Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему «Розробка додатку для систематизації запланованих завдань та подій»

Виконала: студентка групи К19-2

Спеціальність 122 «Комп'ютерні науки»

Лісова І.О.
(прізвище та ініціали)

Керівник д.т.н., доцент Яковенко В.О.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент Університет миної справи та фінансів
(місце роботи)

доцент кафедри кібербезпеки та інформаційних технологій
(посада)

к.т.н., доц. Фролов С.В.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2023

АНОТАЦІЯ

Лісова І.О. Розробка додатку для систематизації запланованих завдань та подій.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 122 «Комп'ютерні науки». – Університет митної справи та фінансів, Дніпро, 2023.

Метою кваліфікаційної роботи було створення програмного продукту для внесення та перегляду інформації про заплановані завдання та події. Додаток надає можливість користувачу створювати завдання та події, переглядати їх в календарі, слідкувати за статистикою виконання завдань та відвідування подій. За необхідності в програмі можна відредагувати вміст, або взагалі видалити заплановане завдання чи подію. Програмний продукт було створено з використанням кросплатформенного фреймворку .NET Multi-platform App UI. Додаток було написано мовою об'єктно-орієнтованого програмування C#, інтерфейс був написаний за допомогою декларативної мови розмітки XAML.

Програмний продукт можна використовувати, як для користувачів мобільних пристроїв із операційною системою Android, так і для користувачів комп'ютерів чи ноутбуків із операційною системою Windows. Але для використання додатку є обмеження в версії операційних систем, як для Android, так і для Windows.

Ключові слова: кросплатформенний фреймворк, додаток, планування, локальна база даних, користувач, користувацький інтерфейс, завдання.

ANNOTATION

Lisova I.O. Development of an application for systematization of scheduled tasks and events.

Qualification work for obtaining a bachelor's degree in specialty 122 «Computer Science». – University of Customs and Finance, Dnipro, 2023.

The goal of the qualification work was to create a software product for entering and viewing information about planned tasks and events. The application allows the user to create tasks and events, view them in the calendar, monitor the statistics of task completion and event attendance. If necessary, you can edit the content, or even delete the planned task or event in the program. The software product was created using the .NET Multi-platform App UI cross-platform framework. The application was written in the object-oriented programming language C#, the interface was written using the declarative markup language XAML.

The software product can be used both for users of mobile devices with the operating system and for users of computers or laptops with the Windows operating system. But there are restrictions of using the application depending on the version of the operating systems, both for Android and for Windows.

Keywords: cross-platform framework, application, planning, local database, user, user interface, tasks.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

WinUI (Windows UI Library) – платформа взаємодії з користувачем для робочого столу Windows та програм універсальної платформи Windows

API (Application Programming Interface) – інтерфейс прикладного програмування

.NET MAUI (.NET Multi-platform App UI) – кросплатформенний фреймворк для створення мобільних та комп’ютерних додатків з використанням C# та XAML

C# – об’єктно-орієнтована мова програмування

XAML – декларативна мова розмітки

ОС – операційна система

ІТ – інформаційні технології

ВСТУП

В умовах швидкого розвитку ІТ (інформаційних технологій) зростає необхідність їх використання у різних галузях професійної діяльності людей та в повсякденному житті. За останні десятиліття значно збільшилася кількість завдань, які людина виконує впродовж дня. Так само збільшилася і потреба у мобільних та комп'ютерних додатках, які допомагатимуть вирішувати необхідні завдання.

Необхідною умовою для успішності виконання всіх необхідних завдань та відвідування подій (робочі наради, свята, зустрічі з друзями) є правильне планування. Саме планування дозволяє використовувати час раціонально та ефективно. Час – один із головних ресурсів в будь-якій сфері діяльності людини, тому його заощадження є однією із ознак успішності та ефективності в роботі. Комп'ютерні та мобільні додатки здатні допомогти користувачам із завданням планування.

Одним із основ планування є запис завдання чи події на конкретну дату. За необхідності людина записує конкретний час та можливо якийсь опис, або додаткову інформацію для запланованого пункту. Раніше люди робили це в зошитах, блокнотах чи навіть просто на аркуші паперу. Але це не зручно, адже аркуш легко загубити, а блокнот чи зошит не зручно брати з собою (на роботу, у відпустку). Для цього в нагоді стають програмні рішення. Якщо людина працює з комп'ютером чи ноутбуком і постає необхідність запису робочих завдань, то комп'ютерна програма буде зручним рішенням. Для більшості користувачів скоріше за все буде зручніше використання мобільного додатку для планування, адже мобільний телефон є неодмінною частиною життя людини.

Метою виконання кваліфікаційної роботи є оптимізація процесу планування. Необхідно дослідити процес планування та його важливість в різних сферах, особливо в галузі ІТ. Також треба провести аналіз наявних рішень для вибору функцій, які треба впровадити до власного рішення.

Тема планування, або як ще його називають тайм-менеджменту є актуальною для будь-якої галузі роботи та життя. Для планування уже є створені програми. Однак всі користувачі використовують не один і той самий додаток для вирішення задачі планування. Перш за все це пов’язано з тим, для якої операційної системи створений додаток, адже різні застосунки передбачають використання певної операційної системи. Другою причиною є версії самих операційних систем, адже кожен додаток містить обмеження щодо мінімальної версії певної операційної системи. І третьою причиною є вподобання конкретної людини через різну функціональність і інтерфейс програм. Деякі застосунки перенасичені функціями, деякі дають можливість виконувати лише основні, а деякі мають досить мало можливостей. Те саме можна сказати і про інтерфейс, кожен додаток має свій вигляд та навігацію. Але через різність людей, їхні різні потреби та вподобання, вони встановлюють різні програми для виконання однієї і тієї ж мети.

Основним завданням кваліфікаційної роботи є створення застосунку для запису, перегляду запланованих завдань та подій. Користувач матиме можливість створити завдання на поточний, або будь-який інший день. Переглянути завдання можна на сторінці для поточного дня, або в календарі. За потреби користувач може відредагувати завдання чи події, або взагалі видалити їх. Також можна переглянути статистику виконаних завдань та відвіданих подій, як за поточний день, так і за весь час.

Об’єктом дослідження є проблема систематизації планів. Проблема полягає в записі планів та показу співвідношення кількості виконаних до невиконаних.

Основними методами дослідження при виконанні роботи були: аналіз наявної інформації та програмних продуктів з теми планування, проведення аналізу наявних технічних рішень для вирішення поставленого завдання.

Для користувача немає можливості створити завдання чи подію без назви та дати. Якщо користувач не введе дату, то дата буде обрана автоматично, як дата поточного дня. Але якщо не буде введено назву та буде

спроба зберегти завдання чи подію, то буде виведено відповідне повідомлення про неможливість збереження такого завдання чи події.

Користувач має змогу зайти в завдання та відмітити його виконання, якщо завдання було виконано. Аналогічно можна відмітити відвідану подію.

Додаток буде працювати для операційної системи Windows – для Windows 11 та Windows 10 версії 1809 та вище з використанням WinUI 3 (Windows UI Library, платформа взаємодії з користувачем для робочого столу Windows та програм універсальної платформи Windows). Також застосунок працюватиме для операційної системи Android – Android 5.0 (API (Application Programming Interface, інтерфейс прикладного програмування) 21) та вище.

Для користувачів з операційною системою Android є можливість ввімкнути отримування сповіщень, які нагадуватимуть не забувати про завдання та події. Якщо сповіщення ввімкнено, то вони будуть надходити кожного дня о восьмій годині ранку.

Для розробки програмного забезпечення було використано C# (об’єктно-орієнтована мова програмування), XAML (декларативна мова розмітки), .NET MAUI (.NET Multi-platform App UI, кросплатформенний фреймворк для створення мобільних та комп’ютерних додатків з використанням C# та XAML).

.NET MAUI кросплатформенний фреймворк дав змогу створити додаток, який можна використати, як для Windows, так і для Android в одному проєкті. Деякі деталі були прописані окремо для тієї чи іншої платформи.

За допомогою XAML Hot Reload при розробці додатку була можливість редагувати інтерфейс користувача не зупиняючи відлагоджування проєкту і відразу бачити зміни.

Застосунок дає можливість користувачу користуватися наявною темою, або увімкнути темну тему інтерфейсу.

Меню та навігація є досить зрозумілими та зручними, що дозволить користувачеві без складностей користуватися додатком.

Зовнішній вигляд для Windows трішки відрізняється від інтерфейсу користувача для Android. Функціональність для обох операційних систем є однаковою. Однак сповіщення є доступними лише для Android. Це так, через те що NuGet пакет, який використовується для сповіщень дає можливість створювати сповіщення лише для мобільних пристроїв.

Роботу програми було перевірено на пристрої з операційною системою Windows 10 Pro, мобільному пристрої з операційною системою Android 29 (API 31) та на декількох різних емуляторах Android.

Програма була розроблена в інтегрованому середовищі розробки Visual Studio 2022. Проєкт створено на .NET MAUI 7. Для зберігання даних використовується локальна база даних SQLite.

При обґрунтуванні вибору теми кваліфікаційної роботи було проведено аналіз сучасних досліджень відповідної галузі, з визначенням сутності й актуальності поставленого завдання. В ході аналізу виявлено, що всі програмні продукти, окрім задоволення потреб користувача, мають кінцеву мету – отримання прибутку. Кожна галузь задля свого розвитку потребує фінансових витрат для вдосконалення.

Розроблений продукт представлений, як економічно-вигідна модель з можливістю подальших змін та покращень. В процесі дослідження, за підсумками та критичним аналізом, було з’ясовано, що в умовах сучасного ринку для розробки кросплатформенного додатку треба великі витрати, тому необхідно розробити додаток із зменшенням цих витрат. Економічну вигідність даної моделі характеризує така основна риса – можливість змін та удосконалення за рахунок однієї кодової бази, що ніяк не зменшує перевагу створеного застосунку.

Структура роботи складається з вступу, трьох розділів, висновку, списку використаних джерел та додатків. Кваліфікаційна робота має п’ятдесят сторінок, робота містить вісімнадцять рисунків, три таблиці, список використаних джерел складається із двадцять одного найменування.

РОЗДІЛ 1.
АНАЛІЗ СУТНОСТІ ЗАДАЧІ ПЛАНУВАННЯ В РІЗНИХ СФЕРАХ ДІЯЛЬНОСТІ, ЗОКРЕМА У ГАЛУЗІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Сутність задачі планування в галузі інформаційних технологій

Планування дає напрямок для досягнення певних цілей в житті людини. Життя людини може прийняти зовсім неочікуваний напрямок, якщо немає взагалі ніякого плану. Якщо спланувати, що ти маєш робити та як цього досягти, то можна успішно побудувати власне майбутнє. Планування допомагає контролювати процеси досягнення цілей в різних сферах життя.

Причини, через які планування є настільки вважливим:

– зміни в житті (процес планування допоможе окреслити майбутнє та зробити певні зміни в житті позитивними та значущими);
– самовизначення (планування завдань для покращення робочих навичок, або розвитку власного хобі);
– будівництво майбутнього (процес планування майбутнього буде приводити до конкретних дій для його втілення);
– вивчення можливостей (планування допомагає іноді подивитися на речі інакше, що призводить до виникнення нових ідей та можливостей);
– вирішення проблем [1].

Систематизація запланованих планів та подій є невід’ємною частиною різних сфер життя та діяльності людини. Зокрема для галузі інформаційних технологій планування є невід’ємною частиною виконання проєкту.

Планування при створенні ІТ проєкту є однією з гарантій його успішності. Команда, яка працює над створенням проєкту докладає чимало зусиль для створення правильного та ефективного плану його виконання. Створення плану виконання проєкту допомагає переконатися, що проєкт відповідає кінцевим термінам і загальним цілям.

Планування є другою фазою ІТ проєкту після його ініціації та перед його виконанням [2].

Переважно участь у процесі планування проєкту мають брати:

– керівник проєкту;
– команда управління проєктом;
– команда проєкту;
– замовник;
– спонсор проєкту;
– вищий менеджмент;
– інші зацікавлені сторони [3].

Планування проєкту це не лише один із етапів управління проєктом. Це один із самих ключових, відповідальних, важливих етапів усього процесу.

Причини важливості планування при створення ІТ проєктів:

– підвищення рівня успіху проєкту та його продуктивності;
– заощадження коштів;
– покращення комунікації команди;
– переконання у найкращому використанні ресурсів;
– полегшене відслідковування цілей проєкту;
– допомагає підтримувати взаємодію між усіма співавторами [4].

Планування проєкту допомагає всім працівникам, які працюють над проєктом розуміти, що треба зробити, та як цього досягти. При створенні плану виконання проєкту визначають результати та вимоги до проєкту, створюється його графік. Це передбачає створення набору задач для того, щоб допомогти команді провести етапи впровадження та закриття проєкту. Задачі, які створені на цьому етапі дадуть змогу керувати змінами, якістю, часом, вартістю, ризиками та проблемами, які можуть виникати в процесі розробки проєкту. Це також допоможе контролювати команду, щоб гарантувати виконання проєкту в межах графіку та бюджету [5].

Основні етапи при плануванні ІТ проєкту:

– визначення зацікавлених сторін та зустріч з ними;

– встановлення цілей та розміщення їх за пріоритетами;
– визначення результатів, яких необхідно досягти;
– створення графіку проєкту;
– визначення проблем та проведення оцінки ризиків;
– представлення плану проєкту зацікавленим сторонам [6].

Планування є важливою частиною не тільки при виконанні ІТ проєкту, а й при загальній організації робочого процесу в ІТ компанії.

В ІТ компаніях використовують планування організації роботи, адже це допоможе визначити напрямок в якому компанія має рухатися для досягнення більших успіхів. Це необхідно для того, щоб бути конкурентоспроможними.

Процес планування для цілої компанії, організації називають організаційним плануванням. Це планування є процесом, який допомагає встановити цілі, яких треба досягти для реалізації всього потенціалу організації та створення більш детальних завдань для досягнення поставленої мети.

Фази організаційного плану (рис. 1.1):
– стратегічна;
– тактична;
– оперативна;
– на випадок непередбачуваних ситуацій.

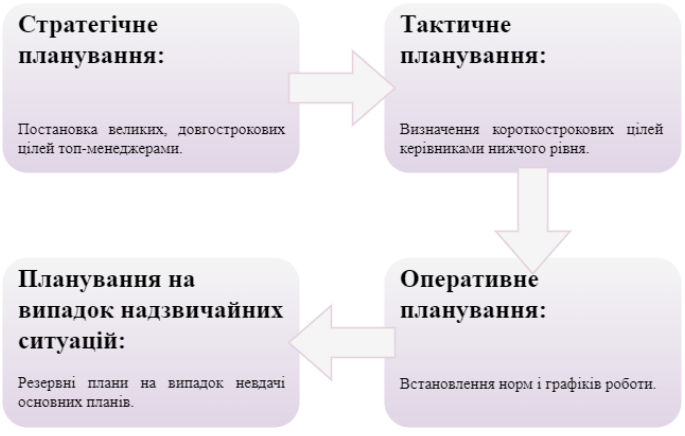

Рисунок 1.1 – Етапи організаційного планування

Процес організаційного планування включає в себе п’ять етапів. При ідеально проведеному процесі організаційного планування ці етапи будуть утворювати цикл.

Першим етапом буде розробка стратегічного плану. Цей етап передбачає спочатку визначення мети. Після чого треба зібрати дані про компанію, при цьому особливу увагу звернути на показники ефективності роботи компанії. Далі вважливим кроком є оцінка сильних та слабких моментів в компанії. Переглянути та можливо скорегувати, доповнити мету, яку було сформульовано раніше.

Другим етапом є переведення стратегічного плану у тактичні кроки. Для того, щоб виконати даний етап необхідно залучити менеджерів середньої ланки. На цьому етапі треба визначити для кожного відділу короткострокові цілі, які допоможуть досягти стратегічної мети. Також треба вигадати, як контролювати процес виконання та досягнення стратегічних та тактичних цілей. Звичайно, потрібно не забути про розроблення плану на випадок виникнення надзвичайних ситуацій.

Третій етап передбачає планування щоденних операцій. Цей етап має на меті встановлення цілей та завдань, яких має досягти конкретний працівник за певний, відведений для цього, період часу.

Четвертий етап передбачає вже впровадження планів в роботу компанії. Щоденні операції, які виконують працівники мають сприяти досягненню тактичних цілей, а отже і підтримці стратегічного плану.

П’ятий етап – контроль прогресу та корегування планів. На цьому етапі треба відслідковувати виконання задач через певні проміжки часу. Якщо результати не будуть задовільними, то постає необхідність в корегуванні планів [7].

Необхідно також розуміти, що стратегічний ІТ-план має містити певні аспекти:

– перелік необхідних застосунків та програмного забезпечення, які необхідні для реалізації стратегічних завдань;

- оновлення додатків, які вже існують для їх ефективної роботи, або перехід на інші додатки, через неефективність роботи наявних;
- визначення ресурсів необхідних для того чи іншого проєкту;
- ініціативи з розвитку персоналу, постійне оновлення та покращення знань працівників;
- план управління безпекою.

Будь-яка організація, яка планує впроваджувати технології має пов’язати стратегічний план із технологіями та бізнес-цілями. Для досягнення такого взаємозв’язку є вісім ключових кроків.

Першим кроком є встановлення конкретних цілей та визначення ролі технологій в компанії. Спочатку треба визначитися з тим, чи є та чи інша технологія необхідною лише для короткострокової цілі, чи є необхідністю в довгостроковій перспективі, або взагалі є невід’ємною частиною для постійної, успішної роботи компанії. Треба провести початковий порівняльний аналіз технологічних стратегій. Також потрібно провести оцінку витрат, які необхідні для впровадження технологій.

Другим кроком є формулювання бачення, яку роль буде виконувати певна технологія для досягнення поставлених цілей згідно з стратегічним планом. Іноді заміна поточних програм на більш ефективні дає можливість стандартизувати деякі процеси. Окрім цього необхідно організувати процес навчання працівників, щоб вони завжди були готовими до змін і могли швидше адаптуватися.

Третій крок полягає у визначенні ІТ-процесів та ІТ-послуг. Треба вирішити для якого процесу потрібне впровадження технологій, щоб забезпечити обґрунтоване, ефективне впровадження інвестицій.

Четвертий крок – оцінка недоліків у впровадженні технологій. Цей крок полягає в тому щоб оцінити, які неточності існують між планом впровадження технологій та тим, як це вийде фактично реалізувати.

П’ятим кроком є формулювання варіантів виправлення недоліків виявлених у четвертому кроці.

Шостим кроком є обрання найкращих варіантів для реалізації впровадження технологій. Треба скласти план із урахуванням періоду часу та інвестиційного бюджету.

Сьомим кроком є встановлення принципів для впровадження програмних ініціатив. Треба визначити чи програмне забезпечення буде розроблятися всередині компанії чи купуватися ззовні (в інших компаніях, установах). Є певні фактори, які впливають на прийняття цього рішення:

– кадрові можливості;
– потреби налаштування;
– витрати на розробку та закупівлі;
– терміни реалізації;
– потреби в підтримці;
– витрати на технічне обслуговування.

Також потрібно визначити чи буде система розміщена, як локальне середовище даних, чи як хмарне середовище. На прийняття даного рішення також впливають деякі фактори:

– стабільність внутрішнього середовища;
– стабільність Інтернет-ліній передачі даних.

Восьмий крок – вимірювання користі. Керівна ланка компанії, організації має обговорити, як виміряти вигоду від інвестування у технології [8].

В ІТ компаніях дуже вважливою є робота в команді. Адже над кожним проєктом працює команда людей. Кількість людей в команді може бути різною, в залежності від складності проєкту. І в цьому випадку є актуальним планування для роботи в команді.

Командне планування являє собою процес розробки оперативних та стратегічних планів саме для цієї команди. Розроблений план дає команді напрямок в якому рухатися для досягнення кінцевої мети. У цьому плані вказується і те, як команда може розподілити між собою цілі та завдання для якомога ефективнішого досягнення поставленої мети всієї команди.

Можна виділити сім основних кроків для створення командного плану:

Першим кроком є створення бачення, якою має бути команда. Для цього треба прописати всіх членів команди та сформулювати основну мету, якої вони мають досягти.

Другий крок полягає в окресленні конкретних завдань команди. Тобто в формулюванні завдань для кожного з команди, які допоможуть виконати основну мету. До виконання цього кроку краще залучити команду, яка буде виконувати ці завдання. Адже команда краще знає скільки часу знадобиться для виконання певного завдання та кому з команди можна доручити конкретне завдання.

Третій крок – встановлення часових обмежень. Цей крок передбачає встановлення строку виконання поставленої мети. У випадку, якщо це ІТ компанія, то це час відведений на виконання проєкту, який доручили команді.

Четвертим кроком є розставлення завдань за пріоритетами. На цьому кроці визначається послідовність, в якій будуть виконуватися завдання, адже одна людина з команди відповідає відразу за декілька завдань, які треба розставити в хронології порядку виконання.

На п’ятому кроці необхідно визначити, як буде визначатися успішність виконання завдань. Вимірювання прогресу досягання мети допоможе контролювати та за потреби корегувати розроблений план.

Шостим кроком є уточнення очікувань керівника проєкту над яким працює команда, чи керівника команди. Це допоможе порівняти початкові очікування та фактичний результат на певному етапі, після проходження певного періоду часу.

Сьомий крок – перегляд прогресу команди у досягненні кінцевої мети. Періодично можна спостерігати за роботою команди та занотовувати ті дії, які можливо покращують роботу та ті, які навпаки її гальмують. За потреби можна запланувати зустрічі всієї команди, керівника проєкту та замовника. Такі зустрічі можна робити один раз на тиждень, або два рази в тиждень. Ці зустрічі дадуть змогу підтримувати зв’язок команди з керівником та замовником.

Також це сприятиме уникненню встановлення нереалістичних для реалізації цілей [9].

Процес планування це завжди циклічний процес (рис. 1.2). Адже постійно треба відслідковувати процес виконання плану та робити певні корегування.

Рисунок 1.2 – Процес планування

Побудова досяжних цілей полягає в формулюванні такої цілі, для досягнення якої є ресурси та можливості.

Створення основи плану передбачає визначення кінцевої основної цілі. В процесі ця ціль може бути інакше трішки скорегована, через ті чи інші зміни, але основа для планування закладається саме на цьому етапі.

Визначення часу для плану – часова тривалість, на яку буде розрахований створений план.

Визначення методів для досягнення цілі полягає в визначені альтернатив за якими можна працювати, щоб план був виконаний.

Оцінка та вибір напрямку дій – вибір найбільш вигідного, ефективного, результативного методу серед визначених на попередньому кроці. Якщо це

планування необхідне для організації, то участь у виборі беруть працівники, які будуть реалізовувати план з використанням цих методів.

