ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(*підпис*)

«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему
«РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ: КАНБАН ДОШКА»

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавра

Виконавець роботи Гераскевич Роман Сергійович
_____ «___»_____2023 р.
(підпис)

Науковий керівник к.ф.-м.н., доцент, Черненко Оксана Олексіївна
_____ «___»_____2023 р.
(підпис)

Рецензент

ПОЛТАВА 2023 р.

ЗМІСТ

ВСТУП

Kanban — дуже проста та ефективна методологія ведення проектів і задач для співробітників компанії. Проекти розбиваються на етапи, задачі перетягуються між ними по мірі готовності. Візуально швидко ясно, що відбувається з проектом в буд-який відрізок часу [12].

Система канбан розпочала свій шлях у 1950-х роках на виробничих лініях корпорації Toyota, після чого перекочувала в офіси і стала важливим інструментом для проектних менеджерів.

Безмежна гнучкість практики і її можливості для самоорганізації персоналу дозволили домогтися ефективності там, де інші підходи не працювали. Це той випадок, коли візитною карткою системи стала сама картка — вона утвердилася як внутрішня валюта в організаціях, які впровадили канбан[13]

Шість основних практик методу Kanban є:

- Візуалізуйте потік роботи:
  - Зафіксуйте завдання, статус по ним і відповідальних співробітників на реальній або віртуальній дошці.
- Обмежте задачі в процесі виконання:
  - Обговоріть з командою, яку кількість робіт по кожному статусу оптимально вести одночасно, а також розставте пріоритети. Над кожним стовпцем дошки вкажіть ліміт.
- Керуйте потоком завдань
  - Своєчасно міняйте статуси завдань та відстежуйте рух. За допомогою цього ви зможете оцінити позитивні та негативні ефекти змін у системі.
- Обговоріть правила роботи
  - Співробітники повинні чітко розуміти, як функціонує система, як структурувати інформацію в ній.

- Аналізуйте діяльність
  - Не забувайте влаштовувати зустрічі з командою і обговорювати нюанси роботи.
- Розвивайтесь експериментально.
  - Розвиток kanban-команди – це пошук ідеалу. Поступово впроваджуйте зміни, відстежуйте результат, отримуйте зворотний зв'язок, переходьте до наступного експерименту.

Мета кваліфікаційної роботи – програмна реалізація канбан дошки.

Об'єктом розробки є процес організації потоку виконання задач.

Предмет розробки — програмний продукт «Kanban дошка».

Обсяг пояснювальної записки: 41 ст., в т.ч. основна частина - 32 ст., додатки - 8 ст., джерела - 14 назв.

1. ПОСТАНОВКА ЗАДАЧІ

Для досягнення поставленої мети у роботі необхідно вирішити такі основні завдання:

- аналіз вимог і формальна постановка задачі;
- аналіз способів вирішення;
- логічне проектування і розробка алгоритму;
- кодування (написання програми);
- тестування і налагодження програмного забезпечення;
- впровадження, використання і супровід програмного забезпечення.

Постановка завдання розробки програмного забезпечення є найпершим і найбільш важливим етапом при проектуванні програмного забезпечення. Саме тут закладається фундамент майбутньої програми. Якщо на цьому етапі допущені помилки або не передбачені якісь важливі деталі, то це відіб'ється на кінцевому результаті, а вносити зміни і виправлення в такий проект буде вкрай проблематично.

Перший крок у проектуванні програмного забезпечення полягає в точному формулюванні цілей впровадження програми. Необхідно, щоб цей процес був гнучко організований і тривав протягом тривалого часу, оскільки на будь-якому етапі розробки або впровадження можуть розкриватися раніше непередбачені проблеми, які потребують внесення в проект певних змін. У той же час для спрощення розробки слід заборонити радикальний перегляд вимог на стадії кодування (реалізації) програми. Необхідно також заздалегідь визначити умови внесення несуттєвих змін в проект. Всі домовленості такого плану повинні оформлятися розробником і замовником програми в офіційному порядку у вигляді окремих пунктів у договорі на розробку програмного забезпечення, додаткових протоколів чи актів.

З офіційним висновком договору зазвичай передують:

- з'ясування реальної необхідності такої системи;
- оцінка можливості її розробки і зразкового обсягу витрат;
- визначення очікуваного ефекту від впровадження.

Після завершення етапу попередніх досліджень складають список вимог, що пред'являються до програмного забезпечення. Сюди входять:

- аналіз обстановки (сукупність умов, в яких передбачається експлуатувати програмну систему);
- опис виконуваних програмою системою функцій (чіткий опис того, що повинна робити система, на підставі яких вхідних даних, які дані є вихідними);
- обмеження, які повинні враховуватися в процесі проектування (терміни завершення; ресурси, наявні в наявності).

Головна мета на етапі аналізу формальної постановки задачі і складання вимог до програми полягає в пошуку і поданні таких ситуацій, які можуть привести до збою в роботі програми і у визначенні причин і способів подолання таких ситуацій .

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Сервіси для планування та візуалізації завдань: спільне та відмінне

Favro

Дозволяє розбивати великі завдання на кілька маленьких і контролювати їх виконання. Списки можна доповнювати необхідними файлами. Але при великому потоці завдань стає важко контролювати дошки, деякі можливості сервісу недоступні.

Підійде для невеликих і середніх команд. Включає три версії програми: Lite (10,2$ в місяць), Standard (13,6$ в місяць) і Enterprise (25,5$ в місяць) [6].

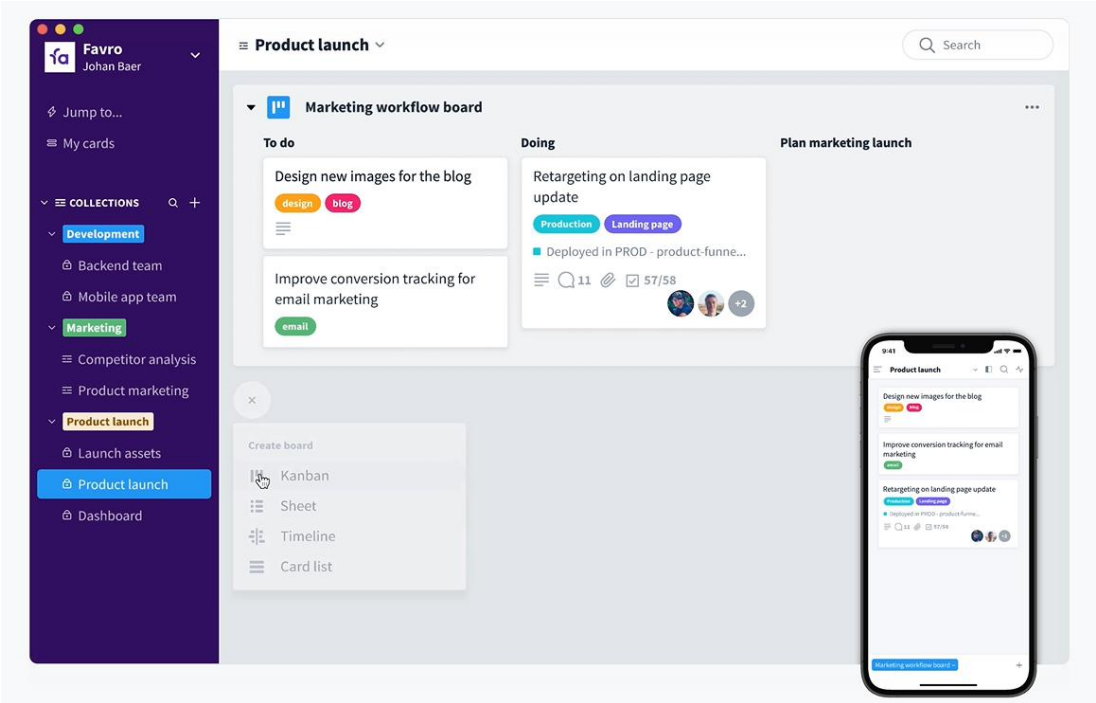

Рисунок 2.1-Favro

Asana

За допомогою цієї платформи дуже зручно розставляти пріоритети. Колонок може бути багато, а сам додаток ясно показує дедлайн і ступінь виконання завдання.

Сервіс підходить для індивідуального використання або роботи в невеликих командах.

Безкоштовна версія сервісу включає необмежене число дощок і проєктів, безлімітного сховища; над одним проєктом може працювати команда з 15 осіб. Premium версія коштує 10,99$ в місяць. Вона дає доступ до розширеного пошуку і звітності, налаштованих полів [7].

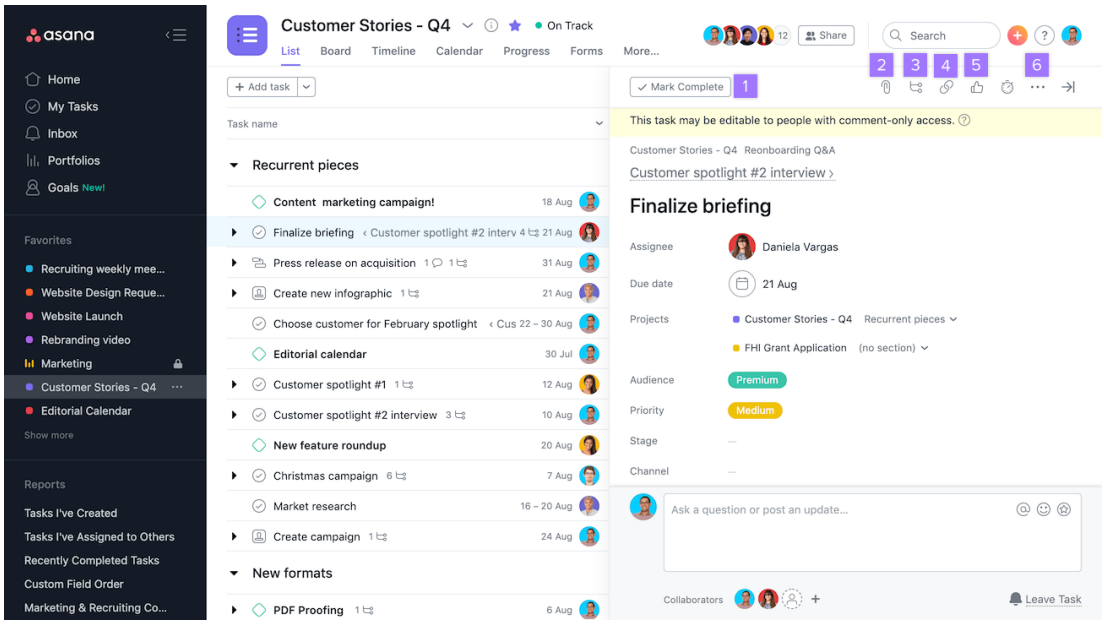

Рисунок 2.2-Asana

Blossom

Зручний інструмент для управління проєктів. Стадії виконання простежуються в списках. Але є одна особливість: додавати нові списки можна тільки після виконання старих. Неможливе індивідуальне використання.

Підійде для невеликих і середніх команд. Вартість підписки: 19 $ на місяць, є можливість спробувати сервіс безкоштовно [8].

