МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ВСП НН ІНСТИТУТ ПРОСТОРОВОГО ПЛАНУВАННЯ ТА ПЕРСПЕКТИВНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних систем і технологій

**ЗВІТ**

про проходження навчальної практики

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ТзОВ "Елекс"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

назва установи

Виконав:

студент групи КН-418

шифр групи

\_\_\_\_\_\_\_\_\_/ \_\_\_Василенко А.А.\_\_\_

підпис прізвище, ім’я, по-батькові

Керівник практики від бази практики Керівник практики від університету

\_\_\_\_\_\_\_\_/ директор Коновалов П.О. \_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

підпис посада, прізвище, ім’я, по-батькові підпис посада, прізвище, ім’я, по-батькові

Робота захищена «\_\_» 2024 р.

\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

оцінка посада, прізвище, ім’я, по батькові та підпис члена комісії

Львів – 2024

Міністерство освіти і науки України

Національний університет «Львівська політехніка»

НН інститут просторового проектування та перспективних технологій

Кафедра інформаційних систем і технологій

**Завдання для проходження практики**

Студент                                           Василенко Андрій Андрійович

(прізвище, ім’я, по батькові)

Освітній рівень         Бакалавр          спеціальність         122 “Комп’ютерні науки”

Скерований на практику                        Навчальну

(вид практики)

в місто\_\_\_\_\_\_\_\_\_\_\_\_\_\_Львів\_\_\_\_\_\_\_\_\_\_\_\_\_\_на\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ТзОВ "Елекс"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва підприємства, організації, установи)

згідно угоди №\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ від \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Термін практики: від       15 квітня 2024р          до       4 травня 2024р

(з врахуванням проїзду туди і назад)

Керівник практики від університету                                                                                      

(посада, прізвище, ім’я, по батькові та підпис) .

*Печатка*

*ВНЗ* Директор інституту \_\_\_\_\_\_\_ІППТ\_\_\_

(абревіатура) .

       Хром’як Й.Я.                                                         «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_ р

(прізвище, ініціали) (підпис)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Відмітки про проходження практики:

Прибув на базу практики “  15  ”     квітня     2024 року

(підпис) (посада, прізвище та ініціали відповідальної особи)

Печатка бази практики

Вибув з бази практики “  4  ”     травня     2024 року

(підпис) (посада, прізвище та ініціали відповідальної особи)

Печатка бази практики

**Зміст завдання**

(перелік питань, які підлягають виконанню)

Під час проходження переддипломної практики за темою кваліфікаційної роботи:

(заповнює керівник практики від кафедри (циклової комісії))

1. Подати загальну характеристику бази практики

2. Опрацювати теоретичний матеріал з теми

3. Провести аналіз предметної області\_\_\_\_\_\_\_\_\_\_\_\_

4. Сформувати вимоги до системи, що розробляється

5. Описати технології, які будуть використовуватись

**Завдання видав*:***  15.04.2024

***(***посада, прізвище, ініціали керівника практики від університету, підпис і дата***)***

**Завдання отримав*:***\_\_Василенко А.А. 15.04.2024\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***(***прізвище, ініціали, підпис і дата***)***

**Відгук та оцінка роботи студента на практиці**

(оцінка базою практики виконання завдання)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** (заповнює керівник практики від бази практики)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, прізвище, ім’я, по батькові та підпис керівника практики від бази практики)

Печатка

бази практики « 4 » травня 2024 р.

**Відгук керівника практики від університету про виконання завдання**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата складання заліку «\_\_\_» 2024р.

Оцінка:

За національною шкалою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прописом)

Кількість балів \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(цифрами і прописом)

Члени комісії, які приймали залік

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, прізвище, ім’я, по батькові та підписи)

**ЗМІСТ**

[Вступ 5](#_Toc164633719)

[1. Опис бази практики 6](#_Toc164633720)

[2. Аналітичний огляд літературних та інших джерел 7](#_Toc164633721)

[3. Постановка задачі дипломного проектування та її формалізація 8](#_Toc164633722)

[3.1. Постановка завдання 8](#_Toc164633723)

[3.2 Теоретичні положення та пояснення 8](#_Toc164633724)

[3.3 Аналіз можливих способів рішення 9](#_Toc164633725)

[3.4 Обґрунтування вибраного шляху рішення 11](#_Toc164633726)

[3.5 Аналіз предметної області та використовуваних методів 13](#_Toc164633727)

[4. Вибір та обґрунтування методів і засобів розв’язання завдання 15](#_Toc164633728)

[4.2 Вибір методів та засобів 15](#_Toc164633729)

[4.3 Обґрунтування методів та засобів 16](#_Toc164633730)

[4.4 Порівняльний аналіз методів та засобів 16](#_Toc164633731)

[5. Опис реалізації завдання 18](#_Toc164633732)

[5.1 Структура даних 18](#_Toc164633733)

[5.2 База знань (правила) 18](#_Toc164633734)

[5.3 Механізми логічного виведення 19](#_Toc164633735)

[5.4 Отримання чи підтримання процесів прийняття рішень 21](#_Toc164633736)

[5.5 Структура, склад, зміст і функції розробленого програмного забезпечення 22](#_Toc164633737)

[5.6. Процеси сумісного функціонування 24](#_Toc164633738)

[5.7 Додаткові функції 24](#_Toc164633739)

[Висновок 25](#_Toc164633740)

[Список використаної літератури 27](#_Toc164633741)

[Додаток 28](#_Toc164633742)

# Вступ

Метою даної навчальної практики було ознайомлення з розробкою програмного забезпечення, зокрема конвертера валют, застосовуючи технології .NET Framework та мову програмування C#. Напрацювання і досвід використання цих технологій буде застосований при розробці бакалаврської кваліфікаційної роботи. Під час проходження практики буде створенено конвертер валют – інструмент, який дозволяє користувачам легко переводити суми з однієї валюти в іншу за поточним курсом обміну.

У нашому проекті ми плануємо створити програмний засіб, який буде простим у використанні та надійним у роботі. Користувачі матимуть можливість вибирати базову валюту та валюту призначення, вводити суму для конвертації, а програма автоматично розрахує кінцеву суму з урахуванням поточного обмінного курсу.

Крім того, ми плануємо додати функціонал для збереження історії конвертацій, можливість автоматичного завантаження останньої конвертації при запуску програми, а також можливість збереження поточного курсу для подальшого використання.

Наша мета - створити зручний та ефективний інструмент для користувачів, який допоможе їм швидко та легко вирішувати фінансові завдання.

# 1. Опис бази практики

Моя навчальна практика проходить в компанії "Eleks", яка є відомою інформаційно-технологічною компанією, що спеціалізується на розробці програмного забезпечення та наданні IT-послуг. "Eleks" має великий досвід у галузі розробки програмного забезпечення і успішно впроваджує інноваційні рішення для своїх клієнтів з різних країн світу.

Компанія "Eleks" увійшла в партнерство з різноманітними клієнтами з усього світу, включаючи підприємства з різних галузей, таких як фінанси, медицина, логістика, телекомунікації та інші. Завдяки своєму значному досвіду, "Eleks" займається розробкою корпоративного програмного забезпечення, мобільних додатків, веб-сайтів та інших інноваційних проектів.

Під час навчальної практики в ELEKS, я працював над розробкою конвертера валют з нуля.

Під керівництвом досвідченного фахівця з ELEKS, ми зосередились на використанні передових технологій та практик розробки для створення надійного десктопного додатка. Ми використовували сучасні мови програмування, фреймворки, бібліотеки та інструменти для забезпечення швидкості, безпеки та зручності використання.

# 2. Аналітичний огляд літературних та інших джерел

Аналітичний огляд літературних та інших джерел є важливою частиною роботи, оскільки він дозволяє ознайомитися з науковими дослідженнями, проведеними в даній галузі, та визначити актуальні підходи та методи, які використовуються для розв'язання подібних завдань. Нижче наведено стислий огляд літературних джерел, які були використані для аналізу та підготовки роботи.

FinTech і конвертери валют: В останні роки FinTech (Financial Technology) став одним з найбільш динамічно розвиваючихся секторів фінансової галузі. Дослідження у цій області часто стосуються розробки та впровадження нових інноваційних інструментів, таких як конвертери валют. Дослідники аналізують різні аспекти цих інструментів, включаючи їхню ефективність, зручність використання та вплив на фінансовий ринок.

Методи конвертації валют: У літературі розглядаються різні методи конвертації валют, включаючи фіксований та плаваючий курс, а також методи, що базуються на індексах та фондових ринках. Аналізуються переваги та недоліки кожного методу з метою забезпечення оптимального вибору для конкретних умов.