Розробка похідних цілей (завдань) передбачає формулювання різних задач для досягнення головної мети. Кількість цих задач залежить від складності завдання та кількості людей, які будуть працювати. Це може бути одна людина, яка хоче досягти певної мети та планує її досягнення, так і компанія, організація.

Регулювання процесу є невід'ємною частиною планування, адже план може виявитися не ефективним, або виникнуть непередбачувані ситуації, які не дадуть змоги продовжувати виконувати раніше створений план. Цей пункт передбачає оцінку ефективності плану та корегування його основної мети чи похідних задач [10].

1.2 Опис завдання

Завданням кваліфікаційної роботи є дослідження планування в різних сферах діяльності людини. Ключовою сферою є дослідження планування в галузі інформаційних технологій. Це створення плану роботи при розробці ІТ проєкту. Також це загальна організація роботи в ІТ компанії (визначення плану для її успішної роботи). Ще одним направленням є дослідження при впровадженні ІТ в роботу якогось підприємства чи організації.

В ході виконання роботи має бути створено додаток, який допоможе користувачу із систематизацією завдань та подій. Для створення цього застосунку спочатку має бути проведена робота із дослідження додатків, які вже існують для вирішення задачі планування завдань та подій. Необхідно виділити основні переваги та недоліки того чи іншого рішення, яке вже існує, для того щоб обґрунтувати, яким має бути власне розроблене рішення. Для розробки власного програмного продукту треба визначитися із платформою для якої воно має працювати та мовою програмування. Також в роботі треба описати розроблений додаток та написати інструкцію користувачу.

1.3 Аналіз і характеристика додатку для систематизації завдань та подій

Результатом практичної частини кваліфікаційної роботи, а саме розробки програмного продукту для систематизації завдань та подій, має стати додаток, який буде працювати, як для операційної системи Windows, так і для операційної системи Android. Тому для розробки даного додатку буде доцільно використовувати можливості для кросплатформенної розробки.

Застосунок має дати можливість користувачу вводити завдання та події на конкретну дату та за потреби вказувати і час. Якщо постає необхідність додати якийсь опис, уточнення, будь-яку додаткову інформацію до завдання чи події, то має бути представлена така можливість для користувача програми. Ще однією необхідною можливістю має бути звичайно відмітка про виконання завдання чи відвідання події та можливість редагувати, видаляти окреме завдання чи подію.

Користувачу має бути зручно користуватися додатком у повсякденному вжитку, тому необхідний простий, зрозумілий, зручний у використанні інтерфейс.

Для користувачів мобільних пристроїв із операційною системою Android знадобляться сповіщення, адже в телефоні людини встановлена велика кількість додатків, тому важливо нагадування про можливі плани.

Забезпечення статистики є також необхідним. Статистики, як за поточний день так і за весь час. Ця статистика має бути представлена, як у вигляді числової кількості, так і у графічному вигляді (діаграми), де можна буде відразу побачити відсоткове співвідношення.

Перегляд завдань має бути можливим як за поточний день, так і в календарі за будь-який інший день.

Загалом додаток має бути функціонально зрозумілим та не перевантаженим великою кількістю функцій. Це допоможе користувачу легко орієнтуватися при використанні додатку для планування свого дня, тижня.

1.4 Висновки до першого розділу

На основі дослідження планування в сфері інформаційних технологій можна зазначити, що правильна побудова плану є великою частиною успіху. Правильно побудований організаційний план дасть змогу організувати роботу у цілій компанії, щоб її функціонування було якнайбільш ефективним та прибутковим. Командний план стане в нагоді, якщо є необхідність працювати в команді, наприклад над IT проєктом. І навіть для виконання самого IT проєкту треба складати план. Якщо якомусь підприємству чи організації потрібно впроваджувати нові технології, то необхідно також скласти план щодо їх впровадження.

Загалом задача планування складається з певних етапів, які утворюють цикл. Утворення циклу пояснюється тим, що останнім етапом є перевірка ефективної роботи плану через певний проміжок часу. Внаслідок цього при не повній ефективності чи зміні якихось деталей постає необхідність змінити деталі плану, а отже від останнього етапу перевірки знову пройти по попередніх етапах.

Планування допомагає досягти необхідного результату з максимальною користю за відведений проміжок часу. Це спрацює при правильно створеному плані та його успішному виконанні. Для планування можна використовувати додатки, які допомагають вносити та редагувати плани та переглядати статистику їх виконання.

В ході виконання роботи має бути розроблено додаток для систематизації завдань та подій.

РОЗДІЛ 2.
АНАЛІЗ НАЯВНИХ ПРОГРАМНИХ РІШЕНЬ ДЛЯ ВИРІШЕННЯ ПРОБЛЕМИ СИСТЕМАТИЗАЦІЇ ЗАВДАНЬ ТА ПОДІЙ

2.1 Microsoft To Do

Microsoft To Do є програмою для складання списку справ. Застосунок буде корисним і для роботи, і для відпочинку. Microsoft To Do можна встановити як на комп’ютер, так і на мобільний телефон.

Основною перевагою є кросплатформенність додатку. Це дозволяє отримувати доступ до одного синхронізованого переліку справ як з дому, так із роботи з використанням комп’ютера чи ноутбука, або в дорозі за допомогою телефону.

Для входу в додаток необхідно зайти за допомогою облікового запису Microsoft. Якщо користувач не має облікового запису, то при вході до додатку крім входу за наявним обліковим записом, буде запропоновано його створити.

За потреби можна зробити спільний доступ з іншими користувачами. Ця можливість є зручною, якщо користувачі використовують додаток для робочих завдань. В такому випадку колеги можуть прописувати робочі задачі і мати доступ до перегляду завдань інших (є корисною практикою при роботі команди над якимось проєктом).

Застосунок надає можливість розбивати великі завдання на менші. Є можливість для вставлення терміну виконання завдання.

Microsoft To Do – програма, яка синхронізується між смартфонами iPhone, пристроями з ОС (операційна система) Android і Windows, також є доступною в Інтернеті. Так само є можливість інтеграції із завданнями Outlook.

В додатку є різні теми, що дозволяє користувачам робити зміну в вигляді. Також є можливість додавати до завдань нотатки та вкладення. Вкладення мають бути до 25 МБ.

Треба також враховувати, що додаток для мобільних пристроїв доступний не у всіх країнах та не у всіх регіонах [11].

Терміни та нагадування можна встановити одноразові та повторювані.

На рис. 2.1 показано головний екран додатку Microsoft To Do для ОС Windows.

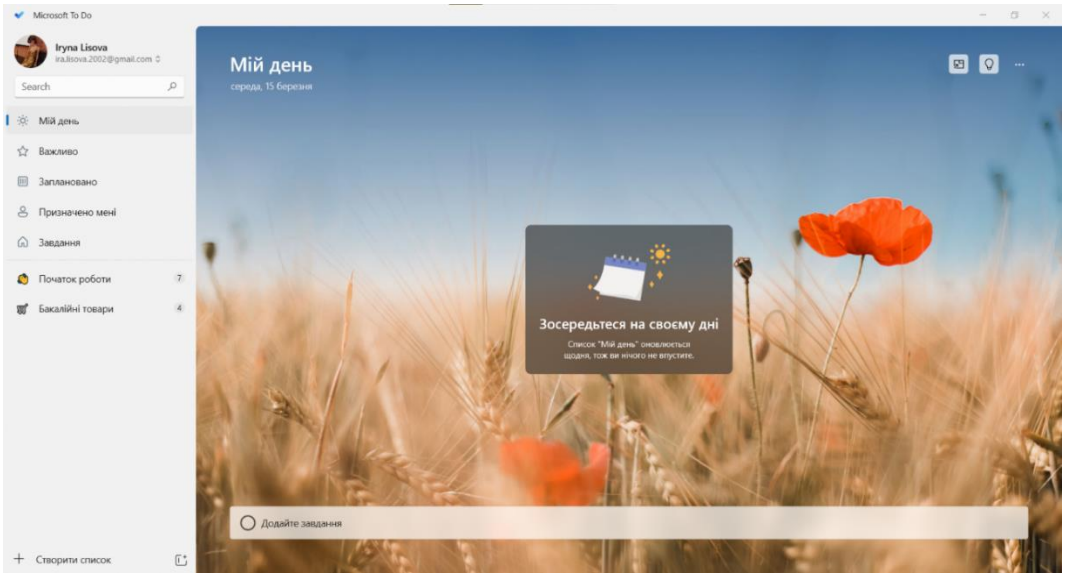

Рисунок 2.1 – Головна сторінка програми Microsoft To Do в ОС Windows

В додатку є меню, яке складається з таких розділів:

– мій день;
– важливо;
– заплановано;
– призначено мені;
– завдання.

На сторінці «Мій день» є можливість додати завдання для поточного дня, а також переглядати завдання, які заплановані для цього ж дня. Завдання можна позначити як завершене, або як важливе (або і завершене і важливе).

Сторінка «Важливо» містить список всіх завдань, які були відмічені, як важливі. Також можна відразу на цій сторінці додати завдання і воно буде помічене, як важливе.

«Заплановано» – сторінка, яка складається з списку завдань, які мають термін виконання чи нагадування. Тут, як і на двох попередніх сторінках можна додати завдання до запланованих. На відмінну від двох попередніх термін виконання має бути вказаний (за замовчуванням – поточний день).

Сторінки «Мій день», «Важливо», «Заплановано» мають можливість встановлення терміну, нагадування, повторюваність та видалення завдання в разі необхідності.

На сторінці «Призначено мені» відображаються завдання, які призначені іншим користувачем (якщо надано спільний доступ).

Всі завдання, які є в користувача, будуть відображатися на вкладці «Завдання». На цій сторінці також можна створити завдання, встановити термін, нагадування, повторюваність чи видалити.

Якщо натиснути на завдання, то буде відкрито меню із можливостями для розділення завдання з великого на менші кроки. Тут дублюються можливості встановлення нагадування, терміну та повторюваності. Також є можливість додати вкладення (тобто файл) до 25 МБ та додати якісь нотатки.

Автоматично на початку є створені два списки: «Початок роботи» та «Бакалійні товари». «Початок роботи» містить інструкцію по користуванню. Список «Бакалійні товари» містить перелік для походу в магазин (список продуктів). Якщо ці списки не потрібні, то їх можна видалити. Також можна створити власні списки з завданнями.

Користувач додатку може змінити тему будь-якої із вкладок.

В місці для пошуку можна знайти необхідне завдання. Пошук буде відбуватися відразу по всіх списках завдань.

В параметрах є можливість виконати ще деякі налаштування:

– вмикання чи вимикання додавання нових завдань на початок;
– вмикання чи вимикання переміщення завдань із зірочкою (помічені як важливі) на початок;
– вмикання чи вимикання відтворення звуку завершення (звук, який лунає коли завдання відмічається як виконане);

- вмикання чи вимикання запиту на підтвердження під час видалення завдання;
- вмикання чи вимикання автозапуску програми під час запуску Windows;
- обрати початок тижня (обрати будь-який день тижня, або за стандартними параметрами системи);
- зміна значка програми;
- можливість закріпити програму на панелі завдань для швидкого доступу;
- перелік сполучень клавіш, які допоможуть виконувати деякі команди в програмі і при цьому використовувати лише клавіатуру.

Це загальні параметри в додатку. Також є параметри пов’язані з темою, умовними списками, підключеними програмами, сповіщеннями, довідкою та відгуками, зв’язком та інформацією про програму.

На мобільному пристрої з ОС Android параметри програми майже не відрізняються. Зовнішній вигляд на різних ОС трішки відрізняється, це зумовлено особливостями різних операційних систем.

2.2 Focus To-Do

Focus To-Do – програма для керування часом і завданнями. В додатку можна складати плани для роботи та навчання, записувати список покупок, налаштовувати нагадування про дні народження. Застосунок також показує статистику по завданням, що дозволяє проаналізувати час витрачений на виконання.

Користуватися додатком можна як з комп’ютера, так і з мобільного телефону та планшету. Програма є доступною для мобільних телефонів та планшетів з ОС Android. Також є для iPhone, iPad та Apple Watch. Ще є доступною і для Mac, Windows, та як розширення до Chrome.

Програма побудована з використанням техніки Pomodoro. Ця техніка полягає в тому, що спочатку користувач обирає завдання, яке хоче виконати. Потім встановлює таймер на 25 хвилин та починає зосереджено працювати. Після завершального сигналу таймера користувачу треба зробити перерву на п’ять хвилин.

Є можливість щодня переглядати звіт по завданням, тобто прогрес у їх виконанні.

Однією з переваг додатку є можливість синхронізації даних між кількома пристроями. Спочатку треба увійти в обліковий запис, або зареєструватися. Можна користуватися застосунком і без входу та реєстрації, але тоді синхронізація не буде доступною.

Є можливість встановити термін виконання завдання та нагадування. Якщо є необхідність, то великі завдання можна розділити на менші завдання. Завдання, які треба буде виконувати не одноразово, можна поставити на повторення. Для внесення додаткової інформації про завдання додається примітка [12].

На рис. 2.2 показано головний екран додатку Focus To-Do в ОС Windows.

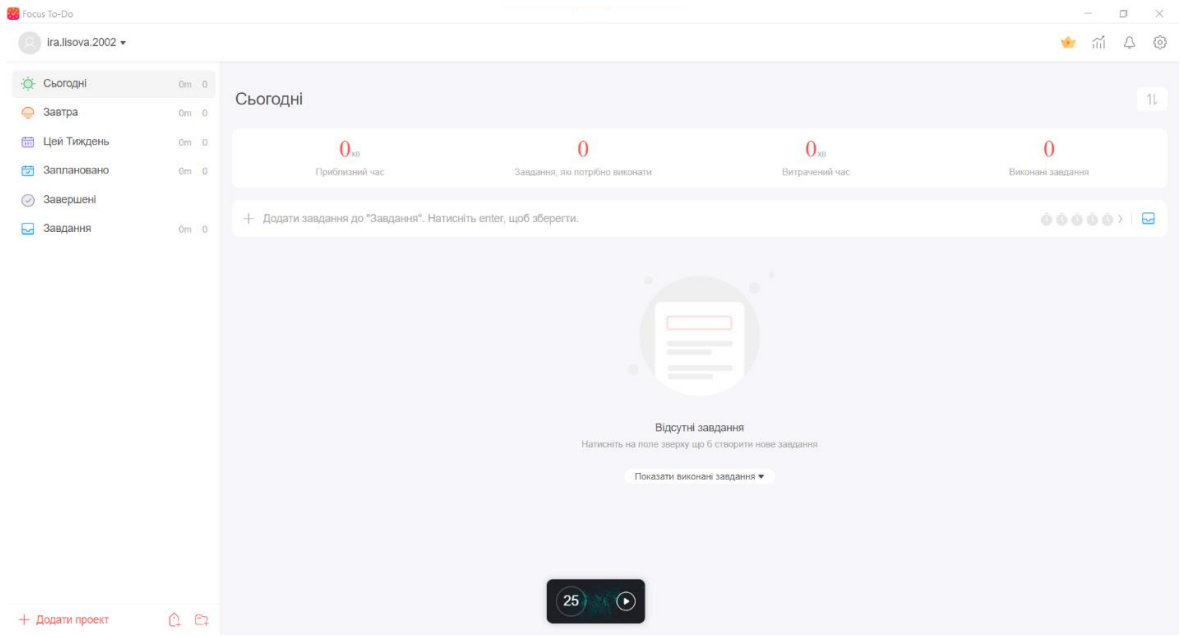

Рисунок 2.2 – Головний екран програми Focus To-Do в ОС Windows

Меню додатку складається з таких розділів:

– сьогодні;
– завтра;
– цей тиждень;
– заплановано;
– завершені;
– завдання.

Вкладка «Сьогодні» відображає приблизну кількість часу, встановленого на всі завдання в сукупності на поточний день та час, який вже витрачений на виконані завдання. При цьому відображається кількість завдань, які потрібно виконати, та кількість вже виконаних. Ще можна створити на цій сторінці завдання на поточний день. За потреби є можливість додати кількість Pomodoro (1 Pomodoro = 25 хв). Ще можна додати нагадування, повторювання, підзавдання чи примітку.

Сторінка «Завтра» аналогічна за змістом до вкладки «Сьогодні», але стосується наступного дня після поточного. Винятком ще є те, що ця сторінка не показує часу, який був витрачений на виконані завдання та кількість таких завдань.

«Цей тиждень» – сторінка, яка повністю співпадає із «Сьогодні», але показує завдання за весь тиждень.

Вкладка «Заплановано» містить перелік завдань, які заплановані, тобто є ще не виконаними. За іншими функціональними можливостями є аналогічною до сторінки «Завтра».

«Завершені» – сторінка, яка містить перелік завершених завдань.

«Завдання» містить такий самий функціонал, як і вкладки «Сьогодні» та «Цей тиждень», але відображає всі завдання за весь час.

На будь-якій сторінці є можливість видалити завдання чи встановити для нього пріоритет.

В техніці Pomodoro можна обрати звук, під який виконувати завдання за необхідності.

Звіт в програмі містить такі дані:

– загальний час у фокусі (час витрачений на виконання завдань);
– час у фокусі цього тижня (час витрачений на виконання завдань за поточний тиждень);
– час у фокусі сьогодні (час витрачений на виконання завдань за поточний день);
– всього виконаних завдань (кількість виконаних завдань за весь час);
– завдання, виконані на цьому тижні (кількість завдань, які виконані на поточному тижні);
– завдання виконані сьогодні (кількість завдань, які виконані в поточний день);
– Pomodoro-документація (відображення часу витраченого на виконання завдань з технологією Pomodoro у вигляді своєрідного графіку, де по одній осі відображені дати, а по іншій час витрачений на виконання завдання з Pomodoro);
– час у фокусі (час витрачений на виконання завдань, відображений у вигляді рядків прогресу);
– розподіл часу проєкту (представлений у вигляді кругової діаграми);
– час фокусування на цілі (календар із відміченим прогресом по кожній даті);
– діаграма часу фокусування (стовпчаста діаграма, співвідношення дати до часу);
– діаграма завдання (стовпчаста діаграма, співвідношення дати до кількості завдань).

Налаштування програми містять:

– загальні (де можна відредагувати наявні сторінки (проєкти));
– налаштування таймеру Pomodoro;
– налаштування звуків;
– налаштування облікового запису;
– налаштування можливості придбання преміум (додаткових) функцій;

– інформація про додаток.

Користувачі без придбаного преміум (додаткових можливостей) мають доступ не до всіх можливостей програми, але основні функції є доступними для всіх.

2.3 To-do List - Schedule Planner

To-do List - Schedule Planner – програма для відстежування списку справ та планування розкладу.

Інтерфейс програми є достатньо простим та ефективним. Однією із основних переваг застосунку є можливість налаштування щоденного віджету списку справ.

Є можливість встановити нагадування, щоб не забути про важливі завдання. Також є повторювані нагадування для завдань, які мають бути виконані неодноразово.

Керувати справами можна за допомогою календаря, категорій, основних моментів і контрольних списків.

Основні моменти – відмічення деяких справ зірочкою, таким чином вони вважаються більш важливими. Контрольні списки – списки підзавдань, які можна створити для об’ємного завдання.

Можна синхронізувати списки справ в хмару за допомогою Google Drive.

В цій програмі є можливість додати віджет списку справ до робочого столу телефону. За допомогою цього віджету можна також відмічати завдання, як завершені по закінченню їх виконання.

Додаток є доступним для мобільних пристроїв із ОС Android та iOS [13].

На рис. 2.3 показано головний екран додатку To-do List - Schedule Planner для ОС Android.

В верхній частині вкладки «Завдання» відображені категорії завдань. Якщо натиснути на якусь із категорій, то будуть відображені завдання саме

для цієї категорії. Після назв категорій розміщений значок трьох крапок, який відкриває додаткові можливості:

- управління категоріями (можливість видаляти, створювати, приховувати, редагувати категорії);
- пошук (дозволяє проводити пошук по всіх завданнях та всіх категоріях);
- сортування завдань (можливість сортування завдань за датою та часом, часом створення завдання, алфавітом, оберненим алфавітом, вручну);
- роздрукувати (можливість роздрукувати список справ);
- оновити до PRO.

На цій вкладці також є функція додати завдання. Під час створення завдання є можливість відразу віднести його до певної категорії, вказати дату, час, нагадування, повторення. Ще є можливість відразу створити підзавдання. Дозволено обрати із готових шаблонів завдань, якщо там є необхідне користувачу завдання.

Коли завдання створено, його можна позначити символом (прапорцем, номером, прогресом, настроєм) та відмітити його, як важливе за потреби. Ще є можливість змінити, чи за відсутності додати, дату, час, нагадування, повторення для вже створеного завдання. Також для вже створеного завдання є можливість додати підзавдання, примітки чи вкладення. Дозволено позначити завдання, як виконане, створити дублікат завдання, роздрукувати, почати зосереджуватися (поставити таймер на вказаний проміжок часу), поділитися та видалити.

На вкладці «Календар» розміщений календар. При натисканні на конкретну дату будуть відображатися справи, заплановані на обраний день. На цій сторінці також можна редагувати завдання. При натисканні на завдання можна створити його, якщо натиснути на відповідну для цього кнопку. Також є можливість за потреби синхронізувати календар із системним календарем, чи Google календарем. Ще за необхідності можна обрати альтернативні

календарі із запропонованого переліку. В наявному переліку є вісім таких календарів.

«Мій профіль» – сторінка, яка дає можливість зайти через обліковий запис Google, що дасть змогу, за необхідності, синхронізувати дані. Також на цій вкладці є можливість оновити додаток до PRO. Ще можна переглянути статистику (кількість виконаних завдань, кількість невиконаних завдань, графік із днями тижня та кількістю виконаних завдань, кругова діаграма із відображенням кількості завдань за категоріями).

Рисунок 2.3 – Головний екран додатку To-do List - Schedule Planner для ОС Android

Внизу сторінки розміщена кнопка для відкриття бокового меню. В цьому меню дублюється можливість оновлення до PRO. Також тут є можливість відкрити завдання із зірочкою (на цій сторінці будуть відображені лише важливі завдання, які користувач позначив зірочкою). Є пункт меню «Категорія», який розгортає вкладки різних категорій. При натисканні на одну з категорій буде відкрита сторінка, де буде відображено всі завдання наявні для цієї категорії. Пункт меню «Тема» дозволяє змінити тему додатку на чистий колір, текстуру, тему (картинку) із запропонованих, або на власну

картинку чи фото. «Віджет» – пункт меню, в якому можна обрати віджет для екрану (стандартний, легкий, місяць, тиждень, зворотній відлік (до певної події)). Деякі функції в програмі є недоступними, якщо користувач немає PRO. «Родина додатків» показує додатки від цього ж розробника. «Зворотній зв'язок» дає можливість написати на електронну пошту додатку в разі виникнення питань, або виявленні несправності в роботі застосунку. Також в цьому меню можна знайти відповіді на найпоширеніші запитання в розділі «FAQ». Останнім пунктом меню є налаштування програми (синхронізація облікового запису, віджет, сповіщення та нагадування, тема, синхронізація події з системним календарем, тон завершення завдання, дата та час, налаштування зовнішнього вигляду завдань (їх сортування), про програму).

