Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

# МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

**Програмний засіб організації та оптимізації часу і виробничого процесу з інтеграцією сервісів від Google та Microsoft**

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав студент 2 курсу, групи 1КІ-21м
Спеціальності 123 — Комп'ютерна інженерія
Цуренко О.І.
Керівник к.т.н., доц. каф. ОТ
Черняк О.І.
" " 2022 р.
Опонент к.т.н., доц. каф. ПЗ
Коваленко О.О
" " 2022 р.

**Допущено до захисту**
д.т.н., проф. Азаров О.Д.
"22" 12 2022 р.

ВНТУ 2022

**ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія

**ЗАТВЕРДЖУЮ**

Завідувач кафедри обчислювальної техніки

О.Д. Азаров
"15" 09 2022 р.

**З А В Д А Н Н Я**
**НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ**

студенту **Цуренку Олексію Ігоровичу**

1 Тема роботи «Програмний засіб організації та оптимізації часу і виробничого процесу з інтеграцією сервісів від Google та Microsoft» керівник роботи Черняк Олександр Іванович к.т.н., доцент кафедри ОТ, затверджено наказом вищого навчального закладу від дата року № номер 205-А 15.09.2022

2 Строк подання студентом роботи дата.

3 Вихідні дані для роботи:

— технічний опис програмного застосунку;

— мова програмування C#;

— віконний додаток;

— середовище розробки Microsoft Visual Studio;

— Аналоги для інтеграції Google Calendar, Microsoft Outlook.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):вступ, аналіз сучасних методів розробки, варіантний вибір засобів для розробки програмного забезпечення, розробка методів інтеграції синхронізації

аналогів, програмна реалізація, тестування розробленого програмного забезпечення.

5 Перелік графічного матеріалу: блок-схема алгоритму роботи методу, блок-схема алгоритму обробки події.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-3 | Черняк О.І. к.т.н доцент | | |
| 4 | Небава М.І, професор, к.е.н | | |

7 Дата видачі завдання .

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

| № з/п | Назва етапів МКР | Строк виконання | Підпис |
|---|---|---|---|
| 1 | Постановка задачі | 20.09 | |
| 2 | Огляд існуючих рішень | 24.09 | |
| 3 | Розробка структурної схеми | 1.10 | |
| 4 | Розробка функціональної схеми | 4.10 | |
| 5 | Вибір ПЗ для моделювання | 6.10 | |
| 6 | Моделювання роботи методів інтеграції | 12.10 | |
| 7 | Розрахунок економічної частини | 26.10 | |
| 8 | Оформлення пояснювальної записки | 4.11 | |
| 9 | Виконання магістерської кваліфікаційної роботи | 8.11 | |
| 10 | Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків | 10.11 | |
| 11 | Підписи супроводжувальних документів у керівника, опонента, нормоконтролера | 12.11 | |
| 12 | Перевірка «антиплагіат» | 14.11 | |
| 13 | Попередній захист | 22.11 | |

Студент Цуренко Олексій Ігорович

Керівник к.т.н., доц. Черняк Олександр Іванович

# АНОТАЦІЯ

УДК 004.4

Цуренко О.І. Програмний засіб організації та оптимізації часу і виробничого процесу з інтеграцією сервісів від Google та Microsoft. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна Інженерія, Вінниця: ВНТУ, 2022.

Дану магістерську кваліфікаційну роботу присвячено створенню додатку для організації та оптимізації часу і виробничого процесу з інтеграцією сервісів від Google та Microsoft

В роботі був виконаний аналіз найбільш сучасних методів розробки віконних додатків та методів інтеграції хмарних сервісів Google та Microsoft.

Перевірка працездатності системи протестована шляхом практичного тестування системи.

Ключові слова: C#, Visual Studio, Google API, Google Calendar API. Outlook PIA

# ABSTRACT

UDC 004.4

Tsurenko O.I. a software tool for organizing and optimizing time and the production process with the integration of services from Google and Microsoft. Master's thesis in the specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2022.

This master's thesis is devoted to the creation of an application for the organization and optimization of time and the production process with the integration of services from Google and Microsoft.

The analysis of the most modern methods of window application development and integration of cloud services from Google and Microsoft was performed in the work.

Checking the system is tested by practical testing of the system.

Key words: C#, Visual Studio, Google API, Google Calendar API. Outlook PIA

**ЗМІСТ**



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | 08-23.МКР.015.00.000 ПЗ | | |
| *Змн.* | *Лист* | *№ докум.* | *Підпис* | *Дата* | | | |
| *Розроб.* | | Цуренко О.І. | | | Програмний засіб організації та оптимізації часу і виробничого процесу з інтеграцією сервісів від Google та Microsoft Пояснювальна записка | *Літ.* | *Арк.* 6 / *Акрушів* 109 |
| *Перевір.* | | Черняк О.І. | | | | | |
| *Реценз.* | | Коваленко О.О. | | | | *ВНТУ, гр. 1КІ-21м* | |
| *Н. Контр.* | | Швець С.І. | | | | | |
| *Затверд.* | | Азаров О.Д. | | | | | |

# ВСТУП

З підвищенням темпів науково-технічного прогресу спостерігається тенденція зменшення часу, що виділяється на виробничі процеси. Це у свою чергу висуває на передні план  питання організації виробничих процесів і оптимізації часу на їх виконання. Тому на даний час існує багато програмних засобів для вирішення цього питання. Проте вони мають або недостатньо широку функціональність, або є занадто складними для використання. Однак навіть при таких недоліках ці програмні засоби користуються широкою популярністю.

**Актуальність дослідження** пов’язана з необхідністю вирішення ряду задач обміну інформацією між різними системами у володінні користувача для розширення функціональних можливостей, а також забезпечення зручності зберігання та передачі інформації про стан подій.

**Об’єкт дослідження —** процес створення програмного засобу організації та оптимізації часу і виробничого процесу з можливістю інтеграції та синхронізації інформації між хмарними сервісами.

**Предмет дослідження** є програмний засіб організації та оптимізація часу та виробничого процесу з можливістю інтеграції хмарних сервісів Google та Microsoft з їхньою синхронізацією.

**Метою роботи** є розширення функціональних можливостей і підвищення зручності користування програмним засобом організації та оптимізації часу та виробничого процесу за рахунок інтеграції та синхронізації хмарних сервісів Google та Microsoft.

**Задачі дослідження** магістерської роботи:

— аналізувати методи та технології створення ООП-додатків;

— аналіз існуючих аналогів;

— аналізувати та вибирати шляхи реалізації системи;

— виконувати моделювання;

— розробити віконну програму з використанням .NET 5.0 framework для подальшого використання;

— створити систему, яка аналізуватиме введені дані та інформуватиме користувача про недоліки;

— створити методи інтеграції аналогів від Google та Microsoft с засобами відправки та завантаження подій.

— протестувати розроблений додаток.

**Апробація результатів роботи** здійснена в доповіді на Молодь в науці: дослідження, проблеми, перспективи (МН-2023) всеукраїнської науково-практична інтернет-конференції.

**Наукова новизна** — вперше запропоновано і реалізовано інтеграцію хмарних сервісів Google та Microsoft в програмному засобі організації й оптимізації часу і виробничого процесу та забезпечено синхронізацію даних між ними, що значно розширило функціональні можливості і підвищило зручність користування.

**Практична цінність роботи** полягає в можливості використання розробленої системи для організації та оптимізації часу, яка відрізняється розширеними функціональними можливостями і зручністю користування.

# 1 АНАЛІЗ ТЕХНОЛОГІЙ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ ДЛЯ РОЗРОБКИ ВІКОННИХ ДОДАТКІВ

Головною ідеєю роботи є створення додатку для організації часу та виробничого процесу використовуючи ООП.

## 1.1 Створення додатків використовуючи об'єктно орієнтоване програмування

Об'єкти та класи є основою об'єктно-орієнтованого програмування. Це парадигма програмування, де розуміння об'єктів і класів є важливим.

Об'єкти програмування включають меню, зображення, кнопки, поля, смуги прокрутки та текстові написи.

Сучасні програми, як правило, використовують об'єктно-орієнтований метод. Це означає, що основною структурою даних програми є об'єкт; все інше вважається підкласом об'єкта. Це пояснюється тим, що більшість сучасних програм потребують використання певних параметрів, пов'язаних із кожним об'єктом. Наприклад, якщо програма використовує кнопку, їй може знадобитися вказати розмір, колір фону, тип шрифту та навіть наявність рамки чи ні. Стани кнопок визначаються параметрами об'єкта в певний момент часу. Наприклад, сіра кнопка матиме поточний стан розміром 50x100 пікселів і шрифтом Arial, не була натиснута або відпущена.

Сучасне програмне забезпечення є подієво-орієнтованим. Це означає, що більшість програм не працюють самі по собі й не виконують раптово нову дію. Натомість вони реагують на якусь зовнішню подію. Ось чому програмні об'єкти не виконують дії спонтанно; вони реагують лише на щось інше. Сучасні прикладні програми зберігають дані у вигляді об'єктів і містять процеси, які реагують на події. Ці функції називаються об'єктно-орієнтованими та подієвими.

Дані, які повинні залишатися прихованими від решти програми, називаються інкапсульованими даними.

Об'єктно-орієнтована мова програмування повинна включати як інкапсуляцію, так і успадкування, щоб вважатися справді об'єктно-

орієнтованою. Однак самої інкапсуляції недостатньо для створення ООП-мови — цього можна досягти лише успадкуванням.

Об’єктно-орієнтоване програмування вимагає поліморфізму, здатності конкретного об’єкта приймати різні реалізації. Інкапсуляція та успадкування не обов’язково роблять мову програмування більш об’єктно-орієнтованою — вони просто надають більше переваг тому, як працює ООП.

Абстракція означає процес зменшення більшого об’єкта до його основних компонентів. Це техніка, яка використовується для підкреслення суттєвих аспектів об’єкта, одночасно усуваючи несуттєві.

Приховуючи методи, які працюють з даними всередині класу, інкапсуляція, дозволяє об’єднувати дані та їх методи.

Класи можуть успадковувати властивості від інших класів для створення нових класів. Це відомо як успадкування, і отриманий клас є дочірнім, батьківським або суперкласом. Нові класи створюються шляхом запозичення функціональності з існуючих класів у системі.

Поліморфізм — властивість системи використовувати об’єкти з однаковим інтерфейсом без інформації про тип і внутрішню структуру об’єкта.

Інкапсуляція дозволить вам приховати деталі реалізації, а відкрити лише те, що необхідно для подальшого використання. Іншими словами, інкапсуляція — це механізм контролю доступу.

Метою інкапсуляції є відхід від залежності зовнішнього інтерфейсу класу (те, що можуть використовувати інші класи) від реалізації. Щоб найменша зміна в класі не спричинила за собою зміни зовнішньої поведінки класу. Давайте подивимося, як ним користуватися.

Існує 4 типи модифікаторів доступу: публічний, захищений, приватний і за замовчуванням.

Загальнодоступний — рівень надає доступ до компонента з цим модифікатором з екземпляра будь-якого класу та будь-якого пакету.

Захищений — рівень надає доступ до компонента з цим модифікатором із екземплярів батьківського класу та дочірніх класів, незалежно від того, у якому пакеті вони знаходяться.

Default — рівень передбачає доступ до компоненту з цим модифікатором з примірників будь-яких класів, які перебувають в одному пакеті з цим класом.

Private — рівень передбачає доступ до компоненту з цим модифікатором тільки з цього класу.

Для різних структурних елементів класу передбачена можливість застосовувати тільки певні рівні модифікаторів доступу:

— для класу  тільки public і default;

— для атрибутів класу  усі 4 види;

— для конструкторів  усі 4 види;

— для методів усі 4 види.

Успадкування — це процес, за допомогою якого один об'єкт може набувати властивостей іншого. Точніше, об'єкт може успадковувати основні властивості іншого об'єкта і додавати до них риси, характерні тільки для нього.

Спадкування підтримує ідею класифікації речей в ієрархічному порядку завдяки важливості спадкування. Це допомагає людям упорядковувати великі обсяги інформації.

Поліморфізм класу — це здатність екземплярів підкласу взяти на себе роль батьківського класу. Це дозволяє використовувати підкласи замість батьківських класів, що робить їх звичайною формою поліморфізму.

Концепція поліморфізму — це ідея «один інтерфейс, багато методів». Це означає, що ви можете створити спільний інтерфейс для групи схожих дій.

Мутації викликають зміну класу функції або процедури. Це дозволяє різним методам реагувати на те саме повідомлення та створює поліморфізм у поведінці. Це відбувається тому, що багато класів використовують той самий метод за допомогою іншого алгоритму.

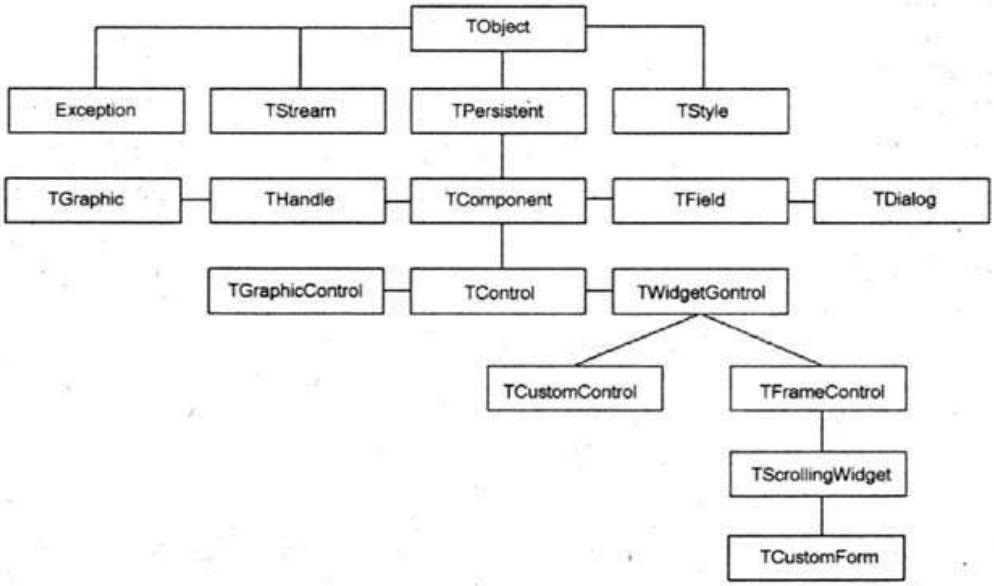


Рисунок 1.1 — Дерево успадкування класів

Поліморфізм означає, що поведінка залежить від класу, в якому ця поведінка викликається, тобто два або більше класів можуть по-різному реагувати на ті самі повідомлення. Це викликано зміною одного з класів методу (процедур, функцій) шляхом написання іншого алгоритму. Наприклад, натискання клавіші Esc призведе до завершення деяких комп'ютерних програм, тоді як інші програми відкриють меню програм лише після натискання клавіші Esc.

```
public class Parent {
   int a = 2;
}
public class Child extends Parent {
   int a = 3;
}
```

Перш за все, слід констатувати, що таке оголошення є правильним. Діти можуть оголошувати поля з будь-якими іменами, навіть такими ж, як у батьківського. Дочірні об'єкти класу міститимуть дві змінні одночасно, і оскільки вони можуть відрізнятися не лише значенням, але й типом (оскільки це два незалежних поля), компілятор визначить, яке значення використовувати.

Компілятор може покладатися лише на тип посилання, за яким посилається на поле:

```
Child c = new Child ();
System.out.println (c.a); // результатом буде 3
Parent p = c;
System.out.println (p.a); // результатом буде 2
```

Обидва посилання вказують на той самий об’єкт, але їх тип різний. Звідси і результат. Оголошення поля в дочірньому класі «приховало» батьківське поле.

У контексті ООП абстракція — це узагальнення даних і поведінки до типу, що знаходиться вище поточного класу в ієрархії.

Коли ви переміщуєте змінні або методи з підкласу в суперклас, ви їх узагальнюєте. Це загальні поняття, і їх можна застосовувати в Java. Але мова також додає поняття абстрактних класів і абстрактних методів[3].

Абстрактний клас — це клас, екземпляр якого неможливо створити.

Наприклад, ви можете створити клас Animal (тварина). Немає сенсу створювати екземпляри цього класу: на практиці вам доведеться створювати екземпляри конкретних класів, наприклад Dog (собака). Але всі класи тварин мають деякі спільні риси, наприклад здатність видавати звуки. Те, що тварина може видавати звуки, нічого не означає.

Видаваний звук залежить від виду тварини.

Визначте загальну поведінку в абстрактному класі та змусьте підкласи реалізувати певну поведінку залежно від їх типу.

Ієрархія може містити як абстрактні, так і конкретні класи.

