBox Control
- ComboBox Control
- DataGrid Control
- DataItem Control
- Document Control
- Edit Control
- Group Control
- Hyperlink Control Type
- Image Control Type
- List Control Type
- ListItem Control Type
- Menu Control Type

Продукти допоміжних технологій та тестові скрипти переходять до дерева автоматизації користувальницького інтерфейсу Microsoft для збору інформації про інтерфейс користувача та його елементи.

У дереві автоматизованого інтерфейсу є кореневим елементом, який представляє вікно робочого столу Windows («робочий стіл») і дочірні елементи представляють вікна програми. Кожен з цих дочірніх елементів може містити елементи, що представляють фрагменти інтерфейсу користувача, такі як меню, кнопки, панелі інструментів і списки. Ці елементи можуть містити елементи, такі як елементи списку.

Дерево автоматизації інтерфейсу користувача не є фіксованою структурою. Його не часто можна побачити у всій сукупності, оскільки він може містити тисячі елементів. Частини дерева автоматичного інтерфейсу побудовані так, як потрібен клієнт, і структура дерева змінюється, коли елементи додаються, переміщуються або видаляються [15].

Постачальники UI Automation підтримують дерево автоматизації інтерфейсу користувача, здійснюючи навігацію серед елементів у фрагменті. Фрагмент - це повне під ділення елементів із певного фрейму і містить

кореневий елемент (званий корінь фрагмента), який зазвичай розміщується у вікні.

Постачальники не переймаються навігацією від одного управління до іншого. Цим керує ядро UIAutomation, яке використовує інформацію від постачальників програм за замовчуванням.

Дерево автоматизації інтерфейсу може бути відфільтровано для створення представлень, які містять лише ті елементи автоматизації користувальницького інтерфейсу, які стосуються конкретного клієнта. Такий підхід дозволяє клієнтам налаштувати структуру, яка представлена через UI Automation, під їх конкретні потреби.

Клієнт може налаштувати подання за допомогою масштабування та фільтрації. Оцінка масштабу - це визначення ступеня перегляду, починаючи від базового елемента. Наприклад, програма може захотіти знайти лише прямих дітей на робочому столі або всіх нащадків вікна програми. Фільтрування - це визначення типів елементів, що входять у вигляд.

Провайдери автоматичної інтерфейсу підтримують фільтрацію шляхом визначення властивостей елементів, включаючи IUIAutomationElement IsControlElement та IUIAutomationElement властивості IsContentElement.

Автоматизація користувальницького інтерфейсу надає три перегляди за замовчуванням: перегляд сировини, режим керування та перегляд вмісту. Ці представлення визначаються типом фільтрації. Обсяг будь-якого виду визначається програмою. Додаток може застосовувати інші фільтри за властивостями; наприклад, щоб включити до перегляду елементів керування лише включені елементи керування.

3.3. Алгоритми підготовки системи до функціонування

3.3.1. Алгоритм реєстрації користувача

Передумова: встановлено інсталер програми на персональний компютер.

1. Користувач натискає на іконку логіну після чого відкривається сторінка авторизації у новій вкладці, вибирає вкладку реєстрації.

2. Заповнює форму реєстрації особистими даними та натискає кнопку Sign Up.

2.1.Вираховується місцезнаходження(країна) користувача.

2.2. Особисті дані записуються в базу даних на сервері. Користувач наразі не підтверджений.

3.Після цього на пошту, вказану в попередньому кроці, відсилається лист підтвердження реєстрації.

4.Клієнт переходить за посиланням вказаним у листі, яке перенаправить його на головну сторінку, реєстрація завершена.

4.1.Користувач стає підтвердженим, оновлення даних БД.

3.3.2. Алгоритм авторизації користувача

1.Користувач натискає на іконку розширення після чого відкривається сторінка авторизації, вкладка входу відкрита по замовчуванню.

2.Заповнює поля логіну та паролю і натискає кнопку Sign In.

3.При успішній авторизації відбувається перехід на головну сторінку сайту.

3.3.3. Алгоритм захоплення дій користувача

1. Користувач натискає на іконку Старт.
2. На клієнтській частині здійснюється зчитування кліків користувача.
3. Виконується попереднє форматування даних.
4. Відбувається перетворення даних для відображення.

3.3.4. Алгоритм обробки та відображення захоплених даних

1. Отримується обєкт з подвійний масивом що представляє зображення.
2. Здійснюється відповідний аналіз позиції мишки на екрані.

2.1. Проводиться вилучення активного елемента під курсором.

2.2. Відбувається проходження по дереву вікон.

2.3. Визначається тип елемента (флажок, випадаючий список, кнопка)

3. Оброблені дані сортуються та передаються в компоненти для відображення.

4. З переданих даних формуються атрибути які додадуться до сформованих за допомогою WPF.

5. Обчислення та відображення є динамічним тому усі модифікації результатів користувачем одразу ж впливають на прорисовку компонентів.

3.4. Захоплення скріншоту

Для захоплення скріншоту потрібні дві речі, а саме:

- Момент коли потрібно робити скріншот
- Розміри та позицію рамки скріншота

Для вирішення цих завдань у попередньому підрозділі ми довідались як підписатися на події що будуть подавати нам сигнал коли створювати скріншоти.

Проте для того щоб програма працювали якомога оптимальніше та зручніше, потрібно обрізати область зображення беручи до уваги лише активне вікно, а не увесь монітор або навіть декілька моніторів.

Для цього нам знову потрібно використати функції операційної системи що дозволяють визначити активне вікно в певний момент часу.

GetForegroundWindow() - функція що дозволяє отримати ід активного вікна в операційній системі. Після чого за допомогою ряду інших функцій:

- CreateCompatibleBitmap
- CreateCompatibleDC

Ми отримуємо розміри активного вікна та перетворимо їх у файл зображення.

3.5. Висновки до розділу

В цьому розділі ми розглянули основні технології, що були вибрані для програмної реалізації даної системи. Також визначили основні переваги та принципи роботи цих технологій, платформ, алгоритмів та підходів.

Основними моментами, що ми дослідили та вибрали являються:

- .Net Framework платформа, яку ми будемо використовувати для написання основного додатка
- UiAutomation - як технологія дослідження користувацького інтерфейсу та пошуку активних елементів взаємодії.
- Пропрацювали основні алгоритми перетворення даних – та дій користувача протягом роботи з програмою.

Розділ 4. Програмне забезпечення

В даному розділі буде розглянуто поступові кроки реалізації даного проекту. Основні кроки:

- Налагодження системи контролю версій – для відстеження змін в проекті та безпечній розробці без можливості втрати коду.
- Створення файлу рішення з використанням Microsoft Visual Studio.
- Безпосередній розгляд настільного модуля з аналізом основних частин з точки зору програмної реалізації.
- Реалізація веб-додатка для авторизації
- Вигляд формату Microsoft Word з середини – трансформація в даний формат.

4.1. Створення локального репозиторія для контролю коду.

Для забезпечення надійної та стабільної роботи над програмним кодом в сучасному світі програмісти не обходяться без системи контролю версіями.

Найбільш популярна і зручна в використанні є GIT – система контролю версій. Дана система являється розділеною системою контролю версій що дозволяє паралельно вести розробку над проектом на необмеженій кількості комп’ютерів. Також з використанням GIT FLOW можна вести розробку нових функцій паралельно не думаючи про втрату коду чи збій при складанні функцій в єдину програму.

Також для забезпечення цілісності і достовірності даних GIT дозволяє синхронізувати локальне сховище з віддаленим. Приклад сервісів і платформ що надають можливість використовувати віддалені репозиторії:

- GitHub
- GitLab
- Bitbucket

Для створення та роботою з GIT було використано програму Git Kraken, детальніший інтерфейс програми можна побачити на Рис.4.1

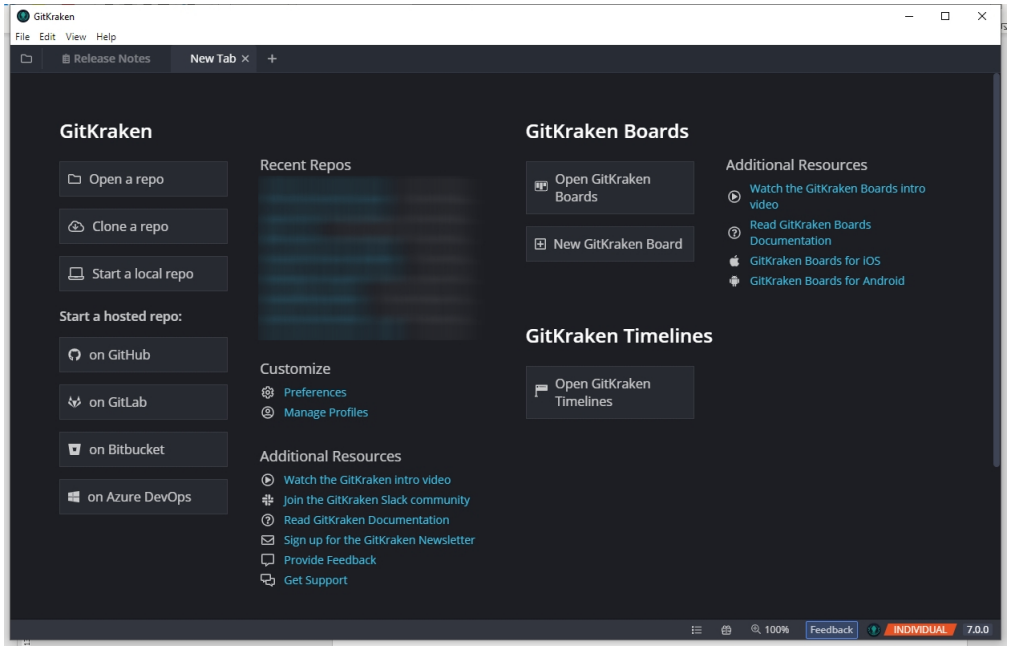


Рис. 4.1. Інтерфейс програми GITKraken

Для того щоб створити новий репозиторій натискаємо на “Start a local repo” і заповнюємо форму, що зображена на рис.4.2.

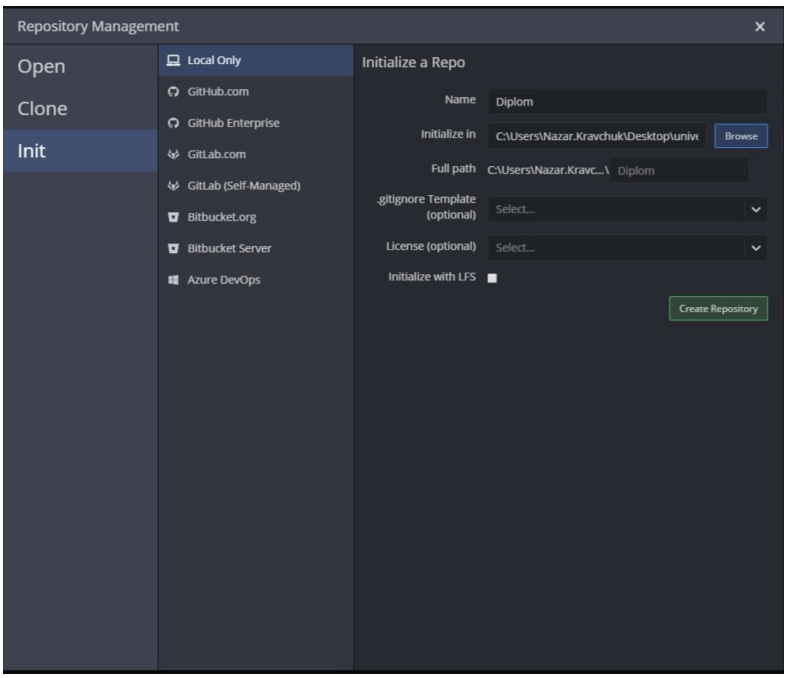


Рис. 4.2. Інтерфейс створення нового репозиторія

4.2. Створення нового рішення з використанням Visual Studio

Для роботи з кодом та для створення програмного рішення було вибрано Microsoft Visual Studio інструмент, та як це надзвичайно потужний

та зручний інструмент для створення графічних програм, особливо під операційну систему Windows. Також в програмі є багато інструментів для побудови графічного інтерфейсу за допомогою графічного покрокового редактора.

