**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК, ФІЗИКИ ТА МАТЕМАТИКИ**
**КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ**

# ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ СЕРВІСНОЇ АРХІТЕКТУРИ УПРАВЛІННЯ БІЗНЕС-ПРОЦЕСАМИ УНІВЕРСИТЕТУ. СИСТЕМА ОПОВІЩЕНЬ ТА ПОДІЙ

**Кваліфікаційна робота (проєкт)**
на здобуття ступеня вищої освіти «магістр»

Виконав: студент 2 курсу 241М групи
Спеціальності: 121 Інженерія програмного забезпечення
Освітньо-професійної програми:
Інженерія програмного забезпечення
Микуленко Костянтин Ігорович
Керівник: доктор педагогічних наук, професор
Співаковський Олександр Володимирович
Рецензент:

Херсон – 2022

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

- HTTP – Hypertext Transfer Protocol – протокол передачі даних, що належить до протоколів прикладного рівня моделі OSI
- HTTPS – Hypertext Transfer Protocol Secure – розширення протоколу HTTP для безпечного зв’язку в комп’ютерних мережах
- API – Application Programming Interface – інтерфейс для взаємодії між програмними компонентами
- HTML – Hypertext Markup Language – мова розмітки документів для відображення у веббраузері
- JSON – JavaScript Object Notation – текстовий формат опису об’єктів для обміну даними між програмними компонентами
- REST – Representational State Transfer – архітектурний стиль програмування для забезпечення єдиного інтерфейсу між окремими програмними компонентами в архітектурі «клієнт-сервер»
- JWT – JSON Web Token – інтернет-стандарт на основі JSON для ідентифікації користувачів
- SQL – Structured Query Language – мова запитів у реляційних базах даних
- UI – User Interface – користувацький інтерфейс
- UX – User Experience – досвід користування – користь від взаємодії з продуктом
- UUID – Universally Unique Identifier – стандарт ідентифікації об’єктів
- SEO – Search Engine Optimization – пошукова оптимізація сайту
- СКБД – система керування базами даних

# ВСТУП

**Актуальність дослідження** полягає в необхідності забезпечення системи управління бізнес-процесами університету можливістю формування й доставки повідомлень про важливі події та оновлення в навчальному процесі на єдиному вебпорталі.

**Метою цієї роботи** є створення системи, яка допомагатиме студентам контролювати свій навчальний процес через сповіщення, в рамках сервісної архітектури управління бізнес-процесами університету.

**Завдання дослідження**:

- ознайомитися з інформаційною інфраструктурою ХДУ;
- провести аналіз існуючого функціоналу системи управління бізнес-процесами університету;
- проаналізувати вимоги до інтерфейсу користувача для системи оповіщень;
- проаналізувати архітектуру системи оповіщень;
- провести аналіз технологій та програмних засобів для розроблення вебдодатку та інтерфейсу взаємодії з користувачем;
- розроблення сервісу сповіщень в інтерфейсі взаємодії з користувачем вебдодатку системи управління бізнес-процесами університету ХДУ24;
- аналіз систем розгортання й тестування вебдодатку.

**Об’єктом дослідження** цієї роботи є архітектура та технології інтерфейсу користувача системи оповіщень та подій вебдодатку ХДУ24.

**Предметом дослідження** є сервісна архітектура управління бізнес-процесами університету. Система оповіщень та подій.

**Практичне значення** роботи полягає в можливості надати адміністрації університету та викладачам інструмент для постійної комунікації зі студентами.

**Структура роботи**. Дипломна робота складається зі вступу, трьох розділів, висновків і списку використаних джерел.

# РОЗДІЛ 1

# АНАЛІЗ ТЕХНОЛОГІЙ ТА РЕАЛІЗАЦІЙ СИСТЕМИ ОПОВІЩЕНЬ ТА ПОДІЙ В УНІВЕРСИТЕТАХ

## 1.1. Роль системи сповіщень в освітньому процесі

Автоматизована система оповіщень в останні роки стала невід'ємною частиною системи управління бізнес-процесами університету. Вона призначена для своєчасного сповіщення всіх зацікавлених сторін про важливі події, надзвичайні ситуації, оновлення та зміни всіх важливих процесів у житті університету.

У зв'язку з усе більш масовим переходом до форм дистанційного навчання значно зростає необхідність автоматизації систем спілкування між адміністрацією та студентами. Система управління сповіщеннями призначена для надсилання персоналізованих текстових повідомлень великій кількості користувачів за короткий проміжок часу та для контролю доставки цих сповіщень.

Надійне та безперебійне функціонування системи сповіщень слугує для комунікації між адміністрацією університету, персоналом та студентами. Студенти можуть отримувати оновлення в розкладі занять, сповіщення про заліки та іспити та інші події.

Важливо врахувати те, що система управління сповіщеннями працює з великою кількістю людей, з різними ролями й обов'язками, комунікаційними привілеями та різними потребами. Необхідно з'ясувати, які потреби необхідні кожній категорії споживачів, і застосувати такі системи доставки сповіщень, аби бути впевненим у тому, що важлива інформація буде доставлена й прочитана адресатом.

Також необхідно побудувати систему підписників на сповіщення для того, щоб інформація була доставлена лише тим групам користувачів, яких вона стосується.

Для доставки сповіщень існують різні канали зв'язку:

1. автоматичні особисті або масові сповіщення через вебдодаток;
2. електронна пошта;
3. push-сповіщення в мобільному додатку;
4. використання соціальних мереж або месенджерів;
5. SMS-повідомлення.

Найбільш оперативним способом доставки повідомлень є SMS-повідомлення та push-повідомлення в мобільному додатку. За аналізом відкриття такого повідомлення в період декількох хвилин після відправлення сягає більше 90%, що є дуже високим показником гарантії прочитання повідомлення.

За допомогою електронної пошти є можливість здійснювати розсилання повідомлень, у які додано необхідні матеріали, файли різних форматів, включаючи графіку. Цей канал зв’язку важливий для двостороннього спілкування.

Найбільш популярним засобом спілкування на сьогодні є месенджери та соціальні мережі. Вони дають змогу не лише відправляти повідомлення, а й отримувати відповіді та коментарі.

Найбільшу увагу буде приділено системі повідомлень за допомогою вебдодатку.

Система управління сповіщеннями повинна мати можливість встановлення пріоритетів повідомленням для того, щоб найбільш важливі повідомлення були доставлені в першу чергу.

Додатковою корисною функцією системи сповіщень є можливість додати теги до кожного сповіщення, що дасть можливість виділяти та сортувати сповіщення за категоріями.

### 1.2. Архітектура системи оповіщень

Загалом системи оповіщення можна розділити на такі типи:

- новини;

- оголошення;
- попередження;
- інформація про заборонені дії;
- інформація про дії, що були успішно виконані;
- інформація про невідкладні дії.