2.4 TickTick - перелік завдань

TickTick - перелік завдань – програма, яка допомагає організувати, як робочі, так і персональні завдання та цілі. В додатку є можливість встановити нагадування, щоб не забувати про справи. Застосунок містить п'ять різних виглядів календарів для зручного користування різними користувачами. Є можливість ділитися планами з друзями, сім'єю, колегами.

Програма є доступною для користувачів Windows, Android Phone, Android Tablet, iPhone, iPad, Apple Watch, Mac, Linux, Web версія, розширення Chrome, розширення Firefox, надбудова Outlook, надбудова Gmail [14].

Для входу в додаток треба ввійти за допомогою Facebook, Google, email, або створити акаунт.

На рис. 2.4 представлено головний екран додатку TickTick - перелік завдань для ОС Android.

Меню додатку (бокове) містить такі розділи:

– сьогодні;
– вхідні;
– вітаємо;

– work;

– personal.

Також є можливість додати власний перелік (розділ).

На сторінці «Сьогодні» є можливість переглянути завдання на поточний день. Також можна створити завдання (встановити за потреби дату, нагадування, повторення, тривалість, задати пріоритет, позначити тегом, додати до категорії (переліку)). «Вхідні» – розділ, де відображені всі раптові завдання. На вкладці «Вітаємо» представлена своєрідна інструкція по виконанню певних дій в програмі. «Work» відображає список всіх справ внесених для цієї категорії, аналогічно для категорії «Personal».

Є календар, де при виборі дати можна переглянути справи заплановані на обрану дату. Також тут є можливість створити завдання. Можна обрати будь-який календар із п’яти запропонованих (перелік, місяць, день, 3 дні, тиждень). Налаштування додатку представлені такі: вигляд, дата та час, звуки та сповіщення, загальні, імпорт та інтеграція, інформація про додаток, можливість оновити до PRO.

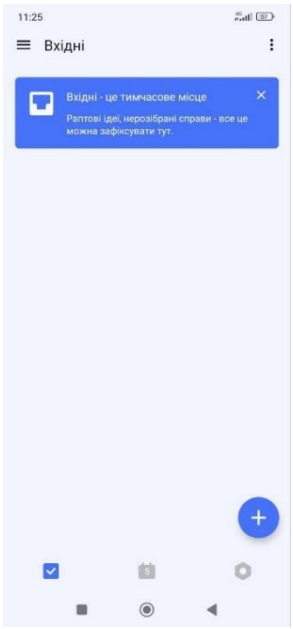

Рисунок 2.4 – Головний екран додатку TickTick - перелік завдань для ОС Android

2.5 Висновки до другого розділу

На основі проведеного огляду інтерфейсу та функціональних можливостей наявних рішень для вирішення проблеми систематизації запланованих завдань та подій можна підвести підсумок, які функції є спільними для всіх програм та ті, які є особливостями конкретного програмного рішення.

В роботі було розглянуто чотири додатки: Microsoft To Do, Focus To-Do, To-do List - Schedule Planner, TickTick - планувальник завдань.

Для всіх цих додатків є характерні такі спільні функції, як створення завдання, додавання дати та часу виконання, сповіщення (нагадування), відмітка виконання справи, видалення завдання. Ці функції можна назвати базовими для додатків, які вирішують завдання планування.

Але кожен додаток має й свої особливості. В деяких додатках є можливість розділення завдань за категоріями. Ще можливо є календар, який дозволяє відслідковувати заплановане. Також в декількох додатках є можливість переглядати статистику в різному вигляді (кількість завдань, графіки, стовпчасті чи кругові діаграми). В одному з додатків, які були розглянуті була можливість використання спеціальної техніки для концентрування на завданні – Pomodoro.

З огляду на основні функції додатків, які були розглянуті та особливих можливостей деяких з програм буде доцільно розробити застосунок, який буде поєднувати основну функціональність та використання деяких особливих функцій, які будуть доречними для вирішення задачі систематизації запланованих завдань та подій.

РОЗДІЛ 3.
ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ – СТВОРЕННЯ ДОДАТКУ ДЛЯ СИСТЕМАТИЗАЦІЇ ЗАПЛАНОВАНИХ ЗАВДАНЬ ТА ПОДІЙ

3.1 Середовище розробки додатку – Visual Studio 2022

Для розробки додатку було використано інтегроване середовище розробки Visual Studio 2022. Це середовище дозволяє створювати, редагувати, налагоджувати код та публікувати програму. Для процесу покращення розробки програмного забезпечення Visual Studio 2022 містить, крім стандартного редактора та налагоджувача, компілятори, інструменти завершення коду, графічні дизайнери та ще чимало корисних функцій.

Це інтегроване середовище для розробки надає можливість створювати програми для різних пристроїв та можливість діагностувати та виправляти проблеми [15].

3.2 Фреймворк .NET MAUI

.NET MAUI є кросплатформенним фреймворком для створення мобільних та комп’ютерних програм з використанням C# та XAML. Цей фреймворк дозволяє через спільну кодову базу реалізовувати програму для Android, iOS, macOS та Windows (рис. 3.1).

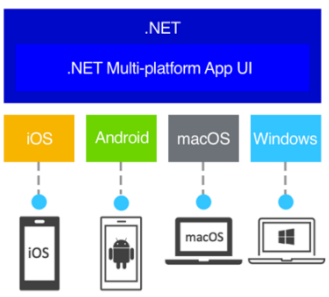

Рисунок 3.1 – Операційні системи для реалізації проєкту на .NET MAUI [16]

.NET MAUI – еволюція Xamarin.Forms, розширена від мобільних до комп'ютерних сценаріїв. За допомогою цього фреймворку є можливість створювати додатки для декількох платформ за допомогою одного проєкту.

Кросплатформенний фреймворк .NET MAUI об'єднує Android, iOS, macOS і Windows API в один API, що дає можливість розробнику працювати із одним спільним кодом, але за потреби можна писати специфічні можливості для кожної платформи окремо.

.NET MAUI для Android, .NET для iOS, .NET для macOS, бібліотека Windows UI 3 (Win UI 3) – фреймворки, які мають доступ до бібліотеки базових класів .NET(BCL). Ця бібліотека абстрагує деталі базової платформи від коду. Середовище виконання для Android, iOS, macOS – Mono, а для Windows – .NET Core CLR.

Фреймворк .NET MAUI має єдину структуру для створення інтерфейсу користувача, як для комп'ютерних, так і для мобільних додатків (рис. 3.2).

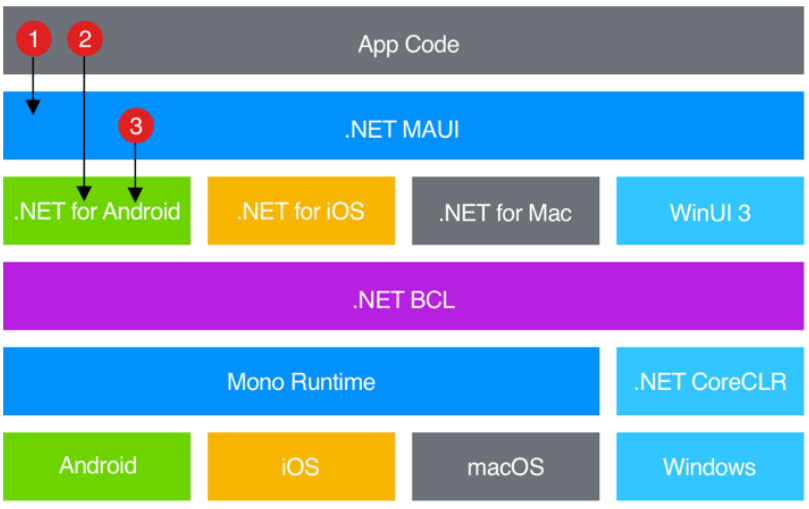

Рисунок 3.2 – Високорівневе подання архітектури .NET MAUI [17]

Код, який написаний, переважно взаємодіє із API .NET MAUI. Після чого вже .NET MAUI використовує безпосередньо API рідної платформи. За потреби код програми може використовувати відразу API платформи [18].

3.3 Мова програмування C#

C# – об’єктно-орієнтована мова програмування. C# дозволяє створювати розробникам чимало програм, які працюють у .NET. Ця мова програмування бере початок із сімейства мов програмування C.

Створювати надійні та довговічні програми C# допомагають деякі функції:

- збирання сміття (якщо є невикористані чи недоступні об’єкти, то вони автоматично звільняються з пам’яті);
- типи, що допускають значення Nullable (захищають від змінних, які не посилаються на виділені об’єкти);
- обробка винятків (структурований підхід до виявлення та виправлення помилок);
- лямбда-вирази (підтримують методи функціонального програмування);
- синтаксис Language Integrated Query (LINQ) (створює загальний шаблон для роботи з даними з будь-якого джерела);
- підтримка асинхронних операцій (забезпечує синтаксис для побудови розподілених систем).

Уніфіковану систему типів має C#, тобто всі типи успадковуються від одного кореневого object типу. Усі типи мають спільний набір спільних операцій. Значення змінних будь-якого типу можна зберігати, транспортувати та працювати з ними узгоджено. Крім цього, C# підтримує, як визначені користувачем типи посилань, так і типи значень. Підвищену безпеку типів та продуктивність забезпечує підтримка загальних методів та типів. C# надає ітератори, які дають можливість розробникам класів колекцій визначати користувальницьку поведінку для клієнтського коду.

C# звертає увагу на версії, щоб бібліотеки та програми могли розвиватися сумісним чином. Аспекти C# включають модифікатори virtual і override (явна декларація членів інтерфейсу та перевантаження методів) [19].

3.4 Архітектура програми

Розробка програми передбачала побудову візуального інтерфейсу та реалізацію логіки програми та обробки даних.

Є діаграми, які описують структуру програмного забезпечення та ті, які описують поведінку програмної системи.

Повна діаграма класів застосунку, яка описує структуру програми та взаємодію класів представлена на рис. 3.3.

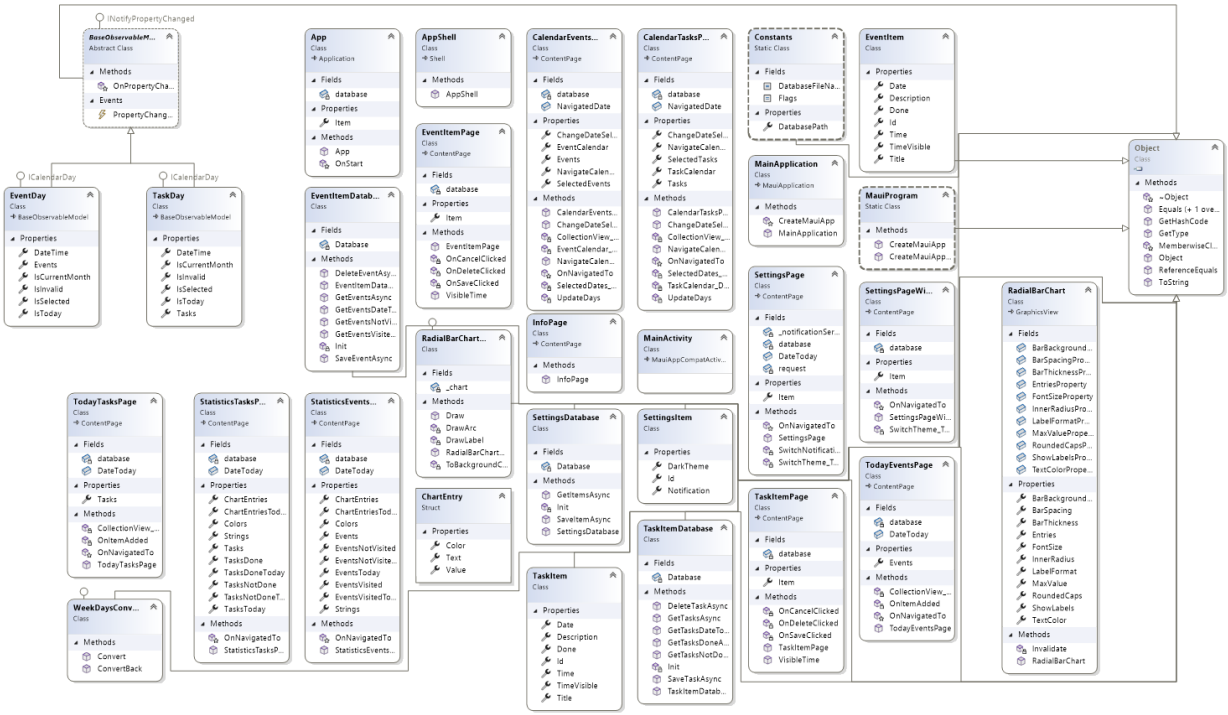

Рисунок 3.3 – Повна діаграма класів програми

Діаграма класів – це важлива статична діаграма, яка відображає статичний вигляд додатку та використовується для моделювання різних аспектів системи, таких як атрибути, операції, обмеження, асоціації та співпраці між класами та інтерфейсами. Ця діаграма зображує колекцію класів та інтерфейсів, які можуть бути відображені на об'єктно-орієнтованих мовах програмування для розробки виконуваного коду програмного застосування.

Окрім того, діаграма класів дозволяє візуалізувати зв'язки між класами та інтерфейсами, такі як асоціації, залежності, співпраці та інші, та використовується для опису системи на високому рівні абстракції. Це дозволяє розробникам легше розуміти структуру системи та знайти спосіб оптимізувати її функціональність. Також важливою складовою діаграми класів є обмеження, які дозволяють забезпечувати правильну взаємодію між класами та інтерфейсами в системі. В результаті, діаграма класів може допомогти в розумінні та описі складної системи, та є потужним інструментом для розробки програмного забезпечення [20].

Діаграма класів є однією з багатьох діаграм програмного забезпечення, яку можна використовувати окремо або в контексті більш широкої UML діаграми програмного забезпечення.

Діаграма варіантів використання, яка описує поведінку програми, взаємодію з користувачем представлена на рис. 3.4.

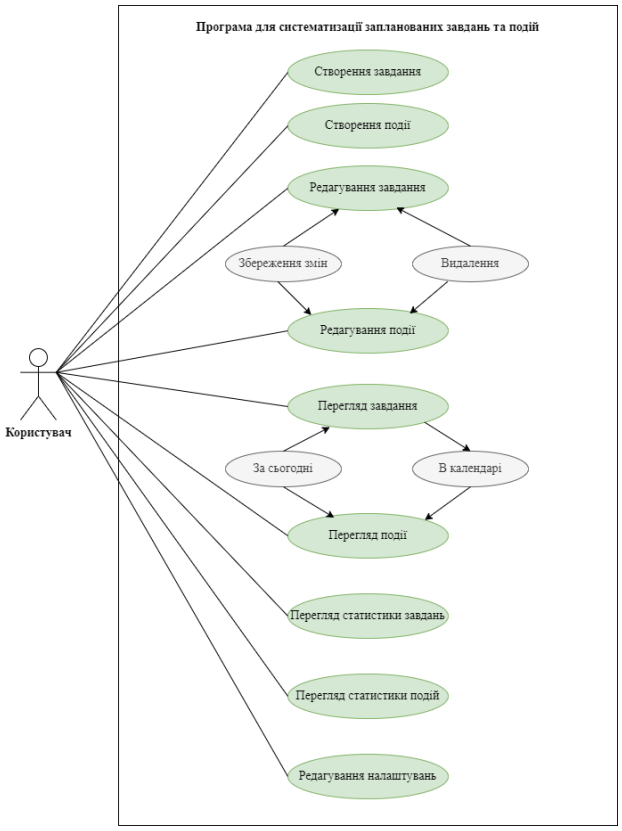

Рисунок 3.4 – Діаграма варіантів використання програми

3.5 Вбудована реляційна локальна база даних SQLite для збереження даних застосунку

SQLite – це бібліотека, що працює в процесі виконання програми та реалізує локальну базу даних SQL з механізмом транзакцій, яка не потребує налаштування серверу. SQLite є найбільш поширеною базою даних у світі з неосяжною кількістю програм, серед яких є кілька дуже відомих проєктів.

SQLite – це вбудований механізм локальної бази даних SQL, який відрізняється від більшості інших баз даних SQL тим, що не потребує окремого серверного процесу. Замість цього, SQLite читає та записує дані безпосередньо у звичайні дискові файли. Одна повна локальна база даних SQL, яка включає кілька таблиць, індексів, тригерів та представлень, може знаходитись в одному дисковому файлі [21].

Для збереження даних про завдання створюється таблиця TaskItem з стовпцями Date, Description, Done, Id, Time, TimeVisible, Title. Друга таблиця EventItem містить аналогічні стовпці, як TaskItem, тільки вже зберігає дані про події, а не завдання. Третя таблиця SettingsItem містить стовпці DarkTheme, Id, Notification та зберігає інформацію про налаштування програми.

3.6 Системні вимоги

Для успішної роботи програми персональний комп’ютер має відповідати таким вимогам:

– встановлена Windows 11 або Windows 10 версії 1809 (чи пізнішої версії) з використанням Windows UI Library (WinUI) 3.

Для правильної роботи програми на мобільному телефоні мають бути виконані такі вимоги:

– встановлена версія Android 5.0 (API 21) чи вище версія.

Для обох варіантів (Windows, Android) має бути достатня кількість пам’яті для встановлення додатку.

3.7 Інсталяція програми

Вимоги до програмного забезпечення, яке має бути попередньо встановлено:

- встановлений .NET Framework версії 7.0;
- встановлене середовище розробки Microsoft Visual Studio 2022 та додаткові розширення (одним з яких є «Розробка за допомогою .NET Multi-Platform App UI»).

Перевірити роботу додатку на телефоні можна за допомогою емулятора телефону (використовуючи Android Device Manager), або встановити додаток на власний телефон.

Встановити додаток на власний телефон можна декількома способами. Одним із таких способів є встановлення через USB. Для цього треба ввімкнути на телефоні режим розробника та змінити деякі налаштування в вкладці «Для розробників». Після чого підключившись до ноутбука через USB запустити програму в Visual Studio, вибравши власний телефон в якості пристрою на якому має запуститися програма.

3.8 Користувацький інтерфейс застосунку

Користувацький інтерфейс у додатку був описаний через XAML.

Головна навігація програми була описана в елементі Shell. Цей елемент вміщує в собі декілька FlyoutItem (цей елемент відображається, як пункт меню в Shell). А кожен FlyoutItem в свою чергу містить один або декілька ShellContent, який відкриває певну сторінку (ContentPage). Всі звичайні елементи, такі як кнопки, місця для введення, тощо, знаходяться в ContentPage.

Код, який написаний для користувацького інтерфейсу програми є загальним та застосовується, як для додатку для комп’ютера (ноутбука), так і для телефону.

Основна частина лістингу інтерфейсу представлена в додатку А.

3.9 Проєктування структури програми

Одними з основних класів програми є TodayTasksPage, TaskItemPage, TaskItemDatabase. Це основні класи, які використовуються для додавання, видалення, редагування завдань. Для подій класи є аналогічними, тільки зберігають інформацію про події.

Клас TodayTasksPage показує завдання заплановані на поточний день, структура цього класу представлена в таблиці 3.1.

Таблиця 3.1

Проєктування структури класу TodayTasksPage

Назва	Тип	Опис
Властивості		
-database	readonly TaskItemDatabase	об’єкт класу бази даних з таблицею завдань
+Tasks	ObservableCollection<TaskItem>	колекція завдань
+DateToday	DateTime	поточна дата
Методи		
+TodayTasksPage (TaskItemDatabase taskItemDatabase)		конструктор
#OnNavigatedTo (NavigatedToEventArgs args)	override async void	оновлює список завдань на сьогодні
-OnItemAdded (object sender, EventArgs e)	void	виконує перехід на сторінку, де можна додати нове завдання
-CollectionView_SelectionChanged (object sender, SelectionChangedEventArgs e)	async void	виконує перехід на сторінку, де можна редагувати завдання

Клас TaskItemPage дозволяє створювати та редагувати завдання, структура цього класу представлена в таблиці 3.2. Якщо відкривати цю сторінку натисканням кнопки для створення завдання, то буде можливість створити завдання. Але якщо відрити завдання натисканням на нього в переліку поточних завдань чи на день, то буде можливість редагувати чи видалити завдання завдяки класу TaskItemPage.

Таблиця 3.2

Проєктування структури класу TaskItemPage

Назва	Тип	Опис
Властивості		
-database	TaskItemDatabase	об’єкт класу бази даних з таблицею завдань
+Item	TaskItem	об’єкт класу з стовпцями таблиці
Методи		
+TaskItemPage (TaskItemDatabase taskItemDatabase)		конструктор
-OnSaveClicked (object sender, EventArgs e)	async void	викликає метод, який зберігає завдання
-OnDeleteClicked (object sender, EventArgs e)	async void	викликає метод, який видаляє завдання
- OnCancelClicked(object sender, EventArgs e)	async void	виконує перехід на попередню сторінку
+VisibleTime(object sender, ToggledEventArgs e)	void	перемикає видимість часу для зазначення часу за необхідності

Клас TaskItemDatabase працює з базою даних, структура цього класу представлена в таблиці 3.3. Цей клас дає змогу створювати, видаляти, редагувати завдання, при цьому зміни зберігаються в локальну базу даних SQLite.

Таблиця 3.3

Проєктування структури класу TaskItemDatabase

Назва	Тип	Опис
Властивість		
-Database	SQLiteAsyncConnection	об’єкт бази даних
Методи		
+TaskItemDatabase()		конструктор
-Init()	async Task	ініціалізує базу даних та створює таблицю для завдань
+GetTasksAsync()	async Task<List<TaskItem>>	отримує всі завдання із таблиці завдань
+GetTasksDateTodayAsync (DateTime dateTime)	async Task<List<TaskItem>>	отримує всі завдання на поточний день із таблиці завдань
+GetTasksNotDoneAsync()	async Task<List<TaskItem>>	отримує всі невиконані завдання
+GetTasksDoneAsync()	async Task<List<TaskItem>>	отримує всі виконані завдання
+ SaveTaskAsync (TaskItem taskItem)	async Task<int>	зберігає завдання
+DeleteTaskAsync (TaskItem taskItem)	async Task<int>	видаляє завдання

3.10 Сценарій роботи користувача з програмою

При запуску програми користувач бачить сторінку із завданнями на поточний день та має можливість відкрити меню, яке допомагає орієнтуватися в додатку (рис. 3.5).

Меню реалізовано за допомогою класі Shell. Кожна позиція в меню є елементом FlyoutItem, а об’єкт ContentPage представлений через об’єкт ShellContent.

Меню складається з таких сторінок:

– «Завдання»;
– «Події»;
– «Налаштування»;
– «Про додаток».

