МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук фізики та математики
Кафедра інформатики, програмної інженерії та економічної кібернетики

ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ СЕРВІСНОЇ АРХІТЕКТУРИ УПРАВЛІННЯ БІЗНЕС-ПРОЦЕСАМИ УНІВЕРСИТЕТУ. ДИЗАЙН ІНТЕРФЕЙСУ КОРИСТУВАЧА

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти "магістр"

Виконав: здобувач 2 курсу 241М групи
Спеціальності 121 Інженерія програмного забезпечення
Освітньо-професійної програми «Інженерія програмного забезпечення» другого (магістерського) рівня освіти
Лукінов Геннадій Геннадійович
доктор педагогічних наук, професор
Співаковський Олександр Володимирович,
Рецензент кандидатка фізико-математичних наук, доцентка
Бистрянцева Анастасія Миколаївна

Херсон – 2021

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

1. **API** — програмний інтерфейс додатку (application programming interface)
2. **HTML** — мова розмітки документів (Hypertext Markup Language).
3. **REST** — підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів (Representational State Transfer)
4. **JSON** — текстовий формат обміну даними на основі синтаксису ECMAScript (Javascript object notation)
5. **HTTP** — протокол передачі даних прикладного рівня (Hypertext Transfer Protocol).

ВСТУП

Необхідність системи для управління бізнес-процесами університету набула потреби як на території України, так і в усьому світі. Особливо цьому сприяє процес діджиталізації у багатьох галузях та явна необхідність видаленого доступу до освітніх ресурсів та взаємодія з ними.

Однією із сфер, що інтенсивно розпочали впровадження у своїй роботі інформаційних технологій, можна відзначити сферу надання адміністративних послуг державними та комунальними органами всіх гілок влади. Так, Кабінет Міністрів України декларує головною метою переведення надання адміністративних послуг в електронний формат збільшення ефективності управління. При цьому, ефективність забезпечується як власне модернізацією бізнес процесів, так і супутнім переглядом взаємодії окремих елементів та скороченням паперового документообігу, що у свою чергу сприяє дебюрократізації.

Все це пов’язано із розвитком інформаційних технологій. Перехід до концепції діджиталізації передбачає впровадження більш гнучких та безшовних процесів, їх об’єднання та оптимізацію. Ця трансформація дозволить якісно задовольняти внутрішні організаційні потреби та взаємодіяти зі студентами.

Актуальність дослідження полягає в необхідності уніфікації підходів до управління електронними ресурсами, прискоренні об’єднання різнорівневих ресурсів навчального закладу в єдиний портал та забезпеченні учасників освітнього процесу доступом до персоніфікованої інформації.

Об’єктом дослідження є системи управління університетами та взаємодія між їх підрозділами.

Предмет дослідження: технології створення user-friendly інтерфейсу користувача.

Мета дослідження: проєктування та розробка клієнтської частини системи керування бізнес процесами університету та реалізація компонентів для взаємодії з системою.

У зв'язку з поставленою метою були визначені **завдання**:

1. Визначити основні принципи розробки інтерфейсу користувача.
2. Проаналізувати підхід до розробки користувацьких інтерфейсів.
3. Розглянути інструменти для розробки інтерфейсу користувача.
4. Виконати аналіз вимог до компонентів React.
5. Спроєктувати інтерфейс користувача веб-додатку.
6. Розробити та протестувати веб-додаток за допомогою бібліотеки React.

РОЗДІЛ 1
СИСТЕМИ УПРАВЛІННЯ УНІВЕРСИТЕТАМИ

1.1 Аналіз систем управління ВНЗ

Метою використання інформаційних технологій в управлінні університетами є покращення ефективності роботи усіх підрозділів, автоматизація процесів. Об’єднання та уніфікація внутрішніх сервісів, що в свою чергу оптимізує роботу університету.

Автоматизація процесів - це використання технологій для виконання повторюваних завдань або процесів у бізнесі, де ручні зусилля можна сильно спростити або замінити. Даний підхід сприяє підвищенню ефективності та спрощення процесів.

Автоматизовані системи у закладах вищої освіти дають змогу вирішувати велику кількість функцій:

– планування навчальної діяльності;

– контроль за навчальним процесом;

– аналіз результатів навчальної діяльності;

– доступ до інформації про хід навчального процесу;

– систему звітів на основі даних та статистичної інформації;

– системи забезпечення безпеки даних та контролю доступу до них

з урахуванням вимог законодавства;

– облік контингенту студентів та співробітників;

– проведення вступної кампанії;

– формування пакетів даних з метою виготовлення тих чи інших документів. [37]

1.2 Огляд існуючих сервісів для управління бізнес-процесами університету

Під керівництвом Владислава Гетьмана проведено огляд серії вітчизняних (АСУ «СТЕП 5 ПРОФ», АСУ навчальним процесом «Директива», АСУ «Університет», Пакет комп'ютерних систем ПП «Політексофт», Програмний комплекс «АЛЬМА-МАТЕР» АСУ «Вищий навчальний заклад» НДІ ПІТ, ІАС «Університет» Херсонський державний університет, Електронна система управління ВНЗ «Сократ» Вінницький національний аграрний університет) та зарубіжних (Classter,

Ellucian, PowerVista RollCall, iGradePlus, CampusAnyware, Administrative Solutions 3, mySkoolApp, Veracross) систем управління ЗВО [23].

Оглянуто основні типи функціоналів, які має та чи інша система, серед них слід відзначити нижченаведені.

1. Управління навчальними матеріалами. Можливість зберігати, редагувати та поширювати лекції, лабораторні та семінари всередині системи управління закладу.
2. Управління харчуванням. Можливість замовити та оплатити продукти харчування на території університету. Відповідно наявність функціоналу для закладів харчування, які співпрацюють з ВНЗ .
3. Звітність / аналітика. Можливість вести звіт діяльності університету. Автоматичний збір та аналіз даних для внесення до звітності.
4. Управління оцінками. Функціонал для викладачів та студентів, що дозволяє вести та контролювати журнал оцінок.
5. Управління бібліотекою. Онлайн база книжок. Бронювання книг. Контроль за поверненням книг.
6. Управління кампусом.

7. Управління навчальними програмами. Система створення та контролю за навчальними програмами дозволяє розподіляти навчальні години, планувати заняття, моніторити навчальний процес онлайн.

8. Управління фінансовою допомогою. Контроль за грантами, соціальними виплатами та стипендіями.

9. Спеціальна та факультативна освіта.

10. Онлайн-платежі.

11. Управління доступом.

12. Фінансовий менеджмент ЗВО

13. Портал для абітурієнтів.

14. Управління випускниками. Моніторинг та аналіз успішності випускників.

15. Факультет, управління персоналом.

16. Управління житлом, гуртожитками.

17. Управління збору коштів. Благодійність в межах ЗВО.

18. Планування. Планування внутрішньої та зовнішньої діяльності ЗВО.

19. Інформація для студентів / записи. Таблиці розкладів, культурних заходів та ін.

20. Студентський портал. Функціонал для студентського самоврядування..

21. Інструменти зв’язку. Чати груп, факультетів. Контакти відділів ЗВО, викладацького складу.

22. Управління громадою. Функціонал для викладацького самоврядування, профспілок.

23. Календар подій.

24. Інтерактивне навчання. Онлайн лабораторії, додатковий медіаматеріал.

25. Навігація по території ЗВО.