У структурі даних сповіщення найчастіше виділяються наступні компоненти:

- відправник сповіщення;
- умови запуску сповіщення;
- час створення сповіщення та час його відправлення;
- одержувачі сповіщення;
- спосіб доставки сповіщення;
- зміст сповіщення;
- реакції на отримання сповіщення – сповіщення може потребувати додаткових операцій після перегляду.

### 1.3. Моделі системи сповіщень

Серед найпоширеніших моделей систем сповіщень можна виділити наступні:

- Публікація-підписка
- Зчитування повідомлень за запитом (pull-підхід)
- Формування повідомлень на основі push-запиту

#### 1.3.1. Публікація-підписка

При цій моделі користувачі можуть отримувати одне й те саме повідомлення, надіслане від одного чи декількох відправників. Відправник надсилає повідомлення до черги повідомлень у залежності від типу повідомлення та правил, які стосуються цього повідомлення.

#### 1.3.2. Зчитування повідомлень за запитом (pull-підхід)

Користувачі опитують сервер і перевіряють, чи є нові повідомлення після останнього запиту. У разі наявності нових

повідомлень усі останні повідомлення будуть завантажені та відображені користувачеві. Недоліком цього підходу є велика кількість запитів до серверу, що не несуть відповіді, а створюють непотрібне навантаження.

### 1.3.3. Формування повідомлень на основі push-запиту

При цьому підході ініціатором надсилання повідомлень є відправник або сервер. Ця технологія є протилежною до технології pull. Технологія базується на моделі «Публікація-підписка». Користувачі підписуються на різні типи повідомлень, і кожного разу, коли відповідне повідомлення публікується на сервері, сервер передає його користувачу. Ця технологія застосовує лише корисне навантаження. Система зберігає повідомлення в базі даних, потім надсилає його до черги, з якої, у свою чергу, повідомлення асинхронно надсилається користувачеві.

## 1.4. Патерни проєктування

Патерн проєктування – це принцип вирішення певної проблеми в рамках проєктування архітектури програм.

Найчастіше в описі патернів указують наступні складові:

- проблема, для якої підходить патерн;
- що саме підштовхує вирішити проблему тим рішенням, який пропонує патерн;
- структура класів та інших складових;
- приклад у вигляді коду, написаного певною мовою програмування;
- особливості реалізації;
- зв’язки з іншими патернами.

У роботі над системами сповіщень найпопулярнішими можна вважати наступні патерни:

- Спостерігач (Observer);
- Стратегія (Strategy);

- Шаблонний метод (Template);
- Будівельник (Builder);
- Фабричний метод (Factory);
- Декоратор (Decorator).

### 1.4.1. Спостерігач (Observer)

Цей шаблон проєктування призначений для автоматичного оповіщення всіх залежних об’єктів-спостерігачів (Observer) при зміні стану одного об’єкта (Subject). Визначає відношення між об’єктами як «один до багатьох».

Складові шаблону «Спостерігач»:

- Інтерфейс Subject – використовується для реєстрації, видалення та оповіщення спостерігачів (Observer).
- Інтерфейс Observer – забезпечує доставку повідомлень при зміні стану об’єкта, за яким ведеться спостереження (Subject).

При зміні стану об’єкта, що спостерігається, доставка повідомлень може здійснюватися за одним із двох сценаріїв:

- Кожному з зареєстрованих спостерігачів надсилається повна інформація про стан об’єкта;
- Спостерігачам надсилається повідомлення, що була здійснена зміна стану, а кожен зі спостерігачів за потреби сам запитує необхідну йому інформацію про суб’єкт спостереження.

### 1.4.2. Стратегія (Strategy)

Це шаблон проєктування із категорії «шаблонів поведінки». Він дозволяє змінювати алгоритми в залежності від типу користувача або типу даних, що оброблюються.

### 1.4.3. Шаблонний метод (Template)

Це шаблон проєктування, що належить до категорії «шаблонів поведінки». Він визначає загальну основу алгоритму, при чому окремі

операції виносяться в підкласи для реалізації кроків алгоритму, не змінюючи його загальної структури. Шаблонний метод описує окремі кроки алгоритму, що реалізовані підкласами, а патерн «Стратегія» визначає алгоритм на етапі виконання програми.

### 1.4.4. Будівельник (Builder)

Це шаблон проєктування, що відноситься до категорії «твірних шаблонів». Він дає змогу створювати покроково складні об’єкти і дає можливість використовувати блоки коду для побудови різних форм об’єктів.

Патерн розбиває процес створення об’єкта на окремі кроки, які викликаються почергово.

### 1.4.5. Фабричний метод (Factory)

Це шаблон проєктування, що відноситься до категорії «твірних шаблонів». Він визначає інтерфейс суперкласу для створення об’єктів і дозволяє під класом змінювати вид об’єктів, що створюються.

### 1.4.6. Декоратор (Decorator)

Це структурний шаблон проєктування, що дозволяє додавати нові функціональні можливості до об’єкта.

Цей патерн дуже зручно застосовувати, коли необхідно надсилати сповіщення за допомоги різних каналів зв’язку – наприклад, через email, через SMS, соціальні мережі, месенджери тощо – та поєднувати кілька способів одночасно. Патерн «Декоратор» використовує агрегацію об’єктів в об’єкті-обгортці, який виконує функціонал базового об’єкта, після чого додає новий функціонал.

## 1.5. Функціональні та нефункціональні вимоги

При створенні системи оповіщень і подій найчастіше враховують такі функціональні вимоги:

- формування та редагування сповіщень;
- надсилання сповіщень;

- забезпечення можливості надсилання як поодиноких, так і групових сповіщень;
- забезпечення можливості скасувати повідомлення або за запитом, або у встановлений час;
- розбиття сповіщень за категоріями, можливість увімкнення або вимкнення сповіщень певних категорій;
- встановлення пріоритетів для сповіщень;
- забезпечення збереження налаштувань користувача – підписки на категорії сповіщень, налаштування послідовностей категорій;
- реєстрація всіх сповіщень: надісланих, доставлених, переглянутих і скасованих;
- забезпечення можливості підписки на сповіщення як окремих користувачів, так і груп користувачів за ієрархією структури університету.

До нефункціональних вимог такої системи можна віднести:

- Надійність – повідомлення не повинні бути втрачені і/або продубльовані.
- Доступність – обов’язкове оброблення помилок при збої будь-якого з компонентів системи.
- Продуктивність
- Кросплатформенність – підтримка більшістю сучасних веббраузерів
- Швидкість – низька затримка при доставці сповіщення
- Масштабованість – можливість підтримки великої кількості користувачів.