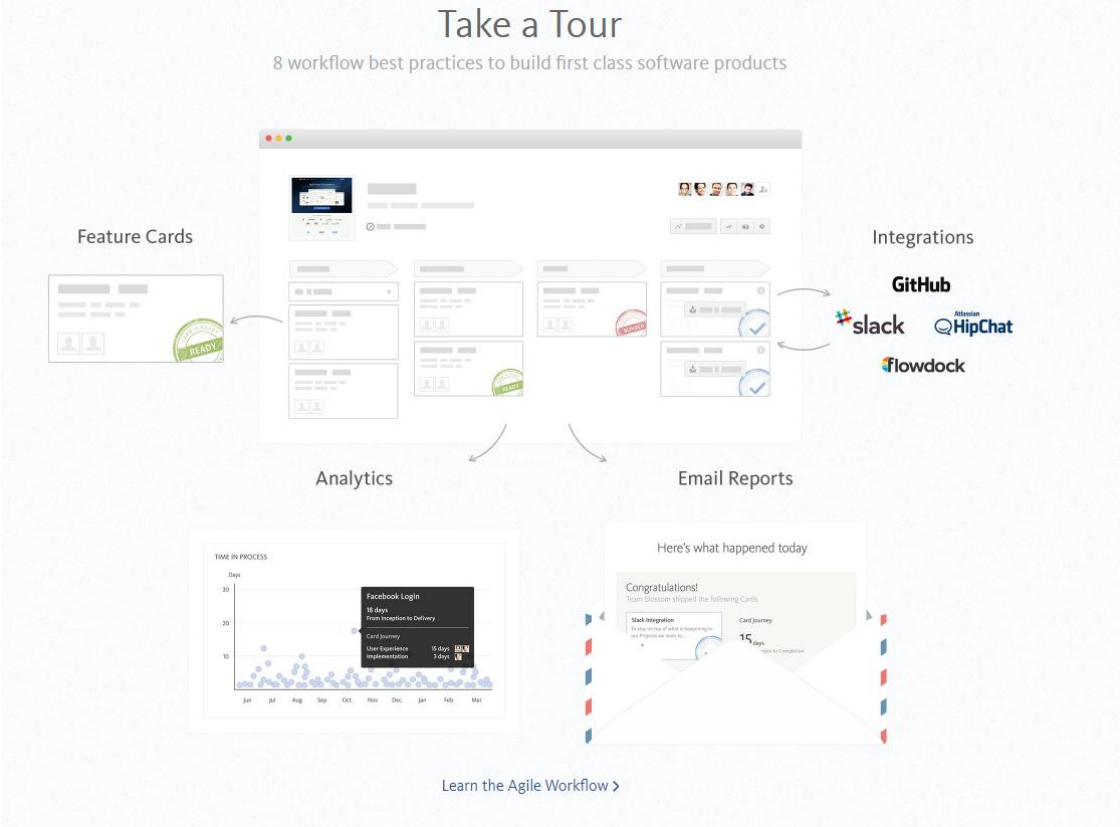

Рисунок 2.3-Blossom

Trello

Додаток призначений для планування завдань. Можна створювати будь-яку кількість проектів з різним складом команди. До карток можна додавати різнокольорові мітки, долучати файли і залишати коментарі.

Безкоштовно доступний майже повний функціонал Kanban. На платному тарифі відсутнє обмеження за обсягом вкладень, можна додавати власні стікери та фони. Але варто зазначити, що у додатку немає функцій WIP limits і Swimlanes.

Інструмент простий у використанні, підходить для командного та індивідуального використання. Підходить всім, кого цікавлять базові функції Kanban.

Дві версії використання: Business Class (9,99$ в місяць) та Enterprise за 17,5$ [9].

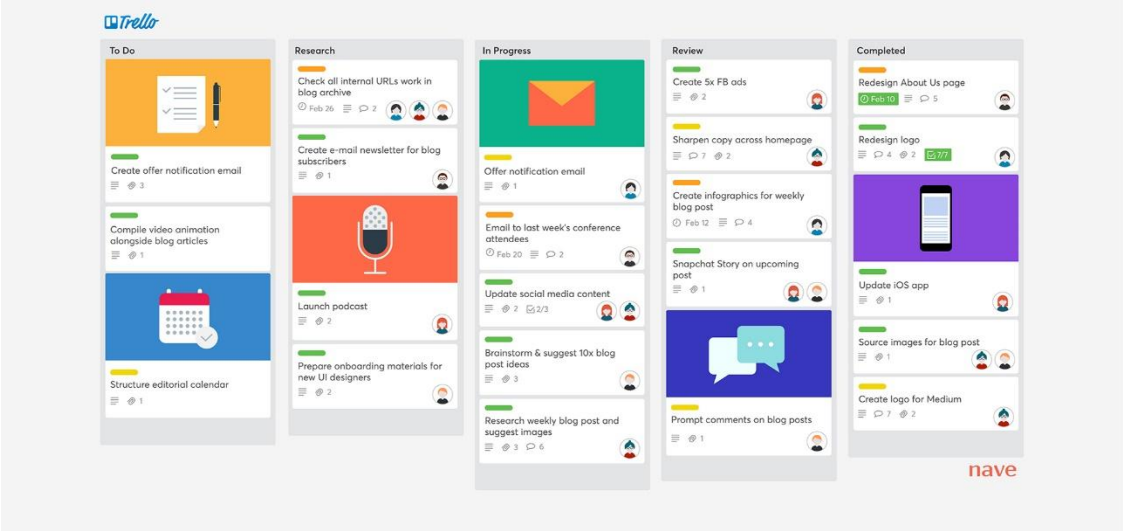

Рисунок 2.4-Trello

Hygger

У розпорядженні користувача виявляються to do листи з Kanban-дошками, функціями WIP limits і Timelines.

При реєстрації облікового запису можна під'єднати декілька користувачів і вказати кількість учасників проекту. Тоді буде простіше розподілити людей по задачам, що зробить роботу більш ефективною. Інтерфейс програми схожий на Trello. Проте відсутня багатомовна підтримка.

В безкоштовній версії програми передбачено необмежене число користувачів, проектів і дощок, а також сховище на 100 MB. У Standard (7$ в місяць) – безлімітне сховище. В Enterprise (14$ на місяць) – можливість створювати кілька проектів на одній дошці, Google Apps SSO, виклики API, преміальна служба підтримки [10].

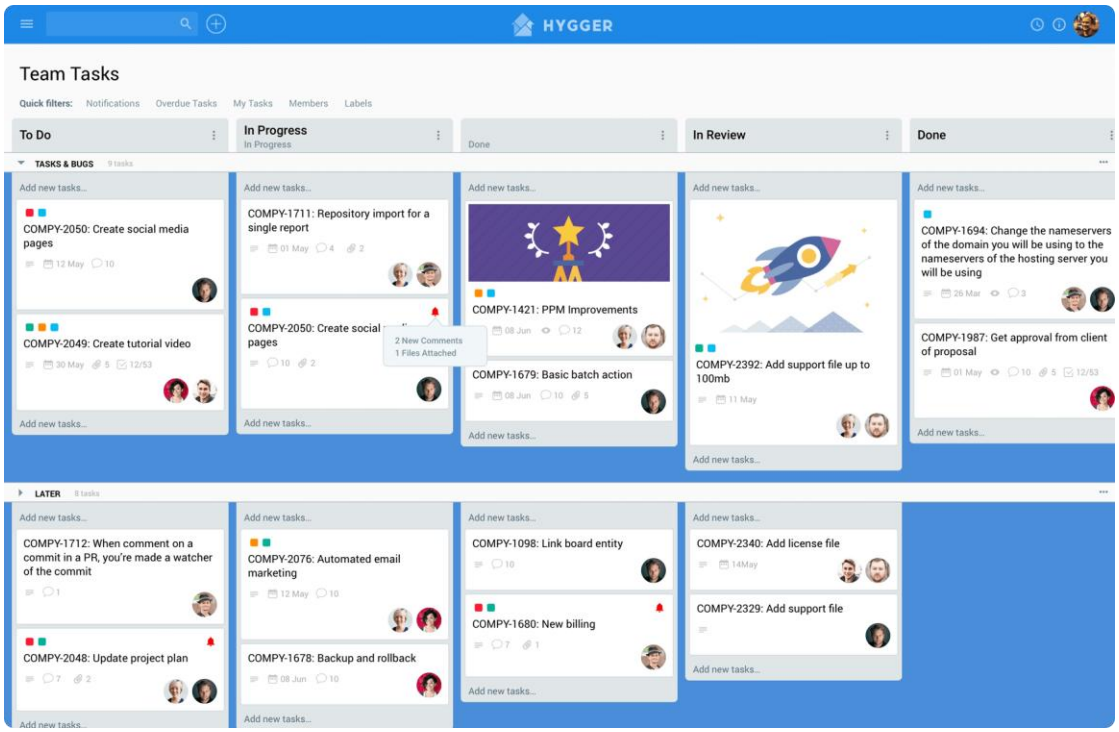

Рисунок 2.5-Hygger

Jira

Це сервіс для планування проектів. Усередині кожного проекту розміщені три дошки: to do, in progress, done, призначені для розставлення завдань. Списки справ можна доповнювати фото-, аудіо- та відеоматеріалами. Проте одну задачу може виконувати тільки один виконавець, сервіс не підходить для нетехнічних компаній.

Підходить для Agile-розробників, особливо для IT-компаній з великим штатом співробітників.

В безкоштовній версії користувач може зберігати 2 GB файлів. У версії Standard (7$ в місяць) – 250 GB. А в Premium (14$ в місяць) сховище необмежено [11] [5].

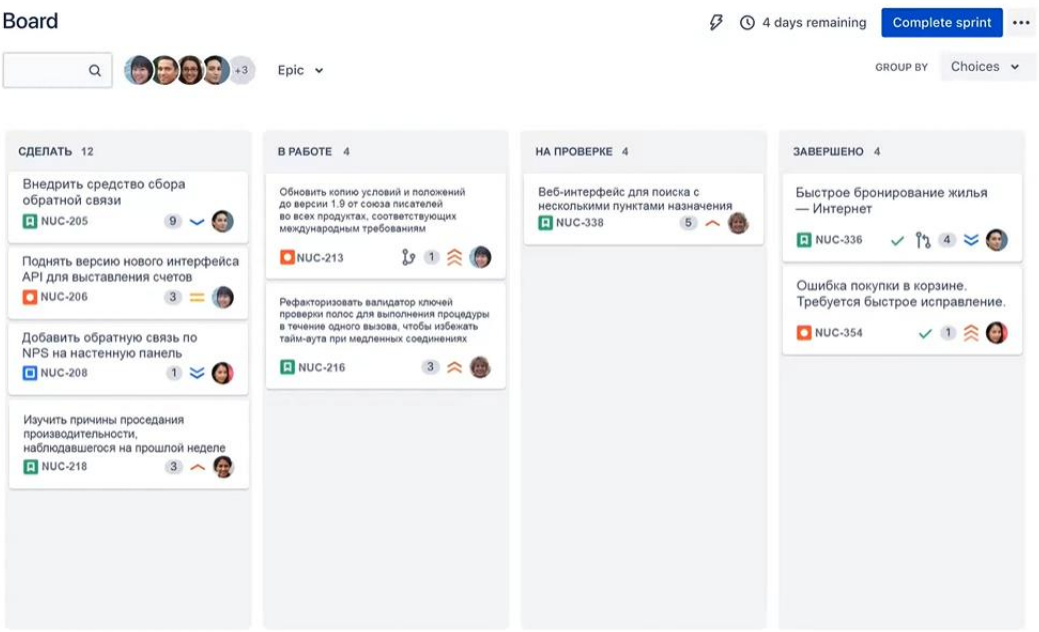

Рисунок 2.6-Jira

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1 Опис мови програмування

Для кваліфікаційної роботи обрано кросплатформений фреймворк Flutter и мору програмування Dart.

Dart — мова програмування, яку розробляє компанія Google, позиціонуючи як мову структурованого програмування для Веб. Розробники вважали, що в довгостроковій перспективі Dart може стати прогресивною заміною JavaScript, котрий потерпає від наявних в даний час проблем з розширюваністю, продуктивністю і підтримкою розробки складних застосунків. Мова має схожий на Java синтаксис, не вимагає явного визначення типів і її можна використовувати для створення серверних та клієнтських застосунків.

У березні 2015 компанія Google представила оновлену стратегію просування Dart, у котрій вирішено не прив'язувати Dart до браузеру і відмовитися від ідеї інтеграції віртуальної машини Dart у Chrome. Розробку буде зосереджено на застосуванні Dart як проміжної мови, скомпільованої в JavaScript. Розвиток Dart як окремої мови, альтернативної JavaScript і безпосередньо підтримуваної у браузерах, визнано недоцільним. Замість цього Dart рухатиметься у бік якіснішої інтеграції з JavaScript і генерації оптимального JavaScript-коду. При цьому розробку віртуальної машини Dart VM буде продовжено, але вона позиціонуватиметься в основному для створення серверних і мобільних застосунків.

Влітку 2014 асоціація ECMA International, що займається стандартизацією інформаційних і комунікаційних технологій, затвердила специфікацію ECMA-408 [Архівовано 30 березня 2015 у Wayback Machine.], яка стандартизує синтаксис і семантику мови Dart, а також склад базових бібліотек і супутніх мові технологій, відтоді мова Dart є офіційним стандартом Ecma. 9 Грудня 2015 в Токіо було затверджено видання [Архівовано 31 липня

2021 у Wayback Machine.]. Надання Dart статусу стандарту Ecma дозволить розширити область використання мови та прискорити забезпечення його підтримки в наявних на ринку браузерах і продуктах. Вибір Ecma International як організації для стандартизації обумовлений тим, що ця асоціація вже розвиває близькі до специфіки Dart стандарти для мов JavaScript, Eiffel і C#. Просуванню Dart як стандарту сприяло надання компанією Google всіх пов'язаних з розробкою патентів у безоплатне використання, що не вимагає оплати відрахувань (royalty free).

Flutter – це розроблений Google фреймворк з відкритим програмним кодом, який дозволяє просто і швидко створювати мобільні додатки для iOS і Android.

