КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття освітнього рівня бакалавра
за спеціальністю 121 Інженерія програмного забезпечення
на тему:

РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБЗАСТОСУНКУ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ОСВІТИ

Виконав студент 4-го курсу
Микола ЧИКІВЧУК _____
(підпис)

Науковий керівник:
асистент, кандидат фізико-математичних наук
Костянтин ЖЕРЕБ _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
« 29 » травня 2023 р.,
протокол № 11
Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 40 сторінок, 18 рисунків, 11 фрагментів коду, 12 посилань.

NEXT.JS, REACT, STORYBOOK, TYPESCRIPT, ВІДГУКИ ВІД СТУДЕНТІВ, ВЕБЗАСТОСУНОК, ОЦІНЮВАННЯ ЯКОСТІ ОСВІТИ

Об’єктом роботи є клієнтська частина вебзастосунку який реалізує систему оцінювання якості освіти, систему модерації даних про заклади вищої освіти та систему авторизації студентів.

Метою роботи є проектування та розробка вебзастосунку для псевдонімного або анонімного оцінювання якості освіти з відкритою для редагування базою даних, який позбавлений недоліків існуючих аналогів та має розширений функціонал.

Інструменти розробки: редактор вихідного коду Visual Studio Code, мова програмування TypeScript, фронтенд фреймворки React та Next.js, середовище для тестування компонентів Storybook, бібліотека для інтернаціоналізації next-translate, мова стилів CSS, метамова SCSS, CSS фреймворк Bootstrap.

Результати роботи: спроектовано систему оцінювання; систему зважування оцінок користувачів та клієнтську частину вебзастосунку, яка використовує зазначену систему і є основою для подальшої розробки та доповнення.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

DOM – Document Object Model, об'єктна модель документа;

SEO – Search Engine Optimization, оптимізація для пошукових систем;

SSG – Static Site Generation, статична генерація сайту;

SSR – Server Side Rendering, рендеринг на стороні сервера;

Бекенд – backend, серверна частина застосунку;

Фронтенд – frontend, клієнтська частина застосунку.

ВСТУП

Оцінка сучасного стану об’єкта роботи. Існують різні способи оцінити якість освіти та викладання. Багато університетів проводять внутрішні анонімні опитування. Подібні опитування дають можливість адміністрації університетів проаналізувати якість роботи викладачів та якість супутніх навчальному процесу елементів: ремонт в аудиторіях, якість харчування, опалення в будівлях університету тощо. Проте результати даних опитувань часто є закритими, тому студенти не можуть орієнтуватись на них при виборі університету, кафедри, спеціальності чи викладачів (коли це можливо).

В університетах Сполучених Штатів та країн, які мають подібну систему вищої освіти, студенти кожен семестр обирають собі навчальні курси та викладачів, які їх викладають, що робить наявність публічних відгуків ще більш важливою.

Існує певна кількість вебсайтів які надають студентам можливість залишати анонімні відгуки та оцінювати якість викладання. Найбільший з них – це RateMyProfessors.com, проте працює він тільки в Сполучених Штатах, Канаді та Великій Британії [1].

Актуальність роботи. Освіта є однією з найбільш соціально важливих сфер діяльності людства. Відповідно покращення якості освіти є дуже актуальним і важливим завданням. Підтримка зворотного зв’язку зі студентами та аналіз отриманих даних дозволяє викладачам, адміністрації навчальних закладів та чиновникам, які є відповідальними за освіту, виявляти та усувати проблеми. Відкритість відгуків та оцінок від студентів дає можливість іншим студентам та абітурієнтам можливість обирати заклад освіти, ґрунтуючись на цій інформації, що створює здорову конкуренцію між закладами освіти та додатковий стимул для розвитку.

Всі існуючі вебсайти, які реалізовують описаний функціонал, мають спільні недоліки, які будуть детальніше описані далі в даній роботі. Також більшість з них доступні в обмеженій кількості регіонів.

Мета та завдання роботи. Метою роботи є проектування та розробка вебзастосунку для псевдонімного або анонімного оцінювання якості освіти з відкритою для редагування базою даних, який позбавлений недоліків існуючих аналогів та має розширений функціонал.

Для досягнення цієї мети поставлено такі задачі:

- розглянути недоліки існуючих аналогів;
- описати способи вирішення недоліків, які необхідно застосувати при розробці нашого застосунку;
- спроектувати систему оцінювання;
- розробити клієнтську частину вебзастосунку.

Об'єкт, методи й засоби розроблення. Об’єктом роботи є клієнтська частина вебзастосунку, який реалізує систему оцінювання якості освіти, систему модерації даних про заклади вищої освіти та систему авторизації студентів.

Інструментами розробки є редактор вихідного коду Visual Studio Code, мова програмування TypeScript, фронтенд фреймворки React та Next.js, середовище для тестування компонентів Storybook, бібліотека для інтернаціоналізації next-translate, мова стилів CSS, метамова SCSS, CSS фреймворк Bootstrap.

Взаємозв’язок з іншими роботами. Розробка застосунку велася в команді. Дана робота складається з розробки клієнтської частини застосунку та проектування всього застосунку на рівні ідеї. Серверна частина проектувалась та розроблялась переважно моїми колегами по проекту та коротко описана в розділі 4 даної роботи.

Можливі сфери застосування. Застосунок частиною якого є ця робота, може застосовуватись:

- абітурієнтами при виборі навчального закладу або факультету;
- студенти для обміну досвідом та при виборі опціональних курсів або викладачів;
- викладачами або адміністраціями навчальних закладів для підтримки зворотного зв’язку зі студентами та його аналізу.

1 ОГЛЯД ЗАВДАННЯ ТА АНАЛОГІВ

1.1 Сутності

Ми проектуємо сайт, який буде працювати в різних країнах та регіонах, тому доцільним було б оцінювати сутності, які присутні в усіх навчальних закладах, незалежно від типу закладу, країни та системи освіти.

До прикладу, “спеціальність”, “навчальна програма”, “відділ кафедри”, є сутностями, які присутні в Київському Національному Університеті, але можуть бути відсутності в приватному університеті або університеті іншої країни.

Найбільша сутність, яку ми можемо оцінювати – це навчальний заклад (університет, коледж, технікум або навчальний заклад іншого типу).

Більшість навчальних закладів ділиться на відділи, в університетах України – це кафедри та інститути.

У відділі викладаються навчальні курси**,** тобто навчальні предмети.

Курс може викладатись одним або декількома викладачами.

Отже, сутності які ми будемо оцінювати на нашому вебсайті – це:

- навчальний заклад;
- відділи;
- курси;
- викладачі.

Також на сайті повинна бути сутність студента. Студент реєструється під псевдонімним ім’ям (або під реальним, за бажанням).

1.2 Користувачі вебзастосунку

Користувачем сайту може бути студент, викладач або адміністрація навчального закладу чи відділу.

Користувач створює обліковий запис, який прив'язується до відповідної сутності, яку він контролює.

Студент може залишати відгуки під іншими сутностями та оцінювати їх.

Якщо реєструється викладач, йому необхідно пройти авторизацію, після чого його обліковий запис буде прив’язано до його сторінки (детальніше:

підрозділ 1.4) та право до редагування даних своєї сутності перейде до нього. Також викладач має можливість коментувати відгуки на свої курси.

Обліковий запис адміністрації відділу отримує право на ексклюзивне редагування своєї сутності та сутностей викладачів, які в ньому працюють, право відповідати на відгуки від імені відділу та робити публікації.

Аналогічно з адміністрацією університету.

1.3 Сторінки

Сутностям відповідають сторінки на вебсайті.

На сторінках навчальних закладів та відділів відображається базова інформація, студенти можуть залишити відгук та оцінити їх за певними характеристиками.

Сторінки викладачів, своєю чергою, містять список курсів які вони ведуть, Студенти можуть оцінити курс викладача на його сторінці.

1.4 Модерація даних

Аби надати можливість студентам обмінюватись корисною інформацією та відгуками, у випадку якщо адміністрація університету чи відділу не зацікавлена у використанні застосунку, необхідно створити систему керування даними, яка може модеруватись кожним користувачем застосунку. Еталонним прикладом реалізації подібної системи є Wikipedia та інші вебсайти, які використовують wiki-рушій.

