МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук, фізики та математики

Кафедра комп'ютерних наук та програмної інженерії

РОЗРОБЛЕННЯ ТА ОПТИМІЗАЦІЯ ІНТЕРФЕЙСУ МОДУЛЯ KSU24 «ЦИФРОВИЙ РОЗКЛАД» ДЛЯ ВИСОКО НАВАНТАЖЕНИХ ОСВІТНІХ СИСТЕМ

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти "бакалавр"

Виконав: здобувач 4 курсу 12-431 групи

Спеціальності: 122 Комп'ютерні науки

Освітньо-професійної програми: Комп'ютерні науки

Безкревний Владислав Олександрович

Керівник: кандидатка технічних наук, доцентка кафедри комп'ютерних наук та програмної інженерії
Шишко Людмила Станіславівна

Рецензент: кандидатка педагогічних наук, доцентка кафедри загальнофахової підготовки та морської безпеки Херсонської державної морської академії
Зайцева Т.В.

Івано-Франківськ – 2026

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

1. API — прикладний програмний інтерфейс (Application Programming Interface).
2. ARIA — набір спеціальних атрибутів розмітки, що дозволяють програмам екранного доступу коректно інтерпретувати стан і призначення елементів вебсторінки (Accessible Rich Internet Applications).
3. DOM — об'єктна модель документа (Document Object Model).
4. EAA — Європейський акт про доступність (European Accessibility Act).
5. JARS — загальносистемна технологія для організації динамічних тимчасових сховищ даних без змін у структурі основної бази даних системи (JSON Archive and Registry System).
6. JSON — документно-орієнтований формат представлення та зберігання даних у вигляді об'єктів (JavaScript Object Notation).
7. KSU24 — корпоративне цифрове освітнє середовище або інформаційна система Херсонського державного університету.
8. UI — інтерфейс користувача (User Interface).
9. UX — користувацький досвід (User Experience).
10. WCAG — міжнародний стандарт цифрової інклюзивності та доступності вебконтенту (Web Content Accessibility Guidelines).
11. IA — інформаційна архітектура (Information Architecture).

ВСТУП

Актуальність теми. Модуль управління розкладом є одним із ключових компонентів корпоративної інформаційної системи університету: його якість безпосередньо визначає операційну ефективність диспетчерів і доступність актуальних даних для студентів та викладачів. Використання табличних редакторів і передача розкладів через месенджери призводять до фрагментації даних і ризику синхронізаційних конфліктів — проблем, які усуває централізований цифровий модуль за принципом єдиного джерела істини.

Наявна реалізація вебінтерфейсу модуля «Цифровий розклад» у системі KSU24 не відповідає операційним потребам диспетчерів: відсутнє сіткове відображення тижня, форма створення занять має критичні ергономічні недоліки, відсутня візуалізація конфліктів. Для студентів і викладачів інтерфейс не відповідає стандартам доступності WCAG. Зазначені недоліки обумовлюють необхідність комплексної модернізації клієнтської частини модуля.

Об'єкт дослідження: підсистема відображення та управління розкладом навчальних занять в інформаційному середовищі KSU24.

Предмет дослідження: методи UX/UI-проєктування, алгоритми фронтенд-оптимізації та технології реактивного рендерингу даних, спрямовані на підвищення ергономічності та продуктивності вебінтерфейсів освітніх систем.

Мета роботи: модернізація інтерфейсу модуля «Цифровий розклад» KSU24 для покращення користувацького досвіду та підвищення швидкодії системи шляхом впровадження адаптивних компонентів, оптимізації клієнтської логіки та механізмів асинхронної обробки даних.

Завдання роботи:

1. Провести аналіз предметної галузі та аудит поточного стану модуля розкладу KSU24 для виявлення ергономічних і технічних недоліків.

2. Розробити оптимізовану інформаційну архітектуру та персоналізовані сценарії взаємодії (User Flow) для ключових груп стейкхолдерів (студент, викладач, диспетчер).

3. Спроєктувати адаптивний та інклюзивний інтерфейс користувача з урахуванням стандартів доступності WCAG та принципів зниження когнітивного навантаження.

4. Обґрунтувати вибір програмного стека та реалізувати фронтенд-частину модуля з імплементацією механізмів кешування та мемоїзації рендерингу.

5. Провести тестування розробленого інтерфейсу за участю реальних користувачів для підтвердження ефективності впроваджених рішень.

Методи дослідження: системний аналіз, порівняльний аналіз інтерфейсів, сценарне моделювання, тестування користувацького досвіду.

Практичне значення одержаних результатів. Розроблений оптимізований модуль інтегровано в екосистему KSU24 та впроваджено в навчальний процес університету. Запропоновані рішення дозволять суттєво скоротити час диспетчерів на виконання рутинних операцій, також забезпечили комфортний доступ до розкладу з мобільних пристроїв для студентів і викладачів та підвищили загальну інклюзивність системи.

РОЗДІЛ 1.
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ІСНУЮЧОЇ АРХІТЕКТУРИ РОЗКЛАДУ В СИСТЕМІ KSU24

1.1. Поняття «розклад" та «цифровий розклад" у контексті освітніх систем

Процес складання розкладу може вважатися одним із найбільш критичних бізнес-процесів закладу вищої освіти, що визначає ефективність виконання основної функції установи. У науковій літературі академічний розклад визначається як упорядкований розподіл навчальних заходів: лекцій, практичних і лабораторних занять, семінарів, за часовими слотами та аудиторіями з урахуванням ресурсних і організаційних обмежень. Разом з тим задача складання розкладу є класичною NP-складною комбінаторною задачею оптимізації [1]. Як показують сучасні дослідження, зокрема ґрунтовний аналіз С. Крістіансена та Т. Стідсена [2], цифровізація освітніх процесів суттєво змінює підходи до розв'язання цієї проблеми: фокус зміщується з простої автоматизації розподілу на створення інтерактивних систем підтримки прийняття рішень для диспетчерів навчального процесу.

Розклад у закладі вищої освіти виконує одразу кілька функцій: операційну (координація навчального процесу), ресурсну (ефективне використання аудиторного фонду і кадрового потенціалу) та інформаційну (забезпечення учасників актуальними даними про час і місце проведення занять). Залежно від типу задіяних ресурсів і характеру обмежень розрізняють кілька видів розкладів: навчальний розклад занять, розклад іспитів та заліків. Розклад дзвінків варто виділити окремо як такий, що позначає усталений розподіл робочого часу закладу, а не порядок занять, тобто це умовний довідник, на основі

якого формується розклад занять. Кожен із цих типів має власну специфіку з точки зору представлення даних і вимог до інтерфейсу.

Ключовими стейкхолдерами, що взаємодіють із розкладом, є три принципово різні групи користувачів: студенти, яким потрібно швидко переглянути власний поточний розклад; викладачі, зацікавлені у контролі свого педагогічного навантаження і розподілі аудиторій; диспетчери, що безпосередньо формують і коригують розклад. Ця рольова модель принципово впливає на проєктування інтерфейсу, адже кожна категорія вимагає окремого контексту подання інформації та набору функціональних можливостей.

Цифровий розклад являє собою програмний модуль, що автоматизує як складання, так і публікацію академічного розкладу в електронному вигляді. На відміну від паперового або табличного аналога, цифровий розклад передбачає інтеграцію з інформаційними системами університету, підтримку доступу на основі ролі в системі, механізми автоматичної перевірки конфліктів і оперативного оновлення даних у режимі реального часу.

Перехід від традиційного до цифрового формату розкладу ставить ряд специфічних вимог до системи:

- актуальність даних — внесені зміни мають негайно відображатися в інтерфейсі всіх зацікавлених сторін;
- чіткість структури подання — розклад повинен наочно демонструвати часову прив'язку занять (день, календарний тиждень, номер пари) і забезпечувати швидке орієнтування без необхідності послідовного перегляду списків;
- фільтрація та пошук — користувачі, які формують розклад, повинні мати можливість переглядати розклад у контексті групи, викладача або аудиторії, користувачі, що ним

користуються, повинні мати швидкий та інтуїтивний доступ до власного розкладу на конкретний обраний день;

- адаптивність — інтерфейс кінцевих користувачів має коректно функціонувати як на десктопних пристроях, так і на мобільних, зважаючи на те, що студенти переважно перевіряють розклад «на ходу";
- зручність складання розкладу — для диспетчерів принципово важливою є наочність при додаванні, переміщенні та редагуванні занять, а також контроль конфліктів та розподілу ресурсів у режимі реального часу.

З точки зору принципів проєктування інтерфейсів людино-машинної взаємодії, цифровий розклад є типовим прикладом інтерфейсу з високою інформаційною щільністю. Згідно з фундаментальними дослідженнями Дж. Свеллера у межах теорії когнітивного навантаження [3], основним ризиком при взаємодії з такими обсягами даних є надмірне навантаження на робочу пам'ять користувача та відповідна втрата точності і ефективності роботи. Для їх зниження Б. Шнайдерман у своїх працях з проєктування інтерфейсів рекомендує дотримуватися принципу прогресивного розкриття інформації та використовувати знайомі користувачам ментальні моделі. Прикладом є табличне подання розкладу у вигляді сітки тижня [4].

З урахуванням специфіки режимів роботи перегляду (студент, викладач) і редагування (диспетчер) в сучасній практиці UX-проєктування можливо виокремити відповідні формати подання інформації.

Аналіз існуючих рішень показує, що сіткове подання інформації є найпоширенішим варіантом на веб сайтах закладів вищої освіти, які публікують розклад у відкритий доступ (найчастіше просто у вигляді документа формату опублікованої електронної таблиці, а не як

інтегроване рішення) [5]. Формат списку найчастіше реалізується як адаптивний варіант інтерфейсу для мобільних пристроїв, трансформуючи багатоколонкову таблицю у вертикальний перелік карток занять. Стандартне календарне подання переважно застосовується у сервісах персонального тайм-менеджменту, тоді як для комплексного академічного розкладу воно є менш ефективним через високу щільність подій у кожній окремій заповненій комірці. Що стосується функціоналу редагування, повноцінні інтерактивні модулі для роботи диспетчера над розкладом традиційно реалізуються у вигляді відокремлених десктопних застосунків, тоді як веб платформам здебільшого відводиться функція трансляції вже готових даних.

Досвід проєктування систем свідчить про те, що вибір формату подання напряму впливає на швидкість виконання задачі та рівень задоволеності користувачів.

1.2. Структура даних та принципи формування розкладу в KSU24

Система KSU24 є корпоративним цифровим освітнім середовищем Херсонського державного університету, що реалізує мікросервісну архітектуру управління основними бізнес-процесами закладу. Модуль розкладу є складовою частиною системи, започаткованою ще у сервісі «Деканат” [6], на основі його архітектурних рішень закладено нинішні принципи роботи з розкладом.

Основою моделі даних модуля розкладу є дві основні сутності: розклад (schedule) і заняття (lesson), між якими встановлено зв'язок один до багатьох, тобто один розклад може містити умовно необмежену кількість занять, тоді як кожне заняття повинне належати рівно одному розкладу.

