

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## **ПОЯСНЮВАЛЬНА ЗАПИСКА**

до кваліфікаційної роботи освітнього ступеня «бакалавр»  
за спеціальністю 121 «Інженерія програмного забезпечення»  
(освітня програма «Інженерія програмного забезпечення»)  
на тему:

«WEB додаток для управління IT проектами  
(канбан дошка, аналог trello)»

Виконав студент групи ІПЗК-20-1  
ПОПЛАВСЬКИЙ Володимир Сергійович

Керівник роботи:  
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:  
ВЛАСЕНКО Олег Васильович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**«ЗАТВЕРДЖУЮ»**

В.о. зав. кафедри інженерії  
програмного забезпечення  
Андрій МОРОЗОВ  
«14» лютого 2023 р.

**ЗАВДАННЯ**  
**на кваліфікаційну роботу**

Здобувач вищої освіти: **ПОПЛАВСЬКИЙ Володимир Сергійович**

Керівник роботи: **ЧИЖМОТРЕ Олена Геннадіївна**

Тема роботи: «WEB додаток для управління ІТ проектами(канбан дошка, аналог trello»,

затверджена Наказом закладу вищої освіти від «14» лютого 2023 р., №70с

Вихідні дані для роботи: **процес управління ІТ проектами за допомогою подібних додатків та застосування сучасних технологій та аналіз їх властивостей.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

| Розділ | Консультант     | Завдання видав | Завдання прийняв |
|--------|-----------------|----------------|------------------|
| 1      | Чижмотря О.Г.,  | 16.02.2023р.   | 16.02.2023р.     |
| 2      | Чижмотря О.Г.,  | 16.02.2023р.   | 16.02.2023р.     |
| 3      | Чижмотря О.Г. , | 16.02.2023р.   | 16.02.2023р.     |

### Календарний план

| № з/п | Назва етапів випускної кваліфікаційної роботи | Термін виконання етапів роботи    | Примітка |
|-------|-----------------------------------------------|-----------------------------------|----------|
| 1     | Постановка завдання                           | 16 лютого 2023 – 18 лютого 2023   | виконано |
| 2     | Аналіз аналогів програмного продукту          | 19 лютого 2023 – 28 лютого 2023   | виконано |
| 3     | Визначення вимог до додатку                   | 1 березня 2023 – 8 березня 2023   | виконано |
| 4     | Вибір інструментальних засобів                | 8 березня 2023 – 15 березня 2023  | виконано |
| 5     | Визначення варіантів використання             | 15 березня 2023 – 16 березня 2023 | виконано |
| 6     | Проектування та реалізація алгоритмів системи | 16 березня 2023 – 31 березня 2023 | виконано |
| 7     | Написання програмного коду                    | 1 квітня 2023 – 2 травня 2023     | виконано |
| 8     | Попередній захист                             | 5 травня 2023                     | виконано |
| 9     | Тестування роботи програмного продукту        | 2 травня 2023 – 11 травня 2023    | виконано |
| 10    | Оформлення пояснювальної записки              | 11 травня 2023 – 30 травня 2023   | виконано |

## РЕФЕРАТ

Випускна робота бакалавра складається з Web застосунку для управління ІТ проектами та пояснювальної записки. Пояснювальна записка до випускної роботи містить 76 сторінок, 66 ілюстрацій, 1 таблицю, 1 додаток та 14 використаних джерел.

Метою роботи є розробка Web застосунку для управління ІТ проектами, в основі якого лежить канбан методологія.

В роботі було визначено основні завдання на розробку Web застосунку, проаналізовано додатки аналоги, визначено основний функціонал та вимоги до системи. Розглянуто Kanban методологію, та було обрано технології розробки Web застосунку, спроектовано та реалізовано алгоритми роботи системи. Програмний продукт був протестований.

**КЛЮЧОВІ СЛОВА:** ІТ, КАНБАН, WEB, УПРАВЛІННЯ ПРОЕКТАМИ.

|           |      |                  |        |      |                                                                                                     |                                             |      |         |  |
|-----------|------|------------------|--------|------|-----------------------------------------------------------------------------------------------------|---------------------------------------------|------|---------|--|
|           |      |                  |        |      | ІПЗ.КР.Б – 121 – 23 - ПЗ                                                                            |                                             |      |         |  |
| Зм.       | Арк. | № докум.         | Підпис | Дата | Пояснювальна записка<br>WEB додаток для управління ІТ<br>проектами<br>(канбан дошка, аналог trello) | Літ.                                        | Арк. | Аркушів |  |
| Розроб.   |      | В.С. Поплавський |        |      |                                                                                                     |                                             |      |         |  |
| Керівник. |      | О.Г. Чижмоторя   |        |      |                                                                                                     |                                             | 4    | 76      |  |
|           |      |                  |        |      |                                                                                                     | Житомирська політехніка,<br>група ІПЗК-20-1 |      |         |  |
| Реценз.   |      | О.В. Власенко    |        |      |                                                                                                     |                                             |      |         |  |
| Зав. Каф. |      | А.В. Морозов     |        |      |                                                                                                     |                                             |      |         |  |

## ABSTRACT

The bachelor's thesis consists of a web application for IT project management and an explanatory note. The explanatory note for the thesis contains 76 pages, 66 illustrations, 1 table, 1 appendix, and a list of 14 used references.

The aim of the work is to develop a web application for IT project management based on the Kanban methodology.

The main tasks for developing the web application were defined, analogous applications were analyzed, and the core functionality and system requirements were determined. The Kanban methodology was examined, and the development technologies for the web application were chosen. Algorithms for the system's operation were designed and implemented. The software product was tested.

KEYWORDS: IT, Kanban, web, project management.

## ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

MVVM – Model-View-ViewModel, шаблон проектування архітектури додатку.

IT - інформаційні технології.

ПМ - проектні менеджери.

BaaS - Backend as a service - тип платформи Firebase.

FB - Firebase.

API – Application Programming Interface (Прикладний програмний інтерфейс).

REST – Representational State Transfer (передача репрезентативного стану).

NoSQL - нереляційна база даних.

## ЗМІСТ

|                                                                                                                         |    |
|-------------------------------------------------------------------------------------------------------------------------|----|
| ВСТУП                                                                                                                   | 8  |
| РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ ТА ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ВЕБ-ДОДАТКУ З УПРАВЛІННЯ ІТ ПРОЄКТАМИ | 9  |
| 1.1. Технічне завдання на розробку системи                                                                              | 9  |
| 1.2. Аналіз аналогів програмного продукту                                                                               | 12 |
| 1.3. Обґрунтування вибору методології для реалізації в додатку                                                          | 23 |
| 1.4. Інструментальні засоби реалізації програмного продукту                                                             | 24 |
| Висновки до першого розділу                                                                                             | 26 |
| РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ДЛЯ УПРАВЛІННЯ ІТ ПРОЄКТАМИ                                            | 27 |
| 2.1. Визначення варіантів використання системи                                                                          | 27 |
| 2.2. Алгоритми роботи системи                                                                                           | 28 |
| 2.3. Структура компонентів системи                                                                                      | 30 |
| 2.4. Реалізація веб-додатку                                                                                             | 32 |
| Висновки до другого розділу                                                                                             | 35 |
| РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ                                                                          | 36 |
| 3.1. Архітектура додатку та його розгортання                                                                            | 36 |
| 3.2. Інтерфейс та порядок роботи з системою                                                                             | 38 |
| 3.3. Тестування роботи веб-додатку                                                                                      | 41 |
| Висновки до третього розділу                                                                                            | 50 |
| ВИСНОВКИ                                                                                                                | 51 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ                                                                                              | 52 |
| ДОДАТКИ                                                                                                                 | 53 |

## ВСТУП

**Актуальність теми:** в сьогоднішніх реаліях кількість ІТ компаній стрімко росте, не тільки в Україні, але і усьому світі. Для того щоб вчасно виконувати проекти, працює багато проектних менеджерів, які в своїй роботі використовують різні програмні додатки, такі як Jira, Trello, MS Project, та інші.

Проте додаток створений не лише для ІТ ринку, але й для будь-якої діяльності, яку можна покласти в рамках методології Kanban. Отже, я вважаю, що тема на сьогодні більш ніж актуальна, також, я впевнений у тому, що актуальність цієї теми не буде спадати найближчі кілька років.

**Метою роботи:** є розробка web-додатку з управління ІТ проектами, за допомогою якого можна буде контролювати виконання завдань.

Поставлена мета обумовила необхідність виконання наступних завдань:

- проведення аналізу функціональних потреб веб-додатку;
- визначення архітектури та узагальненої структури веб-додатку;
- обґрунтування та вибір засобів реалізації веб-додатку;
- проектування структурних складових та алгоритмів роботи веб-додатку;
- реалізація веб-додатку.

**Об'єктом дослідження** є процес управління ІТ проектами за допомогою Kanban методології.

**Предмет дослідження:** функціональні можливості додатків для управління ІТ проектами за допомогою Kanban методології.

**Практичне завдання отриманих результатів:** створений WEB додаток для управління ІТ проектами, який буде містити основні функції для управління ІТ проектами.

**Структура та обсяг роботи.** Випускна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел, 1 таблиці та 66 ілюстрацій по тексту. Повний обсяг роботи становить 76 сторінок, з них 43 сторінки основного тексту.

# **РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ ТА ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ВЕБ-ДОДАТКУ З УПРАВЛІННЯ ІТ ПРОЄКТАМИ**

## **1.1. Технічне завдання на розробку системи**

Менеджмент в ІТ відіграє не останню роль. Саме ПМи дивляться за всім, що відбувається в проєкті, підкидають нові задачі та відповідають за результат перед замовником. І тому розробка саме такої системи полегшує роботу всій команді і ПМам в тому числі

Цей розділ буде присвячений вимогам до проєкту.

### **Призначення системи**

Програмний комплекс призначений для візуалізації робочого процесу розробки або підтримки існуючих додатків. Програма являє собою Kanban дошку на якій є колонки статусів з задачами і дозволяє створювати нові та рухати існуючі задачі.

### **Вимоги до системи**

Програмний продукт повинен володіти наступними функціональними характеристиками:

- Доступ до системи за допомогою web-інтерфейсів;
- Експорт різних звітів та аналітики;
- Можливість одночасної роботи із системою 25 і більше користувачам;
- Наявність двох типів облікових записів.

### **Вимоги до програмного забезпечення**

Програмний комплекс повинен бути побудований на клієнт-серверній архітектурі з використанням веб-технологій таких як JS та його фреймворки типу - vue.js. База даних пропонується - Firebase. При проєктуванні програмного комплексу необхідно застосувати архітектуру Model-View-ViewModel для розмежування алгоритмічної частини від інтерфейсу та забезпечення спільного використання компонентів. При розробці дозволяється застосовувати фреймворки.

Клієнтська частина повинна бути побудована на стандартних веб-технологіях, що не вимагають встановлення додаткових компонентів на комп'ютер (не рекомендується використовувати Flash або Java-аплети).

Сторінки інтерфейсу генеруються сервером за стандартом HTML5 [3], оформлення здійснюється за допомогою таблиць стилів CSS [3], а інтерактивність сторінок забезпечується скриптами Javascript. Інтерфейс повинен коректно працювати у сучасних веб-браузерах Safari, Mozilla FireFox та Google Chrome.

### **Вимоги до інтерфейсу**

Взаємодія користувачів із програмним комплексом повинна реалізовуватися за допомогою веб-інтерфейсів. Інтерфейс програмного комплексу повинен бути зрозумілим та зручним, не перевантаженим графічними елементами. Навігаційні елементи та компоненти інтерактивної взаємодії повинні бути виконані у зручній для користувача формі. Введення та виведення даних в систему, введення керуючих команд та результат їх виконання повинні виконуватися в інтерактивному режимі.

До інтерфейсів користувача висуваються наступні вимоги:

- мінімально можливий час відгуку системи;
- оптимізація інтерфейсів програмного комплексу з точки зору максимальної роботи із системою;
- уніфікація розміщення керуючих елементів;
- підтвердження виконання критичних дій в системі;
- мінімізація кількості полів, що заповнюються.

### **Функціональні вимоги**

1. Авторизація користувачів в системі: В системі повинна бути представлена можливість реєстрації користувача та присвоєння йому відповідної ролі (девелопер, дизайнер, проджект менеджер і т.д.).
2. Лейбли на картки: Можливість додати лейбл для картки щоб було видно це дизайн таск чи баг.

3. Можливість збереження інформації: Система повинна зберігати інформацію і надавати можливість користувачу керувати нею.
4. Можливість створювати репорти.

### **Системні вимоги**

1. Вимоги до середовища виконання: Система повинна задовольняти вимогам на комп'ютері, що знаходиться в наступній мінімальній комплектації:

- Intel Celeron n4000
- Операційна система Windows 10
- Вимоги до операційного середовища (для сервера UNIX, до web-сервера Apache).
- Браузери (Opera, Firefox, ...).
- Загальна освіта з інформатики для користувачів.

### **Нефункціональні вимоги**

#### 1. Сприйняття

- Час, потрібний для навчання інструментами роботи з інформаційною системою для звичайних користувачів – 2-3 години, а для досвідчених – 1 година
- Час відповіді системи для звичайних запитів не повинен перевищувати 5 секунд, а для більш складних запитів – 10 сек.
- Інтерфейс представлення ІС повинен бути інтуїтивно зручним для користувача та не вимагати від нього додаткової підготовки.

#### 2. Надійність

- Доступність – час, потрібний для обслуговування системи не повинен перевищувати 10% від загального часу роботи.
- Середній час безперервної роботи – 20 робочих днів.
- Максимальна норма помилок та дефектів в роботі системи – 1 помилка на 1000 запитів користувача.

#### 3. Можливість експлуатації

- Масштабування – система повинна мати можливість збільшувати потужності (продуктивність), зі збільшенням користувачів таким

чином, щоб це аж ні як негативно не відобразилося на її роботі.

- Оновлення версій – Оновлення версій повинно здійснюватися в автоматичному режимі залежно від уподобань користувачів та розширення списку пропонуємих послуг.

## 1.2. Аналіз аналогів програмного продукту

Дослідивши мережу інтернет, поспілкувавшись з проектними менеджерами та розробниками, та з особистого досвіду - можна виділити основні програмні додатки - Jira та Trello, проте не потрібно забувати про такі додатки як Asana та Wrike.

За моїми спостереженнями найбільш популярним додатком для таких функцій є - Jira. Тому давайте розглянемо Jira більш детально.

