УДК 004.42:378.147:005.7
DOI https://doi.org/10.36994/2788-5518-2025-02-10-06

ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ ПЛАНУВАННЯ ЧАСУ НА ОСНОВІ КРОСПЛАТФОРМНИХ ТЕХНОЛОГІЙ

***Ващіліна О. В.**, к. ф.-м. н., доц., Київський національний університет імені Тараса Шевченка, Київ, Україна. https://orcid.org/0000-0001-6867-6216, olenavashchilina@knu.ua*

***Голотюк В. М**., бакалавр, Київський національний університет імені Тараса Шевченка, Київ, Україна. https://orcid.org/0009-0008-3947-4613, viktoriia_holotiuk@knu.ua*

***Анотація.** У статті розглянуто процес проєктування навчально орієнтованої програмної системи для автономного планування часу, що відповідає сучасним вимогам кросплатформності, адаптивності та конфіденційності. Актуальність дослідження зумовлена потребою в гнучких цифрових інструментах тайм-менеджменту, які можуть бути модифіковані відповідно до індивідуальних потреб користувача. Попри наявність широкого спектра готових рішень, зокрема Todoist, Microsoft To Do та Any.do, актуальним залишається завдання створення відкритої архітектури, орієнтованої на освітнє середовище та практичне засвоєння принципів проєктування програмного забезпечення. У межах дослідження здійснено порівняльний аналіз згаданих цифрових рішень, сформульовано функціональні та нефункціональні вимоги до нової програмної системи, а також обґрунтовано вибір архітектурного шаблону MVVM з урахуванням принципів clean architecture, що сприяють чіткому розмежуванню функцій між компонентами. Побудовано низку UML-діаграм, що відображають функціональну та структурну модель системи: діаграма варіантів використання окреслює основні сценарії взаємодії користувача, діаграма активностей демонструє логіку виконання дій, діаграма послідовності ілюструє часову взаємодію між компонентами, а діаграма класів формалізує структуру сутностей та їх зв'язки. Реалізація системи не входить до складу цієї роботи та розглядається як наступний етап, для якого запропоновано узагальнені рекомендації щодо вибору технологічного стеку, здатного забезпечити кросплатформну розробку, автономність та зручність впровадження. Результати роботи можуть бути використані в освітньому процесі для формування навичок архітектурного моделювання, системного мислення та підготовки до реалізації повноцінних програмних продуктів. Запропонована модель має потенціал для подальшого розширення функціоналу та інтеграції з хмарними сервісами.*

***Ключові слова:** тайм-менеджмент; проєктування програмної системи; архітектура MVVM; UML-діаграми; кросплатформність.*

DESIGN OF TIME MANAGEMENT SOFTWARE SYSTEM BASED ON CROSS-PLATFORM TECHNOLOGIES

***Olena Vashchilina**, Ph.D., Ass. Prof., Taras Shevchenko National University of Kyiv, Kyiv, Ukraine. https://orcid.org/0000-0001-6867-6216, olenavashchilina@knu.ua*

***Viktoriia Holotiuk**, Bachelor, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine. https://orcid.org/0009-0008-3947-4613, viktoriia_holotiuk@knu.ua*

***Abstract.** The article explores the design process of an educationally oriented software system for autonomous time planning that meets modern requirements of cross-platform compatibility, adaptability, and confidentiality. The relevance of the study is driven by the need for flexible digital time-management tools that can be modified to suit individual user needs. Despite the wide range of available solutions, in particular Todoist, Microsoft To Do, and Any.do, the task of creating an open architecture focused on educational environments and the practical assimilation of software design principles remains relevant. The study includes a comparative analysis of the aforementioned digital solutions, the formulation of functional and non-functional requirements for a new software system, and the justification for selecting the MVVM architectural pattern, taking into account the principles of clean architecture, which promote a clear separation of concerns among components. A set of UML diagrams was developed to represent the system's functional and structural model: the use case diagram outlines key user interaction scenarios; the activity diagram illustrates the logic of action execution; the sequence diagram shows the temporal interaction between components; and the class diagram formalizes the structure of entities and their relationships. System implementation is beyond the scope of this work and is considered a subsequent stage, for which general recommendations are provided regarding the choice of a technology stack capable of supporting cross-platform development, offline functionality, and ease of deployment. The results of the study can be used*

in the educational process to develop skills in architectural modeling, systems thinking, and preparation for the implementation of full-fledged software products. The proposed model has the potential for further functional expansion and integration with cloud services.