При цьому в роботі Flutter не використовує нативні компоненти зовсім. Замість цього всі UI-елементи у фреймворку створюються за допомогою власного графічного движка. Flutter дозволяє створювати всі елементи призначеного для користувача інтерфейсу додатку з готових віджетів. У цьому Flutter схожий з іншими фреймворками – React і Vue, і в той же час має ряд відмінностей від них. Так, він не використовує мову програмування Javascript, натомість Flutter використовує мову Dart [14].

3.2 Алгоритм роботи Kanban

1. Реєстрація або авторизація в програмі
2. Створення дошки для проекту
3. Відкриття створеної або існуючої дошки дошки
4. Створення секцій, що описують етапи виконання типової задачі
5. Створення карточок з описом задач які повинні бути виконані
6. Видалення або редагування не неактуальних або застарілих задач
7. Перетягування одної або декількох карточок в наступну секцію яка призначена для задач які зараз виконуються
8. По мірі виконання кожної з задач, перетягування карточок в наступну секцію для перевірки

9. При знаходженні помилок на етапі перевірки, задачі перетягуються на початкову секцію з дописом про те що конкретно потрібно виправити

10. Якщо задача успішно пройшла перевірку, вона переміщається в наступну секцію для виконаних задач

3.3 Алгоритм реалізації Kanban

1. Визначення платформи під яку створюється додаток (OS Android)
2. Визначення технологій для реалізації (Framework Flutter)
3. Визначення способу ідентифікації користувачів (авторизація, демо режим)
4. Визначення бібліотек для реалізації потрібного функціоналу
    a. Бібліотека для реалізації механізму перетягування карток з колонки в колонку за допомогою жестів (boardview)
    b. Бібліотека для роботи з базою даних для збереження стану програми (drift)
5. Створення нового проекту і підключення визначених бібліотек
6. Створення усіх необхідних об’єктів
    a. Користувач
    b. Дошка
    c. Колонка
    d. Карточка
7. Створення і тестування механізму авторизації через демо аккаунт
8. Створення і тестування механізму для додавання і збереження дошок
9. Створення і тестування механізму для додавання і збереження колонок для дошки
10. Створення і тестування механізму для додавання і переміщення карточок між колонками
11. Створення і тестування сторінки авторизації

12. Створення і тестування головної сторінки програми на якій у вигляді блоків, знаходяться дошки користувача
13. Створення і тестування сторінки обраної дошки на якій знаходяться колонки з картками які можна додавати та переміщати
14. Перевірка працездатності системи збереження на кожному етапі роботи програми

3.4 Програмна реалізація Kanban

При відкритті мобільного додатку, користувача зустрічає вікно авторизації Рис 3.1.1, є можливість оцінити програму без реєстрації, натиснувши кнопку «Демо».

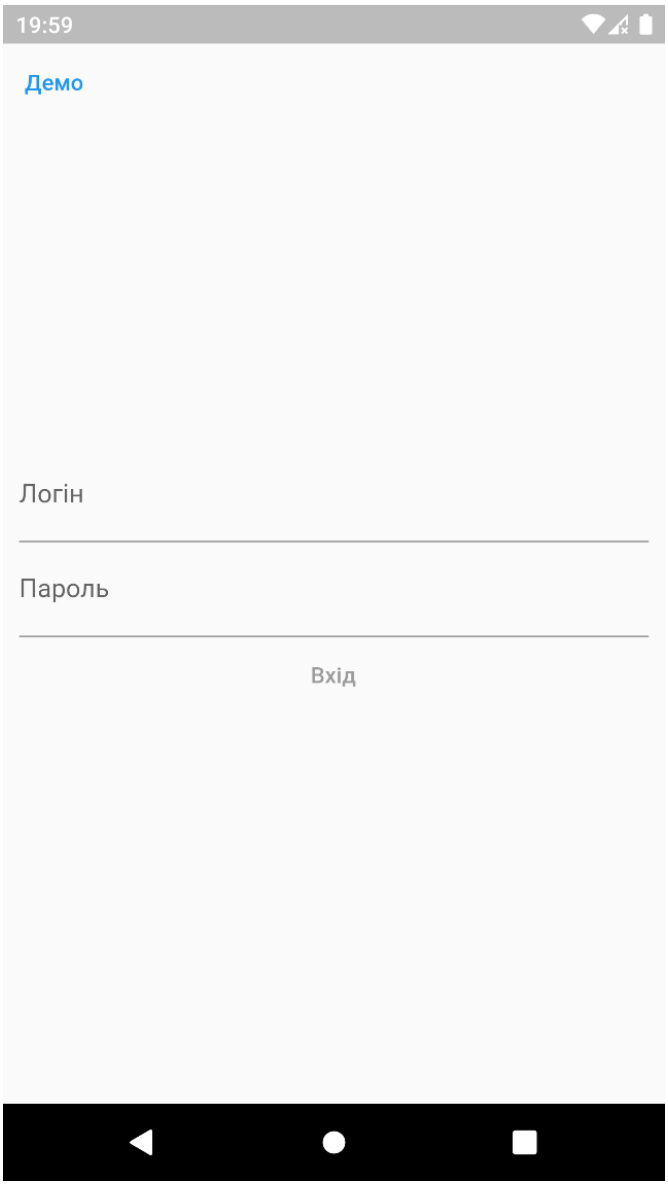

Рисунок 3.1-Сторінка авторизації

Після авторизації, на домашній сторінці Рис 3.1.2, користувач має можливість створити дошку Рис 3.1.3, або вибрати із створених раніше. Всі створені дошки, відображаються у вигляді списку плиток з їх назвами.

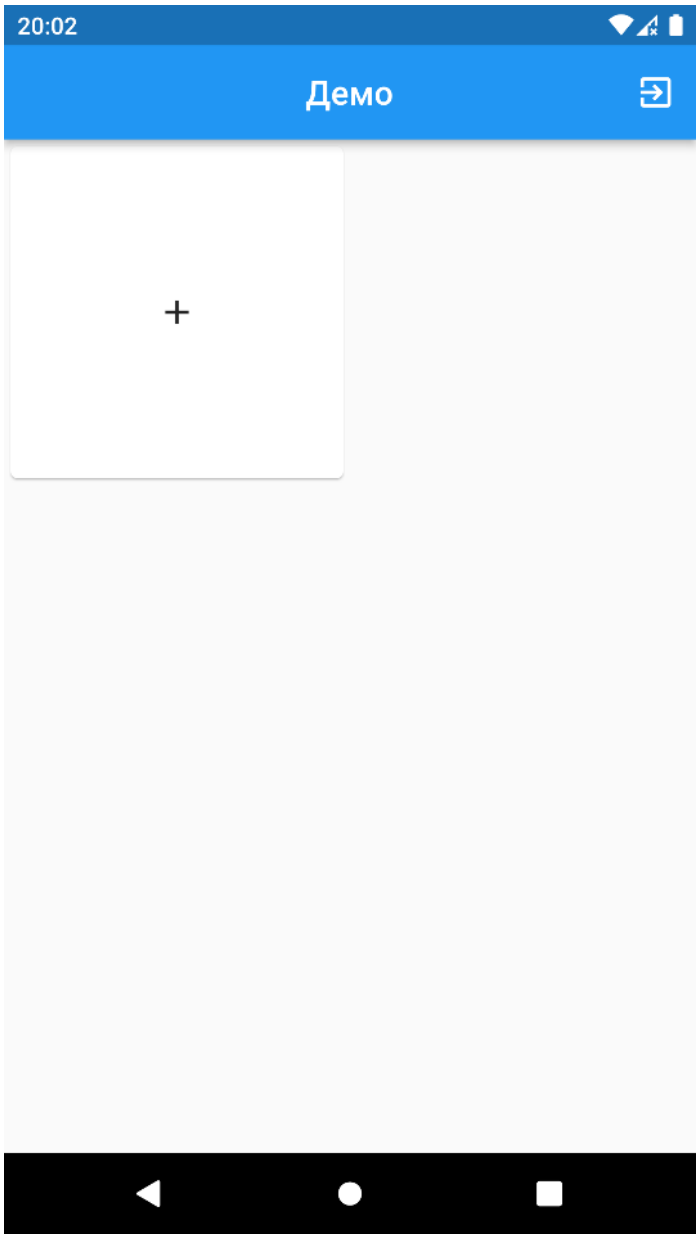

Рисунок 3.2- Домашня сторінка

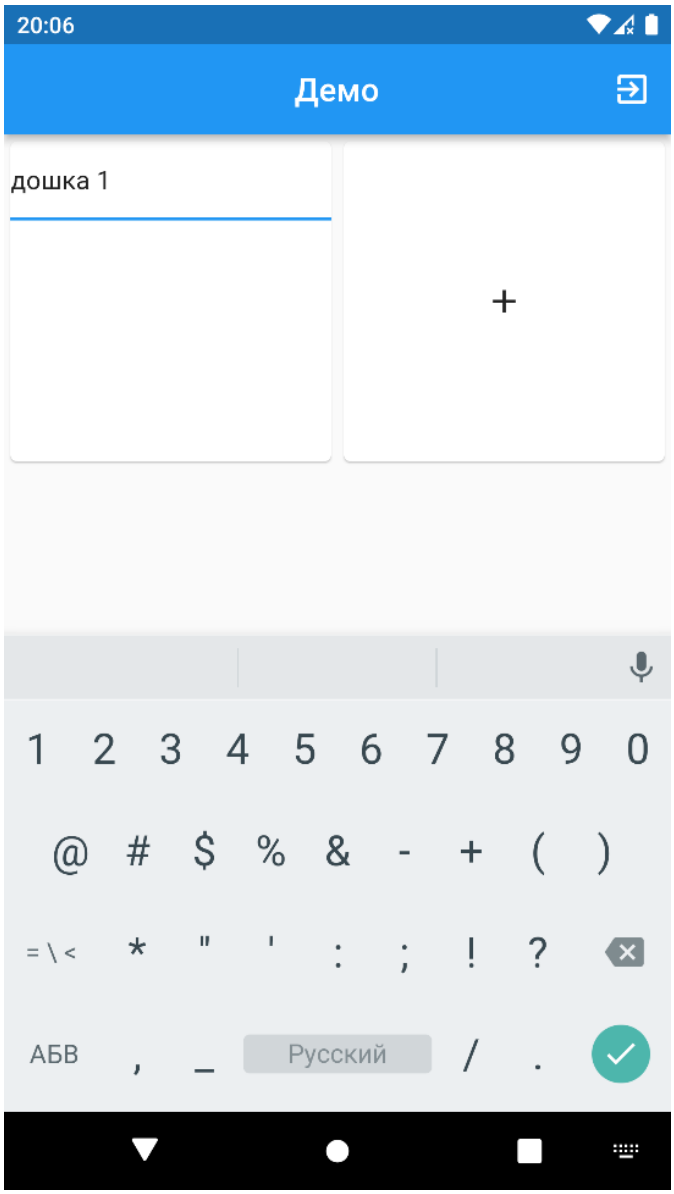

Рисунок 3.3- домашня сторінка

При натисканні на одну із дошок, вона відкривається у вигляді нового вікна, поділеного на секції Рис 3.1.5. Новостворена, ще не має жодної секції Рис 3.1.4, і пропонує користувачеві її створити надавши назву.

