Міністерство освіти і науки України
Дрогобицький державний педагогічний університет імені Івана Франка
кафедра фізики та інформаційних систем

«До захисту допускаю»
завідувач кафедри фізики
та інформаційних систем,
кандидат фіз-мат. наук, доцент
_____ В.Б. Гольський
« ___ » _____ 2026 р.

Розроблення мобільного застосунку для персонального планування завдань

Спеціальність 122 Комп'ютерні науки

Випускова робота
на здобуття кваліфікації – бакалавр з комп'ютерних наук

Автор роботи: Бориславський Максим Сергійович _____
підпис

Науковий керівник: Шаклеіна Ірина Олександрівна _____
підпис

Дрогобич, 2026

Дрогобицький державний педагогічний університет

імені Івана Франка

Зав. кафедрою

_____ _____

(підпис) (дата)

Завдання

на підготовку кваліфікаційної бакалаврської роботи

1. Тема: *«Розроблення мобільного застосунку для персонального планування завдань»*

2. Керівник *доцент, кандидат фізико-математичних наук Шаклеіна І. О.*

(прізвище, ім'я, по батькові)

3. Студент *Бориславський Максим Сергійович*

(прізвище, ім'я, по батькові)

4. Перелік питань, що підлягають висвітленню у кваліфікаційній роботі

1. Аналіз предметної області.

2.Аналіз наявних аналогів. Опис технічного завдання для проєктування та програмної реалізації застосунку.

3. Вимоги, технології та засоби розробки застосунку.

4. Проєктування застосунку.

5. Програмна реалізація та тестування застосунку.

5. Список рекомендованої літератури

1. *Аллен Д. Як привести справи в порядок: мистецтво продуктивності без стресу / Д. Аллен ; пер. з англ. – Київ : Наш формат, 2015. – 352 с.*
2. *Документація Flutter – URL: https://docs.flutter.dev/*
3. *Мова Dart. Документація – URL: https://dart.dev/docs*

6. Етапи підготовки роботи

№	Назва етапу	Термін виконання		Термін звіту перед керівником, кафедрою
1.	Аналіз предметної області	Жовтень 2025		
2.	Аналіз наявних аналогів. Опис технічного завдання для проєктування та програмної реалізації застосунку	Листопад 2025		
3.	Вимоги, технології та засоби розробки застосунку	Грудень 2025		
4.	Проєктування застосунку	Грудень 2025		
5.	Програмна реалізація та тестування	Лютий-травень 2026		

Дата видачі завдання _____

8. Термін подачі роботи керівнику _____

9. З вимогами до виконання кваліфікаційної роботи і завданням ознайомлений _____

(підпис студента)

10. Керівник _____

(підпис)

Захист кваліфікаційної (бакалаврської) роботи

Розроблення мобільного застосунку для персонального планування завдань

Оцінка за стобальною шкалою: _____

Оцінка за національною чотирибальною шкалою: _____

Коротка мотивація захисту: _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____ Голова ЕК _____ _____

Дата *підпис* *прізвище, ім'я*

Секретар ЕК _____ _____

підпис *прізвище, ім'я*

АНОТАЦІЯ

Бориславський М.С. Розроблення мобільного застосунку для персонального планування завдань. Випускова робота, Дрогобицький державний педагогічний університет імені Івана Франка, Дрогобич 2026

В роботі спроєктовано та розроблено мобільний застосунок для персонального планування завдань зі зрозумілою структурою, що забезпечує простоту використання, сучасні підходи до UX/UI та локальне зберігання інформації. Він орієнтований на три чітко визначені типи завдань: ремайндери, списки справ та звички. Програмне рішення реалізовано засобами Flutter/Dart із локальним зберіганням даних у SQLite. Застосунок побудовано за архітектурою MVVM (Model-View-ViewModel), що забезпечує розділення логіки та інтерфейсу користувача. Для забезпечення надійності роботи додатку використовується система станів (state management) на основі Provider та Bloc патернів.

Розроблений мобільний застосунок є повністю автономним рішенням для персонального планування завдань, що не потребує підключення до інтернету та реєстрації зовнішніх облікових записів. Реалізація на Flutter забезпечує кросплатформний потенціал продукту, а локальне зберігання даних – приватність і швидкодію

ABSTRACT

Borislavskyi M.S. Development of a mobile application for personal task planning. Graduation thesis, Drohobych Ivan Franko State Pedagogical University, Drohobych 2026

The work designs and develops a mobile application for personal task planning with a clear structure that ensures ease of use, modern approaches to UX/UI and local storage of information. It is focused on three clearly defined types of tasks: reminders, to-do lists and habits. The software solution is implemented using Flutter/Dart with local data storage in SQLite. The application is built using the MVVM (Model-View-ViewModel) architecture, which provides separation of logic and user interface. To ensure the reliability of the application, a state management system based on the Provider and Bloc patterns is used.

The developed mobile application is a completely autonomous solution for personal task planning that does not require an Internet connection and registration of external accounts. Implementation on Flutter ensures cross-platform potential of the product, and local data storage ensures privacy and speed.

Зміст

Вступ

Швидкий розвиток технологій та інформаційне перенасичення сучасного суспільства призводять до того, що щоденне навантаження на людину невпинно зростає. Люди працюють у багатозадачному режимі, поєднуючи робочі, навчальні та побутові обов'язки. Необхідність одночасно керувати професійними, освітніми та особистими справами вимагає від кожного вміння раціонально розподіляти час і ресурси. Відсутність налагодженої системи планування негативно позначається на результативності роботи, здатності до самоорганізації та загальному емоційному стані.

Мобільні застосунки для організації завдань посіли важливе місце серед інструментів управління часом. Вони дозволяють впорядкувати повсякденні справи, спостерігати за динамікою виконання, культивувати корисні звички та своєчасно отримувати сповіщення. Разом із тим наявні рішення є, зазвичай, платними та мають недоліки: одні пропонують надмірно складний функціонал, інші залежать від хмарних сервісів, що породжує занепокоєння щодо захисту персональних даних. Це зумовлює попит на лаконічні, зрозумілі та незалежні від мережі застосунки, що зберігають інформацію виключно на пристрої користувача.

Актуальним є розроблення мобільного застосунку для персонального планування, побудованого на принципах простоти та доступності. Використання сучасних підходів та технологій дозволяє створити кросплатформне, автономне та швидкісне рішення, орієнтоване на реальні потреби користувачів в галузі тайм-менеджменту.

Метою даної роботи є проектування та практична реалізація мобільного застосунку для персонального планування завдань з підтримкою різних типів завдань та категорій.

Завдання дослідження:

- проаналізувати особливості персонального планування завдань, дослідити існуючі аналоги та їхні функціональні можливості;

- обґрунтувати вибір технологій та засобів для реалізації застосунку;
- сформувати функціональні вимоги та архітектуру системи;
- провести проєктування системи засобами UML-діаграм для візуалізації структури застосунку;
- розробити мобільний застосунок та протестувати його.

Об’єктом дослідження є процеси персонального планування завдань та управління часом.

Предметом дослідження є методи та засоби проектування і розроблення мобільних застосунків для планування завдань.

Методи дослідження: аналіз, порівняння, моделювання, проектування, структурний підхід до розроблення програмного забезпечення та мобільних застосунків зокрема.

Аналіз та узагальнення літературних і наукових джерел мають на меті визначити сучасні тенденції процесів персонального планування завдань та управління часом. Системний аналіз використано для побудови функціональної моделі застосунку. Порівняльний аналіз дає змогу оцінити функціональність існуючих аналогів і визначити конкурентні переваги проєктованого застосунку.

Структура роботи. Випускова робота складається з вступу, основної частини, яка включає теоретичний аналіз предметної області, проєктування застосунку та його практичну реалізацію, висновків, та переліку використаних джерел.

Результати роботи доповідались на XII міжнародній науково-практичній конференції студентів та викладачів факультету фізики, математики, економіки “Актуальні проблеми сучасної науки” (м. Дрогобич, 27-30 квітня 2026 р.)

Розділ 1. Аналіз предметної області

1.1 Поняття персонального планування та управління завданнями

Персональне планування та управління завданнями є фундаментальним аспектом сучасного життя, який визначає ефективність використання часу та досягнення поставлених цілей. У контексті зростаючої кількості щоденних завдань, інформаційного навантаження та багатозадачності, структурований підхід до організації власної діяльності стає не просто корисним навиком, а необхідною умовою продуктивного життя [1].

Під персональним плануванням розуміють цілеспрямований процес, у межах якого людина визначає орієнтири своєї діяльності, конкретизує кроки для їх досягнення та враховує реальні можливості й часові обмеження. Йдеться не про механічне складання переліку справ, а про цілісну систему самоорганізації, що охоплює розстановку пріоритетів, раціональний розподіл часових ресурсів та моніторинг прогресу. Завдяки такому підходу людина переходить від пасивного реагування на обставини до свідомого формування власного шляху до бажаних результатів.

Управління завданнями є невід'ємною складовою планування і становить собою впорядкований процес координації, відстеження та контролю конкретних дій, спрямованих на реалізацію поставлених цілей. Воно передбачає не лише фіксацію переліку справ, а й визначення їхніх ключових характеристик: ступеня важливості, дедлайнів, необхідних ресурсів та поточного стану виконання. Невід'ємною рисою ефективного управління завданнями є гнучкість – здатність оперативно реагувати на зміни, коригувати пріоритети та утримувати увагу на справді значущих напрямах діяльності.

Залежно від характеристик і потреб користувачів завдання, що підлягають плануванню, поділяються на певні категорії (рис. 1.1).

Одноразові завдання є найбільш поширеним типом і являють собою дії, що мають бути виконані один раз у визначений момент або до встановленого

терміну. Їх відрізняє конкретність очікуваного результату та відсутність необхідності повторення. До типових прикладів належать: підготувати та надіслати звіт, придбати подарунок, записатися на прийом до лікаря, здійснити оплату рахунків. Такі завдання проходять повний життєвий цикл від моменту створення до позначення як виконаних або видалення і можуть як мати чіткі часові межі, так і бути відносно гнучкими щодо термінів.

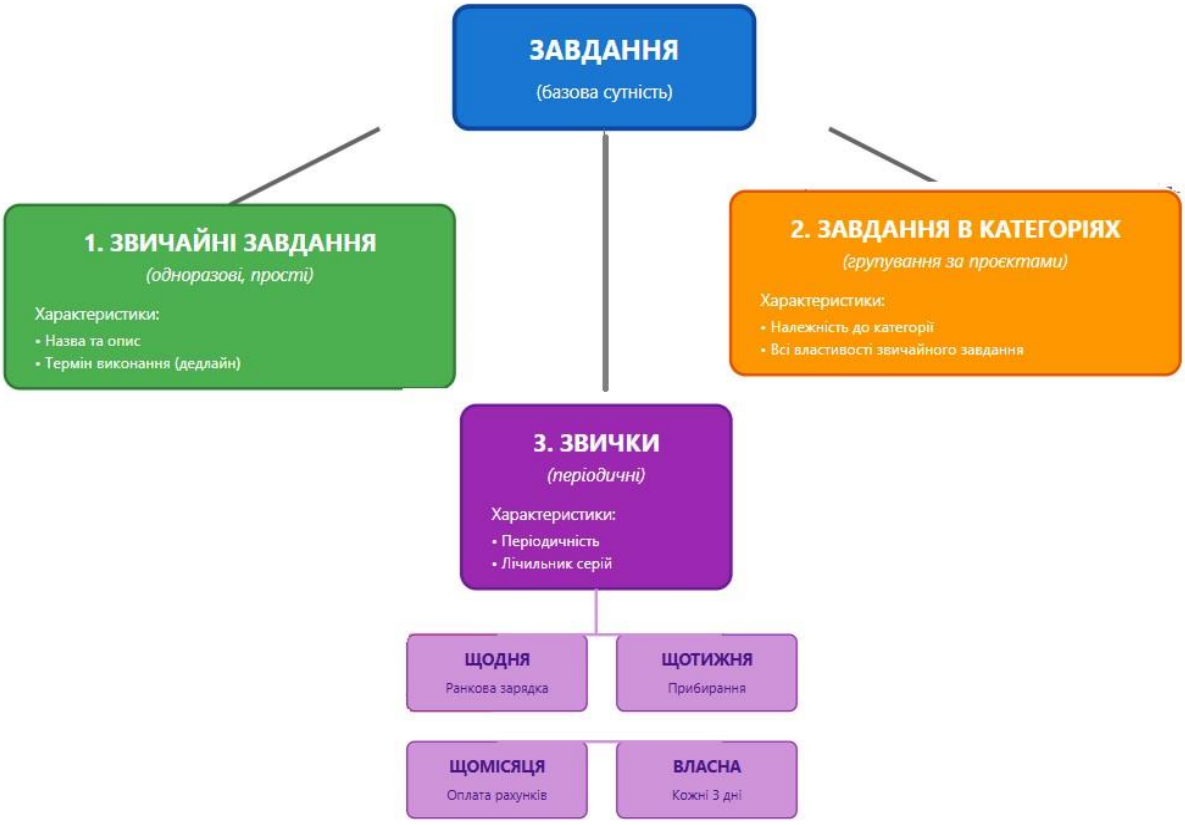

Рис. 1.1. Можливий розподіл завдань за типами

Групування завдань за категоріями відображає природну потребу людини впорядковувати пов'язані між собою справи за спільними ознаками. Категорії можуть охоплювати різні сфери повсякденного життя (професійну діяльність, побут, навчання, здоров'я) або відповідати окремим проєктам чи контекстам виконання, як-от телефонні переговори, заплановані покупки чи робота за комп'ютером. Подібна структуризація спрощує орієнтування у великому масиві справ, дає змогу швидко переключатися між різними напрямами діяльності та отримувати цілісне уявлення про стан виконання в кожному з них. Зокрема,

одна категорія, присвячена певному проєкту, може об'єднувати десятки окремих дій, що пов'язані спільною метою та умовами реалізації.

Звички, або регулярні завдання, являють собою дії, що повторюються з визначеною періодичністю. Це можуть бути щоденні практики (ранкова фізична активність чи читання перед сном), щотижневі справи на кшталт прибирання або планування наступного тижня, а також щомісячні обов'язки (оплата рахунків, створення резервних копій даних тощо). Принципова відмінність звичок від одноразових завдань полягає в тому, що після підтвердження виконання вони не зникають зі списку, а автоматично переносяться на наступний цикл. Систематичне відстеження звичок сприяє формуванню стійких поведінкових моделей і дозволяє контролювати регулярність їх дотримання.

Статуси та стани завдань

Невід'ємною складовою ефективного управління завданнями є можливість відстежувати їхній актуальний стан. За цією ознакою можна виокремити кілька характерних типів.

Активне завдання – це внесений до системи запис, що є актуальним і потребує виконання. Цей стан є початковим для будь-якого щойно створеного завдання. Активні завдання формують основний масив справ, з яким користувач працює в повсякденному режимі.

Термінове завдання є різновидом активного, для якого встановлений дедлайн наближається або вже залишився позаду. Такі завдання вимагають першочергової уваги та невідкладної реакції. Система здатна автоматично визначати ступінь терміновості на підставі зазначених часових меж, виокремлюючи подібні записи візуально, щоб одразу привернути увагу користувача.