Технологічні аспекти розробки конвертерів валют: Окрема увага приділяється технологічним аспектам розробки конвертерів валют. Дослідження включають аналіз різних програмних платформ, мов програмування та методів інтеграції з фінансовими даними для створення ефективних та надійних інструментів.

Критичний аналіз існуючих рішень: Наукові праці також містять критичний аналіз існуючих конвертерів валют, відзначаючи їхні переваги та недоліки. Цей аналіз допомагає визначити ключові аспекти, які слід враховувати під час розробки власного конвертера валют.

Аналітичний огляд літературних джерел зазначеної теми дозволив поглибити розуміння сучасних підходів та методів у розробці конвертерів валют, що стане основою для подальших досліджень та розробки.

# 3. Постановка задачі дипломного проектування та її формалізація

## 3.1. Постановка завдання

Отриманий досвід та вивчені нові технології при проходженні практики будуть використані для бакалаврської кваліфікаційної роботи, основне завдання якої полягає у розробці комплексного програмного забезпечення для дилерів, що займаються продажем програмного забезпечення M.E.Doc та ЕЦП. Сиcтема автоматизованого моніторингу має включає в себе облік клієнтів, контроль терміну дії ліцензій на використання модулів програми M.E.Doc, формування рахунків на сплату за ліцензію та актів виконаних робіт, імпорт інформації про замовлені ліцензії ПЗ з дилерського сервісу, а також планування надходжень за їх купівлю. Повинно бути передбачено інформування клієнтів у випадку, коли закінчується термін дії ліцензії на модуль програми, яким він користується. Також необхідно передбачити можливість формування різноманітних аналітичних даних, які спростять прийняття управлінських рішень та сприятимуть підвищенню рівня доходу.

## 3.2 Теоретичні положення та пояснення

Під час розробки програмного забезпечення автоматизованого моніторингу ліцензій та ЕЦП важливо враховувати різноманітні аспекти, пов'язані з технологічними інструментами, які будуть використані. Ось основні теоретичні положення:

* Windows Presentation Foundation (WPF) на C#: Використання WPF дозволяє створювати графічні користувацькі інтерфейси в десктопних додатках. Мова програмування C# забезпечує можливості для реалізації функціональності додатку та взаємодії з базою даних.
* MySQL для зберігання даних: Для зберігання даних програмного забезпечення буде використана база даних MySQL. MySQL - це потужна та надійна реляційна база даних, яка забезпечує швидкодію та масштабованість.
* Telegram-бот на Python: Створення Telegram-бота на Python дозволить користувачам здійснювати взаємодію з базою ліцензій через популярний месенджер Telegram. Мова програмування Python має багато бібліотек для роботи з Telegram Bot API, що спрощує процес розробки бота.
* Power BI для аналізу даних: Використання Power BI дозволить проводити аналіз даних, візуалізувати їх та отримувати звіти про роботу програми. Power BI надає широкий спектр інструментів для аналізу даних та створення інтерактивних звітів та графіків.

Ці технологічні інструменти допоможуть забезпечити якісний функціонал програмного забезпечення, забезпечити його зручність у використанні та ефективність у вирішенні завдань обліку та аналізу даних користувачів.

## 3.3 Аналіз можливих способів рішення

В сучасному світі існує багато новітніх технологій, які можна використовувати при розробці багатофункціонального програмного забезпечення.

Розробка вимагає вибору ефективних технологій для забезпечення високої продуктивності, масштабованості та функціональності. Було прийнято рішення використовувати Angular, Firebase та Ionic в контексті створення фотохостингу. Аналіз цих технологій може надати комплексний погляд на можливості та переваги для саме їх використання.

3.3.1 Вибір технологічного стеку

На основі теоретичних положень, викладених у попередньому розділі, ,ed був запропонований наступний технологічний стек для розробки програмного забезпечення автоматизованого моніторингу ліцензій та ЕЦП:

* **Інтерфейс користувача:** Windows Presentation Foundation (WPF) на C#. WPF - це сучасна платформа для створення графічних інтерфейсів користувача, яка забезпечує високу продуктивність, гнучкість та зручність розробки. C# - це потужна мова програмування з широким набором функцій, яка добре підходить для розробки складних програмних рішень.
* **Зберігання даних:** MySQL. MySQL - це популярна реляційна база даних з відкритим кодом, яка пропонує високу продуктивність, масштабованість та надійність. Вона добре підходить для зберігання великих обсягів даних, пов'язаних з ліцензіями та ЕЦП.
* **Інтеграція з Telegram:** Telegram-бот на Python. Python - це універсальна мова програмування з великою кількістю бібліотек для роботи з Telegram Bot API. Це робить його зручним інструментом для створення Telegram-бота, який дозволить користувачам взаємодіяти з базою ліцензій.
* **Аналітика даних:** Power BI. Power BI - це потужний інструмент для візуалізації та аналізу даних, який дозволить отримувати цінну інформацію про використання ліцензій та ЕЦП. Він пропонує широкий спектр інтерактивних звітів та графіків, які допоможуть користувачам приймати обґрунтовані рішення.

Кожна з обраній технологій має свої переваги та виклики, але їх комбінація створить сильну, ефективну та швидкодіючу систему.

3.3.2 Архітектура програмного забезпечення

Пропонується наступна архітектура програмного забезпечення:

* **Трирівнева архітектура:**
  + **Презентаційний рівень:** Реалізований за допомогою WPF-додатку, який забезпечує графічний інтерфейс користувача для взаємодії з програмним забезпеченням.
  + **Рівень бізнес-логіки:** Реалізований на C# і містить логіку роботи програми, включаючи роботу з базою даних, Telegram-ботом та іншими компонентами.
  + **Рівень доступу до даних:** Реалізований за допомогою MySQL-сервера, який забезпечує зберігання та доступ до даних про ліцензії та ЕЦП.
* **Мікросервісна архітектура:** Telegram-бот та Power BI можуть бути реалізовані як окремі мікросервіси, які взаємодіють з рівнем бізнес-логіки за допомогою API.

3.3.3 Переваги обраного рішення

Обраний технологічний стек та архітектура програмного забезпечення мають ряд переваг:

* **Висока продуктивність та масштабованість:** WPF, MySQL та Power BI - це добре оптимізовані та масштабовані технології, які можуть підтримувати роботу з великими обсягами даних.
* **Зручність використання:** WPF забезпечує створення зручного та інтуїтивно зрозумілого графічного інтерфейсу користувача. Telegram-бот дозволяє користувачам взаємодіяти з програмним забезпеченням через популярний месенджер. Power BI надає зручні інструменти для візуалізації та аналізу даних.
* **Відкритість та гнучкість:** Python та MySQL - це програмні продукти з відкритим кодом, що робить їх доступними та гнучкими у використанні. C# - це потужна та універсальна мова програмування, яка дозволяє реалізувати складну функціональність програмного забезпечення.
* **Інтеграція з сторонніми сервісами:** Telegram Bot API та Power BI API дають можливість інтегрувати програмне забезпечення з іншими сервісами та системами.

Пропонований технологічний стек та архітектура програмного забезпечення створюють основу для розробки ефективного та зручного програмного забезпечення автоматизованого моніторингу ліцензій та ЕЦП.

## 3.4 Обґрунтування вибраного шляху рішення

При постановці задачі важливо враховувати потреби користувачів, технічні аспекти, забезпечення безпеки та привабливого інтерфейсу.

Вибір технологічного стеку та архітектури програмного забезпечення, описаних у попередніх розділах, ґрунтується на наступних ключових міркуваннях:

***3.4.1 Відповідність вимогам***

Обрані технології відповідають всім вимогам, викладеним у технічному завданні. WPF дозволяє створити зручний та інтуїтивно зрозумілий графічний інтерфейс користувача, MySQL забезпечує надійне зберігання даних, Telegram-бот дає можливість користувачам взаємодіяти з програмним забезпеченням через популярний месенджер, а Power BI надає потужні інструменти для візуалізації та аналізу даних.

***3.4.2 Вартість та доступність***

Більшість обраних технологій є безкоштовними або доступними за ліцензією з відкритим кодом. Це робить розробку та експлуатацію програмного забезпечення економічно вигідною.

***3.4.3 Масштабованість та надійність***

WPF, MySQL, Python та Power BI - це добре оптимізовані та масштабовані технології, які можуть підтримувати роботу з великими обсягами даних. Це робить програмне забезпечення стійким до зростання навантажень та гарантує його безперебійну роботу.

***3.4.4 Безпека***

MySQL пропонує ряд функцій безпеки, які дозволяють захистити дані від несанкціонованого доступу. Telegram Bot API та Power BI API також мають вбудовані механізми безпеки для захисту даних під час взаємодії з сторонніми сервісами.