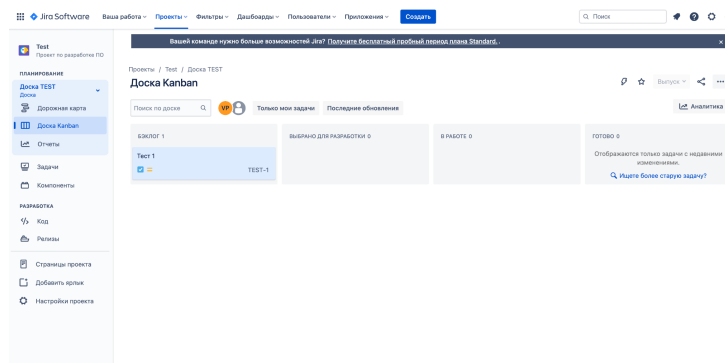


Рис. 1.1 - Головна сторінка з дошкою Jira

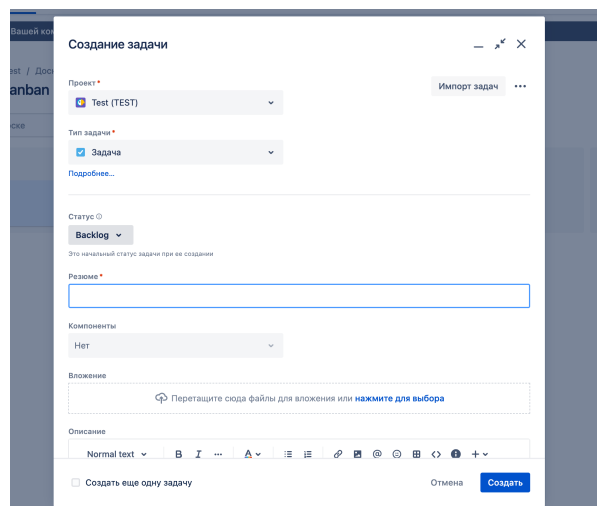


Рис. 1.2 - Створення нової картки Jira

## Доска Kanban

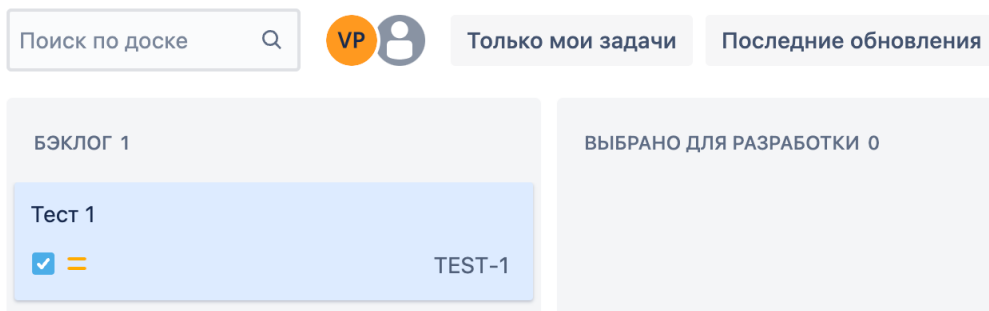


Рис. 1.3 - Видяд картки на дошці Jira

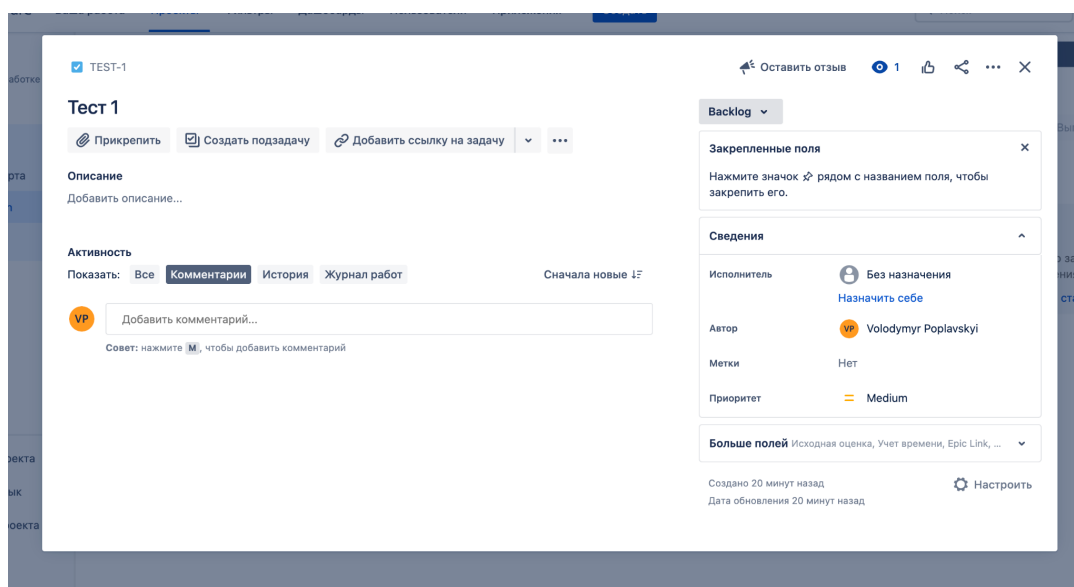


Рис. 1.4 - Видяд попапа картки Jira

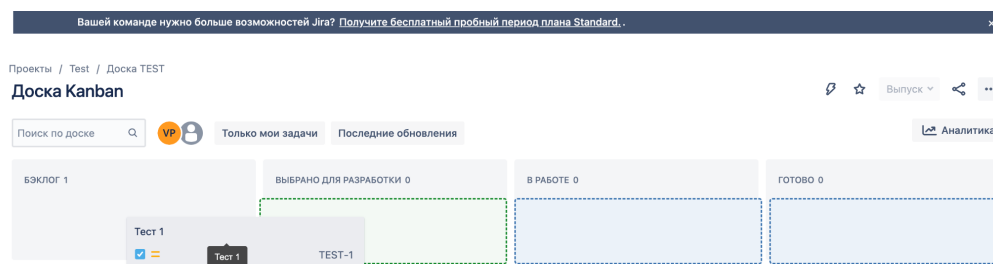


Рис. 1.5 - Перетягування картки по дошці Jira

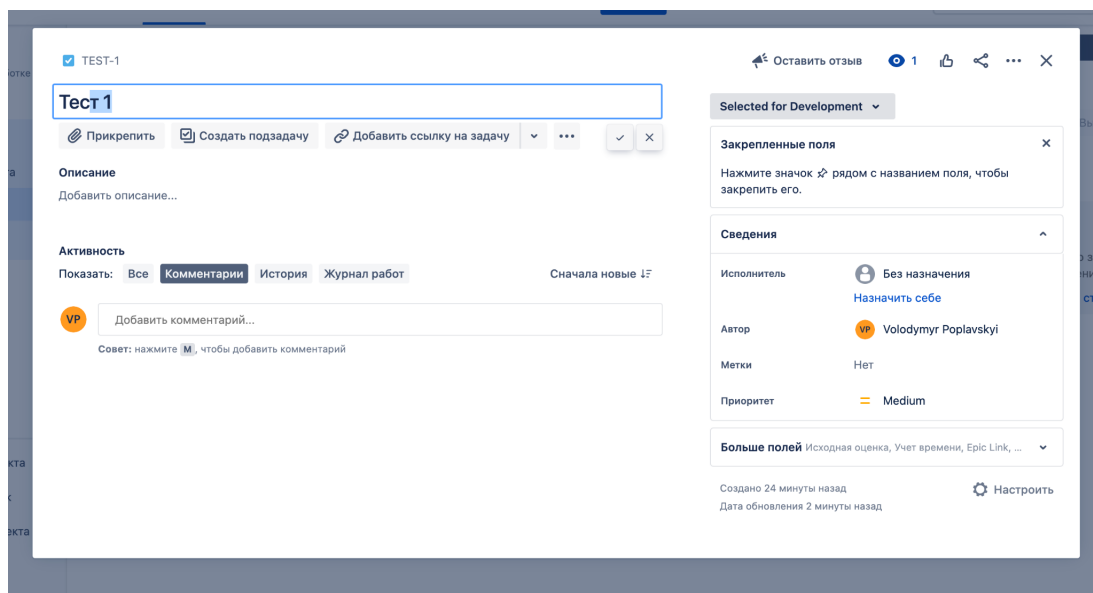


Рис. 1.6 - Оновлення полів в картці Jira

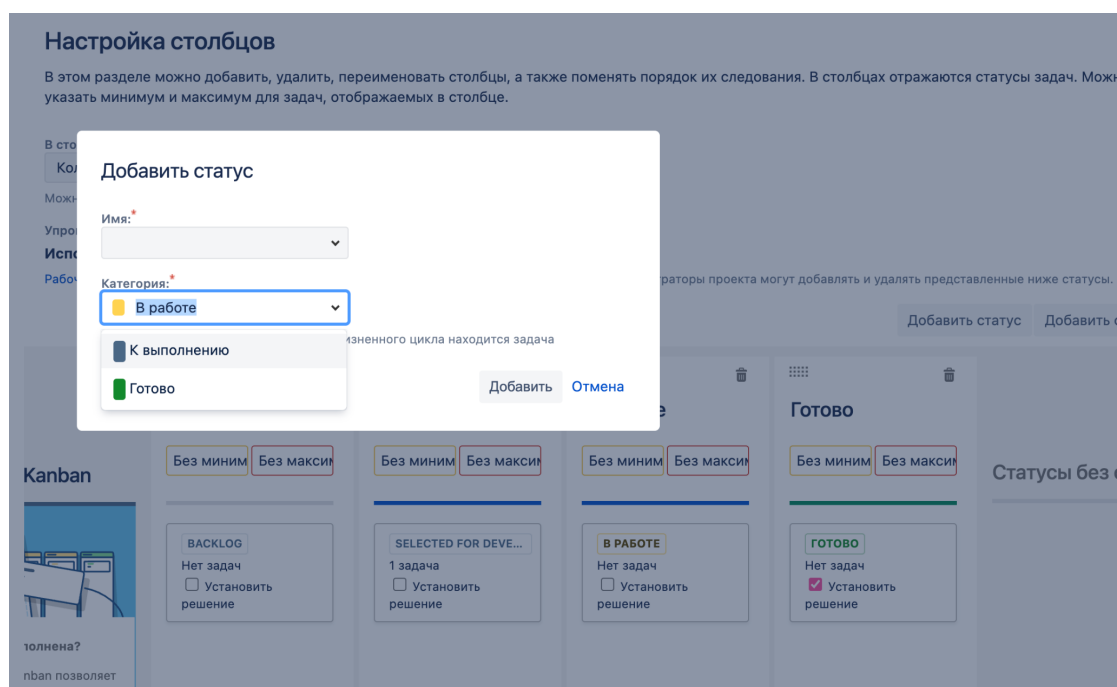


Рис. 1.7 - Додавання колонки в Jira

Після дослідження Jira, можна зробити висновок, що це дуже високо-функціональний продукт, в якому є велика кількість різноманітного функціоналу, не тільки такого як зазначено раніше, а й великої кількості іншого функціоналу, такого як створення бордів для Kanban і Scrum, можна присвоїти картки виконавцю, створення різноманітних репортів, використання різних плагінів для створення різноманітних інформативних

діаграм та графіків. Що робить з Jira справжнього титана серед додатків з управління ІТ проектами. [6]

Тепер можна перейти до огляду Trello.

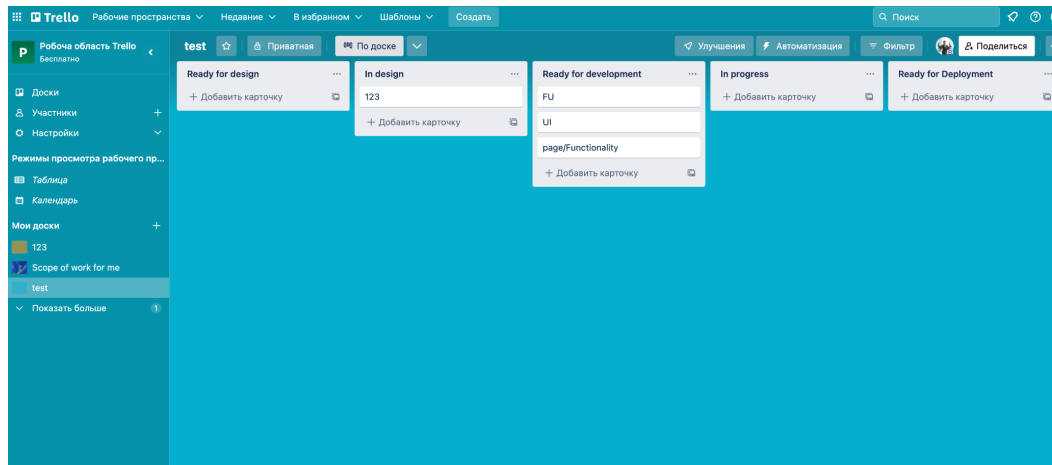


Рис. 1.8 - Головна сторінка з дошкою Trello

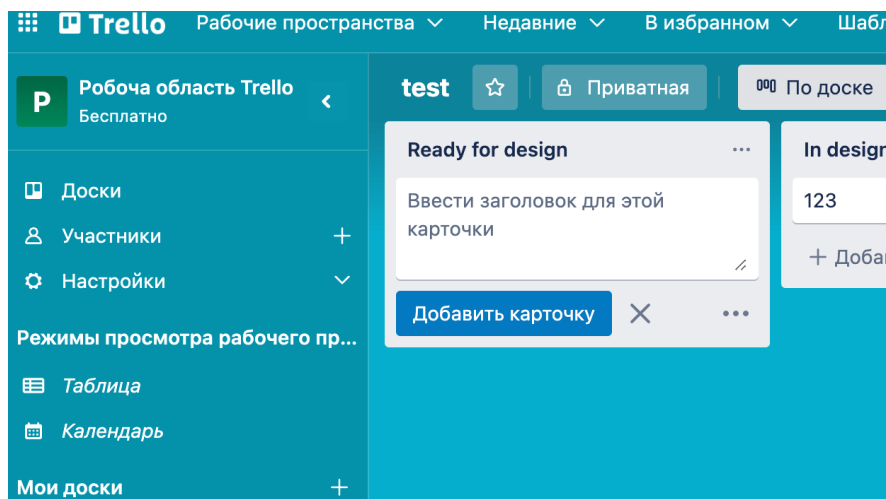


Рис. 1.9 - Створення нової картки Trello

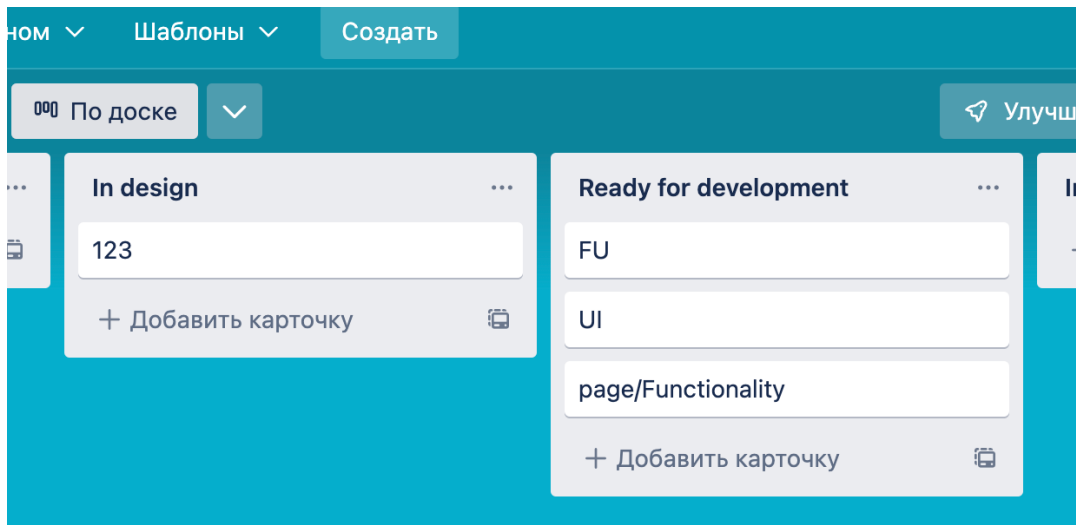


Рис. 1.10 - Вигляд карток Trello

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |

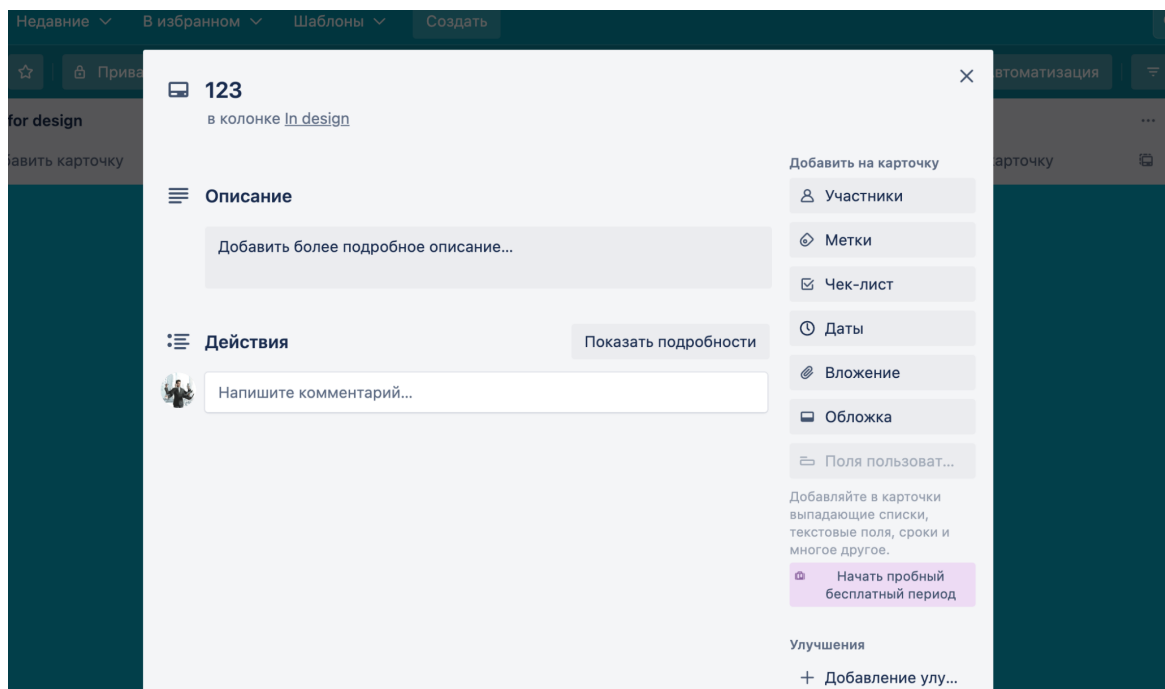


Рис. 1.11 - Попап картки Trello

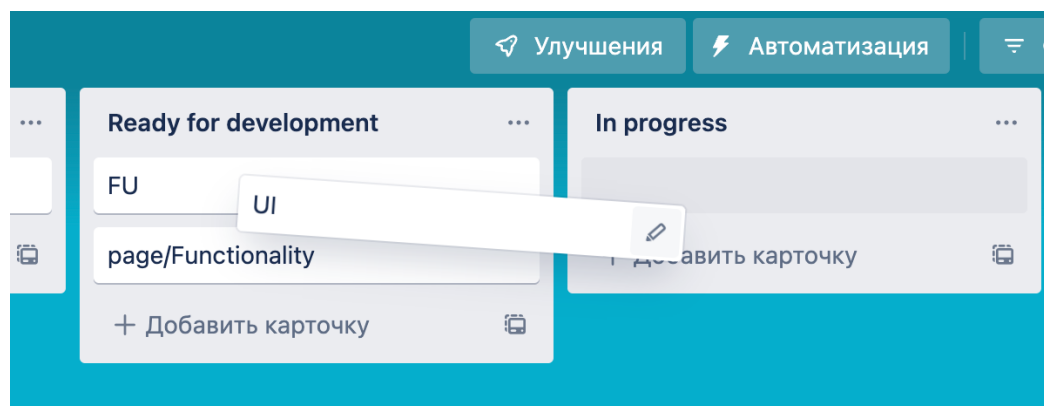


Рис. 1.12 - Перетягивання картки Trello

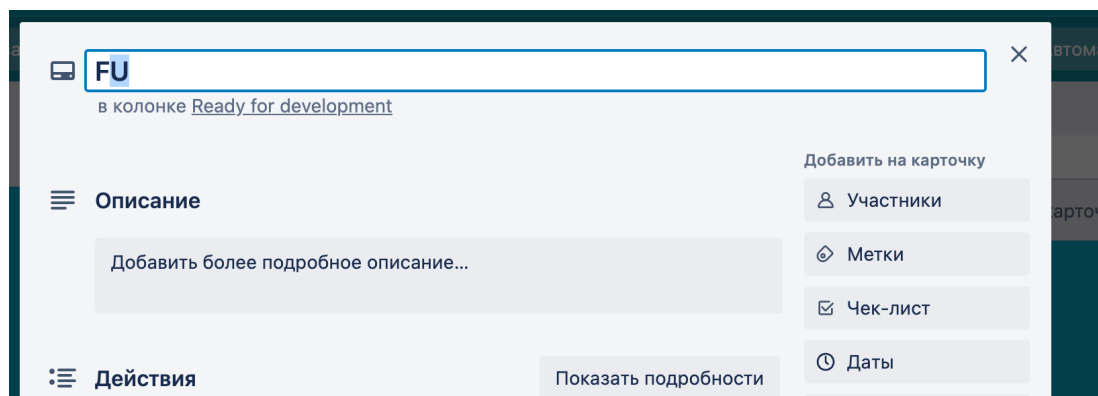


Рис. 1.13 - Оновлення полів картки Trello

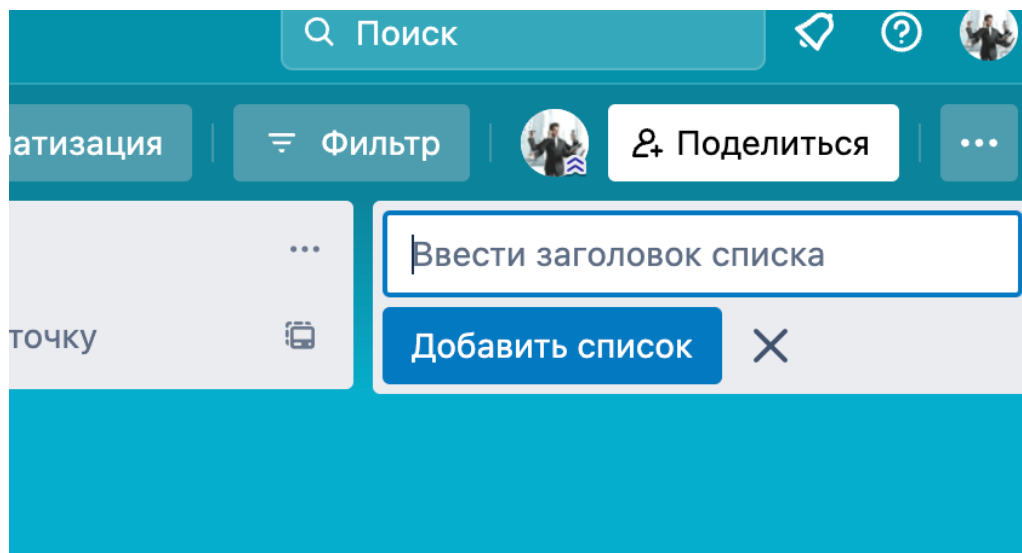


Рис. 1.14 - Додавання нового стовпця Trello

Після дослідження цього додатку можна з упевненістю сказати, що даний продукт має вже трохи менше функціоналу ніж попередній.

Наприклад, немає можливості обирати між Scrum та Kanban. Також цей продукт має дійсно цікавий та корисний функціонал для досвідчених менеджерів - Створення кнопок для автоматизації. Як висновок, можна сказати, що продукт є трохи простішим ніж попередній, але все ще містить велику кількість складного функціоналу. [7]

Можемо рухатись далі і розглянути Asana.