Key words: *time management; software system design; MVVM architecture; UML diagrams; cross-platform development.*

Вступ

У сучасних умовах цифрової трансформації та зростаючої багатозадачності ефективне управління часом набуває особливого значення як у професійній, так і в освітній діяльності. Тайм-менеджмент розглядається як ключовий компонент особистої продуктивності, що потребує гнучких, інтуїтивно зрозумілих та адаптивних інструментів [1]. Попри наявність широкого спектра готових програмних рішень, таких як Todoist, Microsoft To Do, Any.do [2–4], актуальним залишається завдання створення автономних та навчально орієнтованих систем, які можуть бути модифіковані відповідно до специфічних потреб користувача.

Згідно з сучасними підходами до підготовки фахівців у галузі інформаційних технологій, проєктування власного застосунку в межах студентського проєкту сприяє формуванню інженерного мислення, навичок архітектурного моделювання та практичного застосування сучасних технологій [5]. Такий підхід дозволяє не лише сформувати концепцію функціонального програмного продукту, а й забезпечити глибоке розуміння процесів аналізу вимог, побудови архітектури та логічної структури програмної системи.

У межах цього дослідження здійснено проєктування кросплатформної програмної системи тайм-менеджменту з урахуванням принципів адаптивності, автономності та масштабованості. Особливу увагу приділено формалізації функціональних і нефункціональних вимог, побудові UML-діаграм, обґрунтуванню архітектурних рішень та логічній структурі проєкту. Реалізація системи розглядається як наступний етап, для якого в роботі запропоновано рекомендації щодо вибору технологічного стеку (Flutter, Dart, SQLite), що забезпечує кросплатформність та автономну роботу.

Метою роботи є створення архітектурної моделі навчально орієнтованої програмної системи тайм-менеджменту, яка поєднує відкриту структуру, сучасні підходи до моделювання та можливість подальшої реалізації відповідно до індивідуальних потреб користувачів. Запропоноване рішення має потенціал для використання в освітньому середовищі, а також у побутовій та професійній діяльності як зручний інструмент для організації часу.

Виконання досліджень

Аналіз існуючих програмних рішень

У процесі дослідження було проведено аналіз функціональності трьох популярних програмних систем тайм-менеджменту: Todoist, Microsoft To Do та Any.do. Основна увага приділялася ключовим характеристикам, що впливають на зручність використання, гнучкість налаштувань, підтримку автономного режиму та потенціал для освітнього застосування.

Todoist − багатофункціональний застосунок із підтримкою категоризації, підзавдань, пріоритетів, нагадувань, повторюваних дедлайнів та понад 80 інтеграцій. Відзначається високою зручністю інтерфейсу та можливістю швидкого додавання завдань у природній мові [2].

Microsoft To Do − інтегрований у екосистему Microsoft 365, підтримує створення списків завдань, нагадування, групування списків, персоналізацію інтерфейсу та синхронізацію з Outlook. Водночас не передбачає ієрархії підзавдань, що обмежує гнучкість структурування [3].

Any.do − орієнтований на мобільне використання, підтримує голосові нотатки, групову роботу, інтеграцію з Google Календарем, нагадування, планування дня та спільні простори для сімей чи команд. Має простий інтерфейс і функціональність, адаптовану до повсякденного використання [4].

У таблиці 1 наведено порівняння функціональності розглянутих систем за ключовими характеристиками, зокрема: можливістю створення та організації завдань, підтримкою підзавдань, нагадуваннями, синхронізацією між платформами, інтеграцією з іншими сервісами, зручністю інтерфейсу, підтримкою групової роботи та орієнтацією на конкретні платформи.

Таблиця 1

Порівняння основних характеристик систем тайм-менеджменту

Функціонал	Todoist	Microsoft To Do	Any.do
Створення та організація завдань	+	+	+
Підтримка підзавдань	+	–	+
Нагадування та терміни	+	+	+
Синхронізація між платформами	+	+	+
Інтеграція з іншими сервісами	+	+	+
Зручність інтерфейсу	Висока	Середня	Висока
Групова робота	–	–	+

Усі розглянуті для порівняння програмні системи забезпечують базову функціональність для створення та організації завдань, включаючи підтримку нагадувань, термінів виконання та синхронізації між платформами. Водночас лише Todoist та Any.do підтримують створення підзавдань, що є важливою функцією для деталізації планів та побудови ієрархічної структури активностей. Any.do вирізняється додатковими можливостями, такими як голосові нотатки та групова взаємодія, що розширює сценарії використання у командному середовищі. Інтерфейси Todoist та Any.do оцінюються як більш зручні для мобільного використання, тоді як Microsoft To Do орієнтований на інтеграцію з екосистемою Microsoft 365.