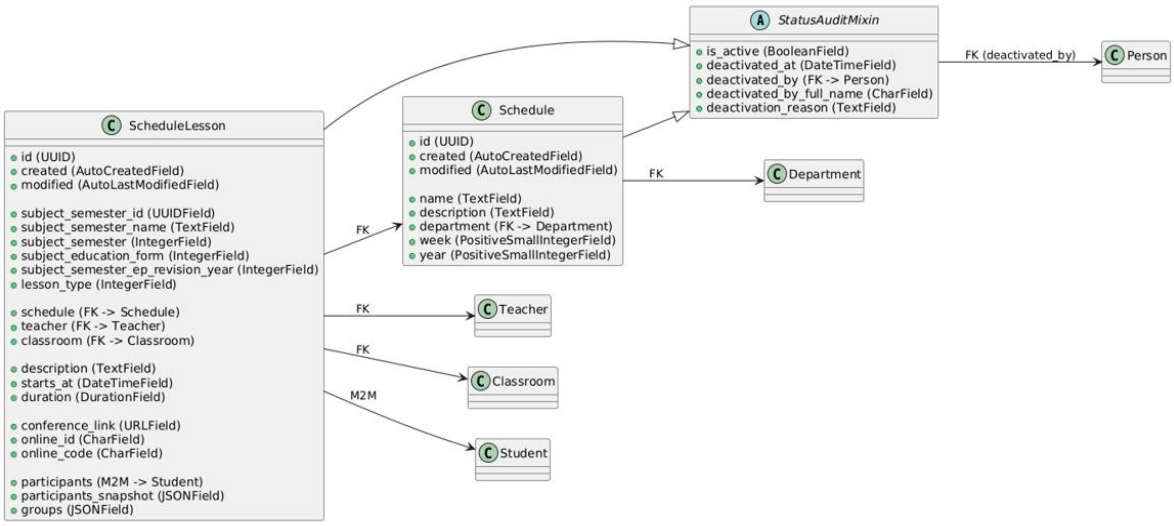

Рис. 1.1. Схема зв'язку сутностей розкладу та заняття у базі даних

Сутність schedule є об'єктом, що виступає своєрідним контейнером для набору занять певного навчального підрозділу на визначений часовий проміжок. Вона містить назву розкладу, опис, прив'язку до відділу в інформаційній системі, а також календарний номер навчального тижня і рік, що визначають контекст, у якому функціонують конкретні заняття.

Сутність lesson безпосередньо описує окреме заняття і містить розгорнутий набір атрибутів, що забезпечують повноту навчального контексту.

Поля на позначення навчальної дисципліни, викладача, переліку учасників заняття зберігаються у денормалізованому вигляді, тобто безпосередньо у документі заняття, що дозволяє уникати зайвих запитів при відображенні розкладу і забезпечує вищу швидкодію під час зчитування даних. Це стандартна практика для систем із переважаючим навантаженням на читання, де дані оновлюються рідше, ніж зчитуються, або де потрібно відображати великий обсяг різноманітних даних [7].

Поля conference_link, online_id та online_code реалізують інтеграцію з платформою відеоконференцій. Система KSU24 інтегрована із Zoom, тому ці поля позначають параметри покликання

саме для цієї платформи. Втім, архітектурно вони є нейтральними відносно конкретної платформи й можуть бути застосовані для будь-якої сумісної системи.

Розглядаючи принцип проміжного зберігання розкладу, варто зазначити, що у процесі модернізації попередню логіку було переосмислено. Структура моделі заняття стосовно цього питання фундаментально не змінилася, була оновлена логіка саме моделі розкладу. Попередня версія мала поля для позначення статусу чернетки, статусу опублікованого, нова версія таких полів не має. Натомість розклади чернетки реалізовано через інтегровану систему архівування та реєстрації даних «JSON Archive and Registry System (скор. JARS)" [8]. JARS є загальносистемною технологією для організації динамічних тимчасових сховищ даних без змін у структурі основної бази даних. У контексті модуля розкладу чернетка представлена JARS-документом, що має структуру, яка зберігає і дані розкладу, і дані переліку занять. Публікація цієї моделі, тобто формування на її основі стандартних сутностей розкладу та занять, додає їх для відображення у кінцевих користувачів.

Основою для цього архітектурного рішення є принцип розподілу відповідальності: основна модель сутності розкладу оперує виключно підтвердженими даними, тоді як JARS забезпечує гнучке зберігання проміжних станів у документно-орієнтованому форматі, не вносячи ускладнень у основну схему бази даних.

Таким чином, у поточній архітектурі система розрізняє два принципово різні стани розкладу:

- чернетка (JARS) — тимчасовий стан розкладу і занять, що існує у форматі JSON-документа і не відображається у персональних розкладах студентів і викладачів;

- опублікований розклад — затверджена версія, що зберігається у реляційній базі даних і стає доступною для всіх залучених учасників навчального процесу.

Такий підхід відповідає принципу «оперативного відображення змін", визначеному серед ключових вимог до цифрових розкладів.

1.3. Критичний аналіз поточного інтерфейсу модуля розкладу KSU24

З метою виявлення функціональних та ергономічних недоліків наявного інтерфейсу модуля розкладу було проведено якісне дослідження у форматі регулярних зустрічей-інтерв'ю з диспетчерами навчального процесу. До участі в опитуванні були залучені диспетчери всіх факультетів Херсонського державного університету, що забезпечило репрезентативність отриманих даних у межах установи. Дослідження було спрямоване на виявлення конкретних сценаріїв використання системи, опис типових труднощів у роботі з інтерфейсом, а також з'ясування того, якою мірою наявний модуль задовольняє реальні операційні потреби диспетчерів.

Частина модулю, призначена для складання розкладу в попередній версії KSU24, складалася з двох вкладок на одній сторінці адміністративної панелі: «Розклади занять" та «Навчальні заняття". Перша вкладка відображала перелік сформованих розкладів у табличному форматі з агрегованими метаданими. Друга вкладка містила безпосередньо перелік занять розкладів з можливістю фільтрації за конкретним розкладом (Рис. 1.2).

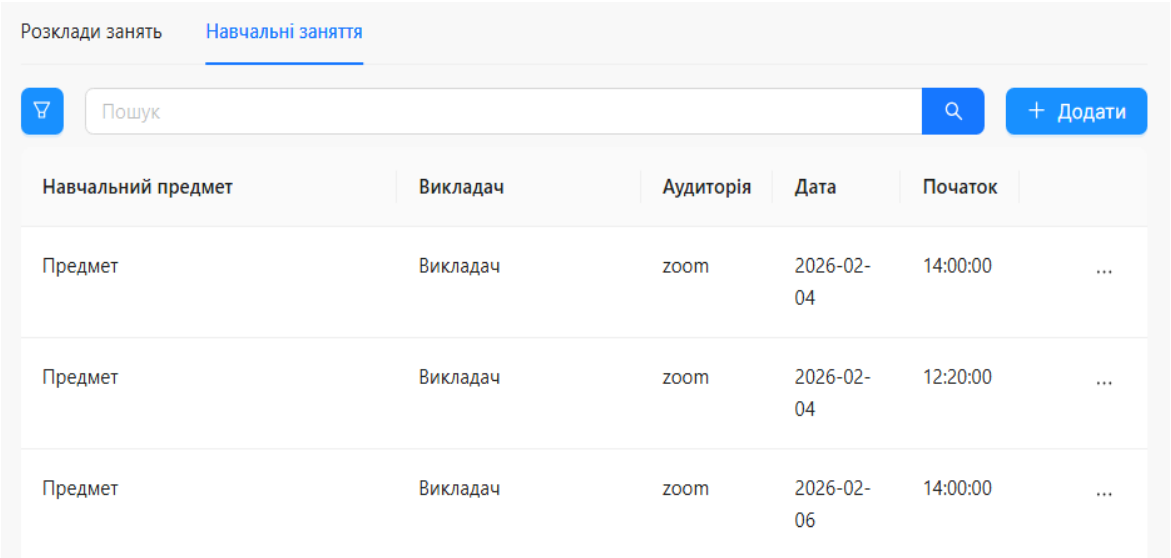

Рис. 1.2. Інтерфейс сторінки управління попередньою структурою розкладу KSU24, список навчальних занять

Список занять був реалізований як плаский перелік із колонками «Навчальний предмет”, «Викладач”, «Аудиторія”, «Дата” та «Початок”. Такий підхід до подання даних принципово суперечить ментальній моделі розкладу як двовимірної сітки. Відповідно до досліджень у сфері проєктування освітніх інтерфейсів [5] та принципів організації візуальної інформації [4], саме сіткова структура дозволяє диспетчеру одразу охопити повну картину тижня, тоді як плаский список вимагає послідовного перегляду кожного запису.

За результатами інтерв'ю було виявлено низку системних проблем інтерфейсу та користувацького досвіду у сценарії створення та редагування занять. Форма створення заняття містила поле вибору розкладу у вигляді селектора, що вимагало від диспетчера додаткового ментального кроку для явного визначення належності заняття до розкладу, при цьому також перевірки того, чи розклад вже створено, замість того, щоб ця прив'язка встановлювалась автоматично з контексту (Рис. 1.3).

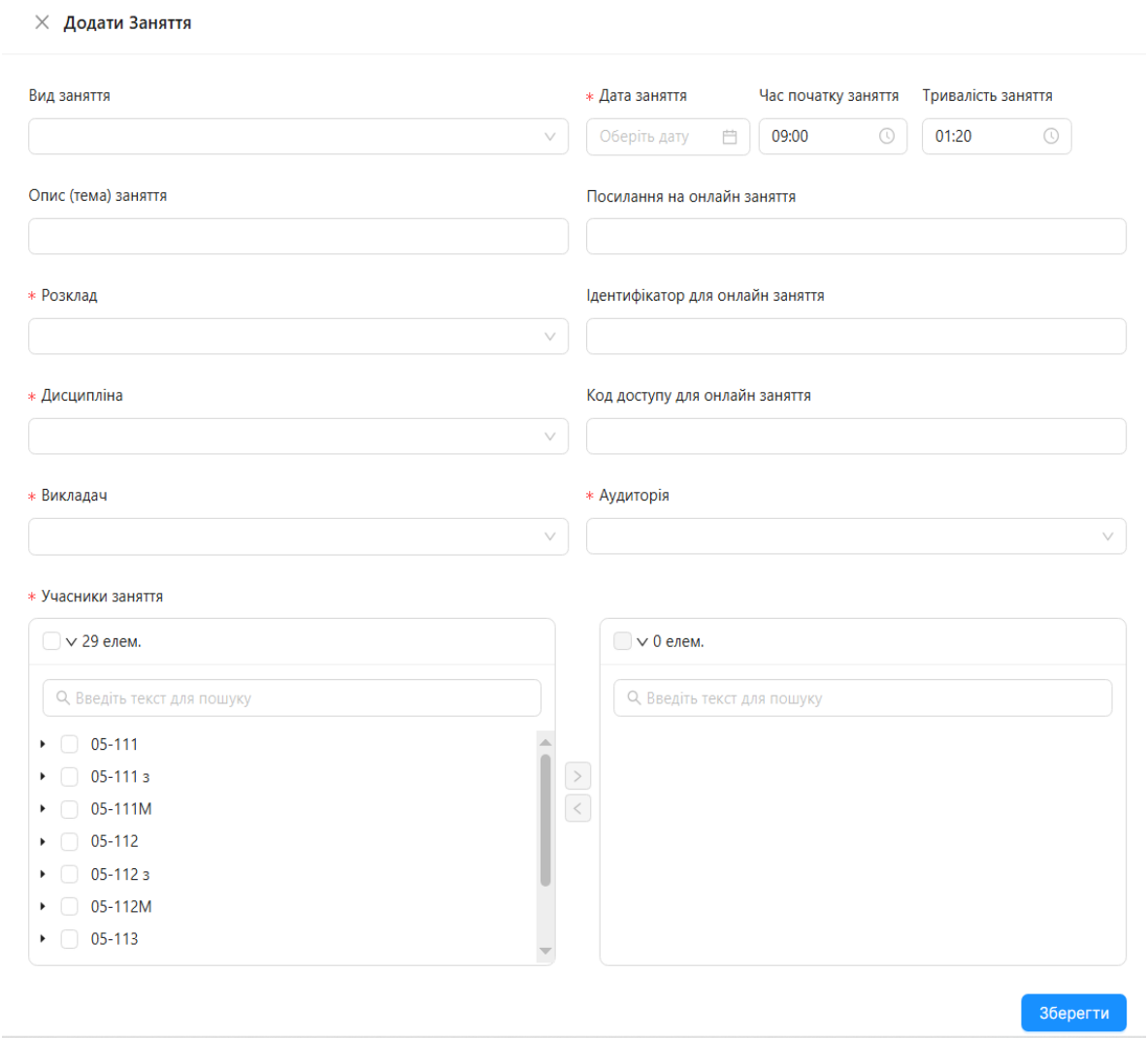

Рис. 1.3. Форма створення заняття старого модуля управління розкладом KSU24

Найбільш проблемним компонентом форми створення виявився елемент вибору учасників заняття. Він складався з двох полів, реалізованих як рядок пошуку з переліком результатів під ним. Введення запиту у будь-яке з цих полів одночасно проводило фільтрацію на обидва переліки, внаслідок чого вже обрані групи зникали зі списку доступних. Додавання або видалення учасника вимагало послідовності з двох дій: встановлення позначки на чекбоксі та натискання кнопки переміщення. При цьому обрані групи продовжували відображатись і в полі пошуку всіх груп. Відповідно до досліджень з оцінки користувацького досвіду [9], відсутність чіткого візуального

зворотного зв'язку та розмиття межі між різними станами об'єктів у робочому інтерфейсі («обрано” / «не обрано”) унеможливлює формування правильної ментальної моделі системи у диспетчера. Такий підхід порушує базові евристики користувацького досвіду та суттєво збільшує ймовірність помилкових дій.