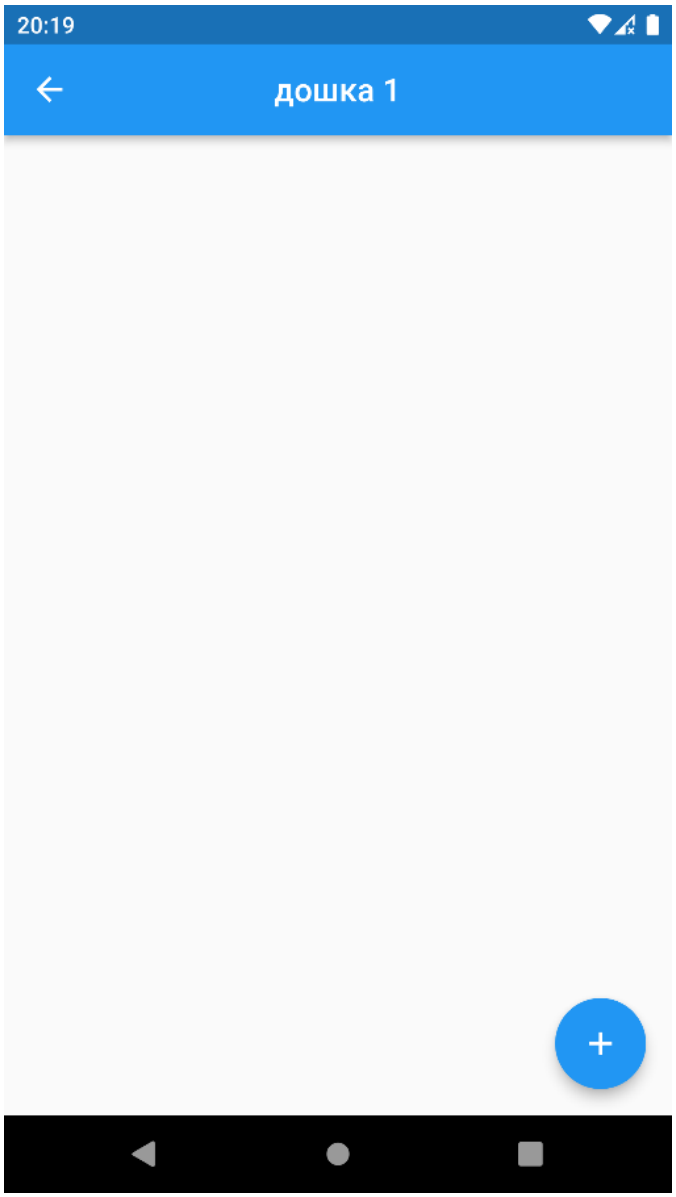

Рисунок 3.4-Сторінка відкритої пустої дошки

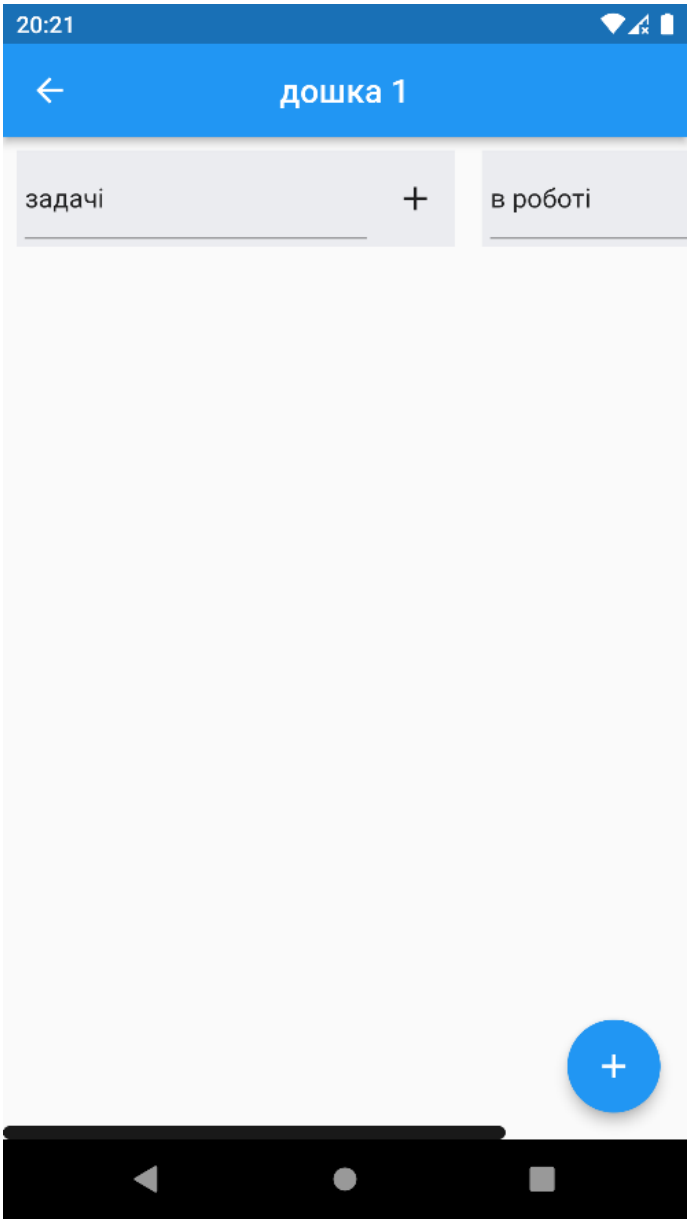

Рисунок 3.5-Сторінка відкритої дошки з секціями

У себе на роботі, я зазвичай використовую 3 основні секції: «Задачі», «В роботі», «Готово», назви говорять самі за себе.

На кожній секції можна створити карточку з текстом Рис 3.1.6, окрім того на ній, можна розмістити картинки, позначити окремим кольором (наприклад для підкреслення важливості задачі), встановити дедлайни виконання (Цей функціонал запланований в наступному релізі).

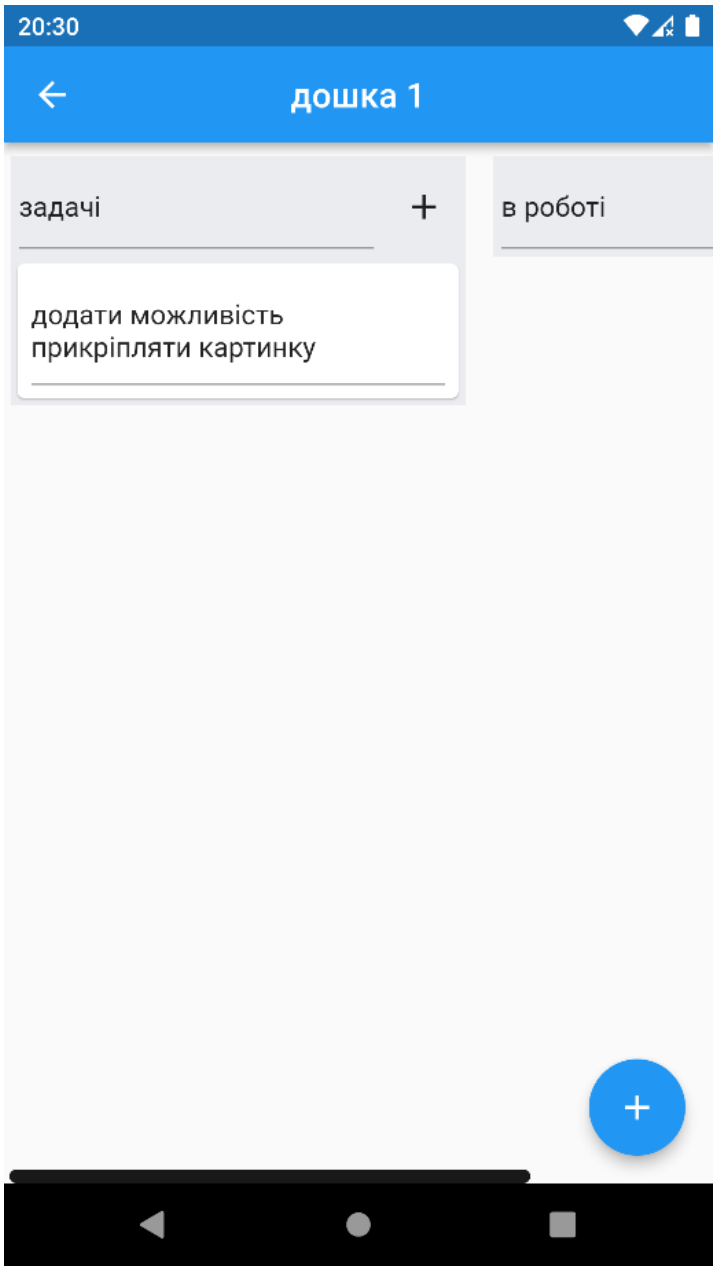

Рисунок 3.6-Новостворена карточка в секції «задачі»

Суть полягає в переміщенні задачі із секції в секцію по мірі її виконання Рис 3.1.7. На відмінну від програм по типу «Список справ», де кожна задача може були лише у стані «створено» та «виконано», (підходить наприклад для списку покупок), канбан використовується для більш складних процесів, як у моєму випадку, для розробки програмного забезпечення. В процесі розробки може брати участь ціла команда розробників.

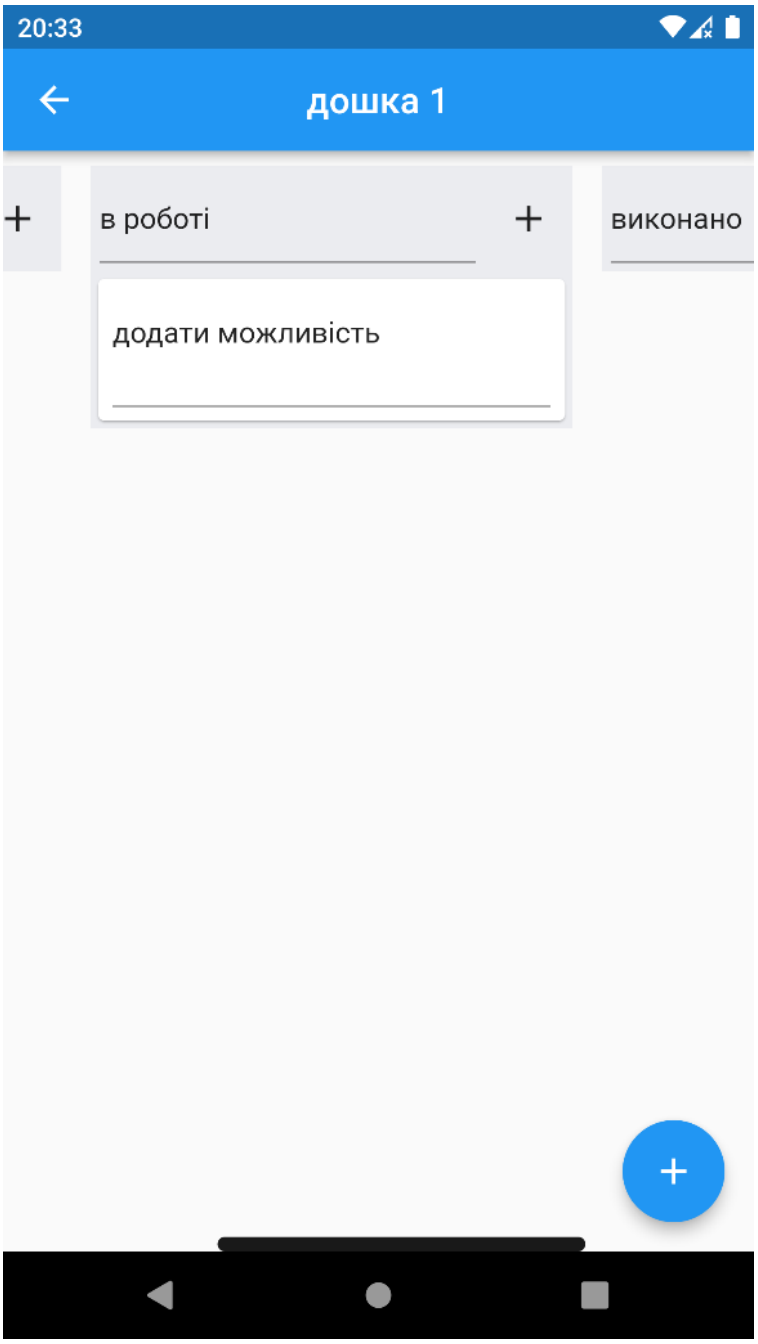

Рисунок 3.7-Карточка перетягнута в секцію «в роботі»

4. ПРАКТИЧНА ЧАСТИНА

4.1 Розробка діаграми класів, яка підлягає програмуванню

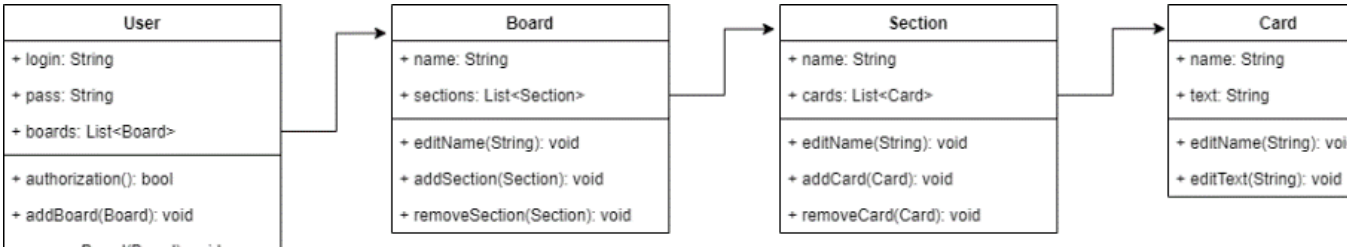

Рисунок 4.1 – Діаграма класів у реалізації

4.2 Опис процесу програмної реалізації

Готову програму, для демонстрації, було зібрано під платформу WEB і розміщено за адресом https://romanhs.github.io/kanban_board_web

Для створення додатку, було обрано мову програмування Dart і фреймворк Flutter. Обраний стек технологій, дозволяє створювати програми з графічним інтерфейсом одразу під шість платформ: Android, IOS, Windows, Mac OS, Linux та Web.