26. Опитування, голосування.
27. Управління безпекою.
28. Управління студентською групою.
29. Електронний документообіг. [37]

1.3 Програмування інтерфейсу користувача

Інтерфейс користувача (UI) - це точка взаємодії людини та комп’ютера. Це може включати екрани дисплея, клавіатури та зовнішній вигляд робочого столу. Це також спосіб взаємодії користувача з програмним продуктом. Зростаюча залежність багатьох підприємств від веб-додатків та мобільних додатків змусила багато компаній приділяти все більший пріоритет інтерфейсу користувача, намагаючись покращити загальний досвід користувача. [1]

Різні типи інтерфейсів користувача включають:

- графічний інтерфейс користувача (GUI)
- інтерфейс командного рядка (CLI)
- інтерфейс користувача за допомогою меню
- сенсорний інтерфейс користувача
- голосовий інтерфейс користувача (VUI)
- інтерфейс користувача на основі форми
- інтерфейс користувача природною мовою [2]

Про інтерфейс користувача часто говорять у поєднанні з користувацьким досвідом (UX), який може включати естетичний вигляд, час відгуку та вміст, який представлений користувачеві в контексті інтерфейсу користувача. Обидва терміни підпадають під концепцію взаємодії людини з комп'ютером, яка є галуззю дослідження,

зосередженою на створенні комп'ютерних технологій та взаємодії між людьми та всіма формами IT-дизайну. [3]

Зростання уваги до створення оптимізованого користувацького досвіду призвело до появи експертів в UI та UX. Деякі мови, такі як HTML та CSS, були спрямовані на спрощення створення міцного користувацького інтерфейсу та досвіду роботи.

У ранніх комп'ютерах користувальницький інтерфейс був дуже малим, за винятком кількох кнопок на консолі оператора. Багато з цих комп’ютерів використовували перфокарти, приготовані за допомогою машин для перфорації, як основний метод введення для комп’ютерних програм та даних. Хоча перфокарти по суті застаріли в обчислювальній техніці з 2012 року, деякі машини все ще використовують систему перфокарт.

Інтерфейс користувача розвинувся з появою інтерфейсу командного рядка, який спочатку з'явився майже порожнім екраном з рядком для введення користувачем. Користувачі покладалися на клавіатуру та набір команд для навігації в обміні інформацією з комп’ютером. Цей інтерфейс командного рядка привів до виду, в якому переважали меню (списки варіантів, написаних текстом).

1.4 Особливості проєктування інтерфейсу користувача для Web

Розробка інтерфейсу користувача для управління бізнес-процесами є досить складним завданням.

Універсальні критерії надійної архітектури додатків:

- Ясність
    - Це найважливіший елемент хорошого дизайну інтерфейсу користувача. Основним мотивом розробки інтерфейсу

користувача є можливість безперебійної взаємодії з системою. Незручний дизайн не дуже цінується.

- Зручність використання
    - Є багато дій, до яких звикли користувачі. Слідкування найпопулярнішим тенденціям в UI/UX дає змогу покращити користувацький досвід.
- Зворотній зв’язок
    - Інтерфейс, повинен бути швидким та реагувати на всі зміни стану всередині додатку. Швидке завантаження або хороша швидкість користувацького інтерфейсу покращують досвід користувача. Інтерфейс повинен взаємодіяти з користувачами та інформувати про те, що відбувається.
    - Користувачі часто виконують дії помилково. Їм потрібен чітко позначений "аварійний вихід", щоб залишити небажану дію.
- Інтуїтивно зрозумілий і послідовний
    - Усі канали управління та інформація повинні бути викладені послідовно та мають бути інтуїтивно зрозумілими, щоб інтерфейс був простим у використанні та навігації. Процес проектування має бути логічним. Тобто переважно використовувані функції мають бути на першому місці.
    - Постійне повідомлення користувача про стан системи - не слід вживати жодних дій із наслідками для користувачів, не повідомляючи їх. Надайте користувачеві зворотний зв'язок якомога швидше (в ідеалі - негайно).
- Ефективність
    - Якісно зроблений інтерфейс повинен дозволити виконувати функції швидко і з найменшими зусиллями. [4]

Безпечні компоненти стійкі до прямого аналізу. Компоненти, розроблені з урахуванням прозорості, забезпечують найсильніший захист від загроз. "Security through obscurity" є протилежністю відкритого дизайну і означає, що зовнішні загрози не знають про внутрішній контроль. Вибір відкритого дизайну допомагає переконатися, що безпека міцна. Елементи керування мають бути максимально простими, але при цьому залишалися ефективними. Чим більша кількість коду та складності системи, тим більша ймовірність наявності в системі помилок та уразливостей. Використання чітких і стислих засобів контролю зменшує ризик завдяки простоті у застосуванні та використанні. В нашому випадку компонентний підхід вирішує проблему нагромадження кодової бази, всі компоненти незалежні один від одного, багаторазові, їх легко розроблювати, тестувати та підтримувати, що добре позначається на системі загалом.

РОЗДІЛ 2
ПРОЄКТУВАННЯ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБ-ДОДАТКУ УПРАВЛІННЯ БІЗНЕС-ПРОЦЕСАМИ УНВІРСИТЕТУ

2.1 Використані програмні рішення та бібліотеки

2.1.1 React

React - це бібліотека, яка поєднує швидкість, гнучкість та універсальність. Основною особливістю бібліотеки являється взаємодія стану всередині додатку і інтерфейсу. Інтерфейс являється функцією від стану, це означає, що додаток розроблений за допомогою бібліотеки реагує на будь-які зміни даних і оновлює інтерфейс синхронно з даними.

Основні особливості React:

- Легке створення динамічних програм: React спрощує створення динамічних веб-додатків, оскільки вимагає менше кодування та пропонує більше функціональних можливостей, на відміну від JavaScript, де кодування дуже швидко ускладнюється.
- Покращена продуктивність: React використовує Virtual DOM, тим самим швидше створюючи веб-додатки. Віртуальний DOM порівнює попередні стани компонентів і оновлює лише ті елементи реального DOM, які були змінені, замість того, щоб знову оновлювати всі компоненти, як це роблять звичайні веб - програми.
- Компоненти багаторазового використання: Компоненти є будівельними матеріалами будь-якої програми React, а одна програма зазвичай складається з декількох компонентів. Ці компоненти мають свою логіку та елементи керування, і їх можна використовувати повторно у всьому додатку, що в свою чергу різко скорочує час розробки програми.

- Односпрямований потік даних: React слідує за односпрямованим потоком даних. Це означає, що при розробці програми React розробники часто вкладають дочірні компоненти в батьківські компоненти. Оскільки дані рухаються в одному напрямку, стає легше налагоджувати помилки та знати, де проблема виникає у програмі на даний момент.
- Невелика крива навчання: React легко засвоюється, оскільки він переважно поєднує основні концепції HTML та JavaScript з деякими корисними доповненнями. Проте, як і у випадку з іншими інструментами та фреймворками, вам доведеться витратити деякий час, щоб правильно зрозуміти бібліотеку React.
- Його можна використовувати для розробки веб та мобільних додатків: ми вже знаємо, що React використовується для розробки веб-додатків, але це не все, що він може зробити. Існує фреймворк під назвою React Native, похідний від самого React, який користується величезною популярністю і використовується для створення мобільних додатків. Отже, насправді React можна використовувати для створення як веб-додатків, так і мобільних.
- Спеціальні інструменти для легкого налагодження: існують розширення для браузерів, які можна використовувати для налагодження програм React. Це робить процес налагодження веб-додатків React швидшим і простішим. [5]

2.1.2. Redux

Redux - це бібліотека управління станом, яку можна використовувати з будь-якою бібліотекою або фреймворком. Основна

особливість Redux полягає в можливості використовувати єдиний стан програми як глобальний і взаємодіяти з ним з будь-якого компонент. [7]

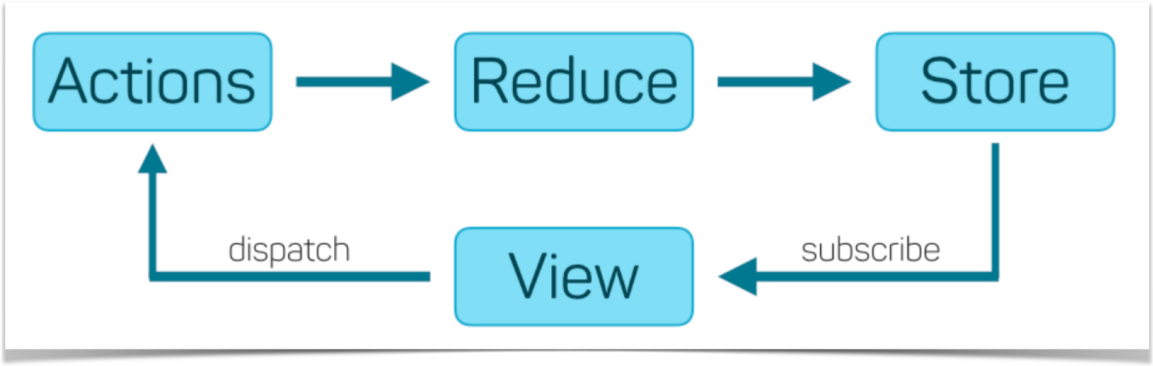


Рисунок 2.1 – Архітектура Redux

Actions: «це корисне навантаження, які надсилають дані з вашої програми до сховища. Вони є єдиним джерелом інформації для *Store*». це означає, що у разі необхідності будь-якої зміни стану необхідна зміна буде надіслана через *Actions*.