Виконане завдання відображає успішно завершену дію. Переведення завдання до цього стану фіксує факт досягнення результату та зберігає відмітку про час завершення. Для регулярних завдань – звичок – виконання не означає остаточного закриття: воно лише підтверджує реалізацію у поточному циклі,

після чого завдання автоматично залишається активним для наступного повторення.

Протерміноване завдання – це завдання, встановлений термін якого сплив, проте результату так і не досягнуто. Поява таких записів є сигналом для перегляду планів, коригування пріоритетів або критичної оцінки реалістичності первісно закладених термінів.

Методології персонального планування

Теоретичну основу сучасних систем управління завданнями формує ряд визнаних методологій, кожна з яких пропонує власний підхід до організації діяльності.

Методологія Getting Things Done ґрунтується на ідеї перенесення всіх незавершених справ із пам'яті до надійної зовнішньої системи. Відповідно до цієї концепції, людський розум найкраще справляється з генеруванням ідей та прийняттям рішень, тоді як утримання в голові великих обсягів інформації про поточні справи суттєво знижує його ефективність. Методологія передбачає послідовне проходження п'яти етапів: фіксацію вхідної інформації, її опрацювання та класифікацію, впорядкування за відповідними категоріями, систематичний перегляд та безпосереднє виконання [1, 2].

Матриця Ейзенхауера пропонує розподіляти завдання відповідно до двох критеріїв: важливості та терміновості [2, 3]. На перетині цих параметрів формуються чотири групи: завдання, що потребують негайного виконання; справи, які варто запланувати на майбутнє; дії, що доцільно делегувати; та завдання, від яких можна відмовитися або відкласти (рис. 1.2). Такий розподіл допомагає зосередити зусилля на дійсно значущих напрямах, а не витрачати ресурси виключно на реагування на терміновість.

Техніка Помодоро орієнтована на структурування часу виконання завдань: робота відбувається інтенсивними 25-хвилинними відрізками, які чергуються з короткими паузами для відновлення. Це сприяє утриманню концентрації та запобігає накопиченню втоми.

Грамотна організація завдань має відчутний вплив на психологічний стан людини. Чітка фіксація справ у зовнішній системі знижує когнітивне навантаження та рівень тривожності, оскільки усуває необхідність утримувати всю інформацію в пам'яті та постійно побоюватися щось забути. Після чітко визначеного переліку запланованих завдань, людина отримує змогу повністю зосередитися на виконанні конкретних дій.

Рис. 1.2. Класифікація завдань за Матрицею Ейзенхауера [3].

Наочне відображення результатів через позначення виконаних справ формує відчуття власної ефективності та слугує додатковим стимулом для подальшої діяльності. Особливого значення це набуває у контексті відстеження звичок, де безперервна послідовність виконання утворює так звані серії та спонукає підтримувати регулярність. Загалом налагоджена система управління завданнями сприяє зростанню відчуття контролю над власним життям,

послаблює схильність до відкладання справ та позитивно впливає на загальну продуктивність.

Сьогодні персональне планування нерозривно пов'язане із застосуванням цифрових інструментів. Мобільні застосунки для організації завдань мають суттєві переваги порівняно з паперовими методами: постійний доступ до актуального списку справ, автоматичні сповіщення, швидке внесення змін і перегрупування записів, пошук за ключовими словами та аналітика виконання. Мобільний формат є особливо зручним, бо дозволяє фіксувати нові ідеї та завдання одразу в момент їх виникнення, не ризикуючи втратити важливу думку.

Застосунок має реалізувати баланс між достатньою функціональністю та зручністю повсякденного використання. Зосередженість на трьох чітко визначених типах завдань (ремайндерах, списках справ і звичках) охоплює ключові потреби користувачів без надмірного ускладнення інтерфейсу. Локальне зберігання даних забезпечує повну приватність та високу швидкодію, а інтуїтивно зрозумілий інтерфейс сподобається новим користувачам.

Таким чином, персональне планування та управління завданнями являють собою комплексну систему, що об'єднує теоретичні методології, психологічні закономірності та технологічні рішення заради спільної мети: раціонального використання часу та послідовного просування до визначених цілей. Розуміння природи різних типів завдань, їхніх станів і особливостей поведінки має нга меті допомгти створити інструмент, що відповідає реальним потребам користувачів у сучасному динамічному середовищі.

1.2 Огляд популярних застосунків для планування завдань

Для формування цілісного уявлення про функціональні можливості та вимоги до розробленого застосунку потрібно проаналізувати наявні на ринку мобільних застосунків для планування завдань рішення. З цією метою розглянуто низку популярних аналогів у відкритому доступі, визначено їхні характерні особливості та сформовано основі вимоги до розробленої системи.

Todoist (https://todoist.com) є одним із найстаріших і найпоширеніших інструментів для організації завдань із багатомільйонною аудиторією користувачів по всьому світу. Застосунок вирізняється продуманим інтерфейсом, що робить роботу із завданнями зрозумілою та результативною [2, 4]. Рішення доступне на всіх провідних платформах: iOS, Android, Windows, macOS та у вебверсії із повноцінною синхронізацією даних між пристроями.

Функціональні можливості Todoist охоплюють створення завдань та їх групування у проєкти, введення завдань на основі природної мови, визначення термінів виконання й налаштування регулярних повторень, а також призначення пріоритетів і міток для зручної категоризації (рис. 1.3). Серед характерних особливостей застосунку є підтримка підзавдань, що дозволяє розбивати складні справи на дрібніші, легші у виконанні кроки [2].

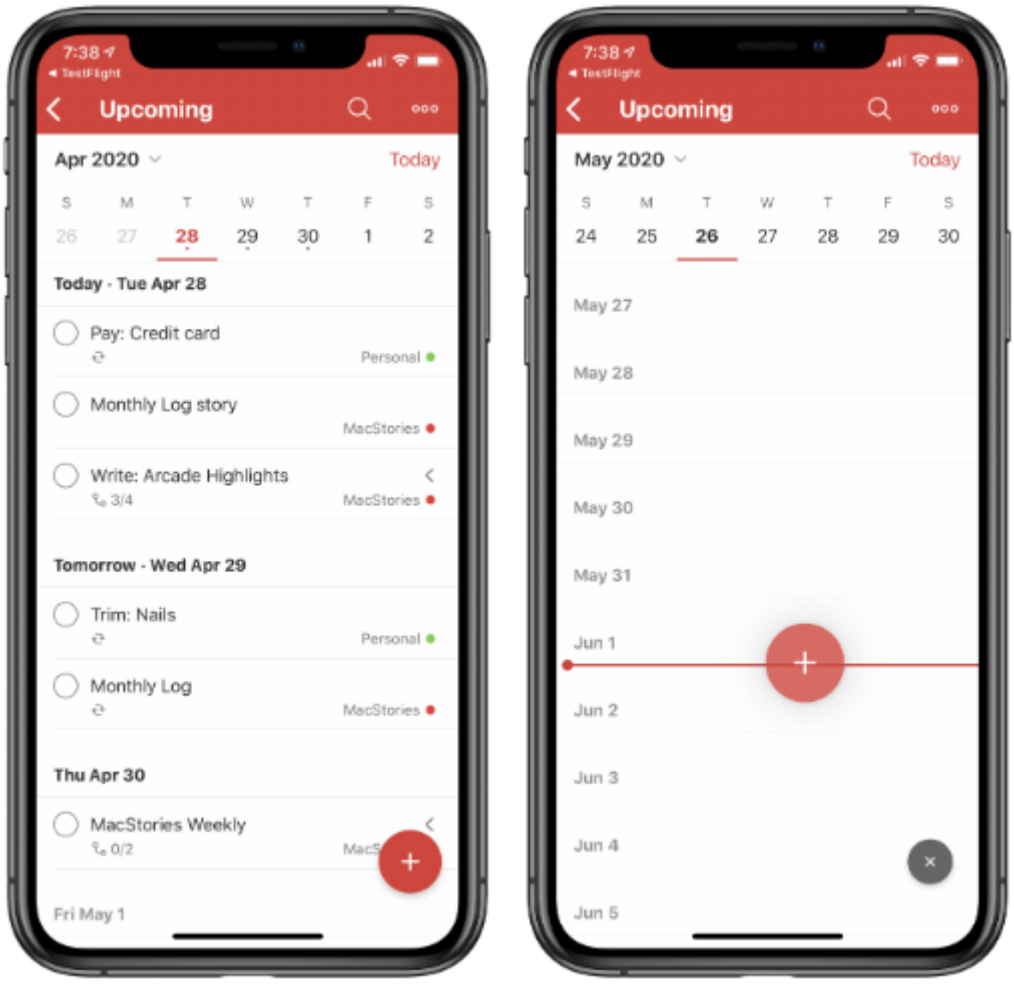

Рис. 1.3. Інтерфейс застосунку мобільної версії Todoist.

Todoist підтримує спільний доступ до проєктів для командної роботи та сумісний із понад вісімдесятьма зовнішніми сервісами, зокрема Google Calendar, Outlook, Slack і Zapier, що суттєво розширює можливості автоматизації робочих процесів [5].

До переваг застосунку належать лаконічний інтерфейс, висока швидкодія та стабільна синхронізація. Водночас Todoist має обмеження. Значна частина корисних функцій, включаючи нагадування та календарні представлення, доступна виключно в платній версії. Безкоштовний план обмежує користувача п'ятьма проєктами та базовим набором інструментів. Окрім того, для повноцінної роботи застосунок потребує обов'язкової реєстрації та постійного підключення до мережі, що викликає занепокоєння щодо конфіденційності персональних даних.

Microsoft To Do (https://to-do.microsoft.com) – безкоштовний продукт компанії Microsoft, для користування яким достатньо мати обліковий запис Microsoft [6].

Базовий функціонал охоплює створення завдань із описом, дедлайнами, нагадуваннями та налаштуванням повторень. Передбачено гнучке впорядкування: групування у списки та сортування за датою виконання, датою створення або в алфавітному порядку. Характерною рисою є функція «Мій день», що щоранку пропонує користувачу сформувати перелік пріоритетних справ на поточний день, відсіюючи другорядну інформацію.

Застосунок відрізняється зручним інтерфейсом із підтримкою перетягування елементів для швидкої реорганізації та щільною інтеграцією з екосистемою продуктів Microsoft. Повна безкоштовність із доступом до всіх функцій без підписки є вагомою конкурентною перевагою.

Разом із тим Microsoft To Do має певні недоліки. Інструменти для спільної роботи залишаються мінімальними: відсутні коментарі до завдань, журнал активності та рольові налаштування доступу. Інтеграція з сервісами поза екосистемою Microsoft є обмеженою, а push-сповіщення подекуди надходять із затримкою або не з'являються взагалі [6]. Для користувачів, які не

використовують інші продукти компанії, практична цінність застосунку може виявитися невисокою.

TickTick (https://ticktick.com) позиціонується як універсальне рішення, що поєднує в одному застосунку список справ, календар, трекер звичок та інструменти для підтримання зосередженості (рис. 1.4). Це дозволяє замінити кілька окремих інструментів єдиною платформою [7].

Функціональні можливості TickTick є досить широкими. Поряд зі стандартним набором застосунок пропонує п'ять варіантів календарного відображення, вбудований таймер Помодоро, матрицю Ейзенхауера для наочної розстановки пріоритетів, а також власні фільтри для швидкого доступу до потрібних записів.

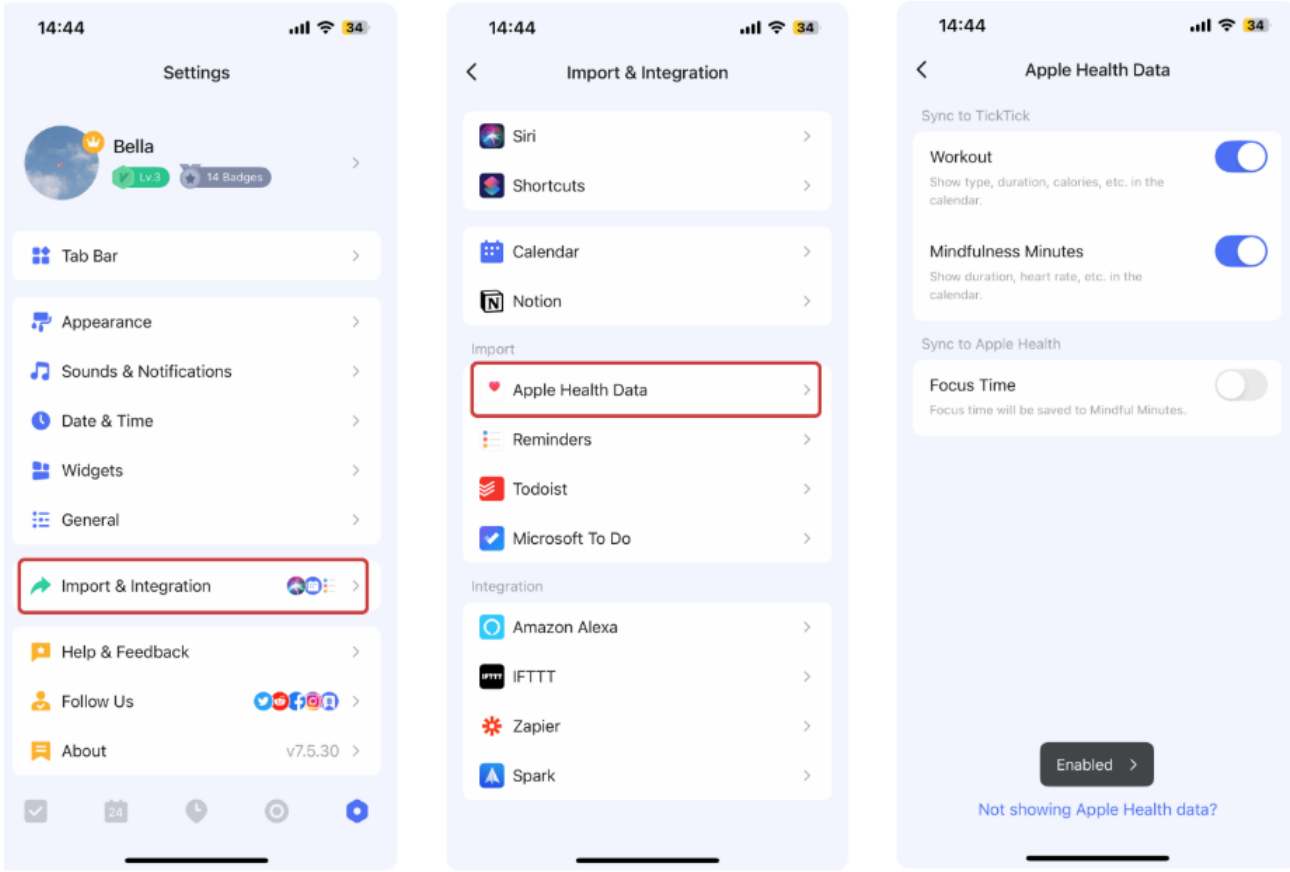

Рис. 1.4. Інтерфейс застосунку Tick Tick.