Код для ініціалізації та старту програми: (рис. А.1).

Для ініціалізації необхідних для роботи програми компонентів, був використаний патерн «Dependency injection» (рис. А.2). При старті також відбувається спроба дістати поточного авторизованого користувача.

Стартова сторінка, визначається в залежності від того чи було знайдено поточного авторизованого користувача (рис. А.3). Якщо користувача не знайдено, відкривається сторінка «Авторизації» (рис. А.4). Якщо знайдено, то «Домашня сторінка» (рис. А.5).

4.3 Опис роботи програми

Наразі у програмі доступний тільки «Демо» режим на сторінці авторизації (рис. 4.10):

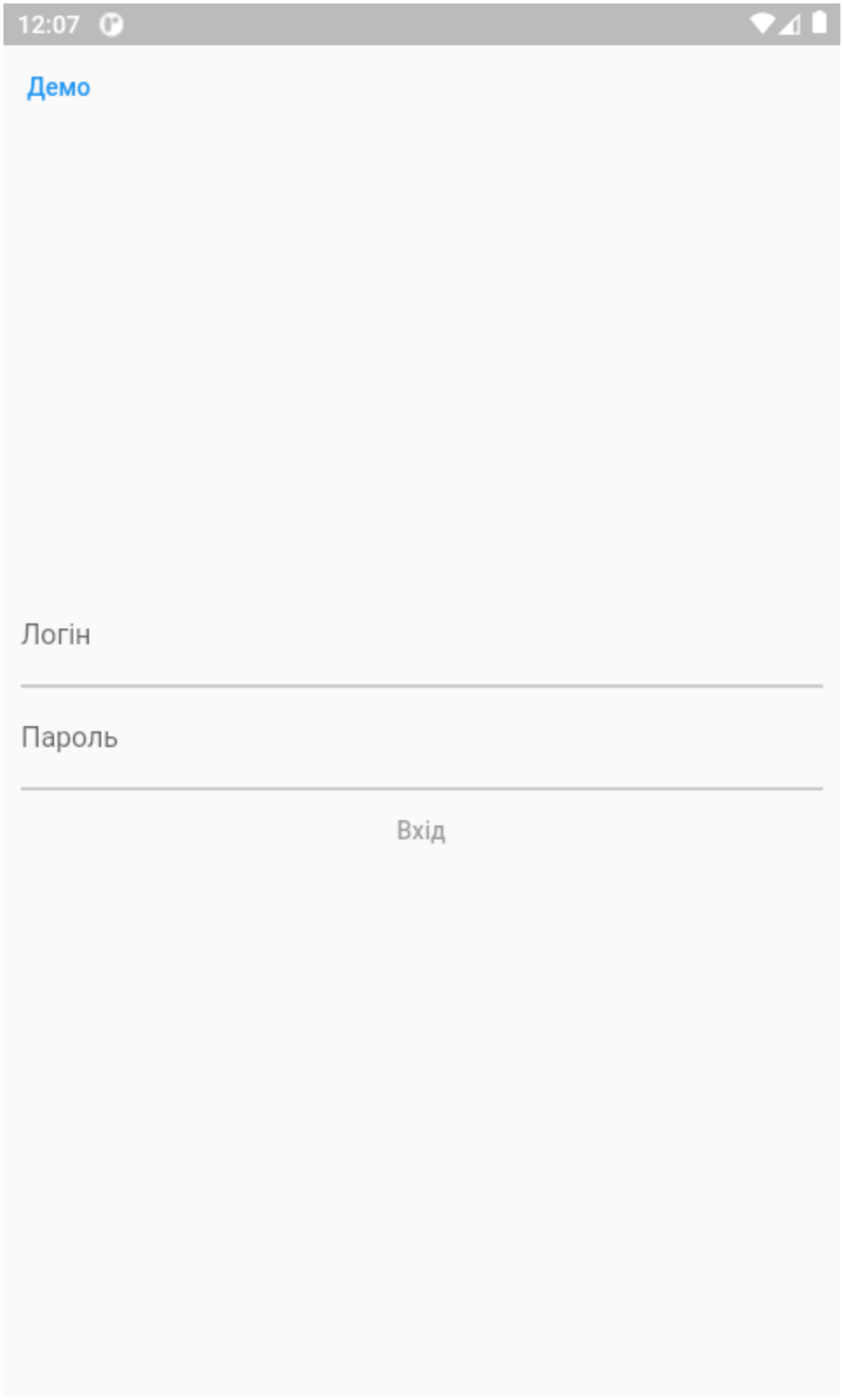

Рисунок 4.10 – Сторінка авторизації

На домашній сторінці (рис. 4.11), розміщені всі існуючі дошки та кнопка «+» для створення нової:

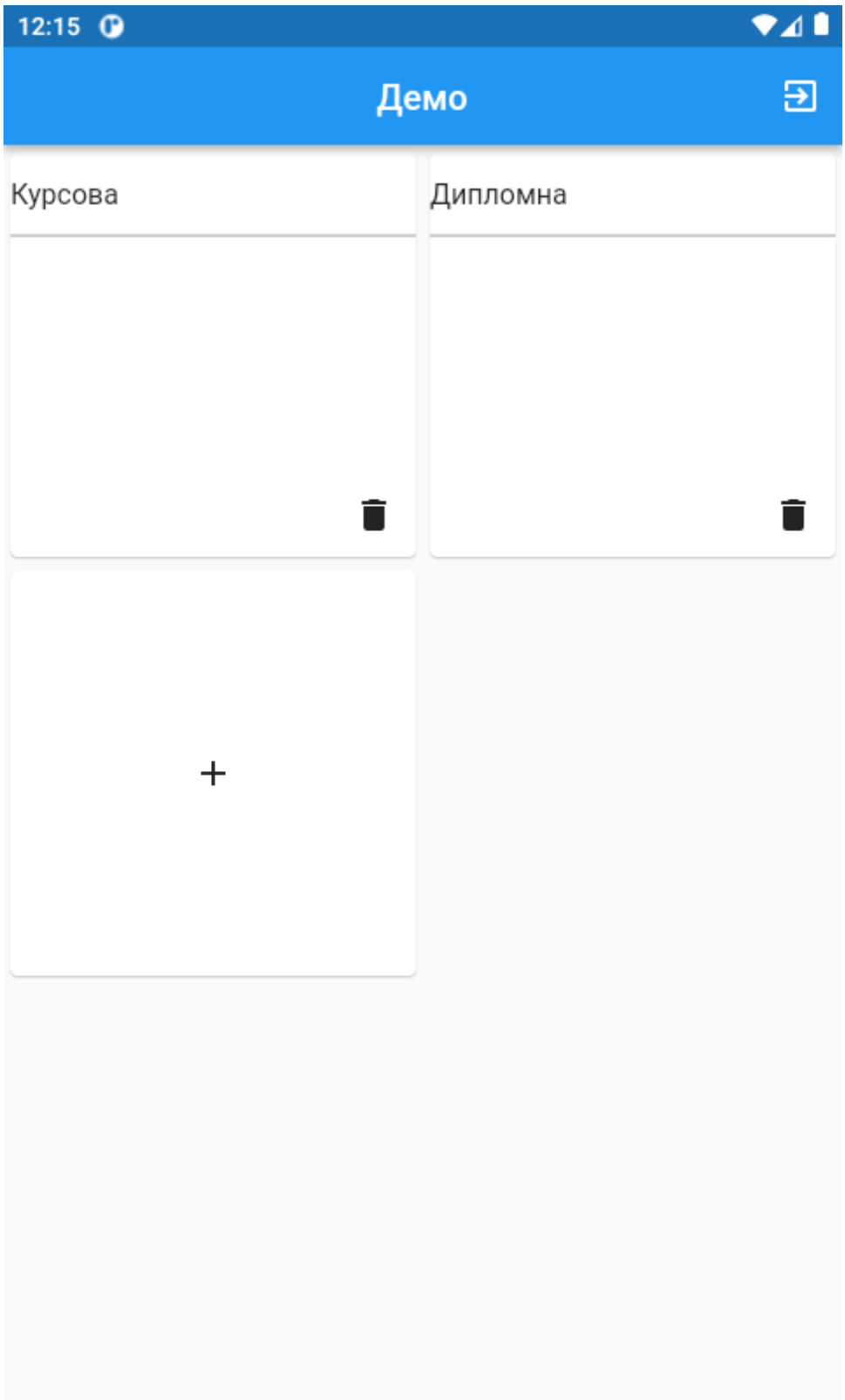

Рисунок 4.11 – Домашня сторінка

На сторінці дошки (рис. 4.12), розміщені колонки та кнопка «+» для створення нової:

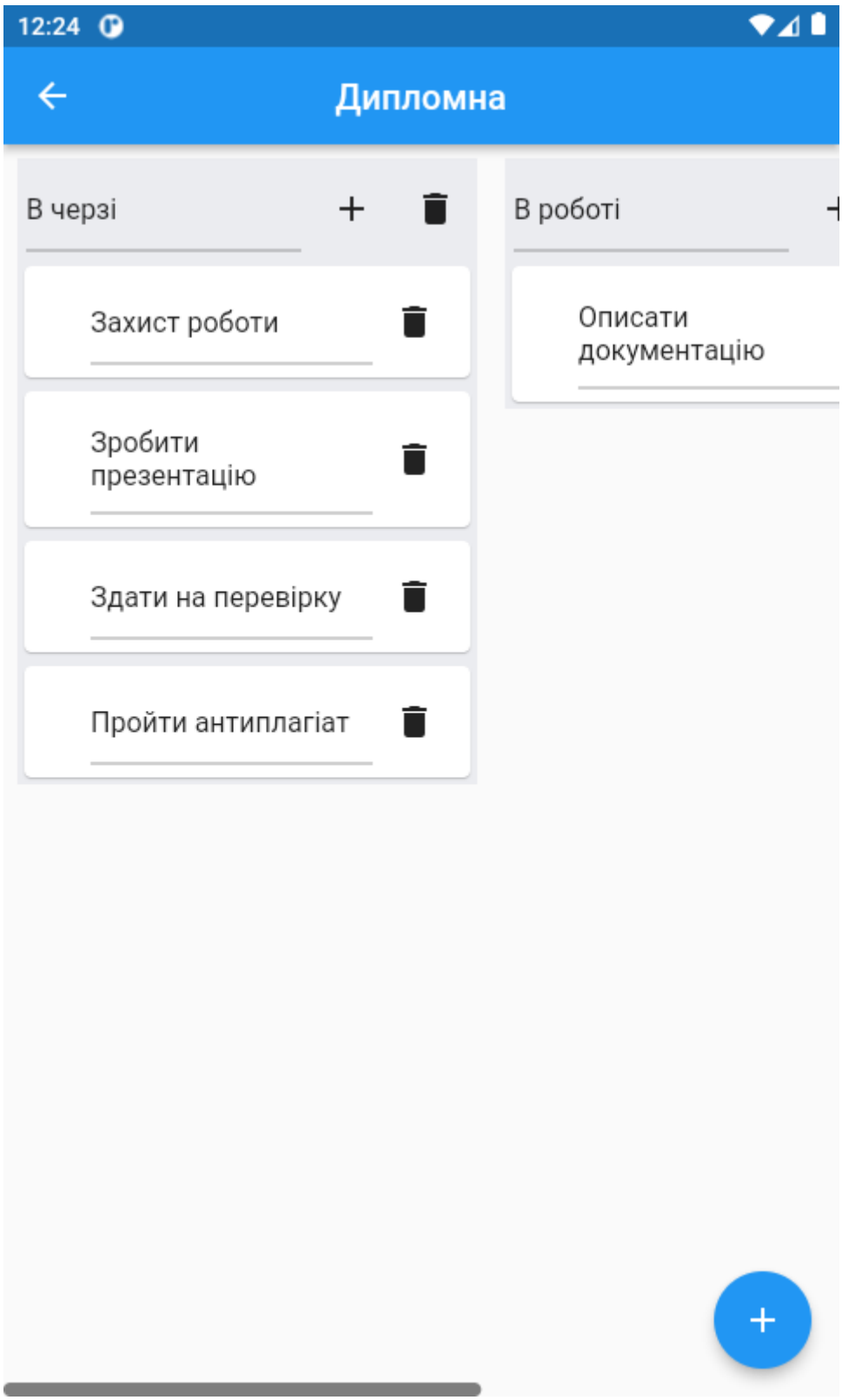

Рисунок 4.12 – Сторінка дошки

По мірі виконання задач, карточки з ними перетягуються з колонки в колонку (рис. 4.13):