Усі три системи мають обмежену підтримку автономного режиму, що стало одним із чинників на користь розроблення власного застосунку з локальним зберіганням даних, відкритою архітектурою та можливістю адаптації до освітніх потреб.

Формування вимог до програмної системи

На основі проведеного аналізу існуючих рішень сформульовано вимоги до програмної системи тайм-менеджменту, орієнтованої на освітнє застосування. Вимоги охоплюють як функціональні, так і нефункціональні аспекти, що забезпечують відповідність системи практичним потребам студентів, а також її придатність для демонстрації архітектурних принципів у навчальному процесі.

Функціональні вимоги:

До функціональних вимог належать базові можливості, необхідні для ефективного планування навчальної діяльності:

- Створення, редагування та видалення завдань − забезпечує повний життєвий цикл управління активностями.
- Групування завдань за категоріями − дозволяє структурувати завдання за темами, курсами або проєктами.
- Встановлення пріоритетів та дедлайнів − сприяє розстановці акцентів і контролю термінів виконання.
- Візуалізація графіка активностей − підвищує наочність планування, дозволяє оцінити навантаження в динаміці.
- Локальне зберігання даних без потреби в синхронізації − гарантує автономність роботи та конфіденційність інформації.
- Пошук та фільтрація завдань − забезпечує швидкий доступ до потрібної інформації при великій кількості записів.

Особливу увагу приділено локальному зберіганню даних, що дозволяє використовувати систему без підключення до мережі, зберігаючи при цьому контроль над персональною інформацією. Візуалізація графіка активностей має додаткову освітню цінність, оскільки дозволяє студентам краще оцінювати власну завантаженість та планувати навчальні блоки.

Нефункціональні вимоги:

Нефункціональні вимоги визначають загальні характеристики системи, що впливають на її придатність до щоденного використання:

- Кросплатформність − забезпечує доступність системи на різних пристроях, що відповідає сучасним звичкам користувачів.
- Автономність роботи без підключення до мережі − важлива для стабільної роботи в умовах обмеженого доступу до Інтернету.
- Мінімалістичний та адаптивний інтерфейс − сприяє зручності використання незалежно від розміру екрана та технічного рівня користувача.
- Швидкодія та стабільність − впливають на загальне враження від роботи системи та її ефективність.
- Можливість масштабування функціоналу − дозволяє розширювати систему в майбутньому без порушення її архітектурної цілісності.

Формування вимог здійснювалося з урахуванням специфіки освітнього середовища, потреб у мобільності, конфіденційності та можливості подальшого розвитку системи. Сформульовані вимоги стали основою для вибору архітектурного підходу, побудови логічної структури проєкту та визначення технологічного стеку, що забезпечує реалізацію системи відповідно до принципів відкритості, автономності та навчальної придатності.

Архітектурне проєктування системи

Архітектура програмної системи тайм-менеджменту сформована відповідно до вимог, визначених у попередньому розділі, з урахуванням принципів модульності, масштабованості та кросплатформності. Для забезпечення прозорості структури та поведінки системи використано низку UML-діаграм, які охоплюють як статичні, так і динамічні аспекти проєкту. Наведемо діаграми у порядку, що відповідає логіці розробки: від загальної архітектури до конкретних сценаріїв і структур. Спочатку представлено архітектурний шаблон MVVM, який визначає загальну організацію компонентів системи. Далі наведено діаграму варіантів використання, що окреслює функціональні можливості з точки зору користувача. Діаграма активностей деталізує поведінку системи у межах окремих сценаріїв, а діаграма

послідовності демонструє часову взаємодію між компонентами. Завершує розділ діаграма класів, яка відображає логічну структуру сутностей та їх зв'язків.

Вибір архітектурного шаблону. Для проєктування програмної системи тайм-менеджменту обрано архітектурний шаблон MVVM (Model–View–ViewModel), який забезпечує чітке розділення відповідальностей між компонентами, спрощує підтримку, тестування та масштабування застосунку. MVVM є загальновживаним підходом у кросплатформній розробці та рекомендований у офіційній документації Microsoft [6]. У межах даного проєкту реалізацію MVVM адаптовано до специфіки освітнього середовища:

- Model відповідає за структуру даних та взаємодію з локальним сховищем (SQLite), без використання зовнішніх API чи хмарних сервісів.
- ViewModel реалізує логіку обробки подій, фільтрації, оновлення інтерфейсу та навігації, не містить мережевої логіки, що спрощує тестування та підвищує автономність.
- View забезпечує мінімалістичний та адаптивний інтерфейс, орієнтований на студентське використання, з акцентом на візуалізацію графіка активностей.