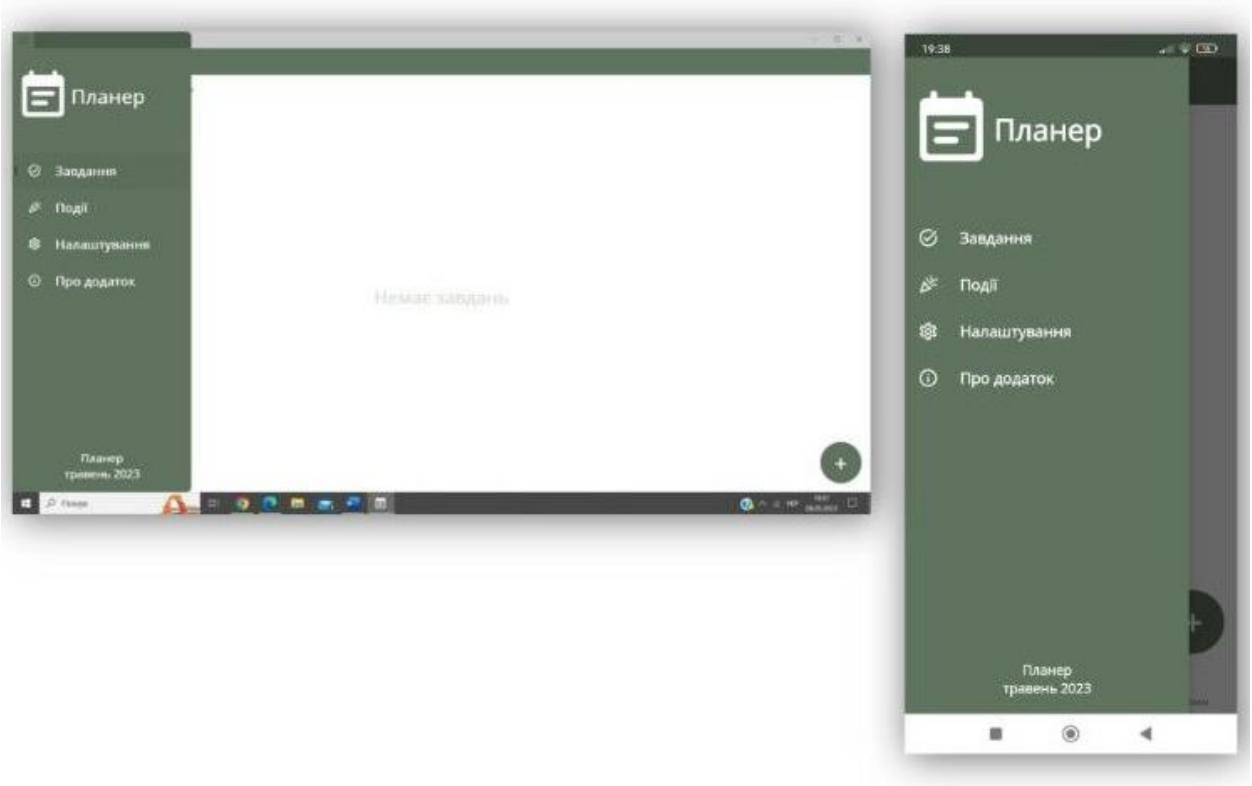

Рисунок 3.5 – Меню додатку для Windows та Android

На сторінці «Завдання» користувачу представлені декілька вкладок: «Сьогодні», «Календар», «Статистика».

Вкладка «Сьогодні» відображає завдання, які є на поточний день, якщо таких завдань немає, то відображається відповідне повідомлення. Також тут є можливість натиснути на кнопку «+», яка відкриє сторінку для додавання нового завдання (рис. 3.6). На цій сторінці можна додати завдання, якщо записати необхідну інформацію в відповідні поля. Якщо цю сторінку викликати натисканням на завдання, то можна редагувати або видалити.

При внесені інформації про завдання чи подію обов’язковими параметрами є назва та дата. Якщо дату не обрано, то буде внесена дата поточного дня. Якщо не обрану назву, то буде виведено повідомлення про необхідність введення назви завдання чи події відповідно.

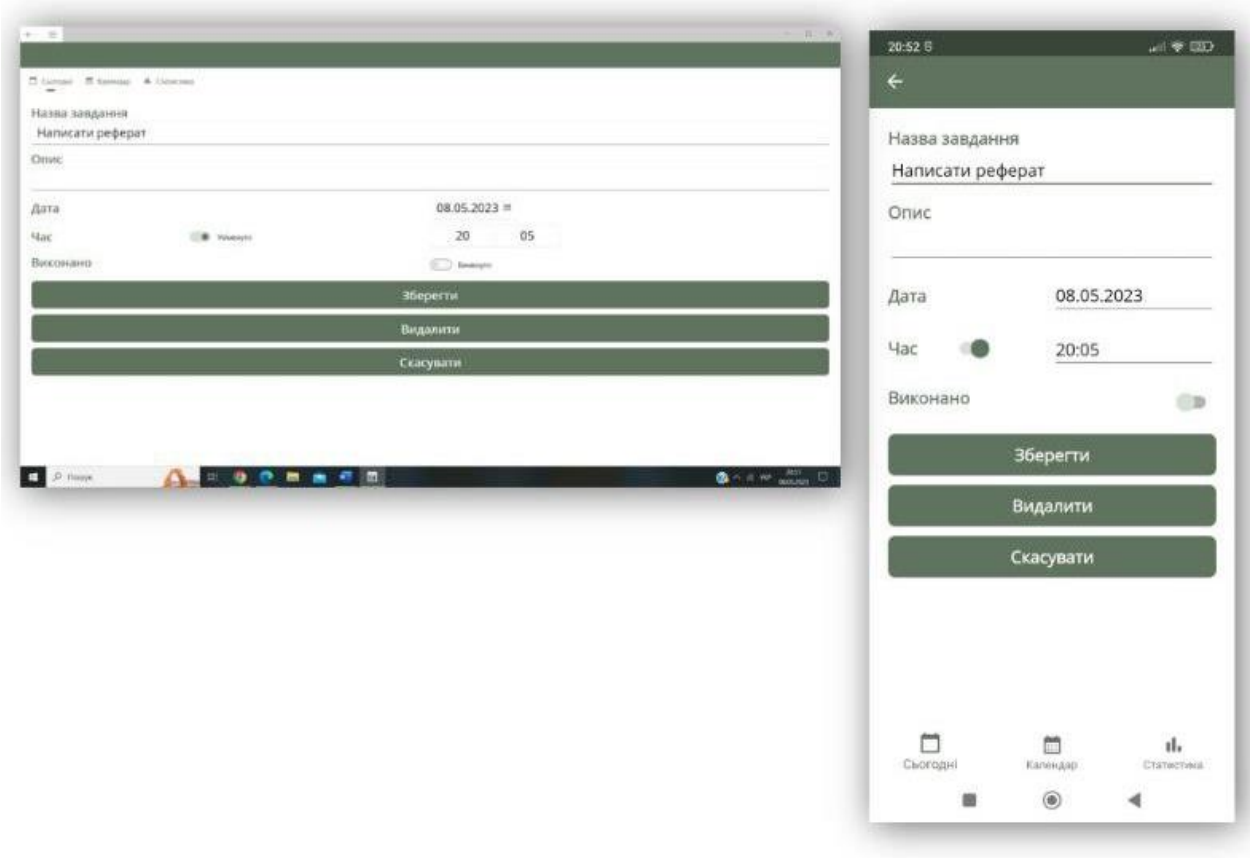

Рисунок 3.6 – Створення нового завдання на вкладці «Сьогодні»

На рис. 3.7 представлена вкладка «Сьогодні» після додавання декількох завдань (три на поточний день та два через один день від поточного).

Внесені завдання зберігаються в базі даних. Отже, на вкладці «Сьогодні» після створення завдань можна побачити три завдання. На поточний день було записано три завдання, тому можна зробити висновок, що програма працює правильно (адже інші два завдання були створенні не для поточного дня, тому відображатися на цій вкладці не повинні).

Якщо завдань немає, то буде відповідний напис про їх відсутність.

При натисканні на завдання на цій вкладці, відкриється сторінка із можливістю редагувати або видалити його. Заголовок вкладки буде завжди демонструвати поточну дату. При внесені великою кількості завдань буде можливість гортати сторінку.

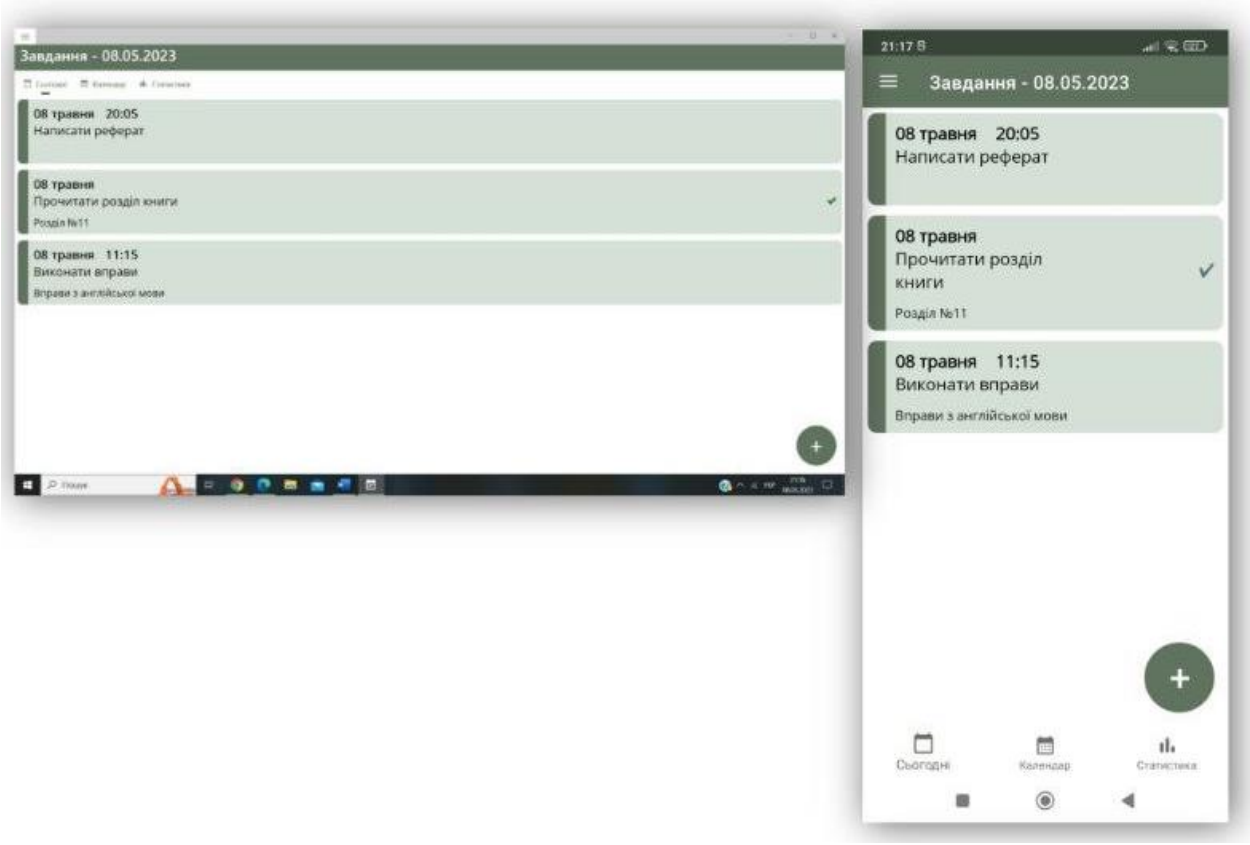

Рисунок 3.7 – Вкладка «Сьогодні» з веденими завданнями

На рис. 3.8 наведено приклад відкриття сторінки для редагування чи видалення при натисканні на перше завдання поточного дня, яке було створено раніше.

Вгорі на сторінці пишеться назва поточного завдання. Для збереження змін треба зробити зміни та натиснути кнопку «Зберегти». Для видалення завдання необхідно натиснути кнопку «Видалити». Для відмітки про виконання – натиснути перемикач з написом «Виконано» і натиснути кнопку «Зберегти».

Редагування, збереження змін та видалення реалізуються завдяки SQLite. Для цього було створено таблицю, яка зберігає стовпці з інформацією про завдання. Було реалізовано клас, який містить методи для збереження, видалення інформації в таблиці бази даних. Також в цьому класі є методи для виведення інформації з бази даних.

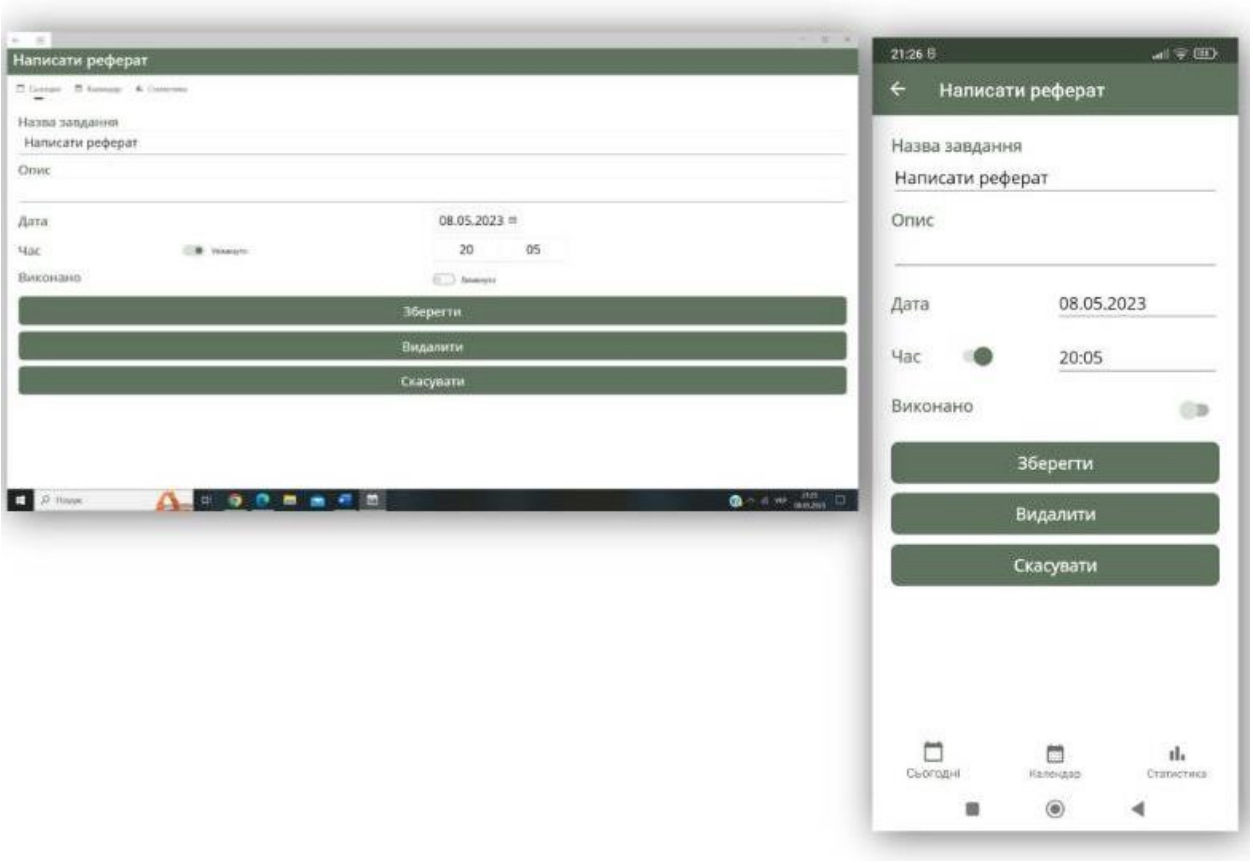

Рисунок 3.8 – Сторінка редагування першого завдання

На вкладці «Календар» можна переглянути завдання по датам (рис. 3.9). Ця інформація також береться з бази даних, виведення по датам можливе завдяки тому, що кожне завдання при створенні має дату (якщо дата не обрана, то буде збережена дата поточного дня).

При натисканні на певну дату відкривається список завдань на обраний день. При натисканні на завдання в цьому списку можна відкрити сторінку для редагування, видалення завдання.

Якщо на певний день нічого не заплановано, то при натисканні на такий день замість списку завдань буде відображено відповідне повідомлення (про відсутність завдань).

Без натискання на дату, можна побачити через індикатори (кружечки) кількість завдань на певний день. Чітко можна побачити коли завдань від одного до трьох, далі вже можна просто зрозуміти, що їх записано більше трьох (індикаторам не вистачає місця в кружечку дати для більшої кількості).

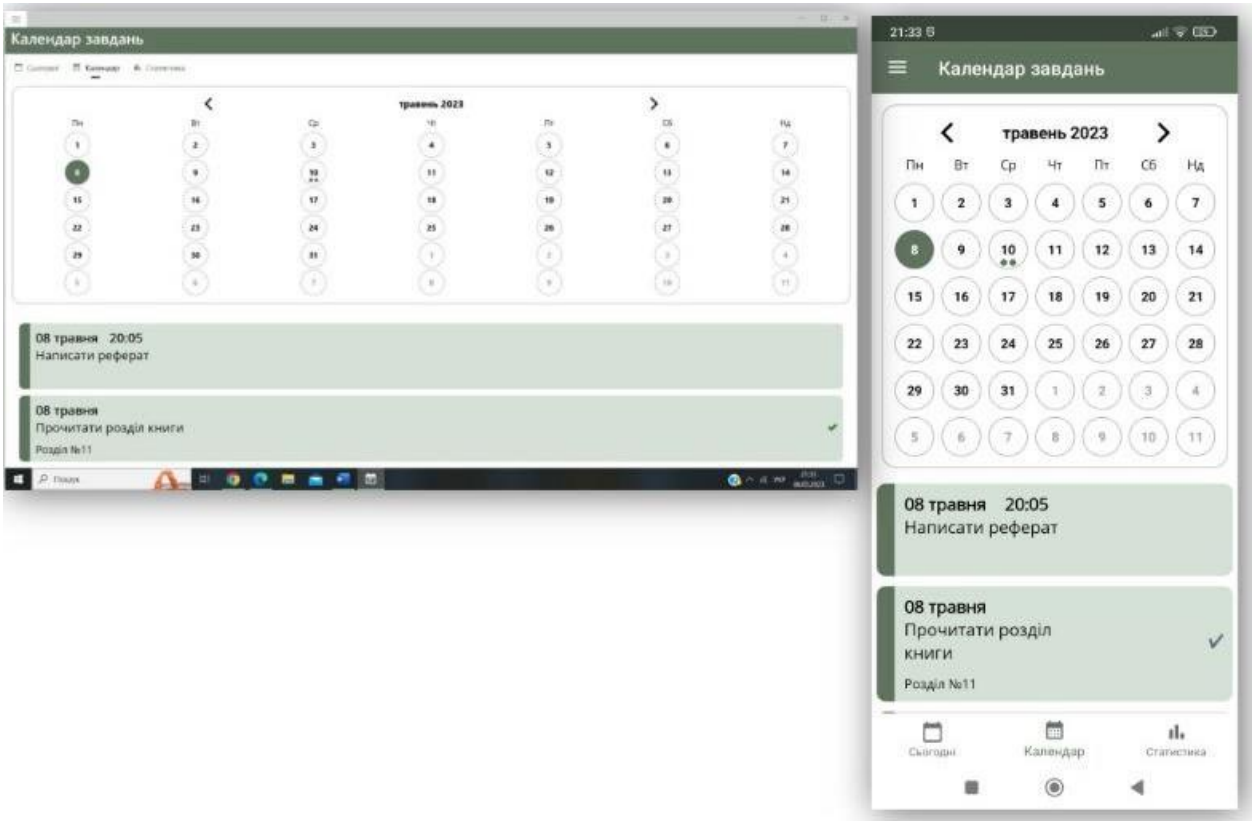

Рисунок 3.9 – Вкладка «Календар»

На вкладці «Статистика» можна переглянути статистику по виконанню завдань за поточний день та весь час (рис. 3.10).

В обох випадках міститься числова інформація про виконані та не виконані завдання та загальна їх кількість.

Також є діаграма, яка відображає графічно відношення виконаних та не виконаних завдань.

Для перегляду обох статистик можна гортати сторінку, адже вміст сторінки може бути великим для певного пристрою.

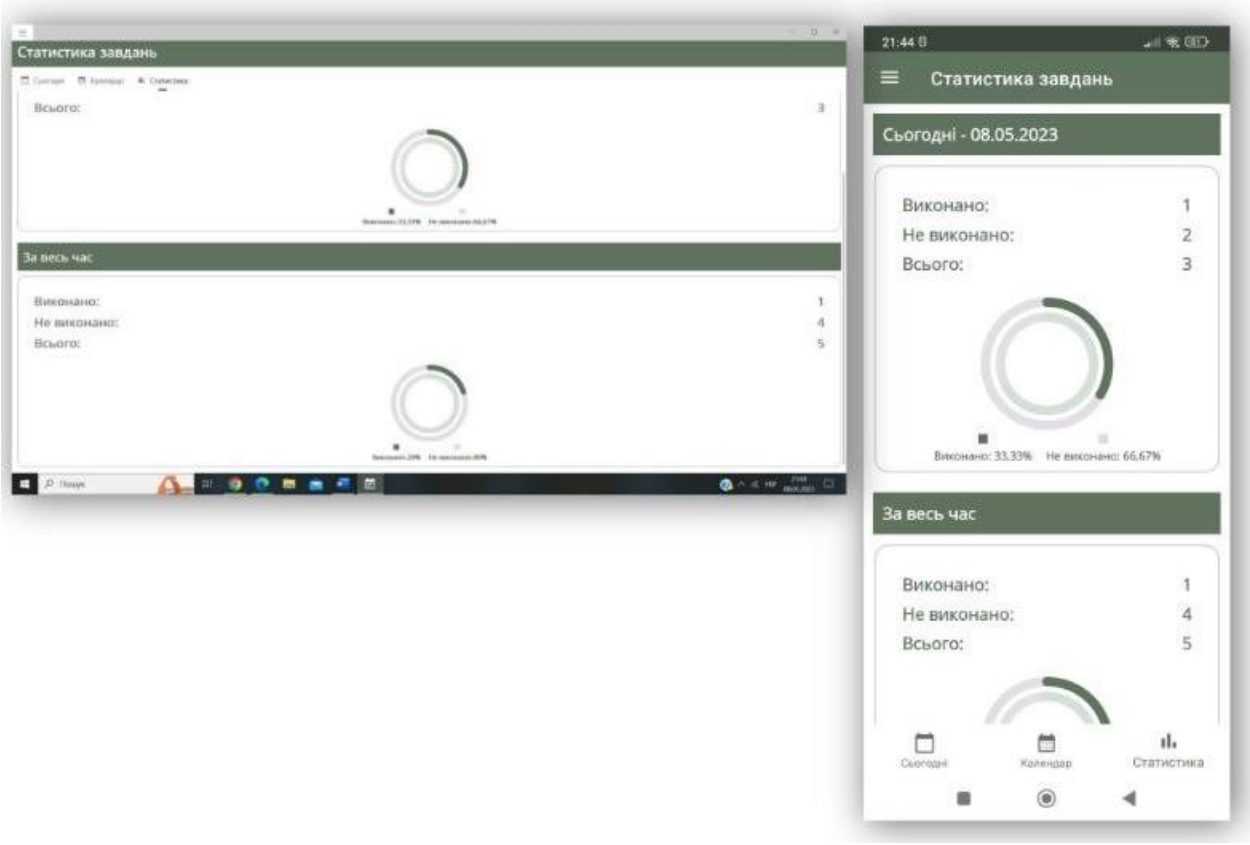

Рисунок 3.10 – Статистика по виконанню завдань за поточний день

Сторінка «Події» аналогічна до сторінки «Завдання», за винятком того, що зберігає дані про події (дані розміщені в іншій таблиці в базі даних). На відміну від завдань помітка про виконання називається не «Виконано», а «Відвідано».