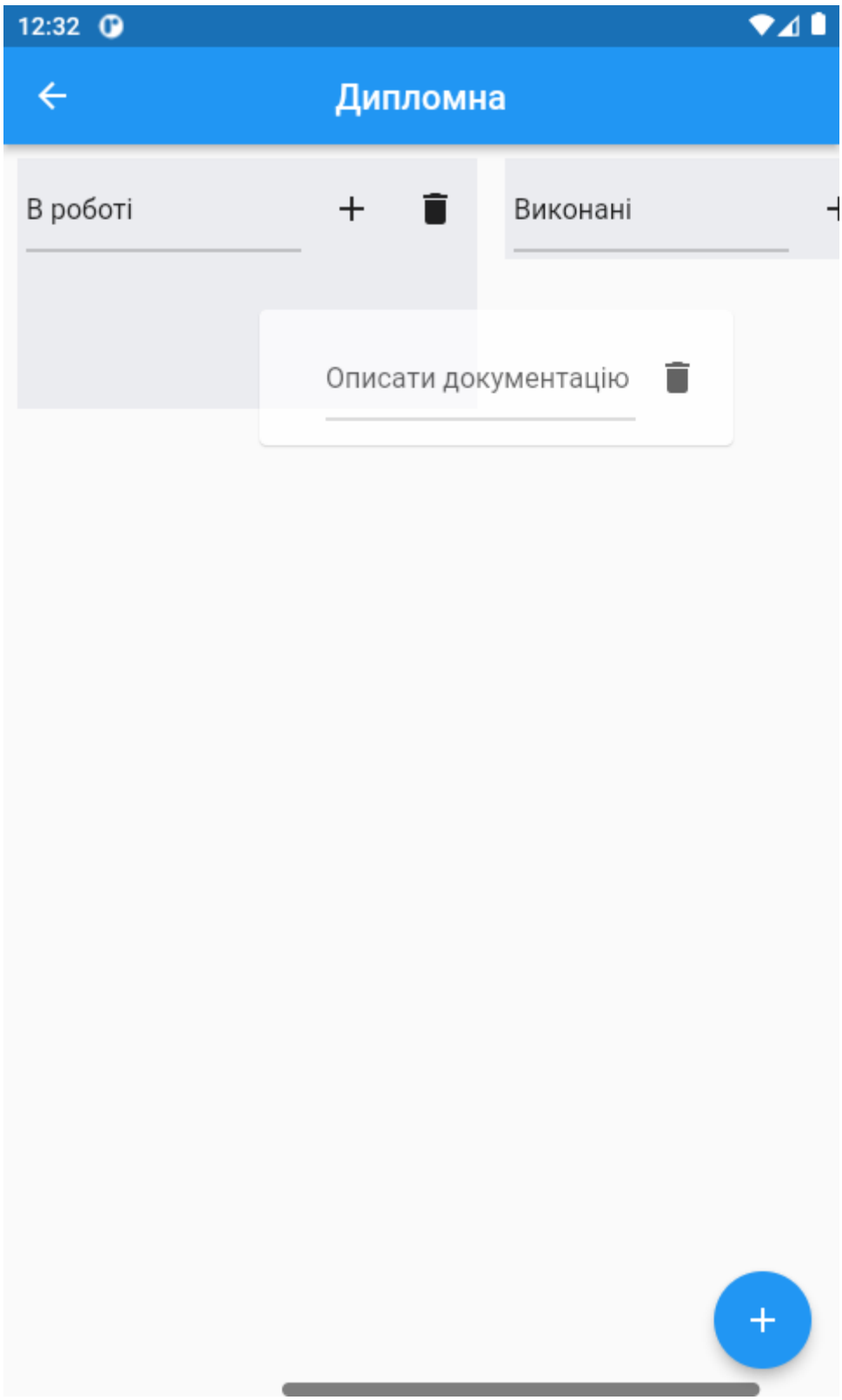

Рисунок 4.13 – Сторінка дошки (процес перетягування картки)

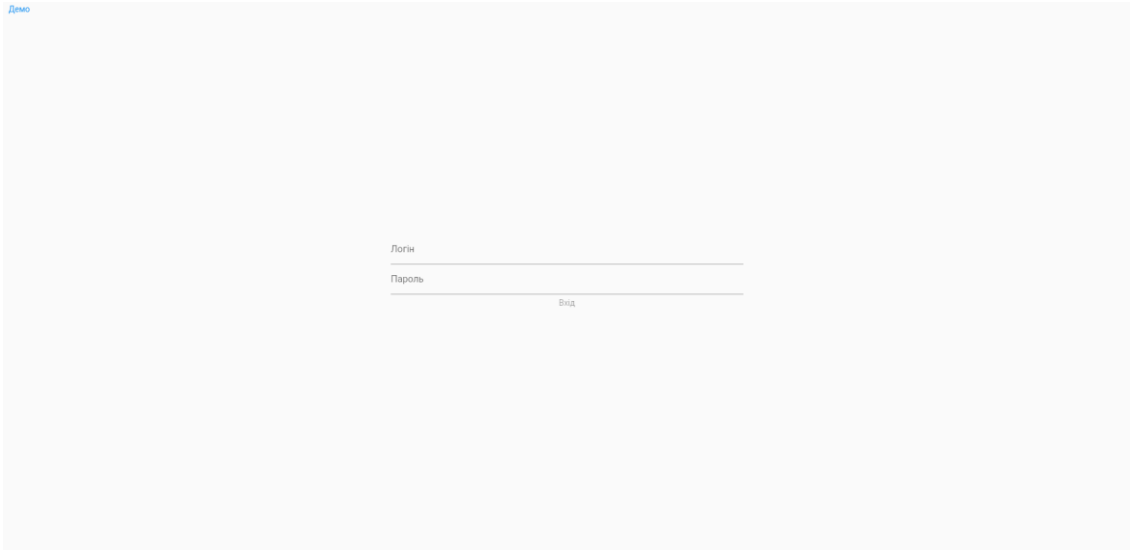

Рисунок 4.14 – Сторінка авторизації відкрита через комп`ютер

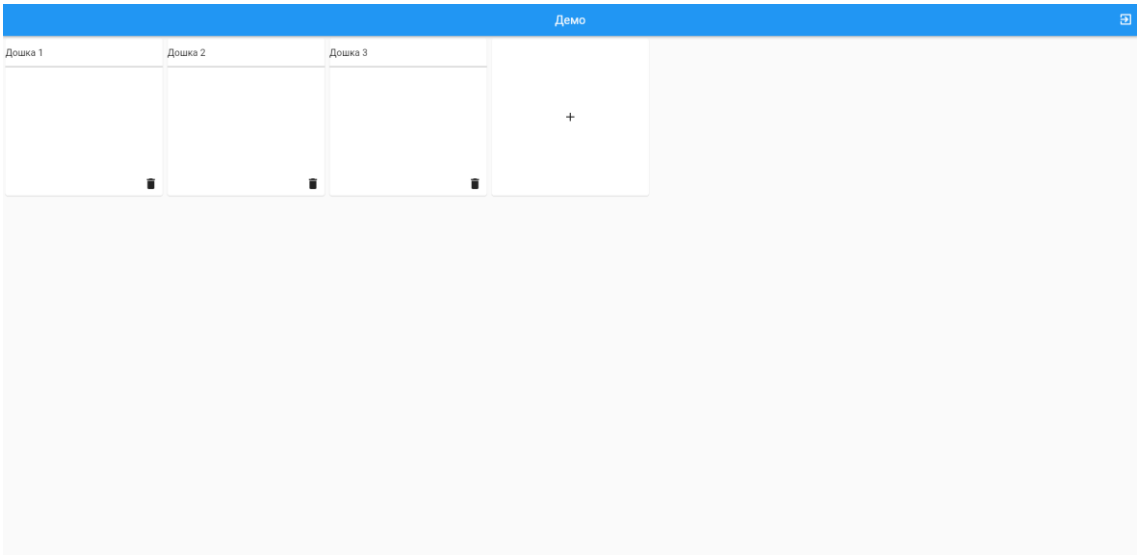

Рисунок 4.15 – Домашня сторінка відкрита через комп`ютер

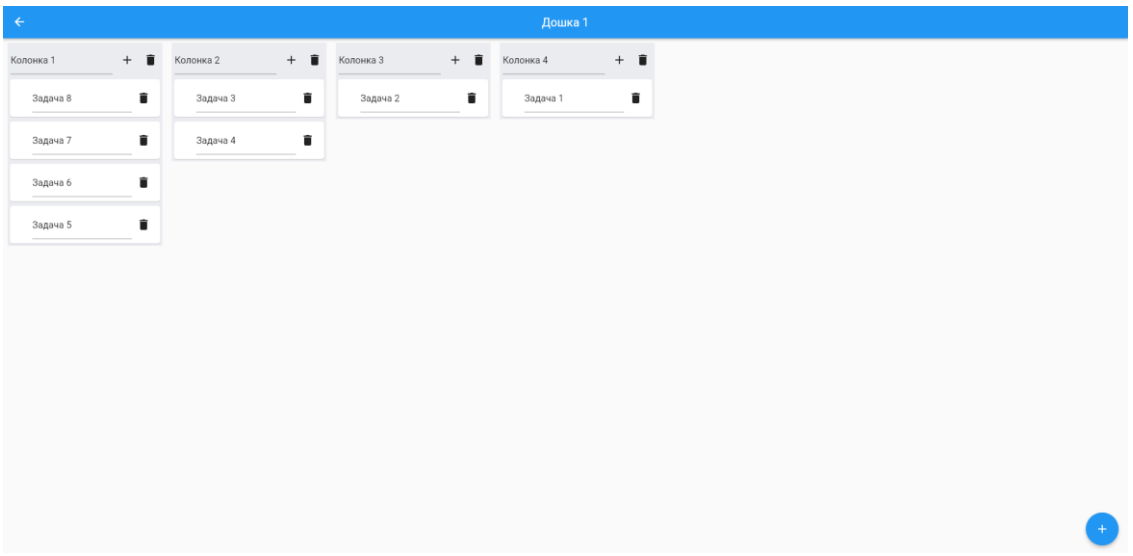

Рисунок 4.16 – Сторінка дошки відкрита через комп`ютер

4.4 Доопрацювання програми

Після демонстрації програмного продукту, було вирішено його доопрацювати, а саме:

1. Оновити дизайн з “Material 2” на “Material 3” (рис. 4.17):

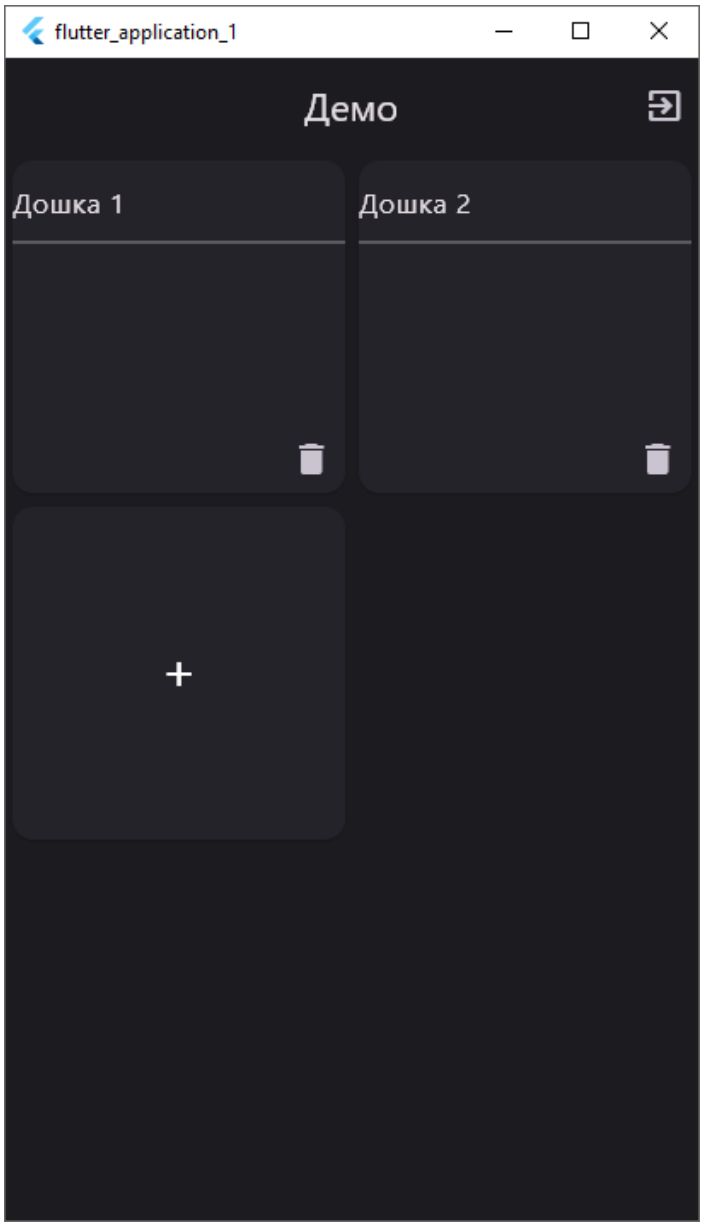

Рисунок 4.17 – Домашня сторінка в новому дизайні

2. Додати опис до карточки та зробити діалог для її створення та редагування (рис. 4.18):