Користувач повинен мати можливість:

- створювати сторінку університету або відділу, якщо вона відсутня в базі даних;
- створювати сторінку викладача (в деяких країнах необхідно при цьому отримати явну згоду викладача [2]);
- редагувати сторінку, додавати додаткову інформацію або видаляти хибну чи застарілу.

Для зменшення кількості вандалізму та порушення правил платформи необхідно надавати право редагування лише зареєстрованим користувачам та забирати це право у випадку серії порушень.

Аби спростити відновлення даних після актів вандалізму необхідно також реалізувати систему контролю версій з можливістю відновлення попередніх версій сторінки.

Також було б доцільним створити ієрархію модераторів, які б слідкували за дотриманням правил та запобігали актам вандалізму, проте розгляд цієї теми виходить за рамки даної роботи.

Якщо викладач або адміністрація навчального закладу чи відділу виявляють бажання зареєструватись на сайті, вони повинні мати можливість отримати повний контроль за своїми сторінками, які до цього модерувались виключно студентами.

1.5 Підтвердження студентів

Вебсайти з наповненням, яке створюють користувачі, часто страждають від спаму та автоматично згенерованого зловмисного контенту. Особливо критичною є ця проблема на вебсайтах з рейтингами та відгуками.

Зловмисники можуть створювати декілька облікових записів або створити бот-мережу, яка буде вносити викривлення в рейтингову систему, оцінюючи відповідну сутність бажаними для зловмисника оцінками.

Є багато способів розв'язувати цю проблему або хоча б зменшити її вплив. Розглянемо декілька з них:

- реєстрація по номеру телефону;
- перевірка студентського квитка;
- підтвердження за університетською електронною поштою.

Реєстрація за номером телефону може значно ускладнити масове створення облікових записів для накручування оцінок, проте не повністю захищає від них. Також надсилання SMS повідомлень є платним, і при великій

кількості користувачів може бути доволі фінансово затратним. Також реєстрація по номеру телефону ніяк не підтверджує статус студенту.

Перевірка студентського квитка є напевно найбільш достовірним способом ідентифікувати студента, проте вона пов’язана з великою кількістю складностей. В різних країнах різні формати студентських квитків, що унеможливлює автоматичну перевірку, тому необхідно мати спеціального оператора який буде її здійснювати. Студентський квиток може містити інформацію мовою, яка не відома оператору. Також цей спосіб вимагає додаткової уваги до безпеки та відповідального ставлення до надісланих документів. Велика кількість студентів не будуть мати бажання надсилати свій квиток через недовіру до сервісу, що негативно вплине на кількість підтверджених студентів.

Підтвердження студентів по університетській електронній пошті – є найкращим методом з доступних, оскільки:

- існують відкриті бази зі списком доменів університетських пошт;
- природа сайту, який модерують користувачі дозволяє швидко оновлювати та додавати відсутні домени;
- цей метод не вимагає додаткових витрат;
- дозволяє підтвердити навчання користувача в університеті.

Обов’язкова необхідність підтверджувати обліковий запис може сильно зменшити кількість студентів, які будуть готові використовувати сервіс. Як компроміс, ми можемо рахувати середню оцінку сутності, зважуючи оцінки студентів. Студент з підтвердженою університетською поштою підвищує вагу своєї оцінки.

1.6 Рейтинг студентів

Для того, щоб мотивувати студентів писати корисні відгуки, додавати та корегувати інформацію на сторінках та робити інші внески, демотивувати спам та вандалізм – ми можемо реалізувати систему рейтингу.

За приклад ми можемо розглянути систему рейтингу на вебсайті Reddit. Рейтинг на цьому вебсайті називається кармою. Користувач може створювати дописи, якщо його дописи подобаються іншим користувачам вони можуть їх апвоутити (upvote), якщо не подобаються, то – даунвоутити (downvote). Кількість апвоутів і даунвоутів впливає на те, наскільки допис високо відображається в стрічці дописів. Користувач отримує карму, якщо його дописи отримують апвоути та втрачає її, якщо його дописи отримують даунвоути.

Подібну систему можна реалізувати в нашому вебзастосунку. Користувачі можуть оцінювати чи є відгук корисним, чи ні. Вплив оцінки студента на загальну оцінку ми можемо зважувати відповідно до його карми.

Вплив рейтингу ми розглянемо детальніше в розділі 2 цієї роботи.

1.7 Огляд аналогів

Розглянемо позитивні та негативні сторони існуючих вебсайтів, які реалізують подібний до нашого функціонал.

Розглянемо вебсайт RateMyProfessors, оскільки він є найбільшим серед аналогів та вебсайт Studzona, оскільки він є найбільшим з небагатьох, які можливо використовувати в нашому регіоні [3].

1.7.1 RateMyProfessors

Переваги:

- Реалізує найбільшу кількість функціонала серед аналогів;
- Простіший та зручніший для використання ніж інші аналоги;
- Користується популярністю серед студентів;
- Містить велику кількість даних.

Недоліки:

- Доступний в обмеженій кількості регіонів (Сполучені Штати, Канада та Велика Британія);
- Тільки анонімні відгуки. Немає можливості оцінити степінь компетентності рецензента;

– Відсутність зважених оцінок. Оцінка студента, який пише корисні відгуки має таку ж вагу, як і студент який займається вандалізмом;
– Відсутність системи підтвердження студентів. Студенти можуть робити фейкові акаунти або бот-мережі та спотворювати рейтинг.

1.7.2 Studzona

Переваги:

– Відсутність прив’язки до регіону. Технічно можливий для використання в різних країнах.

Недоліки:

– Застарілі підходи до вебдизайну;
– Застарілі вебтехнології;
– Активна розробка не ведеться;
– Низький рівень активності користувачів на сайті в останні роки;
– Низький рівень наповненості та актуальності даних на сайті;
– Переускладнена система оцінок.

2 ПРОЕКТУВАННЯ СИСТЕМИ ОЦІНЮВАННЯ

2.1 Відгуки

Студент може залишати відгук на сторінках сутностей. Відгук містить текстову частину та оцінки відповідних характеристик. За бажанням користувача текстова частина може бути пустою та містити тільки оцінку.

Бажано, щоб користувач писав відгук використовуючи псевдонім. Тоді інші користувачі матимуть можливість оцінювати авторитетність рецензента та зможуть ділитись своїми псевдонімами зі знайомими.

Проте повинна бути опціональна можливість залишити анонімний відгук, тому що частина студентів буде реєструватись під реальними іменами, а частина захоче більшої анонімності ніж може надати псевдонімна система. Ім’я користувача та точний рейтинг в анонімних відгуках приховані, проте приблизний діапазон рейтингу повинен залишатись публічним.

2.2 Кількість характеристик

Для кожної сутності можна придумати безліч характеристик, які можна оцінювати. Проте чим більша кількість характеристик – тим більше це ускладнює процес оцінки для студентів, що може зменшити кількість залишених відгуків та відлякати частину користувацької бази. Тому при проектуванні характеристик необхідно прагнути мінімалізму.

Під час роботи вебсайту завжди є можливість додати нову характеристику, якщо буде необхідність. Видалення характеристик несе за собою втрату інформації на яку студенти витратили свій час.

2.3 Характеристики

- Студент виставляє загальну оцінку та складність курсу;
- Оцінка викладача формується з оцінок курсів, які він викладає;
- Середні оцінки курсів також відображаються на сторінках навчального закладу та відділу;

– Навчальний заклад та відділ оцінюються лише загальними оцінками (з причин описаних в підрозділі 2.2).

2.4 Зважування оцінок

Формули та коефіцієнти для зважування оцінок повинні підбиратись експериментально. Маючи активних користувачів та наповнену базу даних, можна буде аналізувати їх вплив на результуючі оцінки. Проте ми можемо спробувати підібрати певні початкові формули.

Розглянемо можливу формулу для підрахунку середніх зважених загальної оцінки та оцінки складності курсу:

$$R = \frac{\sum_{i=0}^{n} r_i w_i}{\sum_{i=0}^{n} w_i}$$

де R – середня зважена оцінка, n – кількість відгуків, r – оцінка користувача, w – вага студента

Як зазначалось в розділі 1, вага студента залежить від того, чи його обліковий запис підтверджений за студентською поштою та його рейтингом – кармою.