Окремої уваги заслуговує вбудований трекер звичок. Система підтримує три способи фіксації виконання: автоматичний, ручний та досягнення заданої кількості повторень. Користувачі можуть налаштовувати індивідуальну частоту

виконання звичок, вести журнал настрою та переглядати детальну статистику прогресу.

Застосунок також надає розгорнуту аналітику: кількість виконаних завдань, дані про сеанси фокусування та показники дотримання звичок [8].

Серед недоліків варто відзначити перевантаженість функціоналом, опанування якого потребує значних часових витрат. Більшість розширених можливостей доступні виключно за платною підпискою. Трекер звичок, попри свою функціональність, поступається за гнучкістю спеціалізованим рішенням у цій категорії.

Any.do (https://www.any.do) – ще один поширений застосунок для організації завдань, головний акцент якого зроблено на щоденному плануванні та простоті взаємодії. Рішення доступне на всіх провідних платформах із синхронізацією даних у реальному часі.

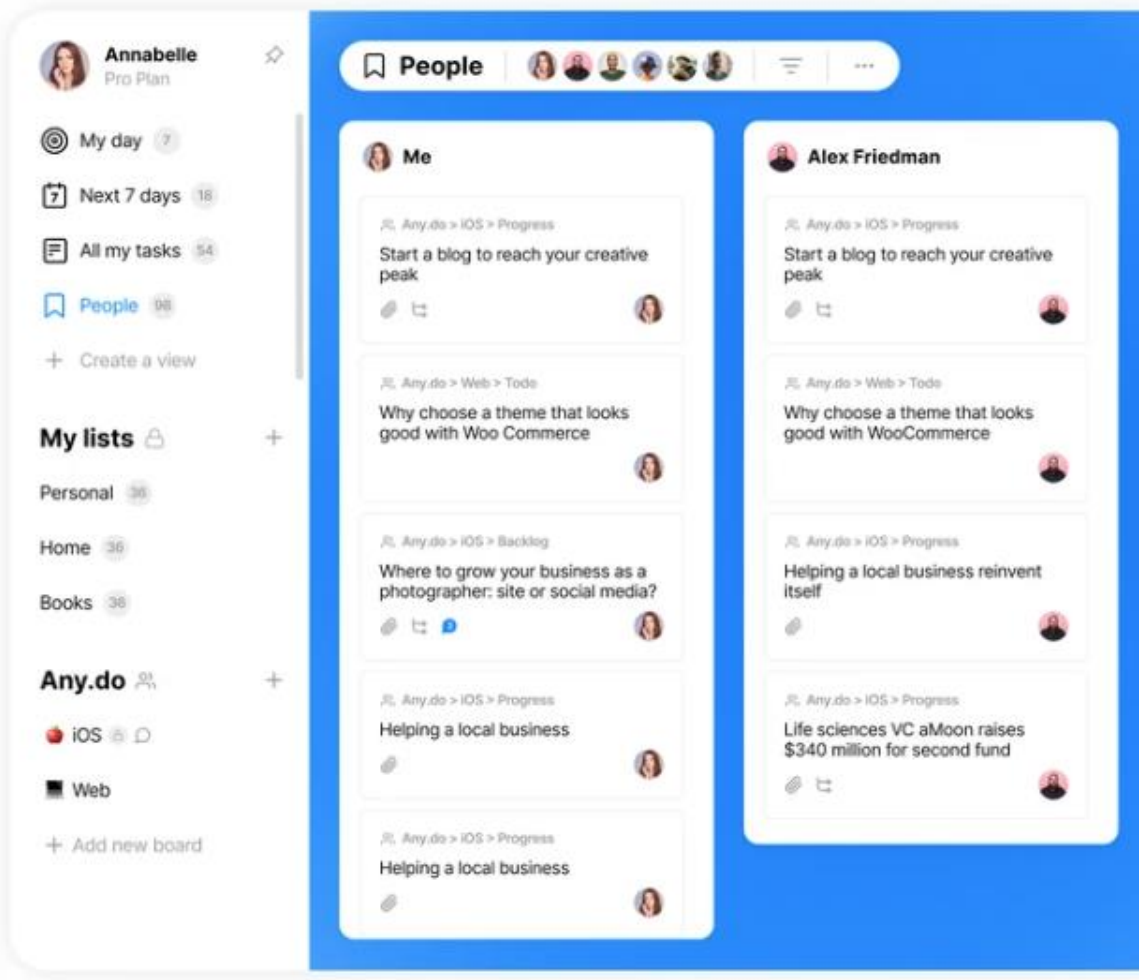

Рис. 1.5. Інтерфейс застосунку Any.do.

Базовий функціонал застосунку охоплює створення завдань із підзавданнями, налаштування нагадувань і повторень, а також організацію записів у списки та папки. Характерною рисою є вбудований календар, інтегрований безпосередньо в головний інтерфейс, що дозволяє відстежувати завдання та події в єдиному просторі. Додатково передбачено функцію «Планувальник дня», яка щоранку пропонує переглянути та впорядкувати справи на поточний день (рис. 1.5).

Any.do підтримує голосове введення завдань і нагадування на основі геолокації. До переваг належать продуманий і привабливий дизайн, висока швидкодія та зручність щоденного планування. Разом із тим, як і більшість конкурентів, застосунок обмежує доступ до розширених можливостей платною підпискою.

За результатами проведеного аналізу вдалося виявити кілька характерних тенденцій, спільних для популярних застосунків із планування завдань. По-перше, всі розглянуті рішення побудовані за моделлю freemium: базові функції доступні безкоштовно, тоді як найбільш потужні інструменти потребують оплати. Винятком є Microsoft To Do, що розповсюджується повністю безкоштовно, однак має найбільш скромний функціональний набір серед усіх проаналізованих варіантів.

По-друге, жоден із застосунків не дозволяє повноцінно працювати без реєстрації облікового запису та підключення до мережі, необхідного для синхронізації між пристроями. Це породжує закономірні питання щодо конфіденційності, оскільки персональні дані користувачів зберігаються на віддалених серверах. По-третє, простежується стійка тенденція до ускладнення інтерфейсів через прагнення розробників охопити якомога ширший спектр функцій, що нерідко дезорієнтує користувачів із відносно простими потребами.

1.3 Вимоги до системи планування завдань

Виявлені недоліки існуючих рішень окреслили орієнтири для розроблення нового застосунку. Серед ключових принципів — зосередженість на трьох чітко визначених типах завдань, повністю локальне зберігання даних без необхідності реєстрації, доступність усього функціоналу без платних підписок та лаконічний інтерфейс без зайвих елементів.

Ефективна система планування завдань має відповідати комплексу вимог, що визначають її функціональність, зручність і технічну надійність. Усі вимоги поділяються на функціональні та нефункціональні, кожна група яких відіграє важливу роль у створенні дійсно корисного інструменту.

Функціональні вимоги

Управління завданнями різних типів. Система підтримує створення, редагування та видалення трьох типів завдань: одноразових ремайндерів, списків справ і звичок. Кожен тип має власний екран створення з відповідним набором полів. Користувач може швидко обрати потрібний тип і заповнити необхідну інформацію без зайвих кроків.

Налаштування періодичності для звичок. Для регулярних завдань передбачено гнучкі параметри повторення: щоденне виконання, виконання у певні дні тижня, щотижневий або щомісячний режим, а також довільні інтервали через задану кількість днів. Система автоматично розраховує наступну дату виконання та відображає її користувачеві.

Відстеження статусу завдань. Система автоматично визначає поточний стан кожного запису: активне, термінове, прострочене або виконане. Зміна статусу відбувається як вручну – через позначення виконаним, так і автоматично при настанні дедлайну.

Перегляд та фільтрація завдань. Користувач може переглядати завдання у різних режимах: усі активні, на сьогодні, на тиждень, прострочені та виконані. Передбачено фільтрацію за датами та статусами.

Пошук завдань. Система забезпечує швидкий пошук за назвою або описом, що особливо важливо при значній кількості збережених записів.

Нефункціональні вимоги

Простота та інтуїтивність інтерфейсу. Інтерфейс орієнтований на користувачів будь-якого рівня технічної підготовки. Головний екран одразу відображає актуальні завдання, а створення нового запису потребує мінімальної кількості дій. Усі основні функції доступні не глибше ніж за два рівні навігації. Складні налаштування винесені в окремі меню, тоді як базовий функціонал доступний одразу. Кожен елемент інтерфейсу має очевидне призначення.

Швидкодія та продуктивність. Застосунок працює стабільно навіть на пристроях середнього класу. Час запуску не перевищує двох секунд, операції зі створення, редагування та видалення завдань виконуються без помітних затримок, а прокрутка залишається плавною навіть при великій кількості записів.

Надійність та стабільність. Усі зміни зберігаються до бази даних одразу після їх внесення. При несподіваному завершенні роботи застосунку дані залишаються цілими. Некоректне введення обробляється через відповідні повідомлення про помилки без аварійного завершення.

Зручність використання. Застосунок надає зворотний зв'язок на кожну дію: анімації при створенні завдання, візуальне підтвердження виконання, плавні переходи між екранами. Видалення завдань супроводжується запитом на підтвердження для запобігання випадковій втраті даних.

Приватність та безпека даних. Уся інформація зберігається локально на пристрої без передачі на зовнішні сервери. Застосунок не потребує реєстрації, підключення до мережі або надання персональних даних. База даних SQLite захищена від несанкціонованого доступу на рівні операційної системи.

Автономність роботи. Увесь функціонал від створення завдань до перегляду статистики – доступний без підключення до інтернету, що забезпечує використання застосунку за будь-яких умов.

Адаптивність інтерфейсу. Застосунок коректно відображається на екранах різних розмірів та орієнтацій, адаптується до системних налаштувань розміру шрифту та підтримує світлу й темну теми оформлення.

Ефективність використання ресурсів. Споживання оперативної пам'яті в межах 100 МБ, розмір встановленого застосунку до 20 МБ, мінімальний вплив на заряд батареї при фоновій роботі.

Якість коду та архітектури. Архітектура побудована за принципом розділення відповідальності з використанням патерну MVVM, що забезпечує чітке розмежування бізнес-логіки, представлення даних і роботи з базою. Код відповідає стандартам оформлення Dart. Структура бази даних підтримує референційну цілісність через зовнішні ключі з каскадним видаленням пов'язаних записів.

Таким чином, сформований комплекс вимог охоплює як функціональні можливості, необхідні для повноцінного управління завданнями, так і якісні характеристики, що визначають рівень користувацького досвіду. Визначальним має бути баланс між достатньою функціональністю та простотою щоденного використання.

Розділ 2. Вибір засобів та технологій розробки

2.1. Платформа розробки Flutter/Dart

Після детального вивчення предметної області та порівняльного аналізу наявних рішень було визначено ключові технологічні вимоги до розроблюваного застосунку. На їх підставі здійснено вибір інструментів розроблення, що найбільш повно відповідають поставленим завданням. Основним середовищем для створення застосунку став фреймворк Flutter, а мовою реалізації програмної логіки Dart.

Flutter є сучасним кросплатформним фреймворком, що дозволяє створювати програмні продукти для різних платформ із єдиної кодової бази [6]. Інструмент охоплює розроблення користувацьких інтерфейсів для мобільних операційних систем, вебплатформ і десктопних середовищ. Широке визнання Flutter у середовищі розробників пояснюється передовим підходом до рендерингу графічного контенту та багатим набором вбудованих інструментів.

Принципова відмінність Flutter від традиційних фреймворків полягає у способі відтворення графічних елементів. Замість звернення до нативних компонентів операційної системи фреймворк використовує власний графічний рушій на базі Skia для безпосереднього рендерингу інтерфейсу. Завдяки цій архітектурній особливості застосунок має ідентичний зовнішній вигляд незалежно від платформи та версії операційної системи. Такий підхід надає розробникам повний контроль над кожним пікселем інтерфейсу та гарантує точну відповідність реалізації дизайнерським макетам.

Серед ключових переваг використання Flutter варто відзначити наступні аспекти:

- технологія hot reload забезпечує миттєве відображення внесених змін в інтерфейсі без необхідності повної перекомпіляції проєкту;
- можливість підтримки єдиної кодової бази для операційних систем Android та iOS, значно знижує фінансові та часові витрати на розробку та подальшу підтримку продукту.

- Flutter пропонує розгалужену екосистему готових компонентів інтерфейсу (віджетів), які дозволяють швидко створювати сучасні, естетично привабливі та високоінтерактивні користувацькі інтерфейси.
- фреймворк підтримується компанією Google, що гарантує регулярні оновлення та активну підтримку з боку спільноти розробників.

Базовою мовою програмування для Flutter є Dart – мова, розроблена компанією Google цілеспрямовано для створення клієнтських застосунків [7]. Вона поєднує сучасний лаконічний синтаксис із принципами об'єктно-орієнтованого програмування, що робить її природним вибором для реалізації бізнес-логіки мобільних рішень. Актуальність Dart підтверджується результатами галузевих досліджень технологічних уподобань розробників. Зокрема, згідно з даними опитування фахівців у сфері мобільної розробки, проведеного порталом DOU у 2025 році (рис. 2.1), Dart увійшла до трійки мов програмування, які розробники найчастіше планують застосовувати у своїх майбутніх проєктах [8].

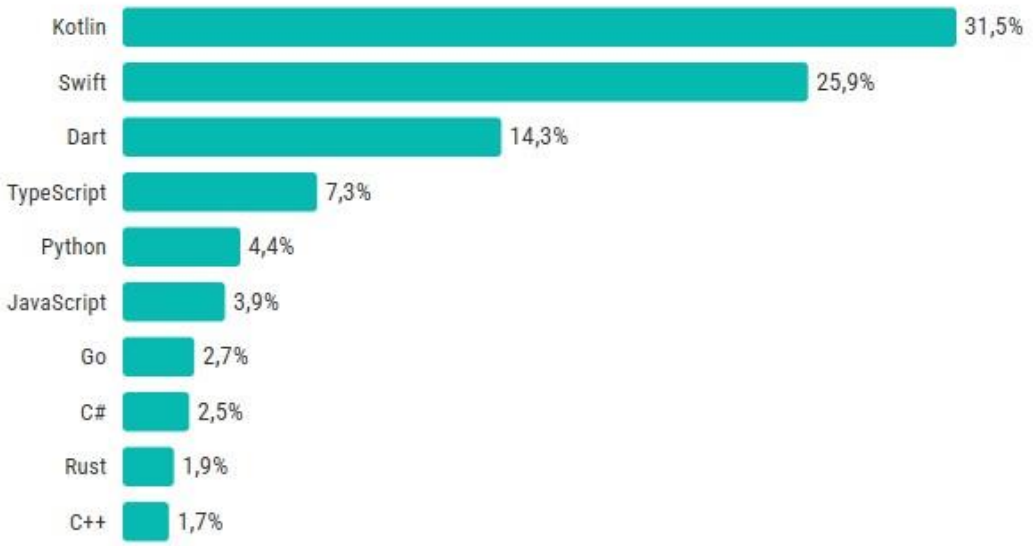

Рис. 2.1. Вибір мови для наступного проєкту mobile-розробниками

Dart вирізняється низкою технічних характеристик, що роблять її придатною для розроблення складних мобільних рішень. Насамперед, мова пропонує розвинений механізм асинхронного програмування через конструкції async та await, що є принципово важливим для підтримання плавної взаємодії з інтерфейсом без блокування головного потоку виконання.