***3.4.5 Перспективи розвитку***

Обрані технології мають широке співтовариство розробників та постійно розвиваються. Це гарантує, що програмне забезпечення буде актуальним та матиме можливість розширюватися в майбутньому.

На основі вищевикладених міркувань можна зробити висновок, що обраний технологічний стек та архітектура програмного забезпечення є оптимальним рішенням для розробки ефективного та зручного програмного забезпечення автоматизованого моніторингу ліцензій та ЕЦП.

## 3.5 Аналіз предметної області та використовуваних методів

*3.5.1 Предметна область*

Предметна область даного програмного забезпечення - це автоматизований моніторинг ліцензій та ЕЦП. Це включає в себе:

* **Зберігання інформації про ліцензії та ЕЦП:** Програмне забезпечення має зберігати інформацію про всі ліцензії та ЕЦП, які використовуються в організації. Ця інформація може включати в себе назву ліцензії, дату придбання, дату закінчення терміну дії, тип ліцензії, стан ліцензії (активна, прострочена, неактивна) та іншу релевантну інформацію.
* **Сповіщення про прострочені ліцензії:** Програмне забезпечення має надсилати сповіщення користувачам та адміністраторам про прострочені ліцензії та ЕЦП. Це дозволить їм вжити заходів для оновлення або видалення ліцензій.
* **Створення звітів:** Програмне забезпечення має генерувати звіти про використання ліцензій та ЕЦП. Ці звіти можуть бути корисними для прийняття рішень щодо управління ліцензіями та ЕЦП.

*3.5.2 Використовувані методи*

Для розробки програмного забезпечення автоматизованого моніторингу ліцензій та ЕЦП будуть використані наступні методи:

* **Об'єктно-орієнтоване програмування (ООП):** ООП буде використовуватися для розробки модульного та гнучкого програмного забезпечення.
* **Шаблони проектування:** Шаблони проектування будуть використовуватися для вирішення загальних проблем проектування програмного забезпечення.
* **Тестування програмного забезпечення:** Програмне забезпечення буде ретельно протестовано, щоб гарантувати його правильну роботу.
* **Інтеграція з сторонніми сервісами:** Програмне забезпечення може бути інтегровано з сторонніми сервісами, такими як Active Directory та LDAP, для отримання інформації про користувачів та комп'ютери.

На основі проведеного аналізу предметної області та використовуваних методів можна зробити висновок, що розробка програмного забезпечення автоматизованого моніторингу ліцензій та ЕЦП є технічно й економічно доцільною.

Програмне забезпечення має бути розроблено з урахуванням потреб та вимог організації. Важливо також забезпечити належне тестування та інтеграцію програмного забезпечення з існуючими системами.

# 4. Вибір та обґрунтування методів і засобів розв’язання завдання

Цей розділ присвячений опису та обґрунтуванню методів, засобів та технологій, які використовувались у бакалаврській кваліфікаційній роботі (БКР) для розробки конвертера валют. Конвертер валют – це інструмент, який дозволяє користувачам легко переводити суми з однієї валюти в іншу за поточним курсом обміну. Дані про валюти та курси обміну отримуються з API за допомогою протоколу HTTP.

## 4.2 Вибір методів та засобів

Вибір методів та засобів для розробки конвертера валют ґрунтувався на наступних факторах:

**Функціональні вимоги:** Конвертер валют повинен мати можливість:

Вводити суму в одній валюті

Вибирати валюту, в яку потрібно конвертувати

Отримувати результат конвертації за поточним курсом обміну

Зберігати історію конвертацій

**Нефункціональні вимоги:** Конвертер валют повинен бути:

Простим у використанні

Швидким та надійним

Зручним та лаконічним

З привабливим інтерфейсом користувача

Портативним та доступним на різних платформах

**Технологічні можливості:** Розробка конвертера валют повинна бути можливою з використанням доступних технологій та інструментів.

Зважаючи на вищезазначені фактори, було обрано наступні методи та засоби:

**Платформа:** WPF (Windows Presentation Foundation). WPF – це платформа розробки десктопних програм для Windows, яка дозволяє створювати графічні інтерфейси користувача з високою продуктивністю та гнучкістю.

**Мова програмування:** C#. C# – це універсальна мова програмування, яка є простою у вивченні та використанні, має велику спільноту розробників та бібліотек, а також підтримується на платформі .NET.

**Бібліотека Material Design:** Material Design – це бібліотека інтерфейсних компонентів для C#, яка дозволяє створювати інтерфейси користувача в стилі Material Design від Google.

**Інші бібліотеки:** Додатково використовувались й інші бібліотеки C#, такі як System.Net.Http для роботи з HTTP-запитами та Newtonsoft.Json для роботи з JSON-даними.

## 4.3 Обґрунтування методів та засобів

Вибір WPF як платформи розробки ґрунтувався на потребі створити десктопний додаток з багатим графічним інтерфейсом. C# – це популярна та добре документована мова програмування, яка добре підходить для розробки десктопних програм на платформі .NET. Бібліотека Material Design використовувалась для створення інтерфейсу користувача в стилі Material Design, який є сучасним, інтуїтивно зрозумілим та привабливим.

Використання цих методів та засобів дозволило створити конвертер валют, який відповідає всім поставленим вимогам. Конвертер простий у використанні, швидкий та надійний, має зручний та лаконічний інтерфейс з приємним дизайном, та доступний на платформі Windows.

## 4.4 Порівняльний аналіз методів та засобів

При виборі методів та засобів для розробки конвертера валют розглядалися й альтернативні варіанти. Наприклад, замість WPF можна було б використовувати інші платформи розробки десктопних програм, такі як WinForms або GTK#. Також можна було б використовувати інші мови програмування, такі як Java або Python. Проте WPF, C# та Material Design були обрані завдяки їх можливостям, простоті використання та широкій поширеності.

Вибір методів та засобів для розробки конвертера валют був ґрунтовним та обґрунтованим. Використання WPF, C#, Material Design та інших бібліотек дозволило створити конвертер валют, який відповідає всім поставленим вимогам. Ці методи та засоби забезпечили:

**Гнучкість та потужність:** WPF надає широкий набір інструментів для створення складних та функціональних графічних інтерфейсів користувача.

**Простоту розробки:** C# є відносно простою мовою програмування, що дозволяє швидко розробляти та підтримувати код конвертера.

**Сучасний дизайн:** Material Design забезпечує привабливий та інтуїтивно зрозумілий інтерфейс користувача, що покращує зручність використання конвертера.

**Ефективність роботи з API:** Використання бібліотеки System.Net.Http дозволяє здійснювати HTTP-запити до API курсів валют ефективно та надійно.

**Легкість роботи з JSON:** Бібліотека Newtonsoft.Json спрощує процес обробки JSON-даних, отриманих з API, що робить конвертер більш гнучким.

Загалом, обрані методи та засоби дозволили створити ефективний та зручний конвертер валют, який відповідає поставленим цілям.

# 5. Опис реалізації завдання

Цей розділ описує реалізацію конвертера валют, розробленого в рамках проходження переддипломної практики. Конвертер дозволяє користувачам легко переводити суми з однієї валюти в іншу за поточним курсом обміну. Дані про валюти та курси обміну отримуються з API за допомогою протоколу HTTP.

## 5.1 Структура даних

Конвертер валют використовує API сервіс для отримання інформації про валюти та курси обміну по HTTP протоколу.

Список валют отримується в JSON форматі за лінком  
https://cdn.jsdelivr.net/npm/@fawazahmed0/currency-api@latest/v1/currencies.json. В цьому списку є код валюти та її повна назва. Самі курси валют отримуються в JSON форматі за лінком

https://cdn.jsdelivr.net/npm/@fawazahmed0/currency-api@{selectedDate}/v1/currencies/{baseCurrencyCode}.json

{selectedDate} це дата у форматі yyyy-MM-dd, на яку треба отримати курси

{baseCurrencyCode} це код базової валюти курси якої треба отримати.

## 5.2 База знань (правила)

Конвертер валют не використовує базу знань у прямому сенсі. Проте, можна вважати, що правилами є алгоритми конвертації валют та отримання даних з API.

**Алгоритм конвертації валют:**

Розрахунок конвертації відбувається у методі CalculateButton (або під час вибору базової валюти та цільової валюти в методі TargetCurrencyComboBox).