Карма, може бути як додатною, так і від'ємною, тому:

– при від'ємних значеннях карми вага повинна наближатись до нуля при -∞;
– при додатних значеннях вага повинна збільшуватись, проте похідна функції повинна наближатись до нуля при +∞;
– при нульовій кармі вага повинна дорівнювати одиниці.

Отже, для від'ємних значень ми можемо використати b_l^{ks}, а для додатних $\log_{b_r} ks + b_r$, де s, b_l, b_r коефіцієнти які повинні підбиратись експериментально, в залежності від розподілу значень карми у користувачів.

В залежності від того чи студент підтверджений, необхідно домножити вагу карми на відповідний коефіцієнт v (на приклад на 0,1 для непідтвердженого студента та 1,0 для підтвердженого)

В результаті у нас виходить функція:

$$w(k,v) = \begin{cases} vb_l^{ks}, & \text{при } k \le 0 \\ v\log_{\mathrm{b_r}} ks + b_r\,, & \text{при } k > 0 \end{cases}$$

Аби “перехід” між функціями був плавним ми можемо виразити b_l через b_r, вирішивши рівняння $\delta\frac{b_l^{ks}}{\delta k} = \delta\frac{\log_{\mathrm{b}_r} ks + b_r}{\delta k}$, при $x = 0$.

$$\delta\frac{b_l^{ks}}{\delta k} = sb_l^{sk}\ln b_l$$

$$\delta\frac{\log_{\mathrm{b}_r} ks + b_r}{\delta k} = \frac{1}{(b_r + k)\ln b_r}$$

$$\ln b_l = \frac{1}{b_r \ln b_r}$$

$$b_l = e^{\frac{1}{b_r \ln b_r}}$$

На Рис. 2.1 зображено графік функції при $v = 1$, $s = 0{,}05$, $b_r = 2{,}5$

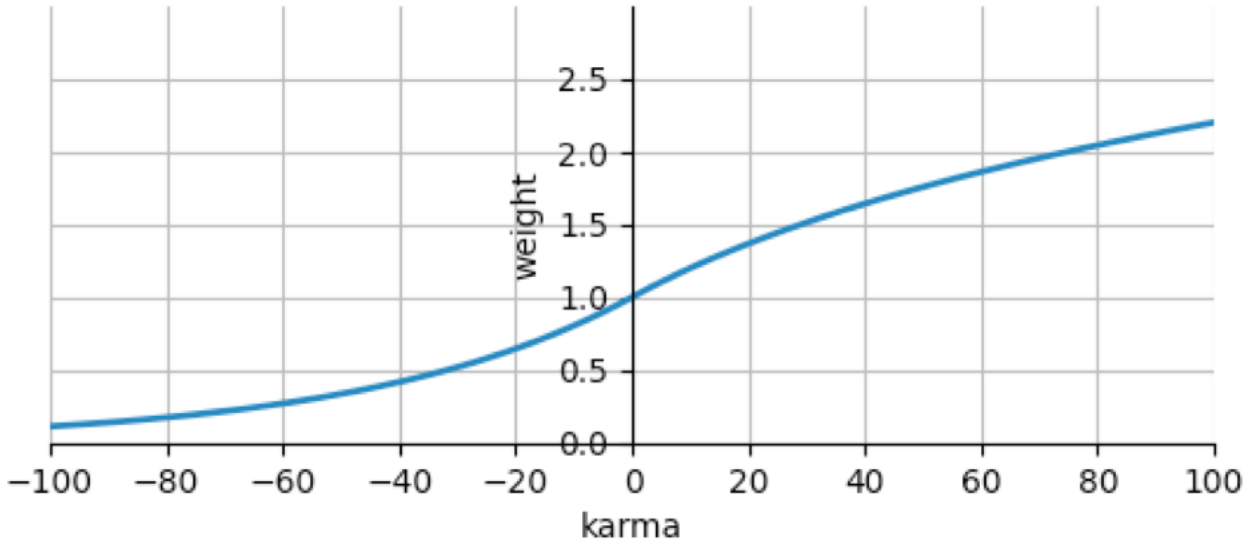

Рис. 2.1 – Графік залежності ваги від карми

3 РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ ЗАСТОСУНКУ

3.1 React

React – це JavaScript бібліотека з відкритим вихідним кодом, призначена для створення користувацьких інтерфейсів, зокрема для single-page application додатків [4].

Компоненти – будівельні блоки будь-якого React-застосунку, являють собою частину користувацького інтерфейсу. Їх можна використовувати повторно і вкладати в інші компоненти, що дозволяє розробникам створювати складні інтерфейси з простих частин.

Стан – це вбудований об'єкт у React-компоненті, який зберігає властивості, що належать компоненту. Коли стан змінюється, компонент перезавантажується, дозволяючи користувацькому інтерфейсу бути реактивним та динамічним.

Хуки – це потужне доповнення, представлене в React 16.8, яке дозволяє використовувати стан та інші можливості React у функціональних компонентах. До хуків ці можливості були доступні лише у класах. У React є кілька вбудованих хуків, таких як `useState` для локального управління станами, useEffect для побічних ефектів (наприклад, отримання даних або підписки на події) та `useContext` для доступу до контекстних даних.

Пропси (властивості) – це те, як компоненти спілкуються один з одним. Пропси передаються від батьківських компонентів до дочірніх і доступні лише для читання, тобто дочірній компонент не може змінювати отримані пропси.

React також представляє контекстний API, який надає можливість обмінюватися значеннями між компонентами без необхідності явно передавати пропси через кожен рівень дерева. Це особливо корисно при обміні глобальними даними, такими як автентифікація користувача або налаштування теми.

Ще одним важливим аспектом React є віртуальний DOM. React створює полегшену копію реального DOM, і коли відбуваються зміни, він спочатку виконує ці операції на віртуальному DOM. Після цього React використовує алгоритм для порівняння старого і нового віртуального DOM, а потім ефективно

оновлює реальний DOM відповідно до нового віртуального DOM, зменшуючи кількість дорогих маніпуляцій [5].

Всі ці можливості та концепції в поєднанні роблять React потужним інструментом для створення швидких, ефективних та складних користувацьких інтерфейсів.

3.2 Структура проекту

На Рис. 3.1 зображено вміст кореневої директорії фронтенд частини проекту.

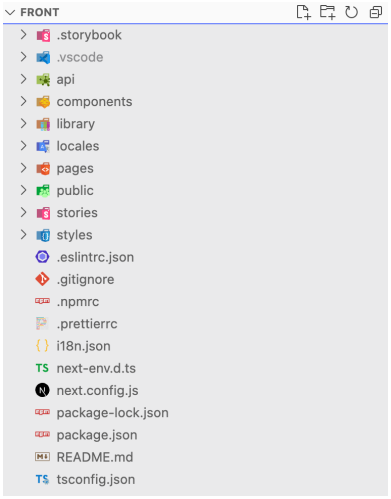

Рис. 3.1 – Вміст директорії проекту

`.storybook` – містить конфігураційні файли середовища для тестування компонентів Storybook.

`api` – містить код функцій обгортки над API бекенду та типи які пов'язані з ним. Для http запитів використовується вбудована в браузери функція `fetch`.

`components` – містить компоненти які тим чи іншим чином отримують глобальний контекст або контекст сторінки та взаємодіють з глобальними станами. Здебільшого містить компоненти, які є складовими частинами сторінок та отримують інформацію про сторінку з її контексту.

`library` (бібліотека компонентів) – містить компоненти, які ніяк не залежать від контексту та керуються виключно пропсами.

`locales` – містить переклади сайту на різні мови.

`pages` – стандартна директорія Next.js, містить компоненти сторінок сайту.

`public` – стандартна директорія Next.js, містить статичні елементи сайту, на приклад: статичні зображення та .svg іконки.

`styles` – директорія, яка містить власноруч написану тему Bootstrap та глобальні стилі які розширюють фреймворк. Локальні стилі компонентів при потребі розміщуються в директорії компонента та використовують CSS Modules.

3.3 Контекстні компоненти

На Рис. 3.2 зображено вміст директорії контекстних компонентів.