На етапі розроблення застосовується JIT-компіляція, яка забезпечує миттєве відображення внесених змін і прискорює цикл тестування. У фінальній збірці використовується AOT-компіляція, що підвищує продуктивність виконання коду та скорочує час запуску програми. Повноцінна підтримка об'єктно-орієнтованої парадигми, своєю чергою, спрощує побудову чітко структурованої та масштабованої архітектури.

Для застосунку з локальним зберіганням даних визначальними є висока швидкодія, надійний захист інформації та зручність повсякденного використання. Поєднання Flutter і Dart повністю відповідає цим критеріям, дозволяючи реалізувати весь необхідний функціонал у межах єдиного технологічного стеку.

Спільне використання Flutter та Dart створює сприятливі умови для розроблення застосунків із однаково високою якістю роботи та візуального відображення як в екосистемі Android, так і в середовищі iOS. Засоби Flutter забезпечують точний контроль над кожним елементом інтерфейсу, що дозволяє досягти простого та зручного досвіду взаємодії, який відповідає сформульованим вимогам до системи планування завдань.

Таким чином, вибір технологічного стеку Flutter/Dart є обґрунтованим рішенням для реалізації розробленого мобільного застосунку для персонального планування завдань, оскільки забезпечує оптимальний баланс між функціональністю, продуктивністю, кросплатформеністю та зручністю розроблення.

2.2. Засоби збереження даних, що містяться в системі

Надійне та безпечне зберігання даних є однією з ключових вимог до мобільного застосунку для планування завдань. У розробленому рішенні для цього обрано локальну реляційну базу даних SQLite.

Уся персональна інформація користувача (завдання, нотатки, облікові дані автентифікації) не передається до хмарних сервісів, а зберігається виключно на пристрої засобами вбудованої бази даних SQLite [9]. Такий підхід дозволяє

мінімізувати ризики, пов'язані з передаванням чутливих даних через мережу, та забезпечує повноцінну автономну роботу застосунку без підключення до інтернету.

Інтеграція SQLite у Flutter-застосунок реалізована за допомогою пакету sqflite, який надає можливість [10, 11]:

- створювати та керувати таблицями за допомогою sql-запитів; о здійснювати операції вставки, оновлення, видалення та вибірки даних;
- шифрувати базу даних (через додаткові плагіни) для підвищення безпеки.

Такий підхід до збереження даних забезпечує автономність та швидкодію, простоту структурування даних завдяки реляційній моделі та захист персональних даних без потреби постійного підключення до мережі, що особливо важливо для застосунків, орієнтованих на конфіденційність.

2.3. Апаратні вимоги до розроблюваного мобільного застосунку

Для забезпечення коректної та стабільної роботи застосунку необхідно враховувати мінімальні апаратні вимоги до мобільного пристрою. Нижче наведено рекомендовані параметри, дотримання яких гарантує оптимальну продуктивність у межах функціонального призначення системи.

Оперативна пам'ять. Рекомендований мінімальний обсяг RAM становить 2 ГБ – достатньо для плавної роботи інтерфейсу та операцій із локальною базою даних SQLite. Для досягнення кращих показників продуктивності бажано використовувати пристрої з 3–4 ГБ оперативної пам'яті.

Процесор. Рекомендується багатоядерний процесор із тактовою частотою не менше 1,4–1,5 ГГц. Такі характеристики забезпечують виконання операцій автентифікації, обробки введених даних і звернень до бази даних без помітних затримок, у тому числі в умовах багатозадачності або фонової роботи застосунку.

Мінімально підтримувані версії ОС: Android 8.0 (Oreo) та iOS 14.0. Вказані версії ОС гарантують стабільну підтримку бібліотек SQLite та необхідних системних сервісів.

Застосунок не потребує значного обсягу дискового простору, однак рекомендується мати щонайменше 100 МБ вільної пам'яті для встановлення, збереження локальних записів, кешованих даних та журналів дій. Більший обсяг вільного простору сприятиме стабільнішій роботі в довгостроковій перспективі.

Дотримання зазначених апаратних вимог забезпечує надійне функціонування застосунку та безпечне зберігання персональних даних користувача.

Розділ 3. Опис функціональної моделі системи

До початку роботи над мобільним застосунком для персонального планування завдань потрібно визначити його основні функціональні можливості та сценарії взаємодії з користувачем. З цією метою побудовано UML-діаграму варіантів використання, яка дозволяє наочно відобразити всі способи взаємодії користувача із системою та встановити зв'язки між її функціональними складовими.

З огляду на результати аналізу вимог, проведеного в першому розділі, та визначені на його основі функціональні особливості, система передбачає лише одного актора – користувача застосунку. Відсутність додаткових акторів, таких як адміністратор або зовнішні сервіси, обумовлена архітектурним рішенням про локальне зберігання даних і повністю автономну роботу застосунку.

Користувачеві доступний такий функціонал, що охоплює всі аспекти управління завданнями:

- реєстрація нового облікового запису, що передбачає вибір аутентифікаційних даних;
- автентифікація (вхід у систему) – є можливість увійти в застосунок, використовуючи біометричну автентифікацію за відбитком пальця. Функція автентифікації забезпечує додатковий рівень захисту персональних даних, запобігаючи несанкціонованому доступу до інформації про завдання. При першому запуску користувач може налаштувати біометричну автентифікацію за відбитком пальця, яка активуватиметься при кожному наступному відкритті застосунку. Якщо використання біометрії неможливе або небажане, доступ забезпечується через логін і пароль.
- Управління ремайндерами передбачає створення нового завдання із зазначенням назви, опису, терміну виконання та рівня пріоритетності, перегляд і редагування будь-яких параметрів, позначення завдання як виконаного з автоматичною фіксацією часу завершення, а також повне видалення запису із системи.

- Робота зі списками справ охоплює створення нового списку з довільною кількістю пунктів, редагування вмісту та назви, позначення окремих пунктів як виконаних і видалення списку.
- Управління звичками включає створення нової звички з налаштуванням параметрів періодичності: щоденне виконання, виконання у визначені дні тижня, щотижневий або щомісячний режим, а також довільний інтервал повторення через задану кількість днів. Доступне редагування параметрів існуючої звички та її видалення.
- Перегляд і фільтрація завдань дозволяє переглядати всі активні записи з розподілом за типами, а також окремий список прострочених завдань, що потребують першочергової уваги.
- Зміна статусу завдань передбачає позначення виконаним із можливістю відновлення до активного стану у разі помилкової дії.
- Налаштування застосунку охоплюють увімкнення або вимкнення біометричної автентифікації, вибір теми оформлення, зміну формату відображення дат, а також очищення архіву виконаних завдань зі збереженням статистики серій для звичок.
- Вихід із системи завершує поточний сеанс роботи з обліковим записом.

На рисунку 3.1 представлено повну UML-діаграму варіантів використання для проектованого застосунку.

На діаграмі чітко видно центральну роль користувача як єдиного актора системи, а також всі основні функціональні можливості, згруповані за типами завдань та операціями над ними.

Побудована функціональна модель системи охоплює повний життєвий цикл завдань різних типів: від створення через редагування та виконання до архівування або видалення.

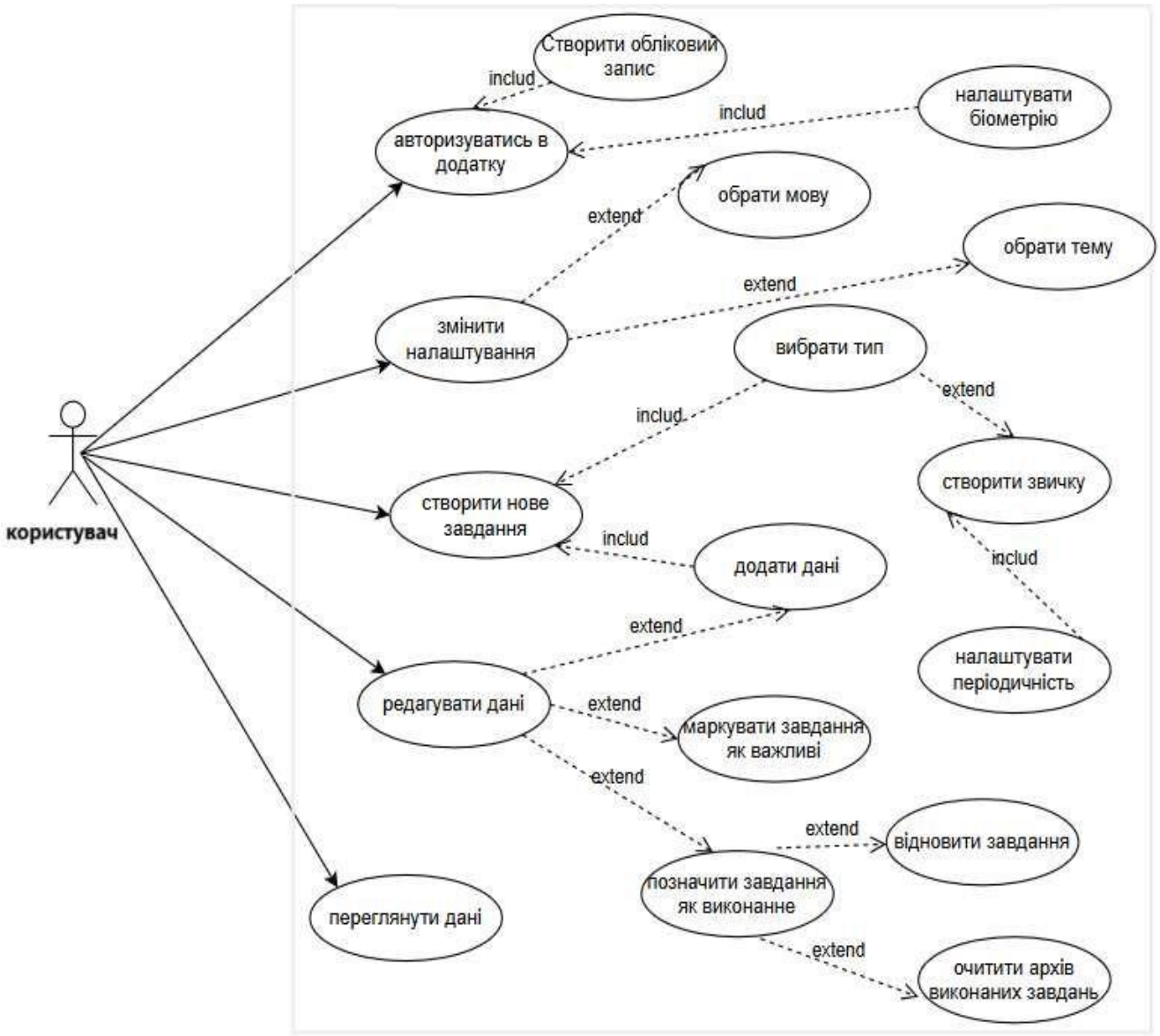

Рис. 3.1. UML-діаграма варіантів використання застосунку

Особлива увага приділена функціоналу роботи зі звичками, оскільки саме цей тип завдань передбачає найбільш складну логіку з автоматичним розрахунком наступних дат виконання та веденням лічильника серій виконань.

Розділ 4. Проектування мобільного застосунку

4.1. Архітектура застосунку та структурна схема системи

На рис. 4.1. подано розроблену структурну схему архітектури мобільного застосунку. Система умовно поділена на шість логічних модулів: обробки даних, зберігання, управління користувачами, інтерфейс, локалізація й інтеграція.

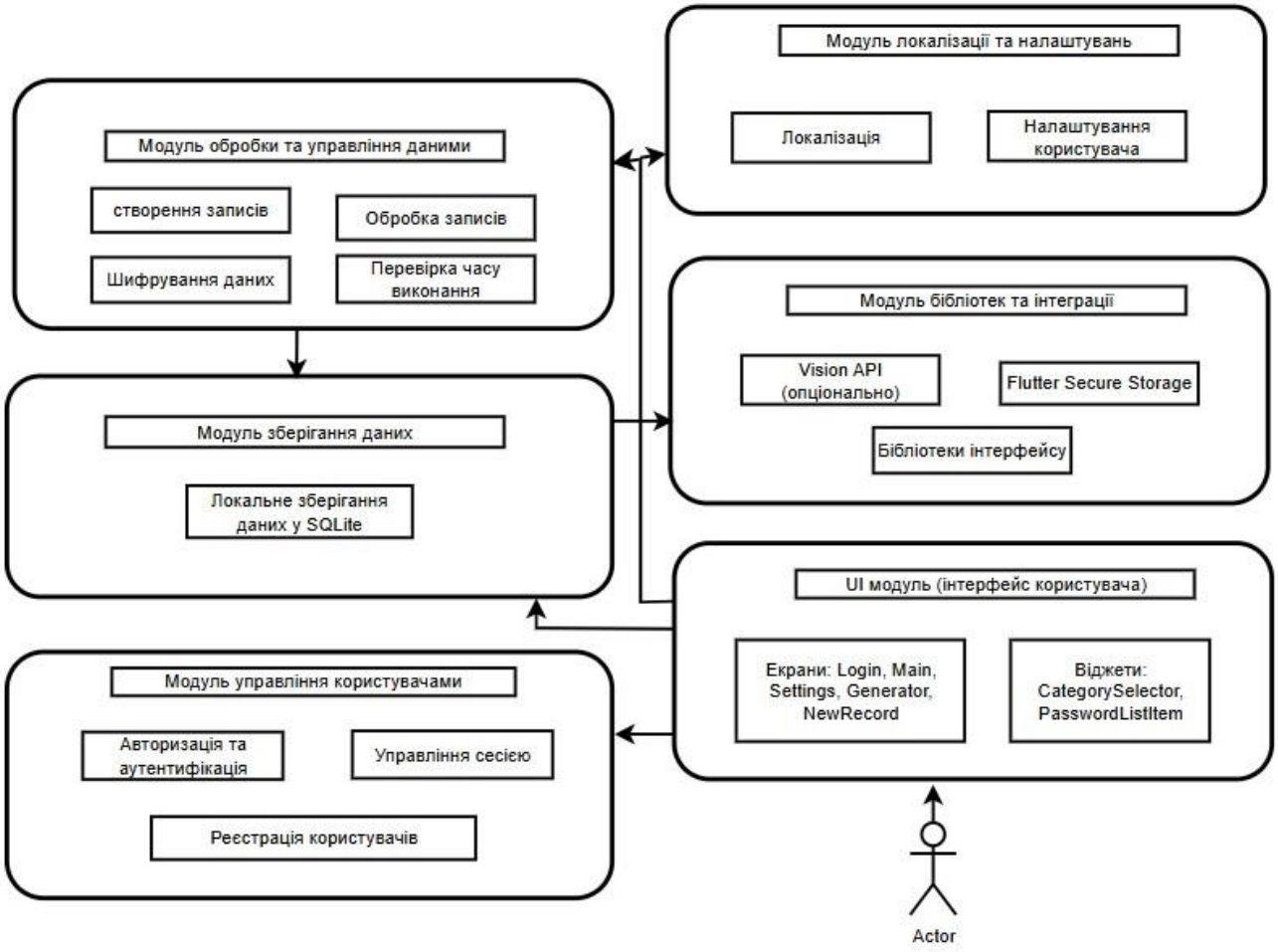

Рис. 4.1. Структурна схема системи розроблювального додатку

Кожен модуль містить відповідні підсистеми, що відповідають за ключові функції. UI-модуль є точкою взаємодії з користувачем і ініціює більшість запитів до решти модулів. Модулі тісно пов'язані між собою через функціональні залежності.