Перед початком роботи були вирішення створити нове рішення на базі шаблону створення додатків з графічним інтерфейсом на основі технології WPF. Для створення нового проекту потрібно знайти потрібний шаблон в графічному інтерфейсі (Рис.4.3) “.Net Framework WPF”

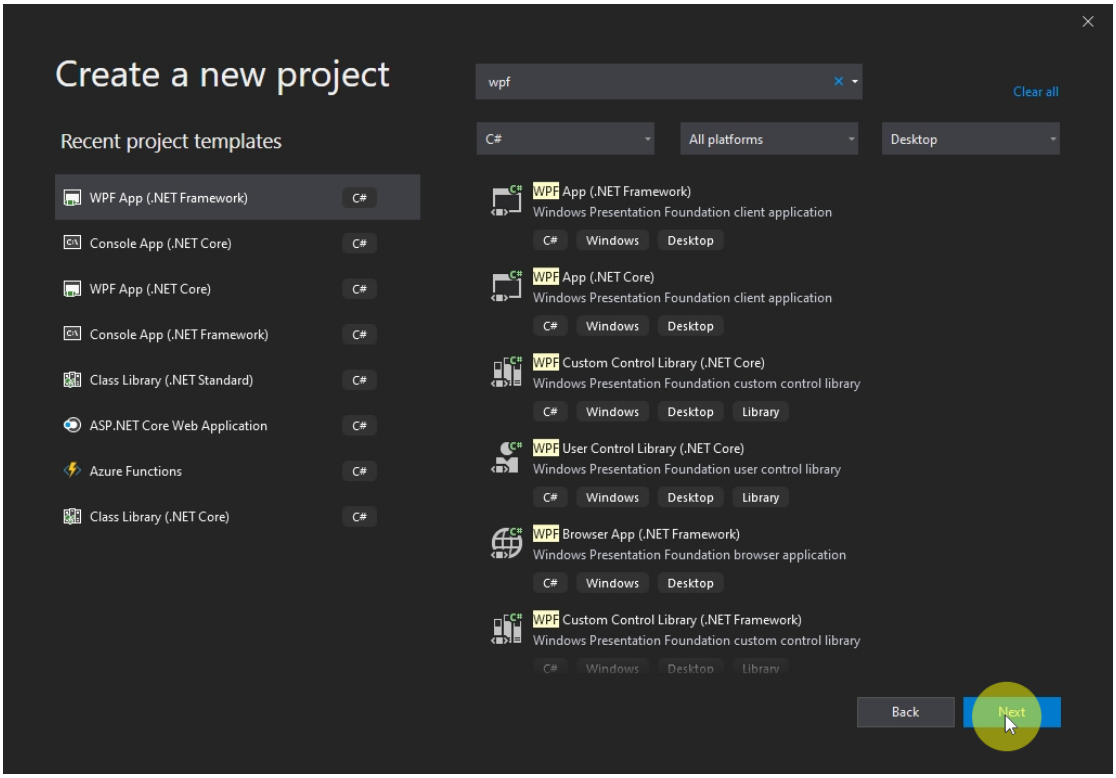


Рис. 4.3. Інтерфейс пошуку шаблону

У наступному інтерфейсі, що зображено на рис.4.4 потрібно вибрати версію .Net Framework та назву рішення.

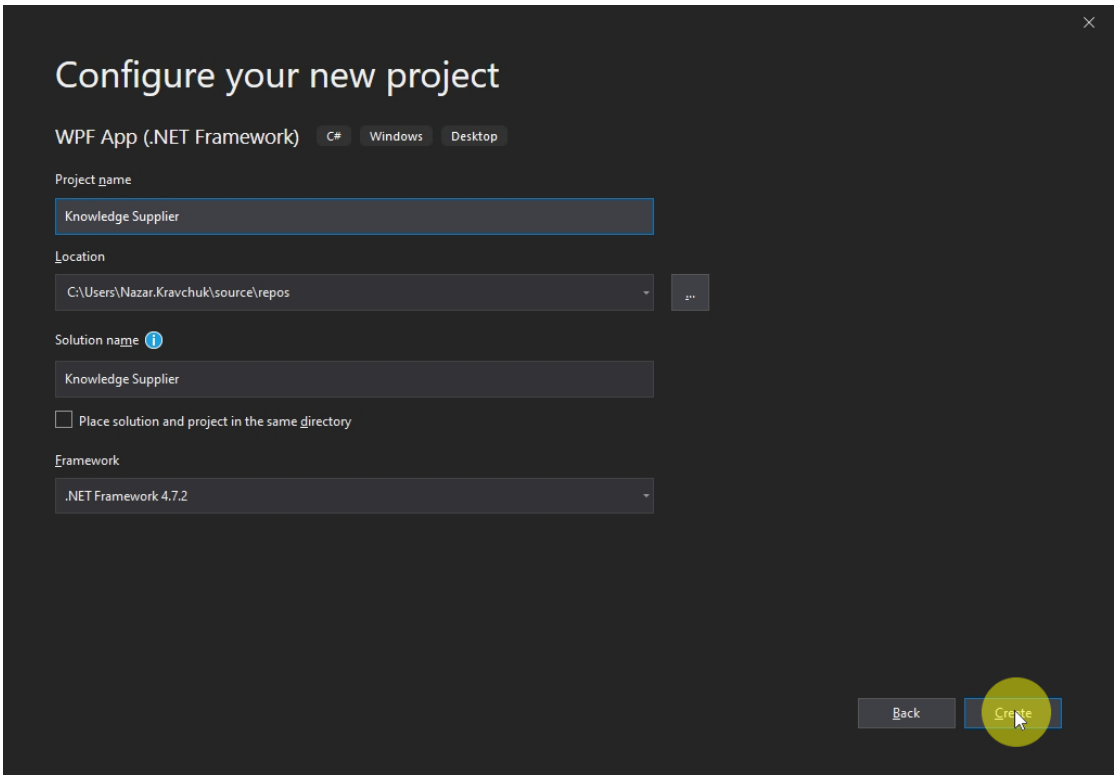


Рис. 4.4. Створення нового рішення

4.3. Модуль захоплення даних

Модуль захоплення дій користувача це основа даного рішення. Для повноцінної роботи програми потрібно налаштувати проект підключивши ряд бібліотек та модулів, а саме:

- Підключити до рішення бібліотеку UiAutomation
- Підключити ряд класів для роботи з Dom Elements
- Підписатися на низько-рівневі події операційної системи
- Визначити активне вікно
- Використати модуль створення скріншотів
- Витягнути активний елемент

4.3.1. Налаштування UiAutomation бібліотеки

UiAutomation бібліотеку можна підключити використовуючи стандартний провідник бібліотек в Microsoft Visual Studio. Ця бібліотека постачається разом з .Net Framework.

Для цього потрібно відкрити провідник бібліотек, що зображений на рис.4.5, написати назву бібліотеки та натиснути кнопку додати.

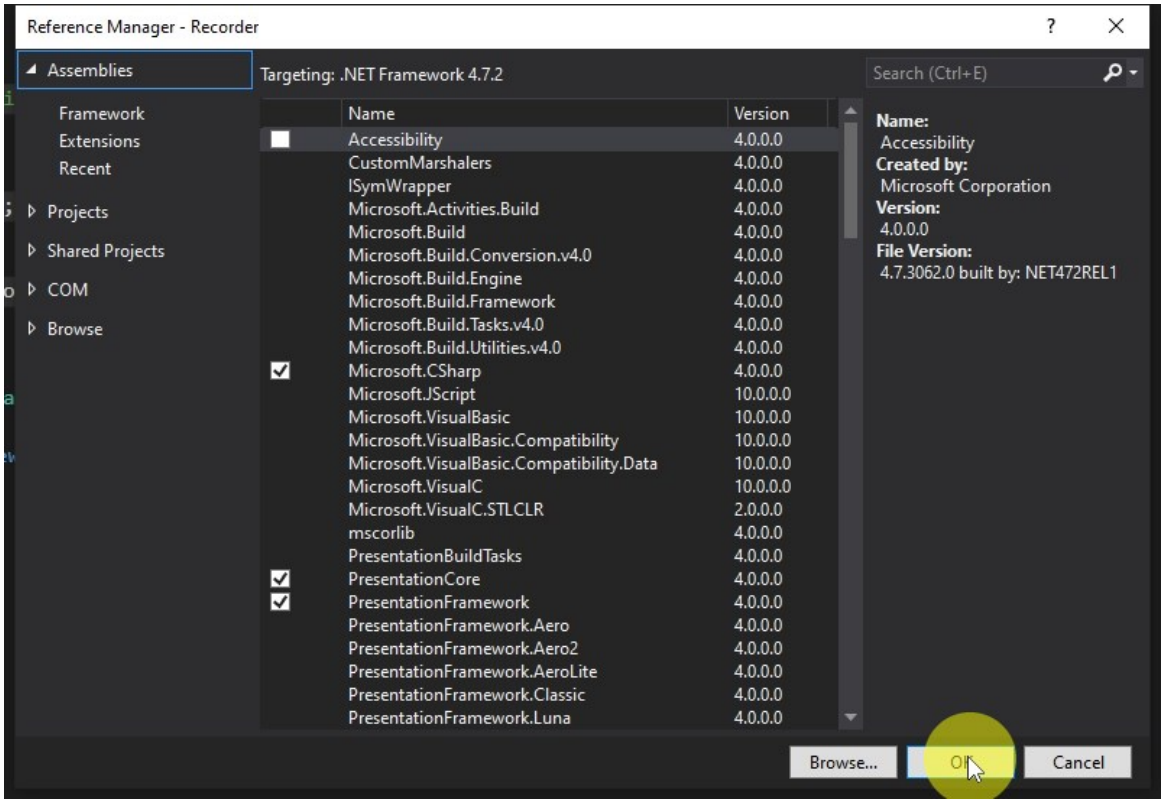


Рис. 4.5. Додавання UiAtomation бібліотек

Основний клас, що міститься в даній бібліотеці і призначений для витягування інформації з графічних компонентів графічних програм є AutomationElement.

У ньому є багато допоміжних методів що дозволяють отримувати інформацію про активні елементи використовуючи координати мишки, або точки на робочому столі. Також він дозволяє проходити по деревах елементів програм та як програма насправді це дерево різник елементів.

Для прикладу основний компонент програми - це вікно, далі на ньому розміщують певні панелі і певному місці для зручного користувацького інтерфейсу і вже на панель додають кнопки, випадаючі списки, поля вводу тексту.

На рис.4.6 зображені усі методи даного класу для роботи з графічними елементами на рівні операційної системи.

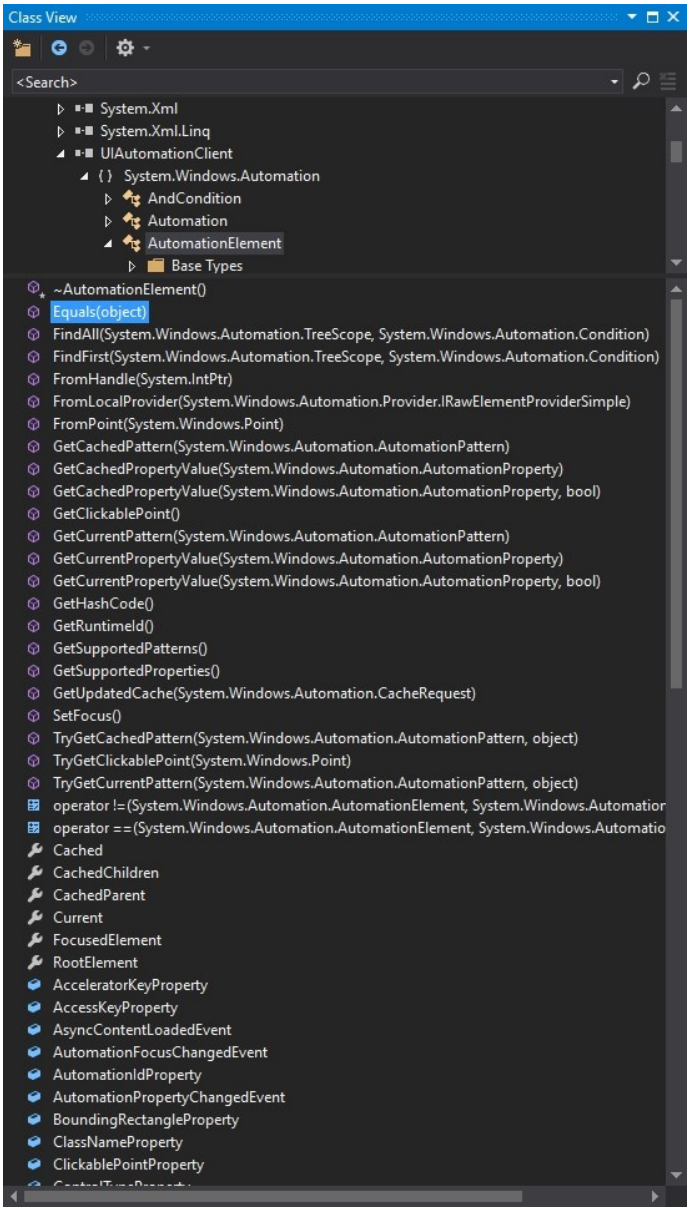


Рис. 4.6. Список компонентів класу Automation Element

Основний метод який використовується в даному рішенні це метод AutomationElement.FromPoint(point).

Він приймає як параметр точку на робочому столі, за нашим припущенням - це точка, де користувач клікнув на елемент, з яким взаємодіє. Та повертає дерево контролерів по яких потрібно пройтись та дослідити на який саме елемент було зроблено клік.

Без посередньо автоматизаційний елемент має такі характеристики:

- Id
- Name

- LocalizedControlType
- IsEnabled
- BoudingRectangle
- Root

Цю інформацію ми використовуємо щоб визначати дію користувача.

4.3.2. Підписання на низько-рівневі події

У більшості графічних програм чи фреймворків є реалізовано можливість підписуватись на події що зроблені безпосередньо над програмою. У нашому випадку потрібно підписуватись на усі кліки що відбуваються на машині користувача.

Для цього потрібно використовувати технологію взаємодії .Net Framework + низько-рівневі бібліотеки що є частиною операційної системи.

Для роботи з операційною системою ми використовуємо бібліотеку user32.dll. Для того щоб використати її в мові програмування високого рівня C# використовує допоміжний механізм атрибутів.

DllImport – атрибут що дозволяє використовувати та імпортувати низько-рівневі бібліотеки.