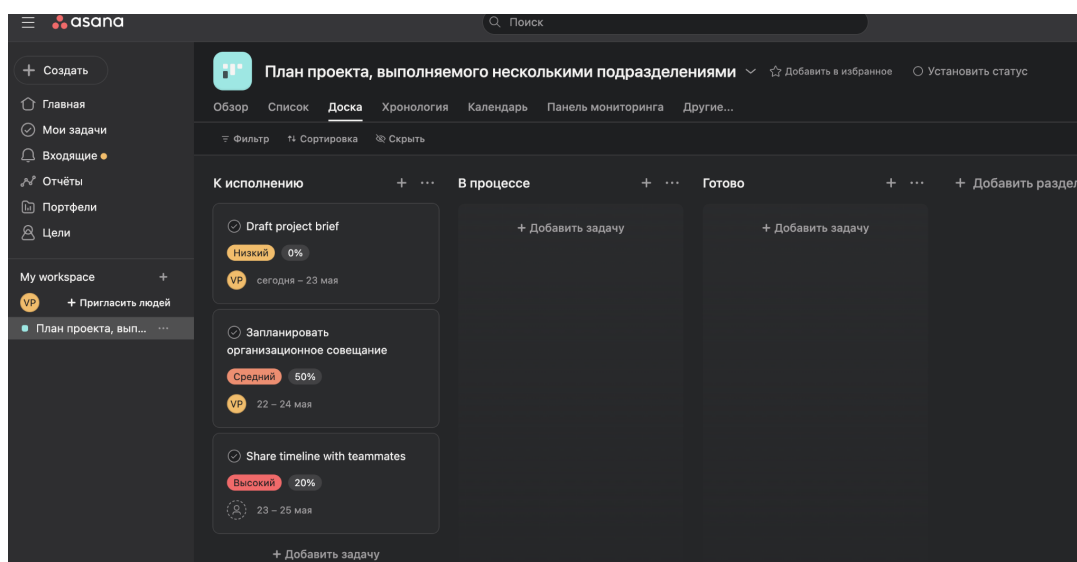


Рис. 1.15 - Сторінка з дошкою Асана

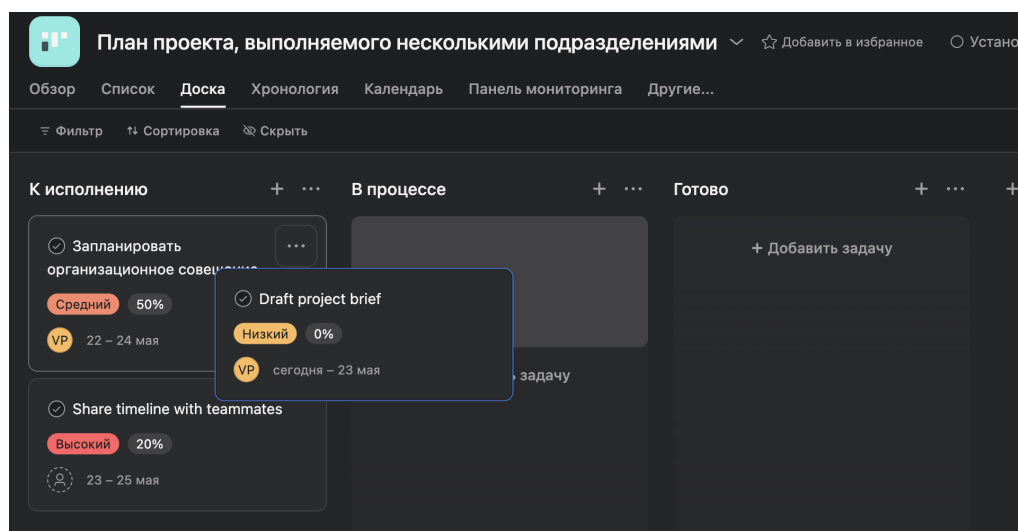


Рис. 1.16 - Перетягивания карточек Асана

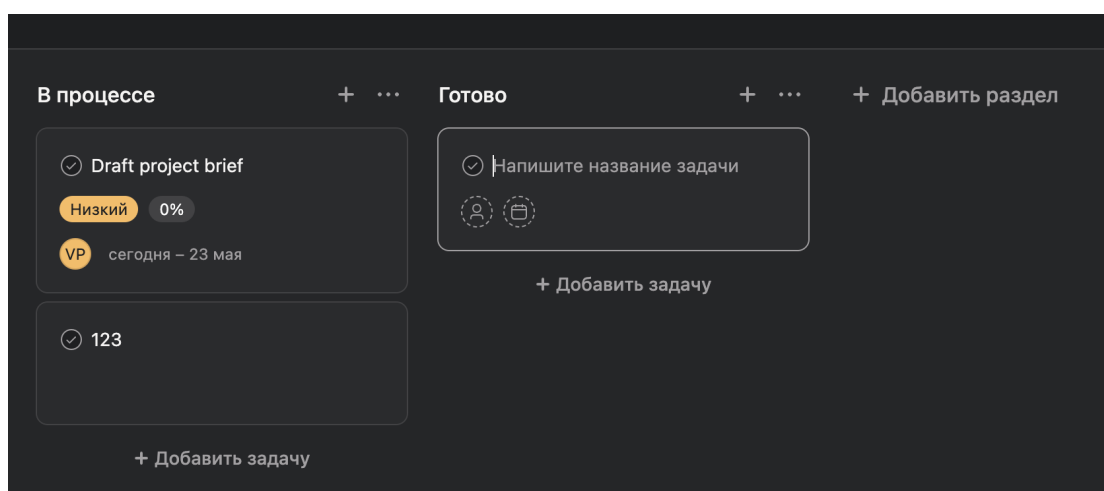


Рис. 1.17 - Створення карточек Асана

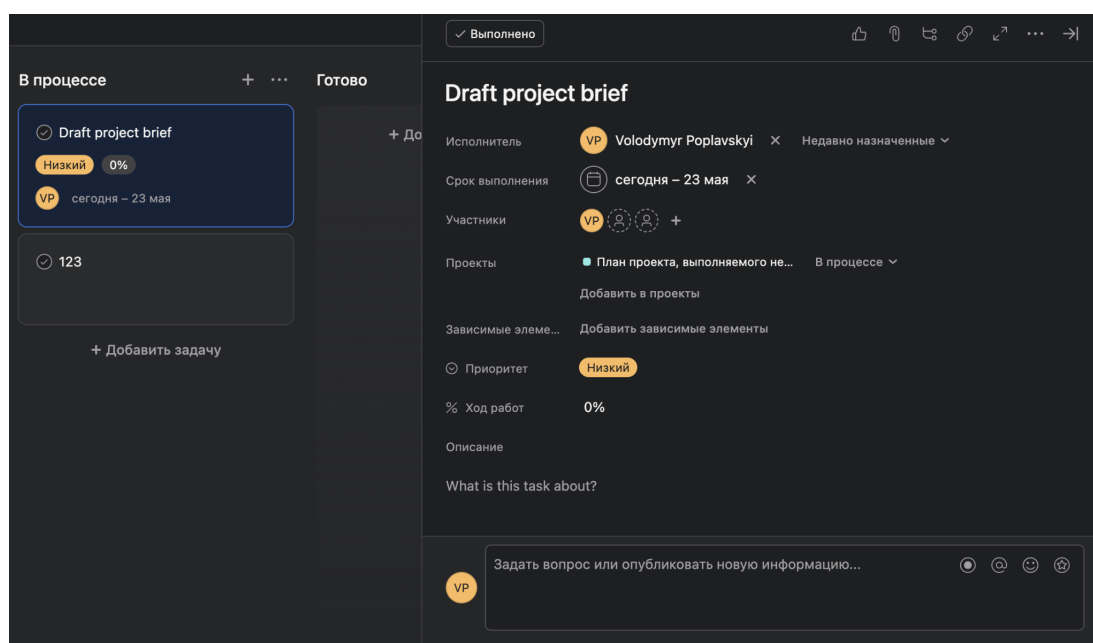


Рис. 1.18 - Попап картки Асана

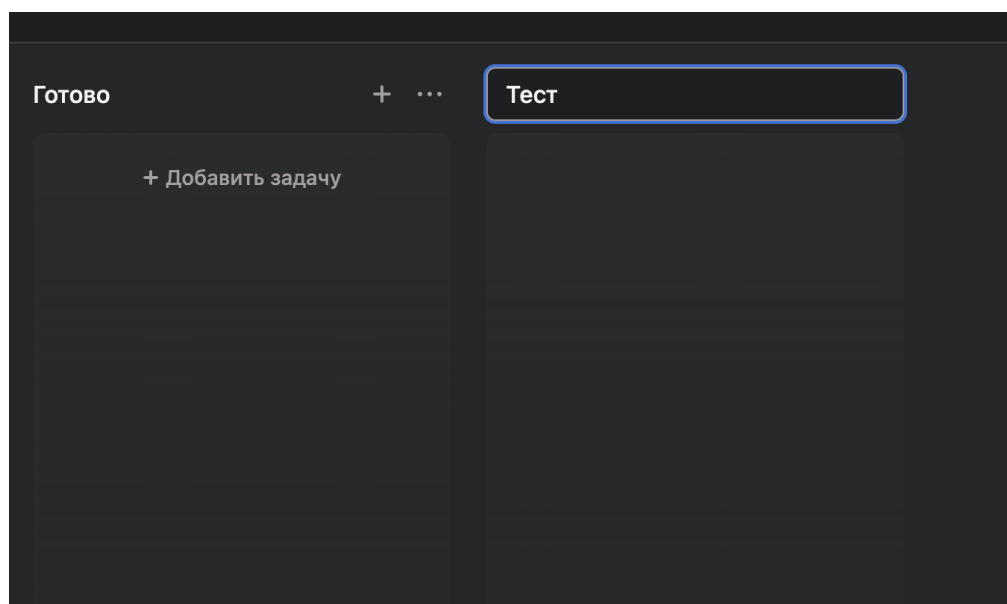


Рис. 1.19 - Створення нових колонок Асана

Додаток Асана, дуже схожий візуально на своїх попередників, проте, як на мене, він зручний не у всіх аспектах, наприклад, тут нема такого зручного переміщення колонок як це було у трело. Але мені дуже подобається як виглядає сторінка з репортами. [8]

І останній із подібних додатків - Wrike.

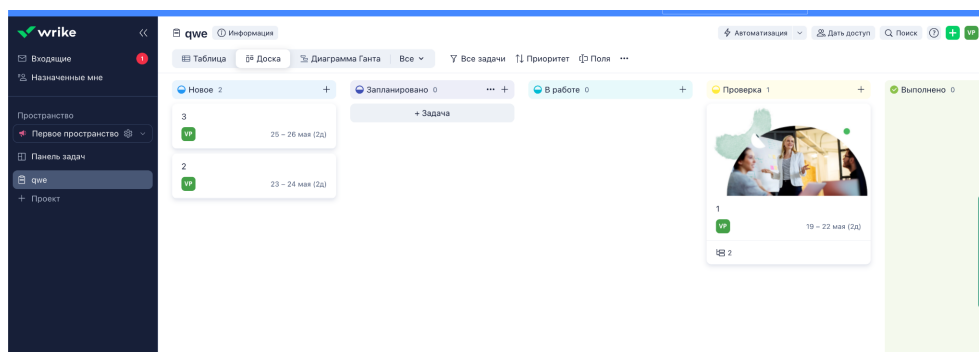


Рис. 1.20 - Сторінка з дошкою Wrike

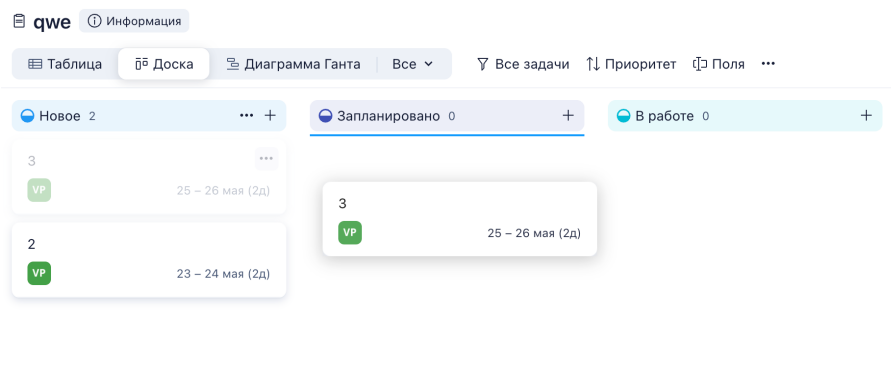


Рис. 1.21 - Перетягування карток Wrike

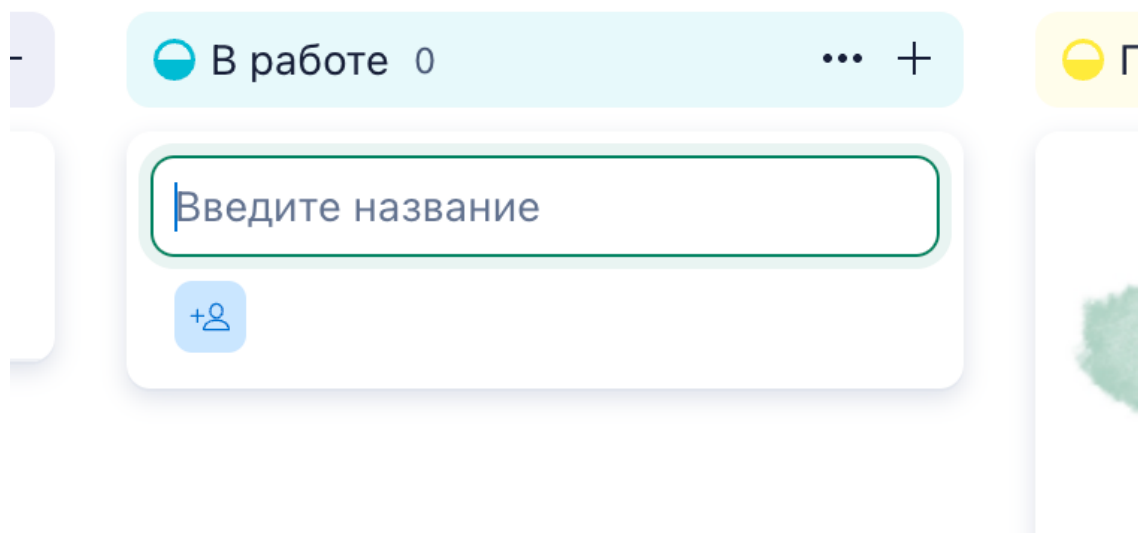


Рис. 1.22 - Створення карток Wrike

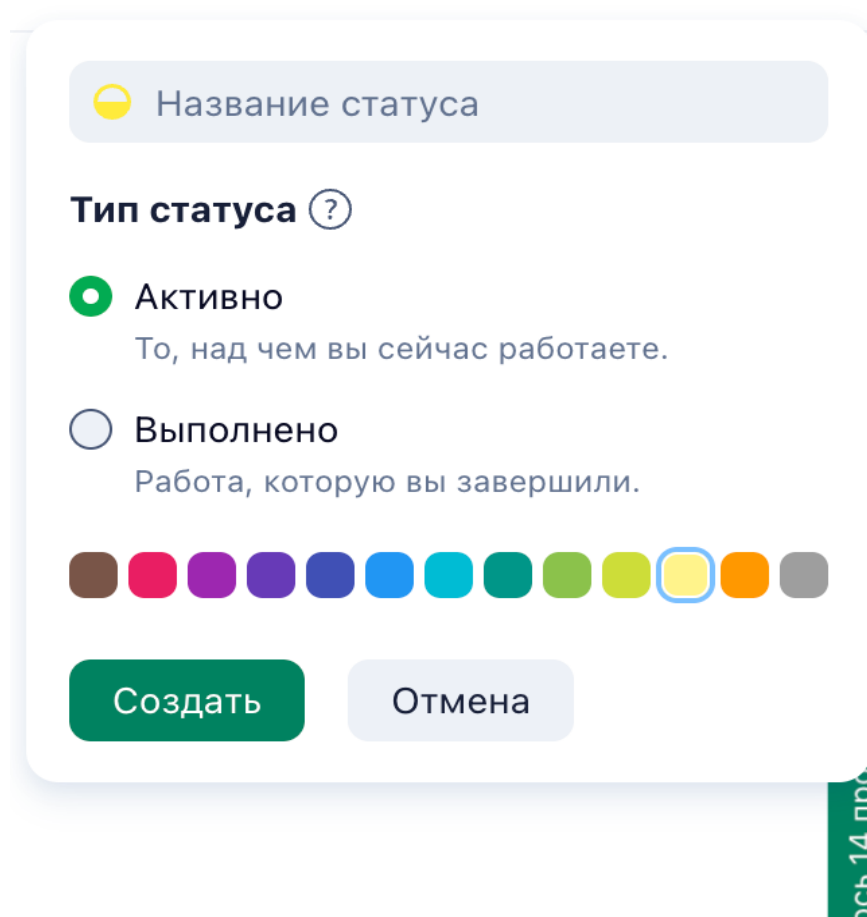


Рис. 1.23 - Створення колонок Wrike

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

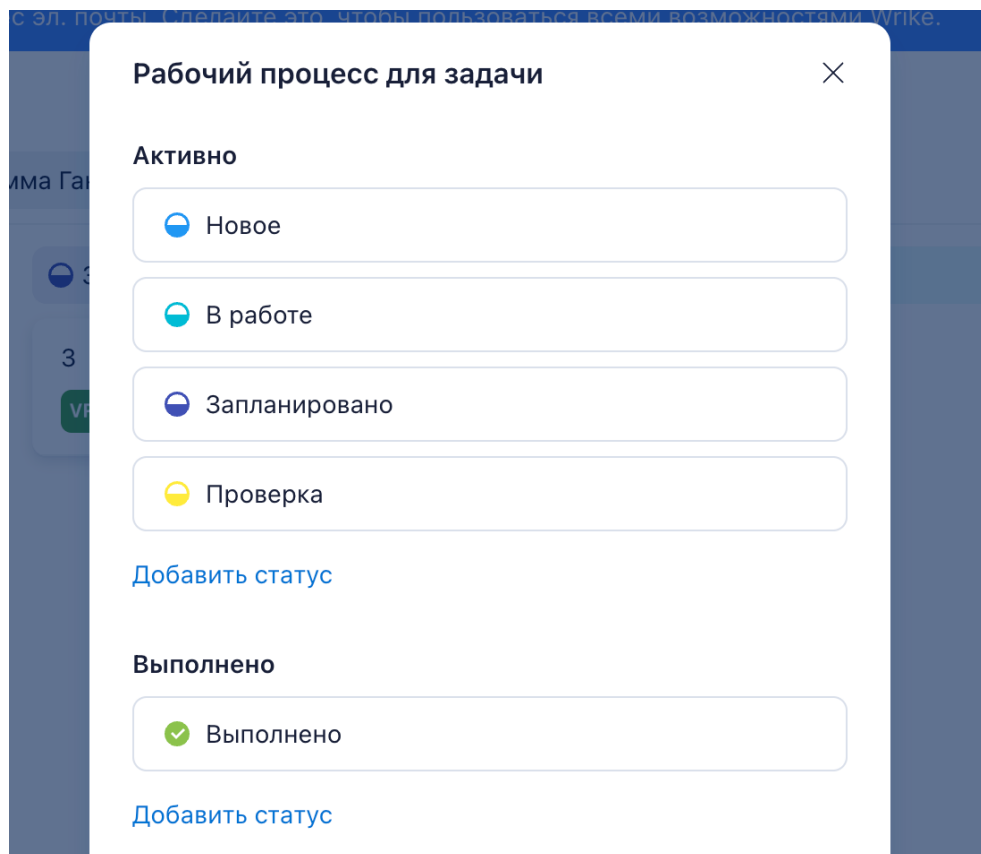


Рис. 1.24 - Налаштування порядку колонок Wrike

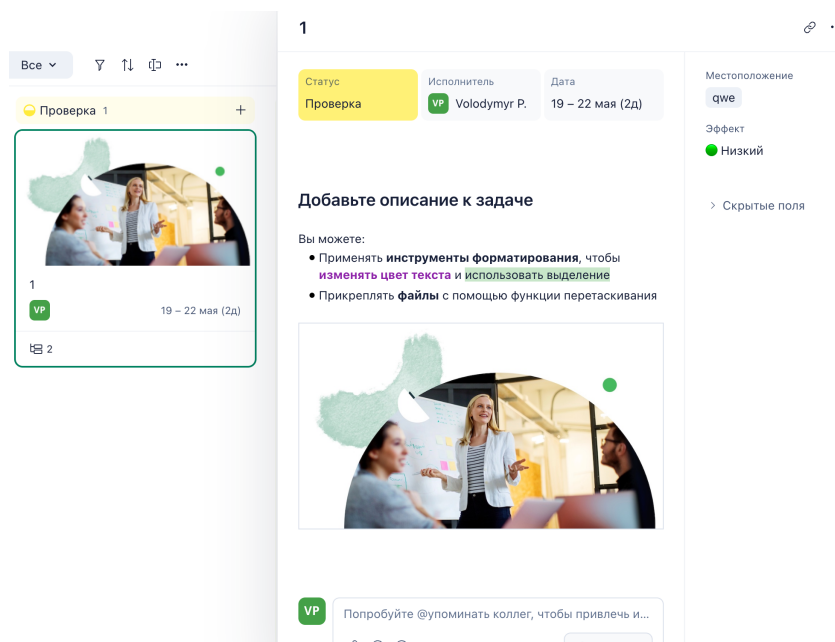


Рис. 1.25 - Оновлення карток Wrike

Отже, переглянувши всі з додатків, можемо зробити порівняльну таблицю, з функціональностями, аби порівняти їх, та провести аналіз, який саме функціонал повинен бути в нашому майбутньому додатку (Таблиця 1.1).

Порівняльна таблиця додатків аналогів

| Порівняльні характеристики                  | Wrike | Jira | Trello | Asana |
|---------------------------------------------|-------|------|--------|-------|
| Можливість створювати нові картки           | +     | +    | +      | +     |
| Зручний інтерфейс                           | +     | +    | +      | ±     |
| Можливість додавання нових колонок          | +     | +    | +      | +     |
| Швидкість обробки створення карток/колонок  | +     | ±    | +      | +     |
| Оновлення карток за допомогою попапу        | +     | +    | +      | -     |
| Наявність сторінки репортів та їх зручність | -     | +    | -      | +     |

З таблиці бачимо наступну картину - майже всі додатки мають однаковий функціонал, схожий дизайн, проте в деяких з них є суб'єктивні мінуси, наприклад менеджмент стовпців в Asana, або репорти у Wrike. Також як вже писалось раніше, Jira має досить великий і різноманітний набір функціоналу, що іноді робить джиру складну для використання, особливо для новачків у цій справі. Тому для свого майбутнього додатку я хотів би виділити наступний функціонал [8]:

1. Простий, зручний, але в той же час гарний дизайн.
2. Можливість входу в систему.
3. Можливість створення та оновлення карток.
4. Можливість створення та менеджмент колонок.

### 1.3. Обґрунтування вибору методології для реалізації в додатку

На разі є досить велика кількість різноманітних методологій в ІТ, та все таки, зважаючи на мій особистий досвід та на відчуття з якими я зіткнувся при роботі з методологіями, я вирішив надати перевагу саме Kanban.

Kanban - це метод управління процесами, що походить з японської системи Toyota Production System. Він був розроблений для оптимізації виробничих процесів, але згодом був успішно застосований в різних галузях, включаючи інформаційні технології (ІТ).

Історія Kanban розпочалася у 1940-х роках в Японії на автомобільному заводі Toyota. Головною ідеєю Kanban була визначення кількості матеріалів, які потрібні на кожному етапі виробництва, та встановлення системи сигналів (карток Kanban), які вказували, коли потрібно поповнити запаси. Це дозволило ефективно управляти запасами та зменшувати затримки виробництва.

У ІТ Kanban широко використовується для управління розробкою програмного забезпечення та інших проектів. Kanban досяг популярності завдяки своїй простоті та гнучкості. Він дозволяє візуалізувати робочий процес на дошці з карточками, які відображають завдання та їхній поточний стан (наприклад, "у процесі", "очікування" або "завершено").

Основні переваги Kanban включають:

- Прозорість: Kanban дає чітке уявлення про стан кожного завдання і загального прогресу проекту.
- Гнучкість: Kanban легко адаптується до змін, оскільки можна додавати, змінювати або видаляти завдання без переривання робочого процесу.
- Контроль витрат: Завдяки обмеженому числу робочих завдань, Kanban допомагає уникнути перевантаження і зосередитися на важливих завданнях.

- Покращення комунікації: Kanban стимулює спілкування та співпрацю між учасниками проекту, оскільки всі бачать поточний стан завдань.
- Керування пріоритетами: Kanban допомагає встановити та пріоритизувати завдання, забезпечуючи більшу ефективність виконання проекту.
- Покращення продуктивності: Kanban зменшує час очікування та затримки, що сприяє підвищенню продуктивності команди.

Загалом, Kanban є потужним інструментом для управління проектами, незалежно від галузі, і особливо корисним для ІТ-проектів, де важливість впорядкування та гнучкості висока. [10,13,14]

#### **1.4. Інструментальні засоби реалізації програмного продукту**

Для розробки додатку, мною було обрано мову програмування Javascript, а саме фреймворк Vue.js. [1,2]

Vue.js — JavaScript-фреймворк, що використовує шаблон MVVM [9] для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних.

Я обрав цей фреймворк, через ряд його переваг [1]:

- висока адаптивність: може бути реалізований швидкий перехід від інших фреймворків до Vue.js;
- масштабування: Vue.js може допомогти в розробці великих реюзабельних компонентів, які можуть бути зроблені майже за той же час, що і більш прості;
- Vue.js має детальну документацію різними мовами, що пришвидшує процес навчання для розробника, який має базові знання HTML [3] та JS.

Вибір бази даних пав на Firebase. Firebase – це платформа для розробки мобільних та веб додатків з величезним функціоналом. Починалася вона як стартап, а сьогодні її використовують при розробці кращих кросс платформних додатків. Головна перевага платформи в тому, що вона дозволяє розробнику не відволікатися на створення backend, тобто

прихованої від користувача програмної частини проекту, наприклад, серверного коду. І це спрощує і прискорює створення додатків, дає можливість повністю зосередитися саме на UX / UI, тобто, на призначеному для користувача інтерфейсі і досвіді. Це і сервер, і база даних, і хостинг, і аутентифікація в одній платформі. Так, Firebase Realtime Database надає розробникам API, який синхронізує дані додатки між клієнтами і зберігає їх в хмарному сховищі.

**Firebase – це одне з BaaS-рішень** (Backend as a Service), яке дає розробнику масу можливостей. Це і сервер, і база даних, і хостинг, і аутентифікація в одній платформі. Так, Firebase Realtime Database надає розробникам API, який синхронізує дані додатки між клієнтами і зберігає їх в хмарному сховищі. Додаток підключається до бази даних через WebSocket, який відповідає за синхронізацію даних протягом усього сеансу.

Проте в своєму проекті я використав не Realtime Database, а саме Cloud Firestore.