4.2. Діаграма послідовності та діаграма класів застосунку

Діаграми послідовності дозволяють відобразити порядок викликів методів між об'єктами під час виконання конкретного сценарію. З огляду на це процеси створення нового запису звички та позначення завдання як виконаного деталізовано за допомогою UML-діаграм послідовності.

Розглянемо спочатку процес створення нової звички. Він охоплює такі етапи:

— Користувач відкриває форму «Створити звичку».

— Інтерфейс відображає параметри вибору частоти повторення.

— Користувач обирає тип повторюваності: щодня, щотижня, через N днів тощо.

— UI передає відповідні дані контролеру через метод setRecurrence(type, interval).

— Контролер формує об'єкт RecurrencePattern на основі отриманих параметрів.

— RecurrencePattern будує внутрішню структуру повторюваності.

— Контролер отримує сформований об'єкт pattern.

— Користувач натискає кнопку «Зберегти».

— UI передає зібрані дані звички через метод createHabit(habitData).

— HabitController направляє звичку разом із патерном до HabitRepository.

— Репозиторій зберігає RecurrencePattern у базі даних.

— Після цього репозиторій зберігає саму звичку.

— DatabaseHelper повертає підтвердження успішного запису.

— Інтерфейс відображає повідомлення про успішне створення звички.

На рисунку 4.2 наведено діаграму, що відображає процес створення нової звички, який включає вибір періодичності та формування шаблону повторюваності:

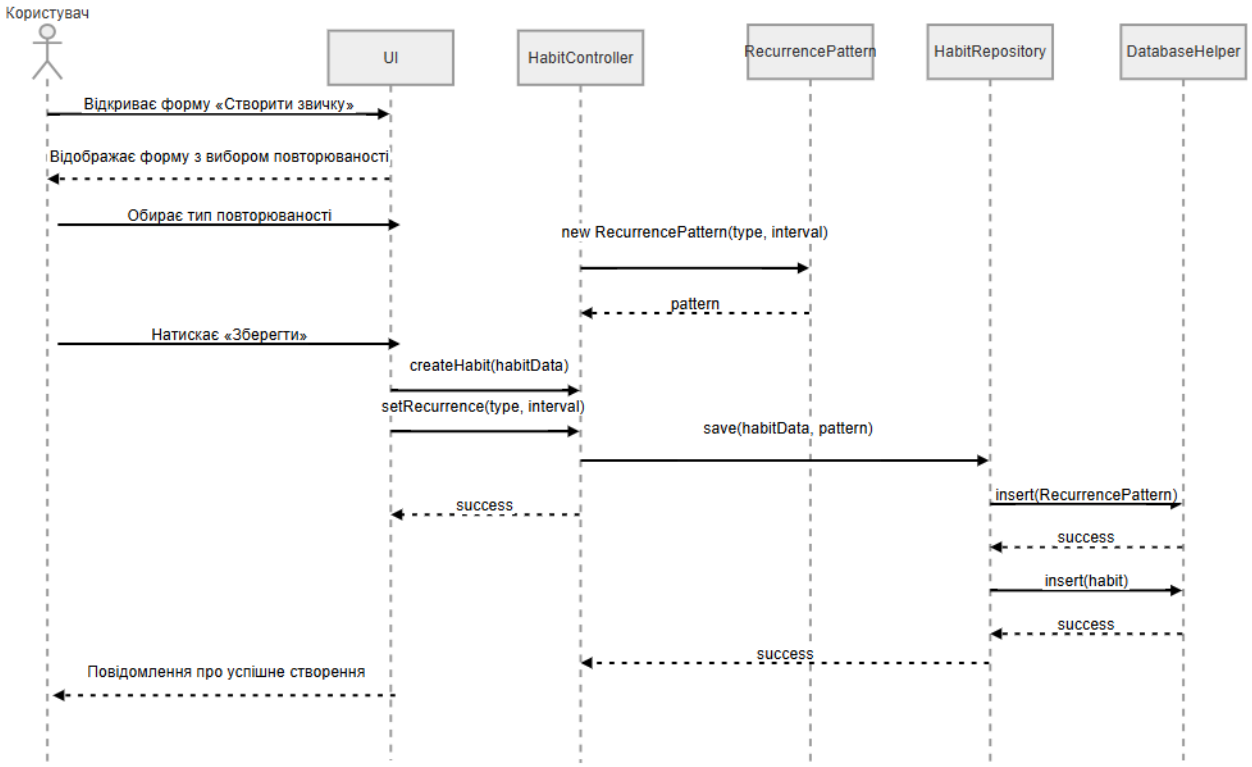

Рис. 4.2. UML-діаграма послідовностей для процесу створення звички з налаштуванням періодичності

Цей сценарій підкреслює важливість розділення логіки: контролер відповідає за координацію, об’єкт RecurrencePattern – за обробку періодичності, репозиторій – за роботу з базою даних.

На рисунку 4.3 наведено діаграму, що відображає процес зміни статусу завдання на «виконано». Основні етапи процесу:

- користувач у списку завдань натискає кнопку позначення виконання;
- UI надсилає контролеру запит markAsComplete(taskid).
- taskController викликає у taskRepository метод updateStatus(taskid, completed) для оновлення статусу.
- репозиторій надсилає sql-запит до dataBaseHelper на оновлення запису.
- dataBaseHelper повертає відповідь про успішне виконання.
- якщо завдання має правило повторюваності, taskController визначає наступну дату виконання через calculateNextDate().
- встановлена нова дата передається до taskrepository для оновлення.

- після успішного оновлення інтерфейс отримує сигнал taskUpdated і відображає зміни.

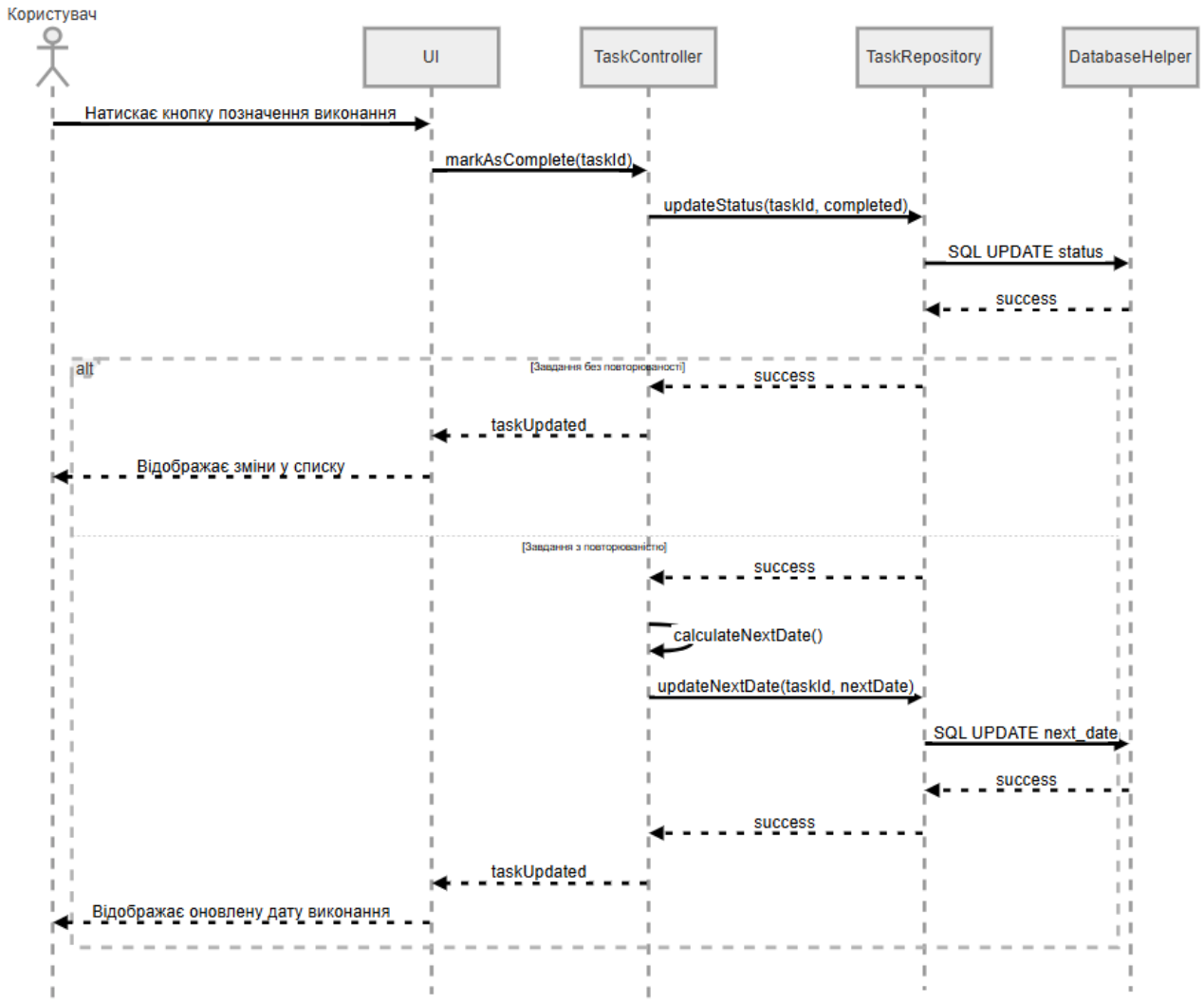

Рис. 4.3. UML-діаграма послідовностей для процесу позначення завдання як виконаного

Наведена діаграма демонструє дві логічні гілки: просте завершення одноразового завдання та оновлення дати наступного виконання для повторюваних завдань.

Для реалізації функціоналу застосунку розроблено його архітектуру, фрагмент якої відображено на діаграмі класів (рис. 4.4).

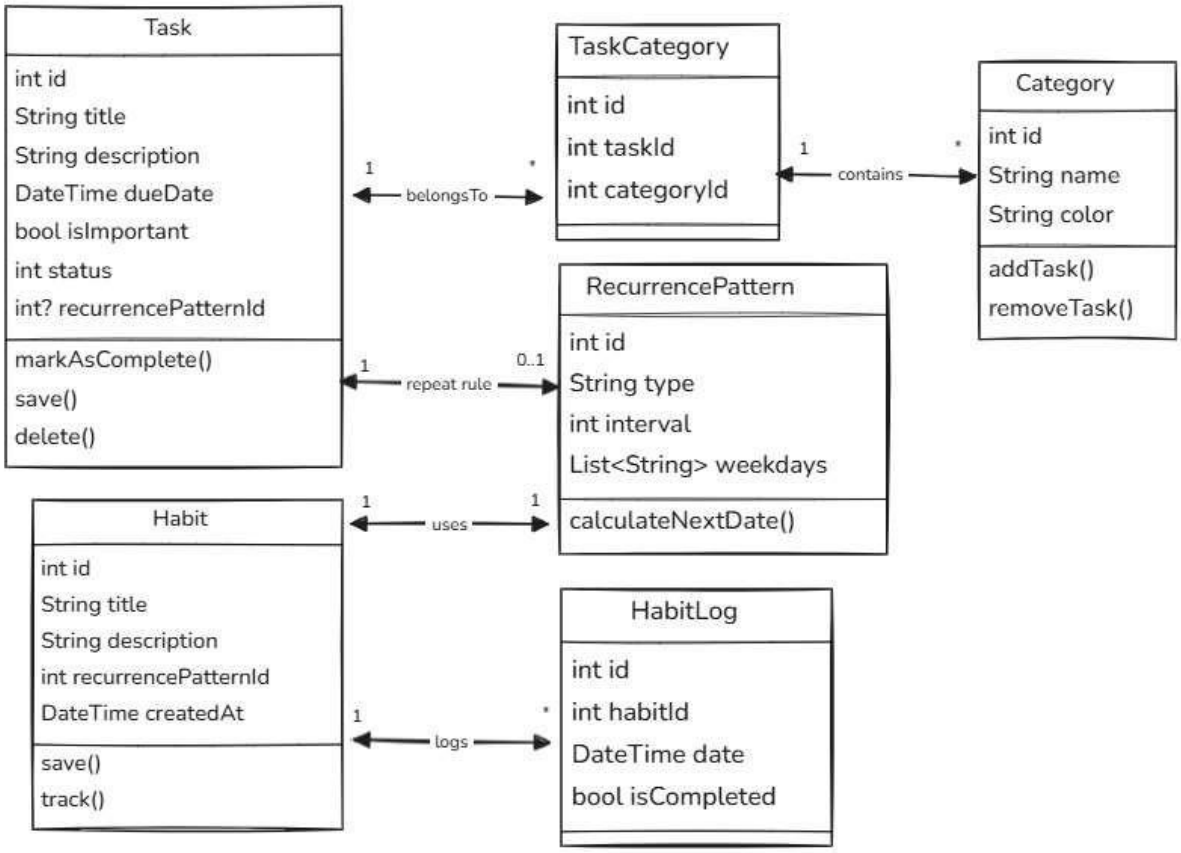

Рис. 4.4. Частина діаграми класів додатку

Фрагмент діаграми класів відображає структуру основних сутностей застосунку та зв'язки між ними, зокрема:

- різні типи сутностей — завдання, категорії та звички;
- механізм повторюваності, реалізований через клас RecurrencePattern;
- журнал виконання звичок;
- зв'язки між завданнями та категоріями.

До діаграми включено класи, що описують завдання, звички, правила повторюваності, категорії та допоміжні таблиці. Діаграма відображає логіку організації даних, що зберігаються у базі SQLite.

4.3. Діаграма діяльності (Activity Diagram)

Для застосунку персонального планування завдань розроблено дві ключові діаграми діяльності, що моделюють найбільш складні та значущі процеси системи.

На рис. 4.5 зображено діаграму, яка відтворює повний процес створення завдання від моменту натискання користувачем кнопки «Додати завдання» до успішного збереження запису в базі даних. Цей процес є центральним у функціонуванні застосунку і демонструє підтримку трьох різних типів завдань.