Для підписання на низько-рівневі події і подальшої взаємодії ми використовуємо наступні функції:

- SetWindowsHookEx(int idHook, HookProc lpfn, IntPtr hMod, uint dwThreadId);
- CallNextHookEx(IntPtr hhk, int nCode, IntPtr wParam, IntPtr lParam);
- UnhookWindowsHookEx(IntPtr hhk);
- ClientToScreen(IntPtr hWnd, ref POINT lpPoint);
- GetWindowRect(IntPtr hWnd, out RECT lpRect);
- GetWindowThreadProcessId(IntPtr hWnd, out uint lpdwProcessId);
- GetDesktopWindow();

Для безпечної роботи додатка потрібно також подбати про наступних виконавців хто підписані на ці події, тому ми не просто підписуємося але й після виконання дії передаємо дані наступному обробнику подій, тобто наступній програмі за допомогою функції CallNextHookEx.

4.4. Створення веб додатка для авторизації

Для забезпечення авторизації на стороні сервера потрібно виконати наступні дії:

- Створити сховище даних
- Використати JWT як підхід авторизації
- Інтегрувати логіку авторизації на сервері
- Створити Web API для авторизації з клієнта

Для початку потрібно створити новий проект для роботи з веб додатком, у вікні контролю рішення клікнемо на правою кнопкою на заголовку рішення та виберемо пункт додати проект.

У вікні вибору типу проекту пишемо Web, та вибираємо .Net Core як основний фреймворк для розробки. Усі налаштування можна побачити на рис.4.7.

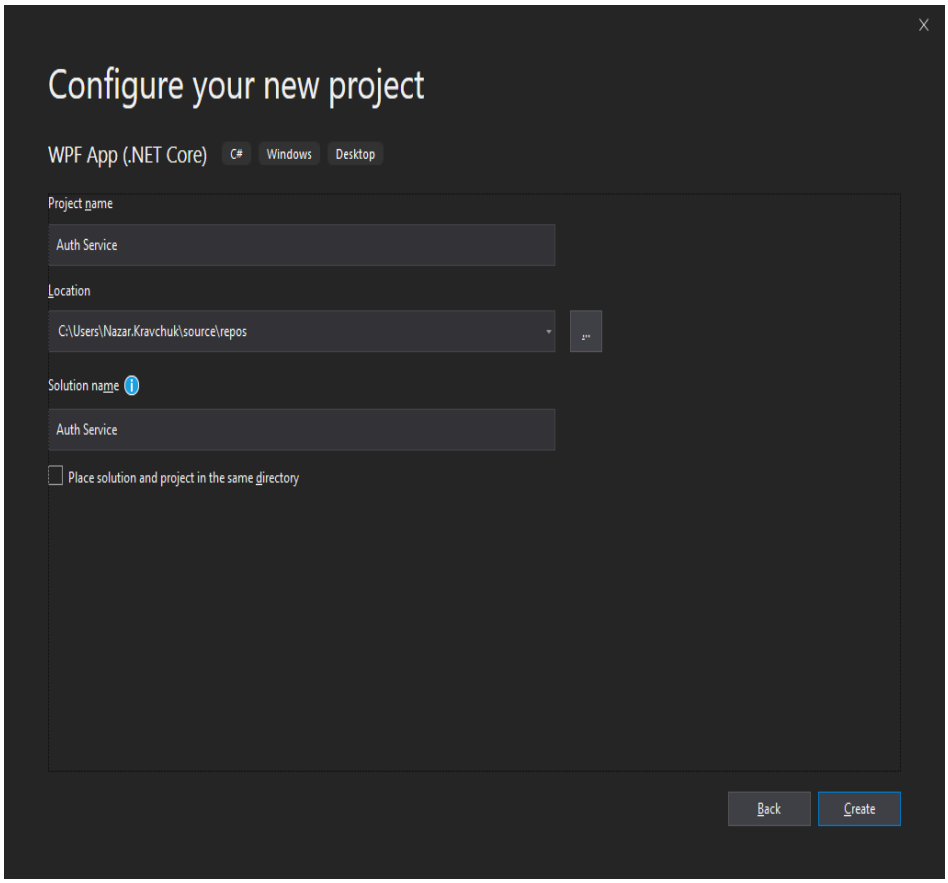


Рис. 4.7. Створення проекту для авторизації

Для авторизації ми використовуємо допоміжну бібліотеку – OpenIdDict.

Щоб підключити її до проекту потрібно використати додатковий пакет програмного забезпечення. У .Net Core для цього є спеціальний інструмент Nuget Package Manager. На рис.4.8 зображено назву пакета та вигляд менеджера зовнішніх програмних пакетів.

Для того щоб підключити авторизацію безпосередньо в програмне рішення потрібно використати методи розширення на конфігураційній частині веб додатка.

Дана платформа дозоляє опрацьовувати запит поступово пропускаючи його через певний конвеєр, та дозоляє вклинюватись в нього додаючи або видаляючи певні стадії опрацювання запитів.

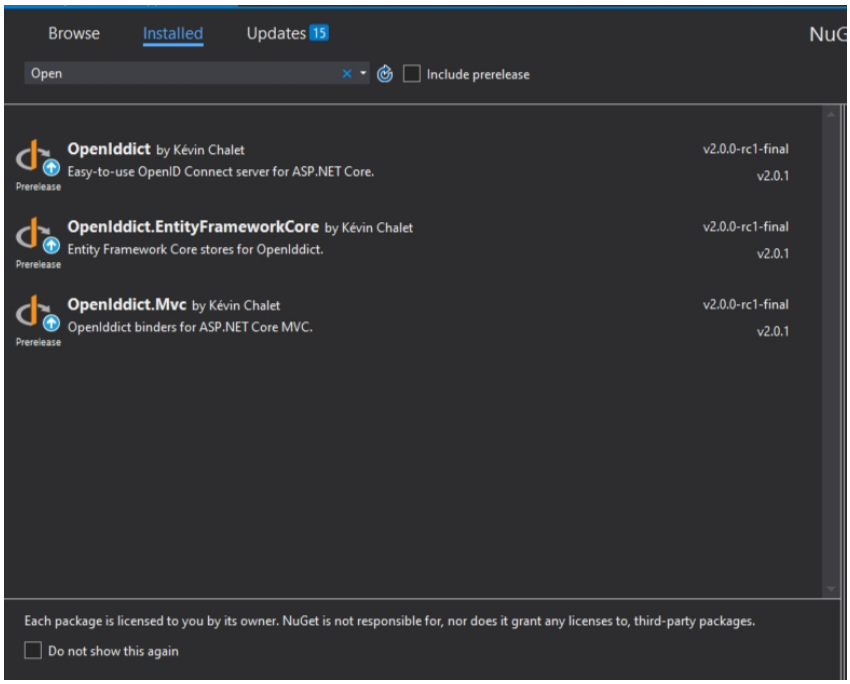


Рис. 4.8. Open Id dict пакети

Кусок програмного коду що зображений на рис.4.9 демонструє як саме потрібно підключити та інтегрувати OpenIdDict в конвеєр опрацювання запиту.

```
var arrayKey = System.Text.Encoding.UTF8.GetBytes(Configuration.GetSection("AppSettings")["EncryptSecretJWTKey"]);

services.AddOpenIddict(options =>
{
    // Register the Entity Framework stores.
    options.AddEntityFrameworkCoreStores<ApplicationDbContext>();

    options.AddMvcBinders();

    // Enable the token endpoint.
    options.EnableTokenEndpoint("/connect/token");

    // Enable the password flow.
    options.AllowPasswordFlow()
           .AllowRefreshTokenFlow()
           .AllowCustomFlow("google_oauth")
           .AllowCustomFlow("google_token");

    options.DisableHttpsRequirement();

    options.UseJsonWebTokens();

    options.AddSigningKey(new SymmetricSecurityKey(arrayKey));

    options.SetAccessTokenLifetime(TimeSpan.FromHours(12));

    options.AddEncryptingKey(new SymmetricSecurityKey(arrayKey));
});
```

Рис. 4.9. Конфігурація авторизації

Також для збереження інформації про користувачів нам потрібна база даних. Дана бібліотека підтримує технологію Entity Framework та ADO NET

що спрощує процес налаштування бази даних, та це вимагає використання MS SQL server для збереження даних.

Авторизація користувачів потребує певної схеми даних в базі даних, а саме вона зображена на рис.4.10. і також OpenIdDict використовує схему що зображена на рис.4.11.

Основні таблиці в схемі це:

- AspNetUser
- AspNetRole
- AspNetUserToken
- AspNetUserLogin
- AspNetUserClaim
- AspNetRoleClaim

