МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) автоматизації і комп'ютерних систем
Кафедра інформаційних систем

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ _Форсюк А.В.__	_____ _Чумаченко С.М.__
(підпис) (прізвище та ініціали)	(підпис) (прізвище та ініціали)
«___» _____ 20__р.	«___» _____ *20__р.*

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки та інформаційні технології»
(код та назва спеціальності)
освітньо-професійної програми Комп'ютерні науки
на тему: Розроблення backend сайту кафедри екологічної безпеки та охорони праці НУХТ.
Виконав: здобувач 4 курсу, групи КН-4-5

Мохонько Іван Олександрович _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Чумаченко Сергій Миколайович _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти Чумаченко Сергій Миколайович _____
(прізвище та ініціали) (підпис)
Чумаченко Сергій Миколайович _____
(прізвище та ініціали) (підпис)
Чумаченко Сергій Миколайович _____
(прізвище та ініціали) (підпис)
Рецензент Смітюх Ярослав Володимирович _____
(прізвище , ім'я та по батькові повністю) (підпис)

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

Здобувач_____
(підпис)

Київ - 2020р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) _автоматизації і комп'ютерних систем_____
Кафедра _інформаційних систем_____
Освітній ступінь_ бакалавр_____
Спеціальність_122 «Комп'ютерні науки та інформаційні технології»_____
(код і назва)
Освітньо-професійна програма ____Комп'ютерні науки _____
_____
(назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри_Чумаченко С.М._

_____
"____" _____20___року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

_____Мохонька Івана Олександровича_____
(прізвище, ім'я, по батькові)
1. Тема роботи: Розроблення backend сайту кафедри екологічної безпеки та охорони праці НУХТ .
керівник роботи__д.т.н., ст. науковий співробітник Чумаченко С.М_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від "27"квітня 2020 року №269-кс
2. Строк подання здобувачем роботи_25 травня_____
3. Вихідні дані до роботи
1. Інформація про події
2. Логін та пароль користувача
3. Зображення
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1.Системний аналіз кафедри «ЕБОП»
2. Моделі бази даних панелі управління
3. Інтерфейс панелі управління
4. Охорона праці та техніка безпеки
5. Перелік графічного матеріалу
1. Організаційна структура підприємства
2. Функціональна та концептуальна моделі
3. Логічна та фізична моделі бази даних
4. Приклади роботи (інтерфейс користувача)
5. Фрагменти коду

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	д.т.н., ст. науковий співробітник Чумаченко С.М.		
2	д.т.н., ст. науковий співробітник Чумаченко С.М.		
3	д.т.н., ст. науковий співробітник Чумаченко С.М.		

7. Дата видачі завдання___27.04.2020_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Приміт ка
1	Системний аналіз кафедри та постановка задачі на проектування	13.03.2020 - 24.03.2020	Викон ано
2	Проектування БД	25.03.2020 - 06.04.2020	Викон ано
3	Створення панелі управління	09.04.2020 - 24.04.2020	Викон ано
4	Написання інструкцій користувача	27.04.2020 - 30.04.2020	Викон ано
5	Оформлення пояснювальної записки	06.05.2020 - 16.05.2020	Вико нано
6	Оформлення презентації	17.05.2020- 18.05.2020	Вико нано

Здобувач _____ __Мохонько О.І.____
(підпис) (прізвище та ініціали)

Керівник роботи _____ ___Чумаченко С.М___
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Головною метою даної дипломної роботи є створення backend частини та панелі управління веб-сайту кафедри «Екологічної безпеки та охорони праці» Національного університету харчових технологій з покращеним інтуїтивним інтерфейсом, який полегшить роботу з адмініструванням контенту веб-сайту. Надаватиме можливість отримувати дані з бази даних, створювати новини, сторінки, вакансії та завантажувати зображення до галереї. Створена система буде мати простий інтерфейс та великі можливості по її розширенню та покращенню, що є перевагою при довгостроковій експлуатації системи та її підтримки.

Об’єктом дослідження є кафедра «Екологічної безпеки та охорони праці» НУХТ.