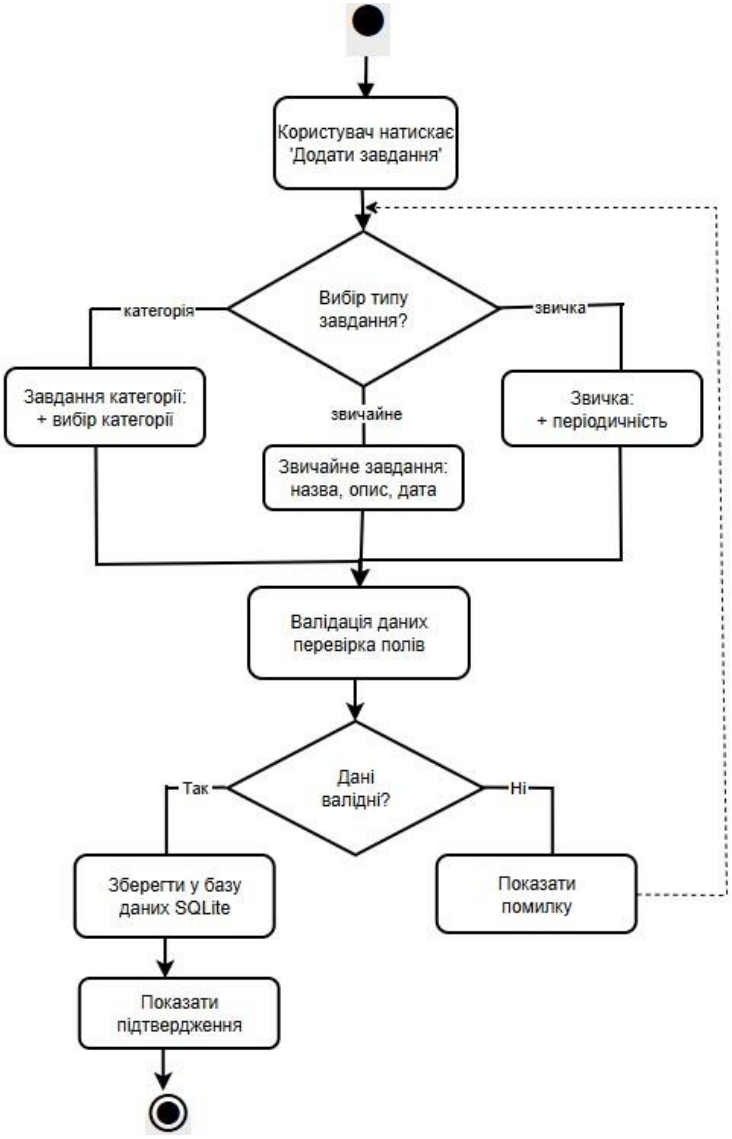

Рис. 4.5. UML Activity Diagram для процесу створення завдання.

Процес створення завдання розпочинається з початкового стану, коли користувач ініціює додавання нового запису. Першою точкою прийняття рішення є вибір типу завдання: ремайндер, список справ або звичка. Залежно від обраного типу система відображає відповідний екран із необхідним набором полів для заповнення.

Після введення даних виконується їх валідація. У разі виявлення помилок користувач отримує відповідне повідомлення та може внести корективи без

втрати вже введеної інформації. При успішному проходженні перевірки запис зберігається до бази даних, а система відображає візуальне підтвердження створення завдання. Процес завершується кінцевим станом. Діаграма наочно демонструє розгалужену логіку обробки різних типів завдань, механізм валідації з можливістю повторного введення даних та послідовність кроків від ініціації до завершення операції.

Наступна діаграма (рис. 4.6) моделює процес позначення звички як виконаної, що включає логіку підтримки серії виконань та автоматичний розрахунок наступної дати. Цей процес є специфічним для звичок і демонструє принципову відмінність їхньої поведінки порівняно з одноразовими завданнями.

Процес розпочинається, коли користувач позначає звичку як виконану через відповідну кнопку або чекбокс у списку. Система перевіряє історію попередніх виконань: якщо попереднє виконання відбулося у межах очікуваного інтервалу – серія продовжується та лічильник збільшується, інакше серія обнуляється і розпочинається заново. Після оновлення лічильника система автоматично розраховує наступну дату виконання на основі налаштованого режиму повторення та оновлює відповідний запис у базі даних. Звичка залишається активною і відображається у списку з оновленою датою наступного виконання.

Діаграма відображає складну бізнес-логіку, яка обробляється системою автоматично без втручання користувача, а також демонструє важливість коректного відстеження стану об'єктів у часі та умовної логіки, що залежить від попередньої історії виконання.

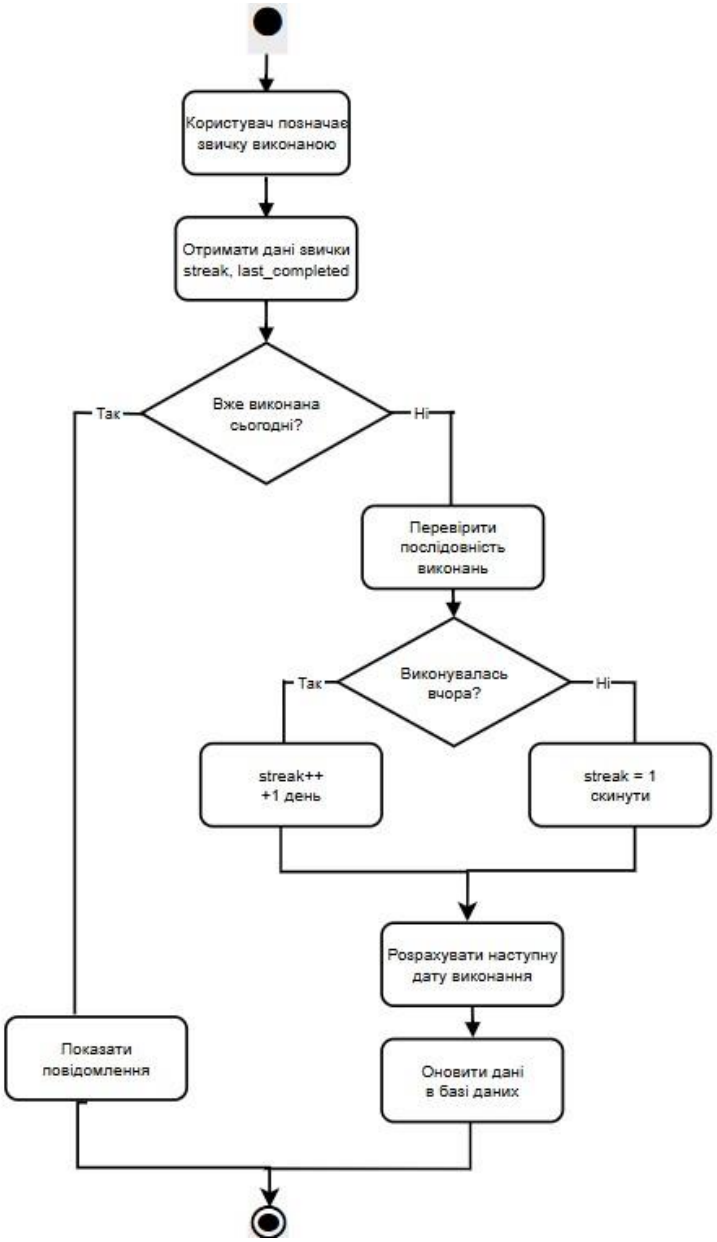

Рис. 4.6. UML Activity Diagram для процесу позначення звички як виконаної.

Розроблені діаграми діяльності надають чітке розуміння складних процесів у системі персонального планування завдань.

Розділ 5. Опис функціоналу розробленого застосунку

5.1. Авторизація в системі

Програмне рішення розроблено з використанням фреймворку Flutter та мови програмування Dart. Для зберігання даних застосовано локальну базу даних SQLite, що розміщується безпосередньо на пристрої користувача. В основу архітектури застосунку покладено патерн MVVM (Model-View-ViewModel), який забезпечує чітке розмежування між бізнес-логікою та рівнем представлення інтерфейсу.

Авторизація є першим екраном, з яким стикається користувач при кожному запуску застосунку. Реалізована система автентифікації забезпечує захист персональних даних від несанкціонованого доступу та підтримує два незалежні режими входу: стандартний вхід через електронну пошту і пароль та біометричну автентифікацію за відбитком пальця.

Екран входу (рис. 5.1а) містить два текстових поля для введення електронної адреси та пароля.

Поле пароля оснащене кнопкою перемикання видимості символів, що підвищує зручність введення. Нижче розташовані кнопки «Вхід» та «Реєстрація». Якщо користувач попередньо налаштував біометричну автентифікацію, на екрані додатково відображається відповідний індикатор та кнопка «Увійти біометрією».

Стандартна автентифікація передбачає введення електронної адреси та пароля, що були вказані під час реєстрації. Усі облікові дані зберігаються виключно локально у базі даних SQLite на пристрої користувача. Жодна інформація не передається на зовнішні сервери, що повністю виключає ризики, пов'язані з витоком даних через мережу.

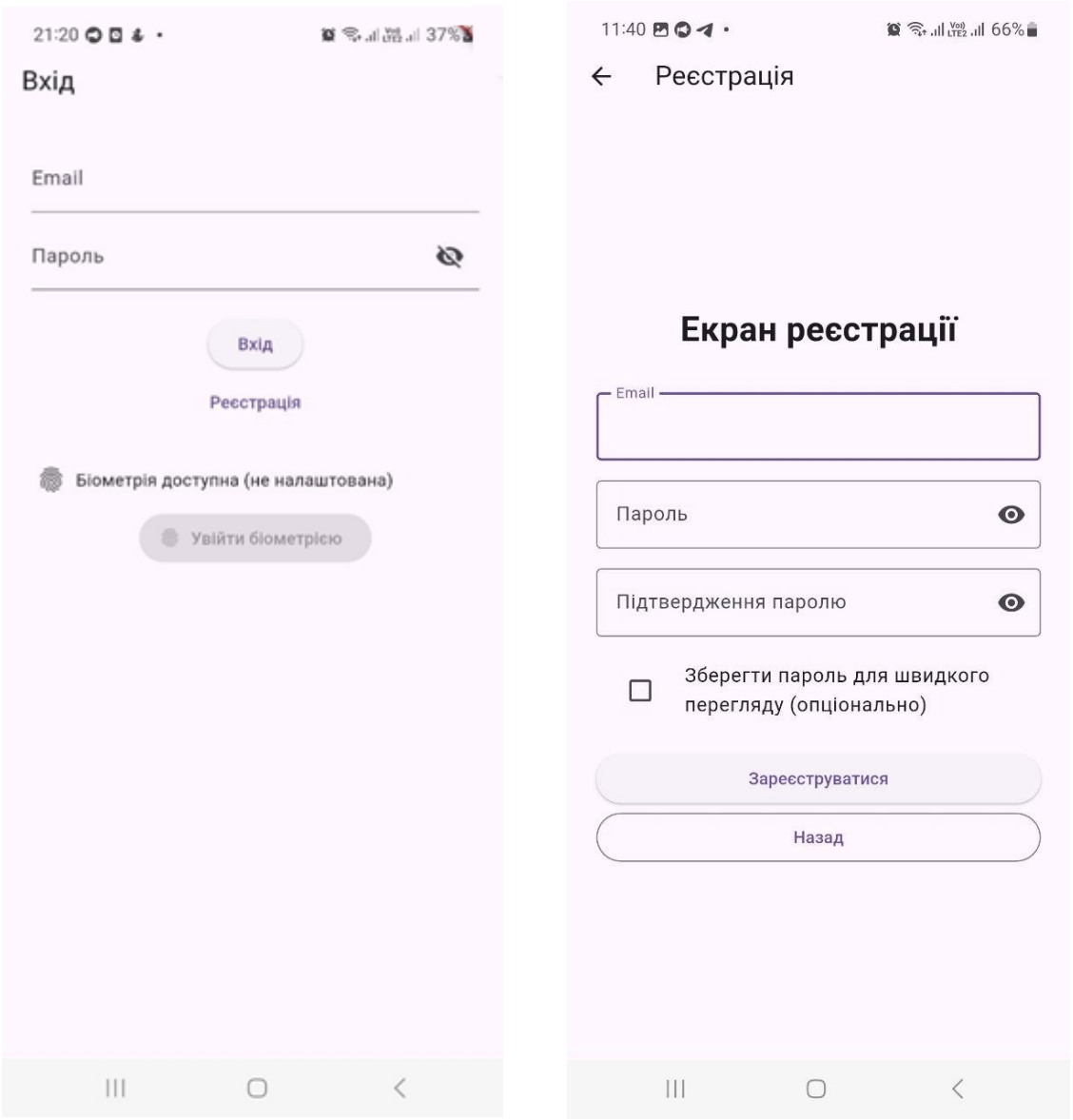

Рис. 5.1. Екран входу з підтримкою біометричної автентифікації та екран реєстрації в застосунку

Біометрична автентифікація реалізована через системний API пристрою з підтримкою сканера відбитка пальця. Цей режим доступний після первинного налаштування у розділі параметрів застосунку. При повторних запусках система автоматично пропонує швидкий вхід за відбитком, що усуває необхідність щоразу вводити пароль вручну. У разі невдалої біометричної перевірки або відмови користувача від цього методу завжди доступний резервний вхід через електронну пошту та пароль.

Реєстрація нового облікового запису відбувається на окремому екрані (рис. 5.1.б) , де користувач вказує електронну адресу та обирає пароль. Після успішної реєстрації користувач автоматично потрапляє до застосунку без необхідності

повторного входу. При першому запуску після реєстрації система пропонує налаштувати біометричну автентифікацію для подальшого зручного входу.