Форма редагування заняття мала додаткові недоліки: окремі поля відображались некоректно, а поле зміни учасників для опублікованого заняття було відсутнє повністю. Відображення конфліктів розкладу було відсутнє, що переносило відповідальність за перевірку коректності розкладу виключно на людський фактор.

Такий стан інтерфейсу зумовив необхідність комплексної модернізації клієнтської частини модуля, що є предметом наступних розділів роботи.

1.4. Технічні обмеження при роботі з високо навантаженими освітніми модулями

Веб-застосунки з високою інформаційною щільністю, у тому числі системи складання розкладів, стикаються з низкою технічних викликів, що безпосередньо впливають на якість користувацького досвіду.

При роботі у веб-середовищі необхідно враховувати обсяги даних і характеристики рендерингу. Як зазначають А. Месбах та А. ван Дьорсен, архітектура односторінкових застосунків вимагає особливого підходу до управління станом і маніпуляцій з деревом DOM для уникнення затримок інтерфейсу [10]. Таблиця занять модуля розкладу є двовимірною структурою, де кількість DOM-елементів зростає пропорційно добутку кількості сутностей і кількості часових слотів. За наявності кількох десятків рядків і шести днів тижня з дев'ятьма номерами пар загальна кількість комірок у таблиці може перевищувати

500 елементів, кожен із яких є потенційним вузлом для повторного рендерингу.

Без використання спеціалізованих інструментів кожна зміна даних вимагає прямого втручання в DOM-дерево, що при частих оновленнях призводить до ресурсомістких процесів перерахунку геометрії (reflow) та перемальовування сторінки (repaint), блокуючи основний потік і спричиняючи візуальні затримки. У системі KSU24 для вирішення цієї проблеми використовується бібліотека React, що застосовує механізм узгодження для визначення мінімального набору змін у DOM під час оновлення стану. Емпіричні дослідження продуктивності сучасних фронтенд-фреймворків підтверджують, що надмірна кількість вузлів у віртуальному DOM без застосування механізмів мемоїзації призводить до критичного падіння швидкодії [11]. Для усунення зайвих обчислень React надає відповідні інструменти: React.memo для компонентів, useMemo для обчислюваних значень і useCallback для функцій-обробників.

Іншим вагомим фактором є мережеві затримки. При навігації між розкладами без кешування кожен перехід ініціює повторний мережевий запит, що за умови затримок відповіді API погіршує сприйняття інтерфейсу. Бібліотека TanStack Query вирішує цю проблему через збереження результатів попередніх запитів у клієнтській пам'яті та їх повторне використання до закінчення визначеного часу актуальності [12], забезпечуючи миттєве відображення даних при поверненні до вже завантаженого розкладу та гарантуючи їх актуальність через механізми інвалідації.

Окремим викликом є збереження балансу між інформаційною щільністю та читабельністю інтерфейсу. Багатовимірна матриця розкладу генерує значний обсяг візуальних даних: назви дисциплін, імена викладачів, типи занять, аудиторії. Надмірна концентрація цих

елементів без оптимізації їх структурного представлення погіршує ергономіку системи, підвищує когнітивне навантаження на користувача та впливає на технічні метрики досвіду, зокрема спричиняючи порушення візуальної стабільності макета (Cumulative Layout Shift). Відповідно, розробка комплексних табличних модулів вимагає ретельного проєктування UI-компонентів, що дозволяє інтегрувати необхідний обсяг інформації без перевантаження інтерфейсу та системних ресурсів.

РОЗДІЛ 2.
ПРОЄКТУВАННЯ ЛЮДИНО-МАШИННОЇ ВЗАЄМОДІЇ ДЛЯ МОДУЛЯ РОЗКЛАДУ

2.1. Розробка інформаційної архітектури та карти переходів (User Flow)

Інформаційна архітектура (ІА) модуля розкладу являє собою концептуальний каркас, що дозволяє користувачеві орієнтуватися у великих масивах даних через ефективні системи класифікації, маркування та навігації [13]. За визначенням Л. Розенфельда, якісна ІА забезпечує передбачуваність взаємодії, організовуючи контент відповідно до ментальних моделей користувачів, що є принциповою умовою для систем із високою інформаційною щільністю. User Flow описує послідовність кроків для досягнення конкретної мети. Згідно з моделлю Дж. Гарретта, на рівні структури проєктування цих шляхів зосереджується на дизайні взаємодії, де кожен перехід є логічним продовженням попередньої дії [14]. Ефективний User Flow передбачає мінімізацію кількості кроків до ключової інформації, що безпосередньо знижує когнітивне навантаження користувача [15].

Інформаційна архітектура модуля розкладу KSU24 побудована навколо трьох сценаріїв, що відповідають ролям стейкхолдерів: студента, викладача та диспетчера.

User Flow студента та викладача (Рис. 2.1) є прямим і спрямованим на споживання інформації. Після авторизації користувач потрапляє на головну сторінку, де персональний розклад підтягується автоматично без додаткових маніпуляцій: студенту відображаються заняття, в яких він є учасником, викладачу — заняття, де він є відповідальним. Така організація відповідає принципу миттєвого доступу до пріоритетних даних.

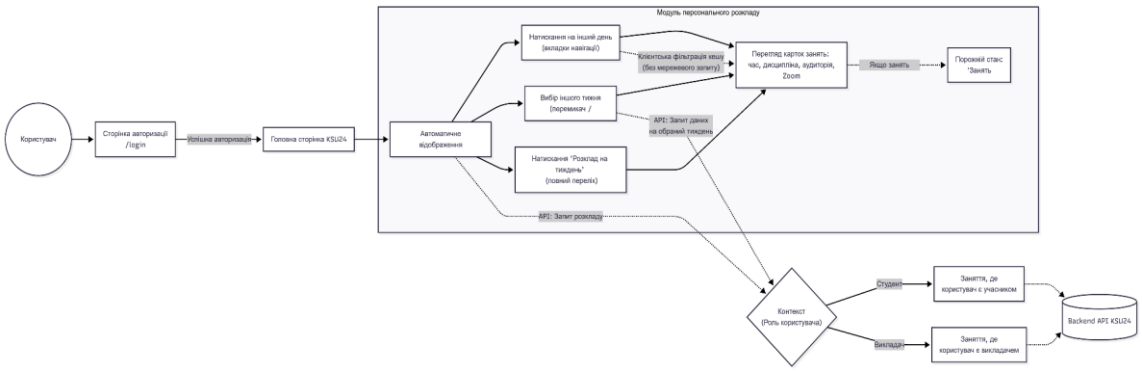

Рис.2.1. Карта переходів модуля персонального розкладу для студентів та викладачів

User Flow диспетчера є більш розгалуженим і передбачає повний цикл управління даними (Додаток А). Після авторизації диспетчер переходить до модуля через бічну панель: «Навчальний модуль” — «Розклад занять”. Доступ надається за наявності відповідної ролі та дозволів у системі, що забезпечує розмежування прав. Процес роботи з розкладом охоплює чотири послідовні етапи:

- Список наявних розкладів — перегляд опублікованих розкладів і чернеток, копіювання існуючого шаблону на інший тиждень, а також деактивація після підтвердження (soft delete), що зберігає запис у системі для можливого відновлення адміністратором.
- Створення або відкриття розкладу — новий розклад створюється через форму із селекторами факультету, назви та календарного тижня; у режимі чернетки дані зберігаються у підсистемі JARS як тимчасовий JSON-документ, не впливаючи на основну базу даних.
- Робота з таблицею — на сторінці обраного розкладу після визначення його даних, чи вибору існуючого, з'являється редактор занять із наявною перевіркою конфліктів у режимі реального часу.

- Публікація — на основі запису чернетки у підсистемі JARS бекенд формує відповідну сутність розкладу та пов'язаний перелік занять у основній базі даних, після чого розклад стає доступним для всіх учасників навчального процесу, а зміни в ньому призводять до змін у персональних розкладах.

2.2. Проєктування адаптивного дизайну: мобільна та десктоп версія

Адаптивний веб-дизайн — це підхід, що забезпечує коректне функціонування інтерфейсу на різних екранах через використання гнучких сіток та медіа запитів [16]. Реалізація модуля базується на принципах І. Маркотта, де інтерфейс не просто масштабується, а структурно адаптується відповідно до контексту пристрою. Принцип Mobile First, запропонований Л. Вробльовскі, передбачає проєктування з найменшого екрана з послідовним розширенням до більшого [17].

Проєктування частини модуля розкладу для відображення персонального списку занять користувача здійснювалось за принципом Mobile First, оскільки студенти та викладачі переважно використовують смартфони «на ходу", тоді як диспетчер працює виключно на великому екрані.

Персональний розклад реалізовано у форматі карток, розміщених в адаптивній сітці (Рис. 2.2). Для забезпечення коректного відображення макета на різних типах пристроїв застосовано стандартизовані брейкпойнти, що інтегровані в базову UI-систему проєкту. Такий підхід гарантує єдність візуального стилю та передбачувану поведінку компонентів при масштабуванні екрана [18]. Кожна картка містить: номер пари, часовий слот, назву дисципліни, ім'я викладача, аудиторію та, за наявності, посилання на відеоконференцію і прописаний диспетчером опис заняття. Навігація між днями тижня реалізована у

вигляді горизонтального рядка вкладок із лічильником занять. У разі відсутності пар на обраний день відображається «порожній стан", що усуває неоднозначність у трактуванні екрана.