Ось кроки розрахунку:

1. Отримання курсу обміну: Програма отримує курс обміну між базовою та цільовою валютами. Це відбувається шляхом отримання тексту з rateTextBox та подальшого розбиття на частини, де курс обміну між валютами міститься у середині рядка.
2. Отримання значень кількості валют: Програма отримує значення кількості базової валюти, введене користувачем у QuantityBaseCurrencyTextBlock.
3. Обчислення кількості цільової валюти: Програма обчислює кількість цільової валюти, використовуючи наступну формулу:

Кількість цільової валюти = Кількість базової валюти × Курс обміну.

І в зворотньому напрямку:

Кількість базової валюти = Кількість цільової валюти / Курс обміну

1. Оновлення відповідних полів: Отримана кількість цільової валюти встановлюється у QuantityTargetCurrencyTextBlock, або наоборот, якщо користувач вводить кількість цільової валюти, то встановлюється кількість базової валюти у QuantityBaseCurrencyTextBlock.
2. Обробка помилок: Програма перевіряє, чи користувач правильно ввів кількість базової валюти, і виводить повідомлення про помилку у випадку некоректного вводу. Розрахувати кінцеву суму, поділивши суму в USD на курс обміну.

**Алгоритм отримання даних з API:**

Сформувати URL-запит до API курсів валют.

Відправити HTTP-запит за URL-адресою.

Отримати відповідь від API у форматі JSON.

Розпарсувати JSON-відповідь та витягти дані про курси обміну.

## 5.3 Механізми логічного виведення

Конвертер валют використовує логіку для реалізації алгоритмів конвертації валют та отримання даних з API. Логіка реалізована на мові програмування C# з використанням стандартних бібліотек та фреймворків.

Аалгоритм роботи програми наступний.

1. Ініціалізація елементів у вікні: Під час завантаження вікна (Window\_Loaded), програма завантажує список валют з сервера, заповнює перший ComboBox (baseCurrencyComboBox) з отриманими даними та налаштовує обробник подій для подальшого вибору базової валюти.
2. Вибір базової валюти: При виборі базової валюти (BaseCurrencyComboBox\_SelectionChanged), програма отримує список доступних цільових валют, виключаючи обрану базову, та заповнює другий ComboBox (targetCurrencyComboBox). Після вибору цільової валюти програма отримує курс обміну між базовою та цільовою валютами.
3. Обчислення курсу обміну: Після вибору базової та цільової валют, а також кількості базової валюти, програма обчислює кількість цільової валюти на основі курсу обміну.
4. Збереження та завантаження останньої конвертації: При закритті вікна програма зберігає останню конвертацію у файл (SaveLastConversion). При наступному відкритті програма завантажує останню конвертацію з файлу та заповнює відповідні елементи.
5. Можливість очищення історії: Кнопка "Очистити історію" дозволяє користувачу очистити історію конвертацій.
6. Можливість виходу з програми: При натисканні на кнопку "Х" у верхньому правому куті вікна, програма відображає діалогове вікно для підтвердження виходу.
7. Можливість переміщення вікна: Користувач може переміщати вікно, утримуючи ліву кнопку миші та пересуваючи вікно.
8. Мінімізація вікна: Користувач може мінімізувати вікно, натиснувши на відповідну іконку.
9. Очищення полів введення: При натисканні на кнопку "С" очищуються поля введення кількості валют.

Отже UML діаграми поведінки системи (діаграми діяльності) матиме такий вигляд зображений на Рис. 5.3.1:



Рис. 5.3.1 Діаграма поведінки системи (діаграма діяльності) конвертера валют

## 5.4 Отримання чи підтримання процесів прийняття рішень

Конвертер валют не використовує складні процеси прийняття рішень. Користувач вводить суму та валюту, яку потрібно конвертувати, а конвертер автоматично розраховує кінцеву суму.

## 5.5 Структура, склад, зміст і функції розробленого програмного забезпечення

Програмне забезпечення конвертера валют складається з наступних компонентів:

**Інтерфейс користувача:** Графічний інтерфейс користувача (GUI) реалізований на платформі WPF з використанням бібліотеки Materialdesign. GUI дозволяє користувачам вводити суму та валюту, яку потрібно конвертувати, а також переглядати результат конвертації і зберігати історію конвертацій, а також при необхідності її очищати.

Розроблений інтерфейс має такий вигляд

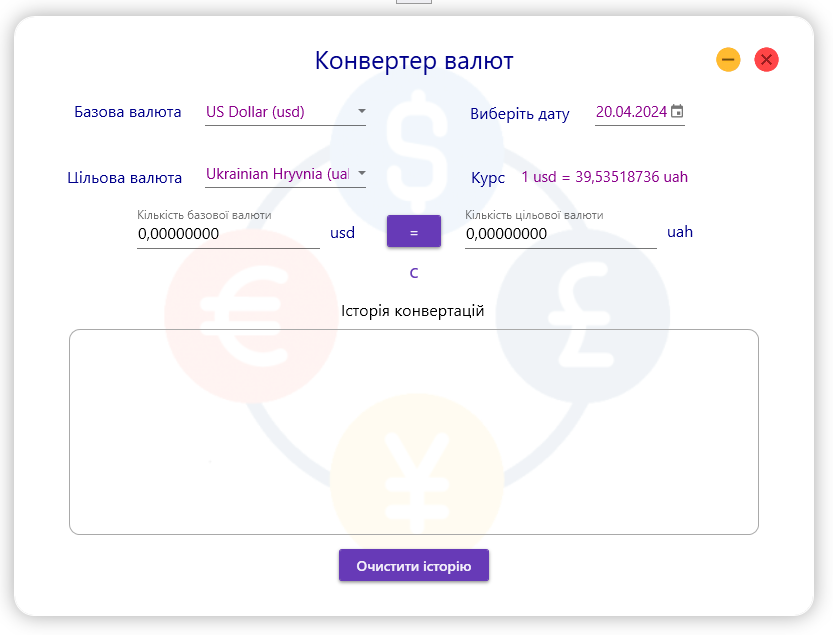
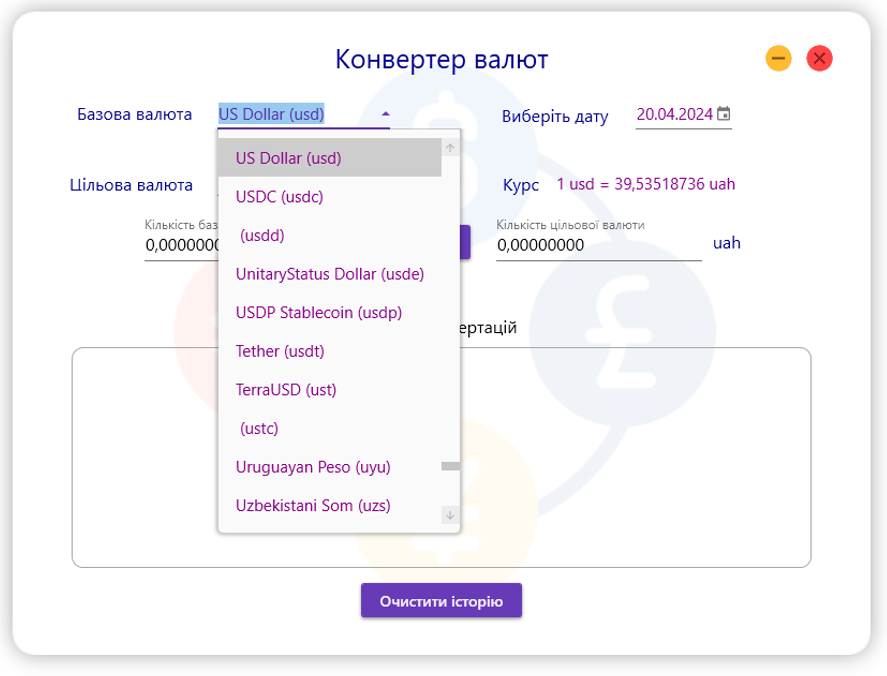
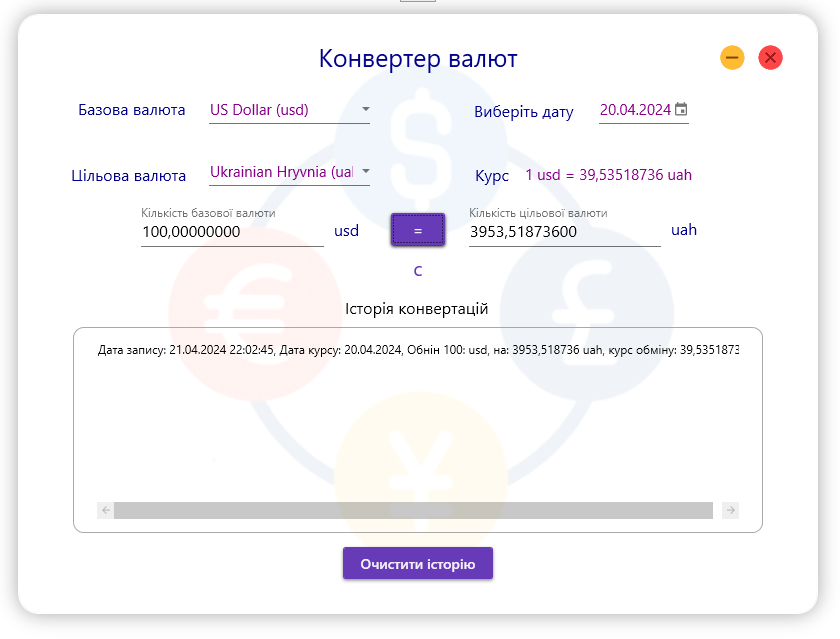