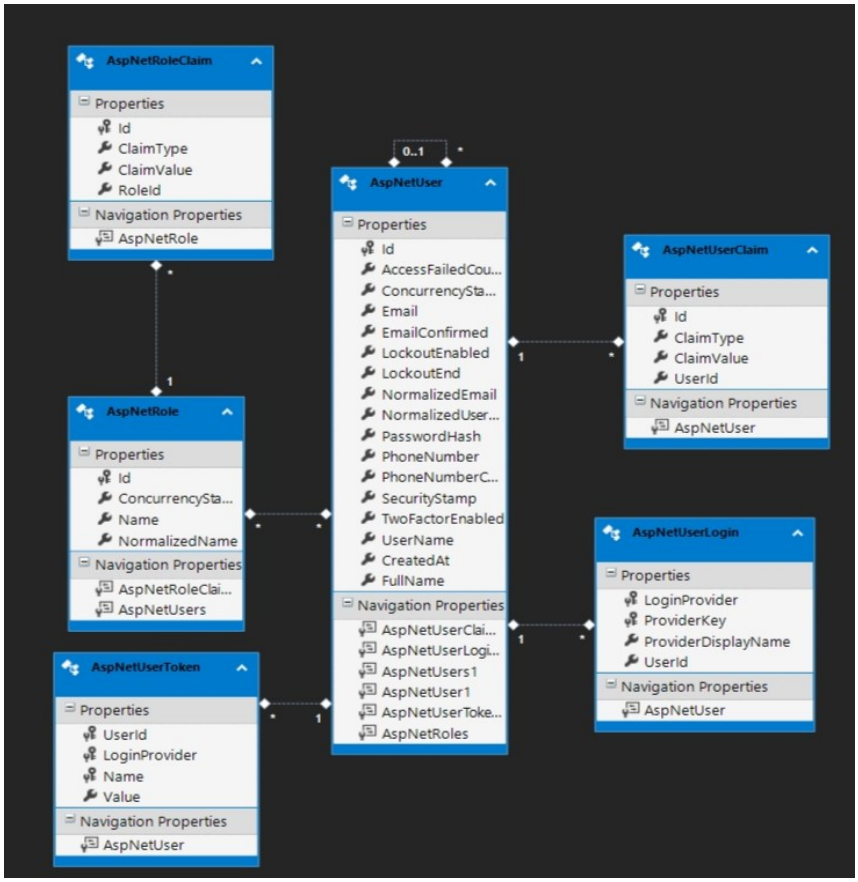


Рис. 4.10. Схема бази даних для авторизації

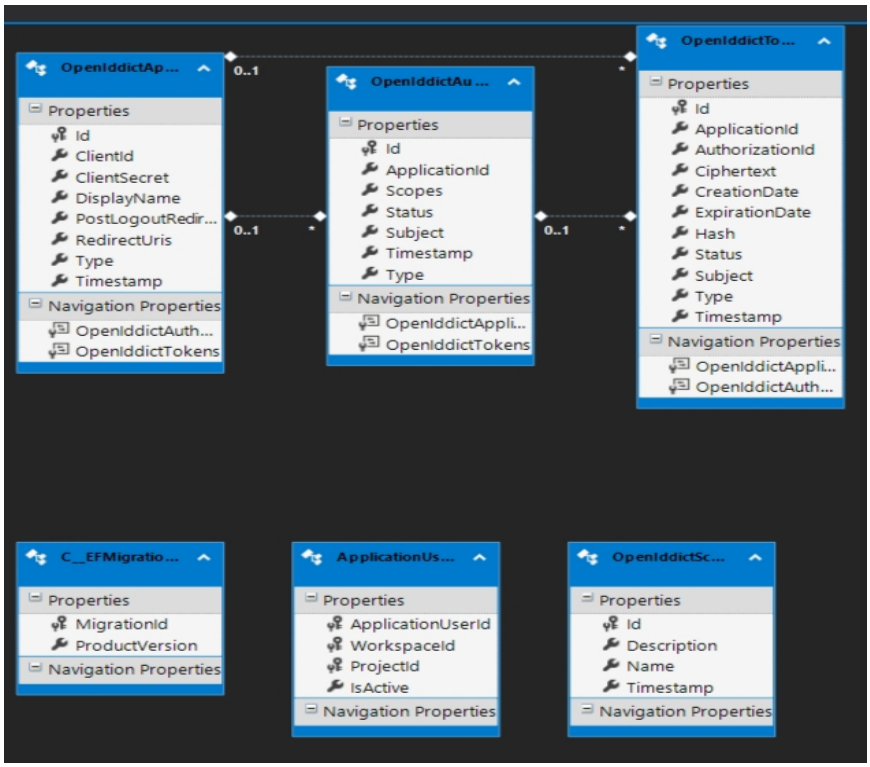


Рис. 4.11. Схема бази даних OpenIdDict

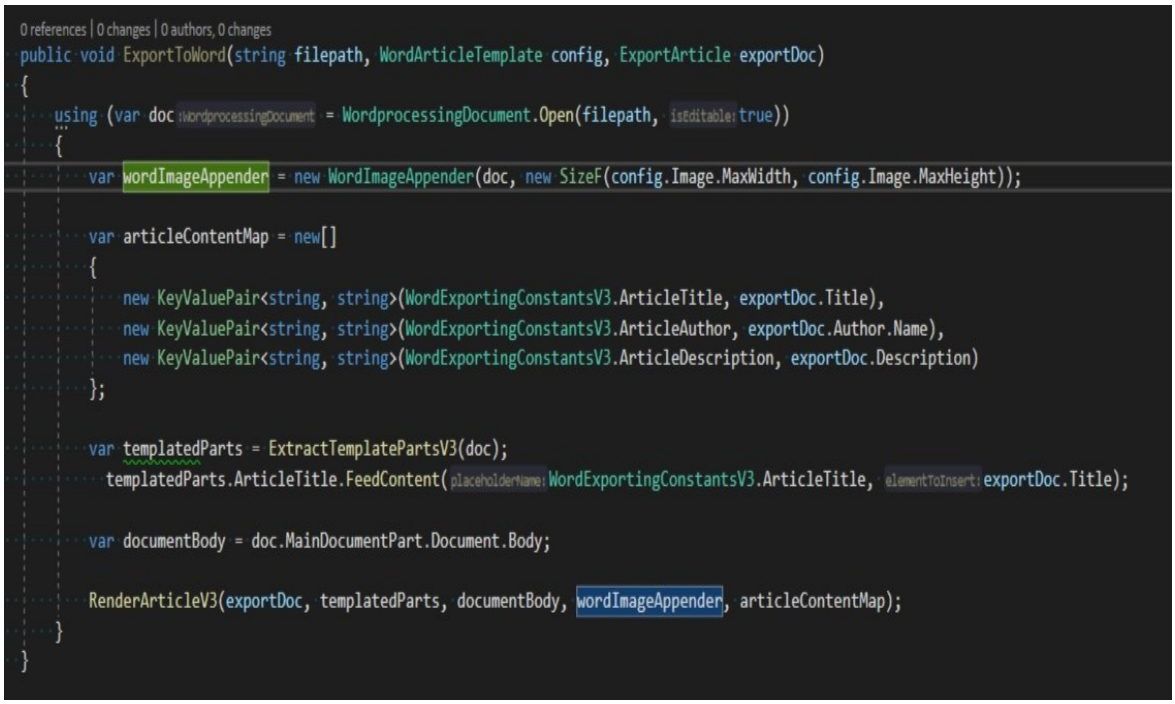
4.5. Формат файлу Microsoft Word

Для роботи з файлами Microsoft Word є ряд додаткових бібліотек. Основна логіка зберігання документу це є ZIP архів що містить всередині xml файл який зберігає форматування, у вигляді параграфів та стилів і безпосередньо контенту. Також є спеціальні папки для збереження допоміжних файлів як зображення.

Безпосередньо для роботи з форматованим XML документом, ми використовуємо наступні бібліотеки та простори імен:

- DocumentFormat.OpenXml
- DocumentFormat.OpenXml.Packaging
- DocumentFormat.OpenXml.Wordprocessing

Безпосередня логіка створення Word документа зображена на рис.4.12.



```
0 references | 0 changes | 0 authors, 0 changes
public void ExportToWord(string filepath, WordArticleTemplate config, ExportArticle exportDoc)
{
    using (var doc WordprocessingDocument = WordprocessingDocument.Open(filepath, isEditable: true))
    {
        var wordImageAppender = new WordImageAppender(doc, new SizeF(config.Image.MaxWidth, config.Image.MaxHeight));

        var articleContentMap = new[]
        {
            new KeyValuePair<string, string>(WordExportingConstantsV3.ArticleTitle, exportDoc.Title),
            new KeyValuePair<string, string>(WordExportingConstantsV3.ArticleAuthor, exportDoc.Author.Name),
            new KeyValuePair<string, string>(WordExportingConstantsV3.ArticleDescription, exportDoc.Description)
        };

        var templatedParts = ExtractTemplatePartsV3(doc);
        templatedParts.ArticleTitle.FeedContent(placeholderName: WordExportingConstantsV3.ArticleTitle, elementToInsert: exportDoc.Title);

        var documentBody = doc.MainDocumentPart.Document.Body;

        RenderArticleV3(exportDoc, templatedParts, documentBody, wordImageAppender, articleContentMap);
    }
}
```

Рис. 4.12. Логіка створення Word документа

4.6. Програмний інтерфейс

Та як основне завдання програми - це робота безпосередньо без інтерфейсу у фоновому режимі, програма містить два основні вікна, а саме: вікно авторизації, що зображено на рис.4.13 та вікно контролю запису та відображення записаних даних перед експортом (рис.4.14).

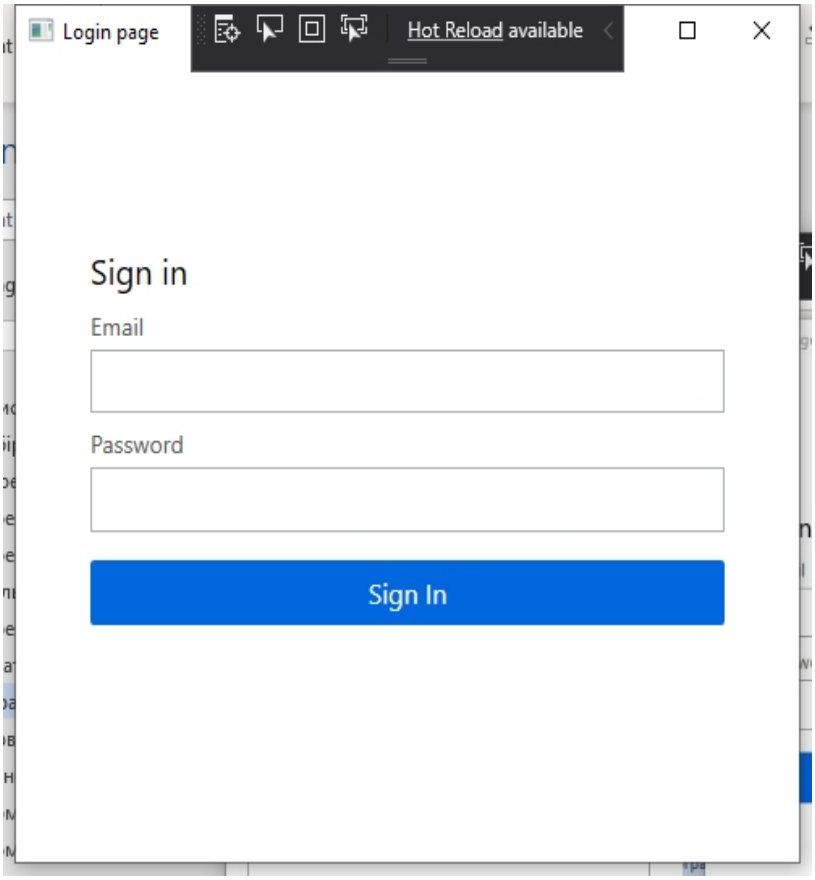


Рис. 4.13. Вікно авторизації

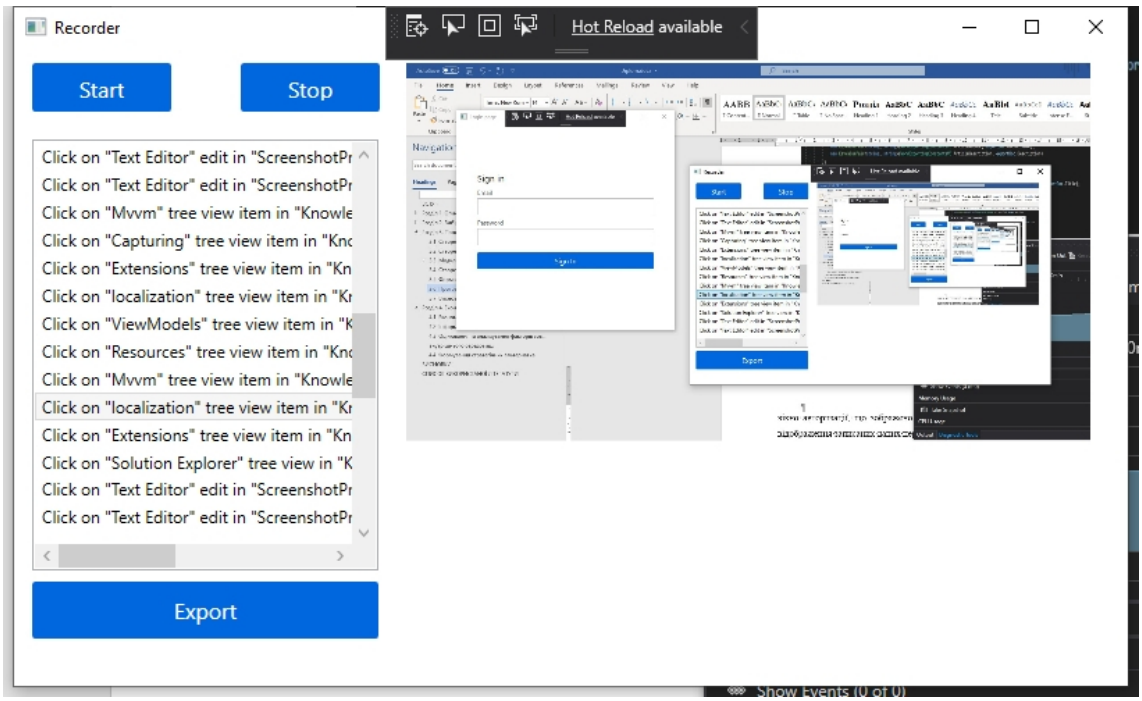


Рис. 4.14. Головне вікно програми

4.7. Висновки до розділу

В даному розділі було розглянуто поступові кроки реалізації даного проекту, а саме:

- Налагодження системи контролю версій – для відстеження змін в проекті та безпечній розробці без можливості втрати коду.
- Створення файлу рішення з використанням Microsoft Visual Studio.
- Безпосередній розгляд настільного модуля з аналізом основних частин з точки зору програмної реалізації.
- Реалізація веб-додатка для авторизації
- Вигляд формату Microsoft Word з середини – трансформація в даний формат.

Розділ 5. Розроблення стартапу-проекту

5.1. Опис ідеї проекту

Питання документації постає дуже часто практично у всіх відомих сферах життя людей. Дуже часто потрібно задокументувати певні процеси або кроки, які потрібно зробити щоб досягнути успіху.

Коли людина самостійно працює над певним процесом і виконує свою роботу, вона генерує якісь нові алгоритми і кроки щоб виконати свою роботу швидше і краще. Та для того щоб інші співробітники чи особи, що задіяні в схожій сфері теж мали змогу оцінити та використати ці кроки, перша людина повинна задокументувати та зберегти свої кроки.

Та для цього людина з досвідом повинна, використовуючи складні інструменти та витрачаючи додаткові години, створити детальну покрокову документацію. Створюючи окремо скріншоти та детально описуючи кожен крок, навіть коли вони доволі прості, але їх кількість дуже велика і вимагає проведення так званої, рутинної роботи.

Тут у пригоді явно стає програма, яка з легкістю дозволить практично не відволікаючи людину від основної роботи, захопити його основні дії, отримати результат та з економити години робочого часу.

Під час дослідження ринку документування мені вдалось знайти потужні інструменти документування та більшість з них вимагають використовувати довгі та рутинні методи для створення документації.

Також є декілька дуже простих інструментів які дозволяють працювати з діями користувача та це практично захоплення поодиноких дії без змоги експорту в інші формати даних, серед них:

- SwipeGuide;
- Snagit;
- ScreenSteps;
- CloudApp.

Виходячи з аналізованої вище інформації можна сказати, що розробка подібної системи є надзвичайно актуальною в наш час, оскільки на ринку

немає її аналогів. Основними споживачами цього програмного продукту є навчальні заклади, інтернет компанії, стартапи, що займаються накопичуванням та поширенням своїх знання.

5.2. Розроблення ринкової стратегії

Для поточного програмного рішення у сфері стратегічних альтернатив на мою думну варто і доцільно обрати стратегію розвитку продукту, та як робота відбувається над новим рішенням, що ще немає аналогів, проте вже утворений сегмент на світовому ринку, де наявні продукти схожої категорії з схожими характеристики.

За результатами мого дослідження впливу факторів зовнішнього та внутрішнього середовища можна прийти до висновку, що у зовнішньому середовищі найбільш позитивним є вплив науково-технічного прогресу, рівню техніки та технологій. Негативно впливають законодавчі акти та стан економіки.

У внутрішньому середовищі найбільш розвиненими є технології.

Отже, сформувалася потреба у продукті і є можливості для виходу на ринок описаної системи.

Витрати на виробництво продукції у вартісному виразі формують її виробничу собівартість. Цей показник є одним з найважливіших економічних показників господарської діяльності підприємства, у якому дістають відображення зростання продуктивності праці, економія ресурсів, технічний прогрес.

Стратегія розвитку продукту полягає у створенні нового продукту (програмного забезпечення) для існуючого сегменту ринку. Ця стратегія є досить ризиковою, оскільки вимагає створення нового продукту (програмного забезпечення) для існуючого сегменту споживачів. Однак, якщо ринок починає зменшувати обсяги та існуючий продукт є на етапі

зрілості та падіння, тоді доцільно застосовувати стратегію розвитку продукту.

Стратегія диверсифікації реалізується шляхом виходу на нові сфери бізнесу. Тобто розширення номенклатури товарів, послуг тощо.

Бюджетування є комплексно обґрунтованою системою розрахунку витрат, пов'язаних з виготовленням та реалізацією продукту, яка дає можливість здійснити аналіз витрат та розробити заходи щодо підвищення рентабельності виробництва. На даному етапі необхідно визначити собівартість продукту, який розробляється та економічно обґрунтувати доцільність вибору однієї із стратегій.

У розробці програми беруть участь програміст та адміністратор. Програміст потрібен для розробки програми, а адміністратор для подальшої підтримки та нарощування ринків збуту.