Предметом роботи є створення backend частини та панелі управління веб-сайту кафедри «Екологічної безпеки та охорони праці» НУХТ.

Дипломна робота містить 83 сторінки, 2 таблиці, 85 рисунків, 4 додатка, 9 літературних джерел.

КЛЮЧОВІ СЛОВА: СИСТЕМА, КАФЕДРА «ЕКОЛОГІЧНОЇ БЕЗПЕКИ ТА ОХОРОНИ ПРАЦІ», БАЗА ДАНИХ, ПАНЕЛЬ АДМІНІСТРАТОРА, ІНТЕРФЕЙС.

ABSTRACT

The problem with the thesis is the creation of support services and panels of the administrator of the website of the Department of "Environmental and Labour Safety" Educational University with its own technologies with improved intuitive interface, which facilitates the introduction of content on the website. The provided panel gets data from the database, creates news, pages, vacancies, uploads images. The system will have a simple interface and large cases when it is expanded and maintained, which leads to the highest exclusive system and its maintenance.

The object of research is the Department of "Environmental and Labours Safety" NUFT.

The subject of the work is the creation of a backend part and a panel of the administrator of the website of the Department of "Environmental Safety and Health" NUHT.

Thesis contains 83 pages, 2 tables, 85 pictures, 4 appendices, 9 literature sources.

KEY WORDS: SYSTEM, DEPARTMENT OF ENVIRONMENTAL SAFETY AND OCCUPATIONAL HEALTH AND SAFETY, DATABASE, ADMINISTRATOR'S PANEL, INTERFACE.

Зміст

ВСТУП

У сучасному світі неможливо представити підприємство, кафедру, та навіть людину яка не має свого сліду у всесвітній павутині. За останні десятиліття вона стала основним майданчиком де знаходиться вся інформація. В ній можливо знайти все, від рецептів незвичайних страв до характеристик найперших танків.

Основною перевагою використання інтернету є швидкість та простота у отриманні будь-якої інформації. Доступ з будь-якого пристрою, у будь-якому куточку світу де є покриття.

Користуючись цим, кожен може створити свій сайт, або зареєструватися у соціальній мережі та поширювати інформацію яка йому цікава. Люди завантажують фотографії до Інстаграму, а підприємства використовують цю можливість як майданчик для реклами та інформування її клієнтів.

Університети та кафедри не відстають від цього тренду, та мають свої веб-сайти. Адже кожен абітурієнт та студент може знайти на них інформацію яка його цікавить.

На цих сайтах, зазвичай, публікується інформація про те які подіїї відбулися на кафедрі чи загалом в університеті, переглянути зображення корпусів та аудиторій, знайти вакансії на яку давно хотів потрапити, але не знав де її шукати. І все це за лічені долі секунди, все що потрібно – знати адресу веб-сайту. Не потрібно їхати і запитувати у викладачів, або шукати вивіски біля деканатів. Все це робить кафедру або університет більш близкою до користувачів мережі.

Але, якщо сайт є некрасивим, або в ньому немає інформації яка потрібна користувачам, він не представляє ніякої користі. Сучасні стандарти розвиваються з шаленим темпом, нові технології з’являються майже кожен рік, і сайт який був гарним два роки тому, вже виглядає погано на фоні інших. Саме для цього потрібно своєчасно удосконалювати та покращувати свої веб-сайти,

адже негативний досвід роботи з веб-сайтом може погано вплинути на потенційного абітурієнта.

Розділ 1. Системний аналіз кафедри «ЕБОП» та постановка задачі на проектування

1.1. Загальна характеристика кафедри «Екологічної безпеки та охорони праці»

Об’єктом дослідження в даній кваліфікаційній роботі є кафедра «Екологічної безпеки та охорони праці» НУХТ.