### 1.6. Компоненти системи

У якості складових системи оповіщень та подій можна виділити такі компоненти:

- служба сповіщень – створення та видалення сповіщень через API, відправлення сповіщення певному отримувачу;
- обробник масових розсилок і розбиття сповіщень за пріоритетами – доставка сповіщень одному певному користувачу, групі користувачів, групам користувачів або ж усім користувачам;
- оброблення та налаштування сповіщень – увімкнення та вимкнення певних сповіщень для конкретного користувача;
- обмежувач – контролювання та обмеження надходження сповіщень (за необхідності) та врівноваження навантаження;
- верифікація й валідація сповіщень і розбиття за пріоритетами – у структурі сповіщення містяться отримувач (або група отримувачів), відправник і позначення пріоритетності, яке може мати одне з таких значень: невідкладний, високий, звичайний, низький;
- черга пріоритетності сповіщень – проходження;
- планування сповіщень – певне сповіщення може приходити до того ж користувача щомісячно;
- база даних сповіщень – зберігає всі відіслані сповіщення включно з часом створення та їхнім статусом, а також з інформацією про відправника та отримувачів;
- служба аналізу сповіщень.

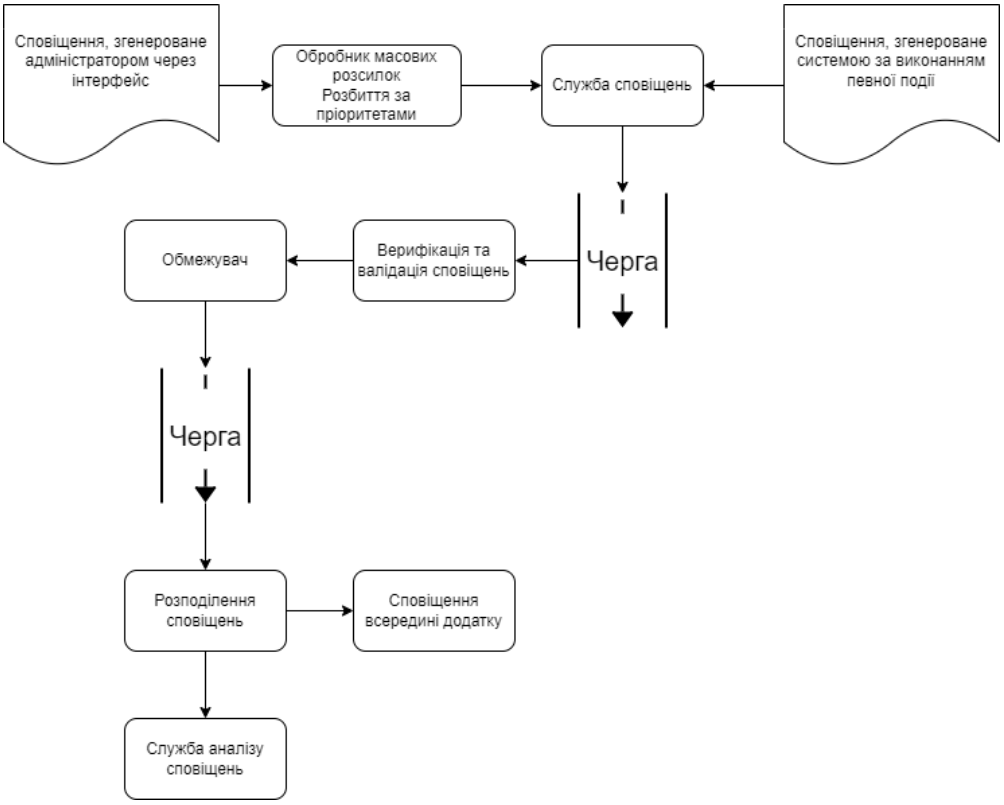



Схема архітектури системи оповіщень та подій

# РОЗДІЛ 2
# ТЕХНОЛОГІЇ ТА ЗАСОБИ СТВОРЕННЯ ТЕХНОЛОГІЇ ОПОВІЩЕНЬ

## 2.1.Фреймворки та бібліотеки

Програмний фреймворк – це комплекс програмних рішень, які можна комбінувати між собою для створення власних додатків. Це забезпечує можливість повторного використання коду та швидкого процесу розроблення, а також полегшує налагодження програми.

Бібліотека – це збірка об’єктів, створена для допомоги у вирішенні задач, близьких за тематикою.

Основна відмінність між фреймворком і бібліотекою полягає в можливості контролю над застосунком. У випадку з бібліотекою, її можна впровадити в додаток лише за наявності в ній потреби. А при використанні фреймворку ситуація цілком протилежна – він дозволяє розробнику додавати свій власний код, але й він вирішує, чи використовувати цей самий код або лише його частину.

Найчастіше для створення інтерфейсу вебдодатку обирають:

- Angular – фреймворк, створений у 2010 році компанією Google;
- React – бібліотека, яку створила компанія Meta у 2013 році (тоді вона називалася Facebook);
- Vue – фреймворк, створений колишнім співробітником компанії Google Еваном Ю (Evan You) у 2013 році.

### 2.1.1. Angular

Серед головних переваг цього фреймворку можна виділити наступні:

- постійні оновлення – вони запускаються кожні 6 місяців, при цьому нові версії спираються на попередні та

залишаються актуальними протягом року, за який розробник за потреби може оновити свій код;

- висока продуктивність за невисоку ціну – Angular має всі необхідні компоненти, упаковані у власну офіційну бібліотеку та забезпечені надійною безпекою, що зменшує потребу в сторонніх бібліотеках і, відповідно, зменшує витрати на створення додатків;
- простота в обслуговуванні – цей фреймворк використовує полегшену версію JavaScript під назвою TypeScript, що полегшує знаходження помилок у коді, а також імпортує залежності, налаштовані під тестування додатку;
- багаторазове використання – одним із головних особливостей Angular є чистий і послідовний код, який можна використовувати при розробленні інших додатків, що зменшує витрати на їхнє створення;
- декларативний інтерфейс – на відміну від React, у якому використовується JSX, верстка вебдодатку відбувається завдяки HTML та CSS/SCSS, а функціонал описується за допомоги JavaScript;
- проста архітектура – вона дозволяє розробникам створювати вебдодатки, не створюючи великі блоки коду, що робить його легким та спрощує користувачам навігацію.

### 2.1.2. React

Це бібліотека для мови JavaScript, яка була створена компанією Meta (тоді вона ще називалася Facebook) у 2013 році.