Наш клас Person містить певний поведінковий метод, і ми ще не знаємо, чи він нам потрібен. Давайте видалимо це і змусимо підкласи реалізувати цю поведінку поліморфно. Ми можемо зробити це, визначивши методи Person як абстрактні. Тоді підкласи повинні будуть реалізувати ці методи.

Оголошення абстрактного методу вимагає, щоб підкласи або реалізували цей метод, або позначили метод у цих підкласах як абстрактний і делегували відповідальність за реалізацію методу наступним підкласам. Деякі методи

можуть бути реалізовані в абстрактному класі, а підкласи можуть реалізовувати інші. Якщо підклас не реалізує абстрактний метод батьківського класу, компілятор видасть помилку.

1.2 Аналіз сучасних сервісів організації та оптимізації часу і виробничого процесу

Перш ніж розробляти вимоги до проектованого мобільного додатку, слід ознайомитися з існуючими програмами, які реалізують можливість планування. Основними сервісами організації заходів є різні організатори, такі як: Google Calendar, Microsoft Calendar.

Календар Google має API для інтеграції даних із сторонніми проектами. Крім того, компанія надає великий вибір бібліотек для полегшення роботи з процесом імпорту даних.

Інтерфейс Календаря Google простий і зрозумілий.

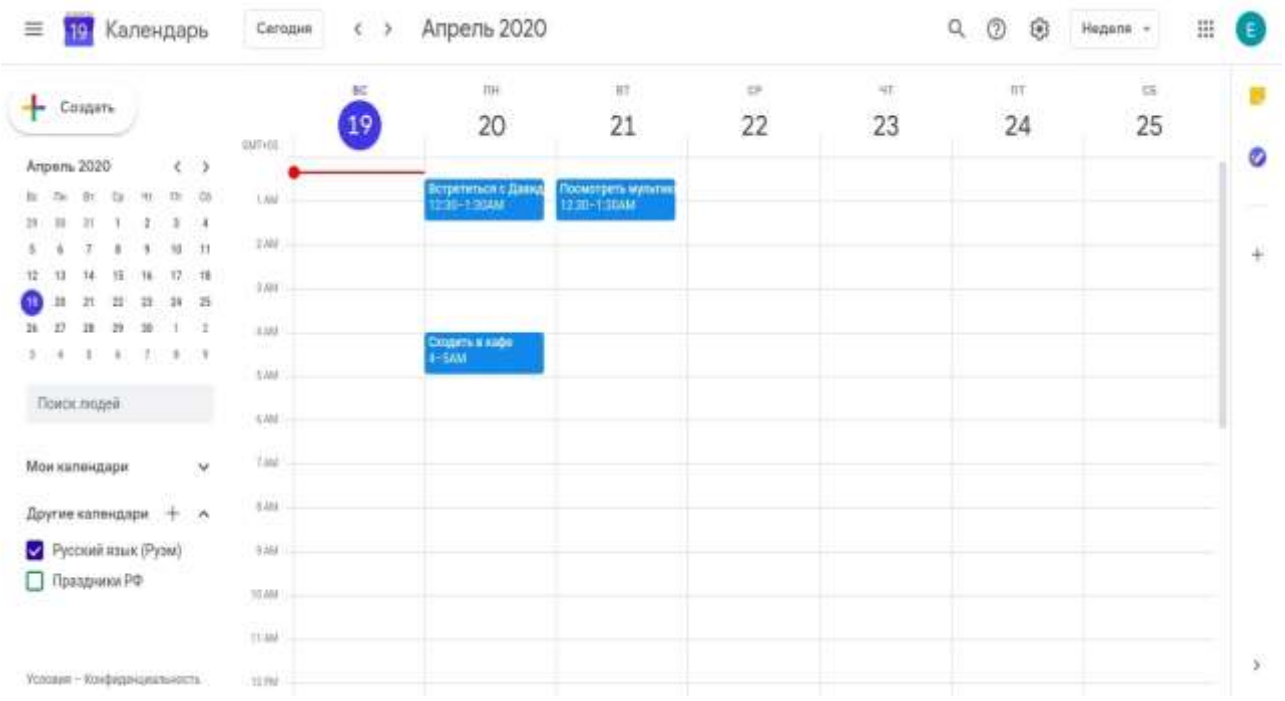


Рисунок 1.2 — Головний інтерфейс користувача Google Calendar

Вікно для створення події дозволяє налаштувати час події, кількість і частоту повторів повідомлень, місце зустрічі, налаштування обробки

відеозустрічі Google Meet, час до події для сповіщення , адреса електронної пошти автора події та можливість запрошувати інших користувачів, додаючи до них ці події, з якими вони можуть навчатися та приєднуватися.

Крім того, Google Calendar дозволяє працювати з іншими додатками Google, такими як пошта, додаючи до нього зручні функції.

Грамотно використовуючи можливості Google календаря, ви зможете дуже зручно організувати події для себе та своїх гостей.

Outlook не такий популярний, але все ще має професійний інтерфейс, доступність на різних пристроях і солідний набір безкоштовних функцій.

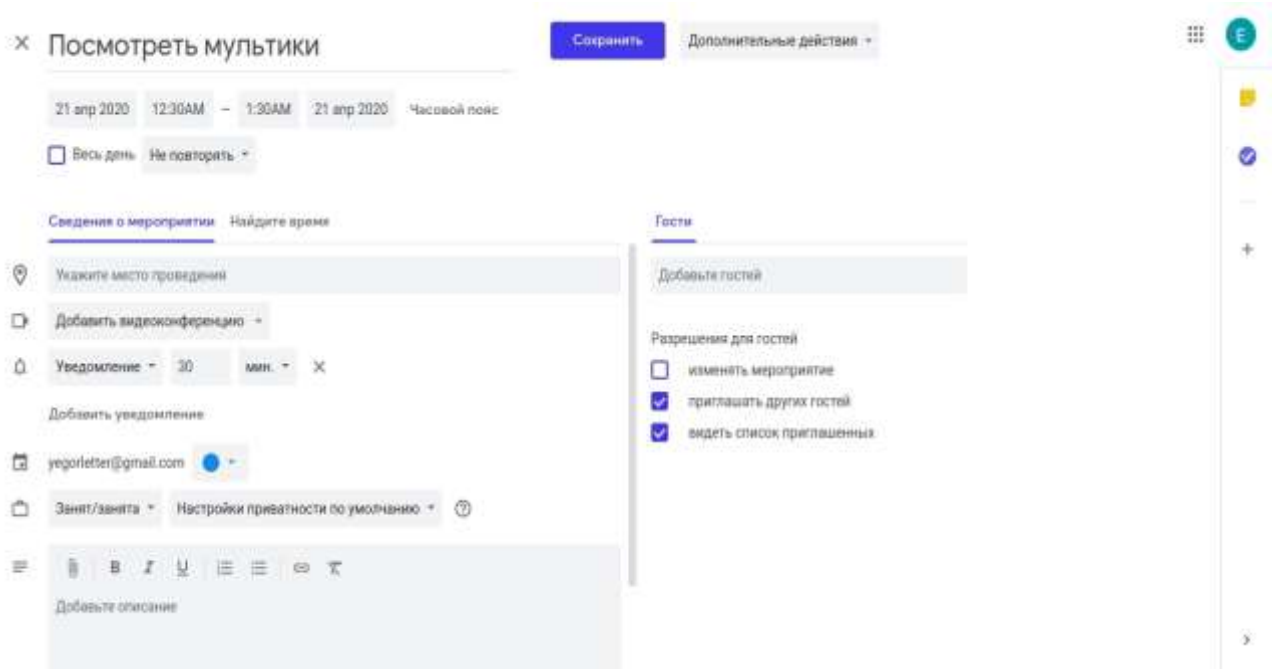


Рисунок 1.3 — Інтерфейс події Google Calendar

У календарі Microsoft є можливість отримувати дані про події у форматі обміну даними — json. Однак цей органайзер не надає готових бібліотек, які спрощують процес імпорту даних

Інтерфейс календаря від Microsoft схожий за можливостями з аналогом від Google, також є можливість встановити час події, запросити учасників, але без можливості організації онлайн-зустрічі.

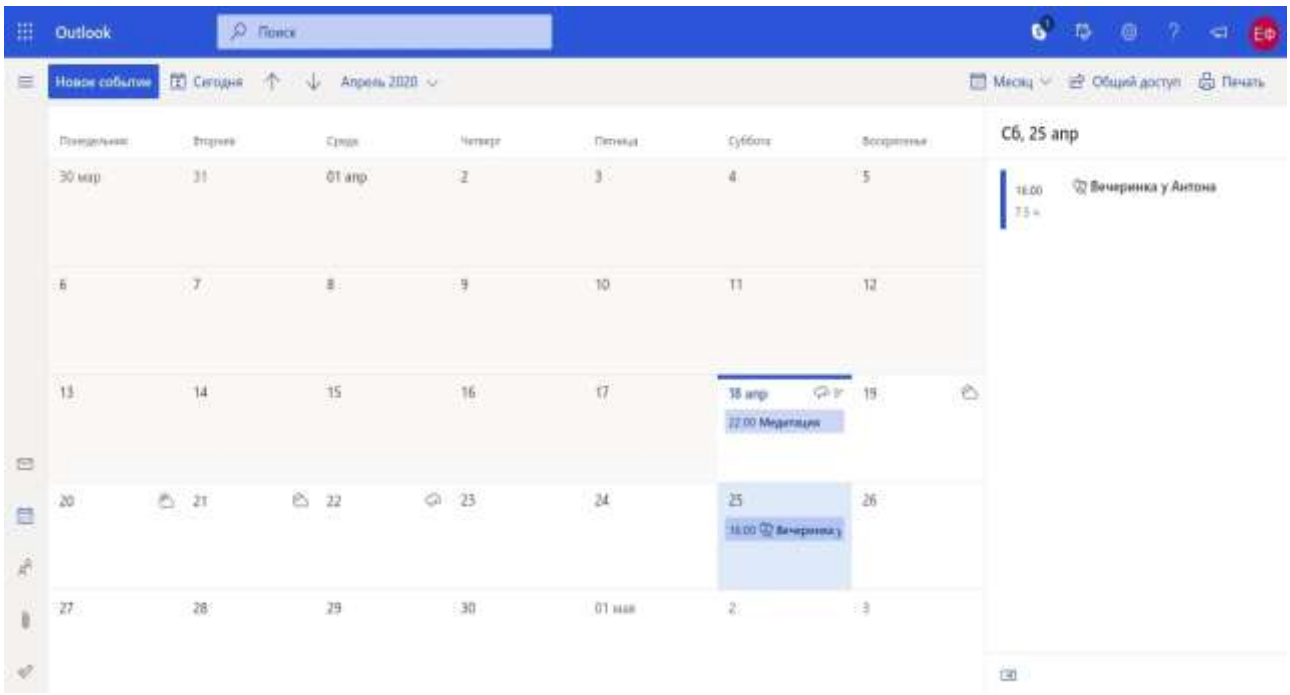


Рисунок 1.4 — Головний інтерфейс користувача календарю Microsoft

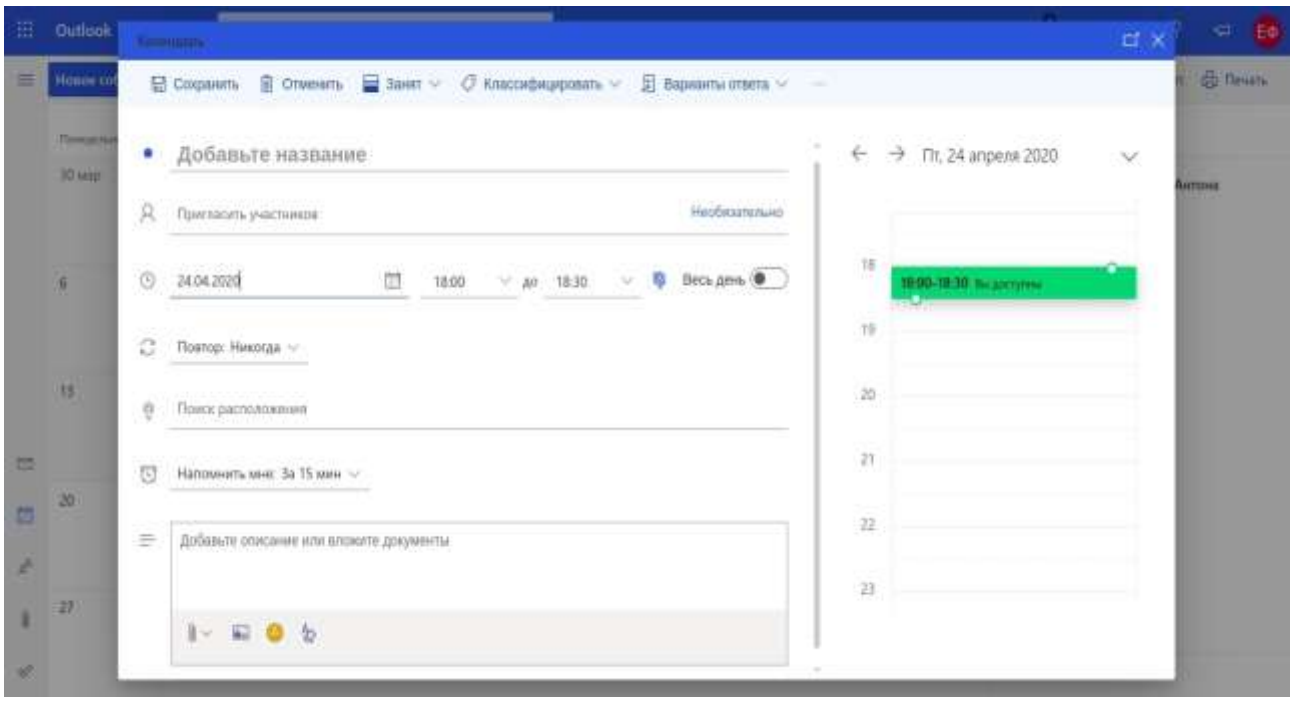


Рисунок 1.5 — Інтерфейс події календарю Microsoft

1.3 Використання середовища .NET 5.0 та мови програмування C# для створення додатків

На даний момент мова програмування C# є однією з найпотужніших, швидкозростаючих і популярних мов в IT-індустрії. На даний момент на ньому

написані найрізноманітніші програми: від невеликих настільних програм до великих інтернет-порталів і інтернет-сервісів, які щодня обслуговують мільйони користувачів.

C# вже не є молодою мовою, і C#, і вся платформа .NET пройшли довгий шлях. Перша версія мови була випущена з випуском Microsoft Visual Studio .NET у лютому 2002 року. Поточна версія мови — C# 9.0, яка була випущена 10 листопада 2020 року з випуском .NET 5.

C# — це мова синтаксису, подібна до C, і в цьому відношенні вона подібна до C++ і Java. Тому, якщо ви знаєте одну з цих мов, вивчити C# буде легше.

C# є об'єктно-орієнтованим і в цьому відношенні він багато чого перейняв від Java і C++. Наприклад, C# підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішувати завдання побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків. А C# все ще активно розвивається, і з кожною новою версією з’являється все більше цікавих функцій, таких як лямбда, динамічне зв’язування, асинхронні методи тощо [5].

Коли говорять C#, вони часто мають на увазі технології .NET (Windows Forms, WPF, ASP.NET, Xamarin). І навпаки, коли люди говорять .NET, вони часто мають на увазі C#. Однак, хоча ці поняття і споріднені, ототожнювати їх неправильно. C# був спеціально розроблений для роботи з .NET, але концепція .NET дещо ширша.

Підтримка багатьох мов програмування. Основою платформи є загальна програма Common Language Runtime (CLR), завдяки якій .NET підтримує кілька мов: крім C#, вона також включає VB.NET, C++, F# і різні діалекти інших мов, пов’язаних з .NET, наприклад Delphi. ІНТЕРНЕТ. Під час компіляції код будь-якої з цих мов компілюється в асемблер за допомогою Common Intermediate Language (CIL), типу асемблера .NET. Тому за певних умов ми можемо створювати окремі модулі однієї програми різними мовами.

Кросплатформенність для роботи на сучасних операційних системах. .NET — портативна платформа (з деякими обмеженнями). Наприклад, остання версія

платформи, .NET 5, підтримується більшістю сучасних операційних систем Windows, MacOS і Linux. Використовуючи різні технології на платформі .NET, ви можете створювати програми на C# для різних платформ — Windows, macOS, Linux, Android, iOS, Tizen.

Потужна бібліотека класів для роботи з фреймворком. .NET надає єдину бібліотеку класів для всіх підтримуваних мов. І незалежно від того, яку програму ми пишемо на C# — текстовий редактор, чат або складний веб-сайт — ми так чи інакше будемо використовувати бібліотеку класів .NET.