Витрати на оплату праці включають заробітну плату всіх працівників зайнятих на всіх етапах проектування. Розміри заробітної плати обчислюються на основі трудоємкості відповідних робіт у людино-днях та середньої заробітної плати відповідних категорій працівників. Витрати на оплату праці складаються:

- місячний оклад програміста становить 15000 грн./міс.

- місячний оклад адміністратора становить 8000 грн./міс.

Так як норма праці це 22 робочих дні і візьмемо середню зарплату для програміста 15000 тис. грн., а для адміністратора 8000 тис. грн. Порахувавши денну зарплату кожного виходить що зарплата програміста на день складає

С = (15000/22) = 682 грн , відповідно адміністратора

С = (8000/22) 363,6 грн.

Сумарні витрати на оплату праці складатимуть:

СВОП = 15000 + 8000 = 23000 грн.

Бюджет витрат на оплату праці наведено в табл. 5.1.

Таблиця 5.1

Бюджет витрат на оплату праці

Посада, спеціальність	**Кількість працівників, осіб**	**Час роботи, дні**	**Денна заробітна плата працівників, грн.**	**Сума витрат на оплату праці, грн.**
Основна заробітна плата				
Програміст	1	22	681,8	15000
Адміністратор	1	22	363,6	8000
Разом:	2	44	1054,4	23000

На заробітну плату нараховується єдиний внесок на соціальне страхування, який відповідає чинній ставці на момент виконання бакалаврської роботи, а саме 22%. Податок на доходи фізичних осіб становить 18%, військовий збір 1.5%. Бюджет обов'язкових відрахувань та податків наведено в табл. 4.5.

Крім прямих виробничих витрат можуть виникати загальновиробничі витрати, які не можна напряму віднести до конкретного виду продукції. За ступенем залежності від обсягу виробництва виділяють змінні та постійні загальновиробничі витрати.

5.3. Розроблення маркетингової програми

Головною метою даної магістерської кваліфікаційної роботи є розробка системи для автоматизація документування процесів та форматування звітів.

Основними вимогами під час розробки є:

- Зручність користування
- Можливість захоплення дій користувача
- Можливість редагування захопленого контенту
- Можливість перетворення даних у формат Microsoft Word
- Авторизація та менеджмент користувачів

Дане програмне рішення – це хороша можливість зробити цінний продукт з точки зори користування, що дозволить користувачам, економити години робочого часу для документування, створюючи покрокові інструкції базуючись на діях користувача.

У свою чергу програма потребує не значні затрати на розробку першої версії продукту, більшість коштів доцільно направити на маркетингові активності, для охоплення нових програмних ринків, та купівлю реклами для її конвертування в активних користувачів.

З економічної точки зору даний проект є економічно вигідним, оскільки практично усі сфери де задіяні комп'ютери потребують створення документації, накопичення нових методів вирішення проблем та передачу знань наступним поколінням.

5.4. Вимоги до технічного та програмного забезпечення

Для користувачів даної системи – інформаційна система повинна представляти зручний інструмент у вигляді настільної програми що працює в двох режимах:

- З графічним інтерфейсом – час коли користувач запускає, зупиняє процес захоплення дій на комп'ютері під час виконання певного завдання.
- Як фонова програма – мається на увазі, момент коли користувач запустив процес захоплення своїх дій, і виконує певну задачу, тоді програма згортає основний інтерфейс, і працює як фоновий сервіс операційної системи з мінімальним інтерфейсом.

Також для того щоб програму могли використовувати лише авторизовані користувачі, система повинна мати змогу підтримувати хороші методи авторизації.

Для цього додатково до настільної програми потрібно забезпечити зовнішню веб-систему що буде контролювати доступ до програми та зберігати данні про користувачів. Основні вимоги до веб частини інформаційної системи:

- Безпека
- Швидка відповідь сервера
- Можливість зберігати дані про користувача

- Можливість зберігати мета дані
- Авторизаційний функціонал з використанням JWT – сучасного методу авторизації запитів до серверів.
- Доступність
- Використання HTTPS протоколів для роботи користувачів

Доступ до створюваної ІС зовнішніми користувачами повинен отримуватись з автоматизованих робочих місць на базі персональних комп'ютерів, ноутбуків або КПК, підключених до мережі Інтернет.

Для роботи з ІС зовнішнім користувачам необхідно мати встановлений та налаштований один із сучасних веб-браузерів.

Користувачем ІС є будь-яка авторизована особа. Аутентифікація користувачів відбувається із застосуванням бібліотек криптографічного перетворення.

ІС розробляється по модульному принципу, тобто система складається з окремих модулів, кожен з яких реалізує певний набір функцій, притаманних виключно йому:

- Модуль авторизації
- Модуль захоплення даних
- Модуль перегляду даних
- Модуль перетворення даних у Microsoft Word формат

Створюване програмне забезпечення ІС повинно відповідати наступним вимогам:

- Відповідність функціоналу інтерфейсу ІС сучасним вимогам (інтерфейс повинен мати звичний для користувача вигляд та набір команд тощо);
- Адаптивний дизайн;
- Підтримка сервісно-орієнтованої архітектури (SOA) (доступ через Web Services до бібліотеки функціональних (високорівневих) компонент системи);

- Підтримувати можливість обміну даними через інтерфейси взаємодії.
- Можливість захоплення дій без складної взаємодії з системою
- Перетворення захопленого контенту в формат Microsoft Word
- Підтримувати шаблонів для різних стандартів

5.5. Висновки до розділу

В цьому розділі ми розглянули основну ідею стартам проекту, дослідили потенційний вибір ринкової та маркетингової стратегій. Визначили вимоги до технічного та програмного забезпечення.

На ринку на сьогоднішній день немає програмних продуктів, які можна використовувати у фоновому режимі для документування, тому логічним є вибір стратегії розроблення нового продукту і його подальший розвиток, оскільки є вже існуючий цільовий сегмент ринку.

ВИСНОВКИ

В процесі виконання дипломної роботи було розроблено автоматизовану інформаційну систему для автоматизації документування процесів та формування звітів.

Дана система розроблена і повністю відповідає поставленим вимогам.

Для користувачів даної системи – інформаційна система представляє зручний інструмент у вигляді настільної програми що працює в двох режимах:

- З графічним інтерфейсом – час коли користувач запускає, зупиняє процес захоплення дій на комп'ютері під час виконання певного завдання.
- Як фонова програма – мається на увазі, момент коли користувач запустив процес захоплення своїх дій, і виконує певну задачу, тоді програма згортає основний інтерфейс, і працює як фоновий сервіс операційної системи з мінімальним інтерфейсом.

Також для того, щоб програму могли використовувати лише авторизовані користувачі, система підтримує надійний метод авторизації з використанням JWT.

Дана тема являється актуальною у наш час та як питання документації постає дуже часто практично у всіх відомих сферах життя людей. Дуже часто потрібно задокументувати певні процеси або кроки, які потрібно зробити щоб досягнути успіху. Для цього використовуючи складні інструменти та витрачаючи додаткові години для написання детальної документації. Створюючи окремо скріншоти та детально описуючи кожен крок, навіть коли вони доволі прості, але їх кількість дуже велика і вимагає проведення так званої рутинної роботи.

Об'єктом дослідження даної роботи була настільна система для захоплення дій користувача та перетворення їх у документаційний файл.

Предмет дослідження – програмна реалізація системи у вигляді декількох складових:

- Модуль авторизації
- Модуль захоплення даних
- Модуль перегляду даних
- Модуль перетворення даних у Microsoft Word формат

Використання мови програмування C#, технології розробки користувацького інтерфейсу WPF виявилося доцільним та раціональним. Завдяки цим інструментам час розробки програми зменшився, а якість та вигляд програмного продукту значно покращилися.

Програма протестована і готова успішно використовуватись для захоплення дій користувача та формування результатів у вигляді файлів формату Microsoft Word.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Кравчук Н., Мокрицька О.В., Інформаційна система автоматизованого документування стартапів, Комп'ютерне моделювання та інформаційні технології: матеріали третьої науково-практичної конференції студентів, аспірантів та молодих вчених (Львів, 14-16 жовтня 2021 р.). – Львів: кафедра інформаційних технологій НЛТУ України, 2021. – С. 33-35
2. Коноваленко І.В. Програмування мовою C# - Тернопіль, 2016. - 300 с.
3. Береза А.М. Основи створення інформаційних систем. Навчальний посібник. 2-ге видання, 2001. – 156 с.
4. Тарасевич В. М. Економічна теорія , 2006. - 107 с.
5. Кравець П.О. Об`єктно-орієнтоване програмування. – Видавництво Львівської політехніки, 2012. – 200 с.
6. Вовчак І. Інформаційні системи та комп'ютерні технології в менеджменті. - Тернопіль: Карт-бланш, 2001. - 286 с.
7. Гончарук А. Г. Формування механізму управління ефективністю підприємства — Одеса, 2010. — 120 с.
8. Інформаційні системи і технології в обліку й аудиті – 2016. – 124 с.
9. Daniel Solis. Illustrated C# 2012. - 320 с.
10. Шилд, Герберт. Довідник по C#, 2007. – 50 с.
11. Кузнєцов М.С. Об'єктно-орієнтоване програмування з використанням UML та мови C++: Навч. посібник. – Дніпропетровськ: 2003. – 90 с.
12. Rudolf Pecinovsky. OOP: Learn Object Oriented Thinking and Programming / Rudolf Pecinovsky – Eva & Tomas Bruckner Publishing , 2013. – 527 с.
13. Ситник В.Ф. Основи інформаційних систем, 2001.- 220 с.
14. А. Хейлсберг, С. Вилтамут, П. Голд Язык программирования C#. Классика Computers Science. 4-е издание = C# Programming Language

(Covering C# 4.0), 4th Ed. — СПб.: «Питер», 2012. — 784 с.

15. Standard ECMA-334. C# Language Specification. 4th Edition. Ecma International. – June 2006
16. International standart ISO/IEC 23270:2006. Information technology – Programming languages – C#. Second edition. – ISO/IEC. - 2006
17. Matthew MacDonald, Pro WPF in C# 2010: Windows Presentation Foundation in .NET 4, 2010. – 217 с.
18. Адам Натан WPF 4 Подробное руководство, 2012. – 245с.

ДОДАТОК А. КОД РОЗРОБКИ

Login.xaml

```
<Window x:Class="Recorder.Login"
        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:Recorder"
        mc:Ignorable="d" Width="500"
        Title="Login page"   >
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" MinWidth="20" />
            <ColumnDefinition Width="Auto" MinWidth="400" />
            <ColumnDefinition Width="*" MinWidth="20" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="*" MinHeight="20" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" MinHeight="20" />
        </Grid.RowDefinitions>
        <Grid Grid.Row="1" Grid.Column="1">
            <StackPanel Visibility="Visible" MaxWidth="400">
                <TextBlock Text="Sign in" Style="{StaticResource TitleTextBlock}"/>
                <TextBlock Text="Email" Style="{StaticResource LabelTextBlock}"/>
                <TextBox Text="{Binding Email, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"
TextChanged="TextBoxBase_OnTextChanged"/>
                <TextBlock Text="Password" Style="{StaticResource LabelTextBlock}"/>
```

```
            <TextBox Text="{Binding Email, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"
TextChanged="TextBoxBase_OnTextChanged"/>

            <Button  Command="{Binding SignInCommand}"

                Content="Sign In" Click="Button_Click"/>

            <TextBlock Style="{StaticResource ErrorLabelTextBlock}"
LineStackingStrategy="BlockLineHeight"

                Visibility="Visible"

                Text="{Binding ErrorMessage}" TextWrapping="Wrap"
TextTrimming="WordEllipsis" MaxHeight="60"/>

        </StackPanel>

    </Grid>

  </Grid>

</Window>
```

MainWindow.xaml

```
<Window x:Class="Recorder.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:Recorder"

    mc:Ignorable="d"

    Title="Recorder" Height="477" Width="803">

  <Grid>

    <Grid>

        <Button Content="Start" HorizontalAlignment="Left" Height="35"
Margin="13,10,0,0" VerticalAlignment="Top" Width="100"
Click="ButtonBase_OnClick"/>
```

```
        <Button Content="Stop" HorizontalAlignment="Left" Height="35"
Margin="163,10,0,0" VerticalAlignment="Top" Width="100"
Click="ButtonBase_OnClick1"/>

        <Image HorizontalAlignment="Left" Height="404" Margin="282,10,0,0"
VerticalAlignment="Top" Width="492" Source="{Binding Image}"/>

        <ListBox HorizontalAlignment="Left" Height="310" Margin="13,65,0,0"
VerticalAlignment="Top" Width="249" ItemsSource="{Binding Steps}"/>

        <Button Content="Export" HorizontalAlignment="Left" Height="41"
Margin="13,383,0,0" VerticalAlignment="Top" Width="249"/>


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

App.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Threading.Tasks;
using System.Windows;
namespace Recorder
{
   /// <summary>
   /// Interaction logic for App.xaml
   /// </summary>
   public partial class App : Application
   {
      protected override void OnStartup(StartupEventArgs e)
```

```
        {
            var loginView = new Login();
            loginView.Show();
            var mainWindow = new MainWindow();
            mainWindow.Show();
            base.OnStartup(e);
        }
    }
}
```

MainWindowViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Drawing.Imaging;
using System.IO;
using System.Runtime.CompilerServices;
using System.Windows.Documents;
using System.Windows.Media;
using Knowledge_Supplier.Extensions;
using PD.Capturing;
using PD.Capturing.Dom;
using PD.Common.Localization;
using Recorder.Extensions;

namespace KnowledgeSupplier.ViewModels
{
```

```
public class MainWindowViewModel : INotifyPropertyChanged
{
    private Screenshot _lastScreenshot;
    private LocalizationService _localizationService;
    public ObservableCollection<string> Steps { get; set; }
    public string Image { get; set; }
    public ScreenshotRecorder ScreenshotRecorder { get; private set; }
    public MainWindowViewModel()
    {
        Steps = new ObservableCollection<string>(new[].{ "One", "Second", "Throuth"
});
        ScreenshotRecorder = new ScreenshotRecorder(new ScreenshotProvider());
        _localizationService = new LocalizationService();
        ScreenshotRecorder.Captured += ScreenshotRecorderOnCaptured;
        ScreenshotRecorder.StartRecording(CaptureType.ActiveWindow, new
RecordingConfig(new ScreenshotConfig()));
    }
    private void ScreenshotRecorderOnCaptured(object sender, ScreenshotEventArgs
e)
    {
        try
        {
            ProcessScreenshot(e.Screeshot);
        }
        catch (Exception ex)
        {
            PDLog.Error(ex);
        }
```

```
            OnPropertyChanged(nameof(Steps));
        }
        private void ProcessScreenshot(Screenshot screenshot)
        {
            PDLog.Debug("RecordingWindow::ProcessScreenshot");
            var imageFormat = ImageFormat.Jpeg;
            var fileName = Guid.NewGuid().ToString() + ".jpg";
            fileName = Path.Combine(Directory.GetCurrentDirectory(), fileName);
            if (screenshot == null)
                return;
            try
            {
                FileExtensions.Save(fileName, (ImageSource)screenshot.ImageSource,
imageFormat);


                _lastScreenshot = new Screenshot(screenshot.Rect, fileName, imageFormat,
screenshot.Actions)
                {
                    //    PartOfSmartCapturing = screenshot.PartOfSmartCapturing
                    PartOfSmartCapturing = false
                };
                var metadata = GetMetadataToLastScreenshot();
                Steps.Add(metadata);
                Image = fileName;
                OnPropertyChanged(nameof(Image));
                PDLog.Debug("RecordingWindow::ProcessScreenshot:: Added screenshot to
the list.");
            }
            catch (Exception ex)
```

```
            {
                PDLog.Error(ex, "RecordingWindow:: failed to save screenshot. adding an empty one.");
            }
        }
        private string GetMetadataToLastScreenshot()
        {
            var gRes = _localizationService.GetRecording("en");
            return WinDomHumanizer.LocalizeScreenshotActions(_lastScreenshot.Actions, gRes) ?? string.Empty;
        }
        public event PropertyChangedEventHandler PropertyChanged;
        protected virtual void OnPropertyChanged([CallerMemberName].string propertyName = null)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

BaseCommand.cs

```
using System;
using System.Windows.Input;

namespace Knowledge_Supplier.Mvvm.Commands
{
    public abstract class BaseCommand : ICommand
    {
        public abstract bool CanExecute(object parameter);
```

```
        public abstract void Execute(object parameter);
        public virtual event EventHandler CanExecuteChanged;
        public virtual void RaiseCanExecuteChanged()
        {
            CanExecuteChanged?.Invoke(this, EventArgs.Empty);
        }
    }
    public abstract class BaseCommand<TParameter> : BaseCommand
    {
        public abstract bool CanExecute(TParameter parameter);
        public override bool CanExecute(object parameter)
        {
            var cast = parameter is TParameter arg ? arg : default;
            return CanExecute(cast);
        }
        public abstract void Execute(TParameter path);
        public override void Execute(object parameter)
        {
            var cast = parameter is TParameter arg ? arg : default;
            Execute(cast);
        }
    }
}
```

FileExtensions.cs

```
using System;
using System.Drawing.Imaging;
using System.IO;
```

```
using System.Windows.Media;
using System.Windows.Media.Imaging;
using Knowledge_Supplier.Extensions;

namespace Recorder.Extensions
{
   public static class FileExtensions
   {
      public static void Save(string destination, ImageSource imageToSave,
ImageFormat format)
      {
         try
         {
            if (imageToSave.CanFreeze)
               imageToSave.Freeze();
         }
         catch (Exception ex)
         {
            PDLog.Warn(ex);
         }
         try
         {
            using (var memoryStream = new FileStream(destination, FileMode.Create))
            {
               var encoder = ToBitmapEncoder(format);
               encoder.Frames.Add(BitmapFrame.Create((BitmapSource)imageToSave));
               encoder.Save(memoryStream);
            }
```

```
            }
            catch (Exception ex)
            {
                PDLog.Error(ex, "Failed to save image");
                throw;
            }
        }
        public static BitmapEncoder ToBitmapEncoder(ImageFormat format)
        {
            if (Equals(format, ImageFormat.Bmp))
                return new BmpBitmapEncoder();
            if (Equals(format, ImageFormat.Jpeg))
                return new JpegBitmapEncoder();
            if (Equals(format, ImageFormat.Png))
                return new PngBitmapEncoder();

            throw new NotSupportedException("Unsupported image format");
        }
    }
}
```

Extensions.cs

```
using System;
using System.ComponentModel;
using System.Drawing;
using System.Windows;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using Knowledge_Supplier.Capturing;
```

```
using Point = System.Windows.Point;


namespace Knowledge_Supplier.Extensions
{
   public static class Extensions
   {
      public static Point ToWpf(this System.Drawing.Point point)
      {
         return new Point(point.X, point.Y);
      }
      public static void ApplyDefaultPropertyValues(this object self)
      {
         foreach (PropertyDescriptor prop in TypeDescriptor.GetProperties(self))
         {
            DefaultValueAttribute attr =
prop.Attributes[typeof(DefaultValueAttribute)].as DefaultValueAttribute;
            if (attr != null) prop.SetValue(self, attr.Value);
         }
      }
      public static System.Drawing.Rectangle ToWinForms(this Rect rect)
      {
         return new System.Drawing.Rectangle((int)rect.X, (int)rect.Y, (int)rect.Width,
(int)rect.Height);
      }
      public static Bitmap ToBitmap(this BitmapSource source)
      {
         Bitmap btm;
         var width = source.PixelWidth;
         var height = source.PixelHeight;
```

```
    var stride = width * ((source.Format.BitsPerPixel + 7) / 8);
    var bits = new byte[height * stride];
    source.CopyPixels(bits, stride, 0);
    unsafe
    {
        fixed (byte* pB = bits)
        {
            var ptr = new IntPtr(pB);

            btm = new Bitmap(
                width,
                height,
                stride,
                System.Drawing.Imaging.PixelFormat.Format32bppPArgb,
                ptr);
        }
    }
    return btm;
}
public static ImageSource ToImageSource(this Image image)
{
    if (image is Bitmap)
        return ((Bitmap)image).ToBitmapSource();
    using (var bitmap = new Bitmap(image))
        return bitmap.ToBitmapSource();
}

public static BitmapSource ToBitmapSource(this Bitmap bitmap)
```

```
    {
        if (bitmap == null)
            throw new ArgumentNullException("bitmap");


        lock (bitmap)
        {
            var hBitmap = bitmap.GetHbitmap();
            try
            {
                var result =
System.Windows.Interop.Imaging.CreateBitmapSourceFromHBitmap(
                    hBitmap,
                    IntPtr.Zero,
                    Int32Rect.Empty,
                    BitmapSizeOptions.FromEmptyOptions());
                result.Freeze();
                return result;
            }
            catch (Win32Exception ex)
            {
                PDLog.Debug(ex);
            }
            finally
            {
                NativeMethods.DeleteObject(hBitmap);
            }
        }
        return null;
```

```
        }
    }
}
```

NativeMethods.cs

```
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Windows;
using Knowledge_Supplier.Capturing.Constants;
using Point = System.Drawing.Point;

namespace Knowledge_Supplier.Capturing
{
    public static class NativeMethods
    {
        public delegate IntPtr HookProc(int nCode, IntPtr wParam, IntPtr lParam);
        public delegate bool EnumWindowsProc(IntPtr hWnd, IntPtr lParam);
        [DllImport("user32.dll")]
        public static extern IntPtr GetWindowDC(IntPtr hWnd);
        [DllImport("gdi32.dll")]
        public static extern bool DeleteDC(IntPtr hDC);
        [DllImport("gdi32.dll")]
        public static extern IntPtr SelectObject(IntPtr hdc, IntPtr hgdiobj);
        [DllImport("gdi32.dll")]
        public static extern bool BitBlt(IntPtr hdc, int nXDest, int nYDest, int nWidth, int
nHeight, IntPtr hdcSrc, int nXSrc, int nYSrc, System.Drawing.CopyPixelOperation
dwRop);
        [DllImport("user32.dll")]
```

```
        public static extern int ReleaseDC(IntPtr hWnd, IntPtr hDC);

        [DllImport("gdi32.dll")]

        public static extern IntPtr CreateCompatibleBitmap(IntPtr hdc, int nWidth, int
nHeight);

        [DllImport("gdi32.dll")]

        public static extern IntPtr CreateCompatibleDC(IntPtr hdc);

        [DllImport("user32.dll")]

        public static extern IntPtr GetDesktopWindow();

        [DllImport("user32.dll")]

        public static extern bool GetClientRect(IntPtr hWnd, out RECT lpRect);

        public static Rect GetClientRect(IntPtr handle)

        {

            RECT rect;

            GetClientRect(handle, out rect);

            var position = (POINT)rect.Location;

            ClientToScreen(handle, ref position);

            return new Rect((System.Windows.Point)position, rect.Size);

        }

        [DllImport("user32.dll")]

        public static extern uint GetWindowThreadProcessId(IntPtr hWnd, out uint
lpdwProcessId);

        [DllImport("user32.dll")]

        [return: MarshalAs(UnmanagedType.Bool)]

        public static extern bool EnumThreadWindows(uint dwThreadId,
EnumWindowsProc lpfn, IntPtr lParam);



        public static Rect GetWindowRect(IntPtr handle)

        {

            RECT rect;
```

```
        GetWindowRect(handle, out rect);
        return rect;
    }
    [DllImport("user32.dll")]
    [return: MarshalAs(UnmanagedType.Bool)]
    public static extern bool GetWindowRect(IntPtr hWnd, out RECT lpRect);

    [DllImport("user32.dll")]
    public static extern bool ClientToScreen(IntPtr hWnd, ref POINT lpPoint);

    [DllImport("user32.dll")]
    [return: MarshalAs(UnmanagedType.Bool)]
    public static extern bool UnhookWindowsHookEx(IntPtr hhk);
    [DllImport("gdi32.dll")]
    public static extern bool DeleteObject(IntPtr hObject);
    [DllImport("user32.dll")]
    public static extern IntPtr SetWindowsHookEx(int idHook, HookProc lpfn, IntPtr
hMod, uint dwThreadId);
    [DllImport("user32.dll")]
    public static extern IntPtr CallNextHookEx(IntPtr hhk, int nCode, IntPtr wParam,
IntPtr lParam);
    [DllImport("user32.dll")]
    public static extern bool GetIconInfo(IntPtr hIcon, out IconInfo piconinfo);
    [DllImport("user32.dll")]
    public static extern IntPtr GetForegroundWindow();
    [DllImport("kernel32.dll")]
    public static extern IntPtr GetModuleHandle(string lpModuleName);
    public static IntPtr SetHook(int hookType, HookProc hookProc)
    {
```

```
        using var currentProcess = Process.GetCurrentProcess();
        using var currentModule = currentProcess.MainModule;
        return SetWindowsHookEx(hookType, hookProc,
GetModuleHandle(currentModule.ModuleName), 0);
    }
    [DllImport("user32.dll")]
    public static extern IntPtr CopyIcon(IntPtr hIcon);
    [DllImport("user32.dll")]
    public static extern bool GetCursorInfo(out CursorInfo pci);

    public const int GCL_HICONSM = -34;
    public const int GCL_HICON = -14;
    public const int ICON_SMALL = 0;
    public const int ICON_BIG = 1;
    public const int ICON_SMALL2 = 2;
    public const int SC_MINIMIZE = 0xF020;
    public const int HT_CAPTION = 2;
    public const int CURSOR_SHOWING = 1;
    public const int GWL_STYLE = -16;
    public const int DWM_TNP_RECTDESTINATION = 0x1;
    public const int DWM_TNP_RECTSOURCE = 0x2;
    public const int DWM_TNP_OPACITY = 0x4;
    public const int DWM_TNP_VISIBLE = 0x8;
    public const int DWM_TNP_SOURCECLIENTAREAONLY = 0x10;
    public const int WH_KEYBOARD_LL = 13;
    public const int WH_MOUSE_LL = 14;
    public const int ULW_ALPHA = 0x02;
    public const byte AC_SRC_OVER = 0x00;
```

```
        public const byte AC_SRC_ALPHA = 0x01;
    }
}
```

ScreenshotProvider.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Diagnostics.Contracts;
using System.Drawing;
using System.Drawing.Imaging;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Media.Imaging;
using Knowledge_Supplier.Capturing;
using Knowledge_Supplier.Extensions;
//using PD.Common;
using Color = System.Drawing.Color;
using PixelFormat = System.Drawing.Imaging.PixelFormat;
using Point = System.Drawing.Point;
namespace PD.Capturing
{
    public class ScreenshotProvider : IScreenshotProvider
    {
        #region _Public methods_
        public Screenshot CaptureFullscreen(ScreenshotConfig config)
        {
            var bounds = CaptureHelpers.GetScreenBounds();
```