Рис. 5.5.1 Інтерфейс користувача при запуску програми

****

**Рис. 5.5.2 Вигляд інтерфейсу користувача при виборі валюти**

****

**Рис. 5.5.3 Вигляд інтерфейсу користувача після проведення конвертації**

**Логіка програми:** Логіка програми реалізована на мові програмування C#. Вона включає алгоритми конвертації валют, отримання даних з API та інші функції.

**API:** Конвертер використовує API курсів валют для отримання актуальних курсів обміну.

## 5.6. Процеси сумісного функціонування

Компоненти програмного забезпечення конвертера валют взаємодіють один з одним наступним чином:

Користувач вводить суму та валюту, яку потрібно конвертувати, в GUI.

GUI передає введені дані логіці програми.

Логіка програми розраховує кінцеву суму за допомогою алгоритму конвертації валют, описаного раніше.

Якщо курси обміну відсутні або застаріли, логіка програми надсилає HTTP-запит до API курсів валют за допомогою бібліотеки System.Net.Http.

Після отримання відповіді від API, логіка програми парсує JSON-дані та використовує оновлені курси обміну.

Логіка програми надсилає результат конвертації (кінцеву суму) назад до GUI.

GUI оновлює інтерфейс користувача, відображаючи результат конвертації.

## 5.7 Додаткові функції

Відповідно завданню, конвертер валют має додаткові функції, такі як:

* Збереження історії конвертацій
* Збереження останнього стану конвертації

В цьому розділі було описано реалізацію конвертера валют, розробленого на платформі WPF з використанням мови програмування C# та різних бібліотек. Конвертер дозволяє користувачам легко переводити суми з однієї валюти в іншу за поточним курсом обміну. Було описано алгоритми конвертації та отримання даних з API, а також взаємодію компонентів програмного забезпечення.

# Висновок

У процесі виконання завдання переддипломної практики були досягнуті наступні основні результати:

Розроблено програмне забезпечення для конвертації валют з використанням мови програмування C# та технологій WPF (Windows Presentation Foundation) для створення десктопного додатку.

Проведено дослідження та вивчення існуючих API для отримання курсів валют, в результаті чого було обрано відповідне API для використання у розробленому програмному забезпеченні.

На основі отриманих курсів валют та введених користувачем даних програмне забезпечення здійснює конвертацію валют та виводить результат користувачу.

Реалізовано можливість збереження історії конвертацій, що дозволяє користувачам відстежувати їхні попередні операції.

Отримані результати відповідають поставленому завданню на переддипломну практику, а саме розроблено програмне забезпечення для конвертації валют з інтерфейсом користувача та функціоналом, необхідним для виконання поставленої задачі. Розроблений конвертер валют є потужним інструментом для користувачів, що дозволяє швидко та зручно переводити суми з однієї валюти в іншу за актуальними курсами обміну. Він простий у використанні та має зручний інтерфейс, який дозволяє користувачам легко вибирати валюти, вводити суми та переглядати історію конвертацій. Крім того, програма автоматично оновлює курси валют з мережі Інтернет, щоб забезпечити точність та актуальність даних. В цілому, розроблений конвертер валют є потужним інструментом для людей, що потребують швидкої та надійної конвертації валютних сум.

Отже, отриманий практичний досвід в компанії Eleks виявився надзвичайно корисним та важливим для мого особистого й професійного зростання. Це дало мені можливість отримати цінні навички і зрозуміти ключові аспекти роботи в даній галузі. На мою думку, ця практика підвищила мою впевненість і мотивацію для подальшого професійного розвитку та реалізації свого потенціалу.

# Список використаної літератури

1. Andrew Troelsen, Philip Japikse. Pro C# 9 with .NET 5; Apress, 2020. – 1784 с.
2. Adam Nathan. WPF 4.5 Unleashed; Sams Publishing, 2013. – 854 с.
3. Matthew MacDonald. Pro WPF in C# 2010: Windows Presentation Foundation in .NET 4; 2010. – 1248 c.
4. Jennifer Greene, Andrew Stellman. Head First C#, 2013. – 848 с.
5. Adam Nathan. XAML Unleashed, 2014. – 864 с.
6. Material Design In XAML Toolkit. [Електронний ресурс]. – Режим доступу: http://materialdesigninxaml.net/winfx/xaml/themes
7. "WPF & XAML" by Microsoft Press. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-8.0
8. C# language documentation. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/
9. WPF. [Електронний ресурс]. – Режим доступу:    
   https://learn.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-8.0
10. XAML. [Електронний ресурс]. – Режим доступу:  
    https://learn.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-8.0

# Додатки

Лістинг XAML MainWindow.xaml

<Window x:Class="CurrencyConverter.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:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:local="clr-namespace:CurrencyConverter"

mc:Ignorable="d"

Loaded="Window\_Loaded"

WindowStyle="None" AllowsTransparency="True" Background="Transparent"

MouseLeftButtonDown="Window\_MouseLeftButtonDown"

Icon="/img/icon.ico"

Title="Конвертер валют" Height="650" Width="850" WindowStartupLocation="CenterScreen" ResizeMode="NoResize">

<Window.Resources>

<Style TargetType="Label">

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="DarkBlue"/>

<!-- Додайте інші налаштування шрифту, які ви бажаєте застосувати -->

</Style>

<ImageBrush x:Key="WindowBackground" ImageSource="/img/icon.png" Stretch="Uniform" Opacity="0.08"/>

</Window.Resources>

<Border BorderBrush="White" VerticalAlignment="Center" BorderThickness="1" Background="White" MaxWidth="800" CornerRadius="20" MinWidth="500" Margin="15,0,15,0" Padding="30" Grid.ColumnSpan="2" Height="600">

<Border.Effect>

<DropShadowEffect BlurRadius="30" Color="LightGray" ShadowDepth="0"/>

</Border.Effect>

<Grid Background="{StaticResource WindowBackground}">

<Image Source="/img/cloce.png" Height="25" VerticalAlignment="Top" HorizontalAlignment="Right" MouseLeftButtonDown="exit\_MouseLeftButtonDown" MouseEnter="Image\_MouseEnter" MouseLeave="Image\_MouseLeave" Margin="0,0,4,0"/>

<Image Source="/img/turn.png" Height="25" VerticalAlignment="Top" HorizontalAlignment="Right" MouseLeftButtonDown="minimize\_MouseLeftButtonDown" MouseEnter="Image\_MouseEnter" MouseLeave="Image\_MouseLeave" Margin="0,0,42,0"/>

<DatePicker x:Name="datePicker" Margin="550,50,98,0" VerticalAlignment="Top" Language="uk-UA" SelectedDateChanged="TargetCurrencyComboBox\_SelectionChanged" Foreground="DarkMagenta" FontSize="15"/>

<ComboBox x:Name="baseCurrencyComboBox" HorizontalAlignment="Left" Margin="160,50,0,0" VerticalAlignment="Top" Width="161" SelectionChanged="BaseCurrencyComboBox\_SelectionChanged" Foreground="DarkMagenta" FontSize="15" IsEditable="True" IsTextSearchEnabled="True" TextSearch.TextPath="Content"/>

<ComboBox x:Name="targetCurrencyComboBox" HorizontalAlignment="Left" Margin="160,112,0,0" VerticalAlignment="Top" Width="161" SelectionChanged="TargetCurrencyComboBox\_SelectionChanged" Foreground="DarkMagenta" FontSize="15" IsEditable="True" IsTextSearchEnabled="True" TextSearch.TextPath="Content"/>

<Label Content="Виберіть дату" HorizontalAlignment="Left" Margin="420,50,0,0" VerticalAlignment="Top"/>

<Label Content="Базова валюта" HorizontalAlignment="Left" Margin="24,48,0,0" VerticalAlignment="Top"/>