На сторінці «Налаштування» можна змінити тему додатку на темну. Також є можливість ввімкнути сповіщення, але лише для Android (рис. 3.11).

Для цієї сторінки було створено два окремих варіанти. Для Windows з одним перемикачем, та для Android з двома. Збереження стану перемикачів відбувається за допомогою ще однієї таблиці, яка мітить поля із відповідними булевими значеннями.

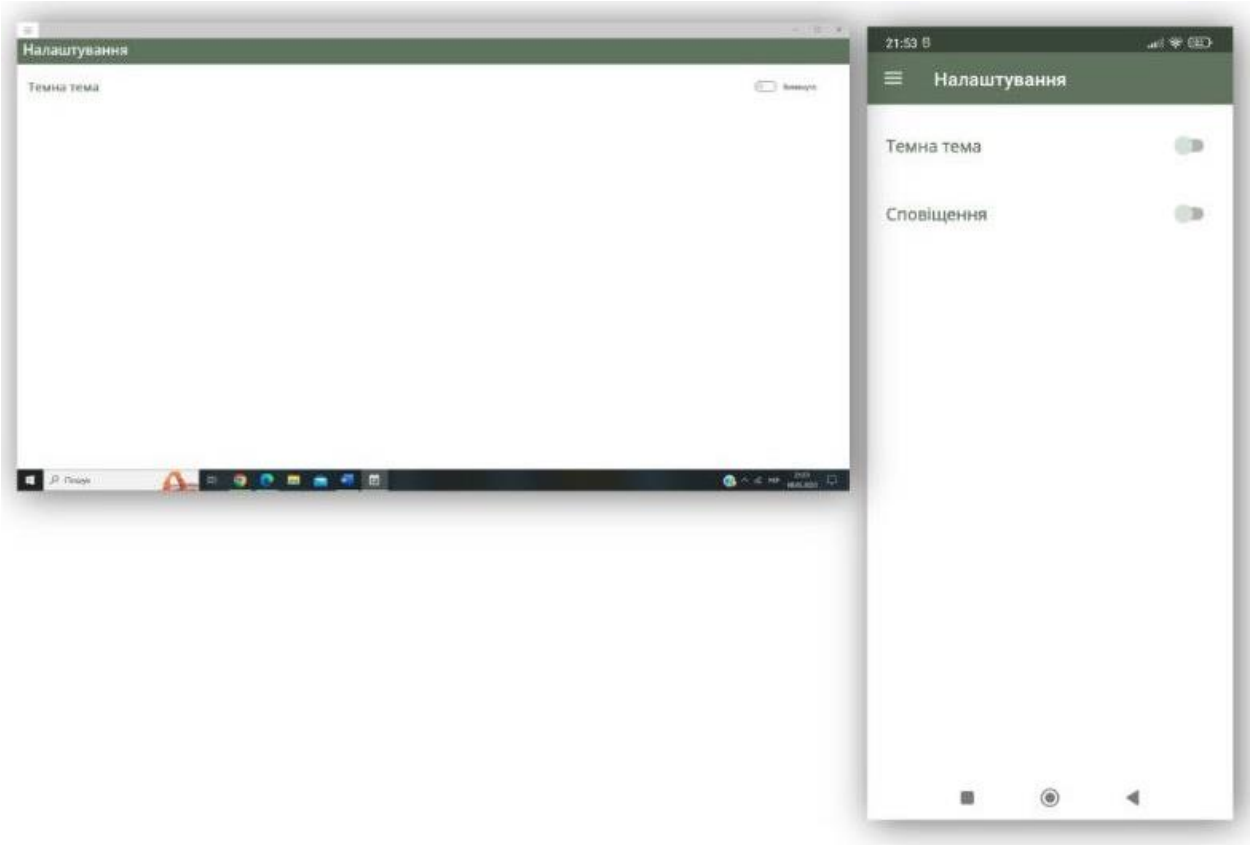

Рисунок 3.11 – Сторінка «Налаштування»

Ще є сторінка «Про додаток», вона не містить ніякого функціоналу, лише інформацію про програму. Інформація містить: ціль створення додатку, групу, виконавця та рік.

Якщо користувач намагатиметься ввімкнути сповіщення, а дозволу на сповіщення не буде, то буде виведено відповідне повідомлення про необхідність надання дозволу.

Всі повідомлення, які видають інформацію про певну помилку створені за допомогою спеціального вікна DisplayAlert. Це повідомлення містить назву та опис проблеми.

Для сповіщень (тільки Android) використовується плагін локальних сповіщень. Цей NuGet пакет дає можливість створювати сповіщення лише для мобільних пристроїв (Android та iOS), через це сповіщення для Windows є недоступними.

Якщо користувач ввімкне сповіщення та матиме дозвіл на надсилання сповіщень даним додатком, то кожного дня о восьмій годині ранку користувач отримуватиме сповіщення, яке буде нагадувати зайти в застосунок та перевірити, чи є в користувача якісь плани на поточний день (або на найближчі дні (рис. 3.12).

При натисканні на сповіщення буде відкриватися додаток, тобто користувач може відразу переглянути чи є плани (можливо створити їх).

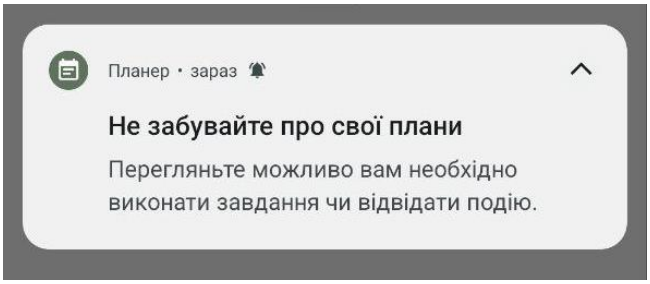

Рисунок 3.12 – Сповіщення від додатку

Основна частина функціональної частини лістингу представлена в додатку Б, де представлено назву та функцію, яку виконує певний клас. В більшості розміщено лістинг для роботи з завданнями, адже функціонал для подій є аналогічним.

3.11 Висновки до третього розділу

В результаті виконання третього розділу було розроблено програмний продукт для систематизації запланованих завдань та подій.

Програма була написана за допомогою кросплатформенного фреймворку .NET MAUI. Інтерфейс додатку був написаний на XAML, а функціональна частина програми написана об’єктно-орієнтованою мовою C#. Для збереження даних внесених користувачем була використана вбудована реляційна база даних SQLite.

В процесі виконання розділу було створено дві діаграми, які частково є UML діаграмою програмного забезпечення. Було створено діаграму класів, яка допомагає зрозуміти логіку роботи програми та зрозуміти взаємодію між класами. Також було створено діаграму варіантів використання програми, яка дозволяє зрозуміти, які можливості є в додатку та взаємодію з користувачем.

Однією із важливих частин виконання третього розділу було проєктування структури програми. Було показано проєктування структури на прикладі декількох основних класів, які відповідають за роботу із завданнями (TodayTasksPage, TaskItemPage, TaskItemDatabase). Проєктування полягало в описі властивостей та методів цих класів.

В розділі було представлено системні вимоги до пристрою для запуску програми та наведено, що необхідно для інсталяції застосунку. Також було описано, як відбувається навігація в користувацькому інтерфейсі. Останнім, але не менш важливим етапом було показано сценарій роботи користувача з програмою, що можна вважати своєрідною інструкцією користування та показом роботи застосунку одночасно.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розглянуто предметну область планування у сфері інформаційних технологій. Це планування, як загальної роботи в ІТ компанії, так і планування ІТ проєкту та планування в командній роботі. Також було розглянуто планування при впровадженні ІТ в установи, організації різних сфер діяльності.

Для успішного формулювання функцій, які має виконати програмний продукт, було розглянуто готові рішення (наявні програми). Наявні програми, які були розглянуті: Microsoft To Do, Focus To-Do, To-do List - Schedule Planner, TickTick - перелік завдань.

Були виділені усі основні функції та якість особливості, які є характерними для того чи іншого додатку. На основі проведеного огляду було вирішено, які основні функції мають бути написані в програмі.

В кінцевому результаті було створено застосунок, який містить такі функції:

– додавання, перегляд, редагування, видалення завдання;
– додавання, перегляд, редагування, видалення події;
– перегляд завдань в календарі за датами;
– перегляд подій в календарі за датами;
– перегляд статистики виконання завдань за день та за весь час;
– перегляд статистики виконання подій за день та за весь час;
– перемикання теми додатку;
– ввімкнення/вимкнення сповіщень (тільки для телефону);
– перегляд інформації про додаток.

При виконані роботи була створена діаграма, яка допомагає зрозуміти логіку роботи програми та взаємодію між класами програми – діаграма класів. Також для розуміння взаємодії користувача з програмою було створено діаграму варіантів використання. Ці дві діаграми частково описують UML діаграму програмного забезпечення.

Основною перевагою розробленого програмного продукту є його лаконічність та простота в використанні. Користувач без особливих складностей може користуватися даним продуктом та зрозуміти як все працює. Програма не вимагає реєстрації, авторизації, тому не треба буде запам’ятовувати пароль та мати хоч якісь складнощі при вході в програму. Також перевагою є те, що застосунок не перенавантажений великою кількістю зайвих функцій. Тобто для користувачів, яким потрібний максимально простий продукт для запису та моніторингу своїх подій та завдань і є потреба в розмежуванні між задачами та подіями, цей додаток буде досить доречним.

Але для користувачів, яким необхідно синхронізувати свої плани між декількома пристроями можливо буде корисно мати авторизацію. Тому, якби додаток десь використовувався за межами кваліфікаційної роботи, то можна було виходячи із коментарів та відгуків людей зрозуміти, що їм потрібніше та вдосконалити даний продукт.

Також, на мою думку покращити застосунок можна було б додаванням можливості вибору різних видів сповіщень. Наприклад, якщо користувач має потребу для того чи іншого завдання або події отримувати нагадування, то щоб була можливість таке встановити. Ще було б гарною ідеєю використати якийсь інший інструмент для написання коду, який створює сповіщення для того, щоб можна було впровадити сповіщення для персонального комп’ютера також. Ще в майбутньому можна реалізувати дану програму для macOS та iOS, адже .NET MAUI підтримує можливість такої реалізації.

Розроблений продукт є досить економічно вигідним, адже через те, що продукт був написаний за допомогою кросплатформенного фреймворку є можливість змінювати (покращувати) код з однієї кодової бази. Тобто для покращення продукту буде достатньо фахівця, який володіє можливостями написання на C# та XAML та знає як працює .NET MAUI фреймворк. Це забирає необхідність залучення спеціалістів окремо для покращення рішення для персонального комп’ютера та для телефону.

В сфері ІТ планування має надзвичайно важливе значення. Необхідність планування пояснюється різноманітністю та складністю завдань, що вирішуються в даній галузі. Від планування залежить якість та ефективність роботи команди, яка займається розробкою програмного забезпечення або іншими інженерними завданнями.

Планування дозволяє побудувати стратегію та визначити підходи до розв'язання проблеми. Крім того, дозволяє складати розклад робіт та координувати роботу між різними командами або співробітниками. Надзвичайно важливо планувати витрати часу та ресурсів, щоб досягти максимальної продуктивності.

Планування також дозволяє визначити пріоритетні завдання та ресурси, необхідні для їх виконання. Планування є ключовим елементом в будь-якому процесі розробки програмного забезпечення, адже дозволяє забезпечити високу якість та ефективність роботи.

У відсутності планування можуть виникнути проблеми, які можуть значно ускладнити роботу команди та призвести до затримок у виконанні проєкту. Крім того, планування дозволяє розподілити відповідальність між різними членами команди та визначити їх ролі та обов'язки.

Тобто можна стверджувати, що планування має надзвичайно важливе значення в сфері ІТ. Взагалі, планування є важливим елементом будь-якої діяльності, оскільки допомагає досягати поставлених цілей та забезпечувати ефективне використання ресурсів. Також планування є важливою складовою повсякденного життя кожної людини, оскільки допомагає організувати час та ресурси для досягнення поставлених цілей. Крім того, планування допомагає людині визначити пріоритети та кроки, необхідні для досягнення поставлених цілей. Також планування допомагає людині бути більш організованою та впевненою.

Отже, додаток, який було реалізовано в кваліфікаційної роботі може бути корисним, як для персонального користування, так і для використання установою, організацією в різних сферах діяльності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Charlotte Dingwall, Kristi Kemp, Barbara Fowke. A Guide on Person-Directed Planning. Why planning is important. 2006. С. 7-11. URL: https://individualizedfunding.files.wordpress.com/2014/07/a-guide-on-person-directed-planning-english.pdf (дата звернення: 07.02.2023).

2. Kate Eby. How to Successfully Plan an IT Project. 2022. URL: https://www.smartsheet.com/content/it-project-plan (дата звернення: 10.02.2023).

3. Eshna Verma. Role of Planning in Successful Project Management Framework. 2023. URL: https://www.simplilearn.com/role-of-planning-in-project-management-framework-rar226-article (дата звернення: 11.02.2023).

4. Why is project planning important? 2023. URL: https://www.jotform.com/blog/importance-of-project-planning/ (дата звернення: 15.02.2023).

5. Adrienne Watt, Merrie Barron, Andrew Barron. Overview of Project Planning. URL: https://opentextbc.ca/projectmanagement/chapter/chapter-8-overview-of-project-planning-project-management/ (дата звернення: 17.02.2023).

6. Emily Bonnie. Project Planning 101: 6 Steps to a Foolproof Project Plan. 2022. URL: https://www.wrike.com/blog/foolproof-project-plan/#How-Wrike-can-help-you-create-your-project-plan (дата звернення: 20.02.2023).

7. Cameron Nouri. Organizational Planning Guide: Types of Plans, Steps, and Examples. 2020. URL: https://pingboard.com/blog/organizational-planning-guide-types-of-plans-steps-and-examples/ (дата звернення: 21.02.2023).

8. Strategic Planning and Strategic IT Planning for Long-Term and Post-Acute Care (LTPAC) Providers. С. 11-13. URL: https://leadingage.org/wp-content/uploads/drupal/Strategic_IT_Planning_Workbook.pdf (дата звернення: 25.02.2023).

9. What is team planning? (Definition and advantages). 2022. URL: https://uk.indeed.com/career-advice/career-development/team-planning (дата звернення: 27.02.2023).

10. Planning. 2021. URL: https://theinvestorsbook.com/planning.html (дата звернення: 03.03.2023).

11. Програма Microsoft To Do. Microsoft: веб-сайт. URL: https://www.microsoft.com/uk-ua/microsoft-365/microsoft-to-do-list-app?rtc=1 (дата звернення: 12.03.2023).

12. Be focused and make things easier. Focus To-Do: веб-сайт. URL: https://www.focustodo.cn/ (дата звернення: 17.03.2023).

13. Use To-do List, Stay organized. Better App Tech: веб-сайт. URL: https://www.betterapptech.com/products/todolist/ (дата звернення: 27.03.2023).

14. Stay Organized Stay Creative. TickTick: веб-сайт. URL: https://ticktick.com/ (дата звернення: 31.03.2023).

15. Visual Studio 2022. Microsoft: веб-сайт. URL: https://visualstudio.microsoft.com/#vs-section (дата звернення: 05.04.2023).

16. maui-overview.png: зображення. URL: https://learn.microsoft.com/en-us/dotnet/maui/media/what-is-maui/maui-overview.png?view=net-maui-7.0.

17. architecture-diagram.png: зображення. URL: https://learn.microsoft.com/en-us/dotnet/maui/media/what-is-maui/architecture-diagram.png?view=net-maui-7.0.

18. What is .NET MAUI? Microsoft: веб-сайт. URL: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-7.0 (дата звернення: 12.04.2023).

19. A tour of the C# language. Microsoft: веб-сайт. URL: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ (дата звернення: 25.04.2023).

20. UML - Class Diagram. Tutorialspoint: веб-сайт. URL: https://www.tutorialspoint.com/uml/uml_class_diagram.htm (дата звернення: 03.05.2023).

21. About SQLite. SQLite: веб-сайт. URL: https://sqlite.org/about.html (дата звернення: 05.05.2023).

ДОДАТОК А

ЛІСТИНГ ІНТЕРФЕЙСУ ПРОГРАМИ

Сторінка із відображенням завдань на поточний день (TodayTasksPage.xaml):

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="DiplomaApp.Views.TodayTasksPage"
             xmlns:models="clr-namespace:DiplomaApp.Models"
             xmlns:views="clr-namespace:DiplomaApp.Views"
             x:DataType="{x:Type views:TodayTasksPage}">

    <Grid>
        <CollectionView
            VerticalOptions="Fill"
            ItemsSource="{Binding Tasks}"
            SelectionMode="Single"
            SelectionChanged="CollectionView_SelectionChanged">

            <CollectionView.EmptyView>
                <Label
                    VerticalTextAlignment="Center" HorizontalTextAlignment="Center"
                    FontSize="Large" FontAttributes="Bold"
                    Text="Немає завдань"
                    TextColor="{StaticResource Secondary}"/>
            </CollectionView.EmptyView>

            <CollectionView.ItemsLayout>
                <GridItemsLayout Orientation="Vertical" Span="1"/>
            </CollectionView.ItemsLayout>

            <CollectionView.ItemTemplate>
                <DataTemplate x:DataType="{x:Type models:TaskItem}">
                    <Border
                        Margin="5"
                        BackgroundColor="Transparent"
                        StrokeThickness="0">

                        <Border.StrokeShape>
                            <RoundRectangle CornerRadius="10"/>
                        </Border.StrokeShape>

                        <Grid ColumnDefinitions="20,*">
                            <BoxView
                                Grid.Column="0"
                                WidthRequest="20"/>

                            <VerticalStackLayout
                                Grid.Column="1"
                                Padding="10"
                                BackgroundColor="{AppThemeBinding
Light={StaticResource Secondary}, Dark={StaticResource White}}">

                                <HorizontalStackLayout
                                    Spacing="20">
                                    <Label
                                        FontSize="Medium" FontAttributes="Bold"
```

```
                                    Text="{Binding Date, StringFormat='{0:dd
MMMM}'}"
                                    TextColor="{StaticResource Black}"
                                    VerticalTextAlignment="Center"/>
                                <Label
                                    FontSize="Medium" FontAttributes="Bold"
                                    Text="{Binding Time,
StringFormat='{}{0:hh}:{0:mm}'}"
                                    IsVisible="{Binding TimeVisible}"
                                    TextColor="{StaticResource Black}"
                                    VerticalTextAlignment="Center"/>
                            </HorizontalStackLayout>

                            <Grid RowDefinitions="Auto,Auto"
ColumnDefinitions="2*,2*"
                                  RowSpacing="2">
                                <Label
                                    Grid.Row="0" Grid.Column="0"
                                    FontSize="Medium"
                                    Text="{Binding Title}"
                                    TextColor="{StaticResource Black}"
                                    VerticalTextAlignment="Center"/>
                                <Label
                                    Grid.Row="0" Grid.Column="1"
                                    Text="✔"
                                    VerticalTextAlignment="Center"
                                    HorizontalOptions="End"
                                    IsVisible="{Binding Done}"/>
                                <Label
                                    Grid.Row="1" Grid.Column="0"
                                    Grid.ColumnSpan="2"
                                    Margin="0,10,0,0"
                                    FontSize="Small"
                                    Text="{Binding Description}"
                                    TextColor="{StaticResource Black}"/>
                            </Grid>
                        </VerticalStackLayout>
                    </Grid>
                </Border>
            </DataTemplate>
        </CollectionView.ItemTemplate>
    </CollectionView>

    <Button
        Text="+"
        FontSize="40"
        WidthRequest="75" HeightRequest="75"
        CornerRadius="50"
        HorizontalOptions="End" VerticalOptions="End"
        Margin="0,0,15,15"
        Clicked="OnItemAdded"/>
</Grid>
</ContentPage>
```

Сторінка для внесення чи редагування інформації про завдання (TaskItemPage.xaml):

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="DiplomaApp.Views.TaskItemPage"
             xmlns:models="clr-namespace:DiplomaApp.Models"
             x:DataType="models:TaskItem"
             Title="{Binding Title}">
```

```
    <ContentPage.Resources>
        <Style TargetType="Label">
            <Setter Property="FontSize" Value="Medium"/>
            <Setter Property="FontAttributes" Value="Bold"/>
            <Setter Property="VerticalOptions" Value="Center"/>
        </Style>
        <Style TargetType="Entry">
            <Setter Property="FontSize" Value="Medium"/>
            <Setter Property="VerticalOptions" Value="Start"/>
            <Setter Property="VerticalTextAlignment" Value="End"/>
        </Style>
        <Style TargetType="DatePicker">
            <Setter Property="FontSize" Value="Medium"/>
            <Setter Property="VerticalOptions" Value="End"/>
        </Style>
        <Style TargetType="TimePicker">
            <Setter Property="FontSize" Value="Medium"/>
            <Setter Property="VerticalOptions" Value="End"/>
        </Style>
        <Style TargetType="Button">
            <Setter Property="FontSize" Value="Medium"/>
        </Style>
    </ContentPage.Resources>

    <ScrollView>
        <Grid RowDefinitions="Auto,Auto,Auto,Auto,Auto,Auto,Auto,Auto,Auto,Auto"
ColumnDefinitions="2*,2*"
              Padding="20">
            <Label
                Grid.Row="0" Grid.Column="0"
                Grid.ColumnSpan="2"
                Text="Назва завдання"/>
            <Entry
                Grid.Row="1" Grid.Column="0"
                Grid.ColumnSpan="2"
                Text="{Binding Title}"
                VerticalOptions="Start"/>
            <Label
                Grid.Row="2" Grid.Column="0"
                Grid.ColumnSpan="2"
                Margin="0,10,0,0"
                Text="Опис"/>
            <Entry
                Grid.Row="3" Grid.Column="0"
                Grid.ColumnSpan="2"
                Text="{Binding Description}"
                VerticalOptions="Start"/>
            <Label
                Grid.Row="4" Grid.Column="0"
                Margin="0,10,0,0"
                Text="Дата"/>
            <DatePicker
                Grid.Row="4" Grid.Column="1"
                Margin="0,10,0,0"
                Date="{Binding Date}"/>
            <Label
                Grid.Row="5" Grid.Column="0"
                Margin="0,10,0,0"
                Text="Час"/>
            <Switch
                Grid.Row="5" Grid.Column="0"
                HorizontalOptions="Center"
                Margin="0,10,0,0"
                IsToggled="{Binding TimeVisible}"
```

```
                Toggled="VisibleTime"
                x:Name="SwitchTime"/>
            <TimePicker
                Grid.Row="5" Grid.Column="1"
                Margin="0,10,0,0"
                Time="{Binding Time}"
                IsVisible="False"
                x:Name="TimePickerTime"
                Format="HH:MM"/>
            <Label
                Grid.Row="6" Grid.Column="0"
                Text="Виконано"/>
            <Switch
                Grid.Row="6" Grid.Column="1"
                Margin="0,10,0,0"
                IsToggled="{Binding Done}"/>
            <Button
                Grid.Row="7" Grid.Column="0"
                Grid.ColumnSpan="2"
                Margin="0,10,0,0"
                Text="Зберегти"
                Clicked="OnSaveClicked"/>
            <Button
                Grid.Row="8" Grid.Column="0"
                Grid.ColumnSpan="2"
                Margin="0,10,0,0"
                Text="Видалити"
                Clicked="OnDeleteClicked"/>
            <Button
                Grid.Row="9" Grid.Column="0"
                Grid.ColumnSpan="2"
                Margin="0,10,0,0"
                Text="Скасувати"
                Clicked="OnCancelClicked"/>
        </Grid>
    </ScrollView>
</ContentPage>
```