Cloud Firestore — це хмарна база даних NoSQL, до якої ваші програми Apple, Android і веб-додатки можуть отримати прямий доступ через власні SDK. Cloud Firestore також доступний у нативних пакетах SDK Node.js, Java, Python, Unity, C++ і Go, на додаток до REST і RPC API. Дотримуючись моделі даних NoSQL Cloud Firestore, ви зберігаєте дані в документах, які містять поля, зіставлені зі значеннями. Ці документи зберігаються в колекціях, які є контейнерами для ваших документів, які можна використовувати для впорядкування даних і створення запитів. Документи підтримують багато різних типів даних, від простих рядків і чисел до складних вкладених об'єктів. Ви також можете створювати підколекції в документах і створювати ієрархічні структури даних, які масштабуються в міру зростання бази даних. Модель даних Cloud Firestore підтримує будь-яку структуру даних, яка найкраще підходить для вашої програми. [4,5]

## Висновки до першого розділу

В першому розділі було поставлено технічне завдання, проаналізовані існуючі аналоги, такі як: Jira, Trello, Asana, Wrike. На основі аналізу було зроблено висновки, щодо функціоналу веб-додатку.

Для реалізації було вирішено використати такі інструментарні засоби: VueJS, базу даних Firebase, мову стилів CSS, наведено їх короткий опис.

Розглянуто методологію Kanban з якою можна буде працювати в рамках розробленого додатку.

## РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ДЛЯ УПРАВЛІННЯ ІТ ПРОЕКТАМИ

### 2.1. Визначення варіантів використання системи

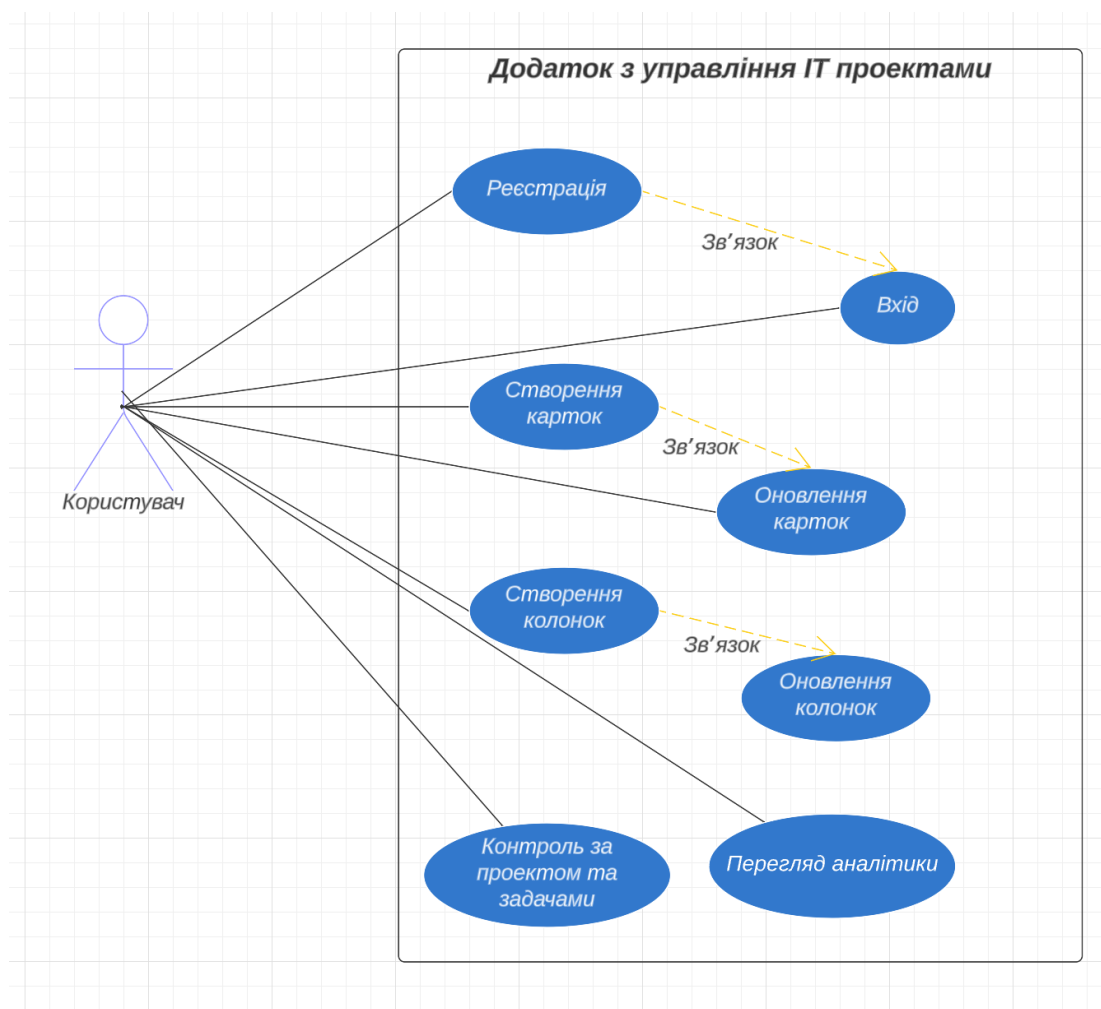


Рис. 2.1 - Діаграма варіантів використання

Діаграма варіантів використання це графічне зображення, що використовується для моделювання функціональних вимог до системи з точки зору її користувачів. Вона відображає взаємодію між акторами (користувачами системи або зовнішніми сутностями) та системою через варіанти використання (Use Case).

Діаграма варіантів використання складається з акторів, варіантів використання та зв'язків між ними. Актори - це ролі або сутності, які взаємодіють з системою. Варіанти використання - це сценарії або функціональні можливості, які система надає своїм акторам. Зв'язки між

акторами та варіантами використання вказують на те, як актори взаємодіють з системою через варіанти використання.

Діаграма варіантів використання допомагає визначити функціональні вимоги до системи, ідентифікувати ролі акторів, розуміти взаємодію між користувачами та системою, а також слугує основою для подальшого аналізу та проектування системи. [12]

## **2.2. Алгоритми роботи системи**

Даний підрозділ покаже вам яким чином взаємодія система між собою, за допомогою діаграм активності та послідовності.

Діаграма активності - це графічний інструмент моделювання, який використовується для візуалізації послідовності дій та поведінки системи, процесу або функції. Вона дозволяє представити послідовність активностей, рішень та взаємодії між ними.

Діаграми активності використовуються для аналізу та проектування бізнес-процесів, алгоритмів, виконавчих процедур та інших послідовних дій. Вони допомагають уявити хід виконання дій, виявити можливі переходи та умови, оцінити часові затрати та ідентифікувати можливі проблеми або неоднозначності. На діаграмі активності активності представляються у вигляді прямокутників з заокругленими кутиками, а переходи між активностями - у вигляді стрілок.

Діаграми активності дозволяють візуалізувати та легше зрозуміти логіку та потік виконання процесу чи функції, сприяють виявленню можливих вдосконалень та оптимізації діяльності. Вони широко використовуються в аналізі та проектуванні програмного забезпечення, бізнес-процесів, систем управління та інших областях.

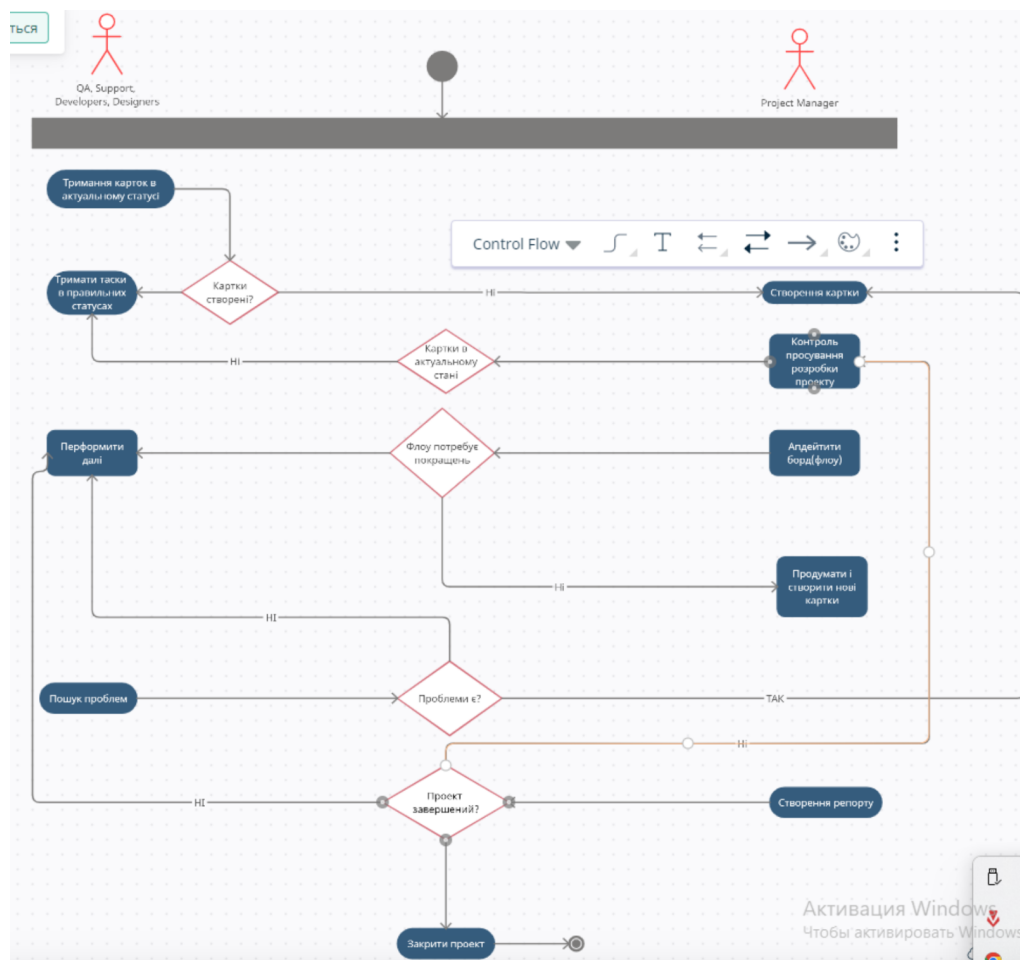


Рис. 2.2 Діаграма активності додатку

Далі розглянемо діаграму послідовності.

Діаграма послідовності - це графічний інструмент моделювання, який використовується для візуалізації послідовності взаємодії об'єктів у системі або між системою та її акторами протягом певного часу.

Діаграми послідовності використовуються для аналізу, проектування та відлагодження системи, де важливо відображати послідовність повідомлень, викликів методів та відповідей між об'єктами чи компонентами системи. Вони дозволяють орієнтуватися в порядку виконання дій, виявити можливі проблеми та зрозуміти взаємодію між об'єктами. На діаграмі послідовності об'єкти представляються у вигляді вертикальних ліній, а повідомлення між ними - у вигляді горизонтальних стрілок з позначенням назви повідомлення та параметрів. Додатково можуть бути використані символи для позначення умов, ітерацій, паралельних викликів та інших сценаріїв.

Діаграми послідовності допомагають зрозуміти, як об'єкти взаємодіють між собою, які дії виконуються в певному порядку та які дані передаються між ними.

Вони широко використовуються в аналізі та проектуванні програмного забезпечення, спрощують спілкування між розробниками та зацікавленими сторонами і допомагають виявляти потенційні проблеми або неоднозначності у взаємодії системи.

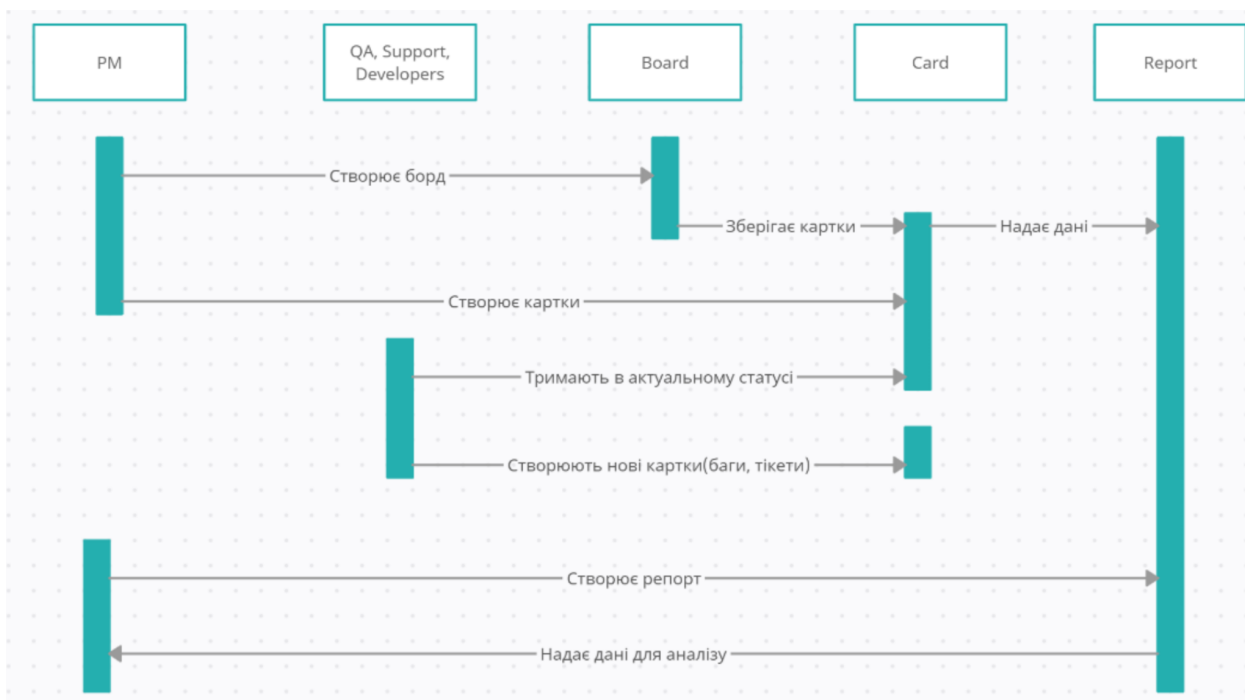


Рис. 2.3 - Діаграма послідовності додатку

### 2.3. Структура компонентів системи

Компоненти Vue - це незалежні та перевикористовувані блоки коду, які включають в себе HTML, CSS [3] і JavaScript для створення роздільних частин веб-інтерфейсу.

Так як Vue.js працює з компонентами, наш додаток містить в собі досить велику їх кількість. Тому вважаю варто показати їх та описати за що кожен з них відповідає.

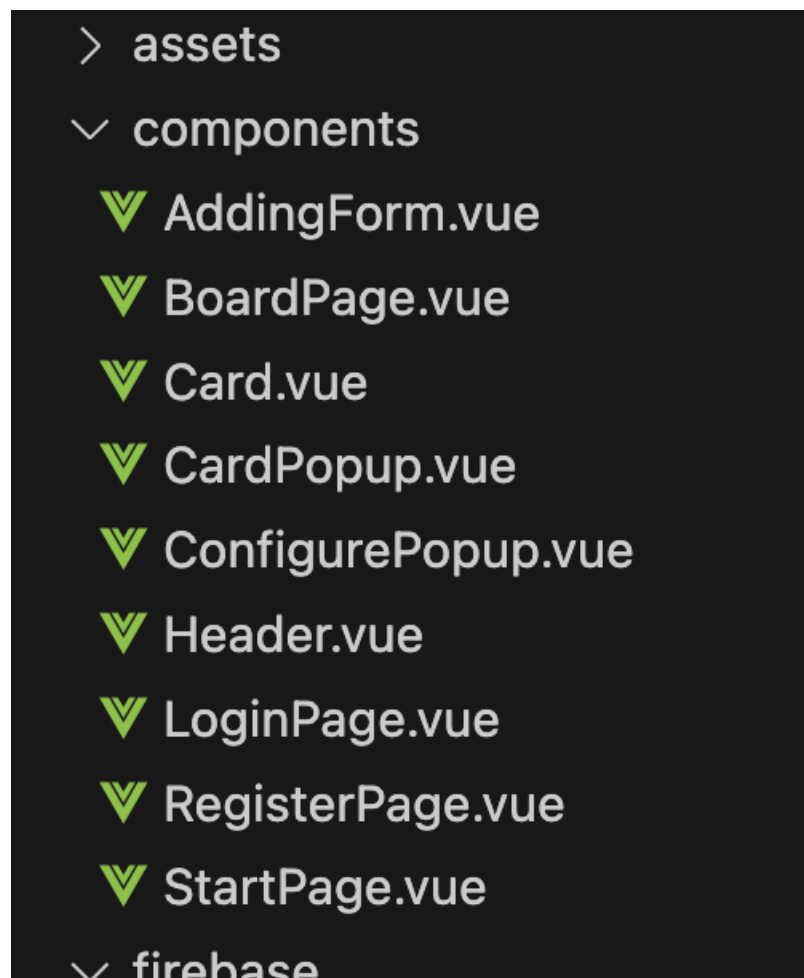


Рис. 2.4 Набір компонентів додатку

AddingForm - являє собою попап, за допомогою якого можна додавати нові картки.

BoardPage - фактично основний компонент додатку, який в собі містить основні функції, та на ньому користувач буде проводити найбільше часу.

Card - компонент, який являє собою карточку, яку можна тягати між колонками.

CardPopup - компонент, який напряду зв'язаний із попереднім компонентом. Являє собою попап в якому міститься вся інформація картки, також за допомогою цього компоненту, можна оновлювати дані в картці.

ConfigurePopup - такий самий попап по вигляду як попередній, проте містить в собі інший функціонал, такий як - додавання видалення та редагування колонок.

Header - компонент, який не несе в собі функціоналу, проте він містить в собі логотип додатку, видний на кількох сторінках.

LoginPage - компонент, що відповідає за вхід в систему.

RegisterPage - майже однаковий з попереднім, проте відповідає за реєстрацію, також містять різні перевірки даних.

StartPage - це компонент, який являє собою стартову сторінку додатку, виглядає як звичайний лендінг, має кнопки які ведуть нас далі на компоненти реєстрації або входу.

## 2.4. Реалізація веб-додатку

Отже, як зазначалось в попередньому розділі, додаток побудований на Vue.js та Firebase. Таким чином роль всього бекенду в нас виконує саме Firebase. Тому мною було створено проект в Firebase. Firebase надає великий спектр різних функцій від бази даних до хостингу.

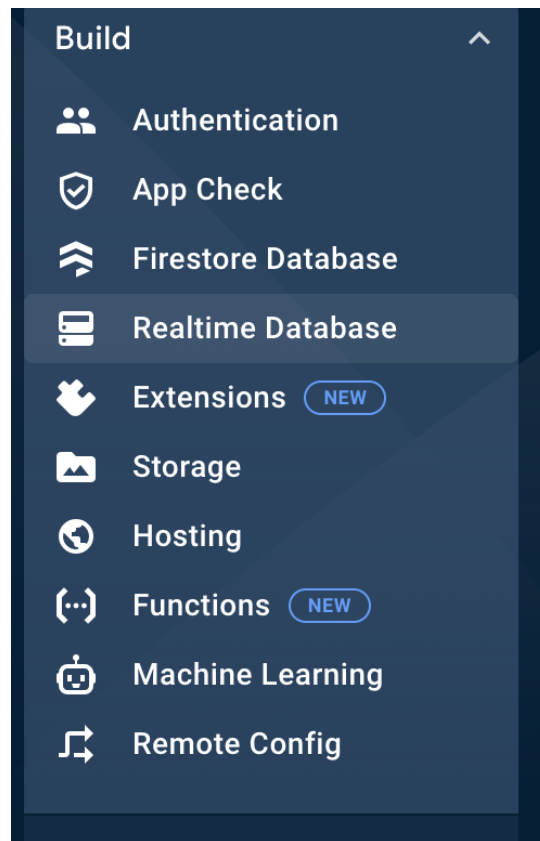


Рис. 2.5 - Сервіси які присутні у Firebase

Проте в нашому додатку для реалізації мінімально необхідного функціоналу для нашого додатку потрібні лише - Firestore Database та Authentication.

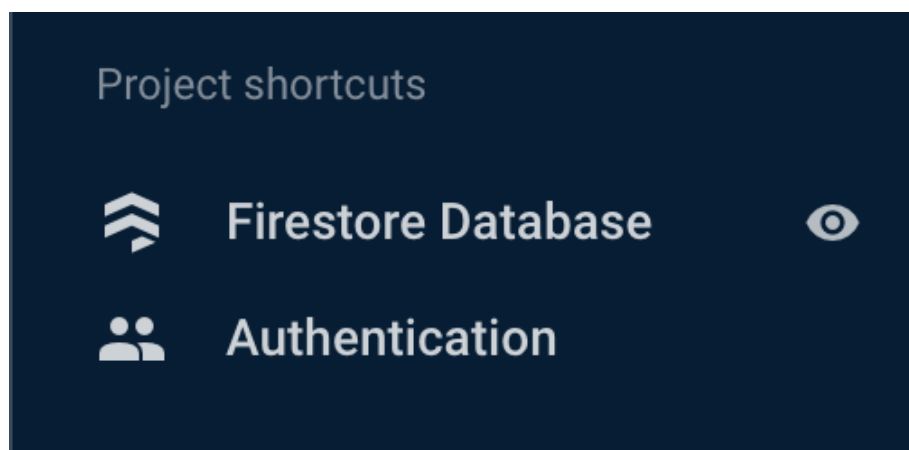


Рис. 2.6 - Сервіси які використовуються у нашому додатку

Firestore Database - це хмарна база даних, розроблена і управляється компанією Google. Вона є частиною сервісу Google Cloud Platform і надає можливість зберігати, синхронізувати та запитувати дані в реальному часі.

Firestore Database забезпечує гнучкість та масштабованість, що робить її популярним вибором для розробників, які будують веб- та мобільні додатки. Вона використовує модель документа, де дані зберігаються у вигляді колекцій, які містять документи з полями та значеннями.

Firestore Database підтримує режим офлайн доступу, що дозволяє працювати з даними навіть при відсутності з'єднання з Інтернетом. Вона також надає можливості автоматичної синхронізації даних між різними пристроями та підтримку розширення функціональності за допомогою серверних функцій.

Firestore Database використовується для розробки різноманітних додатків, включаючи соціальні мережі, електронну комерцію, системи управління вмістом та багато іншого.

