**Vision and Scope Document**

**for**

**Research on the Performance and Efficiency of State Managers for Vue-Based Web Applications**

**Version 1.0**

**approved**

**Prepared by**

**Kornev Olexander**

**NURE**

**03.06.2025**

## 1. Постановка мети.

Ціллю роботи є дослідження та порівняння продуктивності й ефективності різних state-менеджерів для веб-додатків, розроблених на основі фреймворку Vue.js. Зокрема, об'єктами дослідження виступають Vuex, Pinia та Composition API. Результатом роботи є створення клієнт-серверного веб-застосунку, який дозволяє в умовах реального використання протестувати вказані підходи до управління станом.

Для реалізації дослідження було обрано сучасні технології, зокрема:

* Клієнтська частина: реалізована з використанням JavaScript/TypeScript, фреймворку Vue.js, Pinia/Vuex/Composition API як state-менеджерів, Vue Router для маршрутизації, бібліотеки Axios для виконання HTTP-запитів та Tailwind CSS для стилізації інтерфейсу. Розробка організована у вигляді mono-repo на базі NX.
* Серверна частина: реалізована на мові Go з використанням архітектури Clean Layered Architecture. Вона забезпечує обробку запитів, бізнес-логіку та взаємодію з базою даних через окремі шари Transport, Service та Storage.
* База даних: використовується реляційна СУБД PostgreSQL, яка зберігає інформацію про дошки та ідеї. Для забезпечення цілісності й ефективності даних реалізовані індекси, зовнішні ключі та автоматична генерація міток часу.

Для аналізу ефективності state-менеджерів застосовуються такі інструменти: Chrome DevTools, Lighthouse, Jest та Cypress. Система реалізована відповідно до принципів трирівневої архітектури з чітким розподілом функцій між клієнтською частиною, сервером і базою даних.

## 2. Загальний опис системи.

Програмна система являє собою клієнт-серверний веб-додаток для управління ідеями та дошками, реалізований на базі фреймворку Vue.js. Система слугує базовим інструментом для дослідження продуктивності різних state-менеджерів (Vuex, Pinia, Composition API) та дозволяє моделювати типові сценарії використання у SPA-додатках.

Веб-додаток надає користувачу можливість створювати, редагувати, видаляти й шукати дошки й ідеї, а також працювати з ними в зручному UI. Завдяки реалізованому функціоналу, система може бути використана як тестовий полігон для порівняльного аналізу продуктивності state-менеджерів у реальному проєкті.

Клієнтська частина реалізована за допомогою Vue.js і підтримує модульну архітектуру. Залежно від цілей дослідження, додаток може бути зібраний із використанням Vuex, Pinia або Composition API. Для маршрутизації використовується Vue Router, для обміну даними із сервером — Axios. Інтерфейс має адаптивний і мінімалістичний дизайн, побудований із використанням Tailwind CSS.

Серверна частина побудована на мові Go із дотриманням принципів Clean Layered Architecture. Вона відповідає за обробку HTTP-запитів, реалізацію бізнес-логіки та взаємодію з базою даних. Сервер забезпечує CRUD-операції над дошками та ідеями, а також обробку пошукових запитів. Для зберігання даних використовується реляційна СУБД PostgreSQL, що забезпечує високу продуктивність та цілісність даних.

Таким чином, програмна система є гнучким середовищем для аналізу й тестування state-менеджерів у Vue-додатках, а також демонструє практичне застосування сучасних архітектурних і технологічних підходів у веб-розробці.

## 3. Основний функціонал системи

У системі існує лише один тип користувача — User, який має можливість взаємодії з дошками (Boards) та ідеями (Ideas). Основна мета реалізованої системи — забезпечити можливість порівняння state-менеджерів у реальних умовах на базовому, але функціонально повноцінному веб-додатку.

Оскільки робота зосереджена на аналізі state-менеджменту, реалізовано набір функцій, які дозволяють створити типове навантаження на систему стану (див. табл. 3.1).

Таблиця 3.1 – Основний функціонал системи

|  |  |  |
| --- | --- | --- |
| № | Функція | Опис |
| 1 | Створення дошки | Дозволяє користувачу створити нову дошку з назвою. Дошка стає контейнером для ідей. |
| 2 | Редагування дошки | Дозволяє змінити назву існуючої дошки. |
| 3 | Видалення дошки | Видаляє дошку разом із усіма пов’язаними ідеями. |
| 4 | Пошук дошок | Дає змогу знайти дошку за назвою через пошукове поле. |
| 5 | Перегляд списку дошок | Відображає всі доступні дошки у вигляді списку. |
| 6 | Створення ідеї | Дозволяє створити нову ідею, прив’язану до певної дошки. |
|  | Редагування ідеї | Дозволяє змінювати назву або опис ідеї. |
|  | Видалення ідеї | Видаляє конкретну ідею з дошки. |
|  | Пошук ідей | Дає змогу шукати ідеї за назвою або фрагментом опису. |
|  | Перегляд списку ідей на дошці | Відображає всі ідеї, що належать до обраної дошки, у вигляді карток. |
|  | Архівування ідей | Дозволяє приховати ідеї, що неактуальні, без їх повного видалення. |
|  | Перемикання режимів відображення | Дає змогу користувачу обирати між списковим і картковим переглядом. |

Продовження таблиці 3.1

|  |  |  |
| --- | --- | --- |
|  | Керування станом через обраний state-менеджер | Увесь функціонал системи реалізується з можливістю вибору одного з трьох state-менеджерів: Vuex, Pinia або Composition API. Це дозволяє оцінити різницю у продуктивності. |

## 4. Загальні обмеження

Програмна система для дослідження state-менеджерів у веб-додатках має такі обмеження:

* веб-додаток не працює без доступу до інтернету, оскільки для взаємодії між клієнтом і сервером використовуються HTTP-запити;
* система не передбачає авторизацію або реєстрацію користувачів, оскільки основна мета – тестування state-менеджерів, а не моделювання повноцінного користувацького життєвого циклу;
* розроблений функціонал орієнтований на одного користувача без паралельного доступу або багатокористувацької взаємодії;
* тестування ефективності state-менеджерів проводиться лише на клієнтській частині без урахування можливих змін продуктивності при використанні SSR або хмарних рішень;
* продуктивність оцінюється в умовах середнього навантаження (до 10 000 записів), що не охоплює масштабні корпоративні сценарії;
* відсутня повноцінна система безпеки, оскільки безпека не є предметом дослідження в межах поставленої теми;

## 5. Припущення та залежності

Програмна система для дослідження state-менеджерів у Vue-додатках має наступні припущення та залежності:

* користувач має доступ до стабільного інтернет-з’єднання, оскільки система працює за клієнт-серверною архітектурою;
* запуск клієнтської частини передбачає підтримку сучасного браузера (Google Chrome, Firefox, Edge), що підтримує ES6+ та Vue.js;
* серверна частина передбачає встановлене середовище виконання Go та доступ до PostgreSQL-сервера;
* користувач буде тестувати систему в середовищі, що відповідає вимогам (локальна машина або dev-сервер з Node.js, Go, PostgreSQL);
* для тестування state-менеджерів користувач повинен змінити відповідну конфігурацію проекту перед збіркою (використання Vuex, Pinia або Composition API задається на етапі складання);
* структура бази даних повинна бути попередньо створена за наданим SQL-дампом або через ORM-інструменти;
* користувачі будуть створювати велику кількість ідей і дошок для імітації реального навантаження при тестуванні продуктивності.