Серед основних переваг React можна виділити наступні:

- багаторазове використання коду – певні блоки коду розробник може вставляти в інші проєкти;

- простота освоєння – не дивлячись на те, що ця бібліотека складніша за Vue, але крива її освоєння менш крута, аніж в Angular;
- налаштовуваність – на відміну від Angular, який є фреймворком, React дозволяє додавати до проєкту лише ті компоненти, які вам потрібні;
- підходить для SEO – простота коду з використанням React дозволяє швидко завантажувати вебдодаток, що є дуже важливо, оскільки, за аналізом компанії Google, відвідувачі мобільних сайтів закривають їх на 123 відсотки частіше, якщо час завантаження сторінки збільшити з 1 до 10 секунд;
- React Development Tools – розширення для Google Chrome, Mozilla Firefox і Microsoft Edge, яке дозволяє вносити зміни у вебдодаток прямо в браузері та перевіряти окремі його компоненти;
- перенесення на React Native – при міграції компонентів із React JS на цю технологію, яка використовується для створення мобільних додатків, єдине, що потрібно зробити розробнику, – це внести зміни до коду, щоб підлаштувати інтерфейс для мобільних пристроїв;
- одностороннє зв’язування даних – це передача функціоналу від батьківського компоненту до дочірнього у вигляді аргументів, саме за таким принципом працюють додатки на React – будь-які зміни в дочірніх компонентах ніяк не впливають на роботу батьківських;
- перенесення технологій – оскільки React є бібліотекою, розробник може підключити свій React-код до іншого проєкту, незалежно від архітектури.

### 2.1.3. Vue.js

Цей фреймворк є відносно новим на ринку, тож станом на зараз не так багато компаній використовують його для створення проєктів, серед яких, однак, можна знайти дві найбільші компанії в Азії – Alibaba, яка відома своїм інтернет-магазином Aliexpress, і виробник електронної техніки Xiaomi.

Про переваги та недоліки Vue.js доступно небагато інформації, але достеменно відомо, що він об’єднує найкраще зі світів фреймворків і бібліотек, а саме двостороннє зв’язування даних із Angular і гнучкість коду з React.

Vue найкраще використовувати у випадках легких, але високопродуктивних, інтуїтивно зрозумілих додатків, оскільки додатки швидко готуються до виходу на ринок без шкоди для продуктивності або функціональності. Давайте коротко розглянемо, що робить Vue JS вигідним вибором для бізнесу.

Найкраще він підходить для створення легких, але при цьому інтуїтивно зрозумілих додатків.

Серед головних його переваг можна виділити наступні:

- малий розмір – він важить усього 21 кілобайт, тож він швидко завантажується на комп’ютер;
- простота – якщо на тому ж jQuery певний сегмент може займати десятки рядків, то у Vue.js вони перетворюються в один або два рядки;
- підвищення продуктивності – на відміну від Angular, цей фреймворк має свій віртуальний DOM, який швидше складає структуру вебдодатку і є легшим за той, який має React;
- швидке виявлення помилок – процес налагодження коду відбувається паралельно з його написанням завдяки тому, що фреймворк дає розробникові можливість відтворити

інтерфейс користувача на основі написаного ним коду в режимі реального часу.

### 2.1.4. Порівняння

При обиранні технології для роботи над проєктом важливо врахувати такі чинники:

- Angular являє собою розвинений структурний фреймворк JavaScript, який дає доступ до всіх своїх бібліотек, React дозволяє розробникові встановлювати до свого проєкту лише потрібні для нього компоненти, а Vue є легким і прогресивним фреймворком із відкритим кодом;
- Angular і Vue.js використовуються для створення комплексних інтерфейсів до вебдодатків, у цей же час React дозволяє створювати односторінкові та мобільні вебдодатки;
- Vue.js може похвалитися здібністю створювати красиві дизайни для вебдодатків, у той час, як Angular обирають за високу продуктивність і широкий функціонал;
- Робота з Angular і React потребує достатнього освоєння JavaScript, у цей же час Vue.js орієнтований на розробника-початківця.

| | **Angular** | **React** | **Vue** |
|---|---|---|---|
| Автори фреймворку | Google | Facebook | Еван Ю – колишній працівник Google |
| Сайт із офіційною документацією | angularjs.org | reactjs.org | vuejs.org |
| Спільнота | Найбільша спільнота, що включає в себе розробників із Google | Велика спільнота, що включає в себе розробників компанії Meta | Зростаюча спільнота, що включає в себе open-source розробників |
| Кількість вакансій на LinkedIn | Від 5 до 8 тисяч вакансій | Від 8 тисяч вакансій | Менше 3 тисяч вакансій |
| Крива освоєння | Крута | Поступова | Плавна |
| Для чого підходить? | Великі додатки<br>Додатки, що працюють у реальному часі<br>Масштабовані додатки | Крос-платформні додатки<br>Додатки, орієнтовані на швидке розроблення та розгортання<br>Розширення функціоналу існуючого додатку | Додатки з орієнтацією на ранній вихід на ринок<br>Невеликі додатки<br>Інтуїтивно зрозумілі додатки |
| Недоліки | Складний для створення крос-платформних додатків<br>Не підходить для SEO<br>Документація рідко онов-люється<br>Потрібен високий рівень освоєння JSX | Потрібен високий рівень освоєння JSX<br>Документація не відповідає відповідним вимогам<br>Часті оновлення призводять до необхідності зміни коду | Не надто широка спільнота<br>Невелика кількість великих проєктів для освоєння<br>Проблеми з роботою в Safari та iOS |

## 2.2. Системи керування базами даних

Система керування базами даних (СКБД) – це набір програм, який дозволяє користувачам створювати, підтримувати та контролювати доступ до бази даних.

Серед найпопулярніших серед розробників СКБД можна виділити такі:

- PostgreSQL;
- MySQL;
- MongoDB;
- MS SQL Server.

### 2.2.1. PostgreSQL

PostgreSQL – це об'єктно-реляційна система керування базами даних із відкритим програмним кодом.

У якості головних переваг PostgreSQL можна виділити такі:

- широкий функціонал – СКБД надає можливість працювати над керуванням паралельним доступом із допомогою багатоверсійності, відновлювати дані в певний момент часу, доступ до асинхронної реплікації, оперативного резервного копіювання;
- відповідність стандартам – PostgreSQL підтримує більшість типів даних із MS SQL Server 2008, а також двійкові дані на кшталт картинок і відео.

### 2.2.2. MySQL

MySQL – це реляційна система керування базами даних, яка була створена для навчання принципам роботи з базами даних. Ця СКБД найчастіше використовується для розроблення вебдодатків.

У якості переваг можна виділити такі особливості:

- захищеність даних – СКБД може похвалитися забезпеченням безпечної та надійної роботи;

- висока продуктивність – вона забезпечується чіткою структурою зберігання даних;
- безперебійна робота – вона забезпечується завдяки спеціалізованим кластерним серверам і реплікації за методом «master-slave».