5.2. Головний екран та навігація застосунком

Після успішної авторизації користувач потрапляє на головний екран застосунку (рис. 5.2), який є центральним елементом інтерфейсу та забезпечує швидкий доступ до всіх типів завдань. Головний екран побудований за принципом мінімалізму: вся необхідна інформація відображається компактно, без зайвих елементів керування

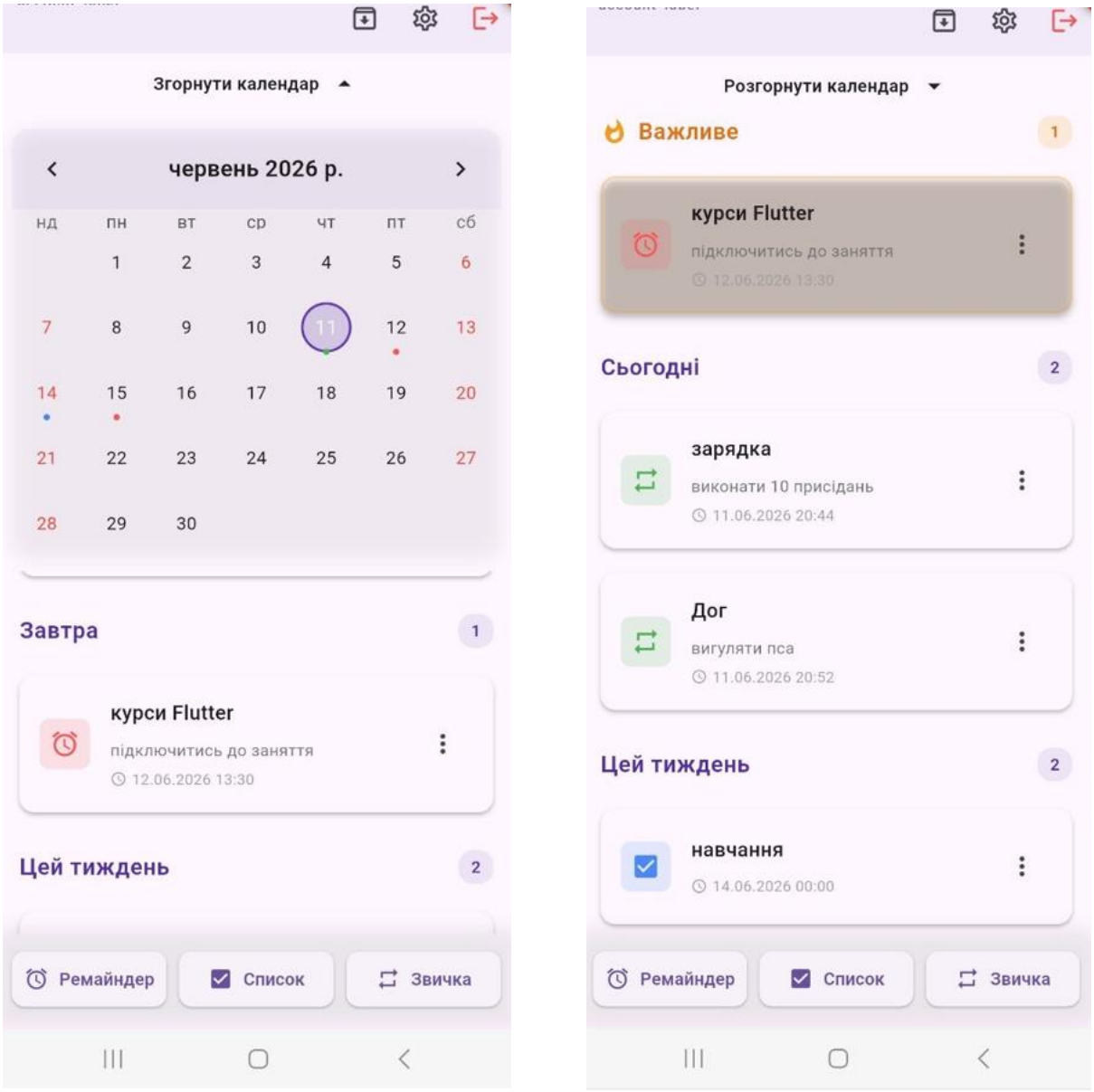

Рис. 5.2. Головне вікно

У верхній частині екрана розташований вбудований календар, який дозволяє переглядати завдання з прив'язкою до конкретних дат. Календар підтримує режим згортання – при натисканні на кнопку «Згорнути календар» він приховується, звільняючи більше простору для списку завдань (рис. 5.2б). Це особливо зручно при великій кількості записів. Поточна дата виділяється візуально, що дозволяє користувачу одразу орієнтуватися у часовому контексті своїх завдань.

Над календарем розташовані кнопки налаштування застосунку та архів виконаних завдань (рис.5.3). Користувач має змогу очистити архів, видаливши запис про виконане завдання назавжди, або повернути завдання, зробивши його знову активним.

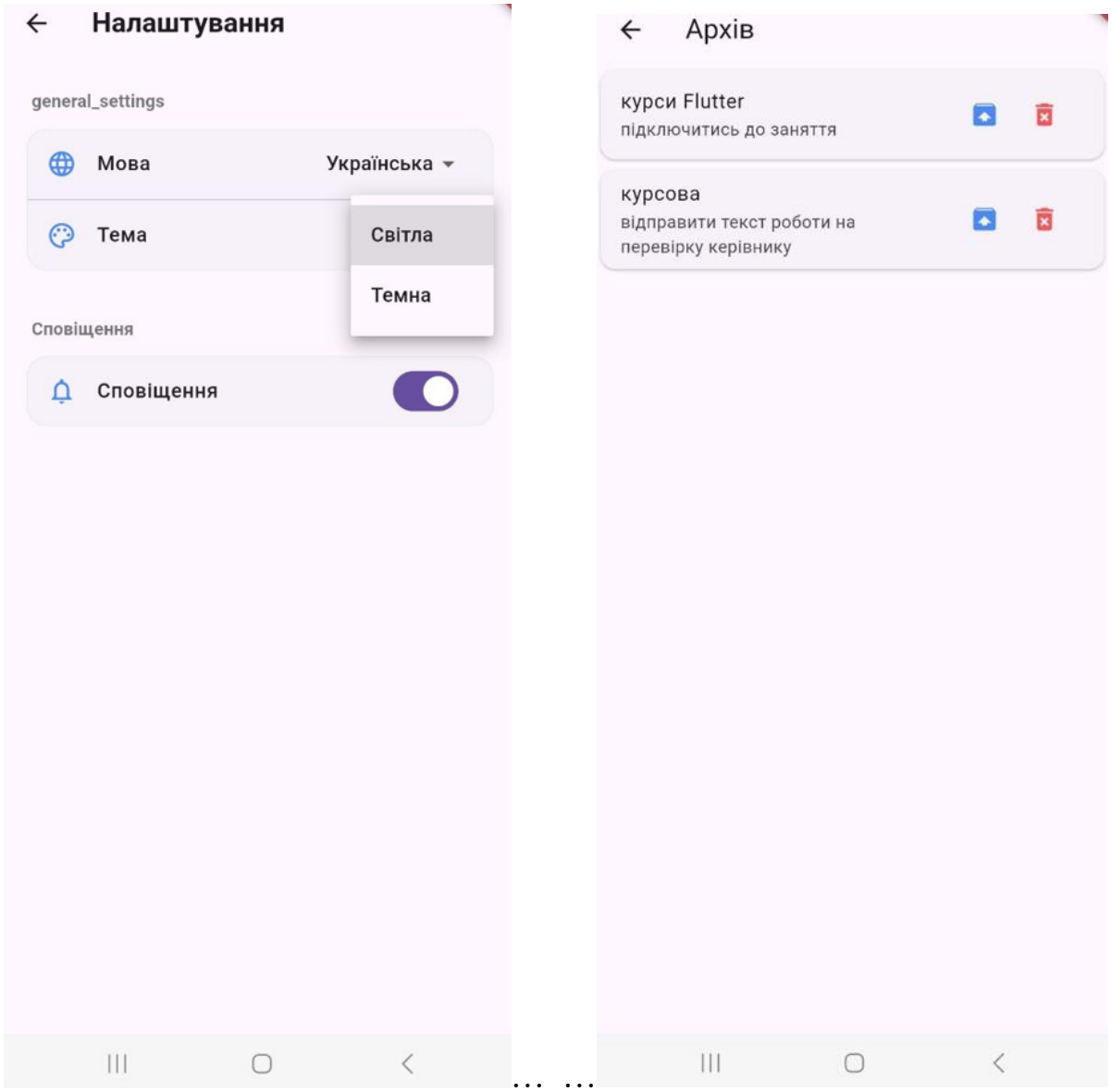

Рис. 5.3. Вікно налаштувань застосунку та архів

Нижче календаря відображається секція «Важливе» з лічильником пріоритетних завдань. У цій секції виводяться картки завдань, позначених як важливі, незалежно від їх типу. Кожна картка містить іконку типу завдання, назву, текст опису (за наявності), дату та час виконання. Контекстне мені картки конкретного завдання дає змогу встановити індикатор важливості змінити його статус – позначити завдання як виконане (рис.5.4).

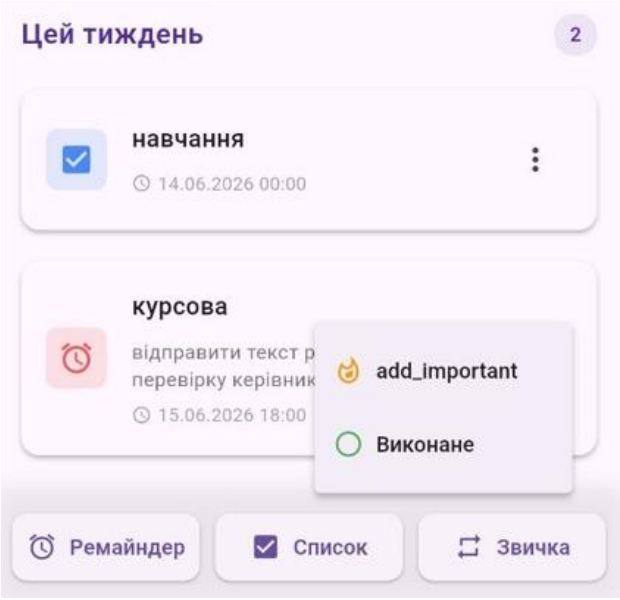

Рис. 5.4. Зміна статусу завдання

У нижній частині екрана розміщена навігаційна панель з трьома вкладками: «Ремайндер», «Список» та «Звичка». Така структура навігації дозволяє чітко розмежовувати різні типи завдань і швидко між ними переходити.

5.3. Робота з ремайндерами та списками справ

Застосунок підтримує два типи завдань з фіксованою структурою даних: ремайндери та списки справ. Кожен тип має власний екран створення та окрему логіку зберігання у базі даних SQLite.

Ремайндери

Ремайндер є найпростішим типом завдання – одноразове нагадування з прив'язкою до конкретного моменту часу. Екран створення ремайндера (рис. 5.5а) містить три елементи введення: поле заголовку, поле тексту нагадування та

вибір часу спрацювання. Вибір часу реалізований через системний засіб вибору дати і часу, що забезпечує звичний для користувача інтерфейс.

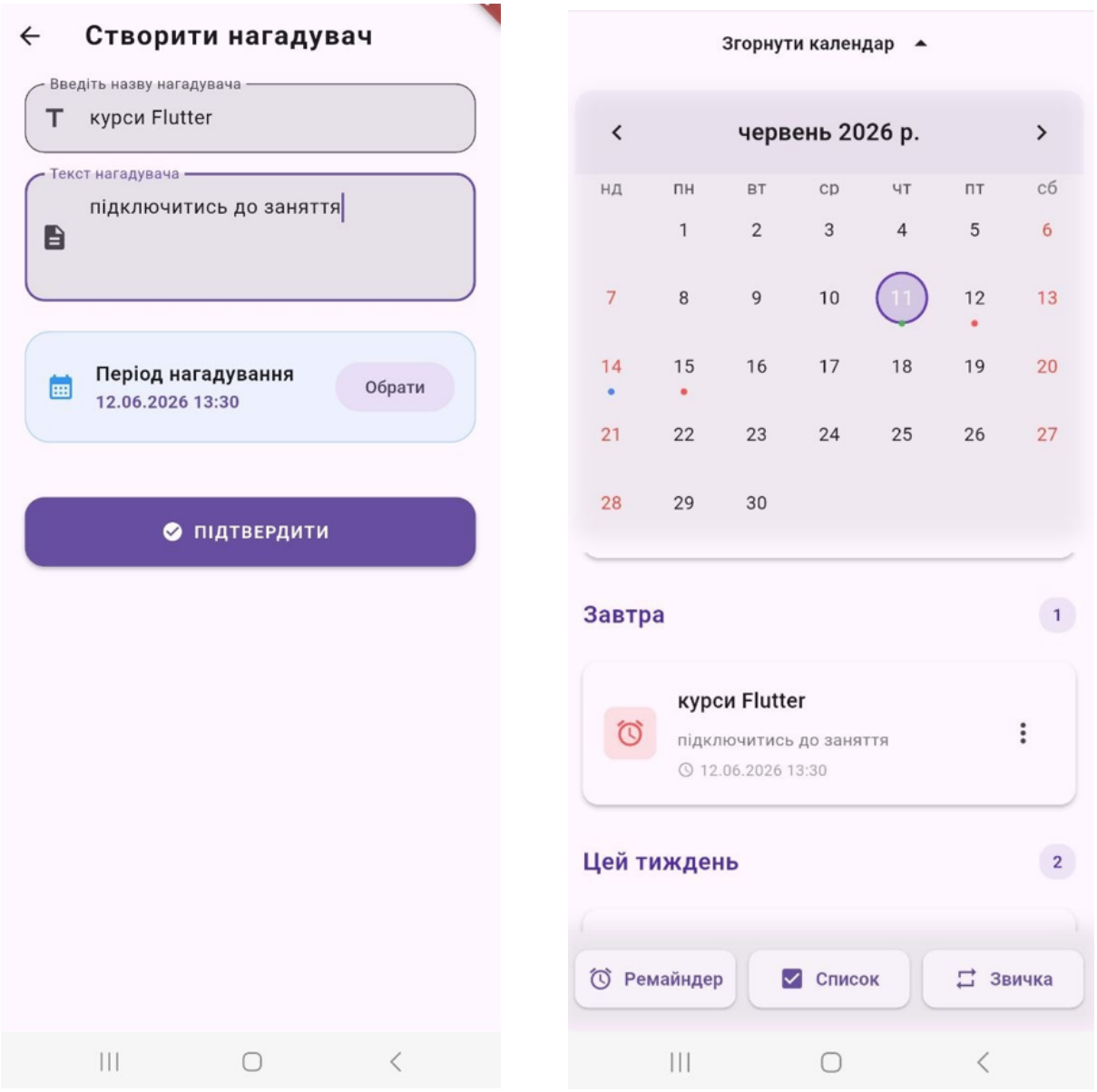

Рис. 5.5. Екран створення нового ремайндера

Після заповнення необхідних полів користувач підтверджує створення, після чого ремайндер з'являється у загальному списку на головному екрані (рис. 5.5 б). Картка ремайндера у списку відображає заголовок, текст нагадування та встановлений час спрацювання. Іконка будильника позначає тип завдання.

Користувач може позначити ремайндер як важливий – у такому разі він відображатиметься у секції «Важливе» на головному екрані та виділятиметься

індикатором пріоритету. Виконаний ремайндер переходить до архіву і більше не відображається в активному списку.

Списки справ

Список справ є більш складним типом завдання, що складається з набору окремих пунктів, кожен з яких може бути позначений як виконаний незалежно від інших. Екран створення списку (рис. 5.6) містить поле для введення назви списку та область для додавання пунктів.

Спочатку область порожня і відображає підказку «Додайте пункти до вашого списку». Додавання пунктів відбувається через кнопку «+ Додати пункт» у нижній частині екрана. Передбачена можливість встановити кінцевий термін виконання завдання як для цілого списку так і до кожного пункту чи підпункту (рис. 5.6 а).

Кожен доданий пункт відображається як окрема картка з текстовим полем для введення змісту, кнопкою додаткових дій та кнопкою видалення. Така структура дозволяє динамічно формувати список безпосередньо на екрані створення, редагувати або видаляти окремі пункти до збереження. Після завершення формування списку користувач натискає кнопку «Зберегти».

У збереженому списку користувач може відмічати окремі пункти як виконані, що дозволяє відстежувати прогрес виконання списку поступово. Список вважається повністю виконаним, коли всі його пункти позначені. Виконання всіх підпунктів для деякого елементу списку також автоматично позначає відповідний пункт як виконаний.