Reducers: «Дії описують той факт, що щось трапилося, але не вказують, як зміниться стан програми у відповідь. Це робота редукторів». Коли операція надсилається для зміни стану, її редуктори зобов'язані внести необхідні зміни до стану та повернути новий стан програми.

Store: за допомогою редукторів можна створити сховище даних, у якому зберігатиметься весь стан програми, рекомендується використовувати єдине сховище для всієї програми, ніж кілька сховищ, що порушуватиме використання redux, що має лише одне сховище.

Компоненти (UI): Тут зберігається інтерфейс програми.

Middlewares: проміжне програмне забезпечення можна використовувати для вирішення великої кількості завдань, включаючи асинхронні виклики API. [8]

Подібна архітектура робить веб-додаток:

- Передбачуваним. Допомагає писати програми, які поводяться послідовно і можуть працювати в різних середовищах (клієнт, сервер та рідний).
- Централізованим. Не потрібно піднімати стан до батьківських компонентів, і ми можемо використовувати стан з будь-якого компонента, який нам потрібен, завдяки такій централізованій поведінці.
- Налагодженим. Є велика кількість інструментів для налагодження програм з Redux. Завдяки цьому можливо побачити, коли, де, чому та як змінився стан вашої програми. Це дуже пришвидшує пошук та виправлення помилок і весь процес розробки та тестування загалом.
- Гнучким. Працює з будь-яким шаром інтерфейсу і має велику екосистему доповнень.

2.1.3 Ant Design

Це набір компонентів, що регулюються стандартами інтерфейсу користувача для шаблонів, тем, стилів та філософії дизайну, які можна використовувати повторно для створення програм. [9]

Система Ant Design - це відкритий вихідний код для мов дизайну інтерфейсу та бібліотеки React UI. Він поставляється з набором високоякісних компонентів React з можливістю налаштування теми. Він побудований за допомогою бібліотеки i18n, яка використовувалась також і при впровадженні багатомовності розроблюваного веб-додатку. Це дозволяє набагато спростити розробку та підтримку компонентів на основі компонентів Ant. Наприклад компоненти “Календар”, “Вибір дати”, “Вибір часу” адаптовані для англійської та української мови. Це важливо, адже перераховані компоненти використовуються майже на

всіх функціональних сторінках для фільтрування та створення даних. [10]

З моменту створення 24 квітня 2015 року система Ant Design отримала близько 75000 зірок на GitHub та понад півмільйона завантажень щотижня з реєстру npm. [11]

Ось тенденції npm для системи Ant Design (antd) порівняно з material-ui та React-bootstrap.[12]

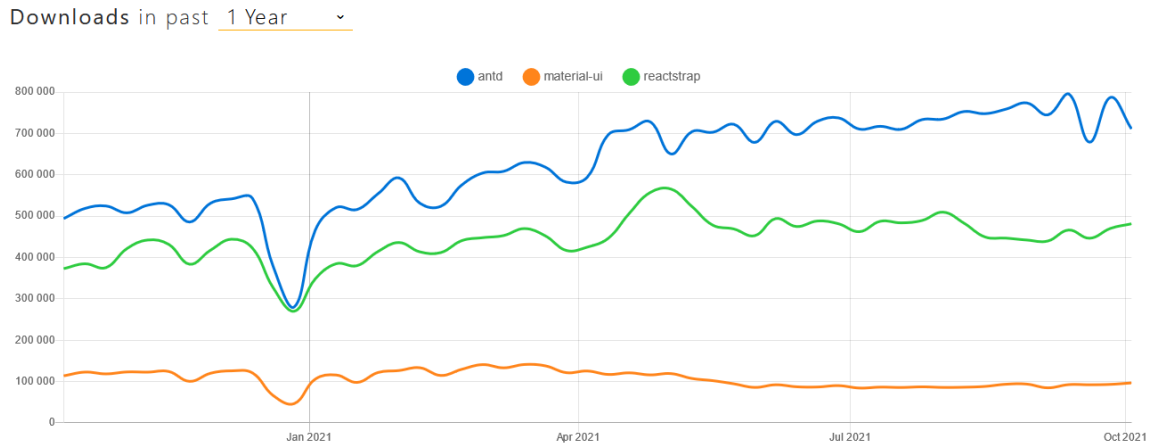
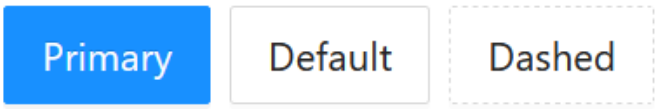


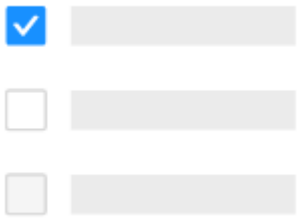
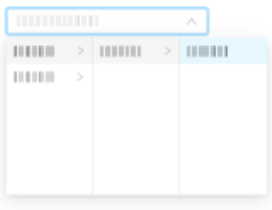
Рисунок 2.2 – Графік завантажень популярних бібліотек компонентів React

Таблиця 2.1

Типи компонентів [13]

Назва	**Опис компонентів**
Button	 Кнопка означає операцію (або серію операцій). Натискання кнопки викликає відповідну бізнес-логіку.
Typography	Основне написання тексту, включаючи заголовки, основний текст, списки тощо.
Divider	Роздільна лінія, що розділяє різний вміст.
Dropdown	Випадаючий список.

Продовження табл. 2.1

Menu	 Універсальне меню для навігації.
Pagination	 Довгий список можна розділити на кілька сторінок за і одночасно завантажуватиметься лише одна сторінка.
PageHeader	 Заголовок із вбудованими загальними діями та елементами дизайну.
Checkbox	 Використовується для вибору кількох значень з кількох параметрів.
Cascader	

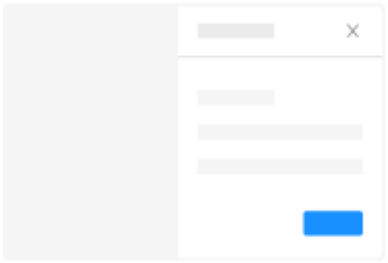
	Коли вам потрібно вибрати з набору пов'язаних наборів даних. Такі як провінція/місто/район, рівень компанії, класифікація речей.

Продовження табл. 2.1

Form	 Високопродуктивний компонент форми з управлінням обсягом даних. В тому числі збір даних, перевірка та стилі.
DatePicker	Для вибору або введення дати.
Input	Основний віджет для введення даних користувачем.
Radio	 Використовується для вибору одного стану з кількох параметрів.
Switch	 Представлення перемикання між двома станами або станом увімкнення-вимкнення.
Select	 Компонент для вибору значення з параметрів.
Transfer	

	Перенесення елементів між двома стовпцями інтуїтивно зрозумілим та ефективним способом.

Продовження табл. 2.1

Drawer	 Це панель, яка зазвичай накладається зверху сторінки. Містить набір інформації або дій. Оскільки користувач може взаємодіяти з компонентом, не виходячи з поточної сторінки, цілі можуть бути досягнуті ефективніше в одному контексті.
Modal	 Дозволяє взаємодіяти з додатком, не переходячи на нову сторінку та не перериваючи робочого процесу користувача, можна використовувати Modal для створення нового плаваючого шару над поточною сторінкою, щоб отримати відгуки користувачів або відобразити інформацію.
Message	 Відображення глобальних повідомлень як зворотний зв'язок у відповідь на операції користувача.

Продовження табл. 2.1

Notification	 Відображення повідомлення.
Popconfirm	 Просте і компактне діалогове вікно для підтвердження дії.
Spin	Відображення стану завантаження сторінки або розділу.