Підтримка крос-технологій середовища виконання CLR і основна бібліотека класів забезпечують основу для цілого набору технологій, які розробники можуть використовувати для створення конкретних програм. Наприклад, ADO.NET і Entity Framework Core використовуються в цьому стеку технологій для роботи з базами даних. Технології WPF і UWP використовуються для створення графічних програм із багатим, насиченим інтерфейсом, а Windows Forms використовуються для створення простіших графічних програм. Для розробки мобільних додатків — Xamarin. Для розробки сайтів і веб-додатків — ASP.NET і т.д.

До цього варто додати динамічно розвивається і набирає популярність Blazor — .NET фреймворк, який дозволяє створювати веб-додатки як на стороні сервера, так і на стороні клієнта. А в майбутньому він підтримуватиме розробку мобільних додатків і, можливо, настільних додатків.

Згідно з багатьма тестами, продуктивність веб-додатків на платформі .NET 5 значно перевищує веб-додатки, створені з використанням інших технологій у багатьох категоріях. Програми на .NET 5 зазвичай характеризуються високою продуктивністю.

Також зверніть увагу на такі особливості мови C# і .NET framework, як автоматичне збирання сміття. А це означає, що в більшості випадків, на відміну від C++, нам не доведеться турбуватися про звільнення пам’яті. Наведена вище загальномовна CLR сама викличе збирач сміття та очистить пам’ять.

Варто зазначити, що .NET довгий час розроблявся переважно як платформа Windows під назвою .NET Framework. У 2019 році вийшла остання версія цієї платформи — .NET Framework 4.8. Більше не розвивається [6].

З 2014 року Microsoft почала розробляти альтернативну платформу — .NET Core, яка вже була призначена для різних платформ і повинна була увібрати всі можливості застарілої .NET Framework і додати нові функціональні можливості. Потім Microsoft послідовно випустила кілька версій цієї платформи: .NET Core 1, .NET Core 2, .NET Core 3. Логічним розвитком .NET Core 3.0 стала платформа .NET 5, яка розглядається в цьому посібнику. — платформа .NET 5.

Також варто згадати платформу Mono, яка була створена ще в 2004 році і представляла собою відкриту версію .NET Framework для Linux і MacOS. Використовуючи Mono, можна було створювати кросплатформні програми на C#. Моно використовується і сьогодні. Наприклад, Xamarin, технологія розробки мобільних додатків для Android та iOS на C#, використовує Mono.

Часто програму, написану на C#, називають керованим кодом. Це означає, що система для цієї програми побудована на основі .NET і, отже, керується звичайним CLR, який завантажує програму та очищає за потреби. Але є також додатки, наприклад, написані на C++, які скомпільовані не до звичайної мови CIL, такої як C# або F#, а до простого машинного коду. У цьому випадку .NET не керує програмою.

Як було написано вище, код C# компілюється в програмі або компілюється з розширеннями exe чи dll у CIL. Потім, коли починається виконання такої програми, вона компілюється JIT (Just-In-Time) у машинний код, який потім виконується. У той же час, оскільки наш додаток може бути великим і містити багато інструкцій, лише та частина програми, яка безпосередньо адресована, буде скомпільована в поточний момент. Якщо ми перейдемо до іншої частини коду, він буде скомпільований із CIL у машинний код. При цьому вже скомпільована частина програми зберігається до кінця програми. В результаті підвищується продуктивність праці.

## 1.4 Огляд API Календаря Google та PIA Microsoft Outlook

API Календаря Google — це RESTful API, до якого можна отримати доступ через явні дзвінки HTTP або клієнтські бібліотеки Google. API надає більшість функцій, доступних у веб-інтерфейсі Календаря Google.

Нижче наведено список загальних термінів, що використовуються в API Календаря Google:

Захід — подія в календарі, що містить таку інформацію, як назва, час початку та закінчення, а також учасників. Події можуть бути як одиничними, так і повторюваними. Подія представлена ресурсом Event.

Календар — колекція подій. З кожним календарем пов'язані метадані, такі як опис календаря або часовий пояс календаря за промовчанням. Метадані одного календаря представлені ресурсом Calendar .

Список календарів — список усіх календарів у списку календарів користувача в інтерфейсі користувача календаря. Метадані для одного календаря, який відображається у списку календарів, представлені ресурсом CalendarListEntry . Ці метадані включають власні властивості календаря, такі як його колір або повідомлення про нові події.

Параметр — налаштування користувача з інтерфейсу користувача календаря, наприклад часовий пояс користувача. Одна користувальницька перевага представлена параметром Setting Resource.

ACL-список — правило керування доступом, яке надає користувачеві (або групі користувачів) певний рівень доступу до календаря. Одне правило управління доступом представлене ресурсом ACL.

Принцип роботи PIA календарю Microsoft Outlook по принципу дії дуже схожий на API від Google.

В Outlook клієнти можуть створювати індивідуальні календарі для роботи, сім’ї та інших цілей і організовувати їх у групи календарів. Вони можуть увімкнути безкоштовний календар днів народження та свят, щоб нагадувати їм про дні народження контактів та місцеві свята. Вони можуть додавати календарі,

які відповідають їхнім інтересам, наприклад, календарі для спортивних команд і телепрограм. Клієнти можуть вибирати та накладати календарі, а також переглядати свої події в одному поданні. За допомогою API календаря програма може так само організовувати календарі в групи календарів і взаємодіяти з цікавими календарями, як і з будь-яким іншим календарем у поштовій скриньці користувача.

Клієнти Outlook можуть узгоджено застосовувати категорії до подій, повідомлень, контактів, завдань і групових публікацій, щоб покращити організацію та пошук. API календаря дозволяє отримати доступ до головного списку категорій користувача та визначити його, що відкриває додаткові творчі сценарії. Наприклад, спортивний клуб може організувати спортивний турнір і запропонувати програму, яка розрізняє електронні листи та події для кожного виду спорту за власною кольоровою категорією. Для останніх новин, таких як непередбачені зміни розкладу, програма також може встановити властивість важливості цих подій і електронних листів, щоб попередити клієнтів.

У папці календаря можливо створювати та оновлювати події окремого екземпляра або планувати та підтримувати повторювані події. Також можливо дозволити своїм клієнтам відповідати на запрошення на зустрічі, а також відкладати чи відхиляти нагадування за допомогою пов’язаної властивості навігації подією.

Основне складання взаємодії Outlook (PIA) повністю підтримує розробку керованого коду за допомогою об'єктної моделі Outlook. Але при першому погляді на PIA в оглядачі об'єктів можна здивуватися великою кількістю зайвих інтерфейсів, що містяться в PIA, і тому факту, що не всі члени методи, властивості і події об'єкта представлені одним і тим же інтерфейсом об'єкта.

# 2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ РОБОТИ ЗАСОБУ ОРГАНІЗАЦІЇ ТА ОПТИМІЗАЦІЇ ЧАСУ І ВИРОБНИЧОГО ПРОЦЕССУ З ІНТЕГРАЦІЄЮ СЕРВІСІВ АНАЛОГІВ

Інструмент для організації та оптимізації часу і виробничого процесу з інтеграцією аналогів ділиться на кілька компонентів:

— процесор дат, що включає створення зручного інтерфейсу календаря, встановлення необхідних дат та обробку дій користувача з ними;

— обробник події, який включає створення, редагування, видалення та публікацію часу події;

— збереження/відновлення подій за допомогою серіалізації для збереження та відтворення створених подій для подальшого виконання, коли програма знову викликається;

— методів завантаження та відправлення подій з віконного додатку на клієнтські облікові записи календарів від Google та Microsoft Outlook.

## 2.1 Розробка методів взаємодії з користувачем

Щоб отримати зручний інтерфейс для дат, необхідно створити базовий інтерфейс, який буде пізніше відредаговано для відображення дат у зрозумілій формі.

За допомогою методів визначення системного часу буде проаналізовано поточний місяць і рік в залежності від того, який день тижня є початковим, скільки днів у місяці, після чого буде відредаговано інтерфейс для зручного відображення календар цього місяця.

Якщо ви бажаєте отримати календар на інший місяць, скористайтеся кнопками зміни дати, щоб змінити дату, щоб відобразити потрібні дні.

Загальна робота методу генерації даних проілюстрована блок-схемою на рисунку 2.1.


Рисунок 2.1 — Алгоритм роботи методу дат

Завдяки створеному простому інтерфейсу користувачу буде дуже легко та зручно перемикати дату для перегляду необхідного місяцю та переглядати події які будуть заплановані

Для обробки подій створюється список доступних подій, який потім передається у друге вікно, відповідальне за події.

У вікні події можна відобразити дату події та її опис. Якщо вікно було запущено через наявну подію, можна змінити час і опис або повністю видалити цю подію.

Після зміни списку подій події, які відображаються в головному вікні, редагуються автоматично.

Для взаємодії користувача з методами синхронізації даних з календарями Google та Microsoft Outlook буде створено вікно у якому буде надано необхідний функціонал.

2.2 Розробка методу для збереження подій

Для зручності використання програми необхідно зберігати створені події та відновлювати події після запуску програми як вказано на рисунку 2.2.

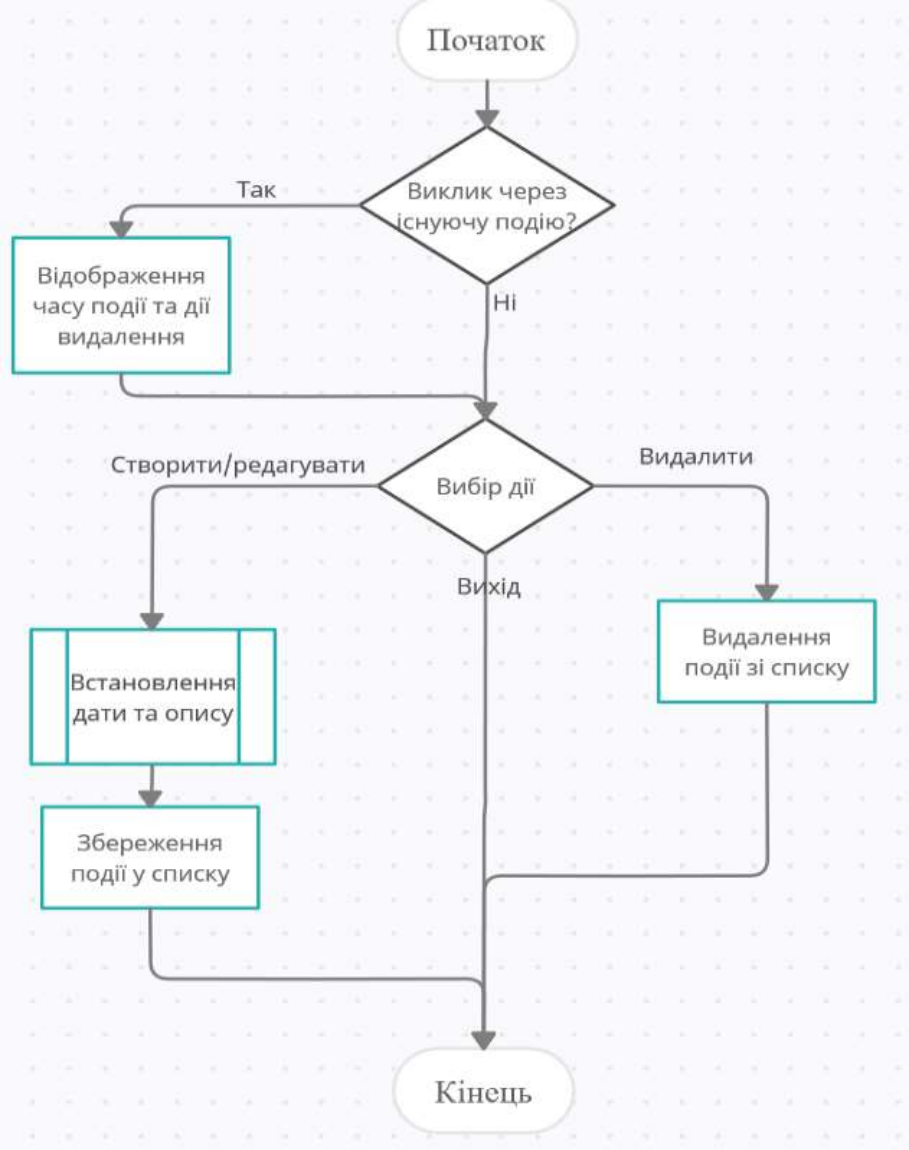


Рисунок 2.2 — Алгоритм обробки події

Для цього потрібно використовувати серіалізацію подій.

Серіалізація — це процес поміщення об’єкта у форму, яку можна записати в потік. Це процес перетворення об’єкта у форму, у якій його можна зберігати у файлі, базі даних чи пам’яті або передати через мережу. Алгоритм роботи зображено на рисунку 2.3.

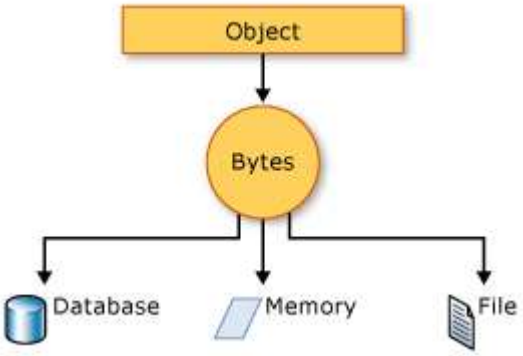


Рисунок 2.3 — Переведення об’єкту в форму для збереження

Його основна мета — зберігати стан об’єкта, щоб його можна було відтворити за потреби.

З його допомогою список подій буде переведений у форму, яку можна буде зберегти та відтворити в майбутньому.

Коли подія створюється, редагується або видаляється, вона буде серіалізована. Серіалізація зберігає список у файлі, який буде створено під час запуску програми.

Є 3 види серіалізації:

— двійкова серіалізація;

— XML серіалізація;

— серіалізація JSON.

Різниця між ними полягає у форматі кінцевого результату. Ця програма використовуватиме двійкову серіалізацію.

Десеріалізація відбувається під час відкриття програми.

Десеріалізація є процесом, зворотним до серіалізації. Це процес повернення серіалізованого об'єкта, щоб його можна було завантажити в пам'ять. Регенерує стан об’єкта шляхом встановлення властивостей, полів тощо.

У нашому випадку після дереалізації буде створено список подій, які програма завантажить в себе для подальшої обробки.

Також серіалізація буде використана при обробці методів синхронізації даних.

## 2.3 Підключення Google API та Outlook PIA у додаток

Для розробки методу синхронізації використовуючи Google API потрібно підключення Google workspace. Google Workspace пропонує широкий спектр продуктів та інструментів для розробників, які дозволяють пов'язати додаток з Google Workspace або розширити можливості програм, таких як Gmail, Google Диск та Google Chat. Кожна програма або інтеграція Google Workspace має власний проект Google Cloud, в якому розробник налаштовуєте API, налаштовуєте автентифікацію алгоритм якої вказано на рисунку 2.4 та керує розгортанням.

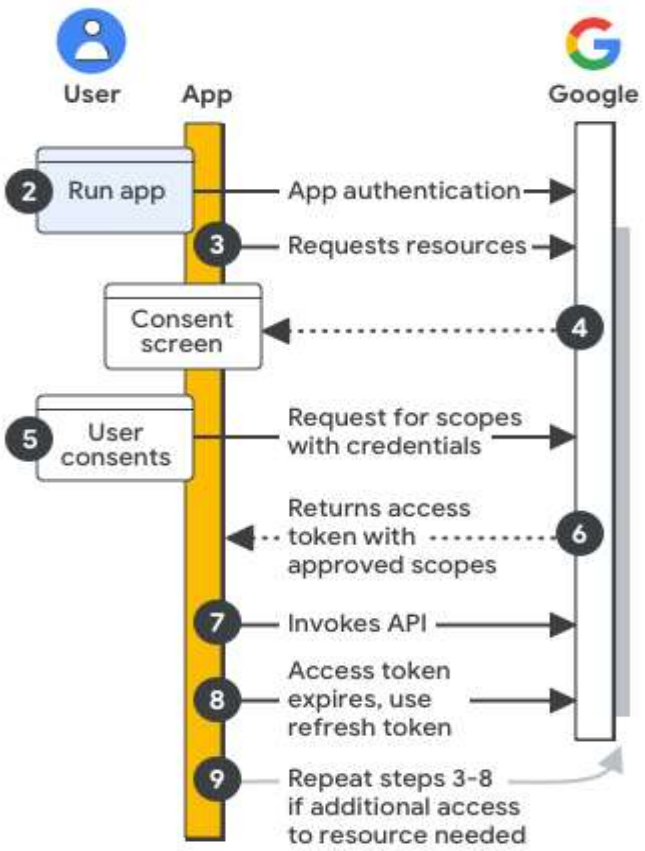


Рисунок 2.4 — Загальні етапи реалізації аутентифікації та авторизації

У консолі Google Cloud огляд API Google Workspace показує безліч спільних завдань, які можна виконати в іншому місці в консолі Google Cloud. Всі API Google Workspace зібрані в одному місці, щоб було зручно керувати ними.