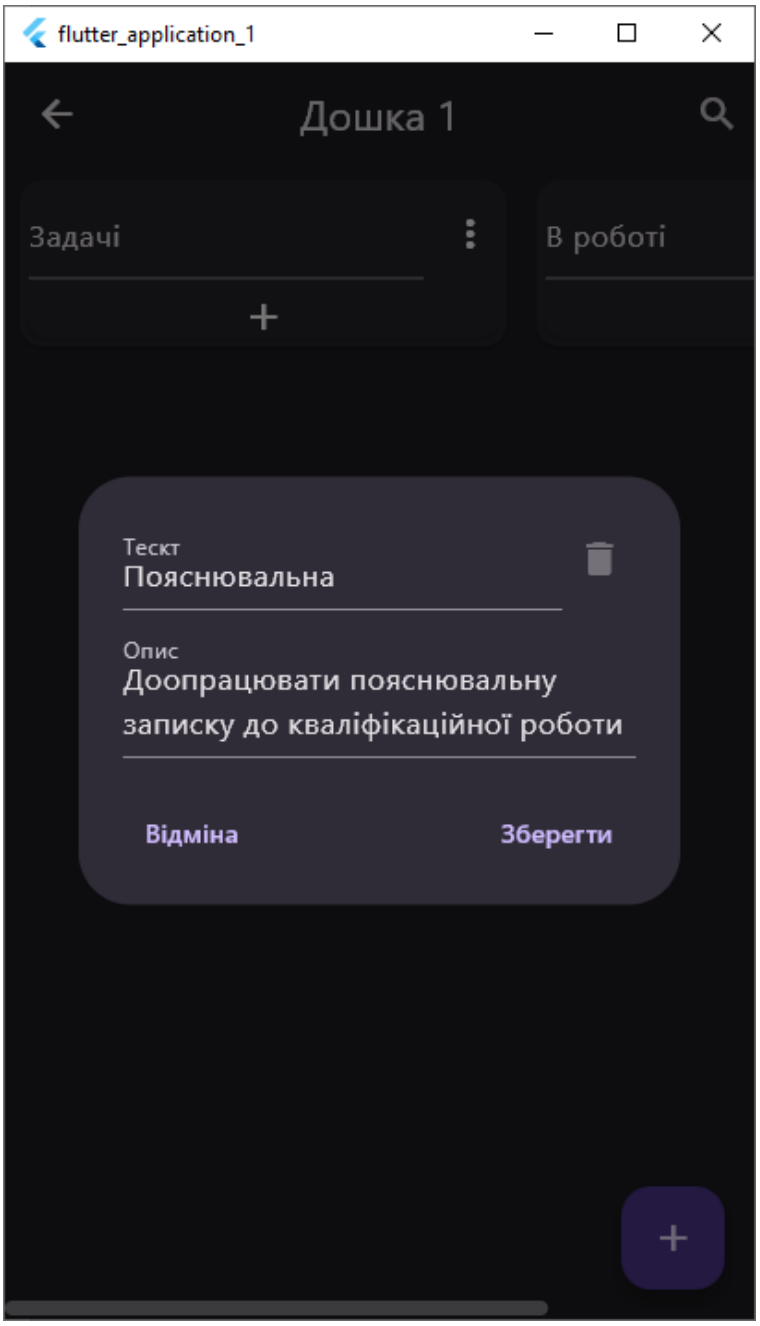

Рисунок 4.18 – Діалог для створення або редагування карточки

3. Додати пошук по назві карточки (рис. 4.19):

Рисунок 4.19 – Сторінка пошуку карточки по назві

4. Кнопки на секції згорнути в меню для подальшого розширення їх кількості (рис. 4.20):

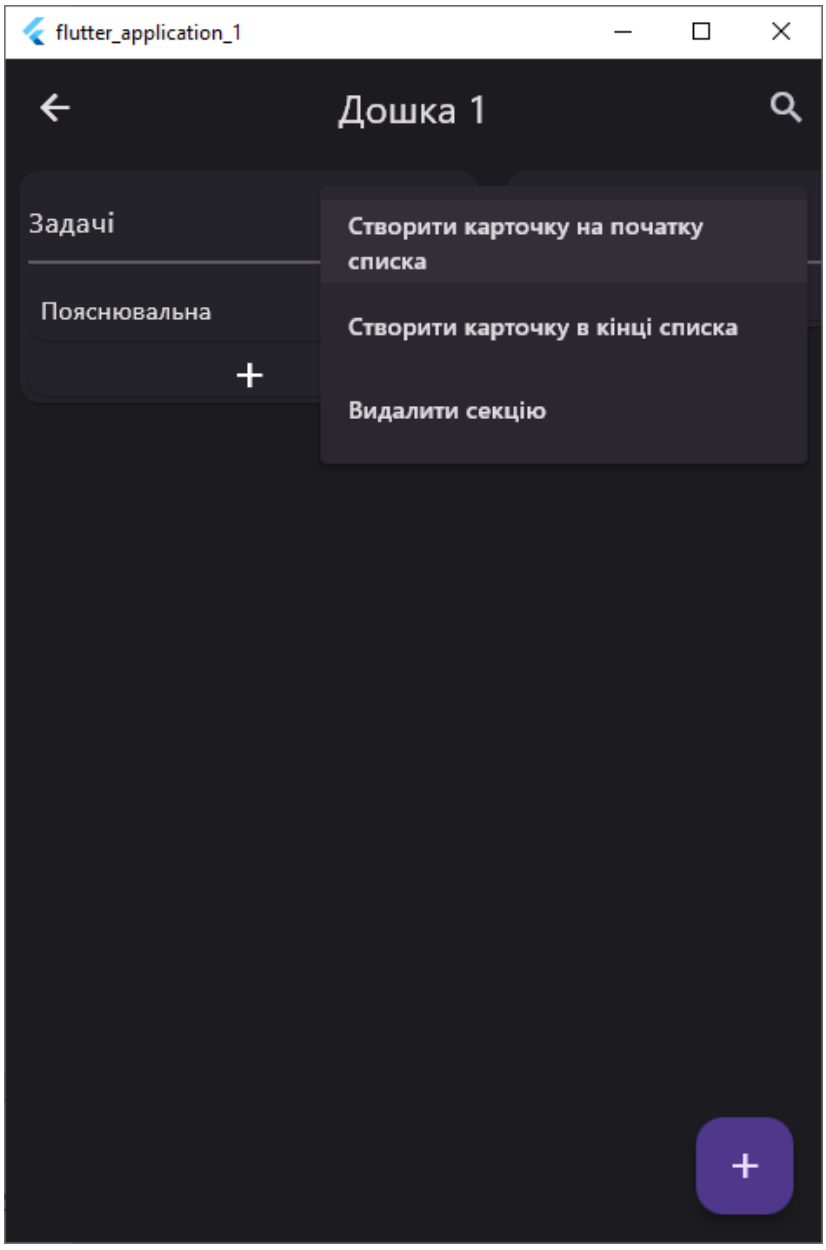

Рисунок 4.20 – Меню на секції

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є мобільний додаток, що реалізує канбан дошку.

Проект виконаний з наведенням актуальності обраної теми, коротким описом функціоналу та вимог продукту, оглядом існуючих подібних програм.

Під час виконання кваліфікаційної роботи було використано мову програмування Dart та фреймворк Flutter.

Основні результати роботи:

- сформульовані основні вимоги продукту, що розроблявся;
- проведено огляд плюсів та мінусі існуючих програм;
- розглянуто доступні у мережі продукти-аналоги;
- складено алгоритм роботи із створеним додатком;
- роботу апробовано: написано тези на науковий семінат "КНІТ";

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Ольховська О.В. Методичні рекомендації до виконання кваліфікаційної роботи для студентів за освітньою програмою «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» / О.В. Ольховська, О.О. Черненко. – Полтава: ПУЕТ, 2022. – 71 с.

2. Роберт К. Мартін. Чиста архітектура: пер. с англ. І. Бондар-Терещенко. — Київ: Фабула, 2019. — 368 с.

3. Олександр Швець. Занурення в Патерни Проектування [Електронний ресурс]: книга про принципи та патерни проектування / О. Швець. — Електрон. дані. – Режим доступу: https://refactoring.guru/uk/design-patterns/book. — Назва з екрана. — Дата звернення: 16.12.2022.

4. Створення додатку на Flutter: перші кроки [Електронний ресурс]: технічні статті. — Електрон. дані. – Режим доступу: https://dou.ua/lenta/articles/flutter-first-steps. Назва з екрана. — Дата звернення: 16.12.2022.

5. Методологія Kanban дошки, принципи та сервіси, які вам допоможуть [Електронний ресурс]: бізнес статті. — Електрон. дані. – Режим доступу: https://uaspectr.com/2021/01/26/shho-take-kanban. Назва з екрана. — Дата звернення: 16.12.2022.

6. Favro [Електронний ресурс]: сайт продукту. — Електрон. дані. – Режим доступу: https://www.favro.com/. Назва з екрана. — Дата звернення: 16.12.2022.

7. Asana [Електронний ресурс]: сайт продукту. — Електрон. дані. – Режим доступу: https://app.asana.com/. Назва з екрана. — Дата звернення: 16.12.2022.

8. Blossom [Електронний ресурс]: сайт продукту. — Електрон. дані. – Режим доступу: https://blossom.co/. Назва з екрана. — Дата звернення: 16.12.2022.

9. Trello [Електронний ресурс]: сайт продукту. — Електрон. дані. – Режим доступу: https://trello.com. Назва з екрана. — Дата звернення: 16.12.2022.

10. Hygger [Електронний ресурс]: сайт продукту. — Електрон. дані. – Режим доступу: https://hygger.io/. Назва з екрана. — Дата звернення: 16.12.2022.

11. Jira [Електронний ресурс]: сайт продукту. — Електрон. дані. – Режим доступу: https://www.atlassian.com/ru/software/jira. Назва з екрана. — Дата звернення: 16.12.2022.

12. Канбан-дошка для управління проектами і задачами [Електронний ресурс]: — Електрон. дані. – Режим доступу: https://nsoft-s.com/kanban-doshka.html. Назва з екрана. — Дата звернення: 16.12.2022.

13. Що таке канбан і чим він корисний? [Електронний ресурс]: бізнес статті. — Електрон. дані. – Режим доступу: https://worksection.com/ua/blog/kanban.html. Назва з екрана. — Дата звернення: 16.12.2022.

14. Dart – Вікіпедія [Електронний ресурс]: Вікіпедія. — Електрон. дані. – Режим доступу: https://uk.wikipedia.org/wiki/Dart. Назва з екрана. — Дата звернення: 16.12.2022.