Сторінка із відображенням календаря завдань (CalendarTasksPage.xaml):

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="DiplomaApp.Views.CalendarTasksPage"
             xmlns:converter="clr-namespace:DiplomaApp.Converter"
             xmlns:models="clr-namespace:DiplomaApp.Models"
             xmlns:views="clr-namespace:DiplomaApp.Views"
             xmlns:mct="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
             xmlns:xc="clr-namespace:XCalendar.Maui.Views;assembly=XCalendar.Maui"
             xmlns:xcConverters="clr-
namespace:XCalendar.Maui.Converters;assembly=XCalendar.Maui"
             xmlns:System="clr-namespace:System;assembly=System.Runtime"
             x:Name="Calendar"
             x:DataType="{x:Type views:CalendarTasksPage}"
             Title="Календар завдань">

    <ContentPage.Resources>
        <!--  returns true if all bindings evaluate to true  -->
        <mct:VariableMultiValueConverter x:Key="AllTrueConverter"
ConditionType="All"/>
        <!--  inverts a binded boolean value  -->
        <mct:InvertedBoolConverter x:Key="InvertedBoolConverter"/>
        <!-- convert names days of week from English to Ukrainian -->
        <converter:WeekDaysConverter x:Key="DayConverter"/>
    </ContentPage.Resources>
```

```
    <ScrollView>

        <Grid RowDefinitions="Auto,*">

            <Border
                Grid.Row="0"
                HorizontalOptions="CenterAndExpand"
                Margin="10" Padding="10"
                BackgroundColor="{AppThemeBinding Light={StaticResource White},
Dark={StaticResource White}}">

                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="15"/>
                </Border.StrokeShape>

                <xc:CalendarView
                    BackwardsArrowCommand="{Binding NavigateCalendarCommand}"
                    ForwardsArrowCommand="{Binding NavigateCalendarCommand}"
                    Days="{Binding TaskCalendar.Days}"
                    DaysOfWeek="{Binding TaskCalendar.DayNamesOrder}"
                    NavigatedDate="{Binding TaskCalendar.NavigatedDate}"
                    BackgroundColor="{StaticResource White}">

                    <xc:CalendarView.DayNameTemplate>
                        <DataTemplate x:DataType="{x:Type System:DayOfWeek}">
                            <Label
                                Text="{Binding .,Mode=TwoWay,
Converter={StaticResource DayConverter}}"
                                HorizontalTextAlignment="Center"
                                HeightRequest="50"
                                TextColor="{AppThemeBinding Light={StaticResource
Black}, Dark={StaticResource Black}}"/>
                        </DataTemplate>
                    </xc:CalendarView.DayNameTemplate>

                    <xc:CalendarView.ForwardsArrowCommandParameter>
                        <x:Int32>1</x:Int32>
                    </xc:CalendarView.ForwardsArrowCommandParameter>

                    <xc:CalendarView.BackwardsArrowCommandParameter>
                        <x:Int32>-1</x:Int32>
                    </xc:CalendarView.BackwardsArrowCommandParameter>

                    <xc:CalendarView.NavigationViewTemplate>
                        <ControlTemplate>
                            <xc:NavigationView
                                ArrowColor="{StaticResource Black}"
                                BackgroundColor="Transparent"
                                BackwardsArrowCommand="{Binding
BackwardsArrowCommand, Source={RelativeSource TemplatedParent}}"
                                BackwardsArrowCommandParameter="{Binding
BackwardsArrowCommandParameter, Source={RelativeSource TemplatedParent}}"
                                ForwardsArrowCommand="{Binding ForwardsArrowCommand,
Source={RelativeSource TemplatedParent}}"
                                ForwardsArrowCommandParameter="{Binding
ForwardsArrowCommandParameter, Source={RelativeSource TemplatedParent}}"
                                HeightRequest="40"
                                Text="{Binding Text, Source={RelativeSource
TemplatedParent}}"
                                TextColor="{StaticResource Black}"/>
                        </ControlTemplate>
                    </xc:CalendarView.NavigationViewTemplate>

                    <xc:CalendarView.DayTemplate>
```

```
                        <DataTemplate x:DataType="{x:Type models:TaskDay}">
                            <Border
                                Margin="2.5"
                                BackgroundColor="Transparent"
                                HeightRequest="45" WidthRequest="45">

                                <Border.StrokeShape>
                                    <Ellipse/>
                                </Border.StrokeShape>

                                <xc:DayView
                                    CurrentMonthCommand="{Binding
BindingContext.ChangeDateSelectionCommand, Source={x:Reference Calendar}}"
                                    CurrentMonthCommandParameter="{Binding
DateTime}"
                                    DateTime="{Binding DateTime}"
                                    IsCurrentMonth="{Binding IsCurrentMonth}"
                                    IsInvalid="{Binding IsInvalid}"
                                    IsSelected="{Binding IsSelected}"
                                    IsToday="{Binding IsToday}"
                                    SelectedCommand="{Binding
BindingContext.ChangeDateSelectionCommand, Source={x:Reference Calendar}}"
                                    SelectedCommandParameter="{Binding DateTime}"
                                    CurrentMonthTextColor="{StaticResource Black}"
                                    TodayTextColor="{StaticResource Secondary}"
                                    FontAttributes="Bold"
                                    SelectedBackgroundColor="{StaticResource
Primary}"
                                    SelectedTextColor="{StaticResource White}"
                                    TodayCommand="{Binding
BindingContext.ChangeDateSelectionCommand, Source={x:Reference Calendar}}"
                                    TodayCommandParameter="{Binding DateTime}">

                                    <xc:DayView.ControlTemplate>
                                        <ControlTemplate>
                                            <!--  TemplatedParent refers to the view
that the ControlTemplate resides in  -->
                                            <Grid
                                                BindingContext="{Binding
BindingContext, Source={RelativeSource TemplatedParent}}"
                                                RowSpacing="2"
RowDefinitions="1.5*,*">

                                                <!--  ContentPresenter displays the
default content for the control  -->
                                                <ContentPresenter
                                                    Grid.Row="0"
                                                    Grid.RowSpan="2"
                                                    VerticalOptions="Center"/>

                                                <HorizontalStackLayout
                                                    Grid.Row="1"

BindableLayout.ItemsSource="{Binding Tasks}"
                                                    HorizontalOptions="Center"
                                                    Spacing="2.5">

                                                    <!--  task indicators to only be
visible when the DateTime is in the currently navigated month  -->

<HorizontalStackLayout.IsVisible>
                                                        <MultiBinding
Converter="{StaticResource AllTrueConverter}">
```

```
                                                        <Binding
Path="IsCurrentMonth"/>
                                                        <Binding
Converter="{StaticResource InvertedBoolConverter}" Path="IsInvalid"/>
                                                    </MultiBinding>

</HorizontalStackLayout.IsVisible>

                                                <BindableLayout.ItemTemplate>
                                                    <DataTemplate
x:DataType="{x:Type models:TaskItem}">
                                                        <BoxView
                                                            CornerRadius="100"
                                                            HeightRequest="7"
WidthRequest="7"

HorizontalOptions="CenterAndExpand" VerticalOptions="Center"

Color="{StaticResource Primary}"/>
                                                    </DataTemplate>
                                                </BindableLayout.ItemTemplate>
                                            </HorizontalStackLayout>
                                        </Grid>
                                    </ControlTemplate>
                                </xc:DayView.ControlTemplate>
                            </xc:DayView>
                        </Border>
                    </DataTemplate>
                </xc:CalendarView.DayTemplate>
            </xc:CalendarView>
        </Border>

        <CollectionView
            Grid.Row="1"
            ItemsSource="{Binding SelectedTasks}"
            Margin="{OnPlatform Android='0', WinUI='20'}"
            SelectionChanged="CollectionView_SelectionChanged"
            SelectionMode="Single">

            <CollectionView.EmptyView>
                <Label
                    FontAttributes="Bold"
                    FontSize="Large"
                    VerticalTextAlignment="Center"
HorizontalTextAlignment="Center"
                    Text="Нічого не заплановано"
                    TextColor="{StaticResource Secondary}"/>
            </CollectionView.EmptyView>

            <CollectionView.ItemsLayout>
                <GridItemsLayout Orientation="Vertical" Span="1"/>
            </CollectionView.ItemsLayout>

            <CollectionView.ItemTemplate>
                <DataTemplate x:DataType="{x:Type models:TaskItem}">
                    <Border
                        Margin="5"
                        BackgroundColor="Transparent"
                        StrokeThickness="0">

                        <Border.StrokeShape>
                            <RoundRectangle CornerRadius="10"/>
                        </Border.StrokeShape>

                        <Grid ColumnDefinitions="20,*">
```

```
                                <BoxView
                                    Grid.Column="0"
                                    WidthRequest="20"/>

                                <VerticalStackLayout
                                    Grid.Column="1"
                                    Padding="10"
                                    BackgroundColor="{AppThemeBinding
Light={StaticResource Secondary}, Dark={StaticResource White}}">

                                    <HorizontalStackLayout
                                        Spacing="20">
                                        <Label
                                            FontAttributes="Bold"
                                            FontSize="Medium"
                                            Text="{Binding Date, StringFormat='{0:dd
MMMM}'}"
                                            TextColor="{StaticResource Black}"
                                            VerticalTextAlignment="Center"/>
                                        <Label
                                            FontAttributes="Bold"
                                            FontSize="Medium"
                                            Text="{Binding Time,
StringFormat='{}{0:hh}:{0:mm}'}"
                                            IsVisible="{Binding TimeVisible}"
                                            TextColor="{StaticResource Black}"
                                            VerticalTextAlignment="Center"/>
                                    </HorizontalStackLayout>

                                    <Grid RowDefinitions="Auto,Auto"
ColumnDefinitions="2*,2*"
                                          RowSpacing="2">
                                        <Label
                                            Grid.Row="0" Grid.Column="0"
                                            FontSize="Medium"
                                            Text="{Binding Title}"
                                            TextColor="{StaticResource Black}"
                                            VerticalTextAlignment="Center"/>
                                        <Label
                                            Grid.Row="0" Grid.Column="1"
                                            Text="✔"
                                            VerticalTextAlignment="Center"
                                            HorizontalOptions="End"
                                            IsVisible="{Binding Done}"/>
                                        <Label
                                            Grid.Row="1" Grid.Column="0"
                                            Grid.ColumnSpan="2"
                                            Margin="0,10,0,0"
                                            FontSize="Small"
                                            Text="{Binding Description}"
                                            TextColor="{StaticResource Black}"/>
                                    </Grid>
                                </VerticalStackLayout>
                            </Grid>
                        </Border>
                    </DataTemplate>
                </CollectionView.ItemTemplate>
            </CollectionView>
        </Grid>
    </ScrollView>
</ContentPage>
```

Сторінка із статистики завдань (StatisticsTasksPage.xaml):

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="DiplomaApp.Views.StatisticsTasksPage"
             xmlns:views="clr-namespace:DiplomaApp.Views"
             xmlns:controls="clr-namespace:DiplomaApp.Controls"
             x:DataType="{x:Type views:StatisticsTasksPage}"
             Title="Статистика завдань">

    <ContentPage.Resources>
        <Style TargetType="Label">
            <Setter Property="FontSize" Value="Medium"/>
            <Setter Property="FontAttributes" Value="Bold"/>
            <Setter Property="VerticalOptions" Value="Center"/>
            <Setter Property="TextColor" Value="{AppThemeBinding
Light={StaticResource Primary}, Dark={StaticResource Primary}}"/>
        </Style>
    </ContentPage.Resources>

    <ScrollView>
        <Grid RowDefinitions="Auto,Auto,Auto,Auto">
            <Label
                Grid.Row="0"
                x:Name="Today"
                Padding="10" Margin="10,10,10,0"
                BackgroundColor="{AppThemeBinding Light={StaticResource Primary},
Dark={StaticResource Primary}}"
                TextColor="{AppThemeBinding Light={StaticResource White},
Dark={StaticResource White}}"/>
            <Border
                Grid.Row="1"
                Margin="10" Padding="10"
                BackgroundColor="{AppThemeBinding Light={StaticResource White},
Dark={StaticResource White}}">

                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="15"/>
                </Border.StrokeShape>

                <Grid RowDefinitions="Auto,Auto">

                    <VerticalStackLayout Grid.Row="0">
                        <Grid RowDefinitions="Auto,Auto,Auto"
ColumnDefinitions="2*,2*"
                              RowSpacing="7"
                              Padding="20">
                            <Label
                                Grid.Row="0" Grid.Column="0"
                                Text="Виконано:"/>
                            <Label
                                Grid.Row="0" Grid.Column="1"
                                HorizontalOptions="End"
                                Text="{Binding TasksDoneToday.Count}"/>
                            <Label
                                Grid.Row="1" Grid.Column="0"
                                Text="Не виконано:"/>
                            <Label
                                Grid.Row="1" Grid.Column="1"
                                HorizontalOptions="End"
                                Text="{Binding TasksNotDoneToday.Count}"/>
                            <Label
                                Grid.Row="2" Grid.Column="0"
                                Text="Всього:"/>
```

```
                        <Label
                            Grid.Row="2" Grid.Column="1"
                            HorizontalOptions="End"
                            Text="{Binding TasksToday.Count}"/>
                    </Grid>
                </VerticalStackLayout>

                <VerticalStackLayout Grid.Row="1">
                    <Grid RowDefinitions="Auto,Auto">
                        <controls:RadialBarChart
                            BarSpacing="16"
                            BarThickness="10"
                            HeightRequest="150"
                            FontSize="12"
                            MaxValue="100"
                            ShowLabels="False"
                            BarBackgroundColor="{StaticResource Gray100}"
                            Entries="{Binding ChartEntriesToday}"/>

                        <HorizontalStackLayout
                            Grid.Row="1"
                            Spacing="15"
                            HorizontalOptions="Center"
                            BindableLayout.ItemsSource="{Binding
ChartEntriesToday}">
                            <BindableLayout.ItemTemplate>
                                <DataTemplate x:DataType="{x:Type
controls:ChartEntry}">
                                    <Grid RowDefinitions="Auto,Auto"
                                          RowSpacing="5">
                                        <Rectangle
                                            WidthRequest="10" HeightRequest="10"
                                            BackgroundColor="{Binding Color}"/>

                                        <HorizontalStackLayout Grid.Row="1">
                                            <Label
                                                FontSize="12"
                                                Text="{Binding Text}"
                                                HorizontalOptions="Center"

HorizontalTextAlignment="Center"/>
                                            <Label
                                                FontSize="12"
                                                Text=": "
                                                HorizontalOptions="Center"

HorizontalTextAlignment="Center"/>
                                            <Label
                                                FontSize="12"
                                                Text="{Binding Value}"
                                                HorizontalOptions="Center"

HorizontalTextAlignment="Center"/>
                                            <Label
                                                FontSize="12"
                                                Text="%"

                                                HorizontalOptions="Center"

HorizontalTextAlignment="Center"/>
                                        </HorizontalStackLayout>
                                    </Grid>
                                </DataTemplate>
                            </BindableLayout.ItemTemplate>
                        </HorizontalStackLayout>
```

```
                        </Grid>
                    </VerticalStackLayout>
                </Grid>
            </Border>

            <Label
                Grid.Row="2"
                Text="За весь час"
                Padding="10" Margin="10,10,10,0"
                BackgroundColor="{AppThemeBinding Light={StaticResource Primary},
Dark={StaticResource Primary}}"
                TextColor="{AppThemeBinding Light={StaticResource White},
Dark={StaticResource White}}"/>
            <Border
                Grid.Row="3"
                Margin="10" Padding="10"
                BackgroundColor="{AppThemeBinding Light={StaticResource White},
Dark={StaticResource White}}">

                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="15"/>
                </Border.StrokeShape>

                <Grid RowDefinitions="Auto,Auto">

                    <VerticalStackLayout Grid.Row="0">
                        <Grid RowDefinitions="Auto,Auto,Auto"
ColumnDefinitions="2*,2*"
                              RowSpacing="7"
                              Padding="20">
                            <Label
                                Grid.Row="0" Grid.Column="0"
                                Text="Виконано:"/>
                            <Label
                                Grid.Row="0" Grid.Column="1"
                                HorizontalOptions="End"
                                Text="{Binding TasksDone.Count}"/>
                            <Label
                                Grid.Row="1" Grid.Column="0"
                                Text="Не виконано:"/>
                            <Label
                                Grid.Row="1" Grid.Column="1"
                                HorizontalOptions="End"
                                Text="{Binding TasksNotDone.Count}"/>
                            <Label
                                Grid.Row="2" Grid.Column="0"
                                Text="Всього:"/>
                            <Label
                                Grid.Row="2" Grid.Column="1"
                                HorizontalOptions="End"
                                Text="{Binding Tasks.Count}"/>
                        </Grid>
                    </VerticalStackLayout>

                    <VerticalStackLayout Grid.Row="1">
                        <Grid RowDefinitions="Auto,Auto">
                            <controls:RadialBarChart
                                BarSpacing="16"
                                BarThickness="10"
                                HeightRequest="150"
                                FontSize="12"
                                MaxValue="100"
                                ShowLabels="False"
                                BarBackgroundColor="{StaticResource Gray100}"
                                Entries="{Binding ChartEntries}"/>
```

```
                            <HorizontalStackLayout
                                Grid.Row="1"
                                Spacing="15"
                                HorizontalOptions="Center"
                                BindableLayout.ItemsSource="{Binding ChartEntries}">
                                <BindableLayout.ItemTemplate>
                                    <DataTemplate x:DataType="{x:Type
controls:ChartEntry}">
                                        <Grid RowDefinitions="Auto,Auto"
                                              RowSpacing="5">
                                            <Rectangle
                                                WidthRequest="10" HeightRequest="10"
                                                BackgroundColor="{Binding Color}"/>

                                            <HorizontalStackLayout Grid.Row="1">
                                                <Label
                                                    FontSize="12"
                                                    Text="{Binding Text}"
                                                    HorizontalOptions="Center"

HorizontalTextAlignment="Center"/>
                                                <Label
                                                    FontSize="12"
                                                    Text=": "
                                                    HorizontalOptions="Center"

HorizontalTextAlignment="Center"/>
                                                <Label
                                                    FontSize="12"
                                                    Text="{Binding Value}"
                                                    HorizontalOptions="Center"

HorizontalTextAlignment="Center"/>
                                                <Label
                                                    FontSize="12"
                                                    Text="%"
                                                    HorizontalOptions="Center"

HorizontalTextAlignment="Center"/>
                                            </HorizontalStackLayout>
                                        </Grid>
                                    </DataTemplate>
                                </BindableLayout.ItemTemplate>
                            </HorizontalStackLayout>
                        </Grid>
                    </VerticalStackLayout>
                </Grid>
            </Border>
        </Grid>
    </ScrollView>
</ContentPage>
```

Сторінка із налаштуваннями для Android застосунку (SettingsPage.xaml):

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="DiplomaApp.Views.SettingsPage"
             xmlns:models="clr-namespace:DiplomaApp.Models"
             x:DataType="models:SettingsItem"
             Title="Налаштування">

    <ContentPage.Resources>
        <Style TargetType="Label">
```

```
            <Setter Property="FontSize" Value="Medium"/>
            <Setter Property="FontAttributes" Value="Bold"/>
            <Setter Property="VerticalOptions" Value="Center"/>
            <Setter Property="TextColor" Value="{AppThemeBinding
Light={StaticResource Primary}, Dark={StaticResource White}}"/>
        </Style>
    </ContentPage.Resources>

    <ScrollView>
        <Grid RowDefinitions="Auto,Auto" ColumnDefinitions="2*,2*"
              RowSpacing="25"
              Padding="20">
            <Label
                Grid.Row="0" Grid.Column="0"
                Text="Темна тема"/>
            <Switch
                Grid.Row="0" Grid.Column="1"
                HorizontalOptions="End"
                IsToggled="{Binding DarkTheme}"
                Toggled="SwitchTheme_Toggled"
                x:Name="SwitchTheme"/>
            <Label
                Grid.Row="1" Grid.Column="0"
                Text="Сповіщення"/>
            <Switch
                Grid.Row="1" Grid.Column="1"
                HorizontalOptions="End"
                IsToggled="{Binding Notification}"
                Toggled="SwitchNotification_Toggled"
                x:Name="SwitchNotification"/>
        </Grid>
    </ScrollView>
</ContentPage>
```

Сторінка із налаштуваннями для Windows застосунку (SettingsPageWindows.xaml):

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="DiplomaApp.Views.SettingsPageWindows"
             xmlns:models="clr-namespace:DiplomaApp.Models"
             x:DataType="models:SettingsItem"
             Title="Налаштування">

    <ContentPage.Resources>
        <Style TargetType="Label">
            <Setter Property="FontSize" Value="Medium"/>
            <Setter Property="FontAttributes" Value="Bold"/>
            <Setter Property="VerticalOptions" Value="Center"/>
            <Setter Property="TextColor" Value="{AppThemeBinding
Light={StaticResource Primary}, Dark={StaticResource White}}"/>
        </Style>
    </ContentPage.Resources>

    <ScrollView>
        <Grid RowDefinitions="Auto" ColumnDefinitions="2*,2*"
              RowSpacing="25"
              Padding="20">
            <Label
                Grid.Row="0" Grid.Column="0"
                Text="Темна тема"/>
            <Switch
                Grid.Row="0" Grid.Column="1"
```

```
                HorizontalOptions="End"
                IsToggled="{Binding DarkTheme}"
                Toggled="SwitchTheme_Toggled"
                x:Name="SwitchTheme"/>
        </Grid>
    </ScrollView>
</ContentPage>
```