Firebase Authentication - це сервіс автентифікації та управління користувачами, який надається Google Firebase. Він дозволяє розробникам швидко та зручно впроваджувати систему аутентифікації в їх додатках.

Firebase Authentication забезпечує різні методи автентифікації, включаючи електронну пошту та пароль, вхід через облікові записи Google, Facebook, Twitter, GitHub та інші провайдери ідентифікації. Він також

підтримує механізми анонімної автентифікації, дозволяючи користувачам використовувати додаток без обов'язкової реєстрації.

Firebase Authentication забезпечує безпеку автентифікації, включаючи захист від злову паролів, багатофакторну автентифікацію і інші функції безпеки. Він також надає можливість керувати користувачами, встановлювати ролі та права доступу, виконувати аутентифікацію на серверному боці та інтегруватися з іншими сервісами Firebase.

Firebase Authentication інтегрований з іншими сервісами Firebase, такими як Firebase Realtime Database, Cloud Firestore, Cloud Functions та інші, що дозволяє розробникам легко керувати доступом до даних та реалізовувати персоналізовані функції для користувачів їх додатків. [4,5]

Так як ми будемо працювати з API нам потрібно завести файл з налаштуваннями аби пов'язати проекти.

```
src > firebase > JS index.js > [e] firebaseConfig
1  import { initializeApp } from "firebase/app";
2  import { getFirestore } from "firebase/firestore";
3  import { getAuth } from "firebase/auth";
4
5  const firebaseConfig = {
6    apiKey: "AIzaSyCGhfxPzunnyB-dFyQSKmUtyXLN0P04IKpc",
7    authDomain: "pmtool-a407b.firebaseio.com",
8    projectId: "pmtool-a407b",
9    storageBucket: "pmtool-a407b.appspot.com",
10   messagingSenderId: "155369036334",
11   appId: "1:155369036334:web:94ac015fc5e52cd9d5ddbfb",
12   measurementId: "G-65QS8V6E13"
13 };
14
15 const app = initializeApp(firebaseConfig);
16 const db = getFirestore(app);
17 const auth = getAuth(app);
18
19 export {
20   db,
21   auth,
22 }
```

Рис. 2.7 - Налаштування FB в проєкті

Наразі, база даних немає великої кількості таблиць(документів) оскільки того не вимагає додаток на цьому етапі. БД складається з 3 колекцій - cards, priorities, statuses.

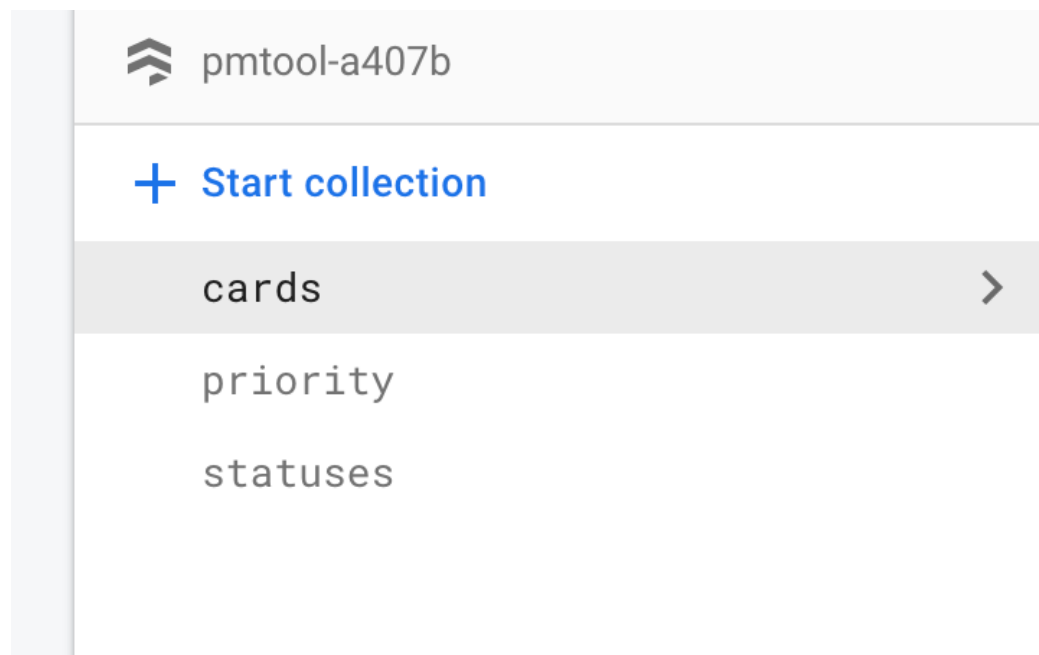


Рис. 2.8 - Колекції в БД

Cards - зберігають всі картки, які були створені.

Priority - зберігають пріоритети карток, які є у випадяючому списку при створенні або оновленні карток.

Statuses - зберігають колонки.

### **Висновки до другого розділу**

В цьому розділі ми розглянули діаграму варіантів використання, яка показала більш детально функціональність, яка повинна бути присутня в нашому додатку. Також подивились на діаграми активності та послідовності для більшого розуміння як компоненти будуть взаємодіяти всередині додатку.

Розглянули компоненти які використовуються в додатку.

Проглянули колекції, які наразі є в базі, та які сервіси Firebase ми використовуємо в нашій системі, та як виглядають налаштування для зв'язування бд з додатком.

## РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ

### 3.1. Архітектура додатку та його розгортання

Наший додаток буде використовувати архітектурний шаблон - MVVM.

MVVM (Model-View-ViewModel) - це архітектурний шаблон, який використовується для розробки програмного забезпечення з графічним інтерфейсом користувача.

Він дозволяє розділити логіку додатку, представлення даних та взаємодію з користувачем. [9]

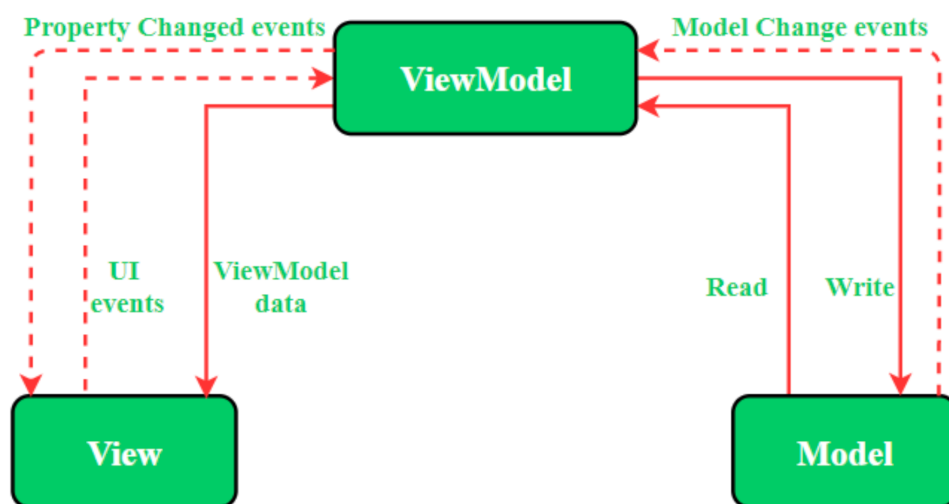


Рис. 3.1 - MVVM архітектура

Основні компоненти MVVM архітектури:

1. **Модель (Model):** Модель представляє дані та бізнес-логіку додатку. Вона може включати структури даних, класи, сервіси для взаємодії з базою даних або зовнішніми джерелами. Модель не залежить від інших компонентів.
2. **Представлення (View):** Представлення відповідає за візуальне відображення даних та інтерфейс користувача. Воно взаємодіє з користувачем та відображає дані з ViewModel. Представлення може бути реалізоване як HTML-шаблони, розмітка або компоненти.
3. **ViewModel:** ViewModel є посередником між Моделлю та Представленням. Він містить логіку, яка взаємодіє з Моделлю та

підготовлює дані для відображення в Представленні. ViewModel також обробляє дії користувача та відповідає за оновлення Моделі.

Даний архітектурний шаблон був використаний через те, що це один із найпопулярніших шаблонів, а саме для додатків із графічним інтерфейсом написаним на vue.js або Microsoft WPF.

Що стосується розгортання, то з цим великих складнощів не виникає, коли ми згадаємо, що ми використовуємо Firebase. Так як Firebase є baas платформою, саме вона надає не тільки можливість працювати з базою даних та аутентифікацією, а ще й надає можливість розгортання веб додатку.

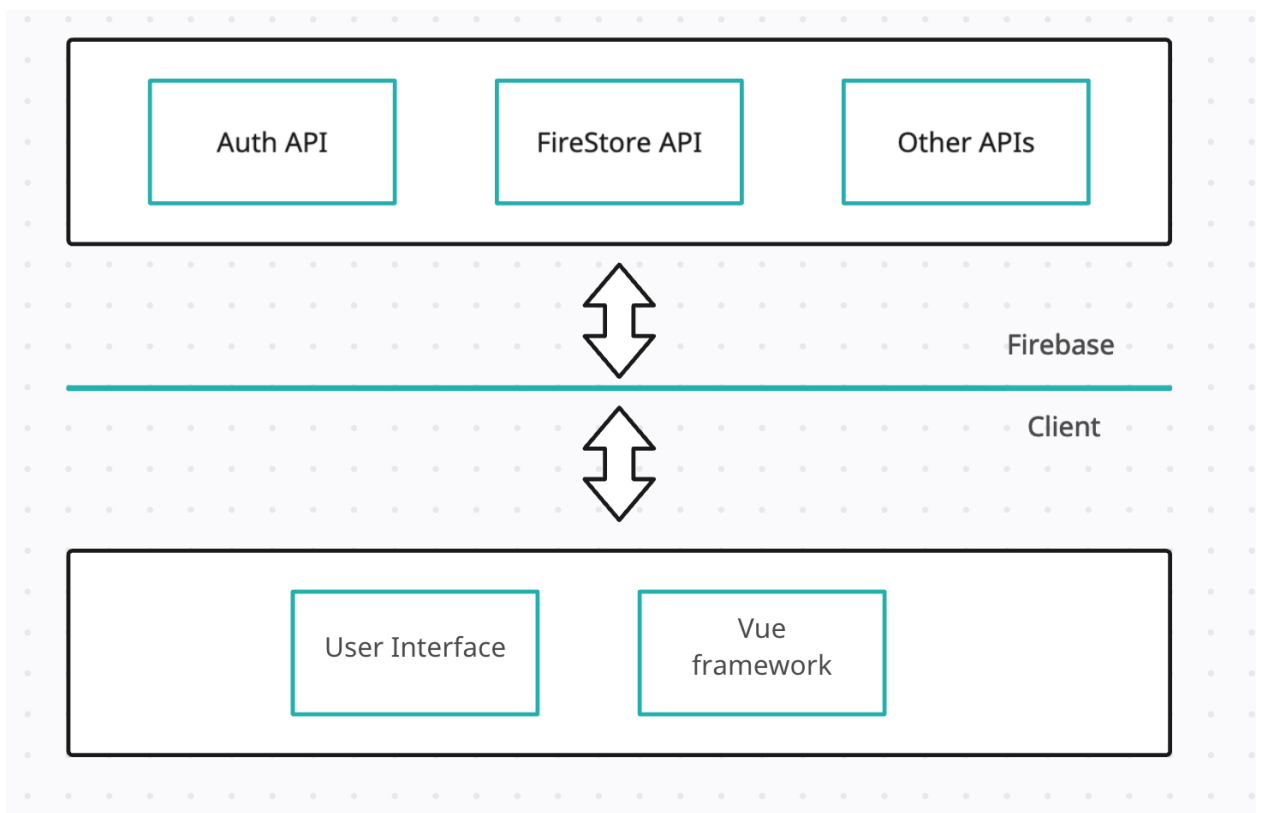


Рис. 3.2 - Схема розгортання [11]

Отже, можна зробити висновок, використовуючи Firebase можна не перейматись створенням бекенд частини та бути спокійним за розгортання. Система надає великий набір сервісів для розробки найрізноманітніших додатків для будь-якої платформи. Також, дуже важливо відмітити той момент, що Firebase дає змогу максимально приділити увагу фронтенду. Для надання максимального зручного та легкого користування додатком.

### 3.2. Інтерфейс та порядок роботи з системою

Перейдімо до інтерфейсу. Як вже зазначалось в попередньому розділі, в нас є кілька окремих попапів та сторінок. Почнемо з того, що нас зустрічає.

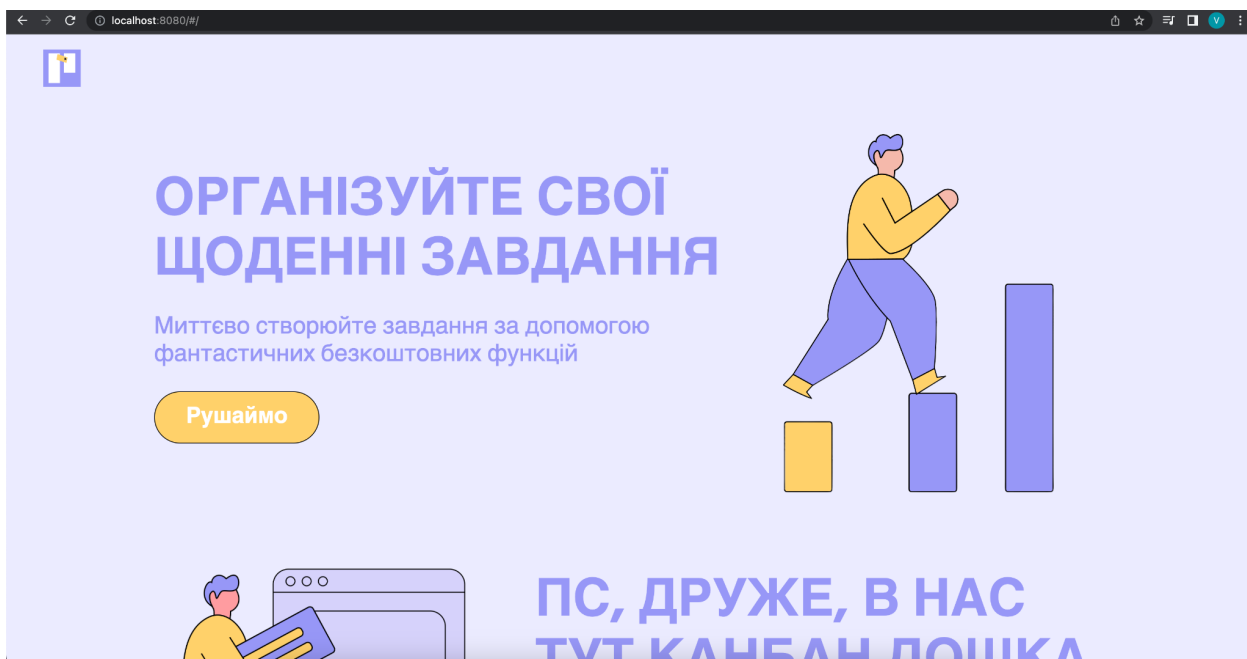


Рис. 3.3 - Головна сторінка

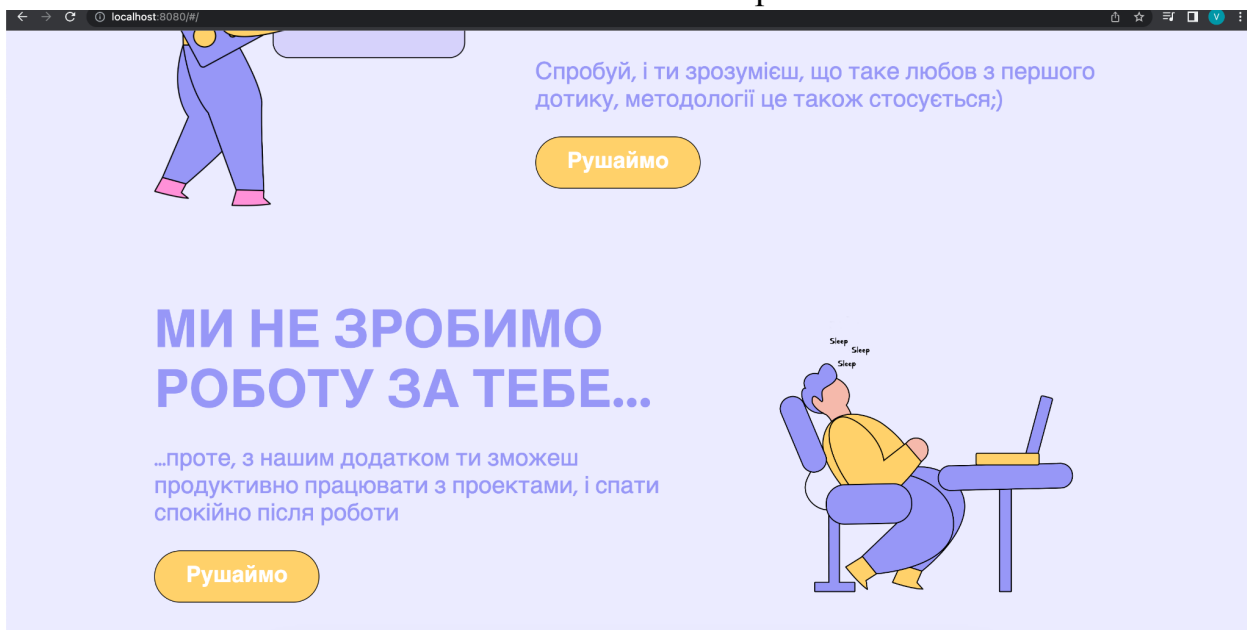


Рис. 3.4 - Головна сторінка

На цій сторінці ми бачимо звичайну інформацію і картинки як на будь-якому сайті. Також при натисненні на кнопку “Рушаймо” ми перейдемо на сторінку реєстрації.

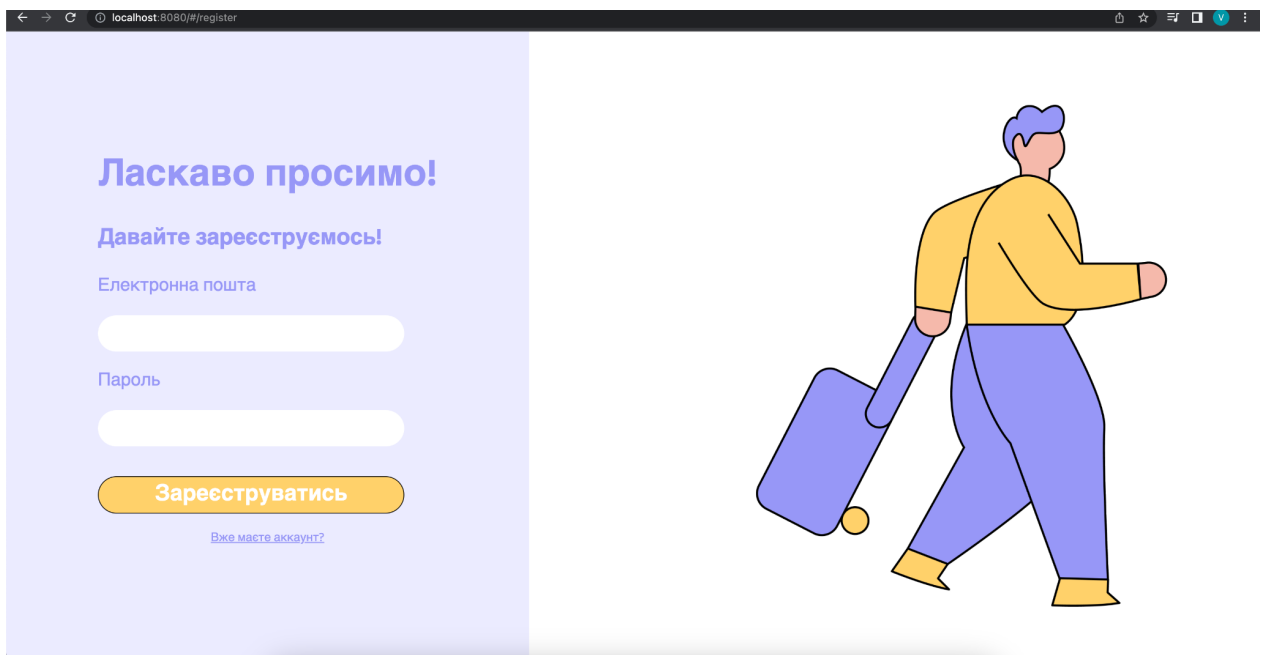


Рис. 3.5 - Сторінка реєстрації

На цій сторінці ми бачимо, поля для електронної пошти та пароль, аби зареєструвати аккаунт. Проте, якщо в нас вже є зареєстрований аккаунт ми можемо натиснути на кнопку, яка знаходиться під жовтою і перейти на наступну сторінку для входу. Вона фактично нічим не відрізняється від цієї, лише кілька невеликих відмінностей у дизайні та інші перевірки введених даних.

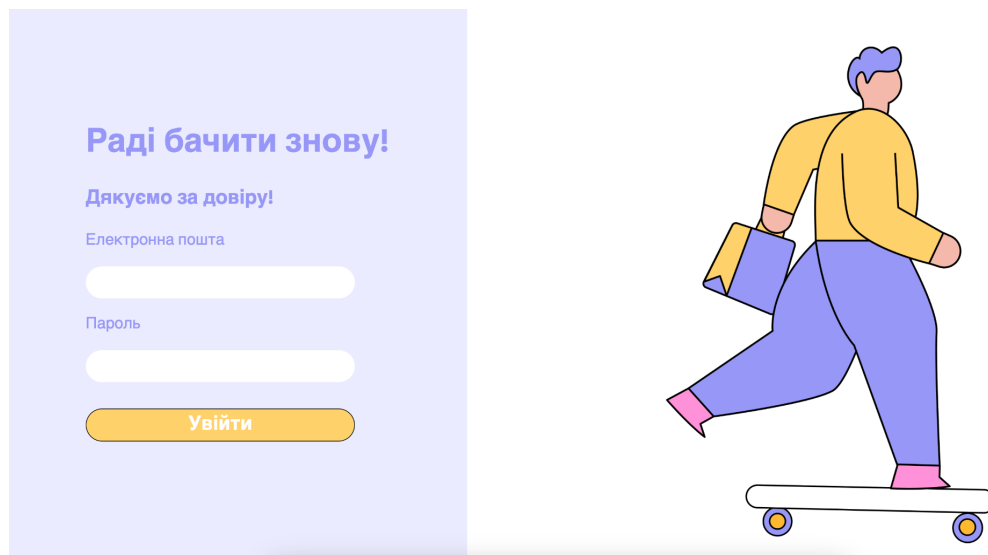


Рис. 3.6 - Сторінка входу

Після введення даних ми потрапляємо на сторінку з нашою Kanban дошкою.