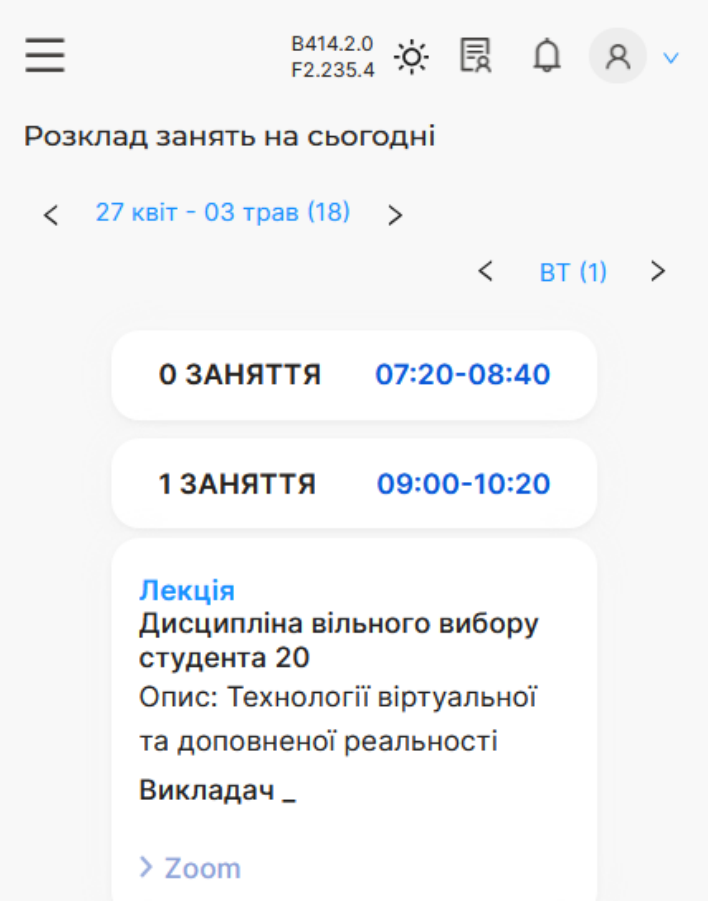

Рис. 2.2. Представлення частини персонального розкладу для мобільного інтерфейсу

Інтерфейс диспетчера орієнтований на десктоп. Таблиця-сітка з рядками сутностей і стовпцями днів є широкоформатним компонентом, коректне відображення якого на мобільному пристрої є технічно можливим та реалізованим, але менш ефективним. Це відповідає UX-практиці: складні операційні інтерфейси проєктуються для великих екранів, а інтерфейси споживання — для мобільних.

2.3. Клієнтська фільтрація та пошук у масивах даних розкладу на рівні користувацького інтерфейсу

Клієнтська фільтрація – це підхід, за якого обробка даних відбувається безпосередньо у браузері на основі вже завантаженого масиву. Вибір цього методу обґрунтований дослідженнями Я. Нільсена щодо часових лімітів реакції системи: оскільки майже миттєва відповідь інтерфейсу створює відчуття безпосереднього керування, обробка даних на стороні клієнта є критичною для продуктивності [15].

У модулі розкладу фільтрація реалізована без додаткових звернень до сервера після первинного завантаження. Це обумовлено специфікою роботи диспетчера: після відкриття розкладу весь масив занять присутній у пам'яті клієнта (через TanStack Query), і будь-яка фільтрація є локальною операцією. Це можливо та доцільно завдяки тому, що розклад розбивається на календарні тижні і користувач в результаті не оперує набагато більшим набором даних модуля чи семестра загалом. Таблиця (Рис. 2.3) підтримує такі режими перегляду:

- За освітнім компонентом (рядок представляє дисципліну).
- За викладачем (рядок представляє викладача).
- За групою (рядок представляє навчальну групу).
- За аудиторією (рядок представляє аудиторію).

Агрегація даних у потрібний формат відбувається локально. Додатково доступна фільтрація за формою навчання, семестром, типом заняття, роком навчальної програми. Доступна можливість перегляду занять лише обраного дня. Усі фільтри можуть бути застосовані одночасно. Також передбачено перемикач між пагінацією та повним списком рядків.

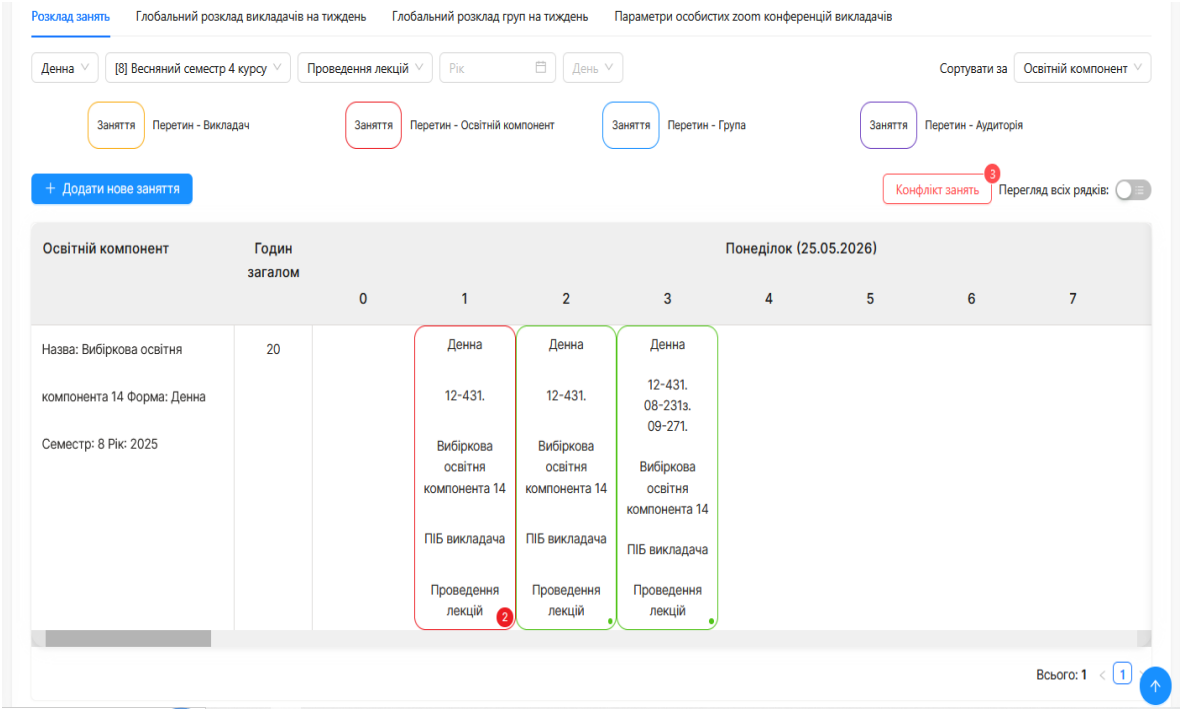

Рис. 2.3. Сіткове відображення розкладу для диспетчера за освітніми компонентами

Окремим сценарієм є виявлення конфліктів, що відповідає принципу «Direct Manipulation" Б. Шнайдермана: користувач отримує негайний зворотний зв'язок на свої дії [4]. Відповідна комірка таблиці, що має у собі конфлікт, змінює стиль границі відповідно до режиму перегляду (Рис. 2.4). Окрім перевірки конфліктів у середині обраного розкладу, модуль надає доступ до глобальної перевірки занять через перехресне зіставлення з іншими розкладами в системі.

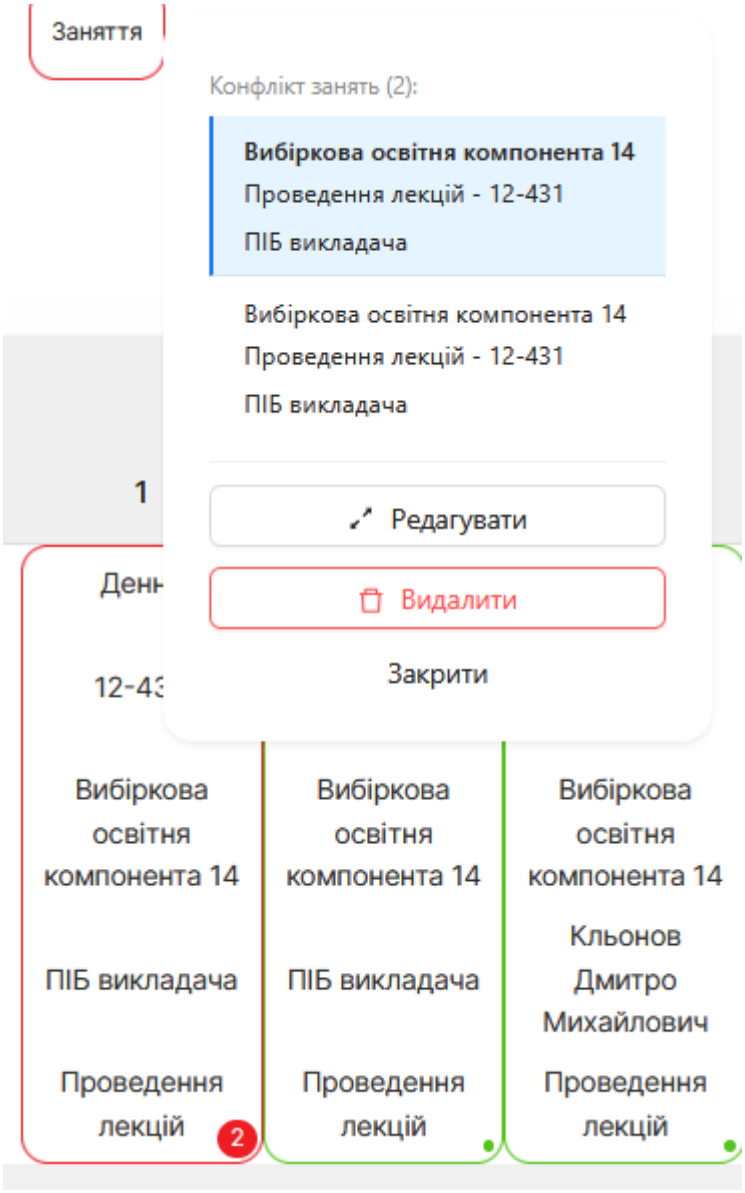

Рис. 2.4 Відображення конфлікту в комірці

2.4. Інклюзивність та доступність інтерфейсу згідно зі стандартами WCAG

Цифрова інклюзивність визначається як забезпечення рівного доступу до системи для всіх користувачів незалежно від їхніх обмежень [19]. Міжнародним стандартом у цій сфері є WCAG (Web Content Accessibility Guidelines). Актуальна версія 2.2 визначає рівень AA як де-факто галузевий стандарт, що вимагається законодавством, зокрема

Європейським актом про доступність (EAA) [20]. Реалізація модуля KSU24 орієнтована на відповідність рівню AA цього стандарту.

Оскільки KSU24 є односторінковим застосунком із постійно присутніми елементами навігації: бічною панеллю та заголовком, питання доступності інтерфейсу стосується всієї системи в цілому, а не лише окремих модулів. Відповідно, частина рішень щодо доступності вже була реалізована на рівні загальної архітектури клієнтської частини.

Усі інтерактивні елементи інтерфейсу доступні для фокусування та активації з клавіатури. Це кнопки перемикання тижня та дня розкладу, картки занять та наявні в них посилання на відеоконференції. Клавіатурна навігація забезпечує повноцінну роботу з модулем для користувачів із моторними порушеннями, а також підвищує загальну ергономічність для досвідчених користувачів, що надають перевагу клавіатурному введенню.

Компонентна бібліотека Ant Design забезпечує вбудовану ARIA-розмітку для всіх стандартних компонентів: кнопок, форм, чекбоксів, модальних вікон і випадаючих списків [21]. Це дозволяє програмам екранного доступу коректно інтерпретувати стан і призначення елементів без потреби у ручному додаванні атрибутів. Для користувацьких компонентів реалізовано відповідні aria-атрибути та role-анотації.

Для розширення можливостей персоналізації інтерфейсу в системі попередньо інтегровано бібліотеку accessibility [22], що надає користувачам інструменти налаштування відображення безпосередньо в інтерфейсі. Панель доступності відображається через фіксовану кнопку та містить такі параметри: збільшення та зменшення розміру тексту, регулювання міжлітерного та міжрядкового інтервалів, інвертування кольорів, перехід до відтінків сірого, підкреслення посилань, збільшення курсора, відображення рядка-підказки для читання, а також вимкнення

анімацій. Наявність цих налаштувань безпосередньо в інтерфейсі знижує поріг доступності для користувачів із різними потребами, не вимагаючи від них налаштування сторонніх засобів.