Після створення аккаунту Google Workspace необхідно створити проект Google Cloud. Проект Google Cloud потрібен для використання API Google Workspace та створення додатків або програм. Цей проект формує основу для створення, включення та використання всіх сервісів Google Cloud, включаючи керування API, включення виставлення рахунків, додавання та видалення співавторів та керування дозволам та підключити необхідне API, у нашому випадку це Admin SDK та Google Calendar., після чого відбувається підключення алгоритму авторизації користувача.

Аутентифікація та авторизація — це механізми, які використовуються для перевірки особистості та доступу до ресурсів відповідно.

Загальний метод реалізації аутентифікації та авторизації використовуючи Google API:

— налаштування проекту і програми Google Cloud: під час розробки користувач реєструє свою програму в консолі Google Cloud, визначаючи області авторизації та облікові дані доступу для аутентифікації програми за допомогою ключа API, облікових даних кінцевого користувача або облікових даних облікового запису служби;

— аутентифікуйте програму для доступу: коли програма запускається, зареєстровані облікові дані доступу оцінюються і якщо програма автентифікується як кінцевий користувач, може з'явитися запит на вхід;

— запит ресурсів: коли додаток потребує доступу до ресурсів Google, він запитує у Google відповідні області доступу, які додаток зареєстрував раніше;

— запросити згоду користувача: якщо програма автентифікується як кінцевий користувач, Google відображає екран згоди OAuth, щоб користувач міг вирішити, чи надавати додатку доступ до запитаних даних;

— надіслати схвалений запит на ресурси: якщо користувач погоджується з областями доступу, програма об'єднує облікові дані та схвалені користувачем області доступу до запиту та надсилається на сервер авторизації Google для отримання токена доступу;

— Google повертає токен доступу: токен доступу містить список областей доступу тобто якщо список областей, що повертається, більш обмежений, ніж запитані області доступу, програма відключає всі функції, обмежені токеном;

— доступ до запитаних ресурсів: програма використовує токен доступу від Google для виклику відповідних API та доступу до ресурсів;

— отримання токену оновлення (необов'язково): якщо програма потребує доступу до API Google після закінчення терміну дії одного токена доступу, вона може отримати токен оновлення;

— запит додаткових ресурсів, тобто якщо потрібний додатковий доступ, програма запитує у користувача надання нових областей доступу, що призводить до нового запиту на отримання токена доступу (кроки 3—6).

Для підключення календарю Outlook у додаток, Outlook PIA буде використовуватися як метод інтеграції використовуючи локальний клієнт Microsoft Outlook. При наявності відповідного клієнту додаток буде використовувати його для створення нових подій у календарю користувача та завантажувати вже існуючі події.

Для інтеграції Google API потрібно створити підключити JSON файл ствроенний для проекту Google Cloud у якому зберігаются данні для ідентифікації як користувачів, так і відповидніх доступів.

Для підключення користувача потрібно створити сервісний аккаунт який буде виконувати всі дії викликані через додаток.

Після встановлення необхідних данних при виклику функції, додаток повинен створювати усі події передані йому, у відповідному Календарі Google та завантажувати з нього майбутні події.

Принцип роботи синхронізації календарю Outlook дуже схожий на аналог із Google, але буде використовуватися локальний клієнт Outlook та відповідний його користувач. Також по виклику відповідних функції створення події та завантаження, буде відбуватися синхронізація із календарем Outlook.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПЛАТФОРМИ

## 3.1 Варіантний аналіз і обґрунтування вибору програмних засобів

### 3.1.1 Вибір мови програмування

Перш ніж почати створювати програмні засоби, вам слід переглянути доступні програмні засоби та вибрати ті, які найкраще підходять для розробки вашої програми.

Мови програмування, такі як C++, Python, Java і C#, будуть перевірені для розробки.

Створена в 1979 році Б'ярне Страуструпом, C++ є мовою програмування високого рівня, яка використовує скомпільовану форму. Останньою версією мови є C++17, яка була випущена в 2017 році. C++ є надмножиною мови C. Він має той самий синтаксис і базові концепції, що й його батьківська мова, тобто він є частиною ширшої родини C. C++ підтримує об'єктно-орієнтовану парадигму, скомпільовано в машинний код для підвищення продуктивності та містить багато сторонніх бібліотек для зручності розробки. Оскільки мова програмування не має автоматичного керування пам’яттю, це дозволяє програмістам створювати небезпечні програми, які призводять до витоку пам’яті. Це призводить до втрати програмних даних, які зазвичай автоматично стираються, коли вони не використовуються. Як наслідок, мови програмування з автоматичним керуванням пам’яттю є кращими.

C# — це мова програмування високого рівня, розроблена Андерсом Гейлсбергом, Скоттом Вільхамутом і Пітером Голдом з Microsoft Corporation. Перша версія мови була випущена в 2002 році. Остання версія C# 7.0 була випущена 7 березня 2017 року. На мову програмування C# вплинули C++ і Java, і вона має синтаксис, подібний до C. Вона підтримує парадигму об’єктно-орієнтованого програмування. Він трохи повільніший за C++, але швидший за Java. Відрізняється наявністю великої кількості сторонніх бібліотек, автоматичним налаштуванням пам'яті і вбудованою реалізацією деяких шаблонів проектування.

Java — мова програмування високого рівня, випущена компанією Sun Microsystems у 1995 році. Остання версія Java Standard Edition 10 була випущена 20 березня 2018 року. Вона має синтаксис, схожий на C. Вона підтримує парадигму об’єктно-орієнтованого програмування, має трохи меншу продуктивність порівняно з C++ і C#. Характеризується наявністю великої кількості сторонніх бібліотек. Java використовує автоматичний збирач сміття для керування пам’яттю протягом життєвого циклу об’єкта. Програміст вирішує, коли створювати об’єкти, а віртуальна машина відповідає за звільнення пам’яті, коли об’єкт більше не потрібен. Коли на об’єкт більше немає посилання, збирач сміття може автоматично видалити його з пам’яті.

Python — це мова програмування високого рівня, яка є об’єктно-орієнтованою, інтерпретується та реалізована на об’єктному рівні; він має функції динамічного набору тексту. Його розробку розпочав у 1990 році Гвідо ван Россум. Він забезпечує швидку розробку додатків завдяки використанню високорівневих структур даних і з’єднувальних компонентів за допомогою динамічного зв’язування та семантики. Мова програмування Python має ряд різних парадигм програмування. До них відносяться функціональні, процедурні, аспектно-орієнтовані та об'єктно-орієнтовані мови. Він також підтримує модулі та пакети модулів, які допомагають розробникам створювати більше багаторазового коду та програм. Python — це інтерпретована мова з динамічними типами, що робить її ідеальною для швидкої розробки програм. Вона має високорівневі структури даних і простий метод об’єктно-орієнтованого програмування, що робить її кращою за інші мови програмування для створення швидких програм. Однак програми Python працюють трохи повільніше, ніж скомпільовані мови.

За результатами огляду для програмування була обрана мова C#, на якій будуть створені всі компоненти. Це виправдано потужністю цієї мови програмування і зручністю роботи з об'єктно-орієнтованими програмами та підтримкою API Google та необхіодного Calendar API, PAI для роботи з Microsoft Outlook

### 3.1.2 Вибір середовища розробки

IDE (інтегровані середовища розробки) використовуються, щоб зробити розробку програмного забезпечення більш зручною.

Інтегроване середовище розробки — це набір засобів розробки для розробки, що складається з редактора текстового коду, засобів для компіляції та налагодження проектів.

Щоб порівняти середовища розробки, JetBrains CLion, Microsoft Visual Studio та Visual Studio Code розглядатимуться для подальшого вибору.

Visual Studio Code — текстовий редактор, призначений для створення програм для хмарних систем і веб-додатків. Visual Studio Code поширюється безкоштовно і є кросплатформенним, тобто підтримує всі сучасні операційні системи.

VS Code підтримує багато плагінів, які можна легко з ним інтегрувати, але він більше підходить для написання коду на інтерпретованих мовах програмування, тому працювати з VS Code буде не дуже зручно.

Microsoft Visual Studio — це інтегроване середовище розробки, розроблене Microsoft. Він підтримує такі мови програмування, як C, C++, C#, F#, VisualBasic та інші. Це середовище підходить лише для операційної системи Windows.

CLion — це середовище розробки C і C++, розроблене JetBrains. Він також є кросплатформним і підтримує всі сучасні операційні системи.

Середовище Microsoft Visual Studio було обрано за його доступність, підтримку всього необхідного для розгортання, включаючи .Net 5.0 Framework.

## 3.2 Створення підключеня між API і клієнтом

Створивши обліковий запис Google Workshop та підключив домен відбувається створення проекту, сервісного облікового запису, підключення необхідних API, налаштування доступів та методів аутентифікації Oauth та сервісного акаунту.

### 3.2.1 Створення та налаштування проекту для підлключення API

Відкрив Google Cloud як супер адміністратор з аккаунту Google Workspace потрібно перейти у меню Ресурсів, де натиснувши кнопку Створити проект користувач вводить ім’я проекту. Приклад існуючого проекту зображено на рисунку 3.1 та процесу створення на рисунку 3.2

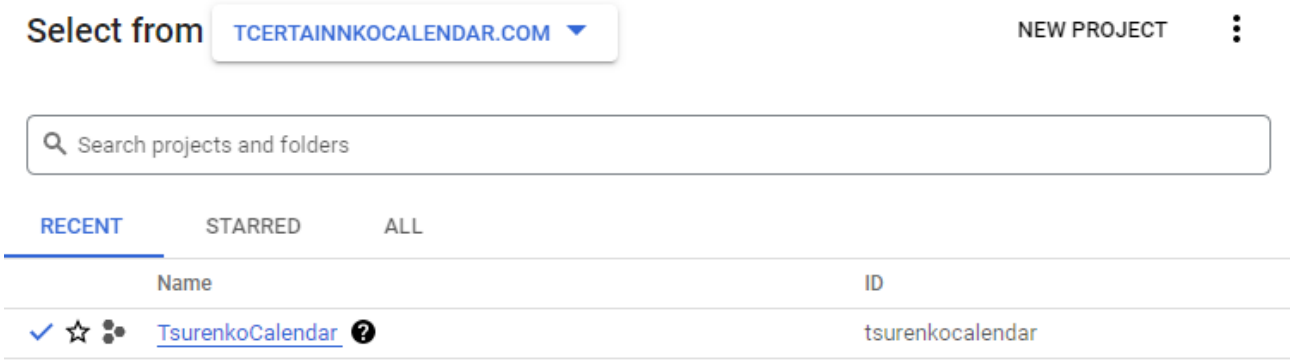


Рисунок 3.1 — Вікно перегляду існуючих проектів Google Cloud

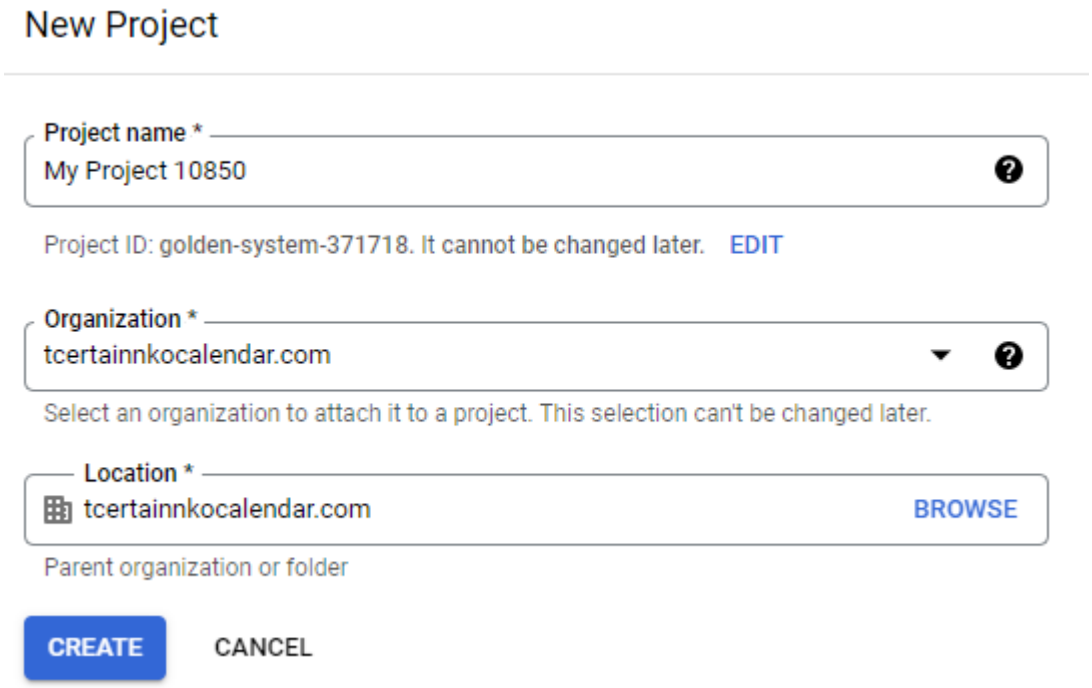


Рисунок 3.2 — Створення проекту в Google Cloud

Після створення користувачу необхідно перейти у меню підключення API які знаходяться за напрямком API&Сервіси, де користувач отримає повний

перелік доступних API на платформі. Необхідно знайти потрібні, а саме Admin SDK та Google Calendar API як зображено на рисунку 3.3.

Для підключення додатку до проекту необхідно налаштувати OAuth проекту перейшовши у вікно OAuth згоди, де вибрати тип користувача внутрішній, створити, ввести ім’я додатку, пошту для зв’язку із підтримкою додатку та контактну інформації розробника. Приклад на рисунку 3.4.

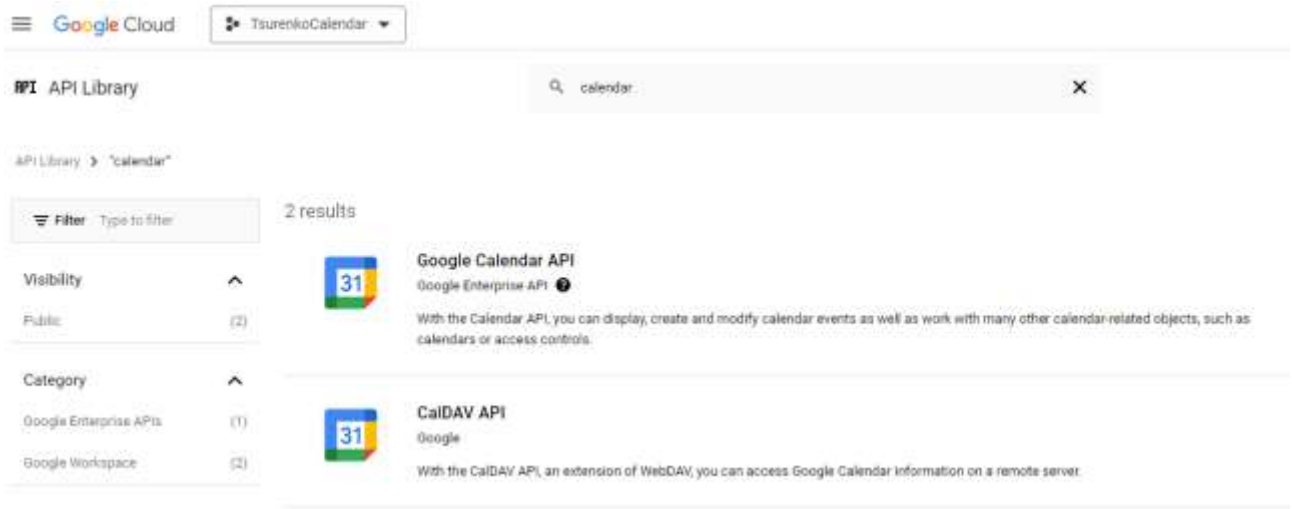


Рисунок 3.3 — Підключення вибраного API

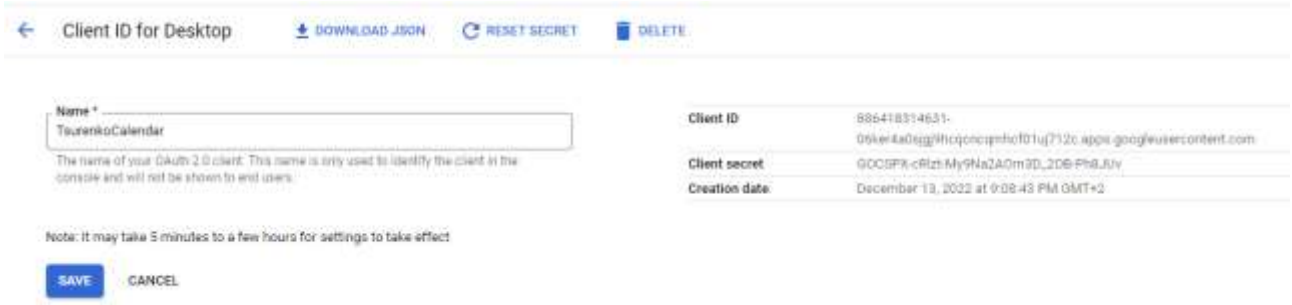


Рисунок 3.4 — Створення інформації для ідентифікації додатку

Налаштувавши JSON файл його потрібно завантажити і він буде використан у майбутньому для підключення додатку до проекту. А саме під час процесу створення GoogleCalendarClient.