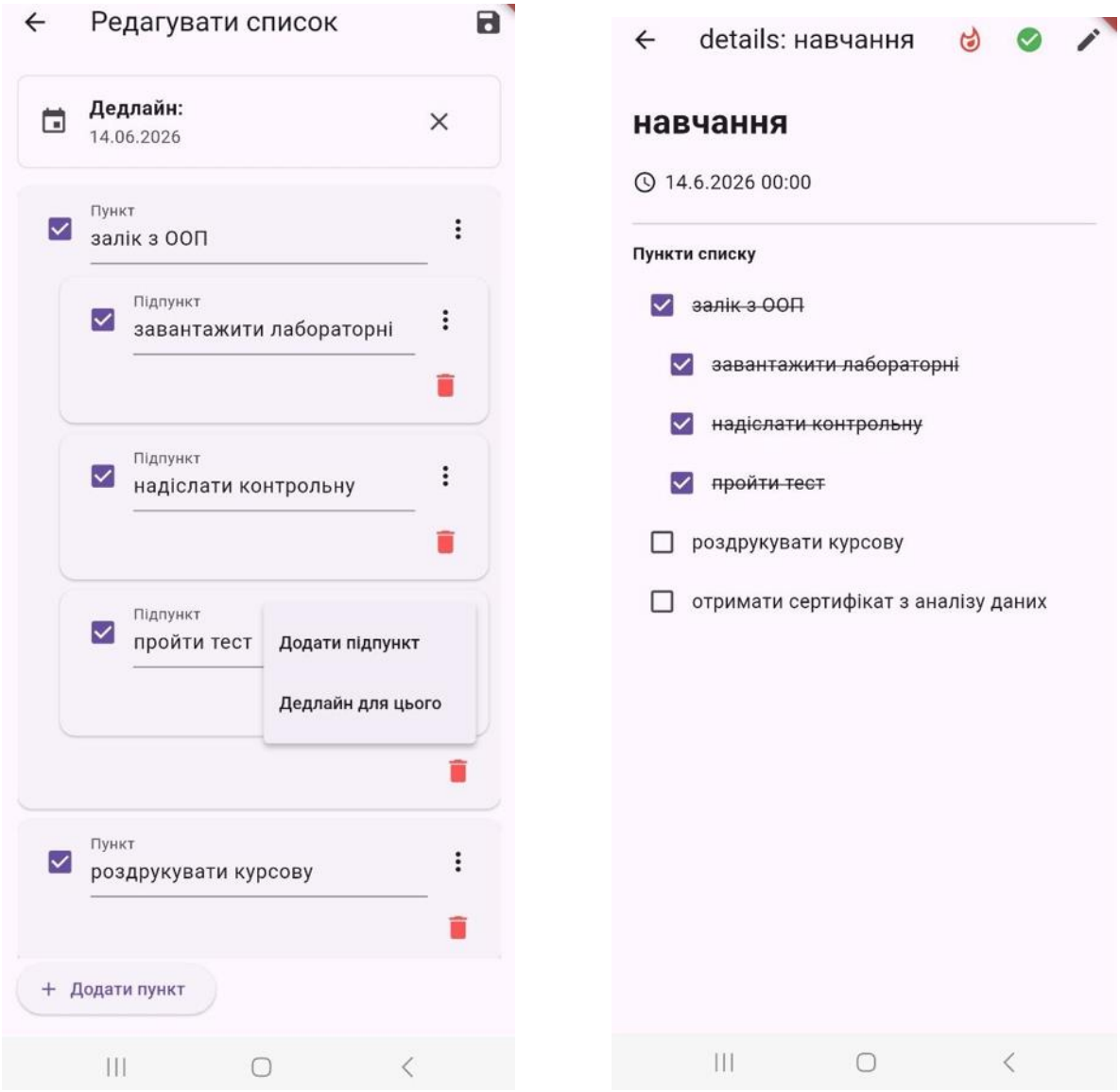

Рис. 5.6 Екран створення нового списку справ

Зв’язок між таблицею списків та таблицею пунктів забезпечується зовнішніми ключами з каскадним видаленням: при видаленні списку автоматично видаляються усі його пункти.

5.4. Робота зі звичками

Звичка є окремим типом завдання, що відрізняється від ремайндерів та списків справ циклічною природою виконання. На відміну від одноразових завдань, звичка після підтвердження виконання не переходить до архіву, а автоматично оновлюється з новою датою наступного виконання. Така логіка дозволяє користувачу формувати стійкі поведінкові патерни та відстежувати регулярність їх дотримання.

Екран створення звички (рис. 5.7) містить поле для введення назви, поле опису, вибір часу нагадування та блок налаштування режиму повторення. Режим повторення є ключовою відмінністю звички від інших типів завдань і підтримує два варіанти налаштування.

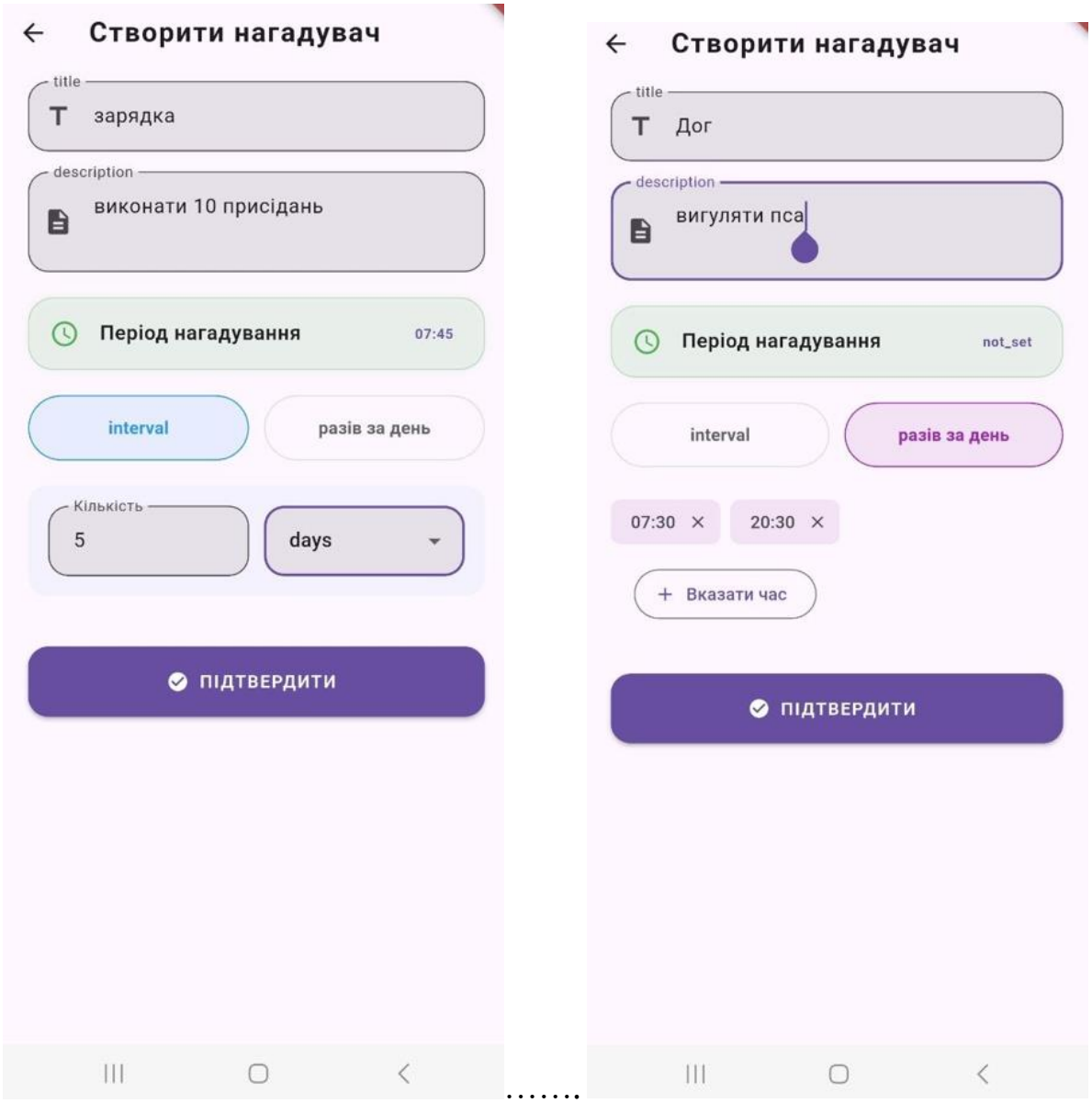

Рис. 5.7. Екран створення звички

Перший варіант – інтервальний режим повторення, при якому користувач вказує числовий інтервал та одиницю вимірювання: дні, тижні або місяці. Наприклад, значення «1 день» означає щоденне виконання, «2 тижні» – виконання кожні два тижні (рис.5.7а). Цей режим обрано за замовчуванням при відкритті екрана створення.

Другий варіант – режим повторення певну кількість разів на день. Він призначений для звичок, які мають виконуватися кілька разів протягом одного дня, наприклад прийом ліків або фізичні вправи (рис. 5.7б). Користувач перемикається між режимами через відповідні перемикачі.

Після збереження звичка з'являється у відповідній вкладці на головному екрані. Система автоматично розраховує наступну дату виконання на основі обраного режиму повторення та фіксує її у базі даних. Кожне підтвердження виконання оновлює дату наступного циклу та збільшує лічильник серій виконань. Журнал виконань зберігається в окремій таблиці бази даних, пов'язаній зі звичкою через зовнішній ключ з каскадним видаленням.

5.3. Тестування розробленого програмного забезпечення

Для перевірки коректності функціонування розробленого застосунку, виявлення потенційних дефектів та підтвердження його відповідності сформульованим технічним вимогам було проведено етап комплексного тестування. Стратегія тестування базувалася на методах функціонального та інтеграційного тестування у формі ручної перевірки сценаріїв використання.

Тестування застосунку проводилось на реальному пристрої під управлінням Android. У ході ручного тестування перевірено основні сценарії взаємодії: створення, редагування та видалення завдань усіх типів, коректність відображення статусів, роботу біометричної автентифікації, відображення завдань у календарі. Застосунок демонструє стабільну роботу та час запуску в межах 1–2 секунд.

Окремо перевірено коректність логіки повторюваності звичок: система правильно розраховує наступну дату виконання для всіх підтримуваних режимів повторення та коректно оновлює лічильник серій. Також підтверджено цілісність даних при видаленні: каскадне видалення пунктів списку та журналу виконань звичок працює відповідно до визначених зовнішніх ключів бази даних.Усі реалізовані функціональні можливості відповідають вимогам, сформульованим у третьому розділі роботи.

ВИСНОВОК

У ході виконання роботи здійснено комплексне проектування та практично реалізовано мобільний застосунок для персонального планування завдань.

Аналіз предметної області показав актуальність розроблення спеціалізованих інструментів для управління завданнями, що підтверджується популярністю існуючих рішень та потребами користувачів у ефективному плануванні. Дослідження аналогів, таких як Todoist, Microsoft To Do, TickTick, Any.do та Google Tasks, виявило як сильні сторони існуючих застосунків, так і їхні недоліки, зокрема надмірну складність інтерфейсу, залежність від інтернет-з'єднання та проблеми з приватністю даних.

В роботі обґрунтовано вибір технологій для майбутнього розроблення застосунку. Flutter як крос-платформний фреймворк забезпечує високу продуктивність та єдину кодову базу для різних операційних систем. SQLite як локальна база даних гарантує швидкодію роботи застосунку та приватність користувацьких даних.

Розроблено повний комплект UML-діаграм, що охоплюють всі аспекти системи. Діаграма прецедентів визначає функціональні вимоги до застосунку та описує основні сценарії використання. Діаграма класів описує структуру даних та взаємозв'язки між основними компонентами системи. Діаграми послідовності деталізують взаємодію компонентів у різних сценаріях роботи застосунку. Діаграми станів та діяльності моделюють поведінку системи та процеси обробки завдань різних типів. Діаграма компонентів визначає архітектурну структуру застосунку та зв'язки між його модулями.

Програмне рішення реалізовано засобами Flutter/Dart із локальним зберіганням даних у SQLite. Застосунок побудовано за архітектурою MVVM (Model-View-ViewModel), що забезпечує розділення логіки та інтерфейсу користувача. Локальне зберігання даних гарантує повну приватність: жодна персональна інформація користувача не передається на зовнішні сервери.

Розроблений мобільний застосунок є повністю автономним рішенням для персонального планування завдань, що не потребує підключення до інтернету та реєстрації зовнішніх облікових записів. Реалізація на Flutter забезпечує кросплатформний потенціал продукту, а локальне зберігання даних у SQLite – приватність і швидкодію. Подальший розвиток проєкту може включати деталізацію архітектури системи та реалізацію додаткових функціональних модулів.

ВИКОРИСТАНІ ДЖЕРЕЛА

4. Аллен Д. Як привести справи в порядок: мистецтво продуктивності без стресу / Д. Аллен ; пер. з англ. – Київ : Наш формат, 2015. – 352 с.
5. Коросташовець А. Матриця Ейзенхауера: Опановуємо пріоритезацію завдань для управління часом [Електронний ресурс]. – URL: https://worksection.com/ua/blog/the-eisenhower-matrix.html – Назва з екрану
6. Васильченко С. Що таке матриця Ейзенхауера та як її використовувати у роботі та житті. [Електронний ресурс]. – URL: https://happymonday.ua/matrytsya-ejzenhauera
7. Todoist. Official Website [Електронний ресурс]. – URL: https://todoist.com – Назва з екрана.
8. Зосим М. Метод GTD (Getting Things Done) [Електронний ресурс]. – URL: https://todoist.com – Назва з екрана.
9. Microsoft To Do. Official Website [Електронний ресурс]. – URL: https://to-do.microsoft.com – Назва з екрана.
10. TickTick. Official Website [Електронний ресурс]. – URL: https://ticktick.com – Назва з екрана.
11. Millie Pham. Honest Todoist Review of 2024 [Електронний ресурс]. – URL: https://bymilliepham.com/todoist-review– Назва з екрана.
12. Any.do. Official Website [Електронний ресурс]. – URL: https://www.any.do — Назва з екрана.
13. Документація Flutter – URL: https://docs.flutter.dev/
14. Мова Dart. Документація – URL: https://dart.dev/docs
15. Рейтинг мов програмування 2025. Опитування DOU – URL: https://dou.ua/lenta/articles/language-rating-2025/
16. SQLite. Documentation. – URL: https://www.sqlite.org/docs.html.

17. Flutter sqflite package [Електронний ресурс]. – URL: https://pub.dev/packages/sqflite – Назва з екрана.
18. Provider package for Flutter [Електронний ресурс]. – URL: https://pub.dev/packages/provider – Назва з екрана.
19. UML 2.5.1 Specification [Електронний ресурс] / Object Management Group. – URL: https://www.omg.org/spec/UML/2.5.1/ – Назва з екрана.
20. Majchrzak T. A. Comparing Cross-Platform Development Approaches for Mobile Applications / T. A. Majchrzak, A. Biørn-Hansen, T.-M. Grønli // Proceedings of the 8th International Conference on Web Information Systems and Technologies. – 2018. – P. 120–131.