<Label Content="Цільова валюта" HorizontalAlignment="Left" Margin="17,114,0,0" VerticalAlignment="Top" Height="27" Width="132"/>

<Label Content="Конвертер валют" HorizontalAlignment="Center" Margin="0,-10,0,0" VerticalAlignment="Top"

Width="266" HorizontalContentAlignment="Center" FontSize="25"/>

<TextBox x:Name="rateTextBox" HorizontalAlignment="Left" Margin="475,115,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="238" IsReadOnly="True" BorderThickness="0" Foreground="DarkMagenta" FontSize="15"/>

<Label Content="Курс" HorizontalAlignment="Left" Margin="421,114,0,0" VerticalAlignment="Top" Height="36" Width="54"/>

<Border BorderBrush="DarkGray" BorderThickness="1" CornerRadius="10" Margin="24,282,24,50">

<TextBox x:Name="historyTextBlock" HorizontalScrollBarVisibility="Auto" VerticalScrollBarVisibility="Auto" VerticalAlignment="Center" Height="186" IsReadOnly="True" BorderThickness="0" Margin="23,0,23,0" FontSize="12"/>

</Border>

<TextBox x:Name="QuantityBaseCurrencyTextBlock" TextWrapping="Wrap" VerticalScrollBarVisibility="Auto" VerticalAlignment="Top" Height="47" Margin="92,155,463,0" materialDesign:HintAssist.Hint="Кількість базової валюти" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontSize="16" TextChanged="QuantityBaseCurrencyTextBlock\_TextChanged"/>

<TextBox x:Name="QuantityTargetCurrencyTextBlock" TextWrapping="Wrap" VerticalScrollBarVisibility="Auto" VerticalAlignment="Top" Height="47" Margin="420,155,126,0" materialDesign:HintAssist.Hint="Кількість цільової валюти" Style="{StaticResource MaterialDesignFloatingHintTextBox}" FontSize="16" TextChanged="QuantityTargetCurrencyTextBlock\_TextChanged"/>

<Button VerticalAlignment="Top" HorizontalAlignment="Center" Width="150" Content="Очистити історію" Click="ClearHistoryButton\_Click" Margin="0,502,0,0" />

<Button VerticalAlignment="Top" HorizontalAlignment="Center" Width="54" Content=" = " Click="CalculateButton\_Click" FontSize="14" Margin="0,168,0,0" />

<Label Content="Історія конвертацій" HorizontalAlignment="Center" Margin="0,247,0,0" VerticalAlignment="Top" Height="36" Width="156" Foreground="Black"/>

<Label x:Name="BaseCurrencyCode" HorizontalAlignment="Left" Margin="280,169,0,0" VerticalAlignment="Top" Height="36" Width="54"/>

<Label x:Name="TargetCurrencyCode" HorizontalAlignment="Left" Margin="617,168,0,0" VerticalAlignment="Top" Height="36" Width="54"/>

<Button VerticalAlignment="Top" HorizontalAlignment="Center" Width="54" Content="С" Click="ClearButton\_Click" Style="{StaticResource MaterialDesignFlatButton}" FontSize="14" Margin="0,209,0,0" Cursor="Hand" />

</Grid>

</Border>

</Window>

Лістинг коду C# MainWindow.xaml.cs

using System;

using System.Net.Http;

using System.Collections.Generic;

using System.Threading.Tasks;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Newtonsoft.Json.Linq;

using System.IO;

namespace CurrencyConverter

{

    public partial class MainWindow : Window

    {

        //Клас для зберігання історії конвертації

        public class ConversionHistoryItem

        {

            public string BaseCurrency { get; set; }

            public string TargetCurrency { get; set; }

            public DateTime Date { get; set; }

            public string RateDate { get; set; }

            public double ExchangeRate { get; set; }

            public double QuantityBaseCurrency { get; set; }

            public double QuantityTargetCurrency { get; set; }

        }

        //Прапорець, який вказує чи включена обробка подій для ComboBox

        private bool isComboBoxEventDisabled = false;

        //Створення колекції для зберігання історії конвертації

        private List<ConversionHistoryItem> conversionHistory = new List<ConversionHistoryItem>();

        //Метод для додавання нового запису про конвертацію до історії обміну валют.

        public void AddConversionToHistory(string baseCurrencyCode, string targetCurrencyCode, double exchangeRate, string \_RateDate)

        {

            // Створення нового об'єкта ConversionHistoryItem

            ConversionHistoryItem historyItem = new ConversionHistoryItem

            {

                BaseCurrency = baseCurrencyCode,

                TargetCurrency = targetCurrencyCode,

                Date = DateTime.Now, // або використайте обрану вами дату

                RateDate = \_RateDate,

                ExchangeRate = exchangeRate,

                QuantityBaseCurrency = double.Parse(QuantityBaseCurrencyTextBlock.Text),

                QuantityTargetCurrency = double.Parse(QuantityTargetCurrencyTextBlock.Text)

            };

            // Додавання об'єкта до історії

            conversionHistory.Add(historyItem);

            // Оновлення відображення історії в TextBlock

            UpdateHistoryTextBlock();

        }

        //Метод для оновлення вмісту текстового блоку, який відображає історію конвертації.

        private void UpdateHistoryTextBlock()

        {

            // Створення рядка з історією

            StringBuilder historyText = new StringBuilder();

            foreach (var item in conversionHistory)

            {

                historyText.AppendLine($"Дата запису: {item.Date}, Дата курсу: {item.RateDate}, Обнін {item.QuantityBaseCurrency}: {item.BaseCurrency}, на: {item.QuantityTargetCurrency} {item.TargetCurrency}, курс обміну: {item.ExchangeRate}");

            }

            // Оновлення вмісту TextBlock

            historyTextBlock.Text = historyText.ToString();

        }

        // Визначаємо екземпляр класу HttpClient, який використовується для виконання HTTP-запитів у програмі

        private readonly HttpClient \_client;

        public MainWindow()

        {

            InitializeComponent();

            \_client = new HttpClient();

        }

        // Метод викликається при завантаженні вікна MainWindow. Він містить асинхронний код, який виконується під час завантаження вікна

        private async void Window\_Loaded(object sender, RoutedEventArgs e)

        {

            try

            {

                // Отримати список валют з JSON

                string currenciesUrl = "https://cdn.jsdelivr.net/npm/@fawazahmed0/currency-api@latest/v1/currencies.json";

                string currenciesJson = await \_client.GetStringAsync(currenciesUrl);

                JObject currenciesObject = JObject.Parse(currenciesJson);

                // Додати кожну валюту до першого ComboBox

                foreach (var currency in currenciesObject)

                {

                    string currencyCode = currency.Key;

                    string currencyName = (string)currency.Value;

                    ComboBoxItem item = new ComboBoxItem();

                    item.Content = $"{currencyName} ({currencyCode})";

                    baseCurrencyComboBox.Items.Add(item);

                }

            }

            catch (Exception ex)

            {

                MessageBox.Show("Помилка", $"Помилка: {ex.Message}", MessageBoxButton.OKCancel);

            }

            LoadLastConversion();

            LoadHistoryFromFile("HistoryExchange.txt");

        }

        //Метод для виклику, коли вибирається нова базова валюта в першому ComboBox (baseCurrencyComboBox). Основна функція цього методу - оновлення другого ComboBox (targetCurrencyComboBox) зі списком цільових валют.

        private async void BaseCurrencyComboBox\_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)

        {

            ClearButton();

            if (!isComboBoxEventDisabled)

            {

                try

                {

                    // Отримати список валют з JSON

                    string currenciesUrl = "https://cdn.jsdelivr.net/npm/@fawazahmed0/currency-api@latest/v1/currencies.json";

                    string currenciesJson = await \_client.GetStringAsync(currenciesUrl);

                    JObject currenciesObject = JObject.Parse(currenciesJson);

                    // Очистити список валют у ComboBox перед додаванням нових елементів

                    targetCurrencyComboBox.Items.Clear();

                    // Отримати обрану базову валюту

                    string selectedBaseCurrency = ((ComboBoxItem)baseCurrencyComboBox.SelectedItem)?.Content.ToString();

                    string baseCurrencyCode = selectedBaseCurrency?.Split('(')[1].TrimEnd(')').Trim();

                    BaseCurrencyCode.Content = baseCurrencyCode;

                    if (baseCurrencyCode == null)

                    {

                        MessageBox.Show("Помилка", "Будь ласка, виберіть валюту.", MessageBoxButton.OK);

                        return;

                    }

                    // Додати кожну валюту до другого ComboBox, виключаючи обрану базову валюту

                    foreach (var currency in currenciesObject)