В усіх частинах модуля розкладу реалізовано коректне відображення порожніх станів. Відсутність даних завжди супроводжується відповідною ілюстрацією (Рис. 2.5) та лаконічним текстовим повідомленням, що усуває неоднозначність інтерфейсу та забезпечує чіткий зворотний зв'язок. Це є критично важливим для коректної роботи допоміжних технологій читання з екрана, оскільки порожній екран без текстового пояснення не може бути коректно інтерпретований програмою екранного доступу.

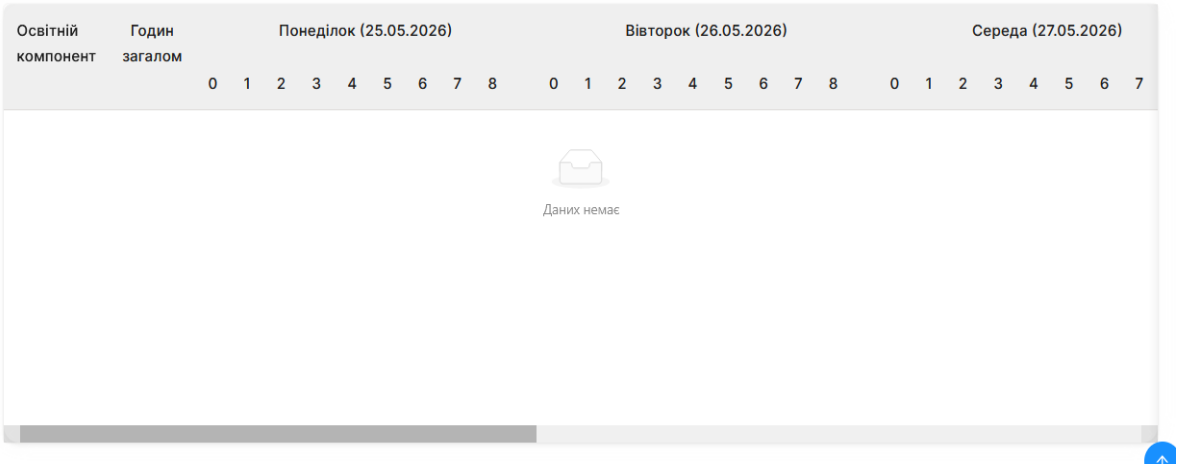

Рис. 2.5. Відображення порожнього стану

Важливим аспектом персоналізації та візуальної ергономіки стала підтримка світлої та темної тем інтерфейсу. Впровадження темної теми дозволяє знизити зорове навантаження при тривалій роботі з щільними масивами даних розкладу, особливо в умовах низького освітлення. Це розширює можливості доступності системи, дозволяючи інтерфейсу адаптуватися до потреб користувача, зокрема через підтримку системних налаштувань prefers-color-scheme.

РОЗДІЛ 3.
ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ОПТИМІЗОВАНОГО ІНТЕРФЕЙСУ

3.1. Вибір програмного стека для модернізації фронтенд-частини

Фронтенд-частина модуля розкладу реалізована у межах існуючої сервісної архітектури системи KSU24 [26]. Перелік використаних технологій із зазначенням версій наведено у таблиці 3.1.

Таблиця 3.1.

Використані технології

Технологія	Версія	Призначення
React	18.x	Побудова компонентного інтерфейсу
Ant Design	5.x	Бібліотека готових компонентів
TanStack Query	4.x	Управління серверним станом і кешування запитів
React Router	6.x	Клієнтська маршрутизація
Axios	1.x	HTTP-клієнт для взаємодії з API
Sass	1.x	CSS-препроцесор для стилізації компонентів
Day.js	1.x	Форматування дат
react-i18next	13.x	Інтернаціоналізація інтерфейсу

React використовується як основа інтерфейсу завдяки моделі на основі компонентів та декларативній парадигмі: замість ручної маніпуляції DOM розробник описує кінцевий вигляд інтерфейсу для конкретного стану даних, а бібліотека самостійно обчислює мінімальний набір змін через механізм Virtual DOM [23]. Управління станом і

побічними ефектами у компонентах забезпечується хуками useState, useEffect, useMemo та useCallback.

Ant Design обрано як UI-бібліотеку через широкий набір готових компонентів (форм, таблиць, модальних вікон, компонентів вибору дат) із вбудованою підтримкою доступності та гнучкою системою тематизації через CSS-змінні [24]. Це дозволяє зосередитись на логіці взаємодії, а не на низькорівневій реалізації стандартних UI-патернів.

TanStack Query відповідає за управління серверним станом: кожен запит ідентифікується ключем, результат зберігається у кеші, і повторний запит на ті самі дані не ініціює мережевий виклик до закінчення часу актуальності [12]. Крім того, бібліотека забезпечує інвалідацію кешу, фонове оновлення даних та відстеження стану запиту через прапорці isLoading, isError, isFetching.

React Router організовує клієнтську маршрутизацію модуля: список розкладів доступний за базовим маршрутом, а сторінка конкретного розкладу — за маршрутом з параметром ідентифікатора та позначенням режиму чернетки [25]. Наявність цього позначення у URL визначає, з яким джерелом даних взаємодіє сторінка — з API сутностей розкладу та занять або з JARS API чернетки.

Допоміжні інструменти вирішують вузькоспеціалізовані задачі. Axios забезпечує взаємодію з REST API через перехоплювачі для авторизації та централізованої обробки помилок. Sass використовується для написання модульних стилів специфічних компонентів розкладу. Day.js відповідає за безпечний парсинг та форматування календарних дат без мутацій вихідних об'єктів. react-i18next забезпечує багатомовність інтерфейсу через хук useTranslation.

3.2. Реалізація механізмів кешування та оптимізації рендерингу

У модулі розкладу засобами TanStack Query кешуються такі основні запити: список розкладів на сторінці їх відображення, розклад і список занять певного обраного розкладу на сторінці керування розкладом. Такий вибір обумовлений тим, що запити виконуються повторно при навігації між сторінками та при операціях зі змінами даних, і без кешування кожне таке повернення супроводжувалося б помітним очікуванням завантаження.

Кожен запит ідентифікується унікальним ключем. Для списку розкладів це ключ зі значеннями встановлених параметрів на пошук розкладів. Для занять це ключ, що включає ідентифікатор конкретного розкладу. Завдяки цьому при переході між різними розкладами дані кожного зберігаються незалежно, і повернення до вже переглянутого розкладу не потребує повторного мережевого запиту до закінчення часу актуальності кешу.

Інвалідація кешу відбувається у двох сценаріях. По-перше, після будь-якої мутації над розкладом (оновлення метаданих, формування копії на інший тиждень, деактивація) кеш списку розкладів інвалідується, що ініціює фонове оновлення даних при якому користувач бачить актуальний стан списку без ручного оновлення сторінки. По-друге, після кожної операції із заняттям (створення, оновлення, видалення) інвалідується кеш занять поточного розкладу, і таблиця автоматично відображає актуальний стан.

Варто зазначити, що у подібних сценаріях технічно можливим є використання оптимістичного оновлення, тобто підходу, за якого інтерфейс оновлюється миттєво на основі очікуваного результату операції ще до отримання відповіді від сервера, а у разі помилки відкочується до попереднього стану [28]. Однак для даного модуля в

системі цей підхід є надлишковим, адже швидкість відповіді серверу та відносно малий обсяг даних роблять повторний запит після інвалідації практично непомітним для користувача, особливо враховуючи проміжок закриття форми керування або видалення конкретного заняття. Через зазначені фактори інвалідація з фоновим оновленням є достатнім і простішим у підтримці рішенням. При цьому кожна дія над списком занять або метаданими розкладу супроводжується візуальним повідомленням для попередження виявлення сумнівів під час роботи диспетчера.

Таблиця занять є найбільш складним компонентом модуля з точки зору рендерингу: вона містить рядки сутностей (освітні компоненти, викладачі, групи або аудиторії) зі стовпцями для кожного часового слоту кожного дня тижня. При будь-якій зміні стану батьківського компонента, наприклад, відкритті спадного списку або перемиканні фільтра, без оптимізації відбувався б повний повторний рендер усіх комірок таблиці, що є зайвою операцією.

Для вирішення цієї проблеми реалізовано два механізми оптимізації. По-перше, у конфігурації стовпців таблиці (Ant Design Table) визначено функцію shouldCellUpdate, яка отримує та порівнює поточні та попередні властивості комірки і повертає хибне булеве значення, якщо вміст конкретного слоту не змінився, і таким чином комірка пропускає рендер, якщо її дані залишились ідентичними [27]. По-друге, агрегація масиву занять у структуру рядків таблиці, їх розподіл занять по сутностях, днях і номерах пар, реалізована з використанням хука useMemo. Це означає, що трудомістка операція побудови структури даних виконується лише при зміні вхідного масиву занять або активних фільтрів, а не при кожному рендері компонента [29]. Разом ці два механізми забезпечують те, що при взаємодії з

інтерфейсом перераховуються лише ті частини таблиці, дані яких фактично змінились.

3.3. Аналіз рендерингу та оцінка доцільності архітектурних рішень

Клієнтська агрегація та фільтрація. Рішення виконувати агрегацію занять у структуру таблиці і всю фільтрацію на стороні клієнта обґрунтоване характеристиками даних: розклад одного факультету на один тиждень містить значно меншу кількість занять ніж таку, що була б критичною для клієнтської обробки. Перевагою такого підходу є миттєва реакція при перемиканні режимів перегляду і фільтрів без мережевих затримок. Враховуючи обсяг даних, з яким диспетчеру доводиться працювати, завантаження всього масиву занять при відкритті розкладу не створює критичного навантаження на клієнтську частину системи. Для поточного масштабу даних це є виправданим рішенням, а кешування засобами TanStack Query забезпечує, що повторне відкриття вже завантаженого розкладу не очікує відповіді від мережевого запиту [12].

Розмежування режимів розкладу і чернетки на рівні маршрутизації. Виділення режиму чернетки як параметра URL дозволяє чітко розрізняти два джерела даних на рівні маршрутизації, а не умовної логіки всередині компонента. Це спрощує підтримку коду: кожен режим має власний набір API-викликів (JARS для чернетки, Schedule/Lesson API для опублікованого розкладу), і вибір між ними визначається один раз на рівні маршруту, а не розпорошується по компонентах [25]. Ще однією перевагою є те, що стан перегляду (конкретний розклад, режим чернетки) зберігається в URL, що дозволяє користувачам зберігати прямі посилання на потрібний контекст.

Інвалідація кешу замість оптимістичного оновлення. Як зазначено у підрозділі 3.2, для операцій із заняттями обрано підхід інвалідації з

повторним запитом замість оптимістичного оновлення. При достатній швидкості API різниця у відчутті відгуку є мінімальною, тоді як повернення візуального стану при помилці сервера в межах оптимістичного підходу здатне дезорієнтувати користувача і призвести до ситуації, коли диспетчер буде впевнений у збереженні заняття, якого фактично не існує в системі [28].

Сторінка керування розкладом реалізована через ієрархічну композицію компонентів. Кореневий компонент сторінки виконує роль «розумного" контейнера: централізовано ініціює запити до API через TanStack Query, визначає режим роботи (стандартний або чернетка), зводить отримані дані до єдиного інтерфейсу та розподіляє їх між дочірніми компонентами через властивості. Структурно сторінку поділено на дві незалежні частини.

Перша частина сторінки керування розкладом – це форма метаданих розкладу, реалізована у компоненті ScheduleData: вона відображає поля назви, опису, факультету та навчального тижня, а також елементи керування (збереження, копіювання на інший тиждень, деактивація). Форма за замовчуванням перебуває у заблокованому стані, і редагування стає доступним лише після явного переводу перемикача у режим редагування, що дозволяє запобігати випадковим операціям над сутністю розкладу при роботі із заняттями, які також вимагають підтвердження натиском на відповідну кнопку (Рис. 3.1).