### 2.2.3. MongoDB

Серед основних переваг цієї СКБД можна виділити такі:

- балансування навантаження – MongoDB може працювати на кількох серверах, розподіляючи навантаження або дублюючи дані, щоб підтримувати працездатність системи.

| | PostgreSQL | MySQL | MongoDB | MS SQL Server |
|---|---|---|---|---|
| Модель бази даних | об’єктно-реляційна | реляційна | документно-орієнтована | реляційна |
| Тип ліцензії | Open-source | Open-source | Open-source | комерційна |
| Механізм запитів | Засновано на SQL | NoSQL | Засновано на SQL | Засновано на SQL |
| Паралельний доступ | Дозволено | Дозволено | Дозволено | Дозволено |
| Підтримка іншими мовами | Perl, Python, .Net, C, C++, Java | C, C#, C++, Java, Perl, Haskell | PowerShell, Python, PHP, Perl, Ruby, Lisp, JavaScript, Java, Go, C++, C#, Delphi, C | .NET, Java, Python, PHP, Visual Basic, Ruby |
| Принцип реплікації | Master-slave | Master-master, Master-slave | Master-slave | Залежить від версії СКБД |

# РОЗДІЛ 3
# РОЗРОБЛЕННЯ СЕРВІСУ СПОВІЩЕНЬ І ОГЛЯД ЙОГО КОМПОНЕНТІВ

## 3.1. Огляд API для сповіщень

У системі управління бізнес-процесами університету сповіщення зберігаються як із боку відправника, так і з боку отримувача. Відповідно, для кожного з цих форматів формуються окремі запити.

### 3.1.1. Сповіщення з точки зору відправника

Для відправника сповіщення зберігаються з наступними параметрами:

- id – ідентифікатор сповіщення, формується при створенні сповіщення відповідно до UUID;
- creator_full_name – повне ПІБ відправника сповіщення;
- creator_id – ідентифікатор відправника сповіщення;
- errors_level – рівень помилок;
- content – зміст сповіщення;
- priority – пріоритет сповіщення (0 – невідкладний, 1 – високий, 3 – звичайний, 4 – низький);
- addressees – масив користувачів-отримувачів сповіщення;
- tags – масив тегів (ключових фраз), що відносяться до сповіщення;
- created – дата й час створення сповіщення за всесвітнім координованим часом (UTC);
- modified – дата й час останньої зміни в сповіщенні за UTC.

Окремо в цій системі зберігаються теги до сповіщень (tag), у структурі яких присутні такі параметри:

- id – ідентифікатор тегу відповідно до UUID;
- title – назва тегу;

- color – колір тегу за системою RGB.

Також свою структуру має об’єкт «addressee», який позначає отримувача сповіщення:

- id – ідентифікатор сповіщення, який формується для окремого отримувача відповідно до UUID;
- created – дата й час створення сповіщення за UTC;
- modified – дата й час останньої зміни в сповіщенні за UTC;
- errors – масив помилок;
- errors_level – рівень помилок;
- full_name – повний ПІБ отримувача сповіщення;
- first_retrieved_on – дата й час першого перегляду сповіщення користувачем за UTC;
- first_read_on – дата й час прочитання сповіщення користувачем за UTC.

Для керування сповіщеннями з боку відправника передбачені наступні запити:

- GET /api/notification/ – отримання списку всіх сповіщень – у якості параметрів передаються ідентифікатор відправника (creator_id), поле, за яким відбувається сортування (ordering), сторінка сповіщень (page), пріоритет сповіщень (priority) та поле пошуку (search), а в тілі відповіді на запит передається JSON-структура, яка містить загальну кількість сповіщень (count), посилання на наступну (next) та попередню сторінки (previous) і масив зі сповіщеннями з тієї сторінки, номер якої було вказано в параметрах запиту (results).

```
{
  "count": 123,
  "next": "http://api.example.org/accounts/?page=4",
  "previous":
           "http://api.example.org/accounts/?page=2",
  "results": [
    {
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "creator_full_name": "string",
```

```
      "creator_id":
"3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "content": "string",
      "priority": 0,
      "addressees": [
        {
          "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
          "created": "2022-11-02T07:54:54.784Z",
          "modified": "2022-11-02T07:54:54.784Z",
          "errors": {
            "additionalProp1": "string",
            "additionalProp2": "string",
            "additionalProp3": "string"
          },
          "errors_level": 0,
          "full_name": "string",
          "first_retrieved_on":
"2022-11-02T07:54:54.784Z",
          "first_read_on": "2022-11-02T07:54:54.784Z"
        }
      ],
      "tags": [
        {
          "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
          "title": "string",
          "color": "string"
        }
      ],
      "created": "2022-11-02T07:54:54.784Z",
      "modified": "2022-11-02T07:54:54.784Z"
    }
  ]
```


Приклад тіла відповіді на запит GET /api/notifications/

- POST /api/notification/ – створення нового сповіщення – у тілі запиту передається JSON-структура, в якій зберігаються ідентифікатор відправника сповіщення (creator_id), зміст сповіщення (content), масив ідентифікаторів отримувачів сповіщення (addressees_ids) та масив ідентифікаторів тегів до сповіщення (tags_ids).


```
{
  "creator_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "content": "string",
  "priority": 0,
  "addressees_ids": [
    "3fa85f64-5717-4562-b3fc-2c963f66afa6"
  ],
  "tags_ids": [
    "3fa85f64-5717-4562-b3fc-2c963f66afa6"
```

```
  ]
}
```

Приклад тіла запиту POST /api/notification/

У тілі відповіді передається JSON-структура з ідентифікатором сповіщення (id), ПІБ (creator_full_name) та ідентифікатором його відправника (creator_id), текстом сповіщення (content), пріоритетом (priority), масивом отримувачів (addressees), масивом тегів (tags), датою й часом створення сповіщення за всесвітнім координованим часом, також відомим як UTC (created), та датою й часом останнього редагування за тим самим часовим поясом (modified).

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "creator_full_name": "string",
  "creator_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "content": "string",
  "priority": 0,
  "addressees": [
    {
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "created": "2022-11-02T08:49:13.012Z",
      "modified": "2022-11-02T08:49:13.012Z",
      "errors": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "errors_level": 0,
      "full_name": "string",
      "first_retrieved_on": "2022-11-02T08:49:13.012Z",
      "first_read_on": "2022-11-02T08:49:13.012Z"
    }
  ],
  "tags": [
    {
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "title": "string",
      "color": "string"
    }
  ],
  "created": "2022-11-02T08:49:13.012Z",
  "modified": "2022-11-02T08:49:13.012Z"
}
```

Приклад тіла відповіді на запит POST /api/notification

- GET /api/notification/{id}/ – отримання одного сповіщення за його ідентифікатором – у якості параметру передається id сповіщення, у відповідь повертається JSON-структура з тими самими параметрами, що й в тілі відповіді на попередній запит.
- GET /api/notification_tag/ – отримання списку всіх тегів до сповіщень – єдиним параметром до цього запиту є сторінка списку (page), у відповідь повертається JSON-структура, що містить загальну кількість записів (count), посилання на наступну (next) та попередню сторінки (previous) і масив тегів за тією сторінкою, яка була вказана в запиті (results).