2.1.4 Axios

У даному проєкті для взаємодії із сервером було використано бібліотеку Axios. Вона має низку переваг, які вигідно виділяють її серед інших:

- Підтримка JSON за замовчуванням
- Назви функцій, які відповідають будь-яким методам HTTP
- Зручна обробка помилок [14]

При створені додатку, в якому потрібно отримувати дані з API, Axios - це простий спосіб це зробити. Практично будь-який динамічний веб - сайт, що відображає дані з різних джерел, потребує певного способу виконання HTTP-запитів, і Axios - один із самих простих та ефективних способів зробити це. [15]

Більшість програм залежить від ресурсів інших програм. Ці ресурси стають доступними за допомогою API. Сервери не виняток. Хоча

сервери зберігають власні відповідні дані, іноді їм потрібні дані з інших серверів. Щоб отримати доступ до даних, серверам потрібен спосіб спілкування з API залежних серверів. Для таких випадків Axios також є рішенням. [16]

2.2 Вимоги до інтерфейсу системи

Ключовими вимогами до створюваної системи є:

- відкритість, тобто вона повинна відповідати всім сучасним стандартам, підтримка веб-технологій, а також можливість додавання функціоналу як сторонніх розробників, так і напрацювань учнів;
- масштабованість;
- здатність працювати на різних пристроях, операційних системах, серверах;
- адаптивність;
- розширюваність, тобто легка можливість нарощування функціоналу
- локалізація, тобто підтримка національних вимог і стандартів в цій області
- легке керування змістом з боку клієнтської частини додатку [17]

РОЗДІЛ 3
РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ СЕРВІСУ “ХДУ24”

У зв’язку з поставленою метою були визначені наступні завдання:

- Доопрацювання компонентів
    - Особистий кабінет
    - Модуль опитувань
    - Модуль новин
- Доопрацювання адаптивності для мобільних пристроїв
- Адміністративна панель
    - Людські ресурси
        - Студенти
        - Академічні групи
        - Ролі
        - Особи
        - Працівники
        - Викладачі
    - Структура університету
        - Факультети
        - Освітні програми
        - Дисципліни
    - Розклад
        - Розклади занять
        - Навчальні заняття

3.1 Клієнт-Серверна архітектура

Архітектура клієнт/сервер - це обчислювальна модель, в якій кілька компонентів працюють у строго визначених ролях для спілкування. Сервер розміщує, надає та управляє більшістю ресурсів та

послуг, які отримує клієнт. [18] Цей тип архітектури спільних ресурсів має один або кілька клієнтських комп’ютерів, підключених до центрального сервера через мережу або підключення до Інтернету. [19]

Архітектура клієнт/сервер працює, коли клієнтський комп'ютер надсилає на сервер запит ресурсу або процесу через мережеве з'єднання, яке потім обробляється і доставляється клієнту. Комп'ютер-сервер може одночасно керувати кількома клієнтами, тоді як один клієнт може бути підключений одночасно до кількох серверів, кожен з яких надає різний набір послуг.

Ось приклад того, як працює зв'язок клієнт-сервер. При середньому використанні браузера для доступу до веб-сайту на стороні сервера користувач або клієнт вводить URL-адресу. DNS-сервер шукає IP-адресу веб-сервера і передає її браузеру. Браузер генерує запит HTTP або HTTPS, а сервер як виробник надсилає файли. Клієнт як споживач отримує їх, а потім, як правило, надсилає додаткові запити.

REST — це набір архітектурних обмежень, а не протокол або стандарт. Розробники API можуть реалізувати REST різними способами.

Коли запит клієнта здійснюється через API RESTful, він передає представлення стану ресурсу запитувачу або кінцевій точці. Ця інформація або подання надається в одному з кількох форматів через HTTP: JSON (нотація об’єктів Javascript), HTML, XLT, Python, PHP або звичайний текст. JSON — це найпопулярніший формат файлів, який використовується, оскільки, незважаючи на свою назву, він мовно-агностичний.

Слід мати на увазі ще дещо: заголовки та параметри також важливі в методах HTTP HTTP-запиту RESTful API, оскільки вони містять важливу інформацію про ідентифікатори щодо метаданих запиту, авторизації, єдиного ідентифікатора ресурсу (URI), кешування, файлів cookie та більше. Існують заголовки запитів та заголовки

відповідей, кожен з яких має свою інформацію про з'єднання HTTP та коди стану.

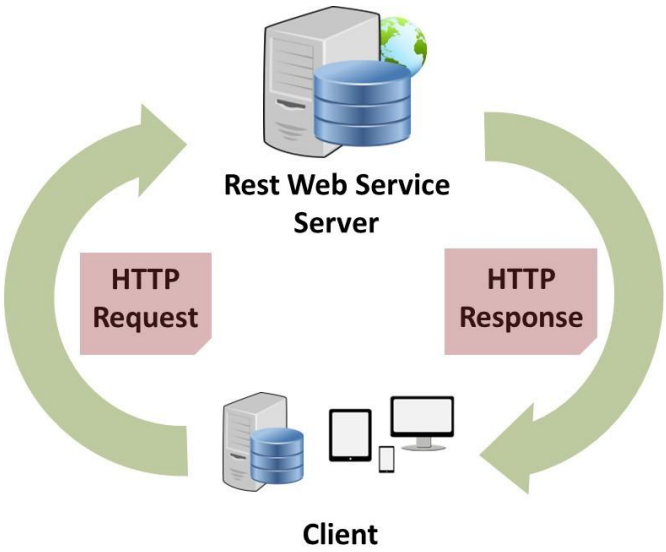


Рисунок 3.1 – Принцип роботи RESTful архітектури

Щоб API вважався RESTful, він повинен відповідати таким критеріям:

- Архітектура клієнт-сервер, що складається з клієнтів, серверів та ресурсів із запитами, керованими через HTTP.
- Зв'язок клієнт-сервер без громадянства, тобто інформація про клієнта не зберігається між запитами на отримання, і кожен запит є окремим і без зв'язку.
- Кешовані дані, які спрощують взаємодію клієнт-сервер.
- Єдиний інтерфейс між компонентами для передачі інформації у стандартній формі.
    - запитувані ресурси ідентифікуються та відокремлюються від подань, надісланих клієнту.
    - Клієнт може маніпулювати ресурсами за допомогою представлення, яке вони отримують, оскільки подання містить достатньо інформації для цього.

  - самоописні повідомлення, повернені клієнту, мають достатньо інформації, щоб описати, як клієнт повинен їх обробляти.
  - доступний гіпертекст/гіпермедіа, що означає, що після доступу до ресурсу клієнт повинен мати можливість використовувати гіперпосилання для пошуку всіх інших доступних наразі дій, які він може виконувати.
- Багаторівнева система, яка організовує кожен тип серверів (відповідальних за безпеку, балансування навантаження тощо), передбачає вилучення запитуваної інформації в ієрархії, невидимі для клієнта.
- Код на вимогу (необов’язково): можливість надсилати виконуваний код із сервера клієнту за запитом, розширюючи функціональність клієнта.

Хоча API REST має відповідати цим критеріям, він все ще вважається простішим у використанні, ніж такий прописаний протокол, як SOAP (простий протокол доступу до об’єктів), який має певні вимоги, такі як обмін повідомленнями XML, та вбудована безпека та відповідність транзакціям повільніше і важче.

На відміну від цього, REST — це набір рекомендацій, які можна впроваджувати за потреби, що робить REST API швидшими та легшими, з підвищеною масштабованістю — ідеально підходить для Інтернету речей (IoT) та розробки мобільних додатків.

API дають змогу вашому продукту чи службі взаємодіяти з іншими продуктами та послугами, не знаючи, як вони реалізовані. Це може спростити розробку додатків, заощадивши час та гроші. Коли ви розробляєте нові інструменти та продукти або керуєте існуючими, API надає вам гнучкість; спростити проектування, адміністрування та використання; та надають можливості для інновацій.

Інколи API вважають контрактами з документацією, що представляє угоду між сторонами: Якщо Сторона 1 надсилає віддалений запит, структурований певним чином, програмне забезпечення сторони 2 реагуватиме так.