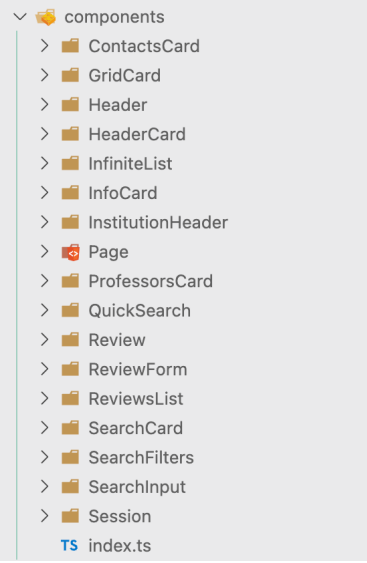

Рис. 3.2 – Вміст директорії `components`

`ContactsCard` – компонент картки контактів. Використовується на сторінках навчального закладу та відділу.

`GridCard` – картка на якій розміщується список елементів у формі сітки. На сторінках навчального закладу – це список факультетів, на сторінці викладача – список курсів.

`Header` – компонент шапки вебсайту. На ній розміщується логотип, який містить посилання на головну сторінку, випадаючий список з посиланнями на інформаційні сторінки, швидкий пошук (QuickSearch) та елементи які відповідають за авторизацію.

`HeaderCard` – картка на якій знаходиться фотографія та ім’я викладача.

`InfiniteList` – компонент, який спрощує автоматичне підвантаження елементів списку при перегортуванні.

`InfoCard` – картка з інформацією у формі списку “поле-значення”, з можливістю об’єднання в групи. Містить поля та групи, які можуть редагуватись користувачем, а також базову інформацію про сутність.

`InstitutionHeader` – картка на якій знаходиться логотип, фотографія, назви англійською та мовою регіону закладу або відділу.

`Page` – містить провайдер контексту сторінки та хук для отримання цього контексту. Використовується для сторінок сутностей.

`ProfessorsCard` – картка зі списком викладачів на сторінках навчального закладу та відділу.

`QuickSearch` – компонент швидкого пошуку.

`Review` – компонент відгуку.

`ReviewForm` – компонент форми відгуку.

`ReviewsList` – компонент списку відгуків.

`SearchCard` – картка, яка відображає пункт результату пошуку на сторінці розширеного пошуку.

`SearchFilters` – картка з фільтрами на сторінці розширеного пошуку.

`SearchInput` – поле вводу для пошуку.

`Session` – містить провайдер глобального контексту та хук для його отримання.

3.4 Розробка компонента

Більшу частину роботи над клієнтською частиною вебзастосунку складає достатньо технічна робота – розробка компонентів. Робота містить досить велику кількість компонентів, процес розробки компонентів доволі схожий. Тому розглянемо процес розробки лише одного, досить простого компонента, на прикладі `IconItem` – допоміжного компонента, який містить рядок з іконки та тексту з опціональним гіперпосиланням та випливаючою підказкою.

Створимо директорію компонента `IconItem`, яка містить:

– `IconItem.tsx` – файл з самим компонентом та типами;

– `index.ts` – файл який експортує компонент та типи.

В файлі компонента спочатку визначимо інтерфейс пропсів компоненту (Фрагмент 3.1).

```
export interface IconItemProps extends
React.HTMLAttributes<HTMLDivElement> {
  icon: string
  text: string
  hint?: string
  href?: string
  isLoading?: boolean
}
```

Фрагмент 3.1 – Інтерфейс `IconItemProps`

Аби ми могли передавати пропси корінному елементу компонента, ми повинні наслідувати його інтерфейс пропсів від інтерфейсу пропсів корінного елемента. Додамо до інтерфейсу наступні ключі:

– `text` – текст який буде відображатись;

– `icon` – назва Google Material іконки;

– `hint` – текст впливаючої підказки;

– `href` – посилання (якщо вказане, то замість звичайного тексту буде відображатись з гіперпосиланням);

– `isLoading` – якщо дорівнює `true`, то замість елементу відображається плейсхолдер (placehlder).

Створюємо функціональний компонент (Фрагмент 3.2). Робимо деструктуризацію параметрів функції таким чином, щоб в `rest` зашились лише пропси, які ми можемо передати корінному елементу `<div>`. Якщо значення `text` пусте, то повертаємо `null`. Якщо `isLoading` дорівнює `true`, повертаємо плейсхолдер.

```
function IconItem({
  icon, text, isLoading, hint, href, className, ...rest
}: IconItemProps) {

  if (!text)
    return null

  if (isLoading)
    return <IconItemPlaceholder
            className={className} {...props} />

// ...
```

Фрагмент 3.2 – Функціональний компонент `IconItem`

Оскільки ми хочемо відображати текстову підказку виключно якщо значення `hint` не пусте, для запобігання дуплікації коду винесемо наповнення компонента в окрему змінну (Фрагмент 3.3).

```
// ...
const content = (
  <div
    {...rest}
    className={classnames(
                'd-flex align-items-start', className)}>
    <Icon>{icon}</Icon>
    {href
      ? <Link href={href}>{text}</Link>
      : <div>{text}</div>
    }
  </div>
)
// ...
```

Фрагмент 3.3 – Наповнення компонента

Кореневому `<div>` ми передаємо залишкові пропси `rest`. Вказуємо класи стилів, які задають вирівнювання дочірніх елементів. Поміщаємо в нього іконку з іменем icon та посилання Next.js якщо пропс `href` заданий, інакше – текстовий div-блок.

Далі (Фрагмент 3.4), якщо підказка (`hint`) задана, ми повертаємо `content` обгорнутий в компонент з бібліотеки Bootstrap React – `OverlayTrigger`, який буде відображати `Tooltip` (з тієї ж бібліотеки) при наведенні курсора на наш компонент. Якщо підказка не задана, повертаємо `content`. Експортуємо за замовчуванням `IconItem`.

```
    // ...
    if (hint)
      return (
        <OverlayTrigger
          placement='left'
          delay={{ show: 500, hide: 100 }}
          overlay={(props) => (
            <Tooltip id='button-tooltip' {...props}>
              {hint}
            </Tooltip>
          )}>
          {content}
        </OverlayTrigger>
      )
    return content
  }

  export default IconItem
```

Фрагмент 3.4 – Умовне повернення компонента

Ми отримали готовий компонент, який ми можемо використовувати в інших частинах коду. Прописування історій для компонентів розглянуте в розділі 3.6, SCSS та CSS-фреймворк Bootstrap в розділі 3.7.

3.5 Сторінки

На Рис. 3.3 зображено вміст стандартної директорії Next.js для сторінок. Шлях в директорії визначає шлях сторінки на вебсайті. Наприклад, якщо ми звернемось до вебсервера Next.js за шляхом `/university/42`, то як компонент сторінки буде братись компонент, який експортується за замовчуванням у файлі `/pages/university/[id].tsx`, і значення `42` буде доступним в контексті сторінки [6].

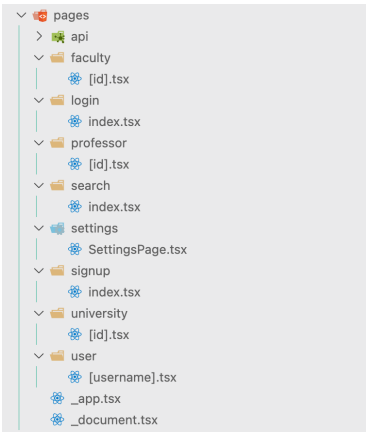

Рис. 3.3 – Вміст директорії `pages`

`_app.tsx` – це кореневий компонент, який обгортає всі компоненти сторінки. Він використовується для збереження узгодженості стану і функцій між різними сторінками застосунку, наприклад, теми або макета. Також може використовуватись для додавання глобального CSS.

`_document.tsx` – використовується для доповнення тегів HTML і Body Next.js застосунку. Він відображається тільки на стороні сервера. Це місце для вставки тегів, які залишаються однаковими для всіх сторінок: метатегів, тегів посилань або скриптів, які повинні бути присутніми на кожній сторінці сайту.

Розглянемо приклад сторінки (Фрагмент 3.5).

```
function UniversityPage({ data }: { data: ApiTypes.University
}) {
  return (
    <>
      <Header />
      <Page.Provider data={data} type='university'>
        <InstitutionHeader />
        <Row className='g-md-3'>
          <Col md={{ span: 4, order: 'last' }}>
            <ContactsCard />
            <ProfessorsCard />
          </Col>
          <Col md={8}>
            <InfoCard />
            <GridCard.Faculties />
            <ReviewsList />
          </Col>
        </Row>
      </Page.Provider>
    </>
  )
}

export default UniversityPage
```

Фрагмент 3.5 – Компонент сторінки університету

3.6 Тестування компонентів

Для тестування компонентів скористаємось Storybook.