Кафедра «ЕБОП» НУХТ була започаткована в 1937 році та є частиною факультету «БТЕК» та підготовлює фахівців у галузі знань «Природничі науки» за освітнім ступенем «бакалавр», спеціальністю «Екологія» освітньою програмою «Екологія та екоменеджмент», «магістр» за спеціальністю «Екологія» та «Технології захисту навколишнього середовища» за освітніми програмами «Екологія та охорона навколишнього середовища» та «Екологічний контроль та аудит». [1]

З 2018 року після була об’єднана з кафедрою безпеки життєдіяльності та було розширено коло дисциплін, а саме «Охорона праці та безпека життєдіяльності», «Основи охорони праці», «Основи будівництва в галузі» та «Основи промислового будівництва та санітарної техніки».

За весь час існування для студентів всіх спеціальностей викладалися загально-наукові дисципліни, а саме «Екологія», «Основи охорони праці та безпека життєдіяльності», «Основи промислового будівництва та санітарної техніки».

Основні аспекти кафедри «ЕБОП» НУХТ:

Спеціалізовані лабораторії для навчання студентів, що обладнані приладами за сучасними стандартами та установками для студентів; також присутні лабораторії, які призначені для екологічного контролю.

Викладачі кафедри покращують методики та принципи навчання студентів. Між НУХТ та організаціями екологічного напряму, підприємствами харчової та переробної промисловості, науково-дослідними та галузевими інститутами укладено певні угоди, які надають можливість студентам

проходити преддипломну практику, а також надають позиції для працевлаштування для студентів після закінчення навчання в університеті. [1]

1.2. Організаціна структура факультету

1.2.1. Структура факультету «БТЕК»

Верхівку факультету «БТЕК» займає декан. Після нього в ієрархії йде вчена рада факультету, заступники декана за напрямами, деканат та рада студентського самоврядування. Повна структура зображена на рис. 1.

Основними структурними підрозділами є кафедри:

- біотехнології та мікробіології;
- фізики;
- екологічної безпеки та охорони праці;
- фізичного виховання.

Кафедри проводять навчально-виховну та методичну діяльність з однієї або кількох споріднених спеціальностей або навчальних дисциплін, а також здійснюють наукову, науково-дослідну та науково-технічну діяльність за певним напрямом.