Оскільки API спрощують, як розробники інтегрують нові компоненти додатків у існуючу архітектуру, вони допомагають співпрацювати будь-яким бізнес-процесам та інформаційним технологіям. Для того, щоб залишатися конкурентоспроможними, важливо підтримувати швидкий розвиток та впровадження інноваційних послуг. Розробка хмарних додатків - це ідентифікований спосіб збільшення швидкості розробки, і він спирається на підключення архітектури додатків через API.

Для взаємодії з API використовувалась документація побудована за допомогою технології Swagger. Це інструмент для автоматизованої генерації документації. Для кожного типу запитів існує їх опис, можливі параметри та формат відповіді на запит.

Рисунок 3.2 – Swagger-специфікації та формат даних, що повертаються

3.2 Ролі користувача

Реалізований вибір ролі користувача з боку клієнтської частини. Кожна роль має власні права до перегляду та взаємодії з тими чи іншими компонентами та сторінками.

Рисунок 3.3 – Приклад вибору ролі користувача

Кожна роль має власний набір дозволів на відвідування і перегляд певних компонентів на стороні клієнта і можливість виконання запитів до API.

```
<ProtectedRoute
  exact
  path="/management"
  component={Administration}
  allowedUserRoles={[
    EDUCATIONAL_PEDAGOGICAL_STAFF,
    DEAN,
    VICE_RECTOR,
    RECTOR,
    HEAD_DEPARTMENT,
    VICE_DEAN,
    DEANS_OFFICE_STUFF,
    ADMIN,
  ]}
/>
```

Рисунок 3.4 Приклад захищеного роуту

Дозволяє відкривати сторінки додатку тільки користувачам з одною із переданих ролей.

```
<ProtectedRender
  allowedUserRoles={[
    EDUCATIONAL_PEDAGOGICAL_STAFF,
    DEAN,
    VICE_RECTOR,
    RECTOR,
    HEAD_DEPARTMENT,
    VICE_DEAN,
    DEANS_OFFICE_STUFF,
    ADMIN,
  ]}
>
```

Рисунок 3.5 – Приклад захищеного компоненту

Дозволяє відображати різні компоненти на одній сторінці веб-додатку в залежності від ролі користувача.

Було перепрацьовано принцип збереження даних користувача, його ролі. Замість використання локального сховища браузеру, було використано сховище redux. Це дозволяє поєднати стан ролі користувача в даним момент часу з інтерфейсом, що в свою чергу вирішує велику кількість помилок пов’язаних з некоректним відображенням даних вже після зміни ролі, допуск користувача на сторінки, які не дозволені для його ролі. Також це доопрацювання робить можливим відкрити певні сторінки та частини сторінок для користувачів без авторизації. Після авторизації та вибору ролі всі необхідні маршрути стануть доступними. Це покращує користувацький досвід та дозволяє працювати з відкритими ресурсами сервісу будь-кому. [37]

Відкриті сторінки та компоненти для не авторизованих користувачів:

- Головна
  - Новини університету
- Компоненти у розробці
- Розробники
- Корисні ресурси
  - Посилання
  - Розклад дзвінків

За перевірку дозволу користувача відповідає компонент ProtectedRoute. Його параметрами є посилання, допустимі ролі, яким дозволено перегляд сторінки за посиланням та сам компонент сторінки, який буде повертатись за переданим посиланням. Якщо користувач має одну із ролей переданих у списку дозволених – повернеться запитувана сторінка, інакше – сторінка авторизації. Даний підхід є безпечним та ефективним рішенням для системи з великою кількістю ролей. Безпека обумовлена тим, що додаток отримує всі дані через запити до API і валідація користувача та його ролі завжди проводиться на боці сервера. Ніякі чутливі дані на захищених сторінках не зберігаються, тому цей функціонал більш націлений на позитивний користувацький досвід.

3.3 Адміністративна панель

Адміністративна панель – це компонент, який дає змогу взаємодіяти користувачу напряму з даними в базі даних. Було розроблено гнучку систему, яку легко налаштовувати та використовувати для будь-якого набору даних. [20] Основою компоненту являється компонент Table із бібліотеки компонентів Ant Design, яка була описана у розділі 2.1.3. За замовчуванням компонент підтримує відображення колекції структурованих даних, сортування, пошук, розподіл даних по сторінкам, фільтрація. [21]

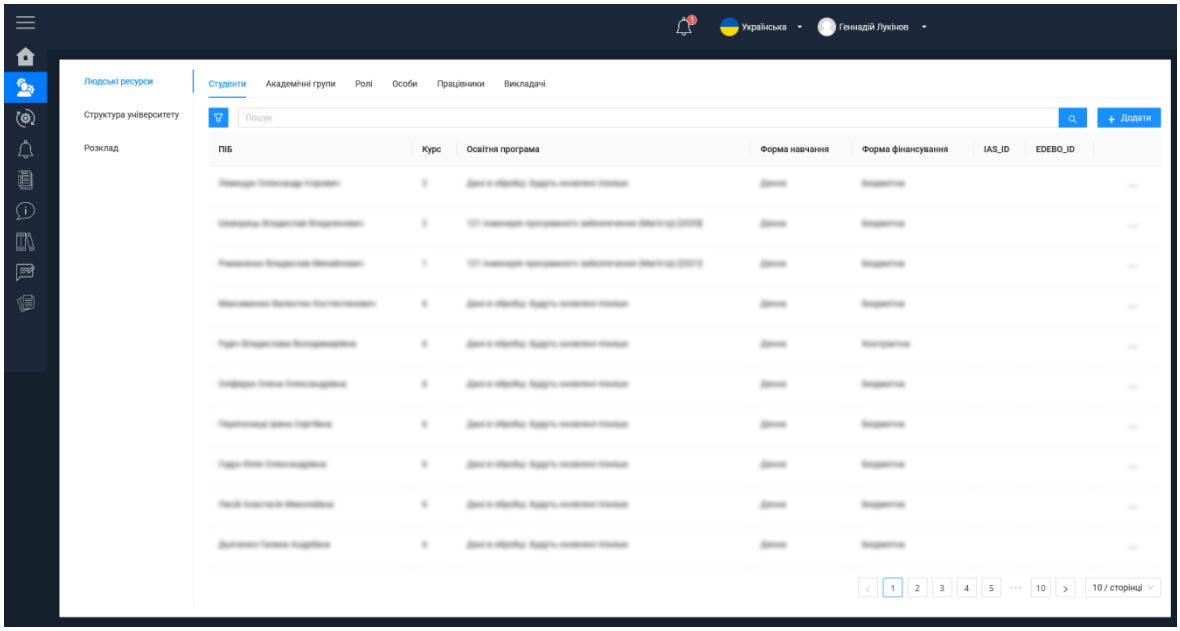


Рисунок 3.6 – Адміністративна панель

Компонент підтримує 4 основні функції керування даними (CRUD):

- Create (створення)
- Read (читання)
- Update (оновлення)
- Delete (видалення) [22]