Сторінка із інформацією про додаток (InfoPage.xaml):

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="DiplomaApp.Views.InfoPage"
             Title="Про додаток">

    <ContentPage.Resources>
        <Style TargetType="Label">
            <Setter Property="FontSize" Value="Medium"/>
            <Setter Property="TextColor" Value="{AppThemeBinding
Light={StaticResource Primary}, Dark={StaticResource White}}"/>
        </Style>
    </ContentPage.Resources>

    <VerticalStackLayout Padding="20">
        <Label
            Text="Ціль створення додатку:"
            FontAttributes="Bold"/>
        <Label
            Margin="0,5,0,0"
            Text="Бакалаврська дипломна робота"/>
        <Label
            Margin="0,25,0,0"
            Text="Група:"
            FontAttributes="Bold"/>
        <Label
            Margin="0,5,0,0"
            Text="К19-2"/>
        <Label
            Margin="0,25,0,0"
            Text="Виконавець:"
            FontAttributes="Bold"/>
        <Label
            Margin="0,5,0,0"
            Text="Лісова Ірина Олегівна"/>
        <Label
            Margin="0,25,0,0"
            Text="Рік:"
            FontAttributes="Bold"/>
        <Label
            Margin="0,5,0,0"
            Text="2023"/>
    </VerticalStackLayout>
</ContentPage>
```

Сторінка із головним меню додатку та навігацією (AppShell.xaml):

```
<?xml version="1.0" encoding="UTF-8" ?>
<Shell xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       x:Class="DiplomaApp.AppShell"
       xmlns:sys="clr-namespace:System;assembly=netstandard"
       xmlns:views="clr-namespace:DiplomaApp.Views"
       FlyoutBackgroundColor="{AppThemeBinding Light={StaticResource Primary},
Dark={StaticResource Primary}}"
```

```
       FlyoutVerticalScrollMode="Auto">

    <Shell.ItemTemplate>
        <DataTemplate>
            <Grid ColumnDefinitions="25,*" ColumnSpacing="25" Padding="15">
                <Image Source="{Binding Icon}"/>
                <Label Grid.Column="1"
                       Text="{Binding Title}"
                       TextColor="{AppThemeBinding Light={StaticResource White},
Dark={StaticResource White}}"
                       VerticalTextAlignment="Center"
                       FontSize="Medium"
                       FontAttributes="Bold"/>
            </Grid>
        </DataTemplate>
    </Shell.ItemTemplate>

    <Shell.FlyoutHeader>
        <Grid ColumnDefinitions="Auto,Auto"
              HeightRequest="150"
              BackgroundColor="{AppThemeBinding Light={StaticResource Primary},
Dark={StaticResource Primary}}"
              Margin="0,10,0,25">
            <Image
                Grid.Column="0"
                Source="{OnPlatform Android=header.svg, WinUI=header.png}"
                WidthRequest="100" HeightRequest="100"
                Margin="5,0,0,0"/>
            <Label
                Grid.Column="1"
                Text="Планер"
                VerticalOptions="Center"
                FontSize="35" FontAttributes="Bold"
                TextColor="White"/>
        </Grid>
    </Shell.FlyoutHeader>

    <FlyoutItem Title="Завдання" Icon="tasks.png">
        <ShellContent Title="Сьогодні" Icon="calendar_today.png"
                      ContentTemplate="{DataTemplate views:TodayTasksPage}"/>
        <ShellContent Title="Календар" Icon="calendar_month.png"
                      ContentTemplate="{DataTemplate views:CalendarTasksPage}"/>
        <ShellContent Title="Статистика" Icon="statistics.png"
                      ContentTemplate="{DataTemplate views:StatisticsTasksPage}"/>
    </FlyoutItem>

    <FlyoutItem Title="Події" Icon="events.png">
        <ShellContent Title="Сьогодні" Icon="calendar_today.png"
                      ContentTemplate="{DataTemplate views:TodayEventsPage}"/>
        <ShellContent Title="Календар" Icon="calendar_month.png"
                      ContentTemplate="{DataTemplate views:CalendarEventsPage}"/>
        <ShellContent Title="Статистика" Icon="statistics.png"
                      ContentTemplate="{DataTemplate views:StatisticsEventsPage}"/>
    </FlyoutItem>

    <FlyoutItem Title="Налаштування" Icon="settings.png">
        <ShellContent ContentTemplate="{OnPlatform Android={DataTemplate
views:SettingsPage}, WinUI={DataTemplate views:SettingsPageWindows}}"/>
    </FlyoutItem>

    <FlyoutItem Title="Про додаток" Icon="info.png">
        <ShellContent ContentTemplate="{DataTemplate views:InfoPage}"/>
    </FlyoutItem>

    <Shell.FlyoutFooter>
```

```
        <StackLayout Padding="0,0,0,20">
            <Label
                Text="Планер"
                TextColor="{AppThemeBinding Light={StaticResource White},
Dark={StaticResource White}}"
                FontSize="Subtitle" FontAttributes="Bold"
                HorizontalOptions="Center" />
            <Label
                Text="{Binding Source={x:Static sys:DateTime.Now},
StringFormat='{0:MMMM yyyy}'}"
                TextColor="{AppThemeBinding Light={StaticResource White},
Dark={StaticResource White}}"
                FontSize="Subtitle" FontAttributes="Bold"
                HorizontalOptions="Center" />
        </StackLayout>
    </Shell.FlyoutFooter>
</Shell>
```

ДОДАТОК Б

ЛІСТИНГ ФУНКЦІОНАЛЬНОЇ ЧАСТИНИ ПРОГРАМИ

Сторінка із відображенням завдань на поточний день (TodayTasksPage.cs):

```
using System.Collections.ObjectModel;
using DiplomaApp.Data;
using DiplomaApp.Models;

namespace DiplomaApp.Views;

public partial class TodayTasksPage : ContentPage
{
    #region Fields
    readonly TaskItemDatabase database;

      public ObservableCollection<TaskItem> Tasks { get; set; } = new();

    public DateTime DateToday = DateTime.Today.Date;
    #endregion

    #region Constructor
    public TodayTasksPage(TaskItemDatabase taskItemDatabase)
      {
            InitializeComponent();

            Title = $"Завдання - {DateTime.Today.ToShortDateString()}";

            database = taskItemDatabase;

        BindingContext = this;
      }
    #endregion

    #region Methods
    protected override async void OnNavigatedTo(NavigatedToEventArgs args)
      {
            base.OnNavigatedTo(args);

            // get all tasks for today from database
        var tasksToday = await database.GetTasksDateTodayAsync(DateToday);
            MainThread.BeginInvokeOnMainThread(() =>
            {
                  Tasks.Clear();
                  foreach (var taskToday in tasksToday)
                        Tasks.Add(taskToday);
            });
    }

      async void OnItemAdded(object sender, EventArgs e)
      {
            // open page where user can enter the information about new task
            await Shell.Current.GoToAsync(nameof(TaskItemPage), true, new
Dictionary<string, object>
            {
                  ["Item"] = new TaskItem()
            });
      }
```

```
      private async void CollectionView_SelectionChanged(object sender,
SelectionChangedEventArgs e)
      {
            if (e.CurrentSelection.FirstOrDefault() is not TaskItem taskItem)
                  return;

            // open page where user can edit the information about selected task
            await Shell.Current.GoToAsync(nameof(TaskItemPage), true, new
Dictionary<string, object>
            {
                  ["Item"] = taskItem
            });
      }
    #endregion
}
```

Сторінка для внесення чи редагування інформації про завдання (TaskItemPage.cs):

```
using DiplomaApp.Data;
using DiplomaApp.Models;

namespace DiplomaApp.Views;

[QueryProperty("Item", "Item")]
public partial class TaskItemPage : ContentPage
{
    #region Fields
    public TaskItem Item
      {
            get => BindingContext as TaskItem;
            set => BindingContext = value;
      }

    readonly TaskItemDatabase database;
    #endregion

    #region Constructor
    public TaskItemPage(TaskItemDatabase taskItemDatabase)
      {
            InitializeComponent();

        database = taskItemDatabase;
    }
    #endregion

    #region Methods
      // save the task (occurs when user clicks on button)
    async void OnSaveClicked(object sender, EventArgs e)
      {
            // display alert if title is not enter
            if (string.IsNullOrWhiteSpace(Item.Title))
            {
                  await DisplayAlert("Необхідна назва завдання", "Будь-ласка
введіть назву завдання.", "ОК");
                  return;
            }

            await database.SaveTaskAsync(Item);
            await Shell.Current.GoToAsync("..");
      }

      // delete the task (occurs when the user clicks on button)
      async void OnDeleteClicked(object sender, EventArgs e)
```

```
        {
                if (Item.Id == 0)
                        return;
                await database.DeleteTaskAsync(Item);
                await Shell.Current.GoToAsync("..");
        }

        // back to to the previous page (occurs when the user clicks on button)
        async void OnCancelClicked(object sender, EventArgs e) =>
                await Shell.Current.GoToAsync("..");

        // make time visible or not
        public void VisibleTime(object sender, ToggledEventArgs e)
        {
                if (SwitchTime.IsToggled)
                        TimePickerTime.IsVisible = true;
                else
                        TimePickerTime.IsVisible = false;
        }
    #endregion
}
```

Сторінка із відображенням календаря завдань (CalendarTasksPage.cs):

```
using DiplomaApp.Data;
using DiplomaApp.Models;
using System.Collections.Specialized;
using System.Windows.Input;
using XCalendar.Core.Enums;
using XCalendar.Core.Models;
using XCalendar.Core.Extensions;

namespace DiplomaApp.Views;

public partial class CalendarTasksPage : ContentPage
{
    #region Fields
    readonly TaskItemDatabase database;

    public Calendar<TaskDay> TaskCalendar { get; set; } = new Calendar<TaskDay>()
    {
        SelectedDates = new ObservableRangeCollection<DateTime>(),
        SelectionAction = SelectionAction.Replace,
        SelectionType = SelectionType.Single
    };

    public ObservableRangeCollection<TaskItem> Tasks { get; } = new
ObservableRangeCollection<TaskItem>();

    public ObservableRangeCollection<TaskItem> SelectedTasks { get; } = new
ObservableRangeCollection<TaskItem>();

    public DateTime NavigatedDate = DateTime.Today;
    #endregion

    #region Commands
    public ICommand NavigateCalendarCommand { get; set; }
    public ICommand ChangeDateSelectionCommand { get; set; }
    #endregion

    #region Constructor
    public CalendarTasksPage(TaskItemDatabase taskItemDatabase)
    {
        InitializeComponent();
```

```
        database = taskItemDatabase;

        NavigateCalendarCommand = new Command<int>(NavigateCalendar);
        ChangeDateSelectionCommand = new Command<DateTime>(ChangeDateSelection);

        Tasks.CollectionChanged += new
NotifyCollectionChangedEventHandler(UpdateDays);
        Tasks.CollectionChanged += new
NotifyCollectionChangedEventHandler(SelectedDates_CollectionChanged);

        if (TaskCalendar.SelectedDates.Count == 0)
            foreach (var Day in TaskCalendar.Days)
                if (Day.DateTime == DateTime.Today)
                    TaskCalendar.SelectedDates.Add(NavigatedDate);

        TaskCalendar.SelectedDates.CollectionChanged +=
SelectedDates_CollectionChanged;
        TaskCalendar.DaysUpdated += TaskCalendar_DaysUpdated;

        BindingContext = this;
    }
    #endregion

    #region Methods
    private void UpdateDays(object sender, EventArgs e) =>
        TaskCalendar.UpdateDays(NavigatedDate);

    protected override async void OnNavigatedTo(NavigatedToEventArgs args)
    {
        base.OnNavigatedTo(args);

        // get all tasks from database
        var tasks = await database.GetTasksAsync();
        MainThread.BeginInvokeOnMainThread(() =>
        {
            Tasks.Clear();
            foreach (var taskItem in tasks)
                Tasks.Add(taskItem);
        });
    }

    private void TaskCalendar_DaysUpdated(object sender, EventArgs e)
    {
        foreach (var Day in TaskCalendar.Days)
            Day.Tasks.ReplaceRange(Tasks.Where(x => x.Date.Date ==
Day.DateTime.Date));
    }

    private void SelectedDates_CollectionChanged(object sender,
NotifyCollectionChangedEventArgs e) =>
        SelectedTasks.ReplaceRange(Tasks.Where(x => TaskCalendar.SelectedDates.Any(y
=> x.Date.Date == y.Date)));

    public void NavigateCalendar(int Amount)
    {
        if (TaskCalendar.NavigatedDate.TryAddMonths(Amount, out DateTime
TargetDate))
            TaskCalendar.Navigate(TargetDate - TaskCalendar.NavigatedDate);
        else
            TaskCalendar.Navigate(Amount > 0 ? TimeSpan.MaxValue :
TimeSpan.MinValue);
    }

    public void ChangeDateSelection(DateTime DateTime) =>
        TaskCalendar?.ChangeDateSelection(DateTime);
```

```
    private async void CollectionView_SelectionChanged(object sender,
SelectionChangedEventArgs e)
    {
        if (e.CurrentSelection.FirstOrDefault() is not TaskItem taskItem)
            return;

        await Shell.Current.GoToAsync(nameof(TaskItemPage), true, new
Dictionary<string, object>
        {
            ["Item"] = taskItem
        });
    }
    #endregion
}
```

Сторінка із статистики завдань (StatisticsTasksPage.cs):

```
using System.Collections.ObjectModel;
using DiplomaApp.Data;
using DiplomaApp.Models;
using DiplomaApp.Controls;

namespace DiplomaApp.Views;

public partial class StatisticsTasksPage : ContentPage
{
    #region Fields
    readonly TaskItemDatabase database;

    public ObservableCollection<TaskItem> TasksDone { get; set; } = new();
    public ObservableCollection<TaskItem> TasksNotDone { get; set; } = new();
    public ObservableCollection<TaskItem> Tasks { get; set; } = new();

    public ObservableCollection<TaskItem> TasksDoneToday { get; set; } = new();
    public ObservableCollection<TaskItem> TasksNotDoneToday { get; set; } = new();
    public ObservableCollection<TaskItem> TasksToday { get; set; } = new();

    public List<Color> Colors { get; set; } = new();

    public List<string> Strings { get; set; } = new();

    public ChartEntry[] ChartEntries { get; set; } = new ChartEntry[2];

    public ChartEntry[] ChartEntriesToday { get; set; } = new ChartEntry[2];

    public DateTime DateToday = DateTime.Today.Date;
    #endregion

    #region Constructor
    public StatisticsTasksPage(TaskItemDatabase taskItemDatabase)
    {
        InitializeComponent();

        database = taskItemDatabase;

        Today.Text = $"Сьогодні - {DateTime.Today.ToShortDateString()}";

        BindingContext = this;
    }
    #endregion

    #region Method
    protected override async void OnNavigatedTo(NavigatedToEventArgs args)
    {
```

```
base.OnNavigatedTo(args);

// get all tasks from database if task is done
var tasksDone = await database.GetTasksDoneAsync();
MainThread.BeginInvokeOnMainThread(() =>
{
    TasksDone.Clear();
    foreach (var taskDone in tasksDone)
        TasksDone.Add(taskDone);
});

// get all tasks from databse if task is not done
var tasksNotDone = await database.GetTasksNotDoneAsync();
MainThread.BeginInvokeOnMainThread(() =>
{
    TasksNotDone.Clear();
    foreach (var taskNotDone in tasksNotDone)
        TasksNotDone.Add(taskNotDone);
});

// get all tasks from database
var tasks = await database.GetTasksAsync();
MainThread.BeginInvokeOnMainThread(() =>
{
    Tasks.Clear();
    foreach (var taskItem in tasks)
        Tasks.Add(taskItem);
});

// get all tasks where tasks is not done for today
var tasksNotDoneToday = await database.GetTasksDateTodayAsync(DateToday);
MainThread.BeginInvokeOnMainThread(() =>
{
    TasksNotDoneToday.Clear();
    foreach (var taskNotDoneToday in tasksNotDoneToday)
        if (taskNotDoneToday.Done != true)
            TasksNotDoneToday.Add(taskNotDoneToday);
});

// get all tasks where tasks is done for today
var tasksDoneToday = await database.GetTasksDateTodayAsync(DateToday);
MainThread.BeginInvokeOnMainThread(() =>
{
    TasksDoneToday.Clear();
    foreach (var taskDoneToday in tasksDoneToday)
        if (taskDoneToday.Done == true)
            TasksDoneToday.Add(taskDoneToday);
});

// get all tasks for today
var tasksToday = await database.GetTasksDateTodayAsync(DateToday);
MainThread.BeginInvokeOnMainThread(() =>
{
    TasksToday.Clear();
    foreach (var taskToday in tasksToday)
        TasksToday.Add(taskToday);
});

double tasksDoneAmount;
double tasksNotDoneAmount;

double tasksDoneAmountToday;
double tasksNotDoneAmountToday;

if (Tasks.Count == 0)
```

```
        {
            tasksDoneAmount = 0;
            tasksNotDoneAmount = 0;
        }
        else
        {
            tasksDoneAmount = Math.Round((double)TasksDone.Count / Tasks.Count *
100, 2);
            tasksNotDoneAmount = Math.Round((double)TasksNotDone.Count / Tasks.Count
* 100, 2);
        }

        if (TasksToday.Count == 0)
        {
            tasksDoneAmountToday = 0;
            tasksNotDoneAmountToday = 0;
        }
        else
        {
            tasksDoneAmountToday = Math.Round((double)TasksDoneToday.Count /
TasksToday.Count * 100, 2);
            tasksNotDoneAmountToday = Math.Round((double)TasksNotDoneToday.Count /
TasksToday.Count * 100, 2);
        }

        Colors.Add(Color.FromArgb("#5f735e"));
        Colors.Add(Color.FromArgb("#D5E1D6"));

        Strings.Add("Виконано");
        Strings.Add("Не виконано");

        ChartEntries = new ChartEntry[]
        {
            new ChartEntry
            {
                Value = tasksDoneAmount,
                Color = Colors[0],
                Text = Strings[0]
            },
            new ChartEntry
            {
                Value = tasksNotDoneAmount,
                Color = Colors[1],
                Text = Strings[1]
            }
        };

        ChartEntriesToday = new ChartEntry[]
        {
            new ChartEntry
            {
                Value = tasksDoneAmountToday,
                Color = Colors[0],
                Text = Strings[0]
            },
            new ChartEntry
            {
                Value = tasksNotDoneAmountToday,
                Color = Colors[1],
                Text = Strings[1]
            }
        };
    }
    #endregion
}
```

Сторінка із налаштуваннями для Android застосунку (SettingsPage.cs):

```
using DiplomaApp.Data;
using DiplomaApp.Models;
using Plugin.LocalNotification;

namespace DiplomaApp.Views;

[QueryProperty("Item", "Item")]
public partial class SettingsPage : ContentPage
{
    #region Fields
    readonly SettingsDatabase database;

    public SettingsItem Item
    {
        get => BindingContext as SettingsItem;
        set => BindingContext = value;
    }

    public DateTime DateToday = DateTime.Today.Date;

    private readonly INotificationService _notificationService;

    NotificationRequest request = new();
    #endregion

    #region Constructor
    public SettingsPage(SettingsDatabase settingsDatabase, INotificationService
notificationService)
    {
        InitializeComponent();

        database = settingsDatabase;

        _notificationService = notificationService;
    }
    #endregion

    #region Methods
    // switch theme
    private async void SwitchTheme_Toggled(object sender, ToggledEventArgs e)
    {
        if (SwitchTheme.IsToggled)
        {
            Application.Current.UserAppTheme = AppTheme.Dark;
            Item.DarkTheme = true;
            await database.SaveItemAsync(Item);
        }
        else
        {
            Application.Current.UserAppTheme = AppTheme.Light;
            Item.DarkTheme = false;
            await database.SaveItemAsync(Item);
        }
    }

    // switch notification
    private async void SwitchNotification_Toggled(object sender, ToggledEventArgs e)
    {
        // create notification
        request = new()
        {
            Title = "Не забувайте про свої плани",
```

```
            Description = "Перегляньте можливо вам необхідно виконати завдання чи
відвідати подію.",
            Schedule = new NotificationRequestSchedule
            {
                NotifyTime = DateToday.AddHours(8),
                RepeatType = NotificationRepeat.TimeInterval,
                NotifyRepeatInterval = TimeSpan.FromDays(1)
            },
            Android =
            {
                IconSmallName =
                {
                    ResourceName = "header"
                },
                Color =
                {
                    ResourceName = "colorPrimary"
                }
            }
        };

        if (SwitchNotification.IsToggled)
        {
            if (await _notificationService.AreNotificationsEnabled() == false)
            {
                SwitchNotification.IsToggled = false;
                await DisplayAlert("Немає дозволу надсилати сповіщення", "Надайте
дозвіл на надсилання сповіщень в налаштуваннях.", "ОК");
                return;
            }
            else
            {
                if (DateTime.Now > DateToday.AddHours(8))
                    request.Schedule.NotifyTime = DateToday.AddDays(1).AddHours(8);
                _ = LocalNotificationCenter.Current.Show(request);
                Item.Notification = true;
                await database.SaveItemAsync(Item);
            }
        }
        else
        {
            _ = LocalNotificationCenter.Current.ClearAll();
            _ = LocalNotificationCenter.Current.CancelAll();
            Item.Notification = false;
            await database.SaveItemAsync(Item);
        }
    }

    protected override async void OnNavigatedTo(NavigatedToEventArgs args)
    {
        base.OnNavigatedTo(args);

        // get all items from database
        var settingsItems = await database.GetItemsAsync();
        MainThread.BeginInvokeOnMainThread(() =>
        {
            foreach (var settingsItem in settingsItems)
                Item = settingsItem;
        });
    }
    #endregion
}
```

Сторінка із налаштуваннями для Windows застосунку (SettingsPageWindows.cs):

```
using DiplomaApp.Data;
using DiplomaApp.Models;

namespace DiplomaApp.Views;

public partial class SettingsPageWindows : ContentPage
{
    #region Fields
    readonly SettingsDatabase database;

    public SettingsItem Item
    {
        get => BindingContext as SettingsItem;
        set => BindingContext = value;
    }
    #endregion