Загальна структура шаблону MVVM реалізована відповідно до загальновизнаних принципів, описаних у [6], з урахуванням автономності, конфіденційності та освітньої придатності системи.

Логічна структура проєкту реалізована у вигляді модульного дерева файлів, що відповідає принципам clean architecture та забезпечує ізоляцію відповідальностей між компонентами MVVM.

Діаграма варіантів використання (Use Case Diagram). Для визначення основних сценаріїв взаємодії користувача з системою побудовано діаграму варіантів використання (рис. 1). Вона охоплює створення, редагування, фільтрацію завдань, перегляд графіка активностей, а також налаштування категорій. Інтерфейс користувача є спільним для ПК та Android, що забезпечує єдність досвіду та зручність використання незалежно від платформи.

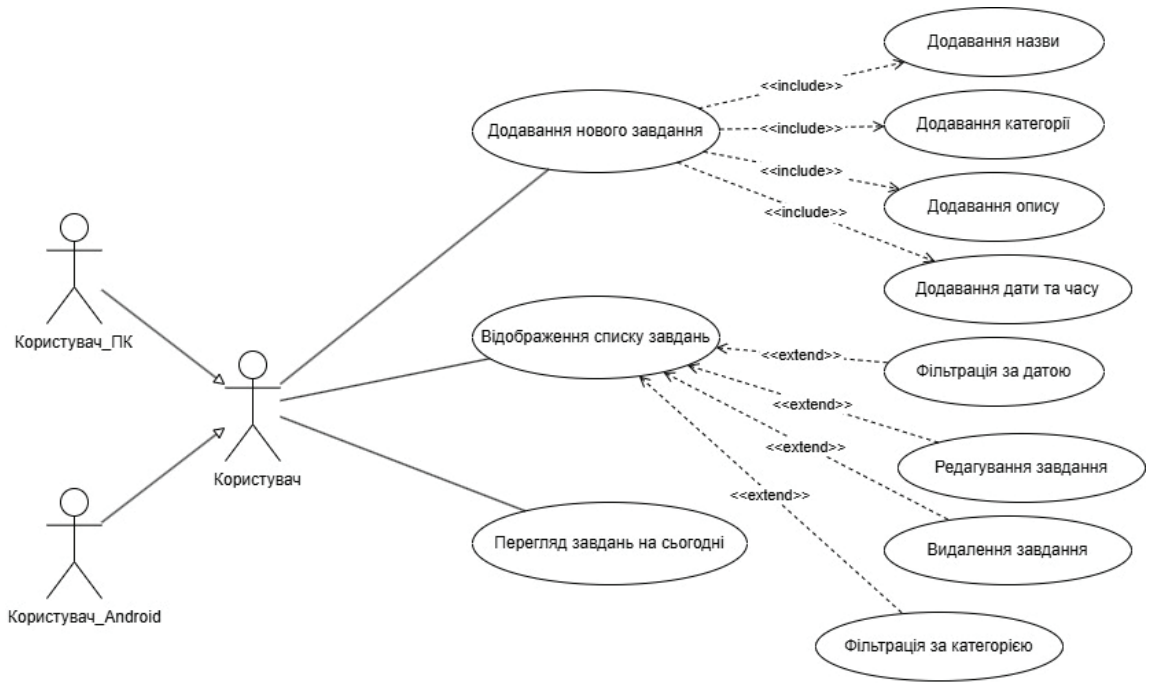
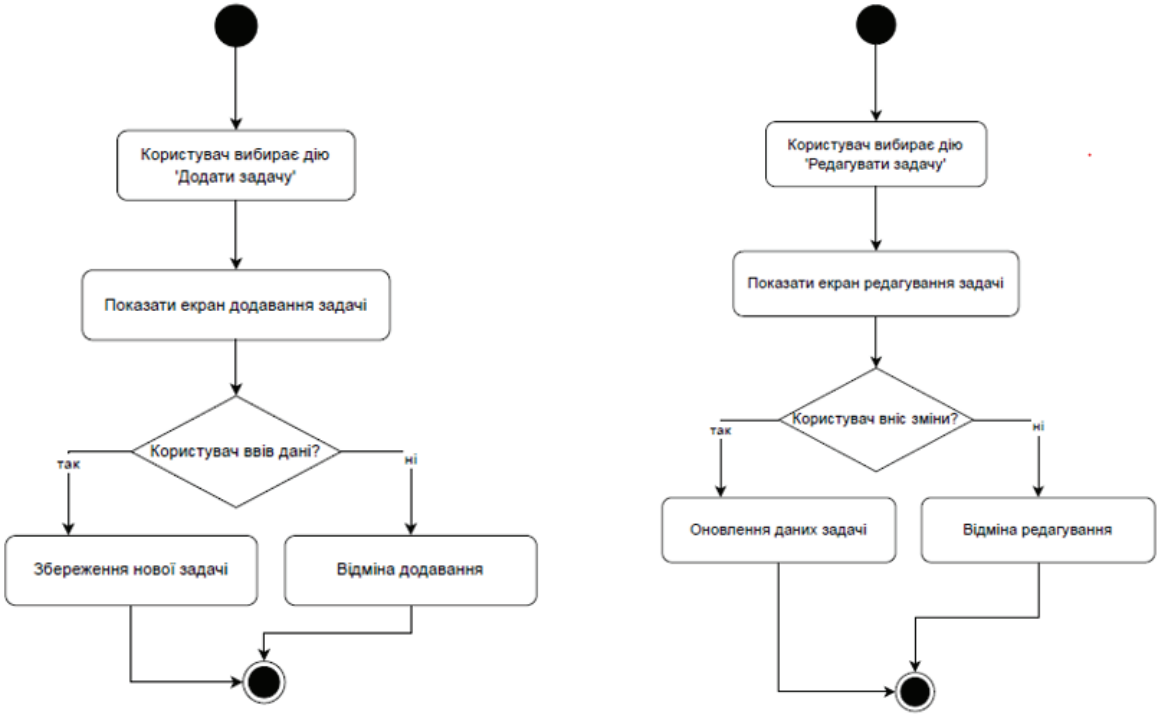


