УДК 004.415.2

АРХІТЕКТУРА ВЕБПЛАТФОРМИ ПРОВЕДЕННЯ ОНЛАЙН-ВІКТОРИН У РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Маренич Ф. А.
0668090426fed@gmail.com
Черкаський державний фаховий бізнес-коледж
Марченко С. В.
м.Черкаси, Україна

Інтерактивні цифрові платформи дедалі ширше застосовуються в освітньому середовищі для підтримки активних форм навчання та підвищення залученості учасників. Одним із найбільш поширених форматів є онлайн-вікторини, що поєднують перевірку знань із елементами гейміфікації. У науковій літературі гейміфікація визначається як використання елементів ігрового дизайну в неігрових контекстах, а її практична цінність полягає у стимулюванні мотивації та підтримці активної взаємодії користувачів із цифровим середовищем [1]. Водночас для платформ цього класу вирішальними є не лише мотиваційні механіки, а й архітектурні властивості системи, передусім здатність підтримувати узгоджений стан гри між усіма учасниками.

Ключовою технологічною вимогою до платформ проведення вікторин є взаємодія у режимі реального часу, оскільки під час гри активне запитання, відповіді учасників і таблиця лідерів мають синхронно оновлюватися для всіх клієнтів. Для таких сценаріїв традиційна модель періодичних HTTP-запитів є менш ефективною через надлишкові мережеві накладні витрати, тоді як WebSocket-підхід забезпечує сталі двонаправлені з’єднання і меншу затримку передавання подій [2]. Аналіз наявних SaaS-платформ для онлайн-вікторин свідчить, що їхні функціональні можливості часто супроводжуються обмеженнями щодо конфігурації, масштабування та керування логікою сесії. Це зумовлює потребу в архітектурному рішенні, оптимізованому саме для інтенсивного обміну короткими подіями в межах керованої ігрової сесії.

Розробити архітектуру вебплатформи проведення онлайн-вікторин у режимі реального часу, що забезпечує синхронізацію стану гри між учасниками та підтримує масштабування кількості підключених клієнтів.

Архітектуру запропонованої платформи представлено за допомогою C4-моделі, яка використовується для ієрархічного опису програмних систем і дає змогу чітко відобразити взаємодію користувачів, клієнтського застосунку та серверних компонентів (рис. 1). Користувачі системи поділяються на дві ролі: організатор, який створює вікторини та керує ігровою сесією, і гравці, які приєднуються до гри за PIN-кодом. Клієнтська частина взаємодіє з сервером двома каналами: REST API використовується для роботи зі статичними даними, тоді як передавання ігрових подій реалізовано через SignalR.

Серверна частина платформи реалізована як модульний моноліт. Такий вибір є обґрунтованим саме для системи, у якій основне навантаження формується не великою кількістю незалежних бізнес-процесів, а швидким циклом обробки коротких подій у межах однієї сесії. На відміну від мікросервісної архітектури, модульний моноліт у такому випадку дає змогу уникнути міжсервісних мережевих викликів, зменшити інфраструктурну складність і зберегти чітке логічне розмежування відповідальностей усередині застосунку. Сучасні дослідження також фіксують зворотний рух частини систем від мікросервісів до монолітів або модульних монолітів у випадках, коли накладні витрати на розподіленість перевищують виграш від неї [4].

До основних компонентів серверної частини належать REST API Controllers, SignalR Quiz Hub, Game State Manager і Data Access Layer. У цій конфігурації критично важливим є те, що архітектура фактично реалізує server-authoritative підхід: єдиним джерелом істини щодо стану гри виступає сервер. Усі події, ініційовані клієнтами, спочатку валідуються та обробляються на сервері, і лише після цього оновлений стан транслюється учасникам. Для сценарію онлайн-вікторини це принципово, оскільки усуває розбіжності між клієнтами, спрощує контроль таймерів, порядку переходів між запитаннями та алгоритмів нарахування балів.

Центральним компонентом системи є Game State Manager, який відповідає за збереження транзиторного стану активних ігор: поточного запитання, балів гравців, таймерів, службових прапорців сесії. Критичний аналіз цього рішення показує, що винесення такого стану безпосередньо в постійне сховище на кожній події створило б надлишкові I/O-затримки і погіршило б часові характеристики системи. Тому використання in-memory моделі для активних сесій є архітектурно виправданим: база даних у такому разі зберігає лише персистентні сутності та результати завершених ігор, тоді як оперативний ігровий цикл не залежить від швидкості дискових операцій. Це рішення не усуває потреби в подальшому масштабуванні, однак на рівні прототипу і цільового сценарію воно забезпечує кращий баланс між продуктивністю, простотою реалізації та керованістю системи [3; 4].

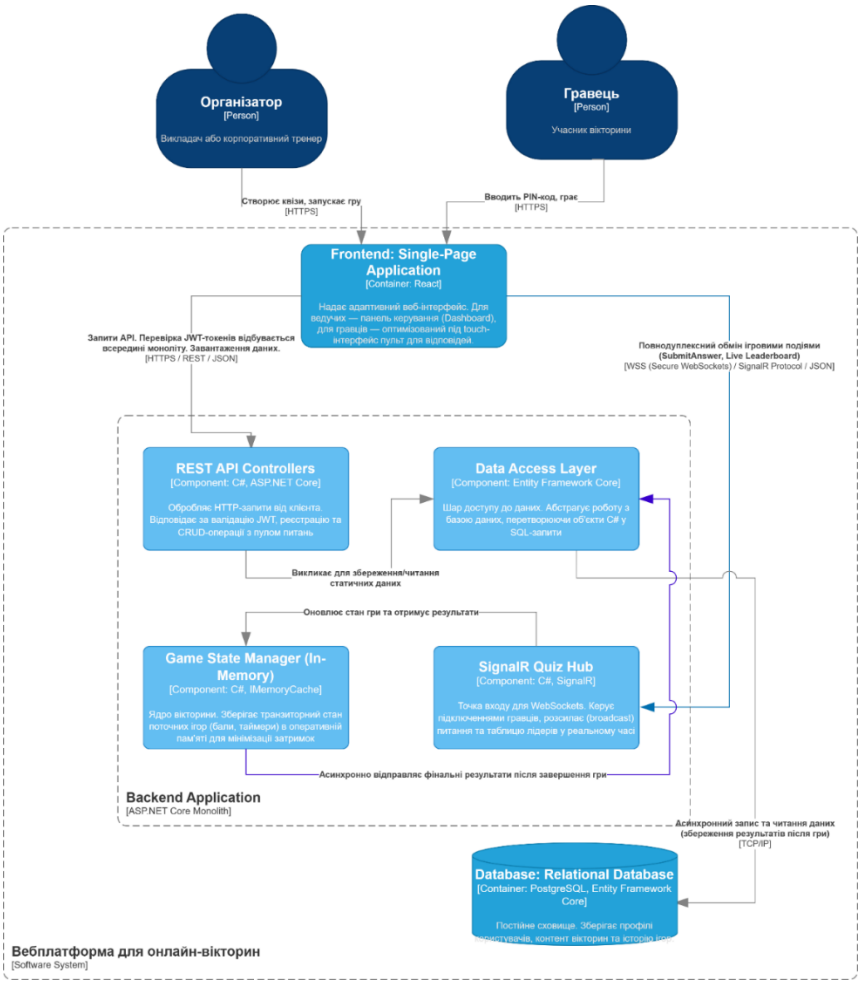

Рисунок 1 – Компонентна архітектура вебплатформи за нотацією C4

Передавання ігрових подій між сервером і клієнтами здійснюється через SignalR, який надає прикладний рівень для організації real-time взаємодії, підтримує хаби, групування підключень і механізми повторного з’єднання [5]. У межах запропонованої архітектури це дає змогу ізолювати події окремих ігрових сесій і транслювати їх лише відповідним учасникам. Таким чином, C4-діаграма не лише описує складники системи, а й відображає її головну архітектурну ідею: розмежування статичного API, realtime-каналу та окремого шару керування станом гри.

У роботі запропоновано архітектуру вебплатформи проведення онлайн-вікторин у режимі реального часу. Її побудовано як modular monolith із чітким відокремленням REST-взаємодії, каналу передавання подій у режимі реального часу та модуля керування станом гри. Критичний аналіз показав, що для системи такого класу обраний підхід є обґрунтованішим за мікросервісну декомпозицію на ранньому етапі, оскільки дозволяє зменшити латентність, уникнути надмірної інфраструктурної складності та забезпечити узгоджений стан ігрової сесії.

Список використаних джерел:

1. Deterding S., Dixon D., Khaled R., Nacke L. From game design elements to gamefulness: defining gamification // Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments. Tampere, 2011. P. 9-15. DOI: 10.1145/2181037.2181040.
2. Pimentel V., Nickerson B. G. Communicating and displaying real-time data with WebSocket // IEEE Internet Computing. 2012. Vol. 16, No. 4. P. 45-53. DOI: 10.1109/MIC.2012.64.
3. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th ed. Boston : Addison-Wesley Professional, 2021. 464 p.
4. Su R., Li X., Taibi D. Back to the future: from microservice to monolith // Electronics. 2024. Vol. 13, No. 8. Art. 1452. DOI: 10.3390/electronics13081452.

5. Microsoft. Overview of ASP.NET Core SignalR. URL: https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-10.0 (дата звернення: 07.03.2026).

УДК 004.932:681.2

ІНЖЕНЕРІЯ ПОРТАТИВНИХ ЗАСОБІВ ВИМІРЮВАННЯ ТА ОПРАЦЮВАННЯ ДАНИХ

Євтушенко Д. В.
yevtushenkodv@outlook.com
Черкаський державний фаховий бізнес коледж
Фальченко Н. Г.
м.Черкаси, Україна

У сучасній комп’ютерній інженерії все більшої популярності набувають автономні пристрої моніторингу з можливістю обробки даних у реальному часі. Однією з ключових складових таких систем є модуль збору даних, який не лише відповідає за взаємодію з сенсорами, а й виконує функції з фільтрації сигналів, керування енергоспоживанням та забезпечення обміну інформацією з зовнішніми мережами. У портативній системі збору даних апаратно-програмна частина відіграє критичну роль у створенні точного, енергоефективного та надійного інструменту вимірювання [1].

Основні завдання розробки включають: обґрунтування компонентної бази, розробку структурної схеми, реалізацію алгоритмів первинної обробки сигналів, організацію передачі даних, а також забезпечення високої автономності пристрою [2].

Система реалізована на базі енергоефективного мікроконтролера, а взаємодія з сенсорним обладнанням базується на використанні цифрових інтерфейсів I2C та SPI. Головною перевагою такого підходу є швидкість зчитування та можливість прямого отримання даних без складних аналогових перетворень. Усі операції з опитування датчиків та локальних обчислень виконуються з високою частотою дискретизації. Передача обробленої