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |

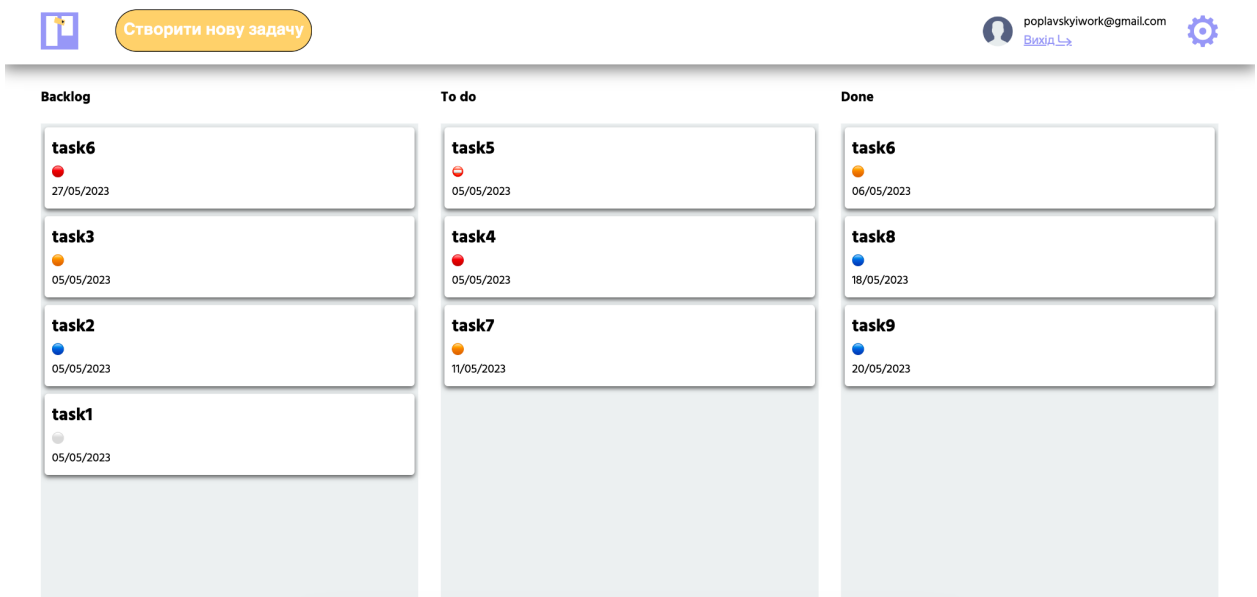


Рис. 3.7 - Основна сторінка з дошкою

Фактично, це була остання сторінка, проте є ще кілька попапів і зараз ми їх розглянемо.

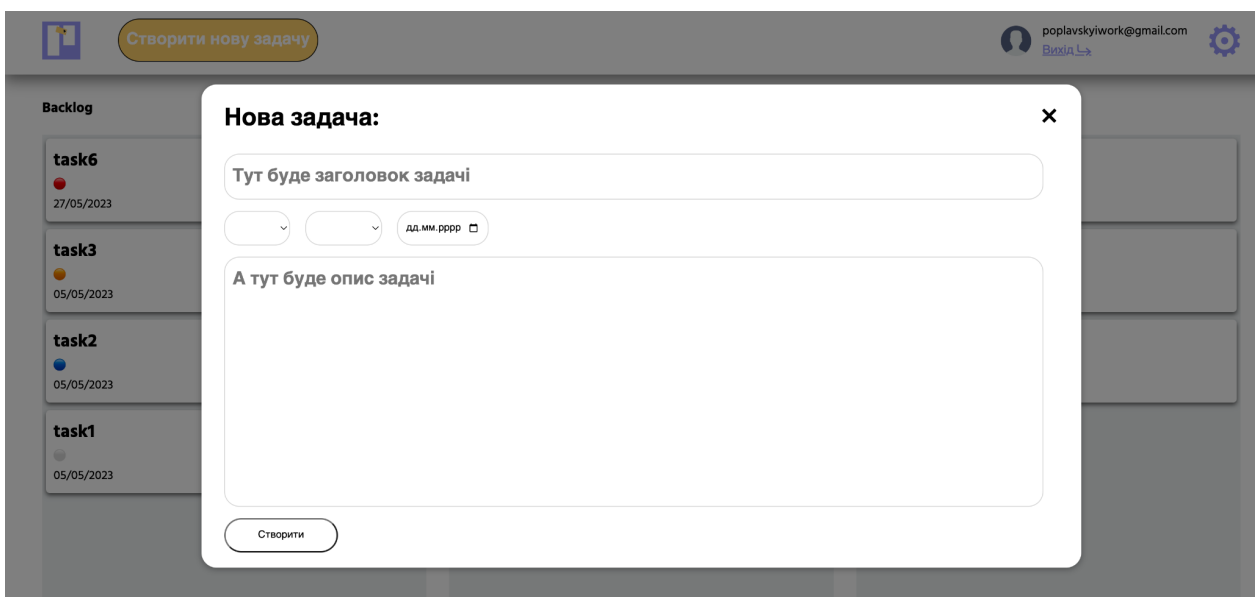


Рис. 3.8 - Попап створення нової картки

На цьому попапі є основні поля які потрібно заповнити для створення нової картки. Такі як заголовок задачі, опис, поле для введення дедлайну і кілька випадаючих списків, які дають можливість обрати потрібну колонку для картки, та її пріоритет.

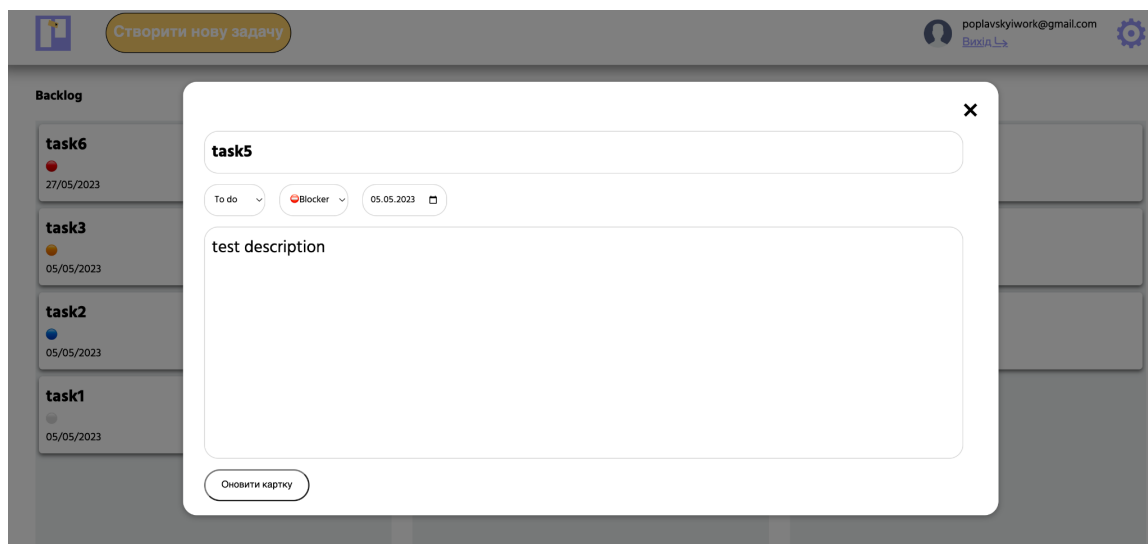


Рис. 3.9 - Попап редагування картки

Фактично він є копією попереднього, проте тут всі дані вже заповнені і їх можна змінювати.

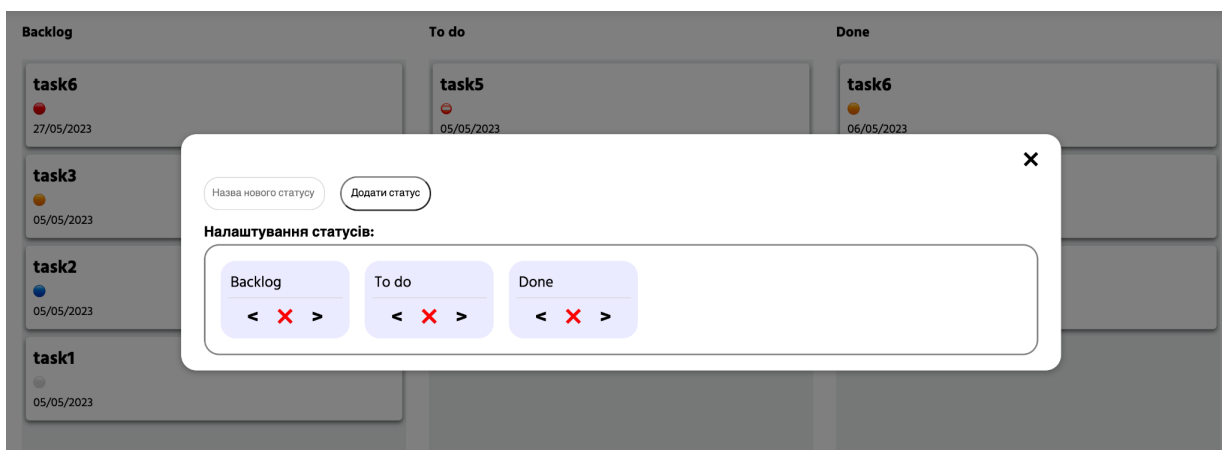


Рис. 3.10 - Попап створення та менеджменту колонок

Останній попап дає змогу створити, видалити та редагувати колонки.

### 3.3. Тестування роботи веб-додатку

На стартовій сторінці нашого додатку при натисненні на кнопку “Рушаймо” маємо перейти на сторінку з реєстрацією.

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |

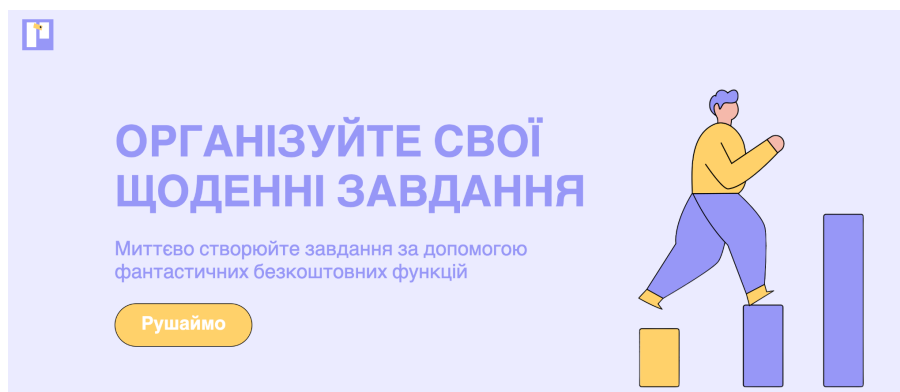


Рис. 3.11 - Стартова сторінка і кнопка “Рушаймо”

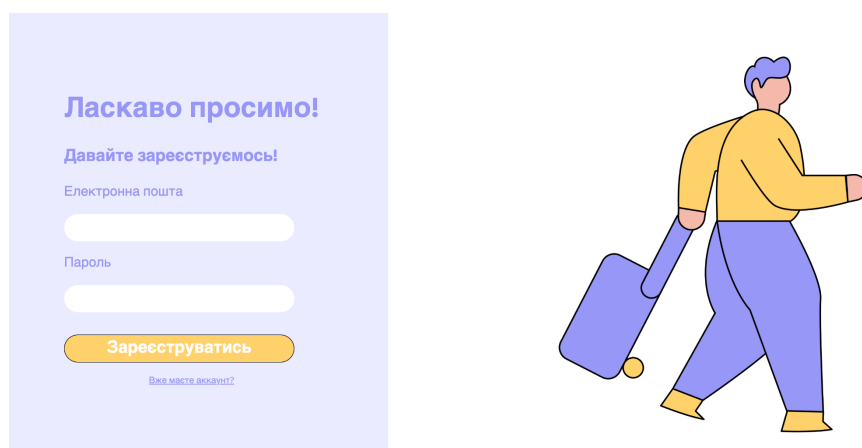


Рис. 3.12 - Сторінка реєстрації після натиснення на кнопку

На цій сторінці зупинимось і перевіримо на введення поля на введення, та спробуємо створити аккаунт за допомогою пошти на яку вже зареєстрований аккаунт.

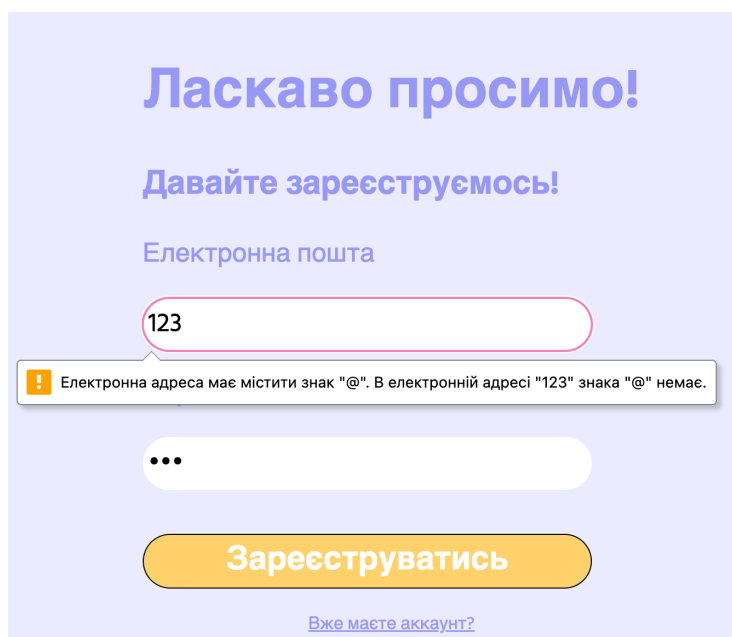


Рис. 3.13 - Введення в поле для пошти некоректних даних

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |

**Давайте зареєструємось!**

Електронна пошта

poplavskyiwork@gmail.com

Пароль

.....

**Пароль повинен мати мінімум 6 символів!**

**Зареєструватись**

Рис. 3.14 - Введення в поле для паролю недостатньої кількості символів

**Давайте зареєструємось!**

Електронна пошта

poplavskyiwork@gmail.com

Пароль

.....

**Користувач з такою поштою вже зареєстрований!**

**Зареєструватись**

[Вже маєте аккаунт?](#)

Рис. 3.15 - Введення пошти на яку вже зареєстрований аккаунт

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |

Рис. 3.16 - Аккаунт який зареєстрований раніше

**Давайте зареєструємось!**

Електронна пошта

test@gmail.com

Пароль

.....

Рис. 3.17 - Створення нового аккаунту з коректними даними

| Search by email address, phone number, or user UID |           |              |              |                              | Add user          |            |     |
|----------------------------------------------------|-----------|--------------|--------------|------------------------------|-------------------|------------|-----|
| Identifier                                         | Providers | Created ↓    | Signed In    | User UID                     |                   |            |     |
| test@gmail.com                                     |           | May 24, 2023 | May 24, 2023 | hn1itvADCBMwSa14fGUadkhjGwJ2 |                   |            |     |
| valdpoplavskiy@gmail.com                           |           | Apr 15, 2023 | May 24, 2023 | FRMaNnTjyTY2QBKTf3iyp4OKjH73 |                   |            |     |
| poplavskiywork@gmail.com                           |           | Apr 15, 2023 | May 24, 2023 | d33Hv2JJuGa8sMoCmVph5gBqm... |                   |            |     |
|                                                    |           |              |              |                              | Rows per page: 50 | 1 – 3 of 3 | < > |

Рис. 3.18 - Створений аккаунт з новими даними

Тепер спробуємо перейти до сторінки з входом і спробувати зайти з неправильним паролем та поштою. Та подивимось як наш додаток обробляє ці дані. А також спробуємо увійти з коректними даними, ми повинні перейти на сторінку з дошкою та побачити там відображення пошти за допомогою якої ми зайшли.

**Раді бачити знову!**

**Дякуємо за довіру!**

Електронна пошта

poplavskyiwork@gmail.com

Пароль

.....

**Некоректний пароль!**

**Увійти**

Рис. 3.19 - Спроба увійти з некоректним паролем

Електронна пошта

poplavskyiwork@gmail.co

Пароль

.....

**Аккаунту з такими даними не було знайдено!**

**Увійти**

Рис. 3.20 - Спроба увійти з некоректною поштою

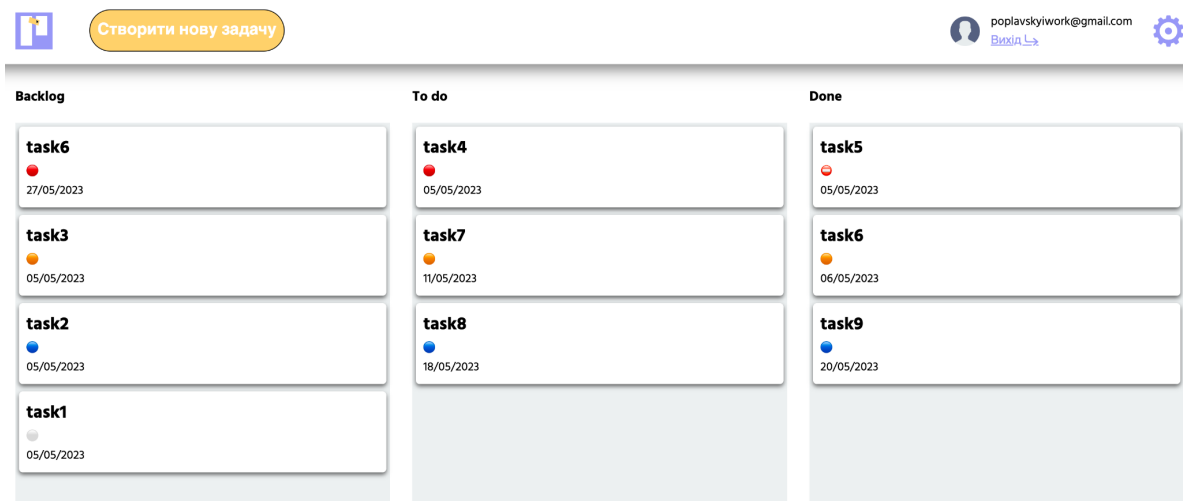


Рис. 3.21 - Успішний вхід з коректними даними

Як бачимо наш додаток відпрацьовує коректно з реєстрацією та входом. Настав час протестувати його роботу з картками та колонками.

Для початку створимо нову картку аби перевірити коректність роботи.

Рис. 3.22 - Заповнення полів попапу для створення нової картки

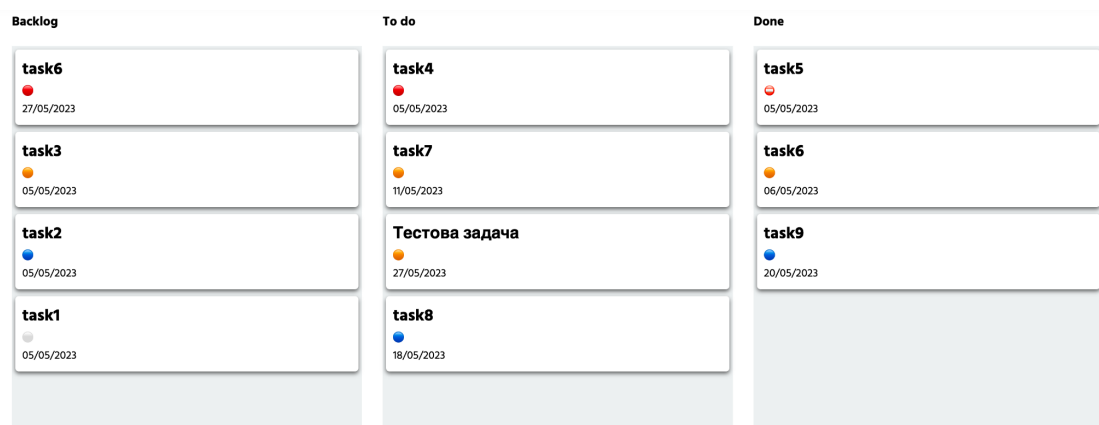


Рис. 3.23 - Дошка з новою карткою

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |

Спробуємо тепер змінити якісь дані в картці.