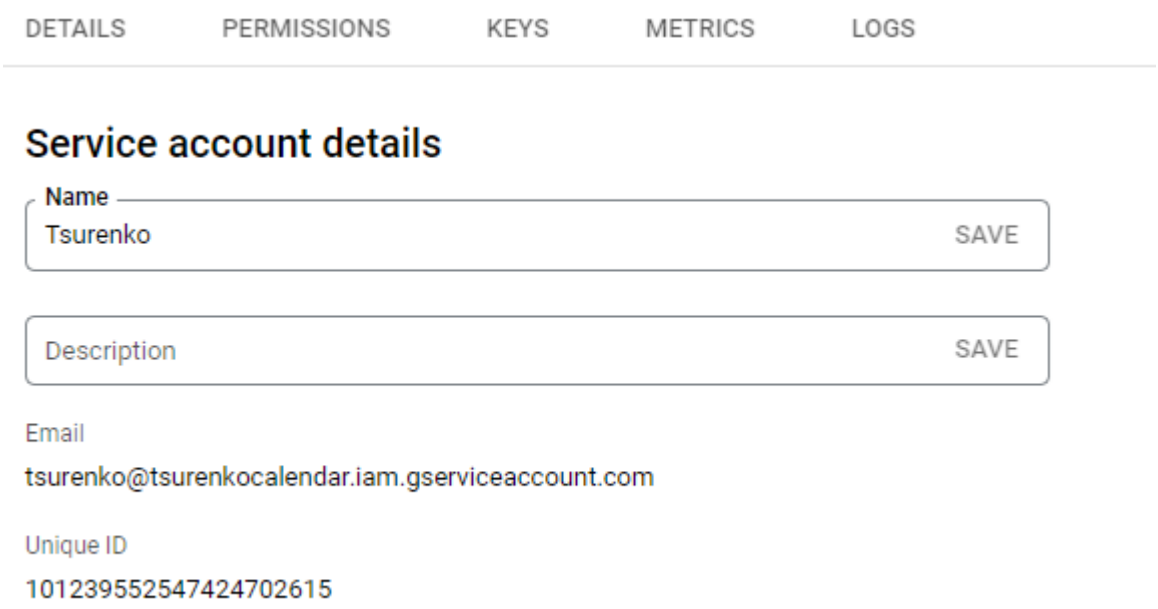


Рисунок 3.5 — Приклад сервісного запису для роботи с додатком

Для роботи додатку між користувачем та аккаунтом Google, а сама зв’язку з календарем, створюється сервісний аккаунт, який вказано на рисунку 3.5, якому надається доступ до усіх необхідних API та зберігається його контактні данні та JSON file з ним.

### 3.2.2 Налаштування Admin SDK та контролю за данними

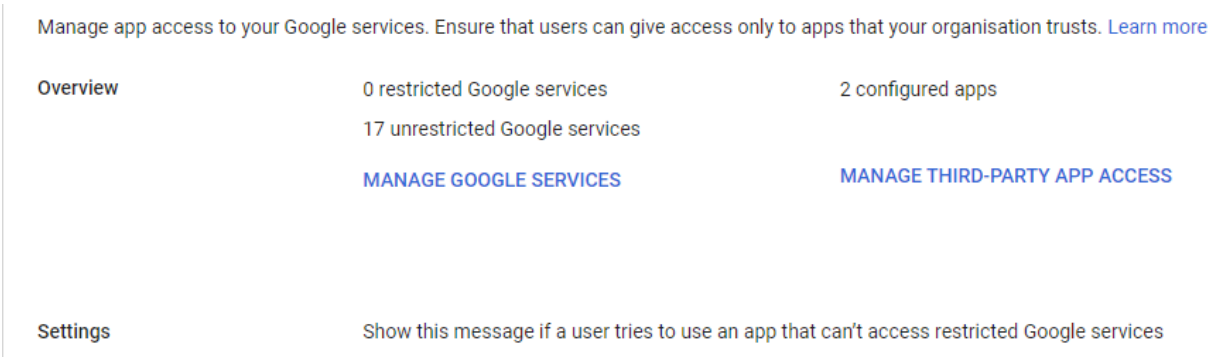


Рисунок 3.6 — Вікно налаштування доступу  додатку до сервісів

Створивши проект та налаштував його необхідно підключити доступ до календарю API та роботи сервера. Це виконується у вікні Безпека — дозвіли та контроль даних, що зображено на рисунку 3.6.

Натиснув «Manage Third-Party App Access» налаштовані програми відображаються за умовчанням. Можна переглянути: назва програми, тип, ID, перевірений статус та доступ.

Перевірений статус — Google перевірив перевірені програми на відповідність певним правилам. Багато відомих програм можуть не перевірятися таким чином.

Доступ — указує на надійний, обмежений або заблокований доступ проекту до API.

У вікні «Configured apps» можливо керуввати доступом програми до служб Google. Переглянути інформацію про програму ідентифікатор клієнта OAuth2 програми, кількість користувачів, політику конфіденційності та інформацію про підтримку. API служби Google (області OAuth), які запитує програма. Переглянути список областей OAuth, які запитує кожна програма.

### 3.2.3 Налаштування делегування домену

Облікові дані використовуються для отримання маркера доступу з серверів авторизації Google, щоб ваша програма могла викликати API Google Workspace. У цьому посібнику описано, як вибрати та настроїти облікові дані, необхідні для вашої програми.

Необхідні облікові дані залежать від типу даних, платформи та методу доступу до програми. Доступні три типи облікових даних:

Ключ API — ці облікові дані для анонімного доступу до загальнодоступних даних у програмі.

Ідентифікатор клієнта OAuth —ці облікові дані для автентифікації як кінцевого користувача та доступу до даних. Потрібно, щоб додаток запитував та отримував згоду від користувача.

Обліковий запис служби. Ці облікові дані для автентифікації як обліковий запис служби роботів або для доступу до ресурсів від імені користувачів Google Workspace або Cloud Identity за допомогою делегування на рівні домену. Приклад зображено на рисунку 3.7.

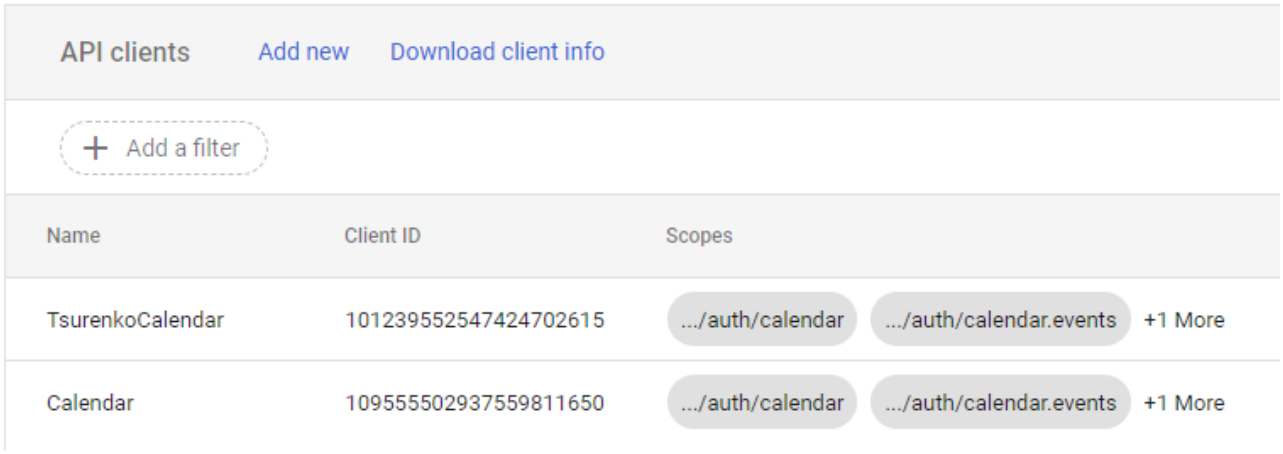


Рисунок 3.7 — Створення делегування домену

У випадку додатку календарю необхідно використовувати Ключ API з делегаціями для роботи додатку з данними Google Calendar:

— https://www.googleapis.com/auth/calendar,

— https://www.googleapis.com/auth/calendar.events,

— https://www.googleapis.com/auth/calendar.events.readonly

## 3.3 Розробка програмних модулів системи

Для ефективної розробки додатку процес розділили на кілька етапів. З самого початку був розроблений зовнішній вигляд головного вікна програми, розташовані та розташовані кнопки інтерфейсу. Було створено:

— 42 радіоперемикачі по днях тижня;

— відображаються кнопки для створення подій і перемикання місяців;

— додано дві текстові стрічки для назви місяця та поточного часу;

— 20 кнопок для подій і 20 стрічок для їх опису;

— кнопку виклику вікна методів синхронізації подій з Google Calendar та Microsoft Outlook.

Вікно до ініціалізації проекту має вигляд який вказано на рисунку 3.8

За допомогою класу DateTime поточний час подається на текстовий рядок у форматі день-місяць-рік години-хвилини-секунди, а індикатор європейського часу — для дня або ранку.

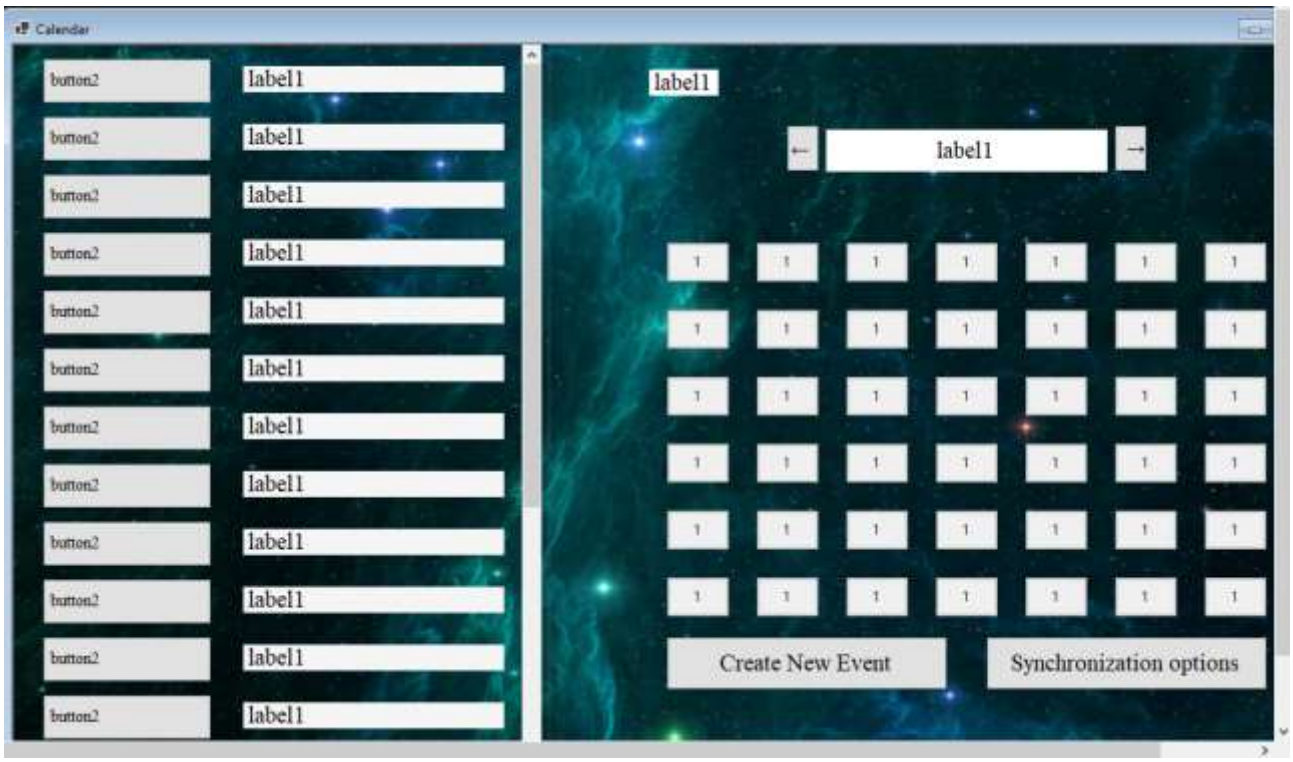


Рисунок 3.8 — Головне вікно програми до виконання

labeltime.Text = DateTime.Now.ToString("dd-MMM-yyyy hh:mm:ss tt", new System.Globalization.CultureInfo("en-EN"));

За допомогою годинника була створена подія, яка відбувається щосекунди в реальному часі, так що час на головному екрані оновлюється щоразу, коли ця подія виконується, і, таким чином, час актуальний.

Під час початкового створення головного вікна створюються масиви кнопок, текстів і перемикачів для подальшого виконання програми які зображено на рисунку 3.9

Після створення метод setdates виконується при ініціалізації додатку. Цей модуль відтворює перемикачі в зручному та легкому для читання користувачеві форматі.

З самого початку метод переходить від теперішнього моменту до першого дня місяця, перевіряє, який це день тижня, потім циклічно переглядає вигляд і назви всіх перемикачів, створюючи звичайний календар.

Крім того, зі створенням календаря до перемикачів додається функціональність методу daydates.

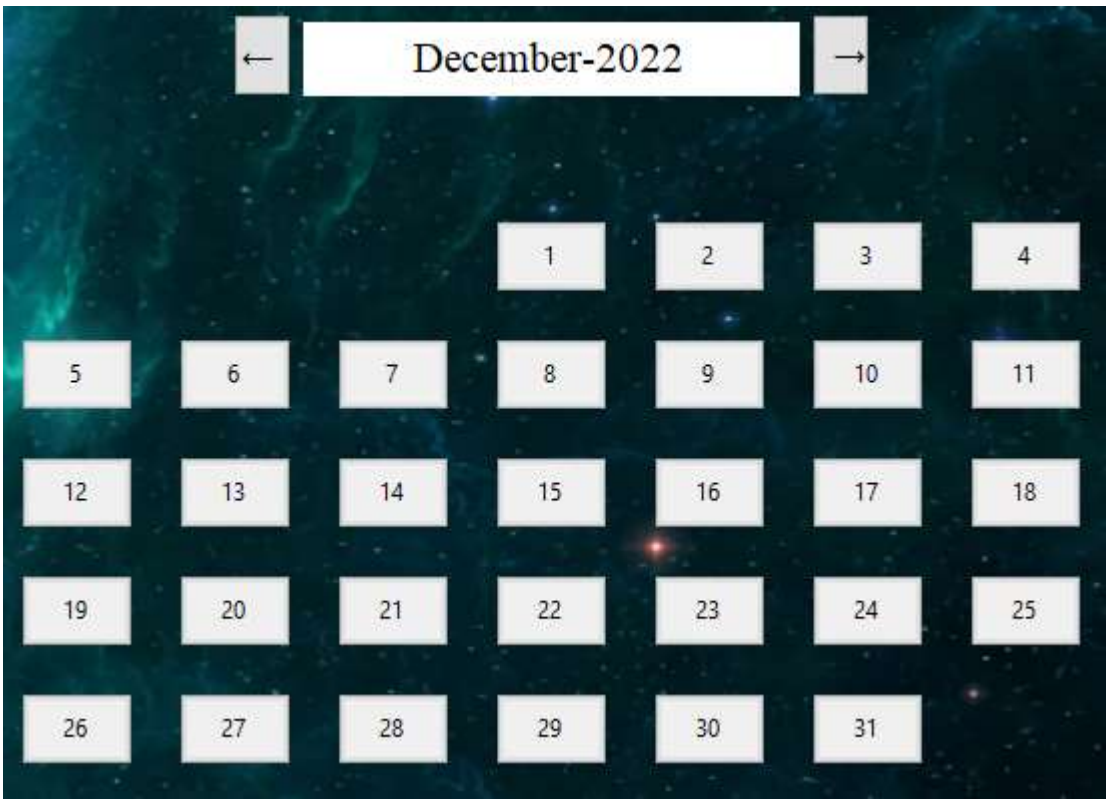


Рисунок 3.9 — Використання setdates для утворення календарю

Лістинг 3.1 — Функція setdates

```
private void setdates()
        {
            for (int j=0;j<42;j++)
            {
                rbArray[j].Visible = false;
            }
            var tempdate = new DateTime();
            tempdate = date;
            do
            {
                tempdate=tempdate.AddDays(-1);
            }
            while (tempdate.Day != 1);
            int i = 6;
```

```
        if ((int)tempdate.DayOfWeek == 2) { i = 1; }
        if ((int)tempdate.DayOfWeek == 3) { i = 2; }
        if ((int)tempdate.DayOfWeek == 4) { i = 3; }
        if ((int)tempdate.DayOfWeek == 5) { i = 4; }
        if ((int)tempdate.DayOfWeek == 6) { i = 5; }
        if ((int)tempdate.DayOfWeek == 1) { i = 0; }
        do
        {
            rbArray[i].Text = tempdate.Day.ToString("d");
            rbArray[i].Tag = tempdate.Day;
            rbArray[i].CheckedChanged += radioButton_CheckedChanged;
            rbArray[i].Click += radioButton_Click;
            rbArray[i].Visible = true;
            i++;
            tempdate = tempdate.AddDays(1);
        }
        while (tempdate.Day != 1);
    }
```

Після створення зручного календаря з файлу завантажуються наявні події та виконується метод Monthdates.