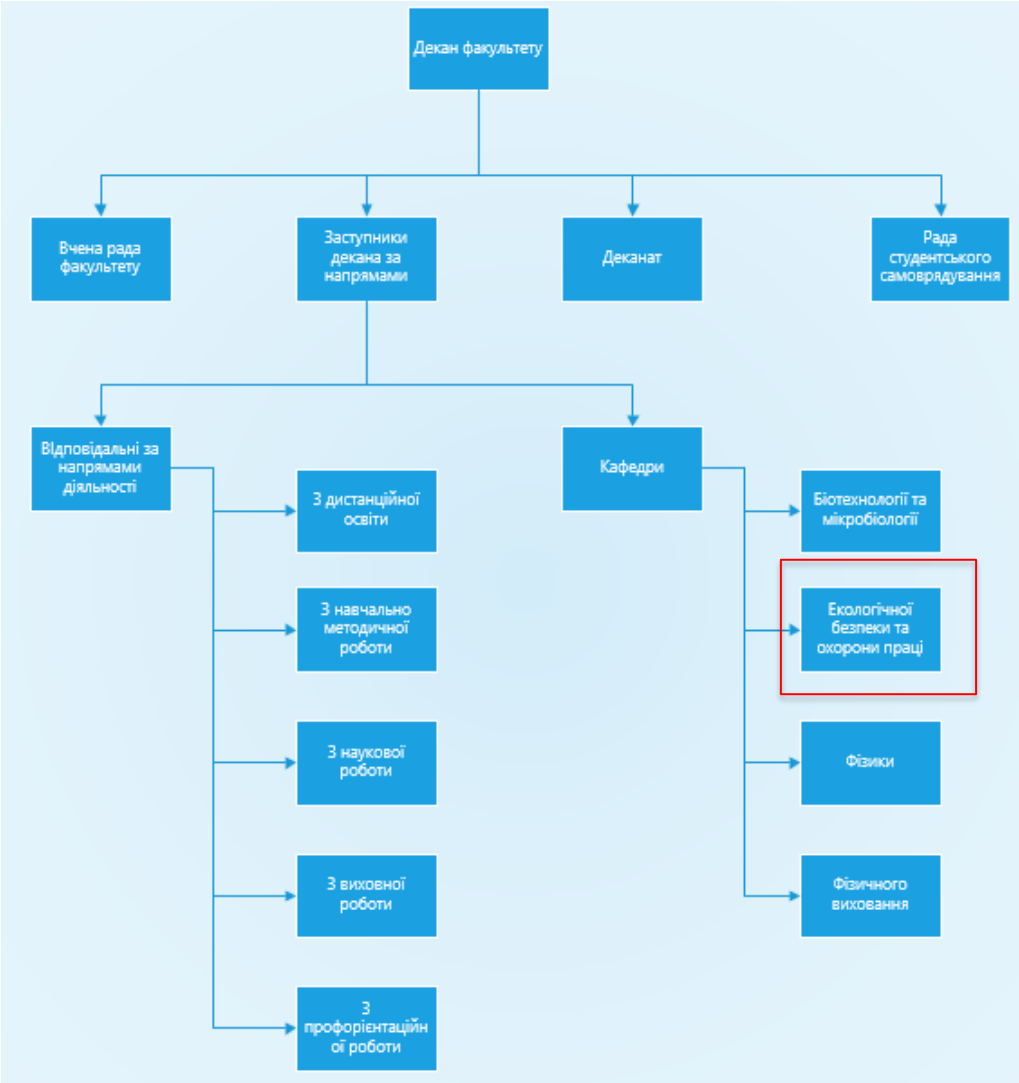

Рис. 1. Структурна схема факультету «БТЕК»

1.2.2. Опис роботи кафедри «ЕБОП»

Головним завданням кафедри є здійснення навчальної і навчально-методичної роботи із навчальних дисциплін, виховної роботи серед студентів, наукових досліджень, підготовки науково-педагогічних кадрів та підвищення їх кваліфікації.

Основними напрямками діяльності кафедри є:

- навчальна робота;
- методична робота;
- науково-дослідна робота;

- організаційна робота;
- виховна робота.

Функції кафедри: розробка навчальних програм дисциплін кафедри, контроль якості навчання студентів, участь у науково-дослідній роботі, організація і супроводження веб-сторінки кафедри.

Кафедра складається із завідувача кафедри, заступник завідувача кафедри, професори, доценти, старші викладачі, викладачі, аспіранти та навчально-допоміжний персонал. Структура кафедри наведена на рис. 2.