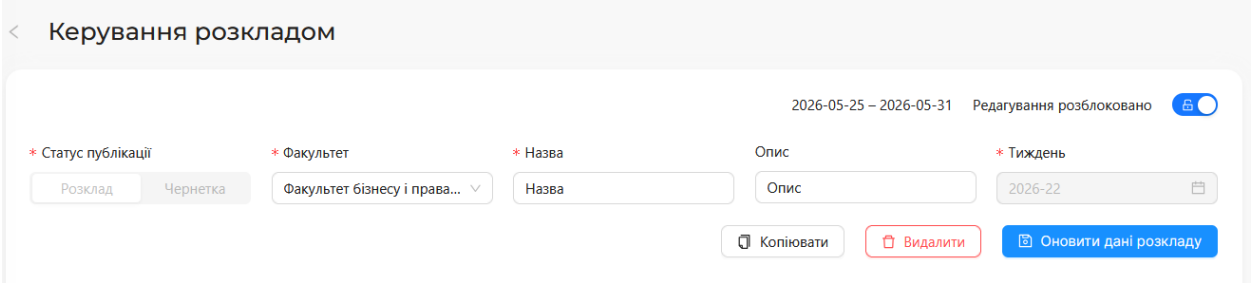

Рис. 3.1. Форма керування даними розкладу

Друга частина сторінки керування розкладом — панель із вкладками, що рендериться умовно: вона відображається лише після підтвердження коректності метаданих та завершення завантаження. До неї входять: основна вкладка розкладу (ScheduleTab), вкладки перевірки занять та, відповідно, конфліктів у глобальному контексті, окремо для різних конфігурацій таблиці (ScheduleConflictsTab), а також вкладка керування Zoom конференцій викладачів, збереження даних для яких реалізовано через JARS. Центральною з-поміж них є основна вкладка розкладу, яка у свою чергу компонує фільтри перегляду (ScheduleFilters), таблицю занять (ScheduleTable) із динамічно згенерованими стовпцями тижня, в яких автоматично відображаються дані занять та візуальна індикація конфліктів, форму для створення й редагування занять (ScheduleLessonDrawer) (Додаток Б) і модальне вікно відображення конфліктів (ConflictsModal) (Рис. 3.2). Агрегація масиву занять у рядки таблиці та обчислення конфліктів винесені в окремий хук useScheduleData, що забезпечує відокремлення логіки обробки даних від логіки відображення. Комірки розкладу для таблиці відмальовуються у скрипті (columns), де кожна комірка реалізована компонентом (ScheduleCell). Пуста комірка дозволяє при натисканні відкрити форму заняття. Комірка з наявним заняттям дозволяє видалити заняття, відкрити форму редагування для заняття, або одного з обраних занять за умови наявності конфлікту.

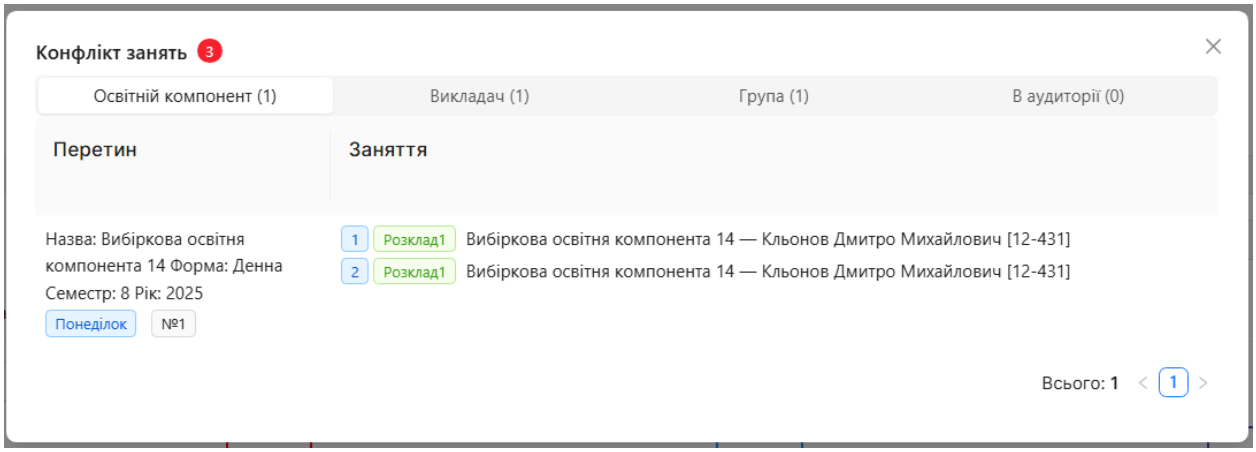

Рис. 3.2. Модальне вікно відображення всіх конфліктів у таблиці

У формі диспетчер заповнює поля заняття. При відкритті форми з пустої комірки автоматично визначається день та номер пари (Додаток Б). При виборі існуючого заняття форма автоматично має заповнені дані та надає можливість як оновити, так і створити нове заняття за вибраними параметрами. Форму можна закрити та відкрити з попередніми даними на кнопку додавання заняття над таблицею занять, також дані форми заняття можна локально скопіювати та вставити у будь-якому доступному розкладі.

3.4. Тестування та оцінка користувацького досвіду

Оцінка результатів модернізації інтерфейсу проводилась у форматі порівняльного тестування користувацького досвіду за протоколом «мислення вголос». Відповідно до методології Дж. Рубіна і Д. Чіснелла [30], цей протокол передбачає, що учасник тестування озвучує свої думки і наміри в процесі виконання завдань, що дозволяє виявляти приховані труднощі взаємодії, які не проявляються у кількісних метриках. Диспетчери навчального процесу, що брали участь в початковому інтерв'ю, мали можливість порівняти виконання типових робочих сценаріїв у попередній та оновленій версіях модуля. Фокус

тестування був на якісній оцінці зниження операційного навантаження при виконанні конкретних задач, а не на статистичних вимірюваннях.

Найбільш показовим сценарієм для порівняння є створення нового заняття, оскільки саме ця операція була визначена як найбільш проблемна у попередній версії.

Попередня версія вимагала від диспетчера виконання наступної послідовності дій. Спочатку необхідно було перейти на вкладку «Навчальні заняття» та натиснути кнопку «Додати». Після цього у відкритій формі користувач мав явно обрати розклад із селектора, а також заповнити поля дисципліни, викладача, аудиторії, дати та часу. Далі, щоб додати учасників, потрібно було ввести запит у поле пошуку, позначити чекбоксом потрібну групу і натиснути кнопку переміщення для її додавання до списку обраних. Наступним кроком диспетчер перевіряв коректність відображення обраних елементів (з огляду на проблему одночасної фільтрації обох полів, що була описана у розділі 1.3) і завершував процес збереженням заняття. Варто зазначити, що попередня форма редагування не містила поля учасників, що унеможливлювало їх зміну без повторного створення заняття. Натомість форма у новому стандарті дозволяє створювати нові заняття, вільно редагуючи контекст існуючих, на противагу створенню повної копії зі зміщенням дати.

Оновлена версія скорочує та спрощує цю послідовність. Спочатку диспетчер перебуває на сторінці конкретного розкладу та натискає на вільну комірку у потрібному часовому слоті. Після цього відкривається форма, де прив'язка до розкладу, день тижня та номер заняття вже визначені контекстом. Далі диспетчер заповнює поля освітньої компоненти, викладача і типу заняття. Потім у секції учасників за допомогою поля пошуку він знаходить групу та додає її цілком кнопкою «+» (або розгортає її і обирає конкретних студентів через чекбокси),

після чого зберігає заняття. Обрані групи відображаються в окремій секції «Обрані групи» і не змішуються з результатами пошуку, що усуває основну проблему попередньої версії. Та сама форма використовується і для редагування, включаючи зміну учасників.

Скорочення кількості обов'язкових кроків і усунення неоднозначності у компоненті вибору учасників відповідає принципу мінімізації когнітивного навантаження: чим менше рішень користувач змушений приймати для виконання задачі, тим нижча ймовірність помилки і тим вищий рівень задоволеності [5]. Принципово важливим є те, що перехід від схеми «відтворення з пам'яті» (явне введення розкладу, дати і часу) до схеми «підтвердження запропонованих даних» (контекстне відкриття форми з попередньо заповненими полями) відповідає правилу скорочення завантаження короткострокової пам'яті [4].

Попередня версія відображала заняття як лінійний хронологічний список із колонками «Предмет», «Викладач», «Аудиторія», «Дата», «Початок». Для відповіді на типовий запит пошуку заняття у певний день тижня на певній парі диспетчер мав послідовно переглядати записи списку, фільтруючи їх ментально за датою та часом.

Оновлена версія відображає ті самі дані у двовимірній сітці, де стовпці відображають дні тижня з підстовпцями номерів пар, а рядки відображають сутності обраного режиму перегляду. Відповідь на той самий запит зводиться до одного погляду на перетин потрібного стовпця і рядка.

За відгуками диспетчерів, що брали участь у тестуванні, оновлений інтерфейс отримав позитивну оцінку за такими аспектами: відсутність необхідності явно вказувати розклад при створенні заняття завдяки наявності відповідного контексту, коректна та зрозуміла робота з учасниками у формі, наявність форми редагування з повним набором

полів включно з учасниками, а також сіткове відображення, що дозволяє одразу охопити структуру тижня. Станом на момент написання роботи диспетчери навчального процесу вже використовують оновлений модуль у штатному режимі, що підтверджує практичну придатність реалізованих рішень.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи проведено комплексну модернізацію інтерфейсу модуля «Цифровий розклад» інформаційної системи KSU24. За результатами дослідження сформульовано такі висновки.

Аудит наявного інтерфейсу у форматі інтерв'ю з диспетчерами всіх факультетів університету підтвердив, що відсутність сіткового відображення, некоректна робота форми занять і брак візуалізації конфліктів унеможливлювали використання системи як основного робочого інструменту, диспетчери були змушені працювати в зовнішніх табличних редакторах.

Розроблена інформаційна архітектура розмежовує сценарії трьох ролей: персональний розклад студентів і викладачів реалізовано за принципом Mobile First, інтерфейс диспетчера орієнтовано на повний цикл управління даними у десктопному середовищі.

Програмна реалізація усунула виявлені недоліки: двовимірна таблиця-сітка з клієнтською фільтрацією забезпечує миттєву реакцію при перемиканні контекстів; контекстне відкриття форми заняття з комірки таблиці замінило ручне введення параметрів; новий компонент вибору учасників усунув неоднозначність попередньої версії; конфлікти візуалізуються безпосередньо в таблиці з підтримкою глобальної перевірки.

Оптимізацію рендерингу забезпечено через shouldCellUpdate у конфігурації таблиці та мемоізацію агрегації даних засобами useMemo; кешування серверного стану через TanStack Query усунуло повторні мережеві запити при навігації.

Відповідність WCAG рівня AA досягнута через клавіатурну навігацію, ARIA-розмітку компонентної бібліотеки Ant Design та інтегровану панель доступності.