Метод Monthdates перевіряє всі події в списку на збіг року і місяця з поточним, якщо в цьому місяці є події, вони будуть відображатися у вигляді дати і опису в лівій частині вікна. , завдяки якому користувач може побачити всі події, які були або будуть у певному місяці.

Приклад списку існуючих у додатку подій, створених користувачем, зображено на рисунку 3.10

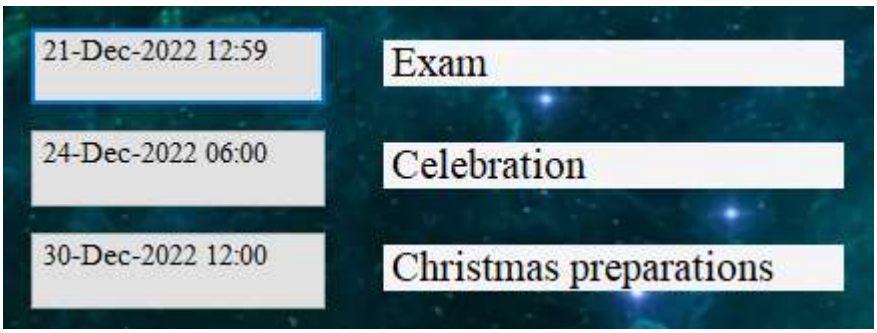


Рисунок 3.10 — Приклад вигляду розділу подій

Лістинг 3.2 — Функція monthdates

```
public void monthdates()
    {
        for (int arrc1 = 0; arrc1 < 20; arrc1++)
        {
            bArray[arrc1].Visible = false;
            lArray[arrc1].Visible = false;
        }
        labelExtraSize.Visible = false;
        int arrc = 0;
        int eventn = 0;
        foreach (Event Event in myEvents)
        {
            if ((Event.Edate.Month == date.Month) && (Event.Edate.Year == date.Year) && (arrc < 20))
            {
                bArray[arrc].Text = Event.Edate.ToString("dd-MMM-yyyy hh:mm tt", new System.Globalization.CultureInfo("en-EN"));
                bArray[arrc].Visible = true;
                bArray[arrc].Click -= buttonll_Click;
                bArray[arrc].Click += buttonll_Click;
                lArray[arrc].Text = Event.Desc;
```

```
                lArray[arrc].Visible = true;
                bArray[arrc].Tag = eventn;
                arrc++;
            }
            eventn++;
        }
        if (bArray[19].Visible == true)
            labelExtraSize.Visible = true;
    }
```

Щоб створити подію, скористайтеся кнопкою під календарем. Після натискання з'являється нове вікно, яке зображено на рисунку 3.11, в якому можна ввести дату і час події, її опис, після чого вона зберігається в списку.

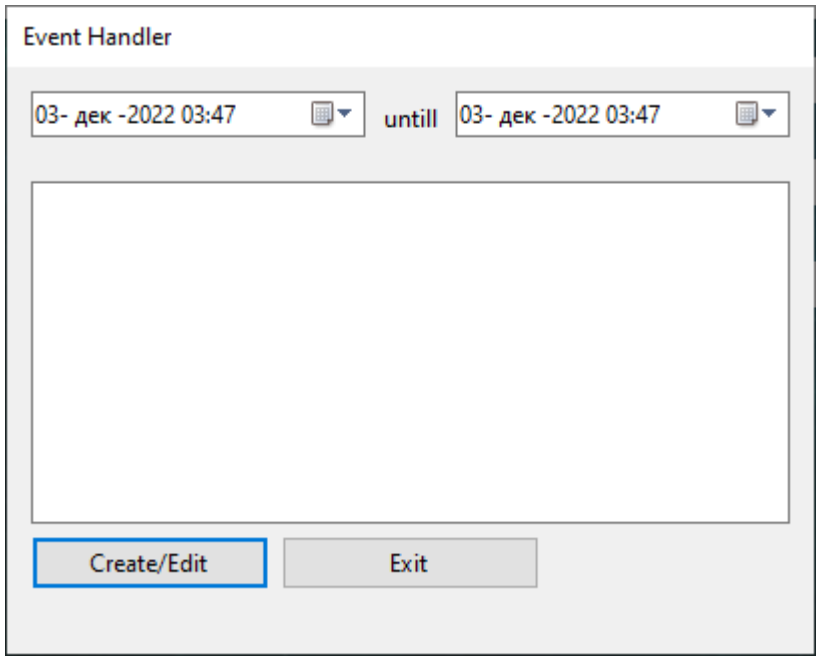


Рисунок 3.11 — Вікно обробки події

Команда для виклику вікна нової події:

```
Form2 frm = new Form2(myEvents,false,0);
frm.Show();
```

Якщо виклик відбувається через кнопку існуючої події на головному вікні, то можливо відредагувати цю подію або видалити її як зображено на рисунку 3.12

```
int bn =  (int)(sender as Button).Tag;
Form2 frm = new Form2(myEvents, true, bn);
frm.Show();
```

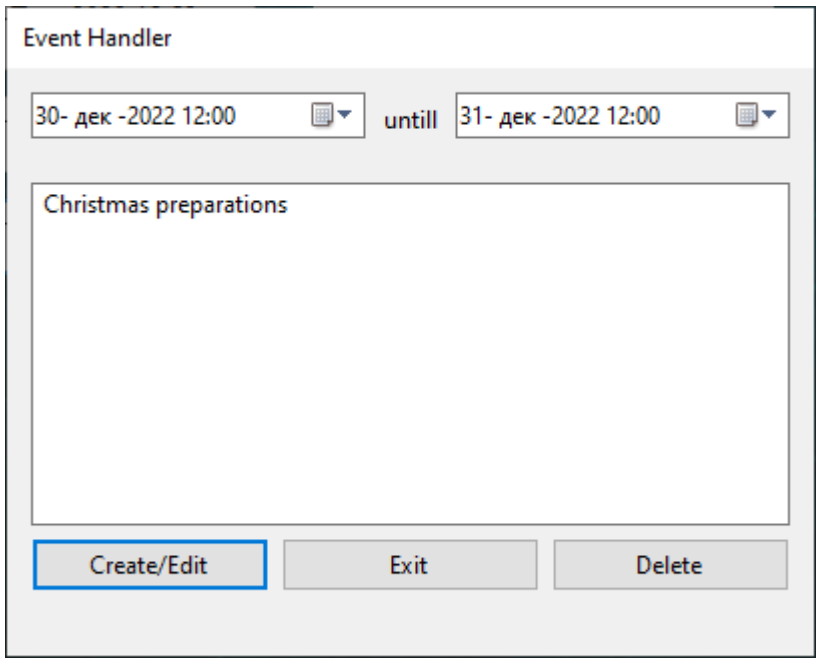


Рисунок 3.12 — Вікно обробки існуючої події

При виклику вікна передається інформація про список подій, підтвердження що це не нова подія та номер події зі списку.

Для додавання події використовується функція:

```
myEvents.Add(new Event() {Edate= dateTimePicker1.Value, Desc = textBox1.Text});
```

Де dateTimePicker1.Value це значення часу виставленого у вікні, textBox1.Text це текст введений у описі події.

Після виконання дії з подіями список подій серіалізується, який фактично зберігається у файлі file.bin у директиві з програмою.

Серіалізація виконується за допомогою команд представлених у лістингу 3.3.

Лістинг 3.3 — Виконання серіалізації

```
if (File.Exists("file.bin"))
        {
           Stream stream = null;
           try
           {
              stream = File.Open("file.bin", FileMode.Open);
              BinaryFormatter bformatter = new BinaryFormatter();
              myEvents = (List<Event>)bformatter.Deserialize(stream);
           }
           finally
           {
              if (stream != null)
                 stream.Close();
           }
```

Де myEvents це список подій.

Крім того, коли події виконуються, вони сортуються за датою командою sort.

```
myEvents.Sort((x, y) => DateTime.Compare(x.Edate, y.Edate));
```

Кожній кнопці, відповідальній за подію, присвоюється тег з номером події, завдяки якому програма визначає, яку подію потрібно видалити із списку користувача.

При створенні календаря на кожну видиму на екрані кнопку встановлюється функція перевірки, чи натиснув користувач на вибрану дату. Якщо так, то в меню подій відображатимуться лише події, які відбуваються в цей день, що видно на рисунку 3.13

Лістинг 3.4 — Перевірка натиску перемикання дати

```
private void radioButton_CheckedChanged(object sender, EventArgs e)
```

```
{
        isChecked = (sender as RadioButton).Checked;
        if ((sender as RadioButton).Checked == true)
        {
           int rn = (int)(sender as RadioButton).Tag;
           dayates(rn);
        }
        else
           monthdates();
```

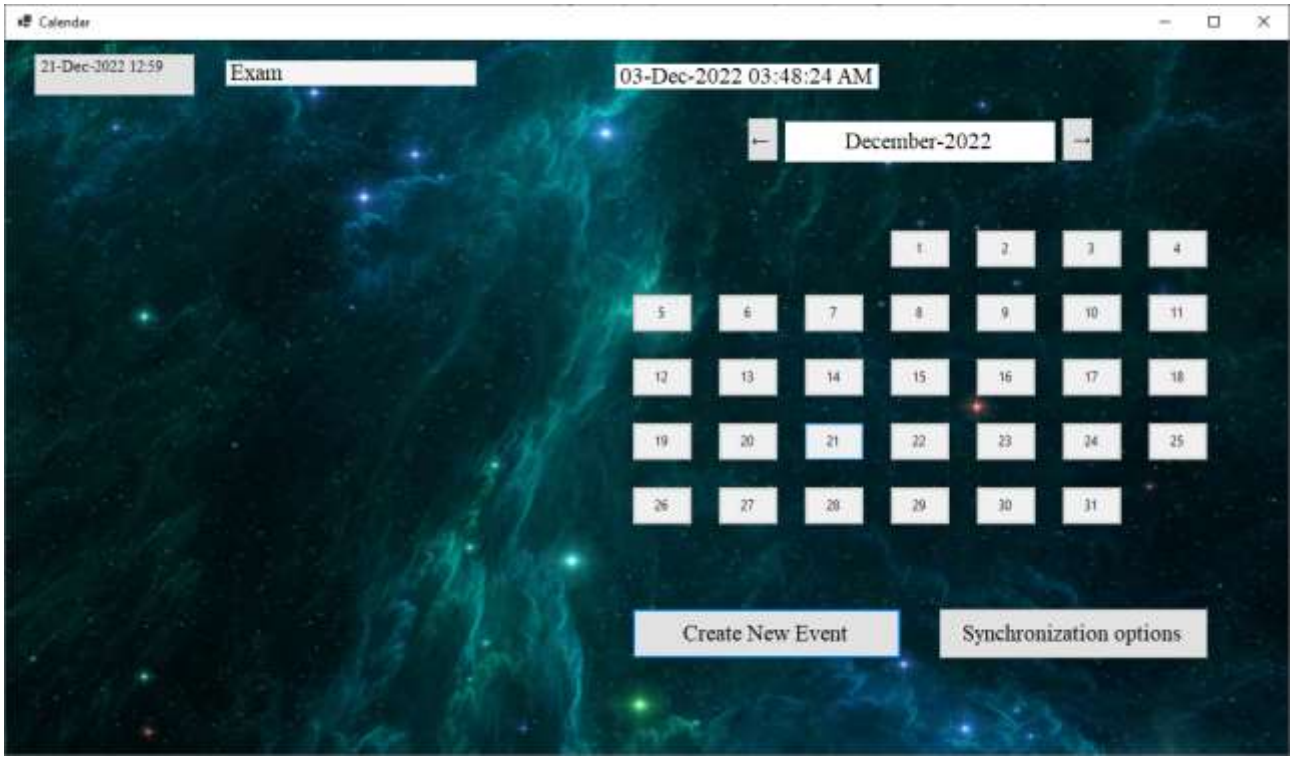


Рисунок 3.13 Головне вікно натиснувши на перемикач в день с подією

Виконується методом daydates якому передають номер дня місяцю який було викликано Приклад роботи зображено на рисунку 3.14

Метод daydates виконує функціонал близький до monthdates, але перевіряє ще день події в списку, того відображаються тільки події яким встановлено цей день.

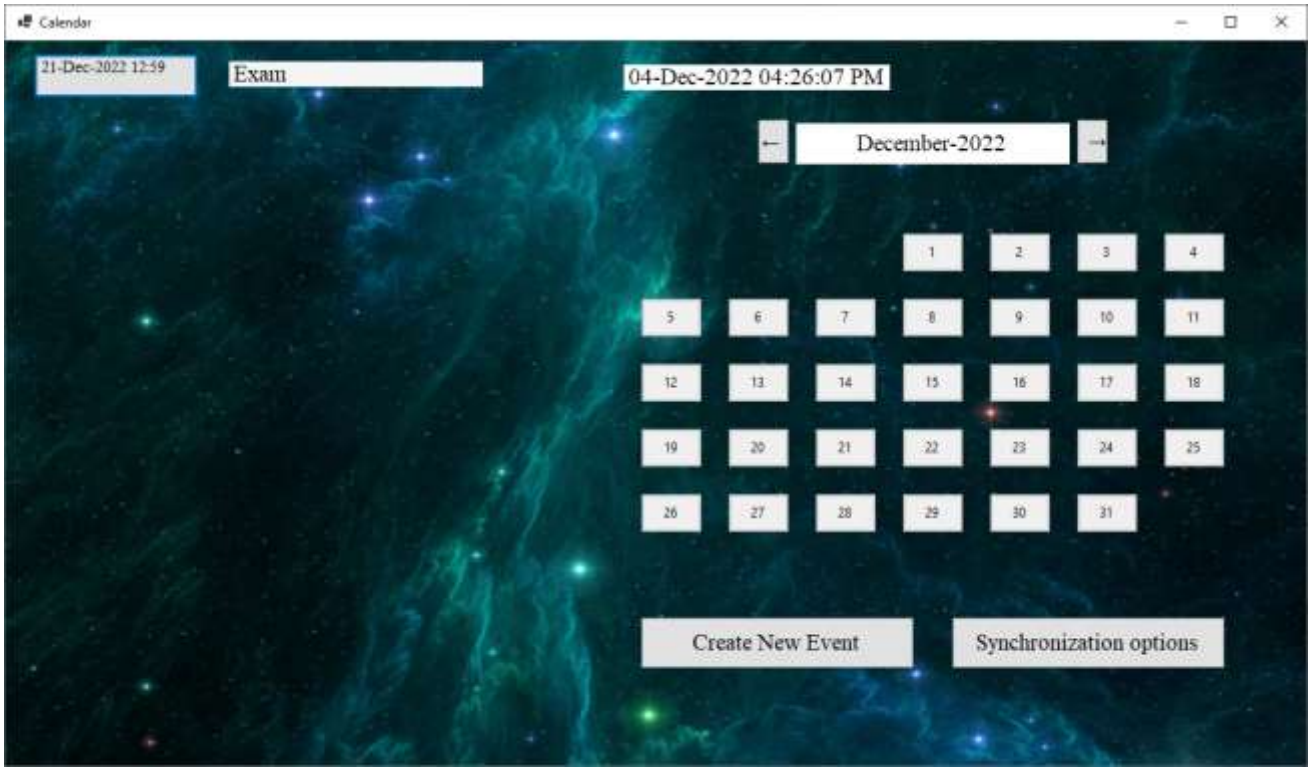


Рисунок 3.14 Головне вікно натиснувши на перемикач в день с подією

Використовуючи функцію таймера, яка кожну секунду оновлює час у головному вікні, було розроблено метод сповіщення користувача про настання часу однієї з подій.