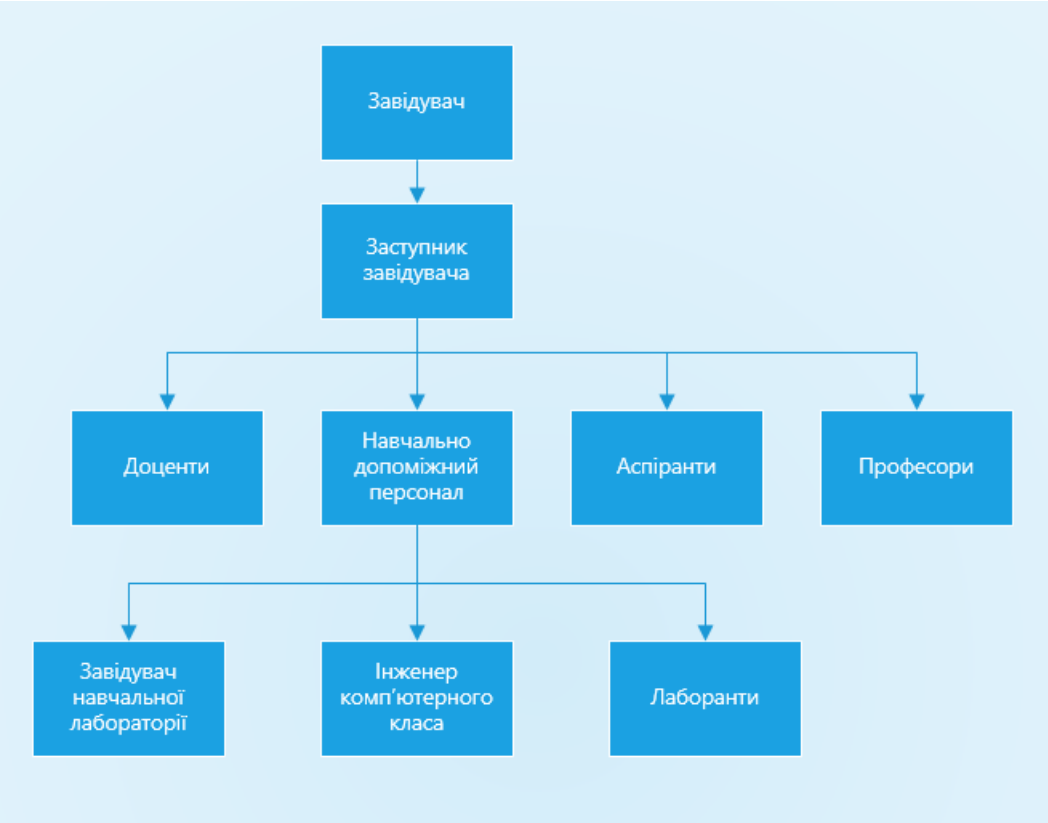

Рис. 2. Організаційна структура кафедри «ЕБОП»

Завідувач кафедри забезпечує організацію освітнього процесу та виконання навчальних планів, контролює якість викладання дисциплін кафедри, контролює виконання правил з охорони праці та протипожежної безпеки.

1.2.3. Взаємодія з іншими структурами факультету

Табл. 1. Приклад взаємодії з іншими структурами

Отримує	**Надає**
Взаємодія з університететом в цілому	
Накази та розпорядження ректора, інформаційні та службові листи проректорів. Матеріали про роботу Вченої ради факультету та університету і його комісій (рішення, постанови). Матеріали по виховнії роботі.	Акти про готовність кафедри до навчального року. Посадові інструкції співробітників кафедри. Звіти про роботу. Матеріали з виховної роботи. Положення про кафедру.
Взаємодія з факультетом	
Навчальні плани. Розклад занять, іспитів та заліків. Проект розкладу кафедри. Списки студентів.	Проекти навчальних планів. Успішність студентів. Розклад. Звіти та про організацію і проходження практик студентами. Графік відпусток.
Взаємодія зі службами проректора по методичній роботі	

Обсяг навчального навантаження кафедри. Інформація про конференції.	Список груп на потоках. Розподіл навчального навантаження для викладачів. Зміни в навчальному процесі викладачів кафедри. Плани роботи. Список дисциплін за вибором студентів. Проекти навчальних планів.
Взаємодія з іншими кафедрами	
Взаємодія задля поліпшення якості навчання. Навчально-методичний комплекс дисциплін.	Плани поліпшення якості освітнього процесу. Комплекс дисциплін.
Взаємодія з вченою радою факультету	
Графік організації підвищення кваліфікації науково-педагогічних працівників. Перелік баз практик.	Звіти про роботу кафедри. Звіти про успішність студентів. Відомості для заповнення модулів. Накази на розподіл студентів на місця проходження практики.

1.3. Аналіз нинішнього стану панелі управління

Поточна панель управління є частиною платформи «Wix», яка дозволяє створювати сторінки за допомогою простого конструктора шаблонів.

Конструктор сайтів — система для створення WEB-сторінок без знань мови розмітки, стилів та програмування інтерфейсу.

Панель являє собою функціональність по створенню сторінок, що складається з файлових даних та коду, які регулюють його роботу. Система має підключення до бази даних, яка надається конструктором «Wix».

Функціональність панелі дозволяє створювати тільки прості сторінки. Можливості по удосконаленню функціональності як панелі так і веб-сайту та його розширенню обмежені. Контент слабо структурований та має фіксовану структуру, яка повною мірою не підходить під потреби кафедри. Тому необхідне вирішення цих проблем шляхом розробки нової версії панелі управління, для полегшення роботи з контентом веб-сайту та підвищення його якості.

1.4. Розроблення функціональної моделі процесів «як-є»

Для повноцінного аналізу поточної версії панелі управління розглянемо процес додавання новини. Схеми процесів 1 та 2 рівня декомпозиції наведено у додатку А на рис. А1 та А2.