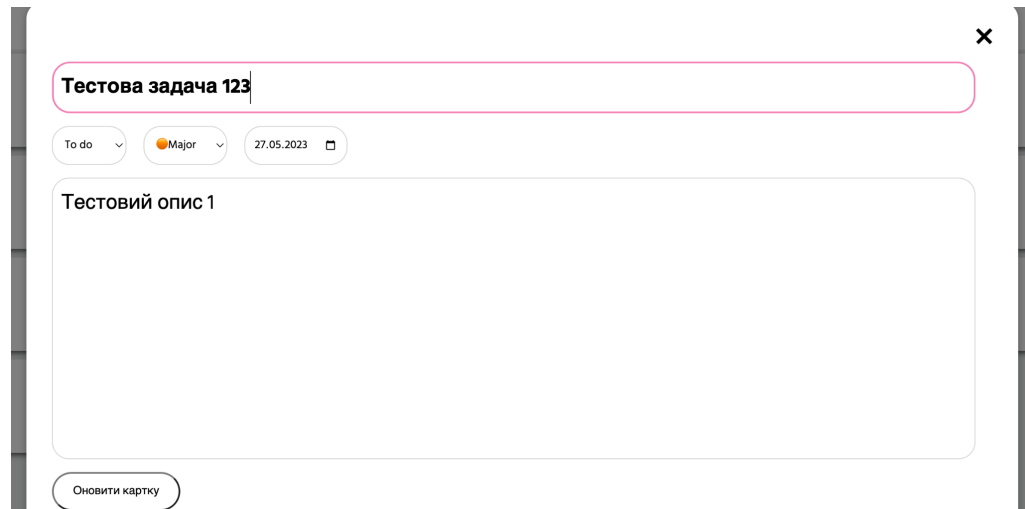


Рис. 3.24 - Оновлення картки

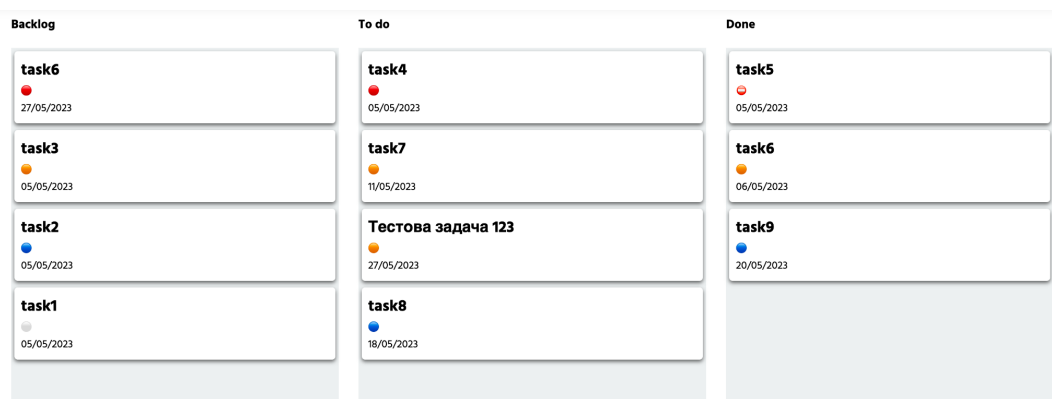


Рис. 3.25 - Дошка з оновленою картою

Як ми бачимо функціонал оновлення картки працює коректно, та картка оновилась.

Тепер спробуємо перетягнути картку по колонкам. Як бачимо зараз створена нами картка знаходиться в колонці “To do”. Зараз перетягнемо її в “Done”, а потім в “Backlog”.

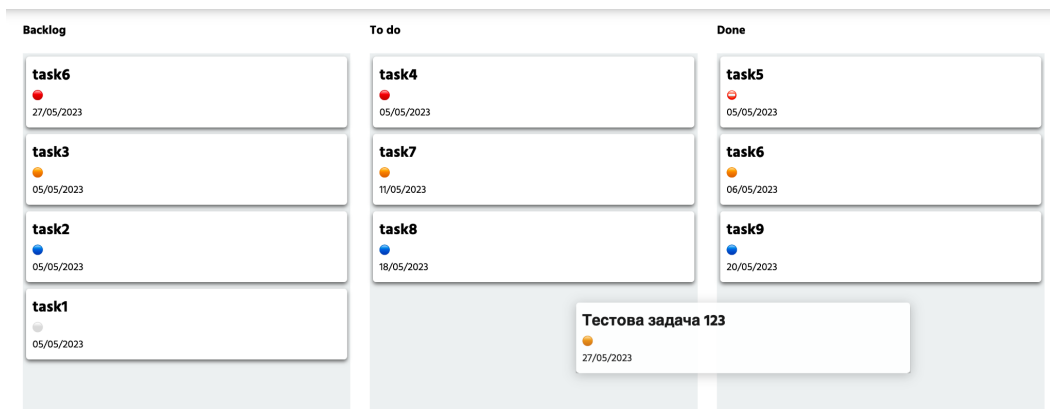


Рис. 3.26 - Процес переміщення картки між колонками

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

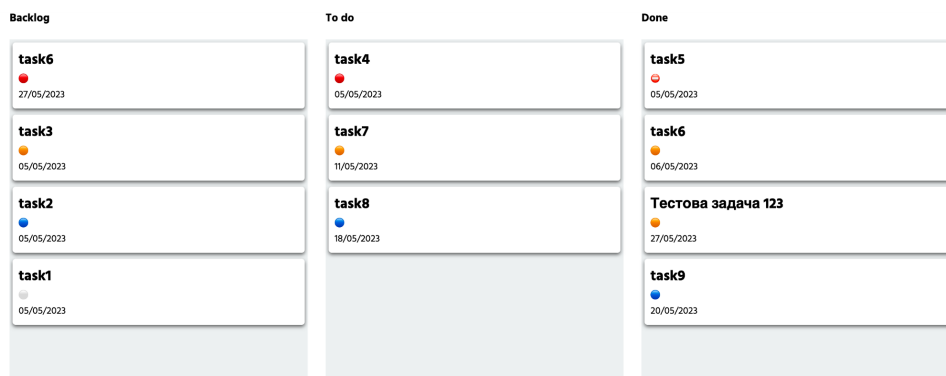


Рис. 3.27 - Картка в колонці “Done”

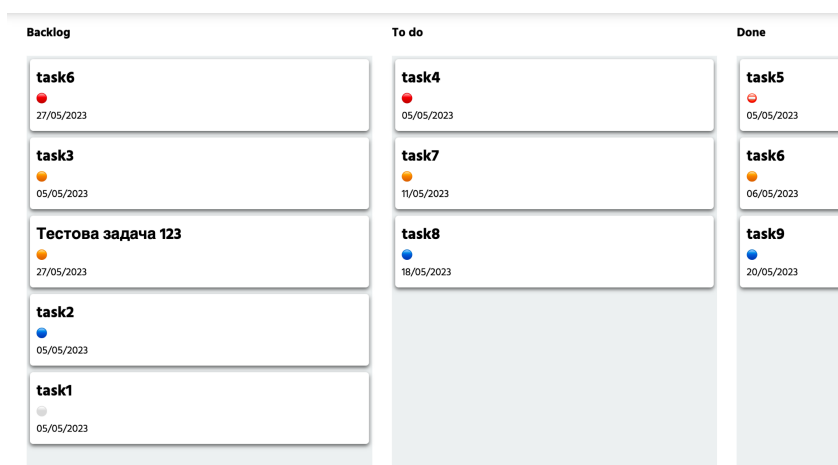


Рис. 3.28 - Картка в колонці “Backlog”

Отже, як видно на рисунках, перетягування між різними колонками працює коректно.

Тепер можемо перейти до останнього функціоналу - до роботи із колонками.

Для роботи з колонками в нас створений наступний попап, наразі він доступний будь-якому користувачу, проте в майбутньому це необхідно буде поправити аби тільки окрема група користувачів мала до цього доступ.

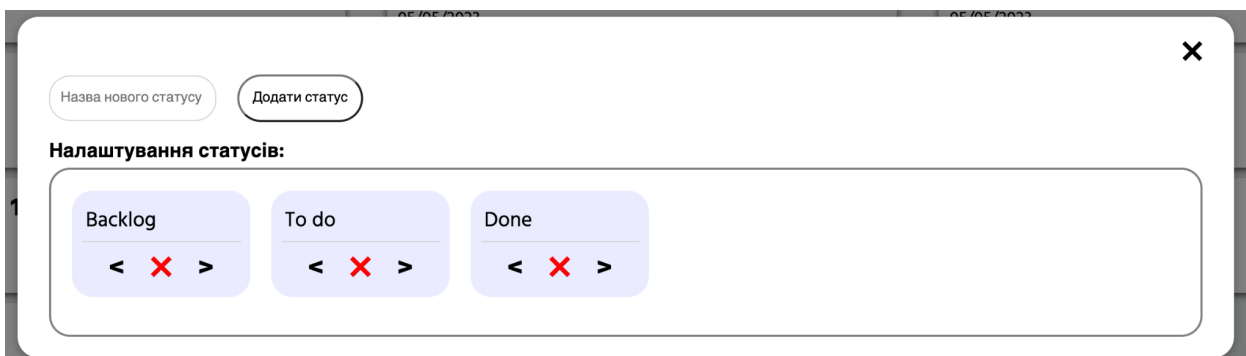


Рис. 3.29 - Попап для роботи з колонками

Тут ми можемо створювати, видаляти та рухати колонки на рисунках нижче буде це продемонстровано.

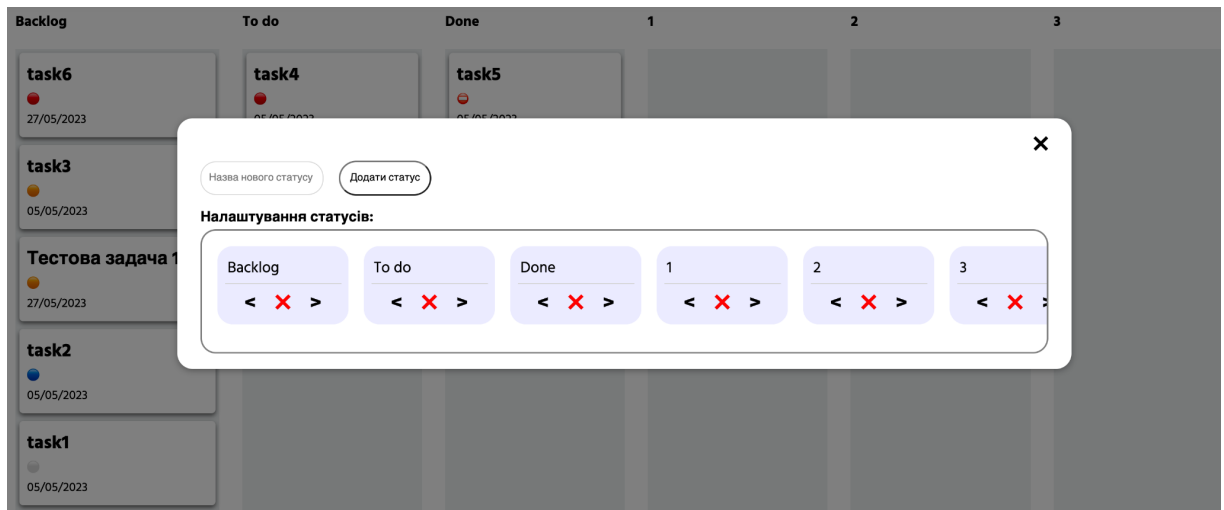


Рис. 3.30 - Створені нові колонки

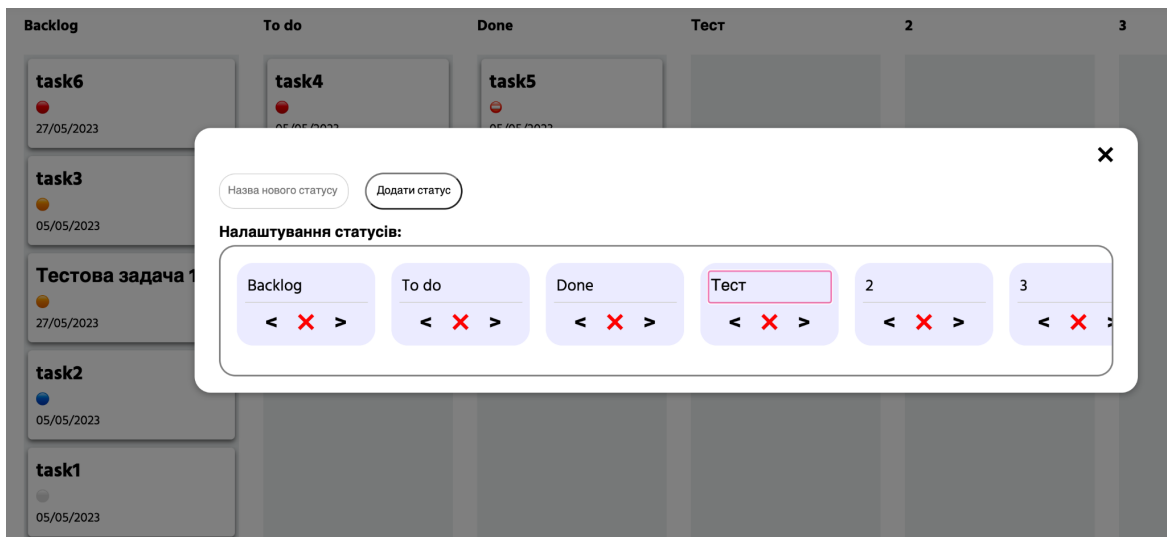


Рис. 3.31 - Оновлення заголовку статусу

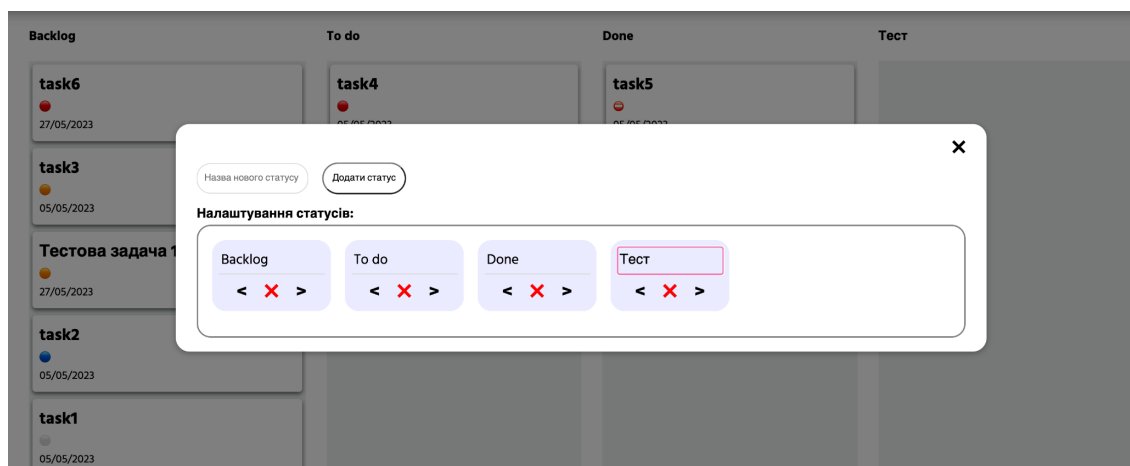


Рис. 3.32 - Видалення колонок

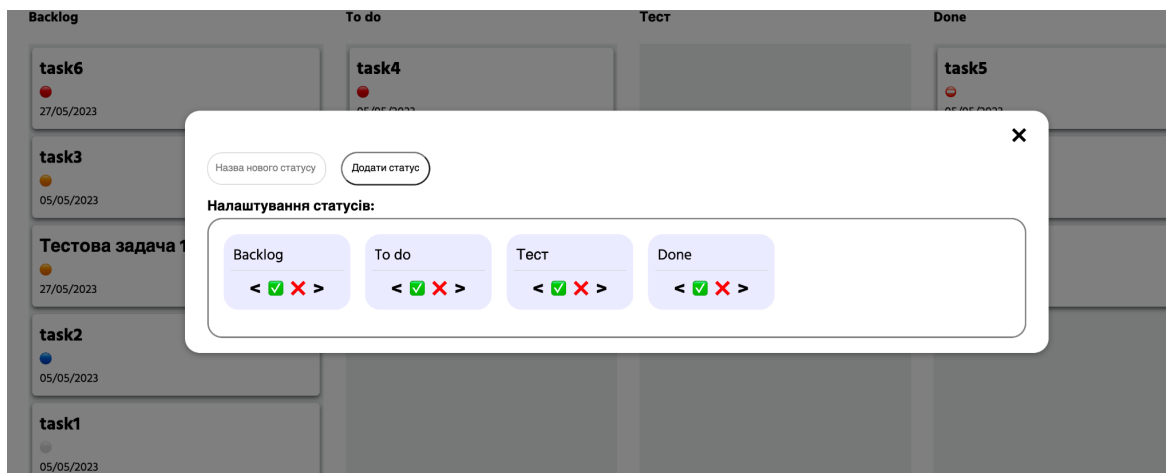


Рис. 3.33 - Зміна порядку колонок

### Висновки до третього розділу

В цьому розділі була розроблена архітектура додатку. Завдяки проектуванню діаграм визначили основні об'єкти моделей в системі.

Було описано роботу додатку. На основі якої зрозумілі можливості користувача, а саме: можливість ввійти в систему, зареєструватись, після чого він потрапляє на сторінку з Kanban дошкою де може додавати, переміщувати, редагувати картки, додавати, оновлювати та видаляти колонки.

Було проведено тестування веб-додатку. виправленні незначні помилки. В цілому додаток придатний до використання і без проблем виконує мінімальний набір функцій для подібних додатків.

## ВИСНОВКИ

В результаті виконання випускної роботи бакалавра було спроектовано та реалізовано “WEB додаток для управління IT проектами (Kanban дошка, аналог Trello)”. Під час роботи над цим проектом було вирішено ряд завдань:

1. Поставлене завдання на роботу.
2. Розглянуто методологію Kanban. Проаналізовані існуючі аналоги. Визначено мінімально необхідний набір функцій для майбутнього додатку. Також визначені функції на майбутнє для покращення і розширення можливостей додатку.
3. Визначено набір технологій для розробки додатку. Як базу даних було обрано NoSQL базу даних - Firestore, яку надає платформа Firebase. Інтерфейс користувача писався за допомогою мови JavaScript, а саме фреймворк Vue.js.
4. Були визначені системні, функціональні, нефункціональні вимоги, також основні бізнес вимоги та історії користувачів до додатку. Вони описані в першому розділі, деякі з них у вигляді схем.
5. Визначено архітектуру системи, що розробляється, встановлено залежності між програмними компонентами.
6. Було проведено розробку загального і функціонального алгоритмів веб-додатку. Також наведено частково код і пояснено, як проходило виконання роботи. Описано інструкцію користування веб-додатком та проведено тестування.

Отже, на даному етапі, ми маємо готовий продукт, з набором необхідних функцій, які повністю задовольняють мінімальні потреби користувача. Даний програмний продукт має приємний та зручний інтерфейс, та простий і зрозумілий у використанні.

Також, на цьому ж етапі, ми маємо чітко видний вектор розвитку програмного додатку на майбутнє. Це стосується як покращення існуючого функціоналу так і написання нового задля розширення можливостей та залучення нової аудиторії.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційна документація Vue.js [Електронний ресурс] – Режим доступу до ресурсу: <https://vuejs.org/guide/introduction.html>
2. Офіційний сайт платформи NodeJS [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/uk/>.
3. Консорціум Всесвітньої Павутини [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3.org/>.
4. Документація платформи Firebase [Електронний ресурс] – Режим доступу до ресурсу: <https://firebase.google.com/docs>
5. Платформа Firebase [Електронний ресурс] - Режим доступу до ресурсу: <https://firebase.google.com/>
6. Jira [Електронний ресурс] - Режим доступу до ресурсу: <https://www.atlassian.com/ru/software/jira>
7. Trello [Електронний ресурс] - Режим доступу до ресурсу: <https://trello.com/en>
8. 18 найкращих програм для відстеження помилок [Електронний ресурс] - Режим доступу до ресурсу: <https://uk.myservername.com/18-best-bug-tracking-software>
9. Model-View-ViewModel (MVVM) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.techtarget.com/whatis/definition/Model-View-ViewModel/>.
10. Kanban короткий посібник [Електронний ресурс] - Режим доступу до ресурсу: <https://www.atlassian.com/ru/agile/kanban>
11. Deployment\_diagram [Електронний ресурс]. – Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Deployment\\_diagram](https://en.wikipedia.org/wiki/Deployment_diagram)
12. Use case diagram [Електронний ресурс]. – Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Use\\_case\\_diagram](https://en.wikipedia.org/wiki/Use_case_diagram)
13. Kanban основні принципи [Електронний ресурс] - Режим доступу до ресурсу: <https://l-a-b-a.com/blog/1529-kanban-principy-i-polza>
14. Що таке Kanban [Електронний ресурс] - Режим доступу до ресурсу: <https://asana.com/ru/resources/what-is-kanban>

# ДОДАТКИ

## ОСНОВНИЙ КОМПОНЕНТ BoardPage

```
<template>
  <Header></Header>
  <AddingForm :priorities="priorities" :boardStatuses="statuses"/>
  <ConfigurePopup :statuses="statuses"/>
  <div id="userInfoWrapper">
    
    <div>
      <p id="userEmail">{{ userEmail }}</p>
      <p id="signOutBtn" @click="signOutHandle">Вихід □</p>
    </div>
  </div>
  <div id="columnsWrapper">
    <div class="status-column-wrapper" :key="boardStatus.id" v-for="boardStatus in statuses">
      <h3 class="status-title">{{ boardStatus.status }}</h3>
      <div @drop="handlingDrop(boardStatus)" @dragenter.prevent @dragover.prevent class="column">
        <div @drag="handlingDrag(card)" :key="card.id" v-for="card in taskCards" draggable="true">
          <Card v-show="!card.hide" :cardPriority="priorities" :taskCard="card" @dblclick="openCard(card)" v-if="card.status===boardStatus.status"></Card>
        </div>
      </div>
    </div>
  </div>
  <CardPopup v-if="showModal" @closePopup="showModal = false" :priorities="priorities" :statuses="statuses" :card="activeCard"></CardPopup>
</template>

<script>

import Header from "../Header.vue"
import Card from "../Card.vue"
import CardPopup from "../CardPopup.vue"
import AddingForm from "../AddingForm.vue"
import ConfigurePopup from "../ConfigurePopup.vue"
import { onMounted, ref } from "vue"
import { db } from "@/firebase"
import { collection, onSnapshot, doc, updateDoc, query, orderBy } from "firebase/firestore"
import { getAuth, signOut } from "firebase/auth"
import { useRouter } from "vue-router";

export default {
```

```

components: {
  Header,
  Card,
  CardPopup,
  AddingForm,
  ConfigurePopup
},
setup() {
  const statuses = ref({})
  const taskCards = ref({})
  const priorities = ref({})
  let showModal = ref(false);
  let activeCard = ref({});
  let draggedCard = ref({})
  let droppedStatus = ref({})
  const userEmail = localStorage.getItem("userEmail");
  const router = useRouter();
  let auth = getAuth();

  const signOutHandle = () => {
    signOut(auth).then(() => {
      router.push("/")
      localStorage.removeItem("userEmail")
    })
  }

  const openCard = (card) => {
    showModal.value = true;
    activeCard.value = card;
  }

  const handlingDrag = (card) => {
    card.hide = true;
    draggedCard.value = card
  }

  const handlingDrop = (boardStatus) => {
    droppedStatus.value = boardStatus;
    updateDoc(doc(db, "cards", draggedCard.value.id), {
      status: droppedStatus.value.status
    });
  }

  const sortedQuery = query(collection(db, 'cards'), orderBy('priority', 'asc'),
    orderBy('dueDate', 'asc'))