Кожну секунду всі події перевіряються на час їх спрацьовування, якщо поточний час збігається з часом події, виводиться вікно з описом події користувача.

Лістинг 3.5 — Цикл перевірки нагадування про подію

```
foreach (Event Event in myEvents)
                {
                    if ((Event.Edate.Year == DateTime.Now.Year) && (Event.Edate.Month == DateTime.Now.Month) && (Event.Edate.Day == DateTime.Now.Day) && (Event.Edate.Hour == DateTime.Now.Hour) && (Event.Edate.Minute == DateTime.Now.Minute) && (Reminder == false))
                    {
                        Reminder = true;
```

```
                    MessageBox.Show(Event.Desc, "Event time",
MessageBoxButtons.OK);
                }
            }
```

Окрім перевірки часу, він також перевіряє, чи була подія запущена протягом останньої хвилини. Якщо ні, вікно викликається, а змінна, що відповідає за перевірку, встановлюється на true.

Якщо керуюча змінна має значення true, лічильник збільшується щосекунди, поки не досягне значення за хвилину.

Лістинг 3.6 — Лічильник нагадування

```
if (Reminder == true)
        {
          RemCoun++;
          if (RemCoun == 59)
          {
             Reminder = false;
             RemCoun = 0;
          }
        }
```

Для роботи з Google Calendar API необхідно завантажити і підключити nuget пакет бібліотек.

Перейдіть до Інструменти >Диспетчер пакетів NuGet >Консоль диспетчера пакетів де потрібно ввести

Install-Package Google.Apis.Calendar.v3 -Version 1.40.3.1692

Підключив потрібні бібліотеки необхідно занести JSON файл tsurenkocalendar-c7c41057d5a5 у кореневу папку додатку що буде використано у функкції створення календарю як об’єкта для роботи.

CalendarService _service = this.GetCalendarService("tsurenkocalendar-c7c41057d5a5.json");

Функція створення календарю GetCalendarService типу CalendarService приймає JSON файл як параметр keyfilepath.

Після створення об’єкту календар з яким буде працювати додаток оголошується змінні представлені у лістингу 3.7.

Лістинг 3.7 — Перелік дозволів для інтеграції календарю

```
string[] Scopes = {
                CalendarService.Scope.Calendar,
                CalendarService.Scope.CalendarEvents,
                CalendarService.Scope.CalendarEventsReadonly };
               GoogleCredential credential;
```

Де Scopes це список дозволів які необхідні додатку для роботи та які він буде запрошувати у користувача.

Далі додаток відкриває JSON файл з інформацією який було створено для ідентифікації сервісного облікового запису під час роботи у Google Cloud та Google Workspace.

Лістинг 3.8 — Відкриття JSON файлу додатком

```
using (var stream = new FileStream(keyfilepath, FileMode.Open, FileAccess.Read))
                {
                 credential = GoogleCredential.FromStream(stream)
.CreateScoped(Scopes).CreateWithUser("tsurenko@tsurenkocalendar.iam.gserviceaccount.com");
                }
```

Разом с файлом оголошується що сервісний акаунт, tsurenko@tsurenkocalendar.iam.gserviceaccount.com, який було створено, буде використан для створення подій.

Також необхідно підключити HTTP сервер, який створили раніше, він також оголошен у файлі JSON.

Лістинг 3.9 — Підключення додаткових файлів

```
var service = new CalendarService(new BaseClientService.Initializer()
        {
           HttpClientInitializer = credential,
           ApplicationName = "TsurenkoCalendar",
        });
        return service;
```

Для того щоб відправити існуючу в календарі подію до Google календарю використовується метод CreateEvent якому як параметр передається створений об’єкт календарю.

private void CreateEvent(CalendarService _service)

Для відправлення усіх подій використовується цикл який працює з кожною існуючею подією.

foreach (EventLocal Event in myEvents)

У самому циклі створюється нова подія типу Event сумісного з Google Calenar API. Для цього визначено —

using Event = Google.Apis.Calendar.v3.Data.Event;

Кожна нова подія в циклі отримує дані отримані з відповідної отриманої події.

Лістинг 3.10 — Перетворювання події у формат для Google Calendar

```
Event body = new Event();
        EventDateTime start = new EventDateTime();
```

```
EventDateTime end = new EventDateTime();
body.Start = Event.Startdate.ToEventDateTime();
body.End = Event.Enddate.ToEventDateTime();
body.Summary = Event.Desc;
EventsResource.InsertRequest request = new EventsResource.InsertRequest(_service, body, "a3fed24596b55d9ebcd9f745028f48dd1ddeb4b91ad3f21486547e80e981709d@group.calendar.google.com");
Event response = request.Execute();
```

Де EventDateTime start — початок події Google,

EventDateTime end — кінець події

body.Start = Event.Startdate.ToEventDateTime() — перетворення часу події з формату DateTime у формат EventDateTime який сумісний з Google Calendar API.

body.Summary = Event.Desc — передача опису події.

Після того як подія отримала всі дані, відбувається запит EventsResource.InsertRequest який отримує параметри об’єтку календар з яким працювати, самою подією та календарем в Google який відправляється подія.

Для завантаження події використовується метод GetEvents з параметром об’єкту календарю.

Перше що робить цей метод, це створює список події які завантажувати.

Лістинг 3.11 — Завантаження подій з Google Calendar

```
EventsResource.ListRequest request = _service.Events.List("a3fed24596b55d9ebcd9f745028f48dd1ddeb4b91ad3f21486547e80e981709d@group.calendar.google.com");
request.TimeMin = DateTime.Now;
request.ShowDeleted = false;
request.SingleEvents = true;
request.MaxResults = 10;
```

request.OrderBy = EventsResource.ListRequest.OrderByEnum.StartTime;

Events events = request.Execute();

Перед відправленням запиту додаток повідомляє цей запит, що потрібно прислати події які:

request.TimeMin = DateTime.Now; — ще не відбулися,

request.ShowDeleted = false; — не видалені з календарю,

request.SingleEvents = true; — не повторюються,

request.MaxResults = 10; — та не більше 10-ти

Після чого request.OrderBy = EventsResource.ListRequest.OrderByEnum.StartTime; просортувати їх у порядку з найбижчої до останньої.

Після налаштування відбувається запит з відповідю.

Отримавши список подій додаток переглядає події на копії із існуючими в додатку, порівнюючи текст та початкову, якщо немає схожих то вона додається додається у список подій додатку, після чого зберігає файл з подіями.

Для роботи з Microsoft Outlook було використано бібліотеку Microsoft.Office з її функціоналом Interop, який дозволяє використовувати локальний клієнт Office для роботи, в нашому випадку використовувати Outlook.

В загальному принцип дії методів для завантаження та відправлення дуже схожі на API від Google, але у їх випадку для роботи не потрібно налаштовувати домен, сервер, дозволи та підключати файли.

Використовуючи метод AddAppointmentOutlook додаток створює подію яку передає у клієнт Outlook.

Лістинг 3.12 — Створення події Outlook

Microsoft.Office.Interop.Outlook.Application otApp = new Microsoft.Office.Interop.Outlook.Application();

```
Microsoft.Office.Interop.Outlook.AppointmentItem oAppointment
= (Microsoft.Office.Interop.Outlook.AppointmentItem) otApp.CreateItem
(Microsoft.Office.Interop.Outlook.OlItemType.olAppointmentItem);
oAppointment.Subject = Event.Desc;
oAppointment.Start = Event.Startdate;
oAppointment.End = Event.Enddate;
oAppointment.Save();
```

Функція oApp = new Microsoft.Office.Interop.Outlook.Application(); використовується для позначення календарю з яким буде працювати програма.

Коли AppointmentItem oAppointment це подія яку програма буде відправляти.

Створивши об’єкт з яким працювати, програма передає йому характеристики отримані з події додатку:

oAppointment.Subject = Event.Desc; — передача опису події, в данному випадку це буде темою події.

oAppointment.Start = Event.Startdate; — дата початку події.

oAppointment.End = Event.Enddate; — дата кінця події.

Сберігши інформацію про подію виконується команда oAppointment.Save(); яка зберігає цю подію у календарі Outlook.

Для завантаження події з календарю використовується функція GetAllCalendarItemsOutlook в якій використовуються метод CalendarFolder.

Лістинг 3.13 — Метод CalendarFolder

```
CalendarFolder =
mapiNamespace.GetDefaultFolder(Microsoft.Office.Interop.Outlook.OlDefaultFolde
rs.olFolderCalendar);
outlookCalendarItems = CalendarFolder.Items;
outlookCalendarItems.IncludeRecurrences = false;.
```

CalendarFolder використовується для читання списку календарів користувача, в нашому випадку програма читає базовий календар користувача з якого і буде завантажено всі події.

outlookCalendarItems — список подій які програма отримає с завантаженого календарю.

outlookCalendarItems.IncludeRecurrences = false; — робить щоб програма не оброблювала повторювані події.

Далі програма виноує ідентичний цикл завантаження подій та перевірки на повтори що були описані у завантажені подій с Google Calendar, сбереження та запис у відповідний файл.

Для роботи програми було створено окреме вікно яке викликається через кнопку Synchronization options у правому нижньому кутку вікна яка викликає вікно яке зображено на рисунку 3.15

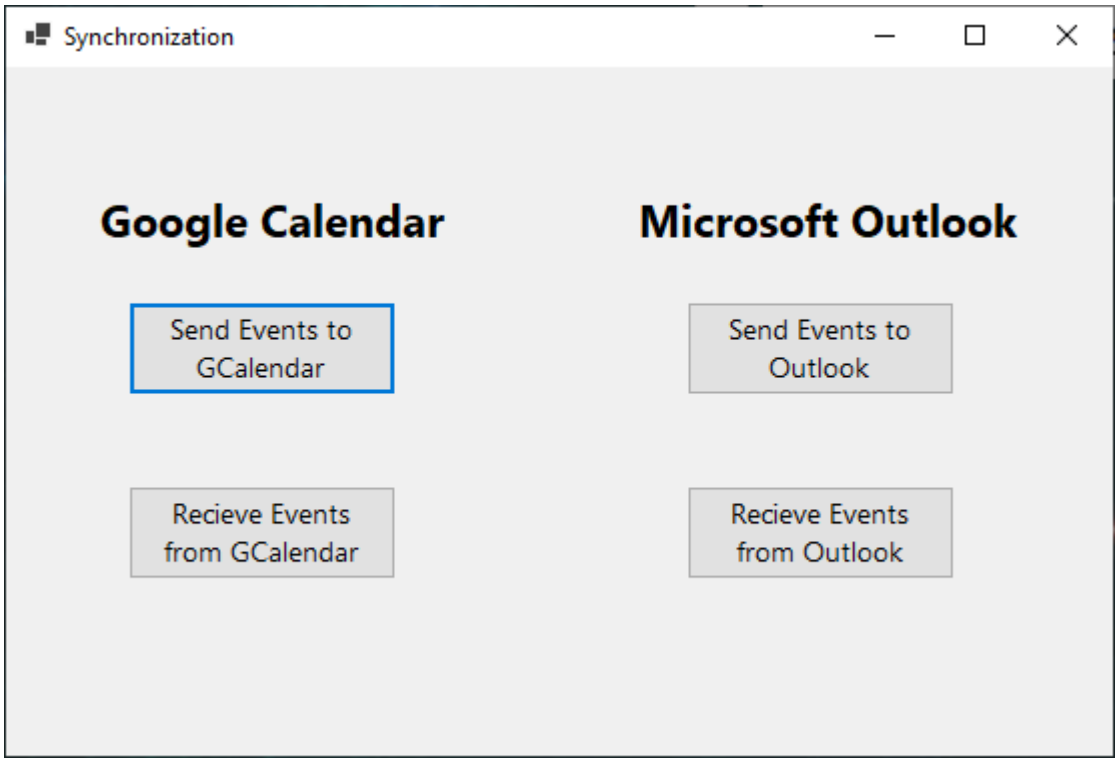


Рисунок 3.15 Вікно після натискання Synchronization options

У цьому вікні користувачеві надається вибір відправити чи завантажити файли до відповідно розділеного сервісу.

Приклад результатів роботи методів у цьому вікні зображено у додатку Е.

Використовуючи їх користувач має можливість:

— відправити події у Google Calendar;

— завантажити події з Google Calendar;

— відправити події до Microsoft Outlook;

— завантажити події з Microsoft Outlook.

Після натискання відповідних кнопок програма отримала подію з Google календарю на 24 грудня, а програма відповідно відправила подію Exam яка вже була у програмі и створила відповідну подію на вказаний час.

## 3.4 Тестування програмного модуля

Тестування — це процес аналізу та дослідження, який надає інформацію про якість продукту відносно умов, за яких він буде використовуватися. Методологія тестування також включає процес пошуку дефектів, помилок і несправностей.

Це також тест програмних компонентів для оцінки готовності програмного забезпечення до використання. Результат тесту оцінюється за такими критеріями:

— відповідність вимогам розробників і проектувальників;

— відповідність вихідних даних;

— допустима тривалість функції;

— практичність;

— відповідність вимогам замовника.

Кількість тестів навіть простих програмних компонентів може бути майже нескінченною, тому тактика тестування повинна полягати в проведенні лише необхідних тестів з урахуванням наявного часу та ресурсів. У результаті програмні засоби тестуються за допомогою стандартного виконання програми для виявлення помилок, помилок або інших дефектів.

Існує багато типів тестів: деякі зазвичай виконуються самими розробниками, інші — спеціалізованими групами. У нашому випадку будуть використовуватися системні тести.

Системне тестування — це виконання програмного забезпечення в його остаточній конфігурації, інтегроване з іншими програмними та апаратними системами.

Існує дві основні методики тестування: тестування «білої скриньки» та тестування «чорної скриньки».

Методика «білої скриньки», Об'єктом тестування тут є не зовнішня, а внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним. Зазвичай аналізуються керуючі зв'язки елементів, рідше—інформаційнізв'язки. Тестування за принципом «білої скриньки» характеризується ступенем, в якому тести виконують або покривають логіку (вихідний текст) програми.

Одним із способів вивчення поставленого питання є дослідження методики «чорної скриньки», Основне місце програми тестів «чорної скриньки» це інтерфейс ПЗ. Ці тести демонструють:

— як виконуються функції програми;

— як приймаються вихідні дані;

— працездатність синхронізації із зовнішніми даними

— зберігання завантажених даних.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Вона базується на використанні шаблонів тестування, бо ж тест-кейсів. Це означає що буде створено декілька ситуацій у яких перевіряється працездатність додатку, коректності основних функцій.

Результати проведення тестів методом чорної скриньки представлено у таблиці 3.1

Таблиця 3.1 — Тест-кейси

<table>
<tr><td rowspan="3">Код тест-кейса</td><td colspan="3">Опис тест-кейса</td></tr>
<tr><td colspan="2">Хід тестування</td><td>Очікуваний результат</td></tr>
<tr><td>Дата тестування</td><td>Результат</td><td>Примітка</td></tr>
<tr><td rowspan="2">001</td><td colspan="3">Перевірка створення коректності створення дат</td></tr>
<tr><td colspan="2">1. Запустити додаток<br>2. Перевести дату на рік перевіряючи місяці</td><td>Коректне виведення дат</td></tr>
<tr><td></td><td>5.12.2022</td><td>Пройдено</td><td>-</td></tr>
<tr><td rowspan="3">002</td><td colspan="3">Створення подій у різні місяці</td></tr>
<tr><td colspan="2">1. Запустити додаток<br>2. Створити події у різні місяці та роки<br>3. Перевірити відображення подій</td><td>Список подій відображається відповідно до місяця та року</td></tr>
<tr><td>5.12.2022</td><td>Пройдено</td><td>-</td></tr>
<tr><td rowspan="3">003</td><td colspan="3">Перевірка роботи запису даних у файл</td></tr>
<tr><td colspan="2">1. Запустити додаток<br>2. Створити події<br>3. Вимкнути додаток<br>4. Відкрити додаток<br>5. Перевірити наявність подій</td><td>Події з файлу загружено у додаток при ввімкнені</td></tr>
<tr><td>5.12.2022</td><td>Пройдено</td><td>-</td></tr>
<tr><td rowspan="3">004</td><td colspan="3">Перевірка можливості редагування та видалення подій</td></tr>
<tr><td colspan="2">1. Запустити додаток<br>2. Створити події<br>3. Редагувати події<br>4. Видалити події</td><td>Події успішно редаговані та видалені</td></tr>
<tr><td>6.12.2022</td><td>Пройдено</td><td>-</td></tr>
</table>

Продовження таблиці 3.1

| | | | |
|---|---|---|---|
| 005 | Перевірка можливості відправки подій | | |
| | 1. Запустити додаток<br>2. Створити події<br>3. Відправити події<br>4. Перевірити події у сервісі | | Події успішно відправлені і відображені у відповідному сервісі. |
| | 6.12.2022 | Пройдено | - |
| 006 | Перевірка можливості завантаження подій | | |
| | 5. Запустити додаток<br>6. Створити події у сервісі<br>7. Завантажити<br>8. Перевірити події у додатку | | Події успішно завантажені та відображені у додатку. |
| | 8.12.2022 | Пройдено | - |

3.5 Інструкція користувача

Для початку роботи необхідно запустити додаток. Користувачеві буде представлений інтерфейс, що складається з місячного розкладу. Є можливість переглянути поточний час, перейти на інші місяці. Після натискання кнопки Створити нову подію з'явиться вікно, в якому необхідно ввести дату події та її опис. Після створення події подія з’явиться в головному вікні після вибору відповідного місяця. Натиснувши на дату, відкриється вікно, де ви можете редагувати або видалити цю подію. Якщо ви створюєте кілька подій, клацніть певний день, щоб відкрити перегляд подій для цього дня. Якщо час події настане, буде отримано повідомлення про неї та її опис. Для синхронізації користувачу необхідно натиснути кнопку Варіанти Синхронізації на головному вікні, яка відкриє панель з можливістю вибору з якого сервісу завантажувати або до якого відправляти події.