Як можно зрозуміти із схеми, спочатку адміністратор, або модератор пише основний текст новини, а саме назву, короткий опис та саму новину. Після цього він обирає зображення яке буде основним зображенням новини, та буде відображене в списку новин на веб-сайті. Після введення ми відправляємо запит до серверу, який запише ці дані до бази даних.

1.4.1. Виявлені проблеми

Відразу не помітно, але цей процес має декалька недоліків. Основний із них це відсутність валідації даних на формі. Наприклад, користувач завантажив дуже маленьке зображення, або його вага дуже велика, що призведе до повільного завантаження зображення на веб-сайті, при повільній швидкості інтернету у користувача. Або формат зображення погано відображається в браузері (таке теж може бути).

Інший недолік полягає у відсутності віднесення новини до тієї чи іншої категорії, що погано впливає на структуру контенту. Якщо новин дуже багато, то вся інформація буде перемішана, і буде дуже складно знайти якісь тематичні новини чи інформацію яка була в них.

Відсутність валідації на дожвину тексту для опису. Опис являє собою коротку інформацію про те, що розповідається у новині, він створений для показу на сторінці всіх новин, де вони показані у хронологічному порядку. Якщо опис дуже великий це перенавантажити інтрефейс веб-сайту, або, в гіршому випадку, при поганій розмітці шаблону, зламати дизайн сторінки.

Всі ці недоліки призводять до того, ще веб-сайт скоріше є блогом, а не інформаційним ресурсом для кафедри університету.

1.4.2. Задачі автоматизації

Перш за все потрібно додати валідацію даних на форму додавання новини, це позитивно вплине на якість контенту на веб-сайті та його оформлення.

По друге, додамо можливість відности новину до якоїсь категорії, задля структуризації новин. Це дозволить знахиди новини та події які були в них освітлені за тематикою.

1.5. Аналіз існуючих аналогів розробки WEB-сайтів

Основними аналогами такого роду веб-сайтів є системи менджементу контенту, або CMS (content management system або система управління контентом) – це основа для розробки і редагування контенту веб-сайту. Простіше кажучи - конструктор, який дозволяє створити веб-сайт і наповнювати його контентом різного роду, який дозволяється системою. Веб-сайт створений за допомогою такої системи може бути простим, без складних елементі, та може бути і досить складним у своїй структурі: маса різних плагінів які надають додаткову функціональність та деталей інтерфейсу.

1.5.1. WordPress

WordPress — одна із найпопулярніших систем управління контентом на веб-сайті та його дизайном. Серверна частина написана на мові PHP, в якості бази даних використовує MySQL. Створювати можно все, від блогів до досить складних веб-ресурсів та інтернет-магазинів. Має систему яка дозволяє вибирати різні теми та елементи дизайну. Дозволяє підключати плагіни для додавання певної функціональності, що разом із вдалою архітектурою дозволяє конструювати практично будь-які проекти.

Головні переваги

- Перш за все - простота. Створення не вимагає практично ніяких навичок у програмуванні чи дизайні. Публікація контенту по складності не перевищує складність друкування документів на принтері. Про те як налаштовувати або використовувати написані тисячі публікацій на інших веб-ресурсах. На будь-яке запитання вже є відповідь.
- Безкоштовні можливості. Тисячі тем дизайну для веб-сайту є у вільному доступі, та встановлюються в один клік, плагіни для розширення функціоналу — за все це не потрібно платити ні копійки і все це можна знайти прямо в панелі керування.
- Підтримка з боку величезного комп’юніті користувачів. Спільно з тисячами волонтерів-ентузіастів та розробників у всьому світі вони до сих пір беруть участь в постійних поліпшеннях та оновленнях для системи.