                    {

                        string currencyCode = currency.Key;

                        string currencyName = (string)currency.Value;

                        // Пропустити базову валюту

                        if (currencyCode == baseCurrencyCode)

                            continue;

                        ComboBoxItem item = new ComboBoxItem();

                        item.Content = $"{currencyName} ({currencyCode})";

                        targetCurrencyComboBox.Items.Add(item);

                    }

                }

                catch (Exception ex)

                {

                    MessageBox.Show("Помилка", $"Помилка: {ex.Message}", MessageBoxButton.OK);

                }

            }

        }

        //Метод для виклику при зміні цільової валюти в TargetCurrencyComboBox

        private void TargetCurrencyComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

        {

            TargetCurrencyComboBox();

        }

        //Метод для отримання курсу обміну між обраною базовою та цільовою валютами і відображення його у текстовому полі (rateTextBox).

        private async void TargetCurrencyComboBox()

        {

            if (!isComboBoxEventDisabled)

            {

                try

                {

                    // Отримати вибрані валюти

                    string selectedBaseCurrency = ((ComboBoxItem)baseCurrencyComboBox.SelectedItem)?.Content?.ToString();

                    string baseCurrencyCode = selectedBaseCurrency?.Split('(')[1]?.TrimEnd(')')?.Trim();

                    string selectedTargetCurrency = ((ComboBoxItem)targetCurrencyComboBox.SelectedItem)?.Content?.ToString();

                    string targetCurrencyCode = selectedTargetCurrency?.Split('(')[1]?.TrimEnd(')')?.Trim();

                    if (string.IsNullOrEmpty(baseCurrencyCode) || string.IsNullOrEmpty(targetCurrencyCode))

                    {

                        MessageBox.Show("Помилка", "Будь ласка, виберыть базову і цільову валюти.", MessageBoxButton.OK);

                        return;

                    }

                    // Отримати обрану дату

                    string selectedDate = datePicker.SelectedDate?.ToString("yyyy-MM-dd") ?? DateTime.Today.ToString("yyyy-MM-dd");

                    string selectedRateDate = datePicker.SelectedDate?.ToString("dd.MM.yyyy") ?? DateTime.Today.ToString("dd.MM.yyyy");

                    // Отримати курс валют

                    string ratesUrl = $"https://cdn.jsdelivr.net/npm/@fawazahmed0/currency-api@{selectedDate}/v1/currencies/{baseCurrencyCode}.json";

                    // Отримати JSON з курсами валют

                    HttpClient client = new HttpClient();

                    string rateJson = await client.GetStringAsync(ratesUrl);

                    // Парсинг JSON

                    JObject rateObject = JObject.Parse(rateJson);

                    JObject secondLevelObject = (JObject)rateObject[baseCurrencyCode];

                    double exchangeRate = (double)secondLevelObject[targetCurrencyCode];

                    rateTextBox.Text = $"1 {baseCurrencyCode} = {exchangeRate} {targetCurrencyCode}";

                    BaseCurrencyCode.Content = baseCurrencyCode;

                    TargetCurrencyCode.Content = targetCurrencyCode;

                    if (QuantityBaseCurrencyTextBlock.Text != "0,00000000" || string.IsNullOrEmpty(QuantityBaseCurrencyTextBlock.Text))

                    {

                        CalculateButton();

                        AddConversionToHistory(baseCurrencyCode, targetCurrencyCode, exchangeRate, selectedRateDate);

                        SaveHistoryToFile("HistoryExchange.txt");

                        SaveLastConversion();

                    }

                }

                catch (Exception ex)

                {

                    MessageBox.Show("Помилка", $"Помилка: {ex.Message}", MessageBoxButton.OK);

                }

            }

        }

        //Метод для збереження історії конвертації у файл

        public void SaveHistoryToFile(string filePath)

        {

           // Відкриття файлу для запису

           using (StreamWriter writer = new StreamWriter(filePath))

           {

               foreach (var item in conversionHistory)

               {

                      // Записування кожного елементу історії у файл

                      writer.WriteLine($"{item.Date};{item.RateDate};{item.QuantityBaseCurrency};{item.BaseCurrency};{item.QuantityTargetCurrency};{item.TargetCurrency};{item.ExchangeRate}");

               }

           }

        }

        //Метод для завантаження даних історії конвертації з файлу

        public void LoadHistoryFromFile(string filePath)

        {

            // Очистка поточної історії перед завантаженням нової

            historyTextBlock.Clear();

            // Перевірка наявності файлу

            if (File.Exists(filePath))

            {

                // Відкриття файлу для читання

                using (StreamReader reader = new StreamReader(filePath))

                {

                    string line;

                    while ((line = reader.ReadLine()) != null)

                    {

                        // Розбиття рядка на частини

                        string[] parts = line.Split(';');

                        // Створення об'єкта ConversionHistoryItem з прочитаних даних та додавання його до історії

                        ConversionHistoryItem historyItem = new ConversionHistoryItem

                        {

                            Date = DateTime.Parse(parts[0]),

                            RateDate = parts[1],

                            QuantityBaseCurrency = double.Parse(parts[2]),

                            BaseCurrency = parts[3],

                            QuantityTargetCurrency = double.Parse(parts[4]),

                            TargetCurrency = parts[5],

                            ExchangeRate = double.Parse(parts[6])

                        };

                        conversionHistory.Add(historyItem);

                    }

                }

                UpdateHistoryTextBlock();

            }

        }

        //Метод для очистки історії конвертації. Викликається при натисканні кнопки очистки історії конвертації.

        public void ClearHistoryButton\_Click(object sender, RoutedEventArgs e)

        {

            MessageBoxResult result = MessageBox.Show("Повідомлення", "Ви впевнені, що хочете історію конвертації", MessageBoxButton.OKCancel);

            // Перевіряється, чи користувач натиснув кнопку "Добре" (OK), та виконується відповідна дія

            if (result == MessageBoxResult.OK)

            {

                // Очищення списку історії

                conversionHistory.Clear();

                // Очищення тексту в TextBlock

                historyTextBlock.Text = string.Empty;

                SaveHistoryToFile("HistoryExchange.txt");

            }

        }

        //Метод для обробки події натискання іконки виходу з програми

        private void exit\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

        {

            MessageBoxResult result = MessageBox.Show("Повідомлення", "Ви впевнені, що хочете вийти з програми", MessageBoxButton.OKCancel);

            // Перевіряється, чи користувач натиснув кнопку "Добре" (OK), та виконується відповідна дія

            if (result == MessageBoxResult.OK)

            {

                ((App)Application.Current).MainWindow.Close();

            }

        }

        //Метод для зберігання у файл даних останньої конвертації

        private void SaveLastConversion()

        {

            try

            {

                string filePath = "last\_conversion.txt";

                // Отримання даних для збереження

                string baseCurrency = baseCurrencyComboBox.SelectedItem.ToString();

                string targetCurrency = targetCurrencyComboBox.SelectedItem.ToString();

                string date = datePicker.SelectedDate.HasValue ? datePicker.SelectedDate.Value.ToString("yyyy-MM-dd") : DateTime.Today.ToString("yyyy-MM-dd");

                // Створення рядка для збереження

                string line = $"{date};{baseCurrency};{targetCurrency};{rateTextBox.Text};{QuantityBaseCurrencyTextBlock.Text};{QuantityTargetCurrencyTextBlock.Text}";

                // Збереження даних у файл

                File.WriteAllText(filePath, line);

            }

            catch (Exception ex)

            {

                MessageBox.Show("Помилка", $"Помилка при збереженні останньої конвертації: {ex.Message}", MessageBoxButton.OK);

            }

        }

        //Метод для завантаження даних останньої конвертації

        private void LoadLastConversion()

        {

            try

            {

                // Встановити прапорець на блокування обробників подій

                isComboBoxEventDisabled = true;

                string filePath = "last\_conversion.txt";

                if (File.Exists(filePath))

                {

                    // Зчитати рядок з файлу

                    string line = File.ReadAllText(filePath);

                    // Розбити рядок на частини

                    string[] parts = line.Split(';');

                    // Перевірити, чи кількість частин правильна

                    if (parts.Length == 6)

                    {

                        // Встановити значення вибраної дати

                        datePicker.SelectedDate = DateTime.Parse(parts[0]);

                        // Встановити значення вибраної базової валюти

                        foreach (ComboBoxItem item in baseCurrencyComboBox.Items)

                        {

                            if (item.ToString().Contains(parts[1]))

                            {

                                baseCurrencyComboBox.SelectedItem = item;

                                break;