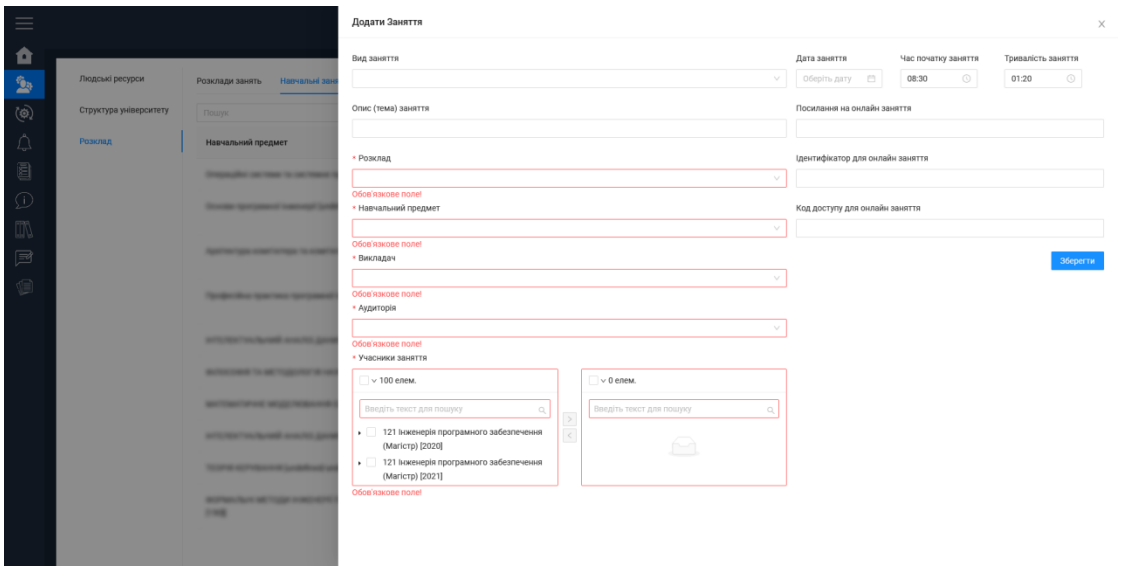


Рисунок 3.7 – Приклад створення сутностей та перевірки даних

Адміністративну панель можливо налаштувати для відображення лише доступних для певної ролі даних, та можливості редагування та видалення. Велика кількість даних розділяється на сторінки, кількість відображуваних елементів можливо налаштувати. Також підтримується гнучке налаштування фільтрів, пошук та перевірка даних при створенні будь-якої сутності в базі даних.

Компонент за замовченням підтримує фільтрацію для окремої колонки, але в ситуації, коли таблиця складається з великої кількості колонок і кожна з них має фільтруватися з'являється проблема у керуванні всіма параметрами. Для покращення користувацького досвіду та спрощення взаємодії користувача з додатком було створено універсальний компонент для фільтрування даних. Від самого початку було створено елементи вибору для більшості можливих даних, таких як вибір освітньої програми, аудиторії, дисципліни та ін. Вибір дати та часу також є можливими параметрами для фільтрації. Кожен із цих компонентів, які використовуються для створення та редагування будь-якої сутності також можуть бути використаними для фільтрування і в результаті формувати запит до серверу з усіма параметрами, які повертають компоненти вибору. Універсальність полягає у можливості передачі лише списку типів даних, які фільтруються та налаштування вихідного об'єкту для запиту до серверу.

3.3.2 Взаємодія з API

Взаємодія з API - це звичайний випадок при розробці сучасних веб-додатків, тому важливо мати спосіб повторного використання загальних функцій у додатку React.

Поширене повторне використання коду в React можна реалізувати за допомогою компонентів вищого порядку та хуків.

Компонент вищого порядку - це функція, яка приймає компонент як аргумент і повертає цей компонент. [23] Це дає нам можливість виконувати деякі функції перед поверненням компонента. Ця функція може бути загальною та багаторазовою, у нашому випадку дана функція додає функціонал для взаємодії з API, що сильно спрощує подальшу розробку та налагодження. У нашому випадку функція приймає будь-який компонент, всередині функції описується додатковий функціонал (функції оновлення даних, взаємодія з сервером) та повертає початковий компонент, в який у вигляді параметрів передає створені функції. Такий підхід дає змогу контролювати роботу багатьох компонентів з одного місця, покращує підтримку веб-додатку. [24]

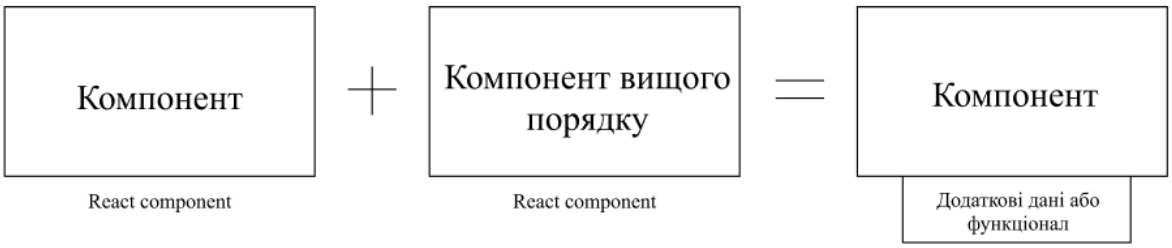


Рисунок 3.8 – Принцип роботи компоненту вищого порядку

Підтримувані запити:

1. GET - Отримання цільового ресурсу;
2. POST - Цільовий ресурс обробляє представлення, укладене у запиті;
3. PUT - Створення або заміна стану цільового ресурсу на стан, визначений представленням, включеним у запит;
4. DELETE - Видалення стану цільового ресурсу.

В залежності від статусу відповіді сервера інтерфейс коректно реагує і відображає помилки у вигляді сповіщень, які не переривають роботу користувача.

Як правило, сповіщення можна використовувати в таких випадках:

- Повідомлення зі складним вмістом.
- Сповіщення, що надає зворотний зв'язок на основі взаємодії користувача. Або він може показувати деякі подробиці про

майбутні кроки, які, можливо, доведеться виконати користувачеві.

- Повідомлення, які надсилаються програмою.

3.3.3 Компонент вибору

Для кожного виду сутностей, які використовуються при створенні та редагуванні, було розроблено компоненти вибору. Підтримується вибір, як статичних даних так і отриманих асинхронно запитом до API. Має підтримку пошуку. Якщо компонент взаємодіє з сервером, стан запиту відображається анімацією завантаження, на час завантаження компонент блокується для взаємодії для протидії виникнення неочікуваних помилок.

```
<Select
  showSearch
  notFoundContent={loading && <Spin size="small" />}
  onFocus={() => updateData()}
  onSearch={debounce(onSearch, DEBOUNCE_DELAY)}
  allowClear={true}
  filterOption={false}
  {...rest}
>
  {options.map(option => (
    <Option value={option?.id} key={option?.id}>
      {option?.name}
    </Option>
  ))}
</Select>
```

Рисунок 3.9 – Приклад універсального компоненту вибору

3.4 Підтримка багатомовності

Важливим аспектом будь-якого програмного продукту є підтримка багатомовності. Це покращує користувацький досвід, збільшує кількість можливих користувачів. Окрім цього винесення тексту всіх елементів додатку в окремий конфігураційний файл спрощує подальшу розробку та підтримку програмного продукту. [25]

Слід звернути увагу на важливі аспекти перекладу й налаштування багатомовності додатку.

Для підтримки багатомовності було впроваджено бібліотеку i18next, що має підтримку бібліотеки React. Це спрощую ініціалізацію та впровадження багатомовності. Переклад зберігається у форматі JSON.

Особливості i18next:

- Розділення перекладів на кілька файлів. Потрібні лише завантажувальні переклади [28]
- Існують плагіни для виявлення мов для більшості середовищ (браузер, рідний, сервер). Це дозволяє встановлювати пріоритет місця виявлення та навіть дозволяє кешувати задані мови над запитами / відвідуваннями [29]
- Існує безліч плагінів для завантаження перекладу з сервера, файлової системи. Можуть забезпечити додатковий рівень для локального кешування, наприклад. у локальному сховищі
- Підтримка об’єктів і масивів
- Повний контроль над керуванням збереженими перекладами

```
i18n
  .use(Backend)
  .use(LanguageDetector)
  .use(initReactI18next)
  .init({
    fallbackLng: 'uk',
    detection: {
      order: ['cookie'],
      cache: ['cookie'],
    },
    interpolation: {
      escapeValue: false,
    },
  });
```

Рисунок 3.10 – Ініціалізація бібліотеки i18next

Для використання перекладених рядків всередині додатку викликається функція useTranslation, яка повертає функцію t та екземпляр i18n. Таким чином стає можливим отримувати рядки саме тієї мови, яка обрана всередині додатку в даний момент часу і змінювати загальний стан додатку. Зміна мови відбувається без перезавантаження

сторінки, перемальовуються лише ті елементи, які змінилися при зміні мови. [30]

```
"Login": {
  "header": "Select user",
  "student": "Student",
  "employee": "Employee",

  "label": {
    "login": "Login",
    "password": "Password"
  },

  "error": {
    "login": "Enter login!",
    "password": "Enter the password!"
  }
},
```

```
"Login": {
  "header": "Оберіть користувача",
  "student": "Студент",
  "employee": "Співробітник",

  "label": {
    "login": "Логін",
    "password": "Пароль"
  },

  "error": {
    "login": "Введіть логін!",
    "password": "Введіть пароль!"
  }
},
```