```
  return CaptureRegion(bounds, config);
}
public Screenshot CaptureActiveMonitor(ScreenshotConfig config)
{
   var bounds = CaptureHelpers.GetActiveScreenBounds();
   return CaptureRectangle(bounds, config);
}
public Screenshot CaptureActiveWindow(ScreenshotConfig config)
{
   var handle =
      NativeMethods.GetForegroundWindow();
   return CaptureWindow(handle, config);
}
public Screenshot CaptureWindow(IntPtr handle, ScreenshotConfig config)
{
   if (handle.ToInt32() > 0)
   {
      var rect = config.CaptureClientArea
         ? NativeMethods.GetClientRect(handle)
         : CaptureHelpers.GetWindowRectangle(handle);

      ExtendRectWithChildWindowSizes(handle, ref rect);
      try
      {
         return CaptureRectangle(rect, config);
      }
      finally  {
      }
```

```
    }
    return null;
}
static IEnumerable<IntPtr> EnumerateProcessWindowHandles(int processId)
{
    var handles = new List<IntPtr>();
    foreach (ProcessThread thread in Process.GetProcessById(processId).Threads)
        NativeMethods.EnumThreadWindows((uint)thread.Id,
            (hWnd, lParam) => { handles.Add(hWnd); return true; }, IntPtr.Zero);
    return handles;
}
private void ExtendRectWithChildWindowSizes(IntPtr handle, ref Rect rect)
{
    uint processId;
    NativeMethods.GetWindowThreadProcessId(handle, out processId);
    var handles = EnumerateProcessWindowHandles((int)processId);
    foreach (var hWnd in handles)
    {
        var childRect = NativeMethods.GetClientRect(hWnd);
        if (childRect.X < rect.X)
        {
            rect.Width += rect.X - childRect.X;
            rect.X = childRect.X;
        }
        if (childRect.Y < rect.Y)
        {
            rect.Height += rect.Y - childRect.Y;
            rect.Y = childRect.Y;
```

```
            }
            if (childRect.Bottom > rect.Bottom)
            {
               rect.Height += childRect.Bottom - rect.Bottom;
            }
            if (childRect.Right> rect.Right)
            {
               rect.Width += childRect.Right - rect.Right;
            }
         }
      }
      public Screenshot CaptureRegion(Rect rect, ScreenshotConfig config)
      {
         return CaptureRectangle(rect, config);
      }
      public Screenshot CaptureRegion(BitmapSource source, Rect rect,
ScreenshotConfig config)
      {
         return CaptureRectangle(rect, config, source);
      }
      #endregion
      #region _Private, protected, internal methods_
      protected Screenshot CaptureRectangle(Rect rect, ScreenshotConfig config,
BitmapSource source = null)
      {
         Contract.Requires(config != null);


         VerifyIsBigEnough(ref rect);
         if (config.RemoveOutsideScreenArea)
```

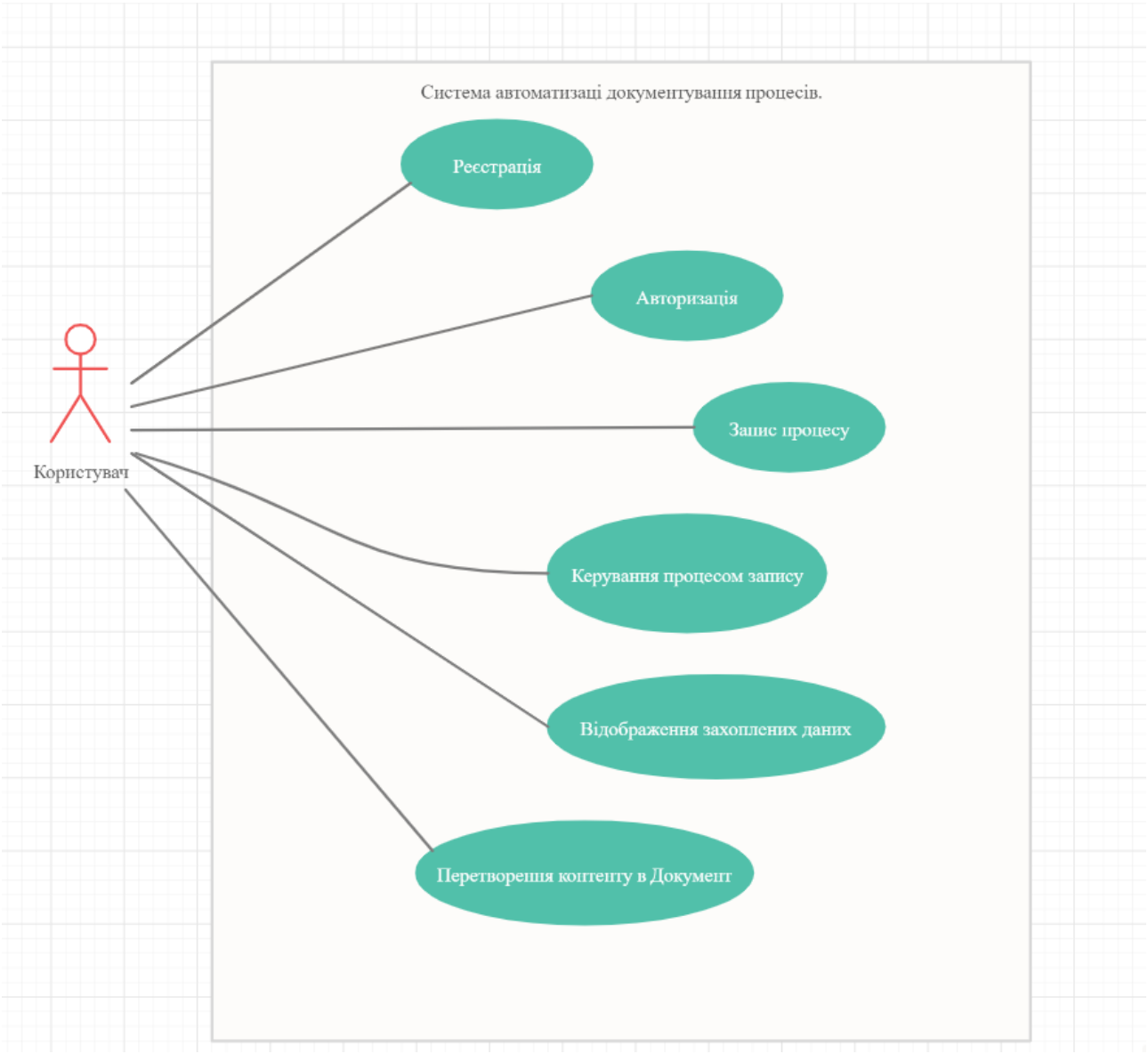
```
        {
            var bounds = CaptureHelpers.GetScreenBounds();
            rect = Rect.Intersect(bounds, rect);
        }
        TryExtendEdge(ref rect, config.EdgeOffset);
        return TryCaptureRectangle(rect, config, source);
    }

    private static Screenshot TryCaptureRectangle(Rect rect, ScreenshotConfig config,
BitmapSource source = null)
    {
        try
        {
            var stopwatch = new Stopwatch();
            stopwatch.Start();
            var image = source != null ? CropImage(source,
CaptureHelpers.ScreenTo0Based(rect)) : CaptureRectangle(rect);

            CursorInfo cursor = null;
            try
            {
                if (config.CaptureCursor)
                {
                    cursor = CursorInfo.GetCursorInfo(rect);
                }
            }
            catch (Exception ex)
            {
                PDLog.Error(ex, "Can't obtain cursorInfo for the screenshot.");
```

```
            }
            var imageSource = image.ToBitmapSource();
            imageSource.Freeze();
            image.Dispose();
            return new Screenshot(rect, imageSource, ImageFormat.Jpeg, new
ScreenshotAction(cursor, null));
        }
        catch (Exception ex)
        {
            PDLog.Error(ex, "Screenshot: Capture failed.");
        }
        return null;
    }
    public static Bitmap CropImage(BitmapSource source, Rect rect)
    {
        Bitmap bmp;
        using (var bitmap = source.ToBitmap())
        {
            var section = rect.ToWinForms();
            // An empty bitmap which will hold the cropped image
            bmp = new Bitmap(section.Width, section.Height);
            using (var g = Graphics.FromImage(bmp))
            {
                // Draw the given area (rect) of the source image
                // at location 0,0 on the empty bitmap (bmp)
                g.DrawImage(bitmap, 0, 0, section, GraphicsUnit.Pixel);
            }
        }
```

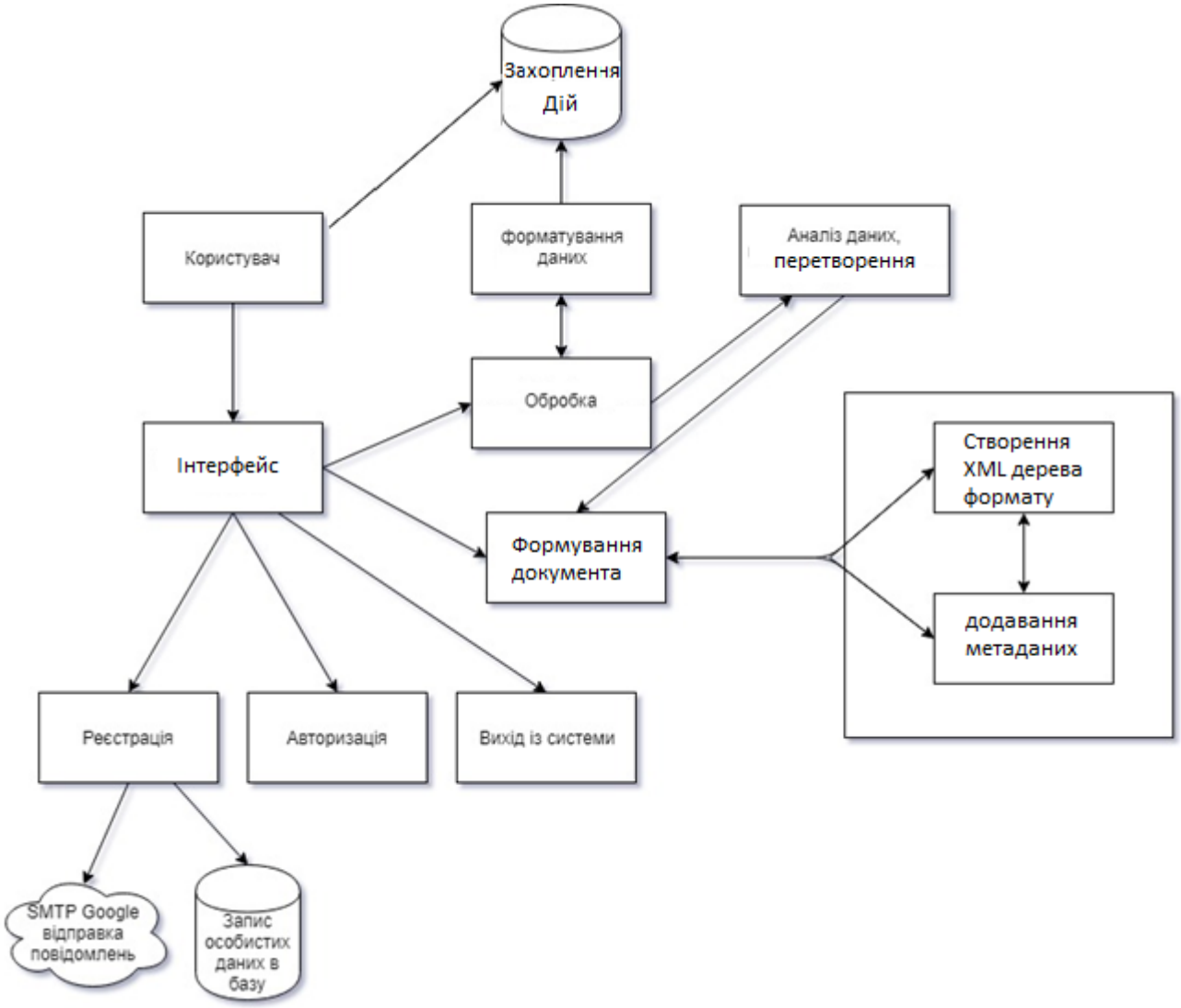
```
        return bmp;
    }
    protected static Bitmap CaptureRectangle(Rect rect)
    {
        return CaptureRectangleNative(rect);
      }
    public static Bitmap CaptureRectangleNative(Rect rect)
    {
        return CaptureRectangleNative(NativeMethods.GetDesktopWindow(), rect);
    }
    public static Bitmap CaptureRectangleNative(IntPtr handle, Rect rectWpf)
    {
        System.Windows.Forms.Application.DoEvents();
        var rect = rectWpf.ToWinForms();
        if (rect.IsEmpty)
            return null;
        IntPtr hdcSrc = NativeMethods.GetWindowDC(handle);
        IntPtr hdcDest = NativeMethods.CreateCompatibleDC(hdcSrc);
        IntPtr hBitmap = NativeMethods.CreateCompatibleBitmap(hdcSrc, rect.Width,
rect.Height);
        IntPtr hOld = NativeMethods.SelectObject(hdcDest, hBitmap);
        NativeMethods.BitBlt(hdcDest, 0, 0, rect.Width, rect.Height, hdcSrc, rect.X,
rect.Y, CopyPixelOperation.SourceCopy | CopyPixelOperation.CaptureBlt);
        NativeMethods.SelectObject(hdcDest, hOld);
        NativeMethods.DeleteDC(hdcDest);
        NativeMethods.ReleaseDC(handle, hdcSrc);
        var img = Image.FromHbitmap(hBitmap);
        NativeMethods.DeleteObject(hBitmap);
        return img;
```