# 4 ЕКОНОМІЧНА ЧАСТИНА

## 4.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового програмного засобу організації та оптимізації часу і виробничого процесу з інтеграцією сервісів від Google та Microsoft. Особливістю програми є можливість інтеграції аналогів для об'єднання даних. Новизна полягає в об'єднанні декількох сучасних використовуваних систем у одному зручному додатку.

Аналогом може бути Microsoft Outlook Calendar 7$/міс., т.б. 3360 грн/рік або Teamup 8-80$/міс., т.б. 3840-38400 грн/рік.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п'ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1.

Таблиця 4.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

| Бали (за 5-ти бальною шкалою) | | | | | |
|---|---|---|---|---|---|
| Крите-рій | 0 | 1 | 2 | 3 | 4 |
| Технічна здійсненність концепції | | | | | |
| 1 | Достовірність концепції не підтверджена | Концепція підтверджена експертними висновками | Концепція підтверджена розрахунками | Концепція перевірена на практиці | Перевірено роботоздат-ність продукту в реальних умовах |
| Ринкові переваги | | | | | |
| 2 | Багато аналогів на малому ринку | Ринкові п Мало аналогів на малому ринку | Кілька аналогів на великому ринку | Один аналог на великому ринку | Продукт не має аналогів на великому ринку |

Продовження табл. 4.1

| | | | | | |
|---|---|---|---|---|---|
| Ринкові переваги | | | | | |
| 3 | Ціна продукту значно вища за ціни аналогів | Ціна продукту дещо вища за ціни аналогів | Ціна продукту приблизно дорівнює цінам аналогів | Ціна продукту дещо нижче за ціни аналогів | Ціна продукту значно нижче за ціни аналогів |
| 4 | Технічні та споживчі властивості продукту значно гірші, ніж в аналогів | Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів | Технічні та споживчі властивості продукту на рівні аналогів | Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів | Технічні та споживчі властивості продукту значно кращі, ніж в аналогів |
| 5 | Експлуатаційні витрати значно вищі, ніж в аналогів | Експлуатаційні витрати дещо вищі, ніж в аналогів | Експлуатаційні витрати на рівні експлуатаційних витрат аналогів | Експлуатаційні витрати трохи нижчі, ніж в аналогів | Експлуатаційні витрати значно нижчі, ніж в аналогів |
| Ринкові перспективи | | | | | |
| 6 | Ринок малий і не має позитивної динаміки | Ринок малий, але має позитивну динаміку | Середній ринок з позитивною . | Великий стабільний ринок | Великий ринок з позитивною динамікою |
| 7 | Активна конкуренція великих компаній на ринку | Активна конкуренція | Помірна конкуренція | Незначна конкуренція | Конкурентів немає |
| Практик на здійсненність | | | | | |
| 8 | Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї | Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців | Необхідне незначне навчання фахівців та збільшення їх штату | Необхідне незначне навчання фахівців | Є фахівці з питань як з технічної, так із комерційної реалізації ідеї |
| 9 | Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні | Потрібні незначні фінансові ресурси. Джерела фінансування відсутні | Потрібні значні фінансові ресурси. Джерела фінансування є | Потрібні незначні фінансові ресурси. Джерела фінансування є | Не потребує додаткового фінансування |
| 10 | Необхідна розробка нових матеріалів | Потрібні матеріали, що використовуються у військово-промисловому комплексі | Потрібні дорогі матеріали | Потрібні досяжні та дешеві матеріали | Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві |

Продовження табл. 4.1

| | | | | | |
|---|---|---|---|---|---|
| 11 | Термін реалізації ідеї більший за 10 років | Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років | Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років | Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років | Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років |
| 12 | Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту | Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу | Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу | Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту | Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту |

Усі дані по кожному параметру занесено в таблиці 4.2

Таблиця 4.2 — Результати оцінювання комерційного потенціалу розробки

| Критерії оцінювання | ПІБ експертів | | |
|---|---|---|---|
| | Експерт 1 | Експерт 2 | Експерт 3 |
| | Бали | | |
| Технічна здійсненність концепції | 3 | 3 | 4 |
| Наявність аналогів на ринку | 4 | 4 | 4 |
| Цінова політика | 4 | 4 | 4 |
| Технічні та споживчі властивості виробу | 4 | 3 | 4 |
| Експлуатаційні витрати | 3 | 4 | 3 |
| Ринок збуту | 4 | 3 | 4 |
| Конкурентоспроможність | 3 | 4 | 3 |
| Фахівці з технічної і комерційної реалізації | 4 | 3 | 4 |
| Фінансування | 4 | 4 | 3 |
| Матеріально-технічна база | 3 | 3 | 3 |
| Термін реалізації ідеї | 4 | 4 | 4 |
| Супровідна документація | 4 | 3 | 4 |
| Сума | 44 | 42 | 44 |
| Середньоарифметична сума балів | (44+42+44) / 3 = 43,33 | | |

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

| Середньоарифметична сума балів СБ , розрахована на основі висновків експертів | Рівень комерційного потенціалу розробки |
|---|---|
| 0-10 | Низький |
| 11-20 | Нижче середнього |
| 21-30 | Середній |
| 31-40 | Вище середнього |
| 41-48 | Високий |

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок того, що програмний продукт є новим програмним засобом організації та оптимізації часу і виробничого процесу з інтеграцією сервісів від Google та Microsoft. Особливістю програми є можливість інтеграції аналогів для об'єднання даних. Привабливість для споживачів полягає в об'єднанні декількох сучасних використовуваних систем у одному зручному додатку.

4.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

4.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t, \qquad (4.1)$$

де М — місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ — число робочих днів в місяці, 22 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.4.

Таблиця 4.4 — Основна заробітна плата розробників

| Найменування посади | Місячний посадовий оклад, грн. | Оплата за робочий день, грн. | Число днів роботи | Витрати на заробітну плату, грн. |
|---|---|---|---|---|
| Керівник проекту | 31000 | 1409,09 | 40 | 56363,636 |
| Програміст | 28000 | 1272,73 | 40 | 50909,091 |
| Всього | | | | 107272,73 |

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

4.2.2 Додаткова заробітна плата розробників, які приймали участь в розробці обладнання.

Додаткова заробітна плата прийнято розраховувати як 12,3% від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 12{,}3\ \% / 100\ \% \qquad (4.2)$$

$$З_д = (107272{,}73 \cdot 12{,}3\ \% / 100\ \%) = 13194{,}55 \ \text{(грн.)}$$

4.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (4.3)$$

$$Н_з = (107272{,}73 + 13194{,}55) \cdot 22\ \% / 100\ \% = 26502{,}80 \ \text{(грн.)}$$

4.2.4. Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

4.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (4.4)$$

де Ц — балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років;

$t_{вик}$ — термін використання під час розробки, місяців.

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 20000 грн., термін його корисного використання згідно податкового законодавства — 2 роки, а термін його фактичного використання — 1,82 міс.

$$A_{обл} = \frac{20000}{2} \times \frac{1{,}82}{12} = 1515{,}152 \ грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення.

Але, так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн. (Visual Studio Professional 1800 грн/міс, використовувався 1,82 місяці), то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $B_{нем.ак.}$ = 3273 грн.

Розрахунки заносимо до таблиці 4.5.

Таблиця 4.5 — Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

| Найменування обладнання | Балансова вартість, грн. | Строк корисного використання, років | Термін використання обладнання, місяців | Амортиза-ційні відрахуван-ня, грн. |
|---|---|---|---|---|
| Комп'ютер та комп'ютерна периферія (Lenovo Ideapad330) | 20000 | 2 | 1,82 | 1515,152 |
| Офісне обладнення (меблі) | 20000 | 4 | 1,82 | 757,576 |
| Приміщення | 600000 | 20 | 1,82 | 4545,455 |
| Всього | | | | 6818,18 |

4.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (4.5)$$

де В — вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П — встановлена потужність обладнання, кВт. П = 0,3 кВт;

Ф — фактична кількість годин роботи обладнання, годин;

$К_п$ — коефіцієнт використання потужності, $К_п$ = 0,9.

$$В_е = 0,9 \cdot 0,3 \cdot 8 \cdot 40 \cdot 6,2 = 535,68 \text{ (грн.)}$$

4.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на

собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.6)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

$$І_в = 107272,73 * 51\% / 100\% = 54709,09 \text{ (грн.)}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.7)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$Н_{нзв} = 107272,73 * 145\ \% / 100\ \% = \ 155545 \text{ (грн.)}$$

4.2.8 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$В_{заг} = 107272,73+13194,55+26502,80+6818,18+3273+535,68+54709,09+$$
$$+15555 = 367851,21 \text{ грн.}$$

4.2.9 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \quad (4.8)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$$ЗВ = 367851{,}21 / 0{,}5 = 735702 \text{ грн.}$$

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

— вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

— зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

— кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб'єктів (зацікавлених осіб) цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов'язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

4.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.10)$$

де $\pm \Delta Ц_o$ — зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

$N$ — кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_о$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_о = Ц_б \pm \Delta Ц_о$;

$Цб$ — вартість програмного продукту у році до впровадження результатів розробки;

$\Delta N$ — збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

$\lambda$ — коефіцієнт, який враховує сплату податку на додану вартість і ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$;

$p$ — коефіцієнт, який враховує рентабельність продукту;

$\vartheta$ — ставка податку на прибуток, у 2022 році $\vartheta$ =18%.

Припустимо, що при прогнозованій ціні 240 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *20* грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року — на 80000 шт., протягом другого року — на 125000 шт., протягом третього року на 150000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П_1 = (0*20 + (240 + 20)*80000)* 0{,}8333* 0{,}27) * (1 - 0{,}18) = 3542399{,}858 \text{ грн.}$$

$$\Delta П_2 = (0*20 + (240 + 20)*(80000+125000)* 0{,}8333* 0{,}27) * (1 - 0{,}18) = 9833849{,}607 \text{ грн.}$$

$$\Delta П_3 = (0*20 + (240 + 20)*(80000+125000+150000)* 0{,}8333* 0{,}27) * (1 - 0{,}18) = 17029349{,}319 \text{ грн.}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 30405598,78 грн.

4.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т — період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

$\tau$ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t — період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП= (3542399,858/(1+0,1)[1])+(9833849,607/(1+0,1)[2])+(17029349,319/(1+0,1)[3])

= 3220363,51 + 8127148,435 + 12794402,19 = 24141914,14 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (4.12)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію і це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

*ЗВ* — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 735702 = 1471404,83 \text{ грн.}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \tag{4.13}$$

$$E_{абс} = 24141914{,}14 - 1471404{,}83 = 22670509{,}31 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.14}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 22670509{,}31/1471404{,}83} - 1 = 1{,}541$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{4.15}$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні $d$ = (0,09…0,14);

f—показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0,14 + 0,05 = 0,19.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \qquad (4.16)$$

$$T_{ок} = 1 / 1,541 = 0,65 \text{ р.}$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,65 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 735702 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,65 роки.

# ВИСНОВКИ

Під час виконання магістерської роботи було здійснено перегляд системи організації та оптимізації часу, а також виробничого процесу та засоби інтеграції аналогів від Google та Microsoft. Вибір фреймворків був обґрунтований тим, що .NET 5.0 має достатньо розвинені можливості в області створення віконних додатків. Оскільки .NET працює на мовах C, вибір мови програмування зрештою був C#. Використання власного календаря дозволяє організувати та оптимізувати, з можливістю подальшого вдосконалення програми, а з можливістю синхронізувати події робить його більш зручним для користування.

Розроблено структуру програми, розроблено алгоритми створення, обробки та збереження подій. Проведено варіативний аналіз та аргументацію вибору програмного забезпечення під час створення платформи, а саме: мови програмування C# та середовища розробки Microsoft Visual Studio. Виконано створення та підключення методів інтеграції API використовуючи засоби Google. Описано програмну реалізацію системи. Було створено віконну програму, яка дозволяє користувачеві організовувати та оптимізувати час і виробничий процес шляхом створення та редагування подій, їх часу та опису. Розроблено методи синхронізації подій між додатком, Google Calendar та Microsoft Outlook. Розглянуто основні методи тестування, проведено тестування методом «чорної скриньки», розроблено список тестових випадків для тестування, розроблено керівництво для користувача. Проведено аналіз економочної частини роботи.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Pro C# 7: With .NET and .NET Core 8-th edition / Andrew Troelse, Philip Japikse. —2017.

2. Об'єктно-орієнтоване програмування [Електронний ресурс] Режим доступу: http://ruslan.rv.ua/python-essential/oop/oop_basis/

3. Об'єктно-орієнтований підхід до програмування [Електронний ресурс] Режим доступу: http://www.znannya.org/?view=csharp-oop

4. Офіційна документація .NET [Електронний ресурс] Режим доступу: https://docs.microsoft.com/en-en/dotnet/

5. Мова програмування C# і платформа .NET Core [Електронний ресурс] Режим доступу: https://metanit.com/sharp/tutorial/1.1.php

6. What's new in .NET 5 [Електронний ресурс] Режим доступу: https://docs.microsoft.com/en-us/dotnet/core/dotnet-five

7. .NET Core/5+ vs. .NET Framework for server apps [Електронний ресурс] Режим доступу: https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server

8. Представляємо .NET 5.0 [Електронний ресурс] Режим доступу: https://temofeev.com/info/articles/predstavlyaemo-net-5/

9. An Overview of Google Cloud Platform Services / Ekaba Bisong —2019.

10. Getting Started with Google Cloud Platform / S. P. T. Krishnan & Jose L. Ugia Gonzalez [Електронний ресурс] Режим доступу: https://link.springer.com/chapter/10.1007/978-1-4842-1004-8_2

11. Google Cloud Platform in Action / John J. Geewax —2017.

12. An Evaluation of API Calls Hooking Performance [Електронний ресурс] Режим доступу: https://ieeexplore.ieee.org/abstract/document/4724665

13. API [Електронний ресурс] Режим доступу: Overview https://usa.ingrammicro.com/cms/media/Documents/customer_service/API-Fact-Sheet.pdf

14. Beginning Google Maps API 3/ Gabriel Svennerberg —2010.

15. Google Calendar API [Електронний ресурс] Режим доступу: https://developers.google.com/calendar/api

16. Outlook Primary Interop Assembly reference [Електронний ресурс] Режим доступу: https://learn.microsoft.com/en-us/office/client-developer/outlook/pia/welcome-to-the-outlook-primary-interop-assembly-reference

17. Using Google Calendar to Manage Library Website Hours [Електронний ресурс] Режим доступу: https://journal.code4lib.org/articles/46

18. Mastering OAuth 2.0 / Charles Bihis —2015.

19. C# 9 and .NET 5 – Modern Cross-Platform Development / Mark J. Price — 2020.

20. Pro C# 10 with .NET 6 / Andrew Troelsen, Phil Japikse —2022.

# ДОДАТОК Ж

Протокол перевірки кваліфікаційної роботи

Назва роботи: Програмний засіб організації та оптимізації часу і виробничого процесу з інтеграцією сервісів від Google та Microsoft

Тип роботи: магістерська кваліфікаційна робота

(БДР, МКР)

Підрозділ кафедра обчислювальної техніки

(кафедра, факультет)

**Показники звіту подібності Unicheck**

Оригінальність 80% Схожість 20%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.

(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Цуренко О.І.

(підпис) (прізвище, ініціали)

Керівник роботи _____ Черняк О.І.

(підпис) (прізвище, ініціали)