Мету роботи досягнуто. Практичну придатність реалізованих рішень підтверджує перехід диспетчерів навчального процесу на використання оновленого модуля у штатному режимі роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Schaerf A. A survey of automated timetabling / A. Schaerf // Artificial Intelligence Review. — 1999. — Vol. 13, № 2. — P. 87–127.
2. Kristiansen S. A comprehensive study of educational timetabling — A survey / S. Kristiansen, T. R. Stidsen // European Journal of Operational Research. — 2013. — Vol. 231, № 1. — P. 1–14.
3. Sweller J. Cognitive Load Theory / J. Sweller, P. Ayres, S. Kalyuga. — New York : Springer Science & Business Media, 2011. — 274 p.
4. Designing the User Interface: Strategies for Effective Human-Computer Interaction / [B. Shneiderman, C. Plaisant, M. Cohen et al.]. — 6th ed. — Pearson, 2016. — 624 p.
5. Ordorica L. UI/UX Case Study for a Student Portal Schedule Feature / L. Ordorica // Medium. — 2022. — Електронний ресурс. Режим доступу: https://medium.com/ux-station/ui-ux-case-study-for-a-student-portal-schedule-feature-30a8c28e9a8b (дата звернення: 12.04.2026).
6. Попов С. А. Проєктування та розроблення сервісної архітектури управління бізнес-процесами університету. Сервіс «Деканат» : кваліфікаційна робота магістра / С. А. Попов. — Херсон : ХДУ, 2020. — 44 с.
7. Fowler M. Patterns of Enterprise Application Architecture / M. Fowler. — Boston : Addison-Wesley, 2002. — 560 p.
8. Лемещук О. В. Тимчасові та постійні рішення інтеграції сторонніх сервісів у віртуальне освітнє середовище KSU24 / О. В. Лемещук, Д. Сенчишен // Вимірювальна та обчислювальна техніка в технологічних процесах. — 2025. — № 81. — С. 235–243. — DOI: 10.31891/2219-9365-2025-81-29.
9. Bargas-Avila J. A. Old wine in new bottles or novel challenges: a critical analysis of empirical studies of user experience / J. A. Bargas-

Avila, K. Hornbæk // Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. — 2011. — P. 2689–2698.

10. Mesbah A. Migrating multi-page web applications to single-page AJAX interfaces / A. Mesbah, A. Van Deursen // 11th European Conference on Software Maintenance and Reengineering (CSMR'07). — 2007. — P. 181–190.
11. Gajera K. Performance Evaluation of Angular and React JavaScript Frameworks / K. Gajera, N. Vaghasiya, K. Kotecha // International Conference on Artificial Intelligence and Smart Systems (ICAIS). — 2021. — P. 1163–1168.
12. TanStack Query. Overview. — Електронний ресурс. Режим доступу: https://tanstack.com/query/latest/docs/framework/react/overview (дата звернення: 12.04.2026).
13. Rosenfeld L. Information Architecture: For the Web and Beyond / L. Rosenfeld, P. Morville, J. Arango. — O'Reilly Media, 2015. — 486 p.
14. Garrett J. J. The Elements of User Experience: User-Centered Design for the Web and Beyond / J. J. Garrett. — New Riders, 2010. — 192 p.
15. Nielsen J. Usability Engineering / J. Nielsen. — Morgan Kaufmann, 1993. — 362 p.
16. Marcotte E. Responsive Web Design / E. Marcotte. — A Book Apart, 2011. — 182 p.
17. Wroblewski L. Mobile First / L. Wroblewski. — A Book Apart, 2011. — 123 p.
18. MDN Web Docs. CSS grid layout. — Електронний ресурс. Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS/Guides/Grid_layout (дата звернення: 12.04.2026).
19. W3C Web Accessibility Initiative. WCAG 2 Overview. — Електронний ресурс. Режим доступу:

https://www.w3.org/WAI/standards-guidelines/wcag/ (дата звернення: 12.04.2026).

20. W3C. Web Content Accessibility Guidelines (WCAG) 2.2. — 2023. — Електронний ресурс. Режим доступу: https://www.w3.org/TR/WCAG22/ (дата звернення: 12.04.2026).
21. Ant Design. Accessibility. — Електронний ресурс. Режим доступу: https://ant.design/docs/react/accessibility (дата звернення: 12.04.2026).
22. ranbuch. accessibility (npm package, v6.0.3). — Електронний ресурс. Режим доступу: https://www.npmjs.com/package/accessibility (дата звернення: 12.04.2026).
23. React. Describing the UI. — Електронний ресурс. Режим доступу: https://react.dev/learn/describing-the-ui (дата звернення: 12.04.2026).
24. Ant Design. Introduction. — Електронний ресурс. Режим доступу: https://ant.design/docs/react/introduce (дата звернення: 12.04.2026).
25. React Router. Main Concepts. — Електронний ресурс. Режим доступу: https://reactrouter.com/start/concepts (дата звернення: 12.04.2026).
26. Лукінов Г. Г. Проєктування та розроблення сервісної архітектури управління бізнес-процесами університету. Дизайн інтерфейсу користувача : кваліфікаційна робота магістра. — Херсон : ХДУ, 2021. — 47 с.
27. Ant Design. Table. — Електронний ресурс. Режим доступу: https://ant.design/components/table (дата звернення: 12.04.2026).
28. TanStack Query. Optimistic Updates. — Електронний ресурс. Режим доступу: https://tanstack.com/query/latest/docs/framework/react/guides/optimistic-updates (дата звернення: 12.04.2026).

29. React. useMemo. — Електронний ресурс. Режим доступу: https://react.dev/reference/react/useMemo (дата звернення: 12.04.2026).
30. Rubin J. Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests / J. Rubin, D. Chisnell. — 2nd ed. — Wiley, 2008. — 384 p.