Storybook – це комплексний інструментарій і середовище розробки, в першу чергу для JavaScript, для створення, розробки та тестування компонентів інтерфейсу користувача. Він дозволяє розробникам візуалізувати різні стани кожного компонента [7].

Storybook дозволяє розробляти та ізолювати компоненти в режимі реального часу, полегшуючи процес розробки, надаючи пісочницю для створення та тестування компонентів інтерфейсу в ізоляції. Це дозволяє розробникам створювати компоненти незалежно та інтерактивно демонструвати їх у середовищі розробки.

Крім того, функції документації Storybook дозволяють розробникам писати документацію разом з історіями компонентів, забезпечуючи уніфікований спосіб управління описами компонентів.

На Рис. 3.4 зображено інтерфейс середовища Storybook.

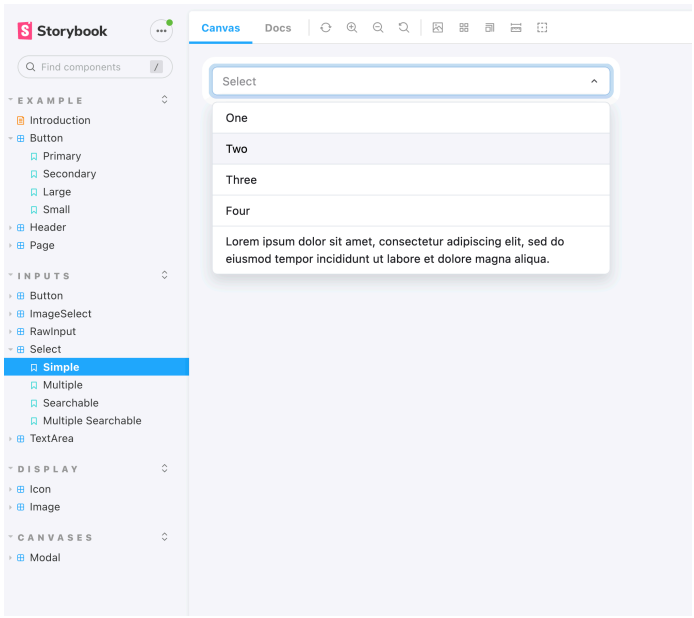

Рис. 3.4 – Інтерактивне середовище Storybook

Приклад історії для кастомного компонента `Select` у React-додатку (Фрагмент 3.6).

```
import { ComponentMeta, ComponentStory } from
'@storybook/react'
import React from 'react'

import Select from './Select'

export default {
  title: 'Inputs/Select',
  component: Select,
} as ComponentMeta<typeof Select>

const Template: ComponentStory<typeof Select> = (args) => {
  const [value, onChange] = React.useState(args.value)

  return (
    <div className='card' style={{ maxWidth: 512 }}>
      <div className='card-body'>
        <Select
          {...args}
          options={[
            { value: '1', label: 'One' },
            { value: '2', label: 'Two' },
            { value: '3', label: 'Three' },
            { value: '4', label: 'Four' },
          ]}
        />
      </div>
    </div>
  )
}
```

Фрагмент 3.6 – Шаблон історії для компонента `Select`

Блок експорту за замовчуванням визначає метадані для компонента `Select`, які будуть використовуватися в Storybook. Поле `title` задає ієрархію для компонента в інтерфейсі Storybook (Inputs – це категорія, а Select – конкретний компонент). Поле `Component` використовується для того, щоб вказати, з яким компонентом пов'язані історії.

`Template` – це стандартний спосіб визначити, як представляти компонент у різних станах. Це функція, яка повертає React-елемент і приймає параметр args, який використовується для налаштування компонента.

Наступна частина коду (Фрагмент 3.7) містить самі історії. Це варіації компонента Select, призначені для відображення у Storybook.

```
export const Simple = Template.bind({})
Simple.args = {}

export const Multiple = Template.bind({})
Multiple.args = {
  multiple: true,
}

export const Searchable = Template.bind({})
Searchable.args = {
  searchable: true,
}

export const MultipleSearchable = Template.bind({})
MultipleSearchable.args = {
  multiple: true,
  searchable: true,
}
```

Фрагмент 3.7 – Історії компонента `Select`

`Simple` історія відображає компонент `Select` без додаткових аргументів. Історія `Multiple` відображає компонент `Select` з пропсом `"multiple"`, що дозволяє вибирати кілька опцій. Історія `Searchable` встановлює пропс `"searchable"` у значення true, що може додати до компонента функцію пошуку. Історія `MultipleSearchable` поєднує в собі обидві опції – `"multiple"` і `"searchable"`.

Кожна історія створюється шляхом прив'язки аргументів до шаблону. Це дозволяє кожній історії передавати шаблону свій набір аргументів, створюючи таким чином варіації компонента Select на основі цих аргументів.

3.7 SCSS та Bootstrap

В проекті використовується SCSS – препроцесор CSS, який вводить у CSS такі програмні конструкції, як: змінні, вкладені правила та міксини (mixins). Він забезпечує модульну конструкцію, багаторазове використання коду та просте обслуговування, тим самим покращуючи робочий процес для розробників. Крім того, SCSS повністю сумісний з усіма версіями CSS, забезпечуючи гнучкість і потужність для більш ефективного керування таблицями стилів [8].

Спочатку розробка велась з використанням стилів написаних власноруч. В результаті розробки сформувався власний, досить об’ємний SCSS фреймворк.

Через складність підтримки великої кількості SCSS-коду було прийняте рішення повністю перейти на фреймворк з відкритим вихідним кодом Bootstrap.

Bootstrap дуже просто кастомізується та розширюється як за допомогою тем, так і за допомогою його SCSS API. Також, при потребі фреймворк завжди можна форкнути. Тому цей фреймворк дуже добре підходить навіть для великих проектів, які потребують гнучкості [9].

Оскільки Bootstrap використовує jQuery для компонентів, які не можуть бути ефективно реалізовані без JavaScript – для React проекту необхідно або писати такі компоненти власноруч, або використати бібліотеку React Bootstrap [10], яка реалізує більшість Bootstrap JavaScript-компонентів з використанням React та має гарну підтримку спільноти.

3.8 Server-Side Rendering

Серверний рендеринг (SSR) в Next.js – це ключова функція, яка полегшує попередній рендеринг вебсторінок на сервері, перш ніж вони будуть відправлені клієнту. Це означає, що браузер отримує повністю відрендерений HTML-файл за запитом, що покращує початковий досвід завантаження сторінки користувачем. SSR надається перевага в SEO, оскільки дозволяє пошуковим роботам отримувати доступ до більш повної версії вебсторінки, тим самим покращуючи індексацію.

Щоб використовувати SSR в Next.js необхідно використовувати функцію `getServerSideProps()`. Ця функція запускається на сервері для кожного запиту та отримує необхідні дані для відображення сторінки. Дані, які повертає `getServerSideProps()`, передаються як пропси до React-компонента сторінки, що дозволяє створювати динамічні сторінки.

Розглянемо приклад `getServerSideProps()` для сторінки навчального закладу (Фрагмент 3.8).

```
export const getServerSideProps: GetServerSideProps<{
  data: ApiTypes.University
}> = async (context) => {
  const id = parseInt(context.params?.id as string)
  const data = await api.page.university.get({ id })
  return {
    props: {
      data: {
        id,
        ...data,
      },
    },
  }
}
```

Фрагмент 3.8 – Функція `getServerSideProps`

3.9 Інтернаціоналізація

Інтернаціоналізація – це проектування та розробка вебзастосунків без мовних чи локальних обмежень, має вирішальне значення для створення доступних та інклюзивних продуктів.

Next-translate – бібліотека, яка спрощує інтернаціоналізацію проектів на Next.js. Вона використовує JSON-файли для зберігання перекладів, пропонуючи зручний метод роботи з мовними ресурсами. Однією з важливих особливостей є автоматичне визначення мови, що покращує взаємодію з користувачем шляхом автоматичного представлення контенту мовою, якій користувач надає перевагу, відповідно до налаштувань браузера [11].