```
{
  "count": 123,
  "next": "http://api.example.org/accounts/?page=4",
  "previous":
"http://api.example.org/accounts/?page=2",
  "results": [
    {
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "title": "string",
      "color": "string"
    }
  ]
}
```

Приклад тіла відповіді на запит GET /api/notification_tag/

- GET /api/notification_tag/{id}/ – отримання одного тегу за його ідентифікатором – у параметрах передається id тегу, а у відповідь повертається JSON-структура, в якій указано ідентифікатор тегу (id), його назва (title), та код закріпленого за ним кольору в системі RGB (color).

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "title": "string",
  "color": "string"
}
```

Приклад тіла відповіді на GET /api/notification_tag/{id}/

### 3.1.2. Сповіщення з точки зору отримувача

Для потенційного отримувача структура сповіщення виглядатиме наступним чином:

- id – ідентифікатор сповіщення відносно користувача, якому воно було відправлене;
- created – дата й час створення сповіщення за UTC;
- modified – дата й час останнього редагування за UTC;
- errors – масив помилок;
- errors_level – рівень помилок;
- errors_confirmed – масив підтверджених помилок;
- notification_id – глобальний ідентифікатор сповіщення;
- creator_full_name – ПІБ відправника сповіщення;
- creator_id – ідентифікатор сповіщення;
- content – текст сповіщення;
- priority – пріоритет сповіщення;
- tags – масив тегів сповіщення;
- changed_entity_type – тип зміненого об’єкта;
- old_entity – попередній стан об’єкта;
- new_entity – новий стан об’єкта;
- first_retrieved_on – дата й час першого перегляду сповіщення користувачем за UTC;
- first_read_on – дата й час першого прочитання сповіщення користувачем за UTC.

Параметри changed_entity_type, old_entity та new_entity використовуються в тих сповіщеннях, які були згенеровані самою системою управління бізнес-процесами за виконанням події внесення змін у запис.

Користувач може через інтерфейс відправляти щодо сповіщень, які він отримав, наступні запити:

- GET /api/personal_notification/ – отримання списку сповіщень, які отримав певний користувач – у цьому запиті

передається номер сторінки (page), а в тілі відповіді передається JSON-структура, в якому містяться загальна кількість сповіщень (count), посилання на наступну (next) та попередню сторінки (previous) і масив сповіщень на заданій у параметрах запиту сторінці (results).


```
{
  "count": 123,
  "next": "http://api.example.org/accounts/?page=4",
  "previous":
"http://api.example.org/accounts/?page=2",
  "results": [
    {
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "created": "2022-11-26T10:15:50.875Z",
      "modified": "2022-11-26T10:15:50.876Z",
      "errors": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "errors_level": 0,
      "errors_confirmed": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "notification_id": "string",
      "creator_full_name": "string",
      "creator_id": "3fa85f64-5717-4562-b3fc-
2c963f66afa6",
      "content": "string",
      "priority": "string",
      "tags": [
        {
          "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
          "created": "2022-11-26T10:15:50.876Z",
          "modified": "2022-11-26T10:15:50.876Z",
          "errors": {
            "additionalProp1": "string",
            "additionalProp2": "string",
            "additionalProp3": "string"
          },
          "errors_level": 0,
          "errors_confirmed": {
            "additionalProp1": "string",
            "additionalProp2": "string",
            "additionalProp3": "string"
          },
          "title": "string",
          "color": "string"
        }
      ],
      "changed_entity_type": 0,
      "old_entity": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "new_entity": {
```

```
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "first_retrieved_on": "2022-11-26T10:15:50.876Z",
      "first_read_on": "2022-11-26T10:15:50.876Z"
    }
  ]
}
```


Приклад тіла відповіді на запит GET /api/personal_notifcation/

- GET /api/personal_notification/{id}/ – отримання певного сповіщення за ідентифікатором для окремого користувача – у параметрах передається ідентифікатор сповіщення відносно певного його отримувача, в тілі відповіді передається JSON-структура з тими параметрами, які вказано вище.


```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "created": "2022-11-03T08:15:58.143Z",
  "modified": "2022-11-03T08:15:58.143Z",
  "errors": {
    "additionalProp1": "string",
    "additionalProp2": "string",
    "additionalProp3": "string"
  },
  "errors_level": 0,
  "notification_id": "string",
  "creator_full_name": "string",
  "creator_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "content": "string",
  "priority": "string",
  "tags": [
    {
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "title": "string",
      "color": "string"
    }
  ],
  "changed_entity_type": 0,
  "old_entity": {
    "additionalProp1": "string",
    "additionalProp2": "string",
    "additionalProp3": "string"
  },
  "new_entity": {
    "additionalProp1": "string",
    "additionalProp2": "string",
    "additionalProp3": "string"
  },
  "first_retrieved_on": "2022-11-03T08:15:58.143Z",
```

```
  "first_read_on": "2022-11-03T08:15:58.143Z"
}
```


Приклад тіла відповіді на GET /api/personal_notification/{id}

- POST /api/personal_notification/{id}/mark_read/ – підтвердження користувачем, що він прочитав сповіщення з певним ідентифікатором (id) – у тілі запиту передається ідентифікатор відправника сповіщення (creator_id), дата й час першого перегляду сповіщення користувачем за UTC (first_retrieved_on) та дата й час першого прочитання сповіщення за UTC (first_read_on).


```
{
  "creator_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "first_retrieved_on": "2022-11-03T08:15:58.158Z",
  "first_read_on": "2022-11-03T08:15:58.158Z"
}
```


Приклад тіла запиту POST /api/personal_notification/{id}/mark_read/

У тілі відповіді передається JSON-структура того самого сповіщення з доданими значеннями first_retrieved_on та first_read_on.


```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "created": "2022-11-26T10:40:16.832Z",
  "modified": "2022-11-26T10:40:16.832Z",
  "errors": {
    "additionalProp1": "string",
    "additionalProp2": "string",
    "additionalProp3": "string"
  },
  "errors_level": 0,
  "errors_confirmed": {
    "additionalProp1": "string",
    "additionalProp2": "string",
    "additionalProp3": "string"
  },
  "notification_id": "string",
  "creator_full_name": "string",
  "creator_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "content": "string",
  "priority": "string",
  "tags": [
    {
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "created": "2022-11-26T10:40:16.832Z",
      "modified": "2022-11-26T10:40:16.832Z",
      "errors": {
```

```
          "additionalProp1": "string",
          "additionalProp2": "string",
          "additionalProp3": "string"
        },
        "errors_level": 0,
        "errors_confirmed": {
          "additionalProp1": "string",
          "additionalProp2": "string",
          "additionalProp3": "string"
        },
        "title": "string",
        "color": "string"
      }
    ],
    "changed_entity_type": 0,
    "old_entity": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "new_entity": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "first_retrieved_on": "2022-11-26T10:40:16.832Z",
    "first_read_on": "2022-11-26T10:40:16.832Z"
}
```