Рисунок 3.11 – Приклад конфігураційного файлу з перекладом

3.5 Розробка компонентів веб-додатку

3.5.1 Новини

Компонент реалізовано синтаксичного аналізатору RSS-стрічки, яку генерує головний сайт університету. [40]

RSS - це веб-канал, що дозволяє користувачам та програмам отримувати доступ до оновлень веб-сайтів у стандартизованому, читаному комп'ютером форматі. Для отримання та обробки даних було створено Redux модуль, що містить функціонал перетворення RSS-стрічки у формат JSON для відображення записів. [31]

Кожен запис може мати поля:

- Заголовок
- Посилання на новину
- Дата публікації
- Автор
- Наповнення

- Категорії

Зображення отримуються із наповнення статті за допомогою регулярного виразу. [32]

Зазвичай такі шаблони використовуються алгоритмами пошуку рядків для операцій "знайти" або "знайти і замінити" над рядками або для перевірки введення. Це техніка, розроблена в теоретичній інформатиці та офіційній теорії мови. [33-34] Підрядок який має посилання на зображення знаходиться в тегі img, для його пошуку використано регулярний вираз `<img[^>]+src="([^">]+).` [35]

У результаті пошуку повертається масив, який складається з посилань за зображення.

Так як зображення отримуються асинхронно та мають різну швидкість завантаження, що обумовлено різним розміром зображень та швидкістю інтернету, було доопрацьовано компонент Image для очікування повного завантаження зображення.

Відкладене завантаження – це стратегія визначення ідентифікації ресурсів як неблокувальних (некритичних) та завантаження їх лише за потреби. Це спосіб скоротити довжину критичного шляху візуалізації, що означає скорочення часу завантаження сторінки. До моменту завантаження відображається анімація завантаження, якщо зображеня присутнє в даному записі.

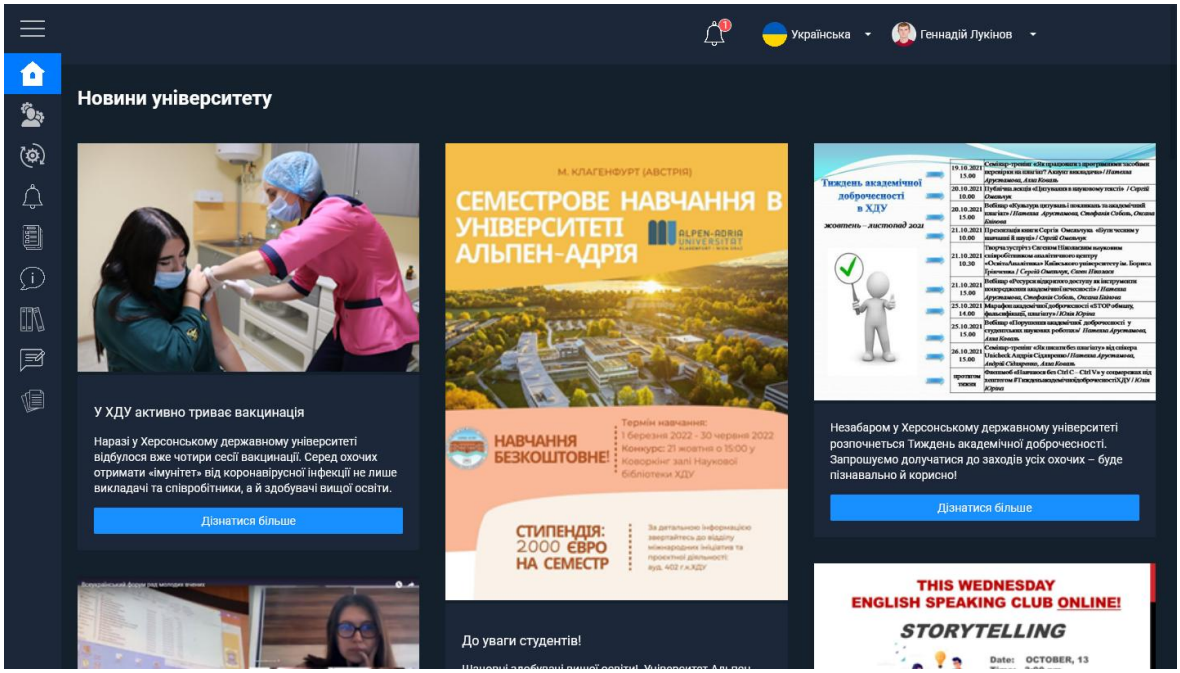


Рисунок 3.12 – Компонент новин університету

3.5.2 Контактні дані користувача

Даний компонент дає можливість користувачу створювати, редагувати та видаляти контактні дані. Все дії відбуваються асинхронно, без перезавантаження сторінки.

Можливі типи контактів:

- Телефон
- Електронна пошта
- Адреса
- Telegram
- Facebook
- Twitter
- Instagram
- Github
- Інше посилання

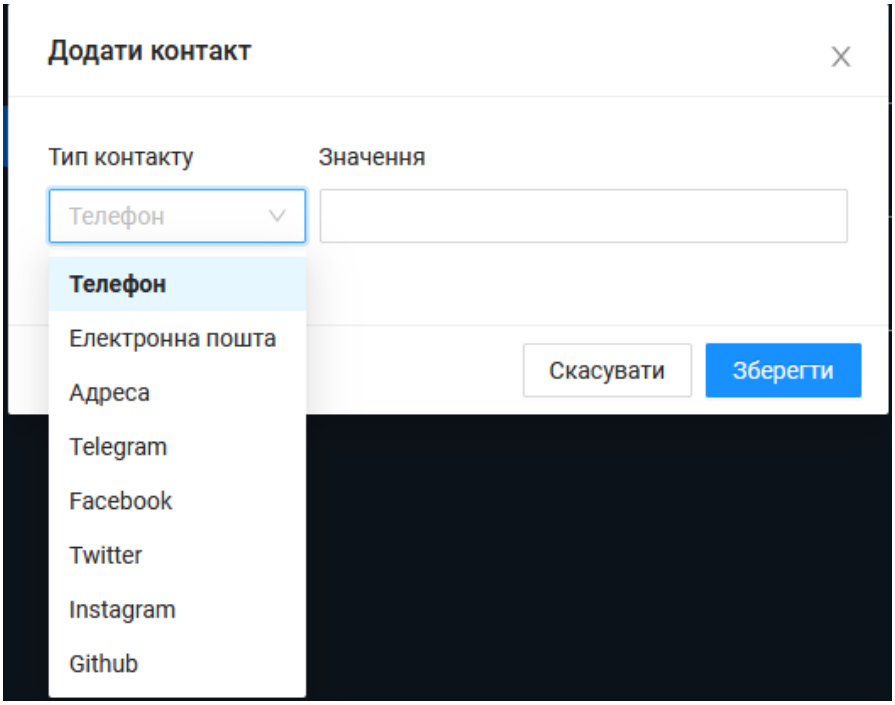


Рисунок 3.13 Приклад створення запису контакту

3.5.3 Опитування

Було доопрацьовано компонент опитувань з боку клієнтської частини веб-додатку. Кожен тип поля являється окремим компонентом, який можливо гнучко налаштувати. Опитування підтримує коротко тексту, довгого тексту, вибір одного варіанту, вибір багатьох варіантів, дата, час.

Можливі параметри:

- Довжина рядка, числа чи масиву
- Максимальна та мінімальна довжина рядка, числа чи масиву
- Патерн регулярного виразу для валідації рядка
- Обов'язкове поле
- Тип рядка (рядок, число, логічний, посилання, пошта)

Отримання опитувань, їх обробка, валідація та збереження відбувається асинхронно. Задаються правила для кожного типу поля на боці і клієнта, і сервера.

ВИСНОВКИ

Метою кваліфікаційної роботи є розробка інтерфейсу користувача веб-додатку “ХДУ24”.