Бібліотека також підтримує простори імен, що полегшує організацію файлів перекладу в окремі сегменти для покращення зручності подальшої підтримки. Крім того, next-translate легко інтегрується як зі статичною генерацією сайтів (SSG), так і з рендерингом на стороні сервера (SSR), надаючи розробникам гнучкість у виборі найкращого методу рендерингу для їхніх потреб.

Попри потенційну складність інтернаціоналізації, next-translate залишається оптимізованим з точки зору продуктивності. Він гарантує, що в будь-який момент часу завантажуються лише необхідні переклади, зменшуючи вплив на продуктивність програми та час завантаження.

Після встановлення бібліотеки, необхідно створити конфігураційний файл `i18n.json` (Фрагмент 3.9).

```
{
  "locales": ["en", "uk", "ru"],
  "defaultLocale": "en",
  "pages": {
    "*": ["common"]
  }
}
```

Фрагмент 3.9 – Конфігураційний файл next-translate

В ньому ми вказуємо, які локалі будуть використовуватись в проекті, стандартну локаль та простори імен (наразі використаємо тільки один простір імен `common` для всіх сторінок проекту).

В конфігураційному файлі проекту `next.config.js` перед експортуванням об'єкта з конфігурацією необхідно передати його в функцію `nextTranslate` (Фрагмент 3.10), яка налаштує конфігурацію для next-translate.

```
const nextTranslate = require('next-translate-plugin')
module.exports = nextTranslate(nextConfig)
```

Фрагмент 3.10 – Фрагмент конфігураційного файлу Next.js

Після конфігурування, в директорії `locales` необхідно створити директорії для кожної з локалей (в нашому випадку: `en`, `uk`, `ru`), а в них створити файли для кожного простору імен (оскільки у нас один простір імен – `common.js`).

В цих файлах для кожної локалі необхідно прописати словник «ключ-значення», де ключем є певна асоціативна назва, а значенням є переклад.

Для керування цими перекладами ми можемо використовувати зручний плагін для Visual Studio Code – i18n Ally. Він значно покращує ефективність та зручність керування перекладами. i18n Ally має вбудовані анотації, пошук використання на льоту та посилання на ключі у кодовій базі, що полегшує навігацію та керування перекладами. Крім того, він надає детальний огляд файлів перекладу, включаючи відсутні ключі та статистику використання, покращуючи таким чином загальну зручність обслуговування [12].

Щоб отримати переклад ми можемо скористатись хуком `useTranslation` (Фрагмент 3.11).

```
function LoremIpsum() {
  const { t } = useTranslation('common')
  return <div>t('lorem-ipsum')</div>
}
```

Фрагмент 3.11 – Використання хука `useTranslation`

Як аргумент хука ми вказуємо простір імен. Хук повертає функцію `t`, яку ми можемо використовувати для отримання перекладу ключа для встановленої локалі. В прикладі ми отримуємо переклад ключа “`lorem ipsum`”.

4 ОГЛЯД СЕРВЕРНОЇ ЧАСТИНИ ЗАСТОСУНКУ

4.1 Мікросервіси

Проектування та розробка бекенду не є частиною даної роботи, але ми поверхнево розглянемо його поточну архітектуру.

Наразі бекенд написаний з використанням мікросервісної архітектури. На Рис. 4.1 зображена схема взаємодії мікросервісів.

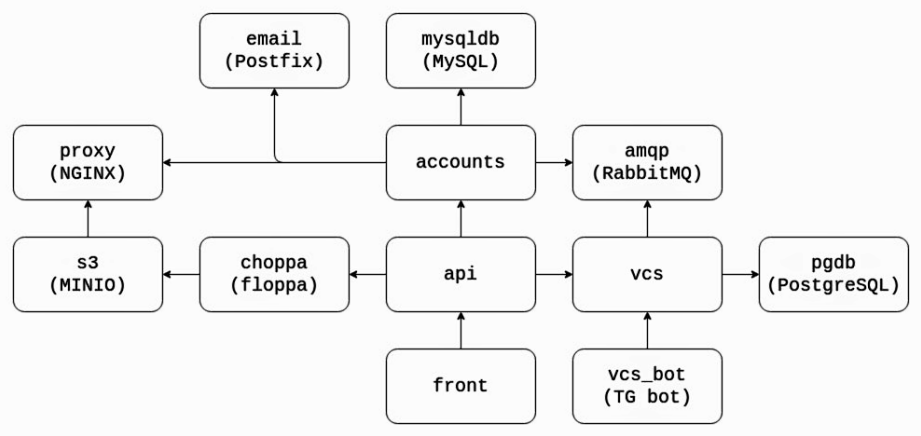

Рис. 4.1 – Схема мікросервісів

`api` – мікросервіс з яким безпосередньо взаємодіє фронтенд частина застосунку. Оскільки ми використовуємо SSR, запити можуть робитись як браузером, так і Next.js сервером при рендерингу сторінки. Цей сервіс надає інтерфейс для взаємодії з іншими мікросервісами. Написаний з використанням Python, фреймворку Flask та ORM SQLAlchemy.

`vcs` – мікросервіс, який реалізує функціонал системи контролю версій (version control system) для зберігання та оновлення станів сторінок на вебсайті. Написаний на Python та Flask.

`vcs_bot` – телеграм бот, для спрощення модерації контенту для розробників сайту, написаний мовою програмування Go. Планується розробка адмінпанелі, яка замінить функціонал бота.

`choppa` – мікросервіс який відповідає за стискання, обробку завантажених користувачами зображень та передачею їх на сервіс-сховище. Написаний з використання Python та Flask.

`accounts` – мікросервіс авторизації, написаний мовою програмування Go. Планується заміна його на готове OAuth рішення.

`MinIO` – вільне сервіс-сховище з відкритим вихідним кодом, self-hosted альтернатива Amazon S3.

4.2 Взаємодія мікросервісів та база даних

Наразі більшість мікросервісів взаємодіють за допомогою http-запитів, проте проводиться поступовий перехід на використання брокера повідомлень RabbitMQ.

В останній стабільній версії бекенду використовується база даних MySQL. Проводиться перехід на PostgreSQL.

На Рис. 4.2 зображена поточна схема бази даних.

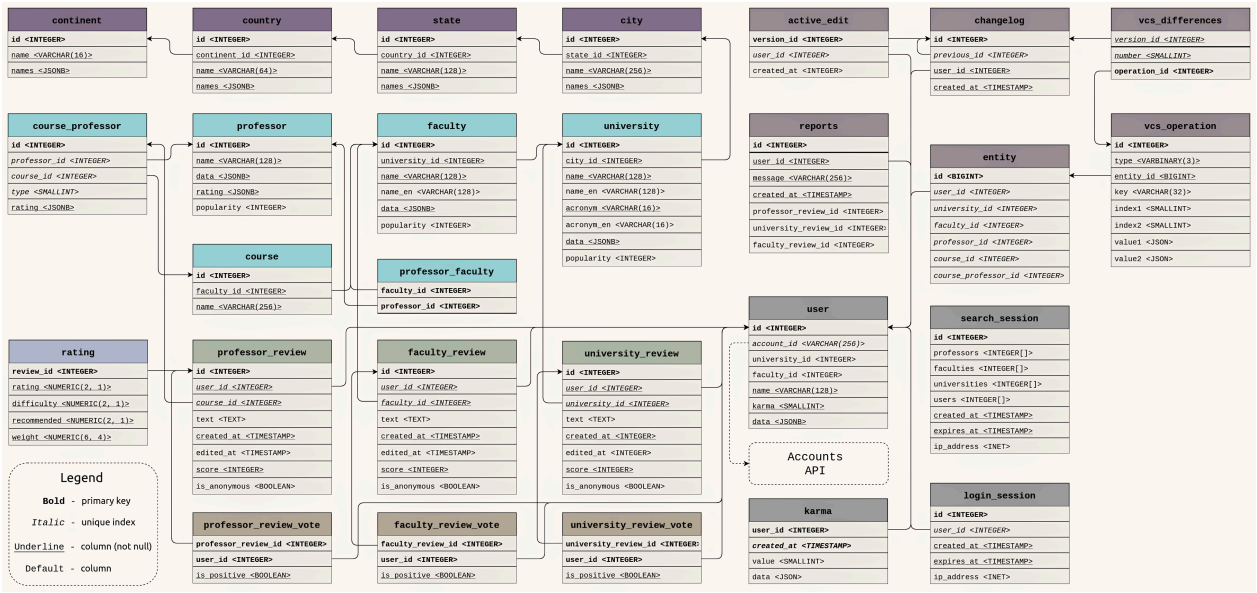

Рис. 4.2 – Схема бази даних

5 ОПИС ГРАФІЧНОГО ІНТЕРФЕЙСУ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

5.1 Пошук

Коли користувач заходить на наш вебсайт, він одразу потрапляє на сторінку пошуку (Рис. 5.1). В правій колонці розміщена картка з опціями впорядкування та фільтрування пошуку.

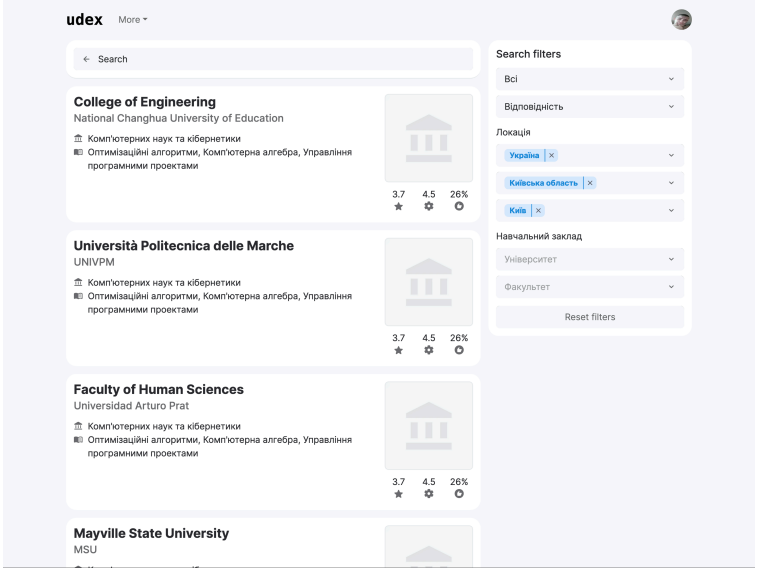

Рис. 5.1 – Сторінка розширеного пошуку

На всіх інших сторінках користувач може скористатись швидким пошуком (Рис. 5.2), не переходячи на сторінку розширеного пошуку, для цього необхідно сфокусуватись на полі пошуку, яке розміщене в шапці сайту або натиснути клавішу “/“.

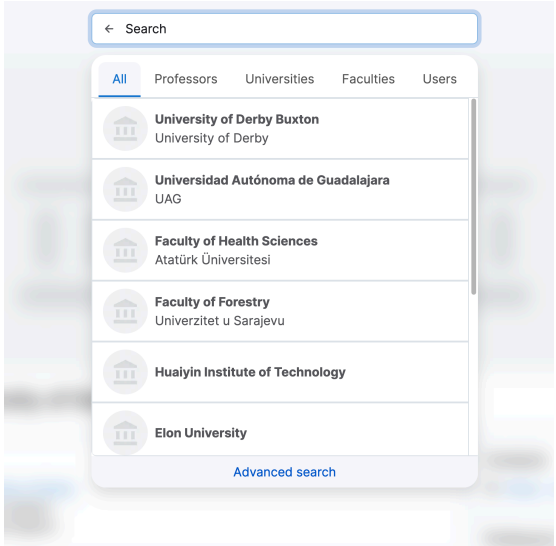

Рис. 5.2 – Швидкий пошук

5.2 Навчальний заклад та відділ

На сторінці відділу (Рис. 5.3) на верхній картці розміщується логотип, фотографія університету, його назви мовою регіону та англійською.

Нижче праворуч розміщені картки з контактною інформацією та списком викладачів, які викладають в відділі навчального закладу.

Ліворуч розміщені: картка з загальною інформацією про відділ, де додатково може бути розміщена довільна інформація та картки з відгуками, які залишили користувачі.

Користувач може редагувати інформацію на сторінці та залишати відгук.

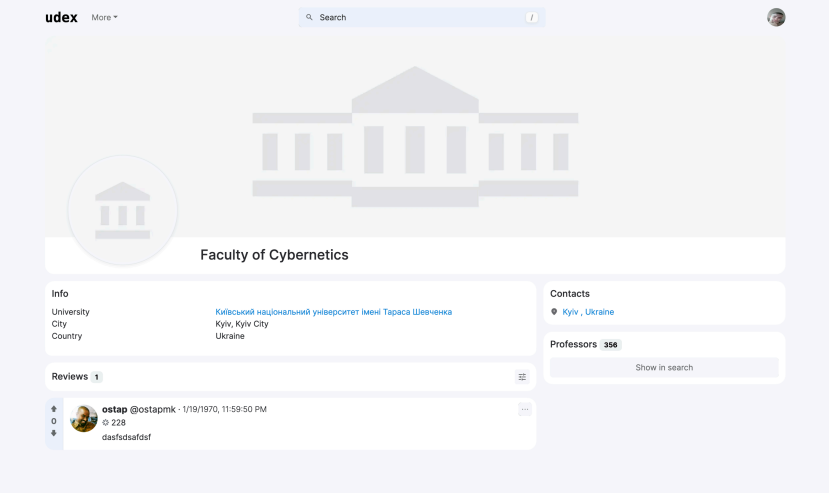

Рис. 5.3 – Сторінка відділу

Сторінка навчального закладу (Рис. 5.4) структурно повторює сторінку відділу, проте також містить список відділів з посиланнями на їхні сторінки в лівій колонці.

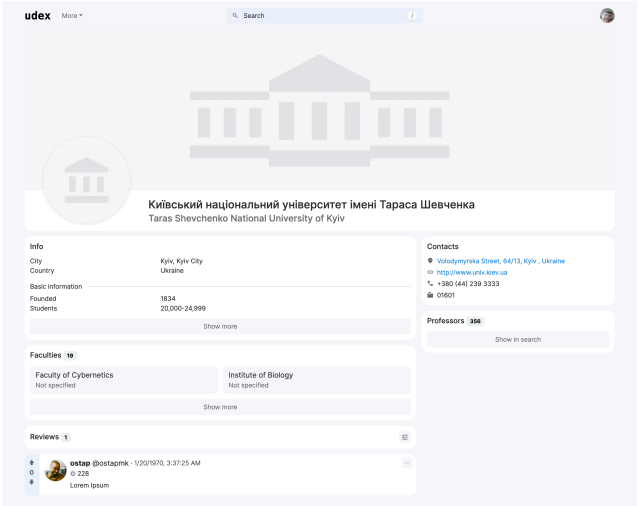

Рис. 5.4 – Сторінка навчального закладу

5.3 Викладач та студент

На сторінці викладача (Рис. 5.5) в лівій колонці розміщена фотографія викладача, його ім’я та середні оцінки курсів, які він викладає.

В правій колонці містяться: картка з загальною та додатковою інформацією, картка зі списком курсів та картки з відгуками користувачів.

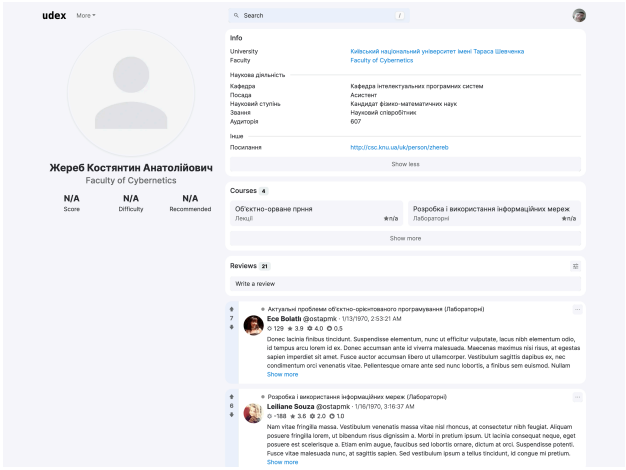

Рис. 5.5 – Сторінка викладача

На сторінці студента (Рис. 5.6) ліва колонка містить інформацію про те, в якому навчальному закладі він навчається, його спеціалізація, курс, рівень карми та його псевдонім. Права колонка містить всі неанонімні відгуки, які залишив студент.

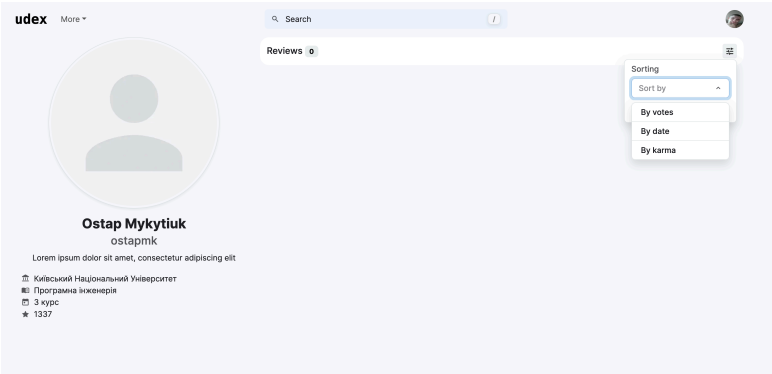

Рис. 5.6 – Сторінка користувача

5.4 Вхід та реєстрація

Користувач може авторизуватись на вебсайті перейшовши на сторінку входу (Рис. 5.7.а) та заповнивши форму. Для створення облікового запису необхідно заповнити форму на сторінці реєстрації (Рис. 5.7.б).

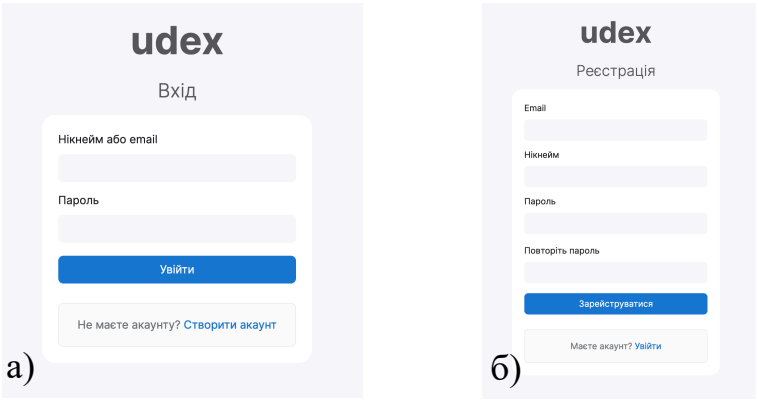

Рис. 5.7 – а) сторінка входу б) сторінка реєстрації

5.5 Налаштування

На сторінці налаштувань (Рис. 5.8) студент може відредагувати інформацію своєї сторінки, змінити параметри приватності та поміняти пошту чи пароль.

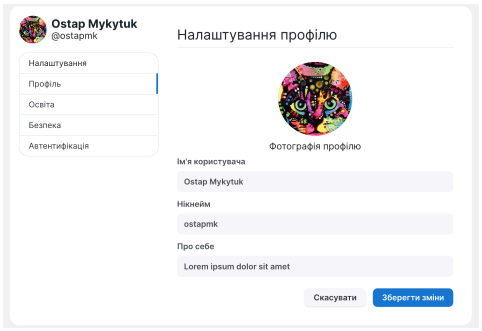

Рис. 5.8 – Сторінка налаштувань

5.6 Відгук

Натиснувши на поле “Написати відгук” на сторінках закладу, відділу або викладача відкривається форма написання відгуку (Рис. 5.9). Студент може вибрати курс, який він оцінює та оцінити його. Також опціонально він може приховати свій псевдонім.

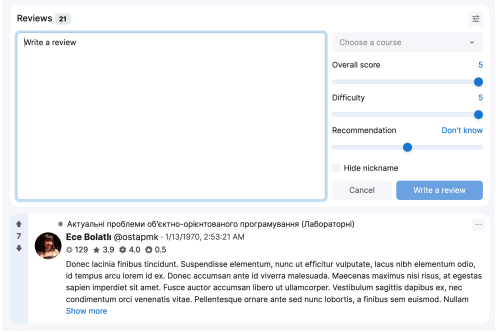

Рис. 5.9 – Форма відгуку

5.7 Редагування курсу

Натиснувши на курс в списку курсів на сторінці викладача, відкривається модальне вікно редактору курсу (Рис. 5.10). Користувач може змінити назву курсу та його тип.

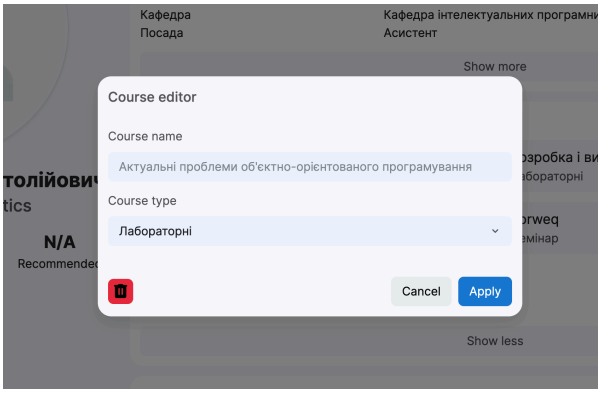

Рис. 5.10 – Форма редактору курсу

5.8 Зміна теми та мови

Натиснувши на зображення профілю в шапці, відкривається випадаюче меню, в якому ми можемо отримати доступ до швидких налаштувань теми та мови (Рис. 5.11). Також ми можемо перейти на свою сторінку або налаштування та вийти з облікового запису.

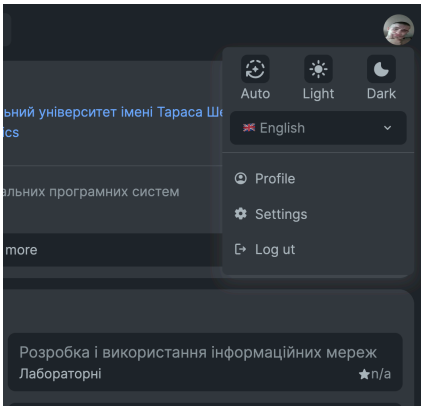

Рис. 5.11 – Випадаюче меню користувача

ВИСНОВКИ

В рамках виконання роботи було успішно реалізовано базову частину вебзастосунку, яка реалізує спроектовану систему оцінювання та зважування оцінок користувачів.

В рамках виконання роботи було виконано проектування та розробка клієнтської частини вебзастосунку для оцінювання якості освіти. Вебзастосунок було створено з використанням сучасних технологій, таких як Next.js, React.js, TypeScript, Storybook і next-translate, які дозволяють реалізувати надійний та ефективний продукт, який легко масштабується та підтримується.

Використання системи підтвердження студентів дозволяє забезпечити автентичність даних, підвищити якість відгуків та оцінок, відфільтрувати некоректну інформацію. Особливість даного застосунку полягає в псевдонімному (опціонально анонімному) оцінюванні, яке дозволяє студентам відкрито висловлювати свої думки та враження про якість освіти.

Цей вебзастосунок має потенціал стати ефективним інструментом для оцінювання якості освіти, що дозволить студентам приймати правильні та обґрунтовані рішення, а викладачам та адміністрації освітніх установ – вдосконалювати навчальний процес на основі отриманих відгуків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. About RateMyProfessors [Електронний ресурс] – Режим доступу до ресурсу: https://www.ratemyprofessors.com/About.jsp.
2. General Data Protection Regulation [Електронний ресурс] – Режим доступу до ресурсу: https://gdpr-info.eu/.
3. Про проект – СтудЗона [Електронний ресурс] – Режим доступу до ресурсу: https://studzona.com/about.
4. Learn React [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/learn.
5. Banks A. Learning React: Functional Web Development with React and Redux / A. Banks, E. Porcello., 2017. – 350 с. – (1st edition).
6. Docs | Next.js [Електронний ресурс] – Режим доступу до ресурсу: https://nextjs.org/docs.
7. Storybook Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://storybook.js.org/docs.
8. Sass: Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://sass-lang.com/documentation/.
9. Bootstrap Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/docs.
10. React Bootstrap Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://react-bootstrap.github.io/docs/getting-started/introduction/.
11. next-translate – GitHub page [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/aralroca/next-translate.
12. i18n-ally – GitHub page [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/lokalise/i18n-ally.