                            }

                        }

                        // Скопіювати елементи з baseCurrencyComboBox до targetCurrencyComboBox

                        targetCurrencyComboBox.Items.Clear();

                        foreach (ComboBoxItem item in baseCurrencyComboBox.Items)

                        {

                            targetCurrencyComboBox.Items.Add(new ComboBoxItem { Content = item.Content });

                        }

                        // Видалити базову валюту зі списку цільових валют

                        foreach (ComboBoxItem item in targetCurrencyComboBox.Items)

                        {

                            if (item.ToString().Contains(parts[1]))

                            {

                                targetCurrencyComboBox.Items.Remove(item);

                                break;

                            }

                        }

                        // Встановити індекс вибраної цільової валюти

                        for (int i = 0; i < targetCurrencyComboBox.Items.Count; i++)

                        {

                            ComboBoxItem item = (ComboBoxItem)targetCurrencyComboBox.Items[i];

                            if (item.ToString().Contains(parts[2]))

                            {

                                targetCurrencyComboBox.SelectedIndex = i;

                                break;

                            }

                        }

                        rateTextBox.Text = parts[3];

                        string[] part = parts[3].Split(' ');

                        BaseCurrencyCode.Content = part[1].Trim();

                        TargetCurrencyCode.Content = part[4].Trim();

                        QuantityBaseCurrencyTextBlock.TextChanged -= QuantityBaseCurrencyTextBlock\_TextChanged;

                        QuantityTargetCurrencyTextBlock.TextChanged -= QuantityTargetCurrencyTextBlock\_TextChanged;

                        QuantityBaseCurrencyTextBlock.Text = parts[4].Trim();

                        QuantityTargetCurrencyTextBlock.Text = parts[5].Trim();

                        QuantityBaseCurrencyTextBlock.TextChanged += QuantityBaseCurrencyTextBlock\_TextChanged;

                        QuantityTargetCurrencyTextBlock.TextChanged += QuantityTargetCurrencyTextBlock\_TextChanged;

                    }

                }

            }

            catch (Exception ex)

            {

                MessageBox.Show("Помилка", $"Помилка при завантаженні останньої конвертації: {ex.Message}", MessageBoxButton.OK);

            }

            finally

            {

                // Скинути прапорець блокування обробників подій

                isComboBoxEventDisabled = false;

            }

        }

        //Зміна типу курсора при наведенні його на відповідний об'єкт

        private void Image\_MouseEnter(object sender, MouseEventArgs e)

        {

            this.Cursor = Cursors.Hand; // Зміна курсора на руку (подібно кнопці)

        }

        //Зміна типу курсора при виходу курсора за межі відповідного об'єкта

        private void Image\_MouseLeave(object sender, MouseEventArgs e)

        {

            this.Cursor = Cursors.Arrow; // Повернення стандартного курсора

        }

        //Метод для переміщення вікна при одночасному натисканні і утримуванні лівої кнопки миші та зміні положенні курсора

        private void Window\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

        {

            if (e.ChangedButton == MouseButton.Left && e.ButtonState == MouseButtonState.Pressed)

            {

                DragMove();

            }

        }

        //Метод для звертання вікна при клікі на відповідному зображенні

        private void minimize\_MouseLeftButtonDown (object sender, MouseButtonEventArgs e)

        {

            // Зміна стану вікна на мінімізацію

            WindowState = WindowState.Minimized;

        }

       //Метод для отримання курсу валют із відповідного текстового поля

        private double GetExchangeRate(string rateString)

        {

            // Розділити рядок по рівню

            string[] parts = rateString.Split(' ');

            // Перевірити, чи є дві частини

            if (parts.Length == 5)

            {

                // Отримати другу частину, яка містить курс

                string ratePart = parts[3].Trim();

                // Спробувати перетворити курс у числовий формат

                if (double.TryParse(ratePart, out double exchangeRate))

                {

                    return exchangeRate;

                }

            }

            // Якщо не вдається знайти курс або перетворити його, повернути нуль або виконати інші дії за необхідності

            return 1;

        }

        //Метод для виклику ClearButton() при натисканні кнопки для очищення полів де відображаються кількості валют

        private void ClearButton\_Click(object sender, RoutedEventArgs e)

        {

            ClearButton();

        }

        //Метод для очищення полів де відображаються кількості валют

        private void ClearButton()

        {

            QuantityBaseCurrencyTextBlock.TextChanged -= QuantityBaseCurrencyTextBlock\_TextChanged;

            QuantityTargetCurrencyTextBlock.TextChanged -= QuantityTargetCurrencyTextBlock\_TextChanged;

            QuantityBaseCurrencyTextBlock.Text="0,00000000";

            QuantityTargetCurrencyTextBlock.Text = "0,00000000";

            QuantityBaseCurrencyTextBlock.TextChanged += QuantityBaseCurrencyTextBlock\_TextChanged;

            QuantityTargetCurrencyTextBlock.TextChanged += QuantityTargetCurrencyTextBlock\_TextChanged;

        }

        //Метод для виклику TargetCurrencyComboBox() при натисканні копки розрахування даних конвертації

        private void CalculateButton\_Click(object sender, RoutedEventArgs e)

        {

            TargetCurrencyComboBox();

        }

        //Метод для розрахунку даних конвертації та заповнення відповідних полів

        private void CalculateButton()

        {

            // Вимкнути обробники подій

            QuantityBaseCurrencyTextBlock.TextChanged -= QuantityBaseCurrencyTextBlock\_TextChanged;

            QuantityTargetCurrencyTextBlock.TextChanged -= QuantityTargetCurrencyTextBlock\_TextChanged;

            try

            {

                // Отримання курсу конвертації між базовою та цільовою валютами

                double exchangeRate = GetExchangeRate(rateTextBox.Text);

                if (!string.IsNullOrEmpty(QuantityBaseCurrencyTextBlock.Text))

                {

                    // Отримати текст з TextBlock

                    string text = QuantityBaseCurrencyTextBlock.Text;

                    // Спробувати конвертувати текст в double

                    if (double.TryParse(text, out double quantityBaseCurrency))

                    {

                        // Тепер змінна quantityBaseCurrency містить значення введеного тексту у форматі double

                        double targetCurrencyCount = quantityBaseCurrency \* exchangeRate;

                        // Розрахунок кількості вибраних валют у цільовій валюті

                        QuantityTargetCurrencyTextBlock.Text = targetCurrencyCount.ToString("F8");

                    }

                    else

                    {

                        // Якщо не вдається сконвертувати текст у double

                        MessageBox.Show("Помилка", "Переконайтеся, що ви правильно ввели число", MessageBoxButton.OK);

                    }

                }

                if (!string.IsNullOrEmpty(QuantityTargetCurrencyTextBlock.Text))

                {

                    // Отримати текст з TextBlock

                    string text = QuantityTargetCurrencyTextBlock.Text;

                    // Спробувати конвертувати текст в double

                    if (double.TryParse(text, out double quantityTargetCurrency))

                    {

                        // Тепер змінна quantityBaseCurrency містить значення введеного тексту у форматі double

                        double targetCurrencyCount = quantityTargetCurrency / exchangeRate;

                        // Розрахунок кількості вибраних валют у цільовій валюті

                        QuantityBaseCurrencyTextBlock.Text = targetCurrencyCount.ToString("F8");

                    }

                    else

                    {

                        // Якщо не вдається сконвертувати текст у double

                        MessageBox.Show("Помилка", "Переконайтеся, що Ви правильно ввели число!", MessageBoxButton.OK);

                    }

                }

            }

            catch (Exception ex)

            {

                MessageBox.Show("Помилка", "Переконайтеся, що Ви правильно ввели число!", MessageBoxButton.OK);

            }

            // Увімкнути обробники подій

            QuantityBaseCurrencyTextBlock.TextChanged += QuantityBaseCurrencyTextBlock\_TextChanged;

            QuantityTargetCurrencyTextBlock.TextChanged += QuantityTargetCurrencyTextBlock\_TextChanged;

        }

        //Метод для очистки поля цільової валюти при внесенні змін в поле базової валюти

        private void QuantityBaseCurrencyTextBlock\_TextChanged(object sender, TextChangedEventArgs e)

        {

            // Очистити поле QuantityTargetCurrencyTextBlock

            QuantityTargetCurrencyTextBlock.Text = "";

        }

        //Метод для очистки поля базової валюти при внесенні змін в поле цільової валюти

        private void QuantityTargetCurrencyTextBlock\_TextChanged(object sender, TextChangedEventArgs e)

        {

            // Очистити поле QuantityBaseCurrencyTextBlock

            QuantityBaseCurrencyTextBlock.Text = "";

        }

    }

}