У ході виконання роботи були виконані наступні завдання:

- Проаналізовано існуючі систему управління ВНЗ
- Визначено основні принципи та особливості проєктування та розробки користувацького інтерфейсу.
- Розглянуто інструменти для розробки користувацького інтерфейсу.
- Спроєктовано інтерфейс користувача.
- Розроблено та протестовано клієнтську частину веб-додатку.

Велику частину роботи було приділено аналізу технологій для розробки додатків та сервісів. Було проаналізовано сучасний підхід до розробки веб-додатків, проводився детальний аналіз технологій розробки інтерфейсу користувача.

Розроблено компоненти для взаємодії з прикладним програмним інтерфейсом (API), зокрема систему для адміністрування, перегляду, редагування та видалення будь-яких сутностей БД, пошуку даних та їх фільтрування за доступними параметрами.

Розроблена система ролей з боку клієнтської частини додатку. Кожна роль має власний набір дозволів на відвідування і перегляд певних компонентів на стороні клієнта і можливість виконання запитів до API.

Розроблена система задовольняє вимоги сформовані на етапі постановки задачі та може використовуватись за прямим призначенням.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Difference Between UX And UI [Електронний ресурс] – Режим доступу до ресурсу: https://www.mageplaza.com/blog/what-is-user-interface.html#what-is-the-user-interface-ui.
2. Exploring the UI Universe: Different Types of UI [Електронний ресурс] – Режим доступу до ресурсу: https://www.altia.com/2014/09/22/different-types-of-ui/.
3. User Experience (UX) Design [Електронний ресурс] – Режим доступу до ресурсу: https://www.interaction-design.org/literature/topics/ux-design.
4. The 6 key principles of UI design [Електронний ресурс] – Режим доступу до ресурсу: https://maze.co/collections/ux-ui-design/ui-design-principles/.
5. What is React: Definition, Why ReactJS, its Features and Installation [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs.
6. React (JavaScript library) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/React_(JavaScript_library).
7. Redux [Електронний ресурс] – Режим доступу до ресурсу: https://redux.js.org.
8. Redux. An Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.smashingmagazine.com/2016/06/an-introduction-to-redux/
9. Ant Design of React [Електронний ресурс] – Режим доступу до ресурсу: https://ant.design/docs/react/introduce.
10. Understanding the Ant Design System — a UI Design for Enterprises [Електронний ресурс] – Режим доступу до ресурсу: https://betterprogramming.pub/understanding-the-ant-design-system-a-ui-design-for-enterprises-39afdb188b06

11. Ant Design GitHub Repository — a UI Design for Enterprises [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ant-design/ant-design/
12. Npm trends for Ant Design [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmtrends.com/antd-vs-material-ui-vs-react-bootstrap
13. Ant Design Components Overview [Електронний ресурс] – Режим доступу до ресурсу: https://ant.design/components/overview/
14. About Axios [Електронний ресурс] – Режим доступу до ресурсу: https://axios-http.com/docs/intro
15. Client-Server Definition [Електронний ресурс] – Режим доступу до ресурсу: https://www.omnisci.com/technical-glossary/client-server
16. Client/Server Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://www.techopedia.com/definition/438/clientserver-architecture
17. What is AXIOS and How To Use it [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/veewebcode/what-is-axios-and-how-to-use-it-4an1
18. Node.js Network Requests using Axios [Електронний ресурс] – Режим доступу до ресурсу: https://www.section.io/engineering-education/nodejs-network-requests-using-axios/
19. User interface requirements [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/docs/en/db2/9.7?topic=considerations-user-interface-requirements
20. Why building an admin panel should be in your first sprint [Електронний ресурс] – Режим доступу до ресурсу: https://www.bytelion.com/why-building-an-admin-panel-should-be-in-your-first-sprint/

21. Table component [Електронний ресурс] – Режим доступу до ресурсу: https://ant.design/components/table/
22. Create, read, update and delete [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Create,_read,_update_and_delete
23. Higher-Order Components In React [Електронний ресурс] – Режим доступу до ресурсу: https://www.smashingmagazine.com/2020/06/higher-order-components-react/
24. Simple and reusable React Context API example (HOC) [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@niwaa/simple-and-reusable-react-context-api-example-hoc-e1e50c0390ec
25. Benefits and Best Practices of a Multilingual Website [Електронний ресурс] – Режим доступу до ресурсу: https://www.inzonedesign.com/blog/benefits-best-practices-multilingual-website/
26. Benefits of Multilingualism in Education [Електронний ресурс] – Режим доступу до ресурсу: https://files.eric.ed.gov/fulltext/EJ1053855.pdf
27. The Importance of Being Multilingual [Електронний ресурс] – Режим доступу до ресурсу: https://thedickinsonian.com/opinion/2018/02/01/the-importance-of-being-multilingual/
28. I18next documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.i18next.com/
29. Building in JavaScript with Internationalization (I18N) in Mind [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/adobetech/building-in-javascript-with-internationalization-i18n-in-mind-815b6bf2c25e

30. What Is I18n: A Simple Definition of Internationalization [Електронний ресурс] – Режим доступу до ресурсу: https://phrase.com/blog/posts/i18n-a-simple-definition/
31. Rich site summary [Електронний ресурс] – Режим доступу до ресурсу: https://www.britannica.com/technology/RSS
32. Regex [Електронний ресурс] – Режим доступу до ресурсу: https://www.computerhope.com/jargon/r/regex.htm
33. Regular Expression Tutorial - Learn How to Use Regular Expressions [Електронний ресурс] – Режим доступу до ресурсу: https://web.archive.org/web/20161101212501/http://www.regular-expressions.info/tutorial.html
34. The Oxford Handbook of Computational Linguistics, 2009. – 785 с.
35. Finite Automata, 2004. – 98-100 с.
36. Patterns of Enterprise Application Architecture / Martin Fowler – 200-214 с.
37. Проєктування та розроблення сервісної архітектури управління бізнес-процесами університету. Сервіс «Відділ кадрів» / Сенчишен, Д. О. 2020. 37-43 с.
38. Система дистанційного навчання «KSU Online». — URL: http: //ksuonline.kspu.edu/
39. Співаковський О., Вінник М., Тарасіч Ю. Побудова ІКТ інфраструктури ВНЗ: проблеми та шляхи вирішення // Інформаційні технології і засоби навчання. — 2014. — 39, вип. 1. — С. 90—115.
40. Співаковський О. В. Web-портал Херсонського державного університету. — 2013. — URL: http://kspu.edu/.

ДОДАТКИ

Додаток А

Компонент новин університету:

```
const RSS_FEED =
'https://www.kspu.edu/RssAggregator.ashx?Name=News&lang=uk';

const defaultState = {
  data: [],
  loading: false,
};

export default (state = defaultState, { type, payload }) => {
  switch (type) {
    case FETCH_NEWS_START:
      return { ...state, loading: true };
    case FETCH_NEWS_SUCCESS:
      return {
        ...state,
        loading: false,
        data: payload,
      };
    case FETCH_NEWS_ERROR:
      return {
        ...state,
        loading: false,
        error: `Error: ${payload.error}`,
      };
    default:
      return state;
  }
};

/*
 * Actions
 */

export const fetchNews =
  (limit = 15) =>
  async dispatch => {
    dispatch({ type: FETCH_NEWS_START });
```

```
  try {
    let { items } = await rssParser.parseURL(RSS_FEED);

    items = items
      .map(item => {
        let img = item.content.match(`<img[^>]+src="([^">]+)"`);
        item.img = Array.isArray(img)
          ? 'https://www.kspu.edu' + img[1]
          : null;
        item.title = item?.title.replace(/&mdash;/g, '—');
        return item;
      })
      .slice(0, limit);

    dispatch({
      type: FETCH_NEWS_SUCCESS,
      payload: items,
    });
  } catch (error) {
    dispatch({
      type: FETCH_NEWS_ERROR,
      payload: { error: error.message || error },
    });
    openNotification(
      'error',
      'Сталася помилка',
      'Помилка при отриманні новин',
    );
  }
};
```

Додаток Б

Додаток 1

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, _Лукінов Геннадій Геннадійович_, учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

– надавати згоду на:

- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

07.09.2020		Лукінов Г.Г.
(дата)	(підпис)	(ім'я, прізвище)