Приклад тіла відповіді на запит POST /api/personal_notification/{id}/mark_read/

### 3.2. Огляд бази даних

Для зберігання даних у системі управління бізнес-процесами університету використовується база даних, яка була реалізована в СКБД PostgreSQL. У цій базі сервіс оповіщень і подій звертається до таких таблиць:

- notification_notification – таблиця, в якій зберігаються записи сповіщень, що містять дату й час створення (created) та останнього редагування (modified), ідентифікатор сповіщення (id), зміст (content), пріоритет (priority), ідентифікатор відправника (creator_id), тип зміненого об’єкта (changed_entity_type), його попередній стан (old_entity) і новий (new_entity), масив помилок (errors), рівень помилок (errors_level) і масив підтверджених помилок (errors_confirmed);

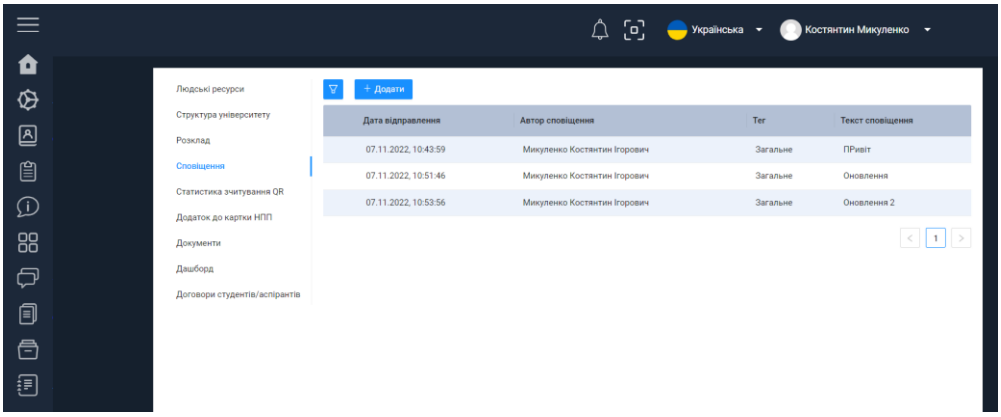
- notification_notificationtag – таблиця, де зберігаються теги, які можна додати до сповіщень, і в якій представлені такі поля: дата й час створення (created) та останнього внесення змін (modified), ідентифікатор тегу (id), колір до тегу в системі RGB (color), масив помилок (errors), рівень помилок (errors_level) і масив підтверджених помилок (errors_confirmed);
- notification_notification_tags – таблиця, в якій зберігаються зв’язки між сповіщеннями та прикріпленими до них тегами у вигляді записів із такими полями: ідентифікатор запису (id), ідентифікатор сповіщення (notification_id) та ідентифікатор тегу (notificationtag_id);
- notification_notificationaddressee – таблиця, в якій зберігаються відношення між сповіщенням і користувачем, якому воно було відправлено, у вигляді записів, що містять такі поля: дата й час створення сповіщення (created) та внесення останніх змін (modified), ідентифікатор відношення (id), дата й час першого перегляду (first_retrieved_on) та першого прочитання сповіщення користувачем (first_read_on), ідентифікатор отримувача (addressee_id), ідентифікатор сповіщення (notification_id), масив помилок (errors), рівень помилок (errors_level) і масив підтверджених помилок (errors_confirmed).

### 3.3. Огляд модулів інтерфейсу системи

Інтерфейс системи оповіщень та подій реалізовано в таких модулях:

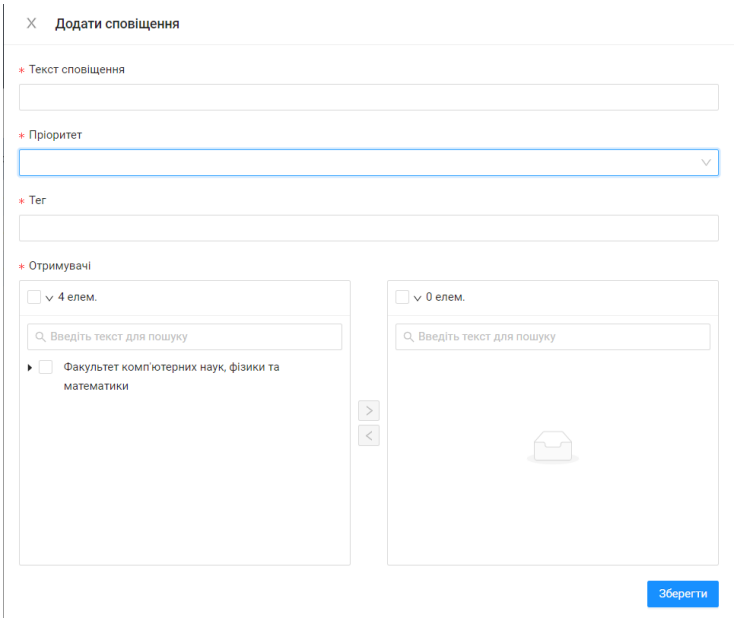
- Notifies.js – при завантаженні цього модулю виконується GET-запит /api/notification/, у відповідь на який надається JSON-структура, з якої у вигляді таблиці виводиться

зазначена сторінка сповіщень з інформацією про дату й час їхнього створення, ПІБ їхніх відправників, теги цих сповіщень та їхній зміст. Доступ до цього модулю з боку фронт-енду відкритий для ректора, проректорів, деканів, віцедеканів, завідувачів кафедр, викладачів та адміністраторів вебдодатку.



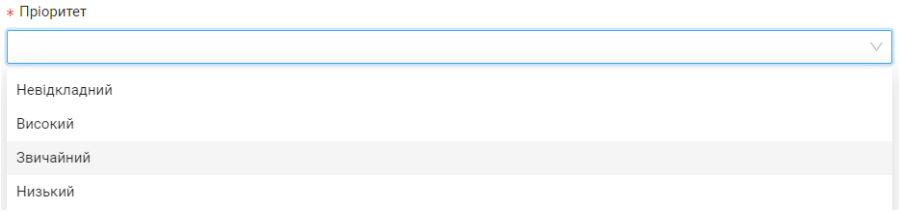

Панель керування додатками

- AddNotify.js – цей модуль представляє з себе форму з полями «Текст сповіщення» (параметр content), «Пріоритет» (priority), «Тег» (tags_ids) та «Отримувачі» (addressees_ids) та кнопкою «Зберегти», при натисненні на яку із заповнених даних у формі, до яких додається ідентифікатор користувача, який створює сповіщення, формується JSON-структура в якості тіла POST-запиту /api/notification/. При введенні даних перевіряється, чи всі поля заповнені.



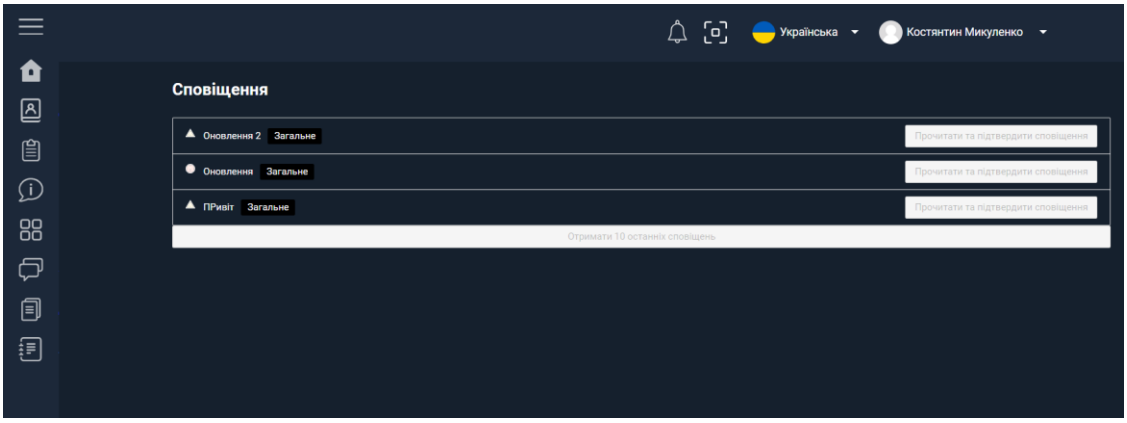

Форма для створення нового сповіщення

- PrioritySelect.js – модуль для встановлення пріоритету сповіщення. Він використовується у формі створення сповіщення. Варіанти вибору взяті з enum-масиву в модулі enums.js. Можливі пріоритети – Невідкладний, Високий, Звичайний, Низький. При відображенні сповіщення пріоритети позначаються відповідною позначкою та кольором.



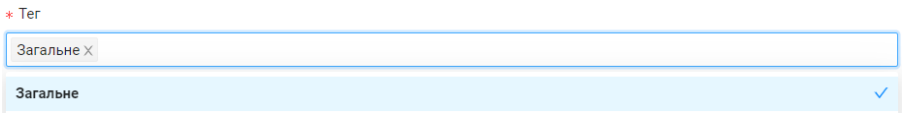

Поле для вибору пріоритету сповіщення

- Events.js – модуль для отримання списку сповіщень, які адресовані певному користувачу. При завантаженні цього модулю виконується GET-запит /api/personal_notification/, у відповідь на який надходить JSON-масив, кожен елемент якого містить структуру сповіщення. Цей масив передається для відображення користувачеві впорядковано в залежності від пріоритету та даті створення.




Сторінка зі сповіщеннями, надісланими авторизованому користувачеві

- MarkRead.js – цей модуль призначений для оброблення події прочитання сповіщення. Ця подія виникає, коли користувач натискає на відповідну кнопку на сповіщенні. Інформація про подію передається в базу даних через модуль API /api/personal_notification/{id}/mark_read/. Після оброблення цієї події кнопка блокується.
- NotificationTagSelect.js – модуль для встановлення тегів сповіщення. Теги – це ключові слова чи фрази, які використовуються для швидкого пошуку інформації та коротко описують суть даних. Теги відображаються поряд із текстом сповіщення. Масив тегів обирається за допомогою API /api/notification_tag/.




Поле для обрання тегів до сповіщення

### 3.4. Розгортання

Для розгортання вебдодатку в мережі використовується програма Docker, яка генерує такі контейнери:

- ksu24.frontend – контейнер, який містить модулі інтерфейсу користувача;

- ksu24.backend – контейнер, в якому зберігається функціонал системи управління бізнес-процесами університету;
- ksu24.postgres – контейнер, в якому зберігається база даних цієї системи.

# ВИСНОВКИ

У процесі дипломної роботи було поставлено за мету створення інтерфейсу користувача для системи оповіщень та подій, в якій генеруватимуться повідомлення, які стосуються навчального процесу та інших важливих подій у житті університету.

Було проведено аналіз існуючих технологій, а саме – фреймворків і бібліотек, СКБД, проведено їхнє порівняння за основними критеріями.

Було проведено аналіз архітектури системи оповіщень та подій, а саме – було визначено її функціонал і створено діаграму сценарію використання.

Було виконано розроблення інтерфейсу користувача, з допомогою якого користувачі отримуватимуть від адміністрації повідомлення, пов’язані з життям університету та навчальним процесом.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Marketing Strategies. App & Mobile. Page load time statistics [Електронний ресурс]. Режим доступу: https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks-load-time-vs-bounce/
2. React Development Tools Documentation Guide [Електронний ресурс]. Режим доступу: https://beta.reactjs.org/learn/react-developer-tools
3. PostgreSQL Documentation Guide [Електронний ресурс]. Режим доступу: https://www.postgresql.org/docs/
4. Бібліотека JavaScript для створення вебдодатків – React [Електронний ресурс]. Режим доступу: https://uk.reactjs.org/
5. Початок роботи з React [Електронний ресурс]. Режим доступу: https://uk.reactjs.org/docs/getting-started.html
6. Angular - https://angular.io/
7. What Is Angular [Електронний ресурс]. Режим доступу: https://angular.io/guide/what-is-angular
8. Vue.js - The Progressive JavaScript framework [Електронний ресурс]. Режим доступу: https://vuejs.org/
9. Vue.js Introduction [Електронний ресурс]. Режим доступу: https://vuejs.org/guide/introduction.html
10. Docker: Accelerated, Containerized Application Development [Електронний ресурс]. Режим доступу: https://www.docker.com/
11. Docker Overview [Електронний ресурс]. Режим доступу: https://docs.docker.com/get-started/overview/
12. Refactoring.Guru - Патерни проєктування [Електронний ресурс]. Режим доступу: https://refactoring.guru/uk/design-patterns
13. Hans-Jürgen Schönig. Mastering PostgreSQL 11: Expert techniques to build scalable, reliable, and fault-tolerant database applications, 2nd Edition. 2018. — 446 c.

14. Salahaldin Juba, Andrey Volkov. Learning PostgreSQL 11: A beginner's guide to building high-performance PostgreSQL database solutions, 3rd Edition. 2019. — 556 с.
15. Django Overview [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/4.1/intro/overview/
16. How the Web works [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works
17. Специфікація HTML [Електронний ресурс]. Режим доступу: https://www.w3.org/TR/html40/
18. Специфікація CSS [Електронний ресурс]. Режим доступу: https://www.w3.org/TR/CSS21/
19. Thomas Connolly, Carolyn Begg. Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition. 2014. 1440 с.