```

```

const orderedStatuses = query(collection(db, 'statuses'), orderBy('order', 'asc'))
onMounted(() => {
  onSnapshot(collection(db, "priority"), (querySnapshot) => {
    const fbPriority = [];
    querySnapshot.forEach((doc) => {
      const priority = {
        name: doc.data().name,
        symbol: doc.data().symbol,
      }
      fbPriority.push(priority)
    });
    priorities.value = fbPriority
  });
  onSnapshot(sortedQuery, (querySnapshot) => {
    const fbCards = [];
    querySnapshot.forEach((doc) => {
      const card = {
        id: doc.id,
        name: doc.data().name,
        status: doc.data().status,
        priority: doc.data().priority,
        dueDate: doc.data().dueDate,
        description: doc.data().description
      }
      fbCards.push(card)
    });
    taskCards.value = fbCards
  });
  onSnapshot(orderedStatuses, (querySnapshot) => {
    const fbStatuses = [];
    querySnapshot.forEach((doc) => {
      const status = {
        id: doc.id,
        status: doc.data().status,
        order: doc.data().order
      }
      fbStatuses.push(status)
    });
    statuses.value = fbStatuses
  });
})

return {
  statuses,
  taskCards,
  priorities,
  showModal,

```

```

        openCard,
        activeCard,
        draggedCard,
        droppedStatus,
        handlingDrag,
        handlingDrop,
        sortedQuery,
        userEmail,
        signOutHandle
    }
}
}
</script>

<style scoped>

#columnsWrapper {
    margin: 30px 50px 0px 50px;
    display: flex;
    overflow-x: scroll;
    justify-content: space-between;
}

.status-title {
    margin-bottom: 20px;
}

.status-column-wrapper {
    display: flex;
    flex-direction: column;
    flex-grow: 1;
}

.column {
    min-width: 210px;
    min-height: 630px;
    background-color: #ecf0f1;
    margin-right: 30px;
    padding: 5px;
}

#columnsWrapper .status-column-wrapper:last-child .column{
    margin-right: 0;
}

#userInfoWrapper {
    display: flex;

```

```

align-items: center;
position: absolute;
top: 20px;
right: 120px;
}
#userInfoWrapper img {
width: 40px;
height: 40px;
margin-right: 15px;
}
#signOutBtn {
cursor: pointer;
color: #9797F7;
text-decoration: underline;
}
</style>

```

## Компонент Card

```

<template>
<div id="cardsWrapper">
  <div class="card" >
    <h2>{{ taskCard.name }}</h2>
    <p>{{ getPrioritySymbol(taskCard.priority) }}</p>
    <p>{{ getCorrectDateFormat(taskCard.dueDate) }}</p>
  </div>
</div>
</template>

```

```

<script>

import { ref } from "vue"

export default {
  name: 'CardItem',
  props: ['taskCard', 'cardPriority'],
  components: {

  },

  setup(props) {
    const symbol = ref("")
    const getPrioritySymbol = (name) => {
      const s = props.cardPriority.find(item => item.name === name);
      return s.symbol;
    }
  }
}

```

```

const getCorrectDateFormat = (date) => {
  const unformatDate = new Date(date)
  return unformatDate.toLocaleDateString('en-GB')
}
return {
  getPrioritySymbol,
  getCorrectDateFormat,
  symbol
}
},
}

</script>

<style scoped>

.card {
background-color: white;
padding: 10px;
border-radius: 5px;
margin-bottom: 10px;
box-shadow: 0px 3px 6px 0px rgba(0,0,0,0.6);
overflow: hidden;
}

</style>

```

## Компонент CardPopup

```

<template>
<div id="popupBackground">
  <div id="popup">
    <div id="popupContentWrapper">
      <span @click="closePopup" id="closeButton2">&#215;</span>
      <div id="fieldsWrapper">
        <input class="card-title" v-model="newCardName" type="text">
        <div class="select-inputs">
          <select v-model="newStatus" name="" id="">
            <option :value="status.status" :key="status.id" v-for="status in
statuses">{{ status.status }}</option>
          </select>
          <select v-model="newPriority" name="" id="">
            <option :value="priority.name" :key="priority.id" v-for="priority in
priorities">{{ priority.symbol + priority.name }}</option>
          </select>

```

```

        <input v-model="newDue" type="date">
      </div>
      <textarea v-model="newDescription" placeholder="Тут може бути новий опис
задачки" name="" id="" cols="30" rows="8"></textarea>
      <button @click="updateCard(card)" id="updateButton">Оновити картку</button>
    </div>
  </div>
</div>
</div>
</template>

<script>

import { ref } from "vue"
import { db } from "@/firebase"
import { doc, updateDoc } from "firebase/firestore"

export default {
  name: 'CardPopup',
  methods: {
    closePopup() {
      this.$emit('closePopup')
    }
  },
  props: ['card', 'statuses', 'priorities'],

  setup(props, context) {

    const newCardName = ref(props.card.name)
    const newPriority = ref(props.card.priority)
    const newStatus = ref(props.card.status)
    const newDue = ref(props.card.dueDate)
    const newDescription = ref(props.card.description)

    const updateCard = (card) => {
      updateDoc(doc(db, "cards", card.id), {
        name: newCardName.value,
        status: newStatus.value,
        priority: newPriority.value,
        dueDate: newDue.value,
        description: newDescription.value
      });
      context.emit('closePopup')
    }
  }
}

```

```

    }

    return {
      updateCard,
      newCardName,
      newDue,
      newPriority,
      newStatus,
      newDescription
    }
  }
}

```

```
</script>
```

```
<style scoped>
```

```

#closeButton2 {
  font-size: 50px;
  line-height: 1;
  cursor: pointer;
  margin-left: 98%;
}

```

```

textarea {
  font-size: 24px;
}

```

```
</style>
```

## Компонент ConfigurePopup

```

<template>
  <button class="open-popup-button" id="settingsBtn"
@click="showPopup=true">&#9881;</button>
  <div v-if="showPopup">
    <div id="popupBackground">
      <div id="popup">
        <div id="popupContentWrapper">
          <p @click="showPopup=false" id="closeButton">&#215;</p>
          <div id="addStatusWraper">
            <input id="statusInput" v-model="newStatus" type="text"
placeholder="Назва нового статусу">
            <button class="column-popup-button" @click="addStatus">Додати
статус</button>
          </div>
        </div>
      </div>
    </div>
  </div>

```

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

```
<h3>Налаштування статусів:</h3>
<div id="statusesAreaWrapper">
  <div class="status-wrapper" :key="status.id" v-for="status in statuses">
    <input class="status-name" v-model="status.status">
    <div class="update-buttons">
      <button id="backButton" class="moving-arrows"
@click="moveStatusBack(status)"><lt;</button>
      <p @click="updColumn(status)" id="updButton">✔</p>
      <p @click="removeStatus(status)" id="deleteButton">❌</p>
      <button id="forwardButton" class="moving-arrows"
@click="moveStatusForward(status)">>></button>
    </div>
  </div>
</div>
</div>
</div>
</div>
</div>
</template>

<script>

import { ref } from "vue"
import { db } from "@/firebase"
import { collection, addDoc, updateDoc, doc, deleteDoc } from "firebase/firestore"

export default {
  name: 'ConfigurePopup',
  components: {
  },
  props: ['statuses'],
  setup(props) {

    let showPopup = ref(false);
    const newStatus = ref("");
    let newOrder1 = 0;
    let newOrder2 = 0;

    const updColumn = (status) => {
      updateDoc(doc(db, "statuses", status.id), {
        status: status.status
      });
    }

    const moveStatusBack = (status) => {
      const els = props.statuses.filter( => .order < status.order);
```

```

    if (els.length === 1) {
      newOrder1 = els[0].order / 2
    } else {
      newOrder1 = (els[els.length - 1].order + els[els.length - 2].order) / 2;
    }
    updateDoc(doc(db, "statuses", status.id), {
      order: parseInt(newOrder1)
    });
  }

  const moveStatusForward = (status) => {
    const els = props.statuses.filter(_ => _.order > status.order);
    if (els.length === 1) {
      newOrder2 = els[0].order + 100
    }
    else {
      newOrder2 = (els[0].order + els[1].order) / 2;
    }
    updateDoc(doc(db, "statuses", status.id), {
      order: parseInt(newOrder2)
    });
  }

  const addStatus = () => {
    let result = props.statuses.map(status => status.order);
    addDoc(collection(db, "statuses"), {
      status: newStatus.value,
      order: Math.max(...result) + 100
    })
    newStatus.value = ""
  }

  const removeStatus = (status) => {
    deleteDoc(doc(db, "statuses", status.id));
  }

  return {
    newStatus,
    showPopup,
    addStatus,
    removeStatus,
    moveStatusBack,
    moveStatusForward,
    updColumn,
    newOrder1,
    newOrder2,
  }
}

```

```

}
}
</script>

<style scoped>
#settingsBtn {
  font-size: 70px;
  line-height: .7;
  color: #9797F7;
  background: none;
  border: none;
  position: absolute;
  top: 24px;
  right: 50px;
  z-index: 100;
}

#statusesAreaWrapper {
  display: flex;
  justify-content: flex-start;
  overflow-x: scroll;
  border: 2px solid gray;
  border-radius: 20px;
  padding: 20px;
}

.status-wrapper {
  max-width: 150px;
  min-width: 150px;
  padding: 10px;
  background-color: white;
  margin-right: 20px;
  background-color: #EBEBFF;
  border-radius: 20px;
}

#deleteButton, #updButton {
  font-size: 20px;
  cursor: pointer;
}

#statusInput {
  border: 1px solid lightgray;
  margin-bottom: 15px;
  padding: 10px;
  border-radius: 20px;
  margin-right: 20px;
}

```

```

}

.column-popup-button {
  background: none;
  padding: 10px;
  border-radius: 50px;
  width: 120px;
  cursor: pointer;
}

.status-wrapper input {
  border-bottom: 1px solid lightgray;
  border-top: none;
  border-left: none;
  border-right: none;
  width: 145px;
  padding: 3px;
  font-size: 20px;
  margin-bottom: 10px;
  background-color: #EBEBFF;
}

.status-wrapper .status-order-input {
  width: 25px;
  padding: 3px;
  border-radius: 0;
}

.update-buttons {
  width: 100px;
  margin: 0 auto;
  display: flex;
  justify-content: space-between;
}

#closeButton {
  line-height: .7;
  margin-left: 98%;
}

.moving-arrows {
  background-color: transparent;
  border: none;
  font-size: 25px;
  line-height: 1.3;
  font-weight: bold;
  cursor: pointer;
}

```

```
}
</style>
```

## Компоненти Login and Register pages

```
<template>
<div id="contentWrapper">
  <div id="formWrapper">
    <form id="form" action="" @submit.prevent="login">
      <h1 id="heading">Раді бачити знову!</h1>
      <h3 id="heading2">Дякуємо за довіру!</h3>
      <label for="emailInput">Електронна пошта</label>
      <input v-model="email" id="emailInput" type="email">
      <label for="passInput">Пароль</label>
      <input v-model="password" id="passInput" type="password">
      <p id="errBox" v-if="errMsg">{{ errMsg }}</p>
      <button type="submit" id="button">Увійти</button>
    </form>
  </div>
  <div id="backgroundWrapper">

</div>
</div>
</template>

<script setup>
import { ref } from "vue";
import { getAuth, signInWithEmailAndPassword } from "firebase/auth"
import { useRouter } from "vue-router";

const email = ref("");
const password = ref("");
const router = useRouter();
const errMsg = ref("");
const login = () => {
  signInWithEmailAndPassword(getAuth(), email.value, password.value)
    .then(() => {
      localStorage.setItem("userEmail", email.value)
      router.push('/board')
    })
    .catch((error) => {
      switch(error.code) {
        case "auth/invalid-email": errMsg.value = "Некоректна пошта!";
        break;
        case "auth/user-not-found": errMsg.value = "Аккаунту з такими даними не було знайдено!";
        break;
      }
    })
  }
}
```

```

case "auth/wrong-password": errMsg.value = "Некоректний пароль!";
    break;
    default: errMsg.value = "Пошта або пароль некоректні!";
    break;
    }
    })
}

</script>

<style scoped>

#contentWrapper {
    display: flex;
}

#formWrapper {
    background-color: #EBEBFF;
    width: 1200px;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
}

#backgroundWrapper {
    background: url("../assets/bg7.png") no-repeat top right fixed;
    background-size: contain;
    height: 100vh;
    width: 100%;
}

#form {
    display: flex;
    flex-direction: column;
}

#form * {
    margin-bottom: 20px;
}

#form input {
    width: 400px;
    border-radius: 25px;
    border: none;

```

```
padding: 5px;
font-size: 24px;
}

#form label {
font-size: 24px;
color: #9797F7;
}

#button {
border-radius: 25px;
border: 1px solid #000000;
font-size: 30px;
margin-top: 20px;
background-color: #FFD16A;
color: white;
font-weight: bold;
width: 410px;
}

#heading {
font-size: 50px;
color: #9797F7;
}

#heading2 {
font-size: 30px;
color: #9797F7;
}

</style>
```

```
<template>
<div id="contentWrapper">
  <div id="formWrapper">
    <form id="form" action="" @submit.prevent="register">
      <h1 id="heading">Ласкаво просимо!</h1>
      <h3 id="heading2">Давайте зареєструємось!</h3>
      <label for="emailInput">Електронна пошта</label>
      <input v-model="email" id="emailInput" type="email">
      <label for="passInput">Пароль</label>
      <input v-model="password" id="passInput" type="password">
      <p id="errBox" v-if="errMsg">{{ errMsg }}</p>
      <button type="submit" id="button">Зареєструватись</button>
      <a href="#/login" id="loginLink">Вже маєте аккаунт?</a>
    </form>
  </div>
</div>
```

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |

```

</div>
<div id="backgroundWrapper">
</div>
</div>
</template>

<script setup>
import { ref } from "vue";
import { getAuth, createUserWithEmailAndPassword } from "firebase/auth"
import { useRouter } from "vue-router";

const email = ref("");
const password = ref("");
const router = useRouter();
const errMsg = ref("");

const register = () => {
  createUserWithEmailAndPassword(getAuth(), email.value, password.value)
    .then(() => {
      localStorage.setItem("userEmail", email.value)
      router.push('/board')
    })
    .catch((error) => {
      switch(error.code) {
        case "auth/weak-password": errMsg.value = "Пароль повинен мати мінімум 6 символів!";
        break;
        case "auth/email-already-in-use": errMsg.value = "Користувач з такою поштою вже зареєстрований!";
        break;
      }
    })
}

</script>

<style scoped>

#contentWrapper {
  display: flex;
}

#formWrapper {
  background-color: #EBEBFF;
  width: 1200px;

```

```

display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
height: 100vh;
}

#backgroundWrapper {
background: url("../assets/bg6.png") no-repeat top right fixed;
background-size: contain;
height: 100vh;
width: 100%;
}

#form {
display: flex;
flex-direction: column;
}

#form * {
margin-bottom: 20px;
}

#form input {
width: 400px;
border-radius: 25px;
border: none;
padding: 5px;
font-size: 24px;
}

#form label {
font-size: 24px;
color: #9797F7;
}

#button {
border-radius: 25px;
border: 1px solid #000000;
font-size: 30px;
margin-top: 20px;
background-color: #FFD16A;
color: white;
font-weight: bold;
width: 410px;
}

```

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |

```
#heading {
  font-size: 50px;
  color: #9797F7;
}

#heading2 {
  font-size: 30px;
  color: #9797F7;
}

#loginLink {
  margin: 0 auto;
  color: #9797F7;
}

</style>
```

## Компонент StartPage

```
<template>
<Header></Header>
<div id="contentWrapper">
  <div class="main-content">
    <div id="mainInfoWrapper">
      <p class="main-text">ОРГАНІЗУЙТЕ СВОЇ ЩОДЕННІ ЗАВДАННЯ</p>
      <p class="additional-text">Миттєво створюйте завдання за допомогою фантастичних
безкоштовних функцій</p>
      <a href="#/register"><span class="button">Рушаймо</span></a>
    </div>
    
  </div>
  <div class="main-content">
    
    <div >
      <p class="main-text">ПС, ДРУЖЕ, В НАС ТУТ КАНБАН ДОШКА</p>
      <p class="additional-text">Спробуй, і ти зрозумієш, що таке любов з першого
дотику, методології це також стосується;</p>
      <a href="#/register"><span class="button">Рушаймо</span></a>
    </div>
  </div>
  <div class="main-content">
    <div >
      <p class="main-text">МИ НЕ ЗРОБИМО РОБОТУ ЗА ТЕБЕ...</p>
      <p class="additional-text">...проте, з нашим додатком ти зможеш продуктивно
працювати з проектами, і спати спокійно після роботи</p>
      <a href="#/register"><span class="button">Рушаймо</span></a>
    </div>
  </div>
</div>
```

```

    
  </div>
</div>
</template>

```

```

<script>

```

```

import Header from "../Header.vue"

```

```

export default {
  name: 'StartPage',
  components: {
    Header,
  },
}

```

```

</script>

```

```

<style scoped>

```

```

#contentWrapper{
  background-color: #EBEBFF;
  padding-bottom: 60px;
}
.main-content {
  display: flex;
  align-items: center;
  justify-content: space-around;
  padding: 20px 200px;
}
.main-text {
  width: 770px;
  font-weight: bold;
  font-size: 70px;
  line-height: 1.15;
  color: #9797F7;
  margin-bottom: 30px;
}
.additional-text {
  font-size: 32px;
  line-height: 1.15;
  width: 770px;
  margin-bottom: 30px;
  color: #9797F7;
}
.button {
  display: block;
  width: 200px;

```

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |

```

text-align: center;
border-radius: 50px;
padding: 10px;
cursor: pointer;
background-color: #FFD16A;
border: 1px solid #000000;
color: #ffffff;
font-weight: bold;
font-size: 30px;
}
a {
color: black;
text-decoration: none;
font-weight: 500;
font-size: 19px;
}
.bg {
width: 40%;
}
#header {
background-color: #EBEBFF;
}
</style>

```

## App.vue

```

<template>
  <router-view></router-view>
</template>

<script>

export default {
  }
</script>

<style>
* {
margin: 0;
padding: 0;
font-family: 'Hind', sans-serif;
user-select: none;
}

#popupBackground {
position: fixed;
z-index: 9998;

```

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |

```

top: 0;
left: 0;
width: 100%;
height: 100%;
background-color: rgba(0, 0, 0, 0.5);
display: table;
transition: opacity 0.3s ease;
}

#popup {
display: table-cell;
vertical-align: middle;
}

#popupContentWrapper {
width: 1100px;
margin: 0px auto;
padding: 20px 30px;
background-color: #fff;
border-radius: 20px;
box-shadow: 0 2px 8px rgba(0, 0, 0, 0.33);
transition: all 0.3s ease;
}

#closeButton {
font-size: 50px;
line-height: 1;
cursor: pointer;
}

#fieldsWrapper,
#addingForm {
display: flex;
flex-direction: column;
}

#fieldsWrapper input,
#fieldsWrapper select,
#fieldsWrapper textarea {
border: 1px solid lightgray;
margin-bottom: 15px;
padding: 10px;
border-radius: 20px;
margin-right: 20px;
}

textarea {

```

```

resize: none;
}

#fieldsWrapper button {
  background: none;
  padding: 10px;
  border-radius: 50px;
  width: 150px;
  cursor: pointer;
}

.card-title {
  font-size: 24px;
  font-weight: bold;
}

.select-inputs {
  display: flex;
}

.header-wrapper {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 20px;
}

.header-wrapper h1 {
  width: 250px;
}

::-webkit-scrollbar {
  width: 0px;
  background: transparent;
}

#errBox {
  color: red;
  font-weight: bold;
  padding: 10px;
  border: 3px solid red;
  width: 380px;
  text-align: center;
}

```

```
@import
url('https://fonts.googleapis.com/css2?family=Hind:wght@300;400;500;600;700&display=sw
ap');

</style>
```

## firebase/index.js

```
import { initializeApp } from "firebase/app";
import { getFirestore } from "firebase/firestore";
import { getAuth } from "firebase/auth";

const firebaseConfig = {
  apiKey: "AIzaSyCGhfxPzunyB-dFyQSKmUtyXLNOP04IKpc",
  authDomain: "pmttool-a407b.firebaseio.com",
  projectId: "pmttool-a407b",
  storageBucket: "pmttool-a407b.appspot.com",
  messagingSenderId: "155369036334",
  appId: "1:155369036334:web:94ac015fc5e52cd9d5ddbdf",
  measurementId: "G-65QS8V6E13"
};

const app = initializeApp(firebaseConfig);
const db = getFirestore(app);
const auth = getAuth(app);

export {
  db,
  auth,
}
```

## router.js

```
import { createRouter, createWebHashHistory } from 'vue-router'
import StartPage from "./components/StartPage.vue"
import BoardPage from "./components/BoardPage.vue"
import RegisterPage from "./components/RegisterPage.vue"
import LoginPage from "./components/LoginPage.vue"
export default createRouter({
  history: createWebHashHistory(),
  routes: [
    { path: "/", component: StartPage, alias: "/" },
    { path: "/board", component: BoardPage },
    { path: "/register", component: RegisterPage },
    { path: "/login", component: LoginPage },
  ]
})
```

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |