Автор:
Казмиренко Денис Юрійович
студент 4 курсу
спеціальності 121 Інженерія
програмного забезпечення
Науковий керівник:
Галицький Олександр Вадимович
Кандидат педагогічних наук, доцент,
доцент кафедри комп'ютерної та
програмної інженерії

РОЗРОБЛЕННЯ AI-СЕРВІСНОЇ ПЛАТФОРМИ ДЛЯ КРОСПЛАТФОРМЕНОЇ СИСТЕМИ УНІВЕРСИТЕТСЬКОГО РОЗКЛАДУ

Анотація. Робота присвячена розробці Alure – self-hosted AI-сервісної платформи для кросплатформної системи університетського розкладу. Актуальна концепція проєкту зміщує акцент з абстрактної backend-платформи на цілісну систему з власним сервером університету, web-адмінкою, Android-клієнтом, майбутньою desktop-адаптацією, CLI-утилітою керування, модульною базою даних і AI-помічником. Alure не є просто календарем: це середовище, у якому адміністратор створює й супроводжує розклад, викладач бачить власні заняття та зміни, а студент отримує персональний денний і тижневий розклад. AI використовується як допоміжний механізм для пошуку конфліктів, рекомендації слотів, пояснення рішень і формування повідомлень, але остаточне прийняття критичних рішень залишається за людиною [1, 2, 5].

Ключові слова: Alure, університетський розклад, self-hosted платформа, Dart, Serverpod, Flutter, AI-помічник, web-адмінка, конструктор розкладу, SQLite, PostgreSQL, Moodle, iCalendar.

Вступ. Розклад у закладі вищої освіти є не лише таблицею занять, а системою координації груп, викладачів, аудиторій, часу, повідомлень і щоденних змін. При ручному або напівручному складанні розкладу виникає багато проблем: зайнятість викладача може перетинатися з іншою парою, аудиторія може не відповідати місткості або обладнанню, а зміни часто поширюються через різні канали й швидко втрачають актуальність. Тому сучасна система розкладу повинна не тільки зберігати події, а й перевіряти правила, вести історію змін, повідомляти користувачів і пояснювати, чому саме таке рішення було обране.

Alure пропонується як платформа, у якій розклад залишається ядром, а всі модулі працюють навколо нього. Self-hosted модель означає, що університет має власний екземпляр системи, власну базу даних і контроль над налаштуваннями. Це важливо для приватності, резервного копіювання, локальної політики доступу та поступового впровадження функцій. У першій версії проєкт не потребує надмірної мікросервісної складності: доцільніше створити стабільний монолітний backend із логічно розділеними модулями та чіткими API-контрактами [1, 3, 4].

Постановка задачі. Необхідно розробити платформу Alure, яка забезпечує створення, редагування, публікацію й супровід університетського розкладу. Система має підтримувати ролі студента, викладача, адміністратора і супер-адміністратора. Для адміністратора потрібно передбачити web-панель із головною сторінкою, довідниками викладачів, груп та аудиторій, сторінками налаштувань, AI-аналітики й конструктором розкладу. Для студента і викладача потрібно сформувати прості персональні екрани денного та тижневого розкладу, картки пар, повідомлення про зміни й налаштування аккаунта.

Також потрібно описати серверну частину: CLI-команди для запуску й обслуговування, файлову структуру, базу даних, резервне копіювання, оновлення, авторизацію, 2FA, аудит, обробку помилок і fallback-поведінку при недоступності інтеграцій. AI-модулі мають допомагати, а не автономно керувати системою: вони рекомендують вільні слоти, підказують конфлікти, пояснюють рішення та формують допоміжні повідомлення.

Мета роботи. Метою роботи є формування оновленого проєктного опису AI-сервісної платформи Alure відповідно до її актуальної архітектури, функцій і візуальної логіки. Для цього необхідно уточнити межі MVP, ролі користувачів, структуру web-адмінки, backend-архітектуру, модель даних, механізми безпеки, інтеграції, AI-функції та напрями подальшого розвитку.

Основна частина. Alure розгортається як self-hosted система за принципом «один університет – один сервер – один екземпляр». Технічний адміністратор керує нею через CLI-утиліту alure. Команди alure init і alure setup створюють базову структуру, конфігурацію, директорії, БД і міграції; alure start запускає сервер; alure doctor перевіряє порти, права, стан БД, дисковий простір і конфіг; alure network змінює host, port, public URL та allowed origins; alure backup create/list/restore відповідає за резервні копії; alure update виконує оновлення з попереднім бекапом. Така модель робить систему зрозумілою для супроводу й аварійного відновлення.

Backend реалізується на Dart із Serverpod як монолітний сервіс із логічно відокремленими модулями [1]. Він є єдиним джерелом правди: frontend не вирішує самостійно, чи можна зберегти конфліктну пару, показати персональні дані або виконати адміністративну дію. Сервер перевіряє авторизацію, ролі, глобальні правила розкладу, валідність даних, стан інтеграцій, кеш і політики приватності. Для звичайних операцій використовуються API-контракти, а критичні realtime-події, наприклад скасування пари або зміна аудиторії, передаються активними каналами оновлення.

База даних поділяє постійні сутності, тимчасові стани й події. Для MVP доцільним є SQLite, а для більшого production-навантаження – PostgreSQL [3, 4]. Основні сутності: User, Group, Teacher, Room, Schedule, Lesson, LessonOverride, Attendance, ConsentRecord, інтеграційні конфіги та audit log. Група містить код, курс, факультет, куратора і email-списки студентів; викладач – ПІБ, корпоративну пошту, факультет, кафедру і преференції; аудиторія – корпус, номер, місткість, обладнання, статус і зайнятість. Пара пов’язує предмет, тип заняття, групу, викладача, аудиторію, день, слот, тижневість і можливі разові зміни.

Web-адмінка є головним інструментом керування, бо саме в браузері зручно працювати з великими таблицями, тижневими сітками та drag-and-drop. Її стиль – мінімалістичний light blue dashboard: світло-блакитний фон, білі картки, сині акценти, округлені елементи, пошук, фільтри, стабільні модальні вікна й права панель із деталями вибраної сутності. Головна сторінка показує стан системи, останні зміни та швидкі дії. Розділи викладачів, груп і аудиторій виконують роль довідників і одночасно джерел даних для конструктора розкладу.

Конструктор розкладу є центральною адміністративною функцією. Він містить тижневу сітку, незаплановані пари, drag-and-drop, підсвічування слотів, перевірку зайнятості викладачів, груп та аудиторій, а також модальні вікна додавання і редагування пар. Якщо глобальні правила забороняють конфлікт, збереження блокується; якщо дозволяють зберегти виняток, система показує попередження і записує дію в аудит. Після публікації зміни мають бути доставлені користувачам через застосунок, push, email, Telegram або календарні підписки.

Студентський і викладацький інтерфейси значно простіші за адміністративний. Студент бачить денний розклад, тижневий огляд, картку пари, статус, аудиторію, викладача, час і повідомлення про зміни. Викладач бачить власні заняття, групи, аудиторії, навантаження і службові повідомлення. Android-клієнт орієнтується на швидкий доступ, нижню навігацію, push-сповіщення і перегляд розкладу без складних адміністративних таблиць [2].

Безпека будується на Google Workspace login, fallback-вході через пошту, токен і 2FA, а для адміністративного доступу 2FA є обов’язковим. Система зберігає записи згод, історію входів, зміни ролей, зміни розкладу та критичні адміністративні дії, щоб було зрозуміло, хто, коли і чому змінив пару, аудиторію або правило.

AI у Alure не є автономним керівником. Він допомагає адміністратору швидше аналізувати ситуацію: рекомендує слоти, підказує аудиторії з потрібною місткістю та обладнанням, пояснює конфлікти, формує короткі повідомлення, підсумовує ризики й може підтримувати Study Buddy-підбір за курсами та вільними вікнами. Генератор розкладу доцільно розділити на deterministic engine, який перевіряє жорсткі правила, і AI assistant, який пропонує зручні варіанти та пояснення. Критичні рішення й публікація розкладу залишаються за адміністратором.

Інтеграції мають бути модульними: Moodle надає дані про курси й дедлайни [7], iCalendar і Google Calendar використовуються для експорту та підписок [5, 6], а Telegram, email і push – для сповіщень. Якщо інтеграція недоступна, базовий розклад усе одно залишається доступним у Alure.

Висновки. Оновлена концепція Alure показує, що проєкт є не просто календарем або окремим backend-модулем, а self-hosted AI-сервісною платформою для університетського розкладу. Її ядром є розклад, а довідники, конструктор, нотифікації, інтеграції, бекапи, аудит, права доступу й AI-підказки працюють навколо нього. Для MVP доцільною є монолітна Serverpod-архітектура на Dart, Flutter-клієнти, web-адмінка, Android-застосунок, SQLite як початкова БД і PostgreSQL як production-опція.

Перевага Alure полягає в балансі між адміністративною керованістю та простотою для кінцевого користувача. Адміністратор отримує інструменти створення і контролю розкладу, а студент і викладач - зрозумілі персональні екрани без зайвої складності. AI виступає пояснюваним помічником, а не заміною відповідальної особи. Подальший розвиток варто спрямувати на UI-прототипи, реалізацію MVP, тестування конструктора, перевірку backup/restore, інтеграції з Moodle і календарями та формалізацію критеріїв приймання.

Список використаних джерел

1. Serverpod. Serverpod Documentation [Електронний ресурс]. – Режим доступу: https://docs.serverpod.dev/ (дата звернення: 29.04.2026).
2. Flutter. Flutter Documentation [Електронний ресурс]. – Режим доступу: https://docs.flutter.dev/ (дата звернення: 29.04.2026).
3. SQLite. Write-Ahead Logging [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/wal.html (дата звернення: 29.04.2026).
4. PostgreSQL. PostgreSQL Documentation [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/ (дата звернення: 29.04.2026).
5. IETF. RFC 5545: Internet Calendaring and Scheduling Core Object Specification (iCalendar) [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc5545 (дата звернення: 29.04.2026).
6. Google. Google Calendar API v3 — API Reference [Електронний ресурс]. – Режим доступу: https://developers.google.com/workspace/calendar/api/v3/reference (дата звернення: 29.04.2026).
7. Moodle Developer Docs. External Services API [Електронний ресурс]. – Режим доступу: https://moodledev.io/docs/apis/subsystems/external (дата звернення: 29.04.2026).
8. Кархут В.Я., Галицький О.В. (2025). Інтеграція принципів вебдоступності в підготовку майбутніх розробників програмного забезпечення. *Науковий часопис Українського державного університету імені Михайла Драгоманова Серія 2 Комп'ютерно-орієнтовані системи навчання,* №24 (31). С. 109-117. https://doi.org/10.31392/UDU-nc.series2.2025.24(31).10
9. Рибачек Д. С., Галицький О. В. Віртуальні навчальні середовища як інструмент інклюзивної освіти. *Наукові записки. Серія: Педагогічні науки,* 215. 2024. С. 257-263. DOI: https://doi.org/10.36550/2415-7988-2024-1-215-257-263