ДОДАТКИ

Додаток А

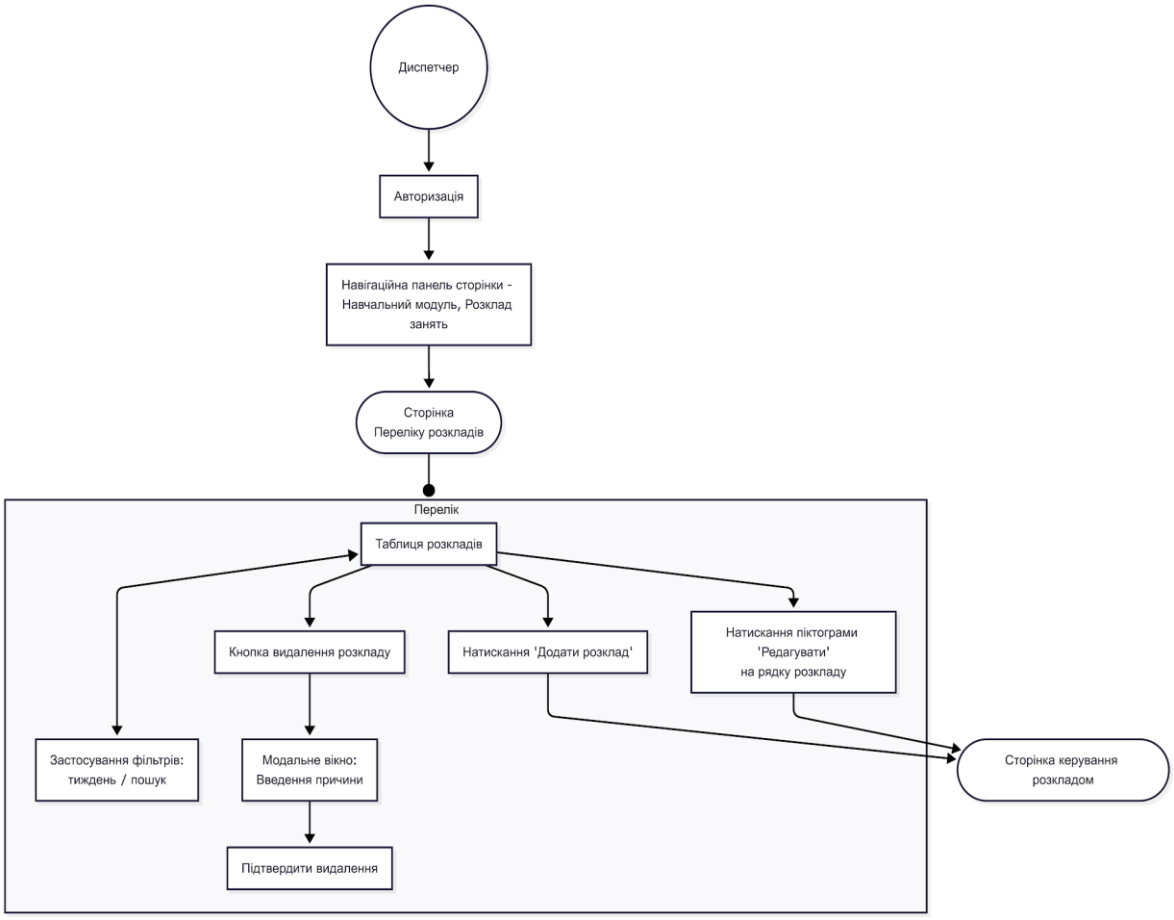

Рис. А.1. Діаграма переліку розкладів

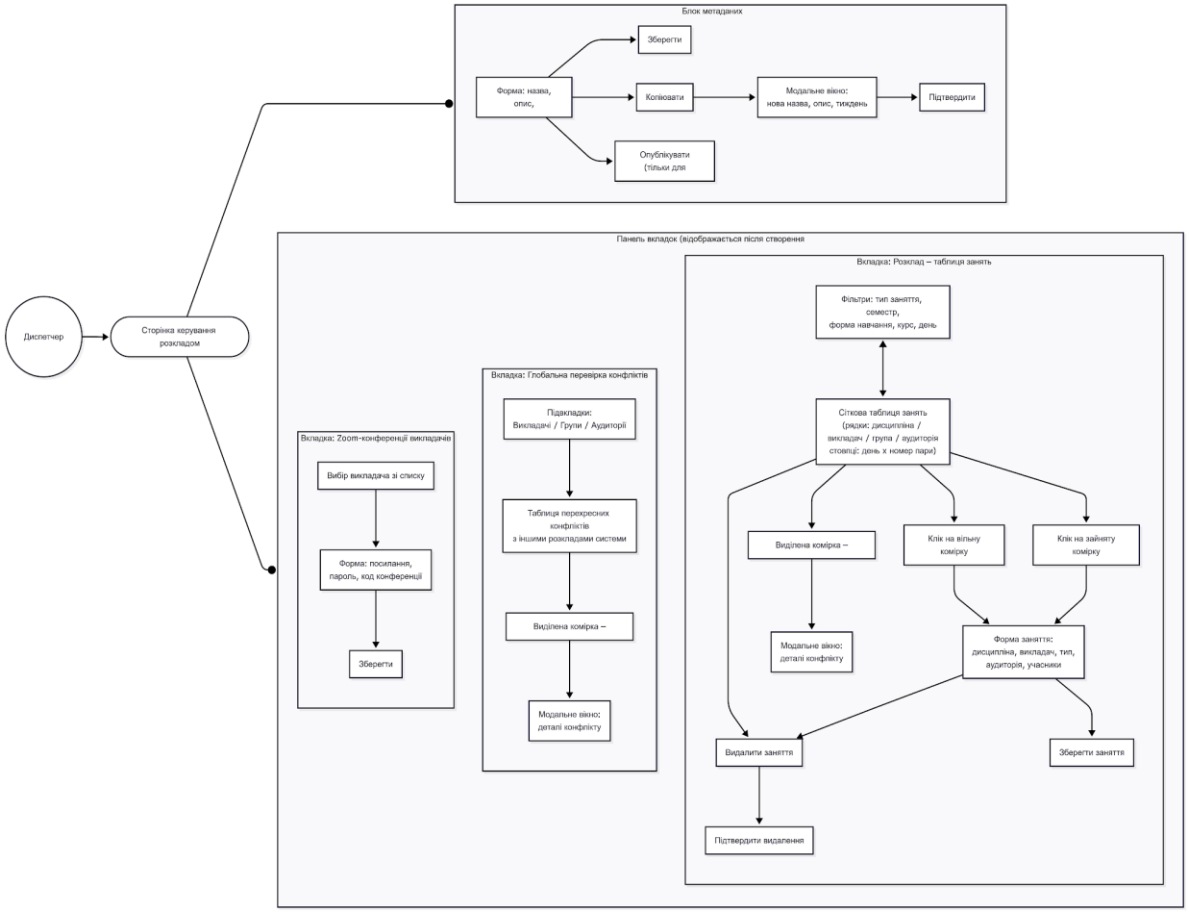

Рис. А.2. Діаграма керування розкладом

Додаток Б

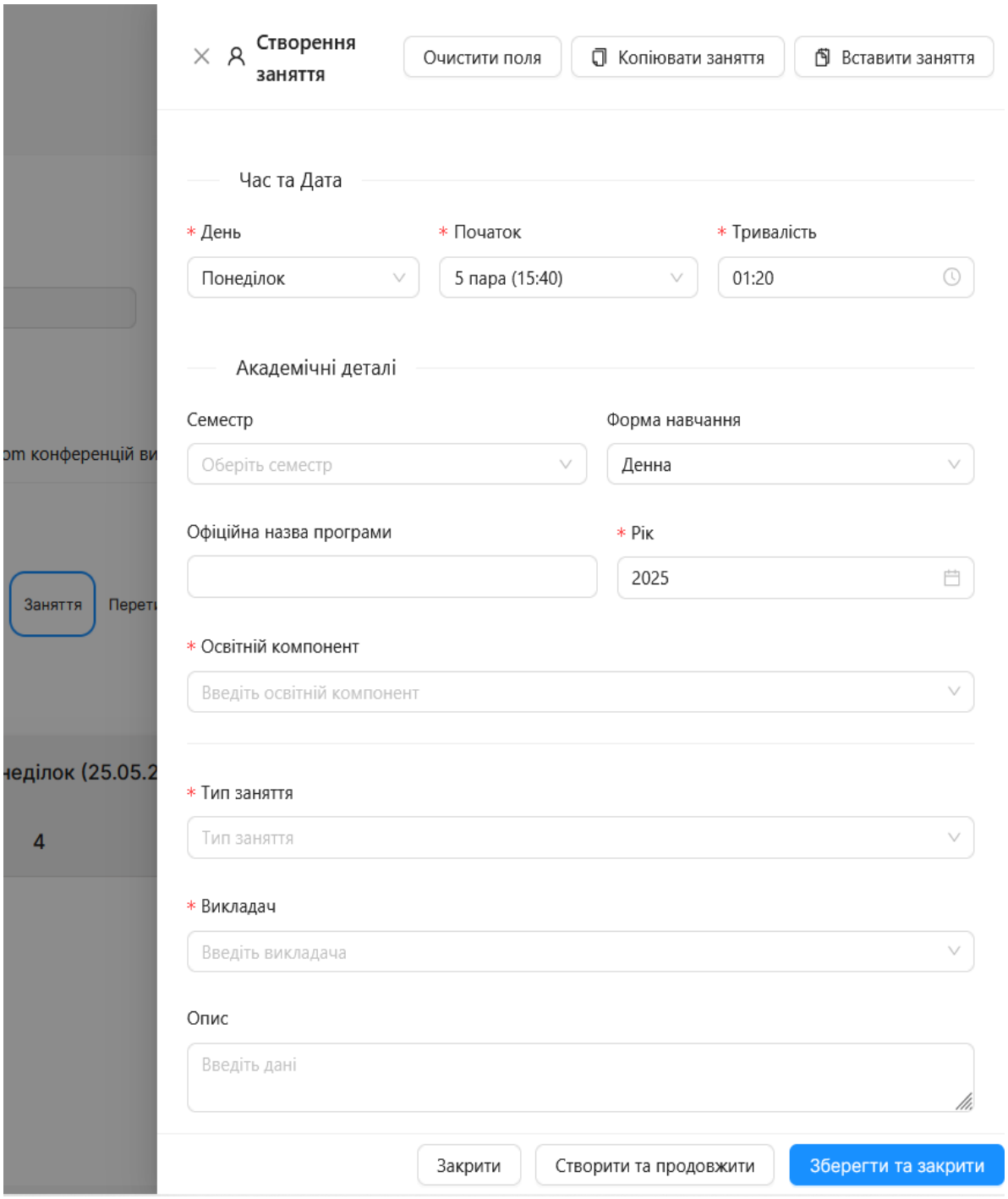

Рис. Б.1. Форма створення заняття – основні параметри

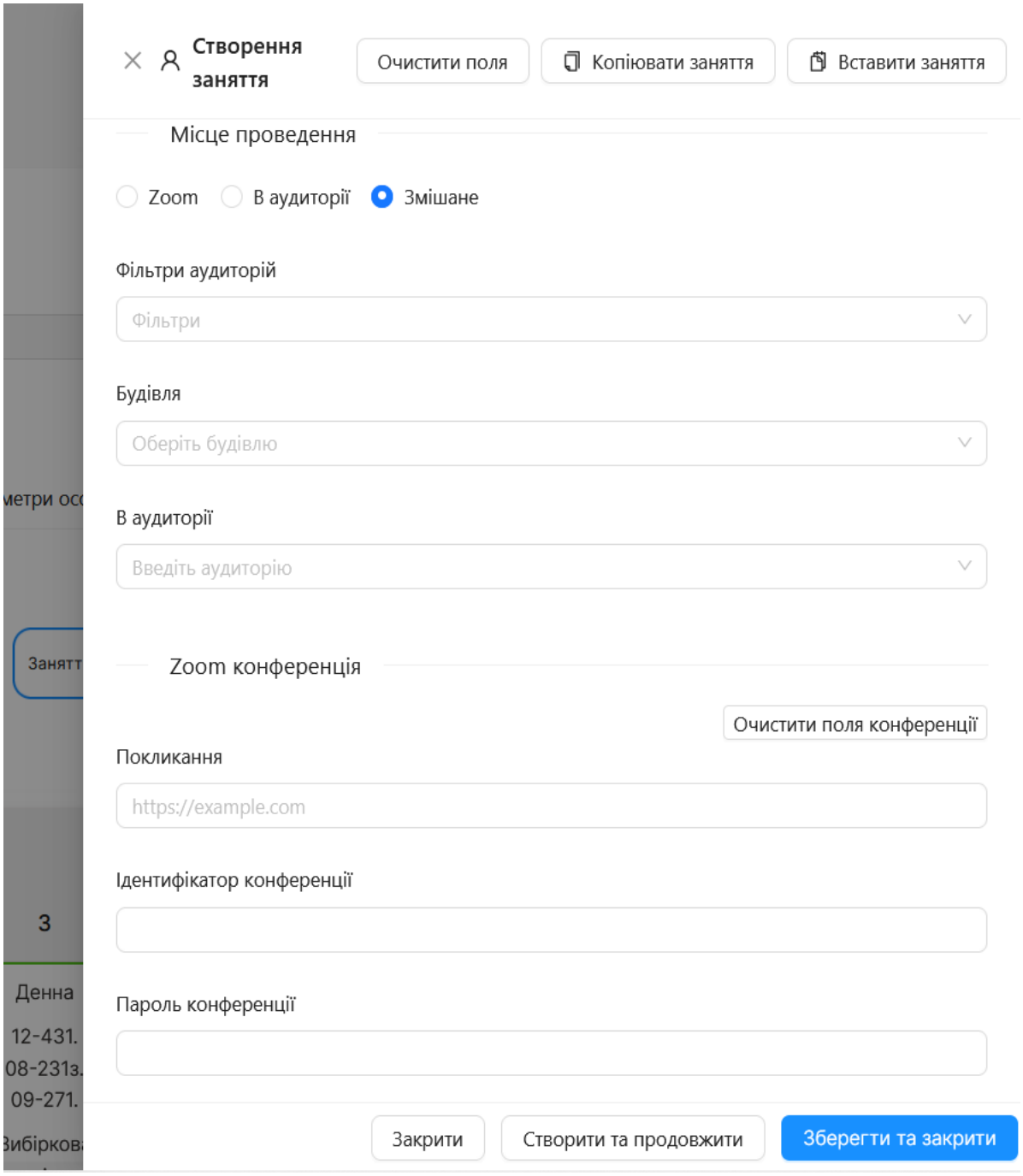

Рис. Б.2. Форма створення заняття – форма та місце проведення

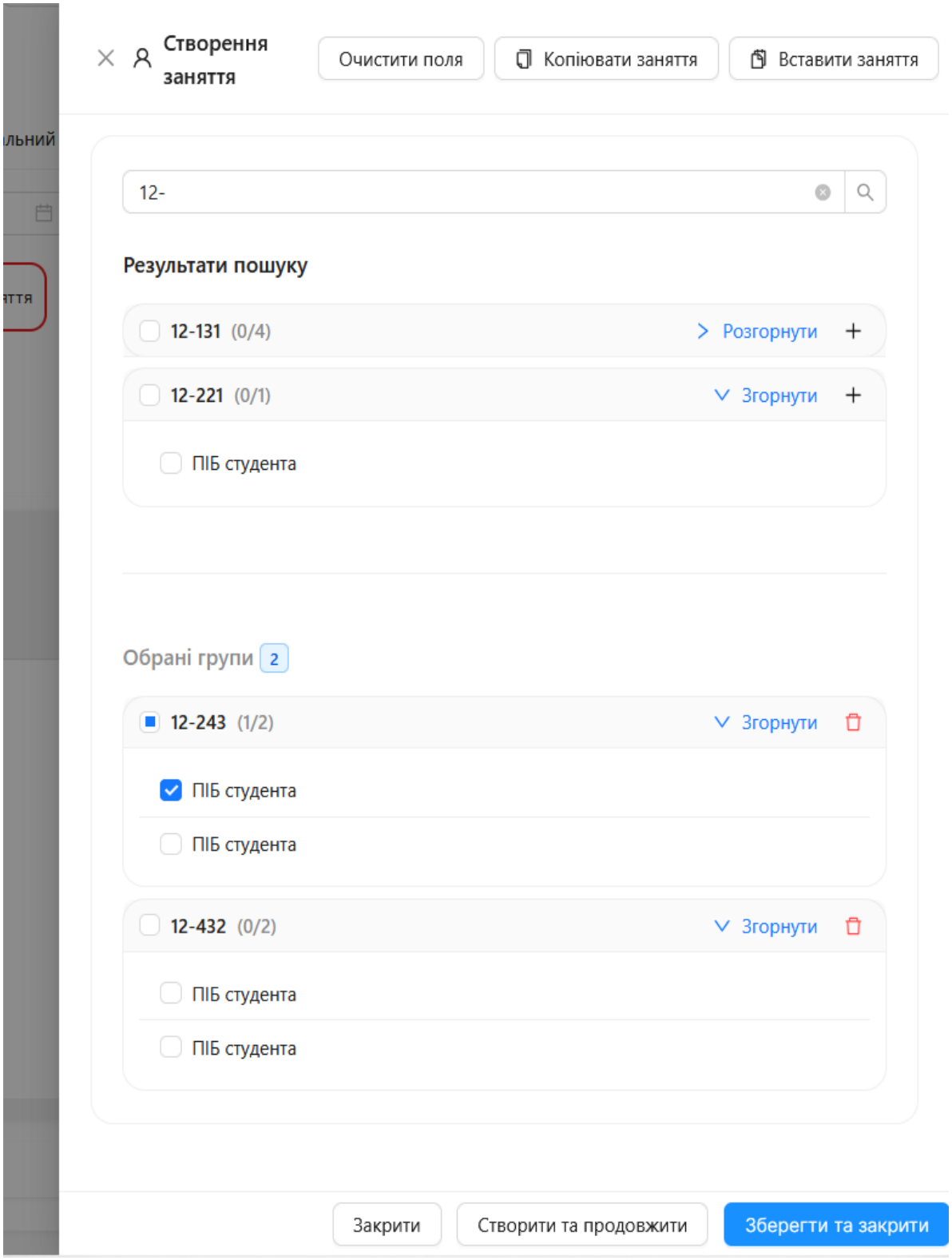

Рис. Б.3. Форма створення заняття – вибір учасників