```
        }
        private void VerifyIsBigEnough(ref Rect rect)
        {
            if (rect.Width < 100)
            {
                var toEx = (100 - rect.Width) / 2;
                rect.X -= (int)toEx;
                rect.Width += (int)toEx * 2;
            }
            if (rect.Height < 100)
            {
                var toEx = (100 - rect.Height) / 2;
                rect.Y -= (int)toEx;
                rect.Height += (int)toEx * 2;
            }
        }
        private void TryExtendEdge(ref Rect rect, int edgeOffset)
        {
            rect.Inflate(edgeOffset, edgeOffset);
            rect.Intersect(CaptureHelpers.GetScreenBounds());
        }
    }
}
```

ДОДАТОК Б. ГРАФІЧНИЙ МАТЕРІАЛ

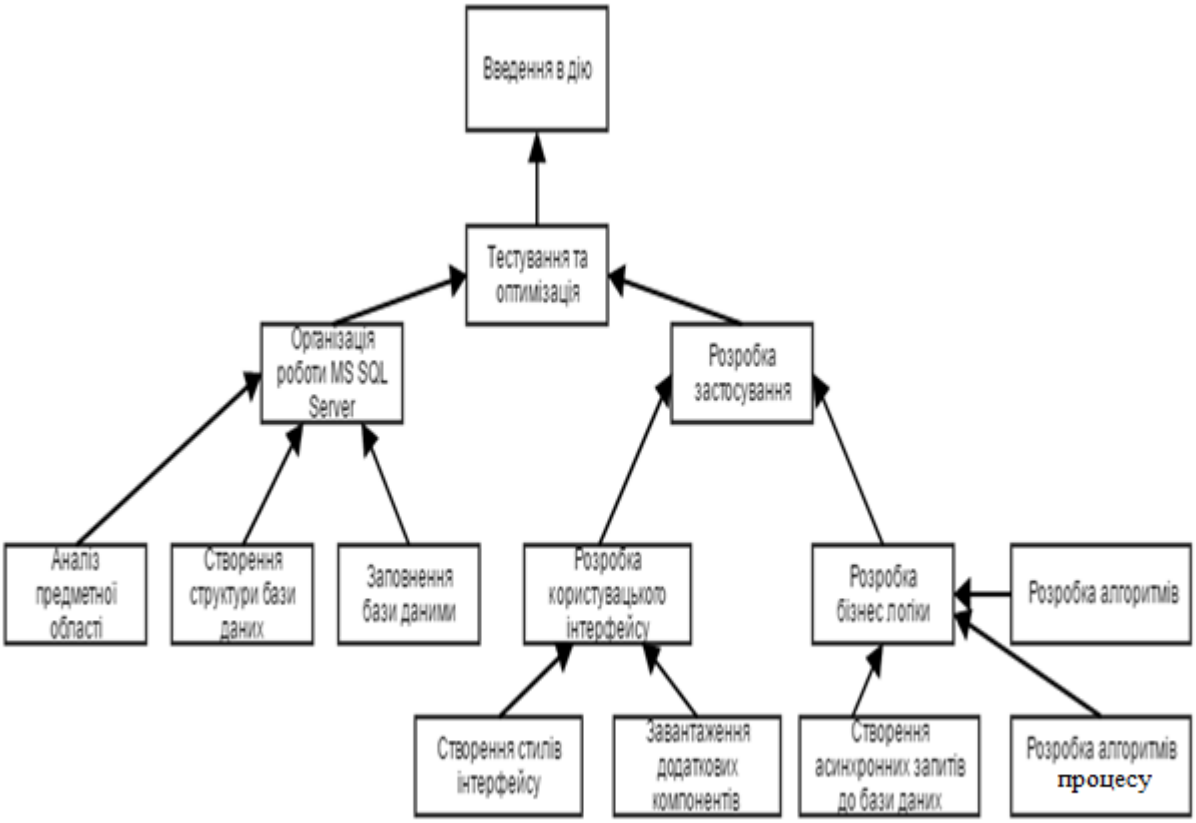


					МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА	
Змн	Арк	Прізвище	Підпис	Дата	Схема варіантів використання системи	Літера: Н; Маса; Масштаб
Розроб.		Кравчук Н.В.				Аркуш 1; Аркушів 6
Н. Контр.						НЛТУ, ДКТД, каф. ІТ, гр. КН-61



					МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА	
Змн	Арк	Прізвище	Підпис	Дата	Функціонально – логічна схема системи	Літера: Н; Маса; Масштаб
Розроб.		Кравчук Н.В.				Аркуш 2; Аркушів 6
Н. Контр.						НЛТУ, ДКТД, каф. ІТ, гр. КН-61

					МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА	
Змн	Арк	Прізвище	Підпис	Дата	Схема бази даних	Літера: Н; Маса; Масштаб
Розроб.		Кравчук Н.В.				Аркуш 3 / Аркушів 6
Керівник						НЛТУ, ДКТД, каф. ІТ, гр. КН-61
Н. Контр.						
Зав. каф.						



Змн	Арк	Прізвище	Підпис	Дата
Розроб.		Кравчук Н.В.		
Н. Контр.				
Зав. каф.				

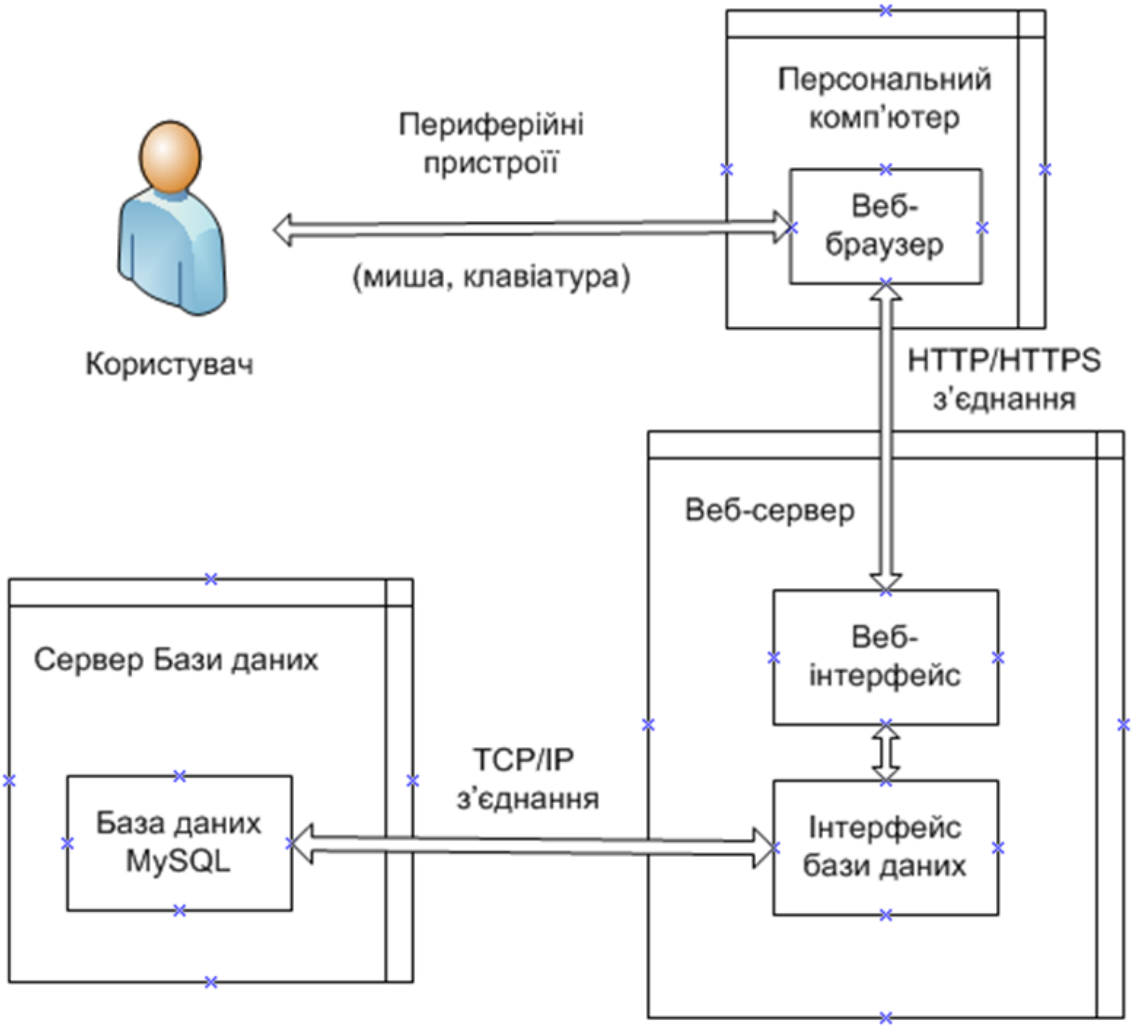
МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА

Дерево цілей

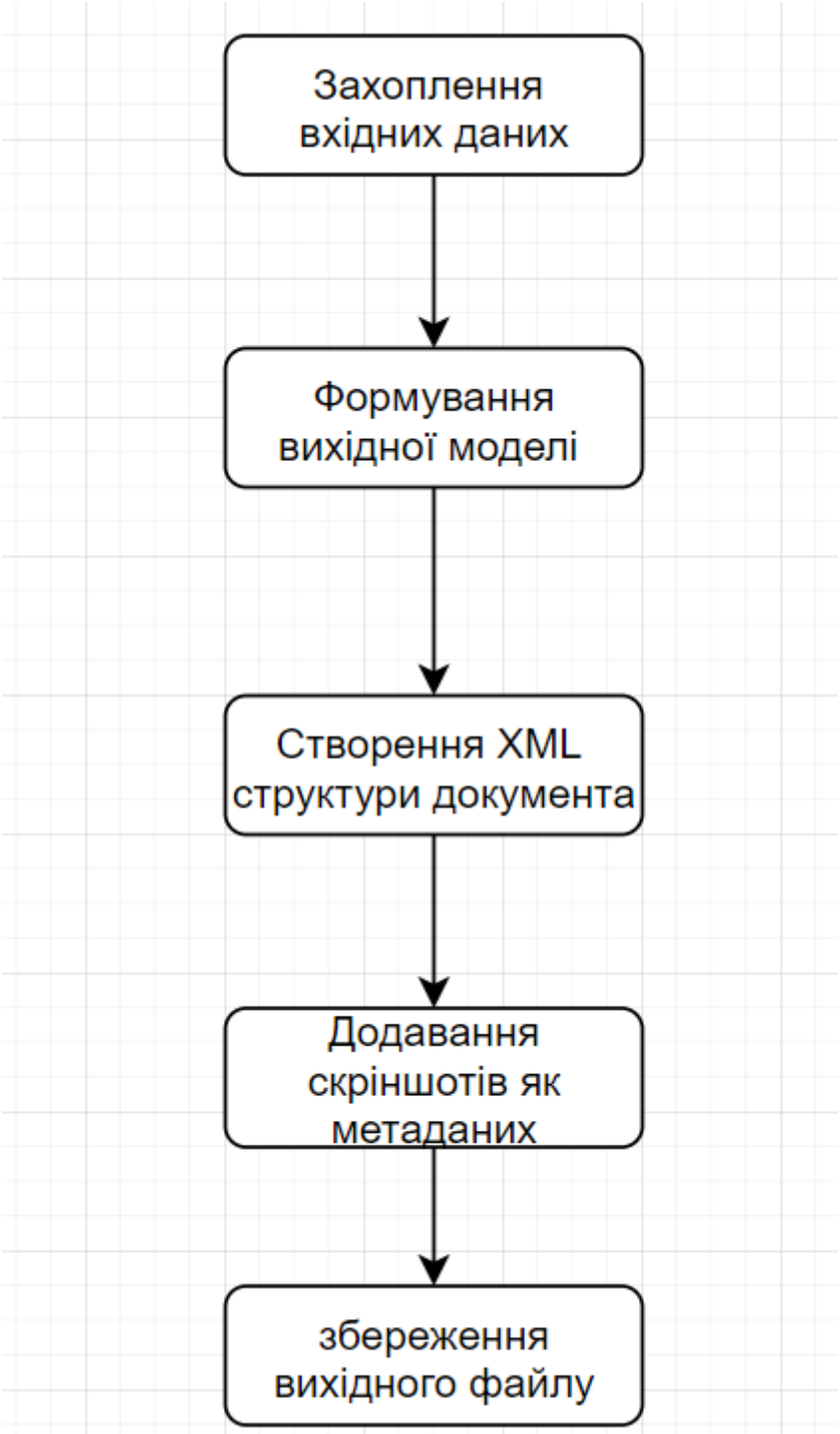
Літера	Маса	Масштаб
Н		

Аркуш 4	Аркушів 6

НЛТУ, ДКТД, каф. ІТ, гр. КН-61



					МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА	
Змн	Арк	Прізвище	Підпис	Дата	Клієнт-серверна архітектура	Літера: Н; Маса; Масштаб
Розроб.		Кравчук Н.В.				Аркуш 5; Аркушів 6
Керівник						НЛТУ, ДКТД, каф. ІТ, гр. КН-61
Н. Контр.						
Зав. каф.						



Змн	Арк	Прізвище	Підпис	Дата
Розроб.		Кравчук Н.В.		
Керівник				
Н. Контр.				
Зав. каф.				

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА

Схема перетворення даних у формат Microsoft Word

Літера	Маса	Масштаб
Н		

Аркуш 6 Аркушів 6

НЛТУ, ДКТД, каф. ІТ, гр. КН-61