Рис. 1. Діаграма варіантів використання

Діаграма активностей (**Activity Diagram).** Діаграма активностей використовується для моделювання поведінки системи, зокрема послідовності дій користувача та реакцій інтерфейсу. Такий тип діаграми дозволяє візуалізувати логіку переходів між екранами, умови виконання дій, перевірку введених даних, збереження інформації у локальне сховище та оновлення інтерфейсу. Діаграма також демонструє можливі розгалуження сценаріїв залежно від вибору користувача, що сприяє точному формулюванню вимог до функціональності та поведінки програмної системи. На рис. 2 представлено два ключові сценарії взаємодії користувача із системою: додавання задачі (рис.2а) та редагування задачі (рис. 2б).



а) додавання задачі; б) редагування задачі

Рис. 2. Діаграма активностей для реалізації дій користувача

Діаграма послідовності (Sequence Diagrams). Для відображення динаміки взаємодії між компонентами системи під час виконання ключових сценаріїв побудовано діаграми послідовності, які ілюструють часову послідовність викликів методів між View, ViewModel, контролерами та сервісами збереження даних. Ці діаграми дозволяють простежити, як ініціатива користувача проходить через усі рівні архітектури, забезпечуючи узгодженість логіки та відповідність шаблону MVVM. На рис. 3 представлено сценарій видалення завдання, який включає перевірку обраного елемента, звернення до локального сховища та оновлення інтерфейсу після успішного видалення.