    #region Constructor
    public SettingsPageWindows(SettingsDatabase settingsDatabase)
       {
             InitializeComponent();

        database = settingsDatabase;
    }
    #endregion

    #region Methods
    // switch theme
    private async void SwitchTheme_Toggled(object sender, ToggledEventArgs e)
    {
        if (SwitchTheme.IsToggled)
        {
            Application.Current.UserAppTheme = AppTheme.Dark;
            Item.DarkTheme = true;
            await database.SaveItemAsync(Item);
        }
        else
        {
            Application.Current.UserAppTheme = AppTheme.Light;
            Item.DarkTheme = false;
            await database.SaveItemAsync(Item);
        }
    }

    protected override async void OnNavigatedTo(NavigatedToEventArgs args)
    {
        base.OnNavigatedTo(args);

        // get all items from database
        var settingsItems = await database.GetItemsAsync();
        MainThread.BeginInvokeOnMainThread(() =>
        {
            foreach (var settingsItem in settingsItems)
                Item = settingsItem;
        });
    }
    #endregion
}
```

Сторінка із інформацією про додаток (InfoPage.cs):

```
namespace DiplomaApp.Views;

public partial class InfoPage : ContentPage
{
    #region Constructor
    public InfoPage() =>
             InitializeComponent();
    #endregion
}
```

Файл зі структурою для побудови діаграми (CharEntry.cs):

```
namespace DiplomaApp.Controls
{
    // for charts in StatisticsTasksPage and StatisticsEventsPage
    public struct ChartEntry
    {
        public double Value { get; set; }
        public string Text { get; set; }
        public Color Color { get; set; }
    }
}
```

Файл, де створюються зовнішні параметри для побудови діаграми (RadialBarChart.cs):

```
namespace DiplomaApp.Controls
{
    public class RadialBarChart : GraphicsView
    {
        #region Constructor
        public RadialBarChart() => Drawable = new RadialBarChartDrawable(this);
        #endregion

        #region Entries
        public static readonly BindableProperty EntriesProperty =
            BindableProperty.Create(
                nameof(Entries),
                typeof(IEnumerable<ChartEntry>),
                typeof(RadialBarChart),
                defaultValue: null,
                propertyChanged: Invalidate);

        public IEnumerable<ChartEntry> Entries
        {
            get { return (IEnumerable<ChartEntry>)GetValue(EntriesProperty); }
            set { SetValue(EntriesProperty, value); }
        }
        #endregion

        #region MaxValue
        public static readonly BindableProperty MaxValueProperty =
            BindableProperty.Create(
                nameof(MaxValue),
                typeof(double),
                typeof(RadialBarChart),
                defaultValue: -1d,
                propertyChanged: Invalidate);

        public double MaxValue
        {
            get { return (double)GetValue(MaxValueProperty); }
```

```
    set { SetValue(MaxValueProperty, value); }
}
#endregion

#region InnerRadius
public static readonly BindableProperty InnerRadiusProperty =
    BindableProperty.Create(
        nameof(InnerRadius),
        typeof(double),
        typeof(RadialBarChart),
        defaultValue: 50d,
        propertyChanged: Invalidate);

public double InnerRadius
{
    get { return (double)GetValue(InnerRadiusProperty); }
    set { SetValue(InnerRadiusProperty, value); }
}
#endregion

#region BarThickness
public static readonly BindableProperty BarThicknessProperty =
    BindableProperty.Create(
        nameof(BarThickness),
        typeof(double),
        typeof(RadialBarChart),
        defaultValue: 10d,
        propertyChanged: Invalidate);

public double BarThickness
{
    get { return (double)GetValue(BarThicknessProperty); }
    set { SetValue(BarThicknessProperty, value); }
}
#endregion

#region BarSpacing
public static readonly BindableProperty BarSpacingProperty =
    BindableProperty.Create(
        nameof(BarSpacing),
        typeof(double),
        typeof(RadialBarChart),
        defaultValue: 20d,
        propertyChanged: Invalidate);

public double BarSpacing
{
    get { return (double)GetValue(BarSpacingProperty); }
    set { SetValue(BarSpacingProperty, value); }
}
#endregion

#region BarBackgroundColor
public static readonly BindableProperty BarBackgroundColorProperty =
    BindableProperty.Create(
        nameof(BarBackgroundColor),
        typeof(Color),
        typeof(RadialBarChart),
        defaultValue: null,
        propertyChanged: Invalidate);

public Color BarBackgroundColor
{
    get { return (Color)GetValue(BarBackgroundColorProperty); }
    set { SetValue(BarBackgroundColorProperty, value); }
```

```
}
#endregion

#region RoundedCaps
public static readonly BindableProperty RoundedCapsProperty =
    BindableProperty.Create(
        nameof(RoundedCaps),
        typeof(bool),
        typeof(RadialBarChart),
        defaultValue: true,
        propertyChanged: Invalidate);

public bool RoundedCaps
{
    get { return (bool)GetValue(RoundedCapsProperty); }
    set { SetValue(RoundedCapsProperty, value); }
}
#endregion

#region TextColor
public static readonly BindableProperty TextColorProperty =
    BindableProperty.Create(
        nameof(TextColor),
        typeof(Color),
        typeof(RadialBarChart),
        defaultValue: Colors.Black,
        propertyChanged: Invalidate);

public Color TextColor
{
    get { return (Color)GetValue(TextColorProperty); }
    set { SetValue(TextColorProperty, value); }
}
#endregion

#region FontSize
public static readonly BindableProperty FontSizeProperty =
    BindableProperty.Create(
        nameof(FontSize),
        typeof(double),
        typeof(RadialBarChart),
        defaultValue: 12d,
        propertyChanged: Invalidate);

public double FontSize
{
    get { return (double)GetValue(FontSizeProperty); }
    set { SetValue(FontSizeProperty, value); }
}
#endregion

#region ShowLabels
public static readonly BindableProperty ShowLabelsProperty =
   BindableProperty.Create(
       nameof(ShowLabels),
       typeof(bool),
       typeof(RadialBarChart),
       defaultValue: false,
       propertyChanged: Invalidate);

public bool ShowLabels
{
    get { return (bool)GetValue(ShowLabelsProperty); }
    set { SetValue(ShowLabelsProperty, value); }
}
```

```
        #endregion

        #region LabelFormat
        public static readonly BindableProperty LabelFormatProperty =
          BindableProperty.Create(
              nameof(LabelFormat),
              typeof(string),
              typeof(RadialBarChart),
              defaultValue: null,
              propertyChanged: Invalidate);

        public string LabelFormat
        {
            get { return (string)GetValue(LabelFormatProperty); }
            set { SetValue(LabelFormatProperty, value); }
        }
        #endregion

        private static void Invalidate(BindableObject bindable, object oldValue,
object newValue) =>
            ((RadialBarChart)bindable).Invalidate();
    }
}
```

Файл, де присутні методи для малювання діаграми (RadialBarChartDrawable.cs):

```
using Color = Microsoft.Maui.Graphics.Color;
using Font = Microsoft.Maui.Graphics.Font;

namespace DiplomaApp.Controls
{
    public class RadialBarChartDrawable : IDrawable
    {
        #region Field
        private readonly RadialBarChart _chart;
        #endregion

        #region Constructor
        public RadialBarChartDrawable(RadialBarChart chart) => _chart = chart;
        #endregion

        #region Methods
        public void Draw(ICanvas canvas, RectF rect)
        {
            if (_chart.Entries == null)
                return;

            var maxValue = _chart.MaxValue >= 0 ? _chart.MaxValue :
_chart.Entries.Max(x => x.Value);

            var center = rect.Center;

            var spacing = (float)_chart.BarSpacing;
            var radius = (float)_chart.InnerRadius;
            var fontSize = (float)_chart.FontSize;
            var barThickness = (float)_chart.BarThickness;

            canvas.StrokeSize = barThickness;
            canvas.FontColor = _chart.TextColor;
            canvas.FontSize = fontSize;
            canvas.StrokeLineCap = _chart.RoundedCaps ? LineCap.Round :
LineCap.Butt;
```

```
            var startingAngle = 90;
            var angleSpan = _chart.ShowLabels ? 270 : 360;
            var isFullCircle = angleSpan == 360;

            foreach (var entry in _chart.Entries.Reverse())
            {
                // draw bar background
                canvas.StrokeColor = _chart.BarBackgroundColor ??
ToBackgroundColor(entry.Color);
                if (isFullCircle)
                    canvas.DrawCircle(center.X, center.Y, radius);
                else
                    DrawArc(canvas, center, radius, startingAngle, startingAngle -
angleSpan);

                // draw bar
                canvas.StrokeColor = entry.Color;
                if (entry.Value == maxValue && isFullCircle)
                    canvas.DrawCircle(center.X, center.Y, radius);
                else
                    DrawArc(canvas, center, radius, startingAngle, startingAngle -
(float)(entry.Value * angleSpan / maxValue));

                // draw label
                if (_chart.ShowLabels)
                    DrawLabel(canvas, center, spacing, radius, fontSize, entry);

                radius += spacing;
            }
        }

        private void DrawLabel(ICanvas canvas, PointF center, float spacing, float
radius, float fontSize, ChartEntry entry)
        {
            string text = entry.Text;
            if (!string.IsNullOrEmpty(_chart.LabelFormat))
                try
                {
                    text = string.Format(_chart.LabelFormat, entry.Text,
entry.Value);
                }
                catch (FormatException)
                {
                }

            var textSize = canvas.GetStringSize(
                text,
                Font.Default,
                fontSize,
                HorizontalAlignment.Center,
                VerticalAlignment.Center);

            var textRect = new Rect(
                center.X - textSize.Width - spacing,
                center.Y - radius - textSize.Height / 2,
                Math.Ceiling(textSize.Width) + fontSize / 3f,
                Math.Ceiling(textSize.Height));

            canvas.DrawString(
                text,
                textRect,
                HorizontalAlignment.Center,
                VerticalAlignment.Center,
                TextFlow.OverflowBounds);
        }
```

```
        private static void DrawArc(ICanvas canvas, PointF center, float radius,
float start, float end)
        {
            // angle 0 is the right
            // degrees go counter clockwise, meaning 90 is top
            canvas.DrawArc(
                center.X - radius,
                center.Y - radius,
                2 * radius,
                2 * radius,
                startAngle: start,
                endAngle: end,
                clockwise: true,
                closed: false);
        }

        private static Color ToBackgroundColor(Color color)
        {
            return new Color(color.Red, color.Green, color.Blue, color.Alpha *
0.1f);
        }
        #endregion
    }
}
```

Файл, де написано перетворення назв днів тижня із англійської на українську (WeekDaysConverter.cs):

```
using System.Globalization;

namespace DiplomaApp.Converter
{
    // convert names DayOfWeek from English to Ukrainian
    public class WeekDaysConverter : IValueConverter
    {
        #region Methods
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            var day = value;
            return day switch
            {
                DayOfWeek.Sunday => "Нд",
                DayOfWeek.Monday => "Пн",
                DayOfWeek.Tuesday => "Вт",
                DayOfWeek.Wednesday => "Ср",
                DayOfWeek.Thursday => "Чт",
                DayOfWeek.Friday => "Пт",
                DayOfWeek.Saturday => "Сб",
                _ => "",
            };
        }

        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            var day = value;
            return day switch
            {
                DayOfWeek.Sunday => "Нд",
                DayOfWeek.Monday => "Пн",
                DayOfWeek.Tuesday => "Вт",
                DayOfWeek.Wednesday => "Ср",
```

```
                DayOfWeek.Thursday => "Чт",
                DayOfWeek.Friday => "Пт",
                DayOfWeek.Saturday => "Сб",
                _ => "",
            };
        }
        #endregion
    }
}
```

Файл, де прописані методи для запису, отримання, видалення даних до таблиці SettingsItem (SettingsDatabase.cs):

```
using SQLite;
using DiplomaApp.Models;

namespace DiplomaApp.Data
{
    public class SettingsDatabase
    {
        #region Field
        SQLiteAsyncConnection Database;
        #endregion

        #region Constructor
        public SettingsDatabase() { }
        #endregion

        #region Methods
        async Task Init()
        {
            if (Database is not null)
                return;

            Database = new SQLiteAsyncConnection(Constants.DatabasePath,
Constants.Flags);
            _ = await Database.CreateTableAsync<SettingsItem>();
        }

        // get all items from table
        public async Task<List<SettingsItem>> GetItemsAsync()
        {
            await Init();
            return await Database.Table<SettingsItem>().ToListAsync();
        }

        // save item in table
        public async Task<int> SaveItemAsync(SettingsItem item)
        {
            await Init();
            if (item.Id != 0)
                return await Database.UpdateAsync(item);
            else
                return await Database.InsertAsync(item);
        }
        #endregion
    }
}
```

Файл, де прописані методи для запису, отримання, видалення даних до таблиці TaskItem (TaskItemDatabase.cs):

```
using SQLite;
using DiplomaApp.Models;

namespace DiplomaApp.Data
{
    public class TaskItemDatabase
    {
        #region Field
        SQLiteAsyncConnection Database;
        #endregion

        #region Constructor
        public TaskItemDatabase() { }
        #endregion

        #region Methods
        async Task Init ()
        {
            if (Database is not null)
                return;

            Database = new SQLiteAsyncConnection(Constants.DatabasePath,
Constants.Flags);
            _ = await Database.CreateTableAsync<TaskItem>();
        }

        // get all tasks from table
        public async Task<List<TaskItem>> GetTasksAsync()
        {
            await Init();
            return await Database.Table<TaskItem>().ToListAsync();
        }

        // get all tasks from table where date equal today
        public async Task<List<TaskItem>> GetTasksDateTodayAsync(DateTime dateTime)
        {
            await Init();
            return await Database.Table<TaskItem>().Where(d => d.Date ==
dateTime).ToListAsync();
        }

        // get all tasks from table where task is not done
        public async Task<List<TaskItem>> GetTasksNotDoneAsync()
        {
            await Init();
            return await Database.Table<TaskItem>().Where(t => t.Done ==
false).ToListAsync();
        }

        // get all tasks from table where task is done
        public async Task<List<TaskItem>> GetTasksDoneAsync()
        {
            await Init();
            return await Database.Table<TaskItem>().Where(t => t.Done ==
true).ToListAsync();
        }

        // save task in table
        public async Task<int> SaveTaskAsync(TaskItem taskItem)
        {
```

```
            await Init();
            if (taskItem.Id != 0)
                return await Database.UpdateAsync(taskItem);
            else
                return await Database.InsertAsync(taskItem);
        }

        // delete task from table
        public async Task<int> DeleteTaskAsync(TaskItem taskItem)
        {
            await Init();
            return await Database.DeleteAsync(taskItem);
        }
        #endregion
    }
}
```

Файл, де написаний клас, який необхідний для впровадження INotifyPropertyChanged інтерфейсу (BaseObservableModel.cs):

```
using PropertyChanged;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace DiplomaApp.Models
{
    // a base class for Models that need to implement the INotifyPropertyChanged
interface
    [AddINotifyPropertyChangedInterface]
    public abstract class BaseObservableModel : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        protected virtual void OnPropertyChanged([CallerMemberName] string
PropertyName = null)
        {
            PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(PropertyName));
        }
    }
}
```

Файл, де написаний клас, який містить стовбці для налаштувань в таблиці налаштувань бази даних (SettingsItem.cs):

```
using SQLite;

namespace DiplomaApp.Models
{
    // columns in table of settings
    public class SettingsItem
    {
        [PrimaryKey, AutoIncrement]
        public int Id { get; set; }
        public bool DarkTheme { get; set; }
        public bool Notification { get; set; }
    }
}
```

Файл, де написаний клас, який містить параметри для завдань в календарі завдань (TaskDay.cs):

```
using XCalendar.Core.Interfaces;
using XCalendar.Core.Models;

namespace DiplomaApp.Models
{
    // represent TaskDay in CalendarTasksPage
    public class TaskDay: BaseObservableModel, ICalendarDay
    {
        public DateTime DateTime { get; set; }
        public ObservableRangeCollection<TaskItem> Tasks { get; } = new
ObservableRangeCollection<TaskItem>();
        public bool IsSelected { get; set; }
        public bool IsCurrentMonth { get; set; }
        public bool IsToday { get; set; }
        public bool IsInvalid { get; set; }
    }
}
```

Файл, де написаний клас, який містить стовбці для завдань в таблиці завдань бази даних (TaskItem.cs):

```
using SQLite;

namespace DiplomaApp.Models
{
    // columns in table of tasks
    public class TaskItem
    {
        [PrimaryKey, AutoIncrement]
        public int Id { get; set; }
        public string Title { get; set; }
        public string Description { get; set; }
        public DateTime Date { get; set; } = DateTime.Today.Date;
        public bool TimeVisible { get; set; }
        public TimeSpan Time { get; set; }
        public bool Done { get; set; }
    }
}
```

Файл, в якому відбувається переключення теми та задається розмір вікна для додатку Windows (App.cs):

```
using System.Globalization;
using DiplomaApp.Data;
using DiplomaApp.Models;

namespace DiplomaApp;

public partial class App : Application
{
    #region Fields
    readonly SettingsDatabase database;

    public SettingsItem Item
    {
        get => BindingContext as SettingsItem;
        set => BindingContext = value;
    }
    #endregion
```

```
    #region Constructor
    public App(SettingsDatabase settingsDatabase)
    {
        InitializeComponent();

        database = settingsDatabase;

        // provides information about a culture
        CultureInfo.CurrentCulture = new CultureInfo("uk-UA", false);
        CultureInfo.CurrentUICulture = new CultureInfo("uk-UA", false);

        MainPage = new AppShell();
    }
    #endregion

    #region Methods
    protected override async void OnStart()
    {
        base.OnStart();

        // get all items from database
        var settingsItems = await database.GetItemsAsync();
        MainThread.BeginInvokeOnMainThread(() =>
        {
            foreach (var settingsItem in settingsItems)
                Item = settingsItem;
        });

        if (Item == null)
        {
            Item = new SettingsItem()
            {
                Id = 0,
                DarkTheme = false,
                Notification = false
            };
            await database.SaveItemAsync(Item);
            settingsItems = await database.GetItemsAsync();
            MainThread.BeginInvokeOnMainThread(() =>
            {
                foreach (var settingsItem in settingsItems)
                    Item = settingsItem;
            });
        }

        if (Item.DarkTheme == true)
            Current.UserAppTheme = AppTheme.Dark;
        else
            Current.UserAppTheme = AppTheme.Light;
    }

#if WINDOWS
    protected override Window CreateWindow(IActivationState activationState)
    {
        var window = base.CreateWindow(activationState);

        window.X = 200;
        window.Y = 125;

        return window;
    }
#endif
    #endregion
}
```

Сторінка із головним меню додатку та навігацією (AppShell.cs):

```
using DiplomaApp.Views;

namespace DiplomaApp;

public partial class AppShell : Shell
{
    #region Constructor
    public AppShell()
       {
             InitializeComponent();

        Routing.RegisterRoute(nameof(TaskItemPage), typeof(TaskItemPage));
        Routing.RegisterRoute(nameof(EventItemPage), typeof(EventItemPage));
    }
    #endregion
}
```

Файл містить конфігураційні дані для бази даних (Constants.cs):

```
namespace DiplomaApp
{
    // common configuration data
    public static class Constants
    {
        public const string DatabaseFileName = "DiplomaAppSQLite.db3";

        public const SQLite.SQLiteOpenFlags Flags =
            // open the database in read/write mode
            SQLite.SQLiteOpenFlags.ReadWrite |
            // create the database if it doesn't exist
            SQLite.SQLiteOpenFlags.Create |
            // enable multi-threaded database access
            SQLite.SQLiteOpenFlags.SharedCache;

        public static string DatabasePath =>
            Path.Combine(FileSystem.AppDataDirectory, DatabaseFileName);
    }
}
```

Файл для побудови додатку (MauiProgram.cs):

```
using DiplomaApp.Data;
using DiplomaApp.Views;
using CommunityToolkit.Maui;
using Microsoft.Extensions.Logging;
using Plugin.LocalNotification;
#if WINDOWS
    using Microsoft.Maui.LifecycleEvents;
    using Microsoft.UI;
    using Microsoft.UI.Windowing;
#endif

namespace DiplomaApp;

public static class MauiProgram
{
    public static MauiApp CreateMauiApp()
    {
        var builder = MauiApp.CreateBuilder();
        builder.UseMauiApp<App>().ConfigureFonts(fonts =>
        {
            fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
            fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");
        }).UseMauiCommunityToolkit()
```

```
        .UseLocalNotification();

        // register pages and the database access class as services on the
IServiceCollection object
        builder.Services.AddSingleton<App>();
        builder.Services.AddSingleton<TodayTasksPage>();
        builder.Services.AddSingleton<TodayEventsPage>();
        builder.Services.AddSingleton<SettingsPage>();
        builder.Services.AddSingleton<TaskItemDatabase>();
        builder.Services.AddSingleton<EventItemDatabase>();
        builder.Services.AddSingleton<SettingsDatabase>();
        builder.Services.AddTransient<CalendarTasksPage>();
        builder.Services.AddTransient<CalendarEventsPage>();
        builder.Services.AddTransient<TaskItemPage>();
        builder.Services.AddTransient<EventItemPage>();
        builder.Services.AddTransient<StatisticsTasksPage>();
        builder.Services.AddTransient<StatisticsEventsPage>();

#if DEBUG
        builder.Logging.AddDebug();
#endif

        return builder.Build();
    }

    public static MauiApp CreateMauiAppWindows()
    {
        var builder = MauiApp.CreateBuilder();
        builder.UseMauiApp<App>().ConfigureFonts(fonts =>
        {
            fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
            fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");
        }).UseMauiCommunityToolkit();

        // register pages and the database access class as services on the
IServiceCollection object
        builder.Services.AddSingleton<App>();
        builder.Services.AddSingleton<TodayTasksPage>();
        builder.Services.AddSingleton<TodayEventsPage>();
        builder.Services.AddSingleton<SettingsPageWindows>();
        builder.Services.AddSingleton<TaskItemDatabase>();
        builder.Services.AddSingleton<EventItemDatabase>();
        builder.Services.AddSingleton<SettingsDatabase>();
        builder.Services.AddTransient<CalendarTasksPage>();
        builder.Services.AddTransient<CalendarEventsPage>();
        builder.Services.AddTransient<TaskItemPage>();
        builder.Services.AddTransient<EventItemPage>();
        builder.Services.AddTransient<StatisticsTasksPage>();
        builder.Services.AddTransient<StatisticsEventsPage>();

#if WINDOWS
        builder.ConfigureLifecycleEvents(events =>
        {
            events.AddWindows(windowsLifecycleBuilder =>
            {
                windowsLifecycleBuilder.OnWindowCreated(window =>
                {
                    var handle = WinRT.Interop.WindowNative.GetWindowHandle(window);
                    var id = Win32Interop.GetWindowIdFromWindow(handle);
                    var appWindow = AppWindow.GetFromWindowId(id);

                    var presenter = appWindow.Presenter as OverlappedPresenter;
                    presenter.Maximize();
                    presenter.IsResizable = false;
                });
```

```
            });
        });
#endif

#if DEBUG
        builder.Logging.AddDebug();
#endif

        return builder.Build();
    }
}
```