ДОДАТКИ

Додаток А

```
Run | Debug | Profile
void main() async {
  final SL sl = await SL.instance();

  final UserRepository userRepository = sl();

  final User? user = await userRepository.getAuthorizedUser();

  runApp(App.create(sl: sl, user: user));
}
```

Рисунок А.1 – Код для ініціалізації та старту програми

```
abstract class SL {
  T call<T extends Object>();

  static Future<SL> instance() => SLGetIt.instance();
}

class SLGetIt extends SL {
  final GetIt _getIt;

  SLGetIt._({
    required GetIt getIt,
  }) : _getIt = getIt;

  static Future<SL> instance() async {
    WidgetsFlutterBinding.ensureInitialized();

    final GetIt getIt = GetIt.instance;

    getIt.registerLazySingleton(() => GlobalKey<NavigatorState>());

    getIt.registerLazySingleton<UserRepository>(() => UserRepositoryMock());

    return SLGetIt._(getIt: getIt);
  }

  @override
  T call<T extends Object>() => _getIt.call<T>();
}
```

Рисунок А.2 – Dependency injection

```dart
class App extends StatelessWidget {
  const App._();

  static Widget create({
    required SL sl,
    required User? user,
  }) =>
      MultiProvider(
        providers: [
          ///
          Provider.value(value: sl),

          ///
          ChangeNotifierProvider<AuthorizationProvider>(
            create: (BuildContext context) => AuthorizationProviderImpl(
              user: user,
              userRepository: sl(),
              navigatorKey: sl(),
            ),
          ),
        ],
        child: const App._(),
      );

  @override
  Widget build(BuildContext context) {
    final AuthorizationProvider authorizationProvider = context.read();

    final AuthorizationState state = authorizationProvider.state;

    return MaterialApp(
      ///
      debugShowCheckedModeBanner: false,

      ///
      navigatorKey: context.read<SL>()(),

      ///
      onGenerateRoute: (RouteSettings settings) {
        final Object? arguments = settings.arguments;

        if (arguments is RHSRoute) return arguments.route;

        return null;
      },

      ///
      onGenerateInitialRoutes: (_) => [
        state.call(
          authorizationNotState: (AuthorizationNotState state) => AuthorizationViewRout(context: context, authorizationNotState: state).route,
          authorizationDataState: (AuthorizationDataState state) => HomeViewRout(context: context, authorizationDataState: state).route,
        ),
      ],
    );
  }
}
```

Рисунок А.3 – Код ініціалізації стартового маршруту (стартової сторінки)

```
class AuthorizationView extends StatelessWidget {
  const AuthorizationView._();

  static Widget create({
    required AuthorizationViewModel Function(BuildContext) create,
  }) =>
      MultiProvider(
        providers: [
          ChangeNotifierProvider<AuthorizationViewModel>(create: create),
        ],
        child: const AuthorizationView._(),
      );

  @override
  Widget build(BuildContext context) {
    final AuthorizationViewModel authorizationViewModel = context.watch();

    return Scaffold(
      body: SafeArea(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: [
            ///
            Align(
              alignment: Alignment.centerLeft,
              child: TextButton(
                onPressed: authorizationViewModel.demo,
                child: const Text('Демо'),
              ),
            ),

            ///
            Padding(
              padding: const EdgeInsets.all(10),
              child: Column(
                children: const [
                  TextField(
                    decoration: InputDecoration(
                      label: Text('Логін'),
                    ),
                  ),
                  TextField(
                    decoration: InputDecoration(
                      label: Text('Пароль'),
                    ),
                  ),
                  TextButton(
                    onPressed: null,
                    child: Text('Вхід'),
                  ),
                ],
              ),
            ),

            ///
            const SizedBox(
              height: 40,
            ),
          ],
        ),
      ),
    );
  }
}
```

Рисунок А.4 – Код сторінки авторизації

```dart
abstract class AuthorizationState {
  final AuthorizationProvider authorizationProvider;

  AuthorizationState({
    required this.authorizationProvider,
  });

  factory AuthorizationState.fromUser({
    required User? user,
    required AuthorizationProvider authorizationProvider,
  }) =>
      user == null
          ? AuthorizationNotState(
              authorizationProvider: authorizationProvider,
              failure: null,
            )
          : AuthorizationDataState(user: user, authorizationProvider: authorizationProvider);

  UserRepository get userRepository => authorizationProvider.userRepository;

  T call<T>({
    required T Function(AuthorizationDataState value) authorizationDataState,
    required T Function(AuthorizationNotState value) authorizationNotState,
  }) {
    final AuthorizationState thisValue = this;

    if (thisValue is AuthorizationDataState) return authorizationDataState(thisValue);

    if (thisValue is AuthorizationNotState) return authorizationNotState(thisValue);

    throw thisValue;
  }
}

class AuthorizationDataState extends AuthorizationState {
  final User user;

  AuthorizationDataState({
    required this.user,
    required super.authorizationProvider,
  });

  void exit() {
    authorizationProvider.setState(AuthorizationNotState(authorizationProvider: authorizationProvider, failure: null));
  }
}

class AuthorizationNotState extends AuthorizationState {
  final Failure? failure;

  AuthorizationNotState({
    required super.authorizationProvider,
    required this.failure,
  });

  void authorization(Authorization authorization) async {
    final AuthorizationResult authorizationResult = await userRepository.authorization(authorization);

    authorizationResult.call(
      ///
      failure: (Failure failure) =>
          authorizationProvider.setState(AuthorizationNotState(failure: failure, authorizationProvider: authorizationProvider)),

      ///
      data: (User data) => authorizationProvider.setState(AuthorizationDataState(authorizationProvider: authorizationProvider, user: data)),
    );
  }
}
```

Рисунок А.5 – Код стану та логіки авторизації

```dart
class HomeView extends StatelessWidget {
  const HomeView._();

  static Widget create({
    required DataProvider Function(BuildContext) createDataProvider,
    required HomeViewModel Function(BuildContext) createHomeViewModel,
  }) =>
      MultiProvider(
        providers: [
          ///
          ChangeNotifierProvider<DataProvider>(create: createDataProvider),

          ///
          ChangeNotifierProvider<HomeViewModel>(create: createHomeViewModel),
        ],
        child: const HomeView._(),
      );

  @override
  Widget build(BuildContext context) {
    final MediaQueryData mediaQueryData = MediaQuery.of(context);
    final double screenWidth = mediaQueryData.size.width;

    final HomeViewModel homeViewModel = context.watch();

    final User user = homeViewModel.user;

    final List<Board> boards = user.boards;

    int crossAxisCount = screenWidth ~/ 250;

    if (crossAxisCount == 0) {
      crossAxisCount = 1;
    }

    Widget title() => user.call(
          userDemo: (UserDemo value) => const Text('Демо'),
          userServer: (UserServer value) => Text(value.login),
        );

    AppBar appBar() => AppBar(
          centerTitle: true,
          title: title(),
          actions: const [
            ExitWdget(),
          ],
        );

    return Scaffold(
      ///
      appBar: appBar(),

      ///
      body: GridView.count(
        ///
        crossAxisCount: crossAxisCount,

        ///
        children: [
          ///
          ...boards.map(
            (Board e) => BoardWidget.create(
              board: e,
            ),
          ),

          ///
          Card(
            child: IconButton(
              onPressed: homeViewModel.addBoard,
              icon: const Icon(Icons.add),
            ),
          ),
        ],
      ),
    );
  }
}
```

Рисунок А.6 – Код домашньої сторінки

```
abstract class HomeViewModel with ChangeNotifier {
  User get user;
  void addBoard();
  void removeBoard(Board value);
  FocusNode getFocusNode(Board board);
}

class HomeViewModelImpl with ChangeNotifier implements HomeViewModel {
  final DataProvider dataProvider;

  final Map<String, FocusNode> _focusNodes = {};

  HomeViewModelImpl({
    required this.dataProvider,
  });

  @override
  User get user => dataProvider.user;

  UserBoard get userBoard => dataProvider.userBoard;

  @override
  FocusNode getFocusNode(Board board) => _focusNodes.putIfAbsent(board.uid, () => FocusNode());

  String getUid() => dataProvider.getUid();

  @override
  void addBoard() {
    final Board board = Board.createEmpty(uid: getUid(), user: userBoard);

    user.addBoard(board);
    notifyListeners();

    WidgetsBinding.instance.addPostFrameCallback((_) {
      _focusNodes[board.uid]!.requestFocus();
    });
  }

  @override
  void removeBoard(Board value) {
    user.removeBoard(value);
    notifyListeners();
  }
}
```

Рисунок А.7 – Код логіки домашньої сторінки

```
class MyBoardView extends StatelessWidget {
  const MyBoardView._();

  static Widget create({
    required Board board,
    required DataProvider dataProvider,
  }) =>
      MultiProvider(
        providers: [
          ChangeNotifierProvider<DataProvider>.value(value: dataProvider),

          ///
          ChangeNotifierProvider<BoardViewModel>(
            create: (BuildContext context) => BoardViewModelImpl(
              board: board,
              dataProvider: context.read(),
            ),
          ),
        ],
        child: const MyBoardView._(),
      );

  @override
  Widget build(BuildContext context) {
    final BoardViewModel boardViewModel = context.watch();
    final Board board = boardViewModel.board;

    AppBar getAppBar() => AppBar(
          centerTitle: true,
          title: Text(board.name),
        );

    FloatingActionButton getFloatingActionButton() => FloatingActionButton(
          onPressed: boardViewModel.addSection,
          child: const Icon(Icons.add),
        );

    BoardItem getBoardItem(Section section, CardItem cardItem, int index) => BoardItem(
          onDropItem: (
            int? listIndex,
            int? itemIndex,
            int? oldListIndex,
            int? oldItemIndex,
            BoardItemState state,
          ) {
            boardViewModel.dragAndDrop(
              fromSection: section,
              toSection: board.sections[listIndex!],
              fromSectionIndex: index,
              toSectionIndex: itemIndex!,
            );
          },
          item: CardWidget(
            key: Key(cardItem.uid),
            cardItem: cardItem,
            index: index,
            section: section,
          ),
        );

    BoardList getBoardList(Section section) => BoardList(
          headerBackgroundColor: const Color.fromARGB(255, 235, 236, 240),
          backgroundColor: const Color.fromARGB(255, 235, 236, 240),
          header: [
            Expanded(
              child: Padding(
                padding: const EdgeInsets.all(5),
                child: TextField(
                  focusNode: boardViewModel.getFocusNodeSection(section),
                  controller: TextEditingController(text: section.name),
                  onChanged: (String value) => section.setName(value),
                ),
              ),
            ),
            IconButton(
              onPressed: () => boardViewModel.addCardStart(section),
              icon: const Icon(Icons.add),
            ),
            IconButton(
              onPressed: () => boardViewModel.removeSection(section),
              icon: const Icon(Icons.delete),
            ),
          ],
          items: section.cards.mapIndexed((int index, CardItem e) => getBoardItem(section, e, index)).toList(),
        );

    Widget body() => BoardView(
          scrollbar: true,
          lists: board.sections.map((Section e) => getBoardList(e)).toList(),
          boardViewController: boardViewModel.boardViewController,
        );

    return Scaffold(
      appBar: getAppBar(),
      body: body(),
      floatingActionButton: getFloatingActionButton(),
    );
  }
}
```

Рисунок А.8 – Код сторінки обраної дошки

```dart
abstract class BoardViewModel with ChangeNotifier {
  Board get board;
  BoardViewController get boardViewController;

  FocusNode getFocusNodeSection(Section section);

  void addSection();

  void removeSection(Section section);

  void dragAndDrop({
    required Section fromSection,
    required Section toSection,
    required int fromSectionIndex,
    required int toSectionIndex,
  });

  void setCard(Section section, CardItem card, int index);

  void addCardStart(Section section);

  void removeCard(Section section, int index);

  void addCardAnd(Section section);
}

class BoardViewModelImpl with ChangeNotifier implements BoardViewModel {
  @override
  final Board board;
  final DataProvider dataProvider;
  @override
  final BoardViewController boardViewController = BoardViewController();
  final Map<String, FocusNode> _mapFocusNodesStrings = {};

  BoardViewModelImpl({
    required this.board,
    required this.dataProvider,
  });

  @override
  FocusNode getFocusNodeSection(Section section) => _mapFocusNodesStrings.putIfAbsent(section.uid, () => FocusNode());

  @override
  void setCard(Section section, CardItem card, int index) {
    section.putCard(card, index);

    notifyListeners();
  }

  @override
  void addSection() {
    final Section section = Section.createEmpty(uid: dataProvider.getUid());

    board.addSection(section);

    notifyListeners();

    Future.delayed(
      const Duration(milliseconds: 100),
      () => WidgetsBinding.instance.addPostFrameCallback((_) {
        boardViewController.animateTo(
          board.sections.length,
          duration: const Duration(milliseconds: 100),
          curve: Curves.bounceOut,
        );

        _mapFocusNodesStrings[section.uid]!.requestFocus();
      }),
    );
  }

  @override
  void removeSection(Section section) {
    board.removeSection(section);

    notifyListeners();
  }

  @override
  void dragAndDrop({
    required Section fromSection,
    required Section toSection,
    required int fromSectionIndex,
    required int toSectionIndex,
  }) {
    final CardItem card = fromSection.cards[fromSectionIndex];

    board.dragAndDrop(
      fromSection: fromSection,
      toSection: toSection,
      fromSectionIndex: fromSectionIndex,
      toSectionIndex: toSectionIndex,
      card: card,
    );

    notifyListeners();
  }

  @override
  void addCardStart(Section section) {
    section.addCardStart(CardItem.createEmpty(uid: dataProvider.getUid()));

    notifyListeners();
  }

  @override
  void addCardAnd(Section section) {
    section.addCardAnd(CardItem.createEmpty(uid: dataProvider.getUid()));

    notifyListeners();
  }

  @override
  void removeCard(Section section, int index) {
    section.removeCard(index);

    notifyListeners();
  }
}
```

Рисунок А.9 – Код логіки сторінки обраної дошки