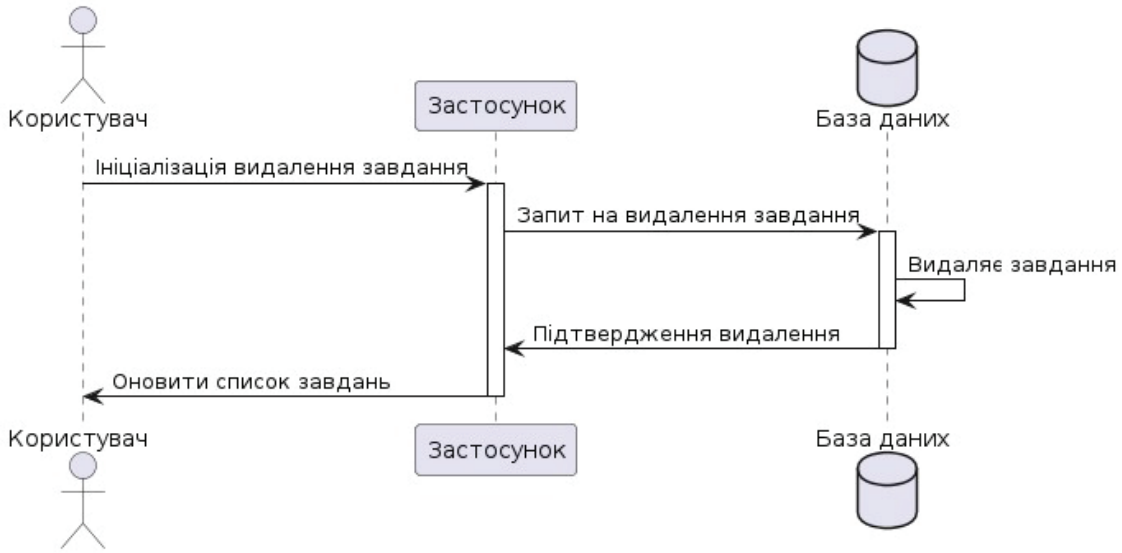


Рис. 3. Діаграма послідовності «Видалення завдання»

На рис. 4 показано процес фільтрації завдань, де ViewModel обробляє параметри фільтрації, виконує запит до сховища та оновлює відображення без модифікації даних.

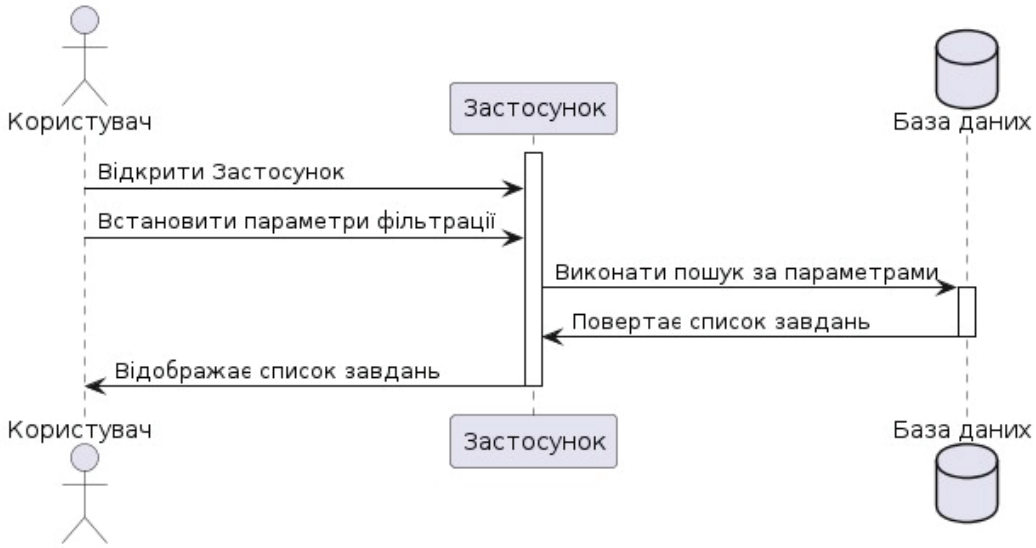


Рис. 4. Діаграма послідовності «Фільтрація завдання»

Обидва сценарії демонструють узгоджену взаємодію між компонентами системи та підтверджують ефективність застосування архітектурного шаблону MVVM у контексті кросплатформної розробки.

Діаграма класів (Class Diagram). Діаграма класів (рис. 5) відображає логічну структуру системи, включаючи основні сутності: Task, Category, Schedule, StorageService, TaskController, ViewModel. Вона демонструє зв'язки між класами, їх атрибути та методи, що реалізують функціональність системи. Клас Task містить інформацію про окреме завдання, включаючи назву, опис, дату виконання та статус. Category відповідає за класифікацію завдань, Schedule − за формування списку завдань на поточний день. StorageService реалізує доступ до локального сховища, TaskController координує взаємодію між моделлю та інтерфейсом, а ViewModel забезпечує оновлення інтерфейсу відповідно до змін у даних.

Завдяки чіткому розділенню відповідальностей між класами, система залишається масштабованою, зручною для тестування та придатною для подальшого розширення функціональності.

Рекомендації щодо вибору програмних засобів для реалізації системи

Для реалізації програмної системи тайм-менеджменту, спроєктованої відповідно до архітектурного шаблону MVVM, рекомендовано використовувати набір програмних засобів, що забезпечують кросплатформність, автономність, масштабованість та зручність розробки. Вибір інструментів обґрунтовано специфікою освітнього середовища, потребою в локальному зберіганні даних та можливістю швидкої адаптації інтерфейсу. До рекомендованого стеку технологій належать:

- Flutter Framework − сучасний фреймворк для кросплатформної розробки мобільних застосунків, що дозволяє створювати єдину кодову базу для мобільних платформ (Android, iOS), а також для настільних операційних систем − Windows, macOS і Linux [7].
- Dart Programming Language − мова програмування, оптимізована для роботи з Flutter, підтримує реактивну модель та спрощує реалізацію ViewModel-компонентів [8].
- Flutter Widgets − набір вбудованих компонентів для побудови адаптивного інтерфейсу, що відповідає принципам MVVM [7].
- Flutter Packages, зокрема:
  - *provider* − для реалізації зв'язку між ViewModel та View [9];
  - *flutter_local_notifications* − для локальних нагадувань [10];
  - *shared_preferences* − для збереження налаштувань користувача [11].
- SQLite − локальна база даних, яка відповідає вимогам автономності, конфіденційності та простоти інтеграції як у мобільні застосунки, так і в настільні програмні рішення [12].

Застосування зазначених засобів дозволить ефективно реалізувати архітектурну модель MVVM, забезпечити масштабованість та адаптацію системи до потреб освітнього процесу.

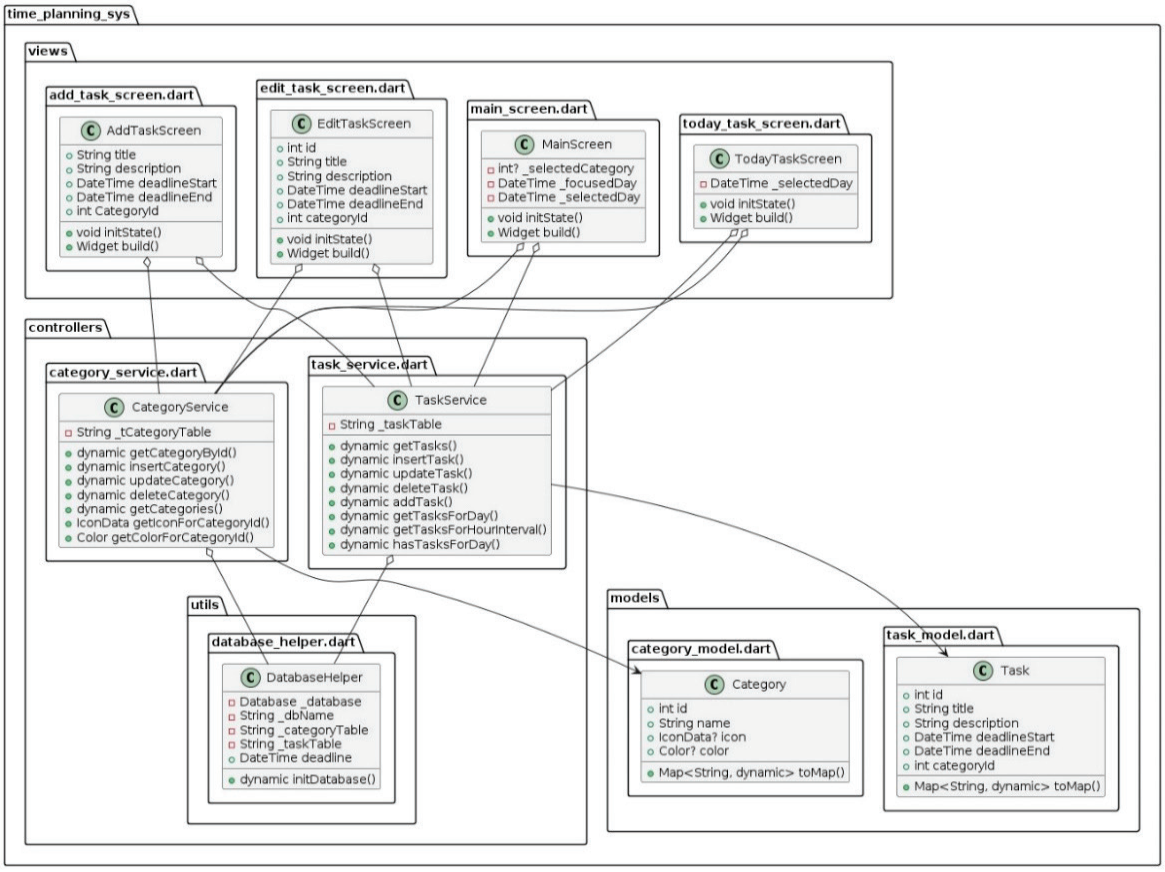


Рис. 5. Діаграма класів

Висновки

У межах дослідження здійснено архітектурне проєктування навчально орієнтованої програмної системи тайм-менеджменту, що відповідає сучасним вимогам кросплатформності, автономності та конфіденційності. Обґрунтовано вибір архітектурного шаблону MVVM, який забезпечує чітке розділення відповідальностей між компонентами та сприяє масштабованості й тестованості системи. Для формалізації структури та поведінки системи побудовано низку UML-діаграм, що охоплюють як статичні, так і динамічні аспекти: діаграму варіантів використання, активностей, послідовності та класів. Логічна структура проєкту організована відповідно до принципів clean architecture, що забезпечує ізоляцію функціональних зон та підтримує освітню придатність моделі. Завершальний розділ містить рекомендації щодо вибору програмних засобів, які забезпечують ефективну реалізацію системи планування часу з урахуванням вимог кросплатформності.

Запропонована модель є придатною для використання в освітньому процесі, зокрема для формування навичок архітектурного моделювання, системного мислення та реалізації повноцінних програмних продуктів. У подальшому система може бути розширена за рахунок інтеграції з хмарними сервісами, синхронізації з календарем користувача та реалізації мережевої взаємодії, що відкриває перспективи її застосування в побутовій та професійній діяльності.

Література

1. Савченко О. В. Тайм-менеджмент як інструмент підвищення ефективності навчальної діяльності студентів. *Педагогічний альманах*. 2021. № 49. С. 112–117.
2. The to-do list to organize work & life. *Todoist*. URL: https://todoist.com
3. Stay organized and manage your day. *Microsoft To Do*. URL: https://to-do.microsoft.com
4. Your all-in-one productivity platform. *Any.do*. URL: https://www.any.do
5. Глушков І. В. Методика навчального проєктування в IT-освіті. *Інформаційні технології в освіті*. 2020. № 42. С. 45–52.
6. Model-View-ViewModel – .NET MAUI Architecture. *Microsoft Learn*. URL: https://learn.microsoft.com/en-gb/dotnet/architecture/maui/mvvm
7. Build apps for any screen. *Flutter*. URL: https://flutter.dev
8. A programming language optimized for UI. *Dart*. URL: https://dart.dev
9. Provider package for Flutter. URL: https://pub.dev/packages/provider
10. Flutter Local Notifications. URL: https://pub.dev/packages/flutter_local_notifications

11. Shared Preferences plugin. URL: https://pub.dev/packages/shared_preferences
12. Lightweight, embedded SQL database engine. *SQLite*. URL: https://www.sqlite.org

References

1. Savchenko O. V. (2021). Taim-menedzhment yak instrument pidvyshchennia efektyvnosti navchalnoi diialnosti studentiv [Time management as a tool for improving students' learning efficiency]. *Pedagogical Almanac*. 49. Pp. 112–117 [in Ukrainian].
2. The to-do list to organize work & life. *Todoist*. URL: https://todoist.com
3. Stay organized and manage your day. *Microsoft To Do*. URL: https://to-do.microsoft.com
4. Your all-in-one productivity platform. *Any.do*. URL: https://www.any.do
5. Hlushkov I. V. (2020). Metodyka navchalnoho proiektuvannia v IT-osviti [Methodology of educational design in IT education]. *Information Technologies in Education*. 42. Pp. 45–52 [in Ukrainian]
6. Model-View-ViewModel – .NET MAUI Architecture. *Microsoft Learn*. URL: https://learn.microsoft.com/en-gb/dotnet/architecture/maui/mvvm
7. Build apps for any screen. *Flutter*. URL: https://flutter.dev
8. A programming language optimized for UI. *Dart*. URL: https://dart.dev
9. Provider package for Flutter. URL: https://pub.dev/packages/provider
10. Flutter Local Notifications. URL: https://pub.dev/packages/flutter_local_notifications
11. Shared Preferences plugin. URL: https://pub.dev/packages/shared_preferences
12. Lightweight, embedded SQL database engine. *SQLite*. URL: https://www.sqlite.org

Дата першого надходження статті до видання: 24.10.2025
Дата прийняття статті до друку після рецензування: 25.11.2025
Дата публікації (оприлюднення) статті: 26.12.2025