Недоліки WordPress:

- Обмежені можливості розширення функціональності під окремі потреби. Це створено спеціально, аби не робити із системи солянку із заплутаного коду. Веб-сайт легко створити, використовуючи цю систему, але також легко і зіпсувати встановивши тонну непотрібних плагінів, які скорші зе все тільки сповільнюють роботу веб-сайту чим надають додатковий функціонал. Великий процент відео-уроків, статей і гайдів записана користувачами, які не мають повноцінних навичок у веб-розробці. Як

приклад, вони можуть показати як зробити ту або іншу штуку чи функціональність, але не завжди попереджають, що все це може перезаписатися після оновлення версії движку або встановлення нової теми для дизайну. Це все може зламати існуючий веб-сайт. Увага, це перш за все що повинно бути у вас на думці коли плануєте редагувати вихідний код.

1.5.2. Joomla

Joomla є ще одною безкоштовною системою для управління контентом веб-сайтів. «Движок» має відкритий вихідний код, як і більшість подібних систем. Розвиток цієї системи почався сім років тому, і користується популярністю у користувачів і дотепер.

Joomla — займає другу сходинку за популярність систем менеджменту контенту веб-сайтів. Як і WordPress, вона безкоштовна, проста, доступна і надійна. Система володіє багатьма якостями WordPress. Головна відмінність — вищий порог входження для розробки веб-сайту. Її не так просто встановити, як WordPress, налаштувати і наповнити контентом.

Головні переваги

- Для сисетми існують багатофункціональні плагіни для створення інтернет-магазинів різної складності: VirtueMart, Joomshopping, Tienda і т.п.. Joomla вважається набагато більш підходящим вибором, якщо потрібно розробити інформаційний веб-ресур. Створити веб-саторінку на Joomla простіше і швидше, ніж на WordPress. Сайт можна оптимізувати без додаткових плагинів, оскільки це вже зроблено за вас. Користувач з легкістю може прописати мета-теги і налаштувати URL.
- В Joomla є кешування. Це значною мірою прискорює завантаження сторінок та підвищує позиції сайту у пошукових системах.
- Сайт на Joomla можна редагувати, не заходячи в адміністративну панель.

Недоліки CMS Joomla

Об'єктивно недоліки у Joomla такі ж, як і у WordPress — занадто багато навчальної інформації, серед якої даремні і навіть шкідливі поради; трохи обмежені можливості (хоч і не так обмежені, як у WordPress), велика кількість бажаючих знайти слабкі місця в коді і навчитися зламувати сайти на цій CMS.

1.5.3. Wix

Це умовно безкоштовна платформа, яка дозволяє створити веб-сайти. Wix надає готові професіональні шаблони дизайну і HTML5-редактор, який працює за принципом Drag-and-Drop. Шаблони налаштовуються у таких категоріях:

- додавання нових функцій та медіа;
- можливість зміни стилю;
- можливість зміни кольорів;
- можливість зміни тексту;
- додавання фонових зображень;
- додавання кнопок та інше.

Існує колекція безкоштовних зображень, форм і іконок.

Wix працює по бізнес-моделі freemium, пропонуючи можливість створити сайт безкоштовно і розвивати його, збільшуючи функціонал. Наприклад, тарифи Premium дозволяють підключити до сайту власний домен, прибрати банери Wix, додати онлайн-магазин, отримати додаткове місце для зберігання даних, купони на рекламу та інше.

Усередині редактора також інтегрований App Marker, в якому представлені додатки, створені різними компаніями з використанням автоматизованої технології веб-розробки Wix. App Market пропонує безкоштовні та платні програми і дозволяє інтегрувати на сайт такі функції як галереї фотографій, блоги, плейлисти, онлайн-спільноти, розсилки електронних листів і файлові менеджери. В App Market можна знайти готові рішення від Google, Instagram, LiveChat, Shopify та інших провідних компаній.

Ключові переваги Wix:

- безкоштовний хостинг;
- підключення власного домену;
- оптимізація для мобільних пристроїв;
- додавання зовнішнього HTML-коду;
- захист сторінок;
- модулі блогу, інтернет-магазину, списку, галереї зображень,
- відео та аудіо;
- постинг в соціальних мережах;
- блоки статистики.

Wix хвалять за зовнішню привабливість, яка робить роботу з конструктором приємною. Але це далеко не всі переваги сервісу. Також можна відзначити:

- адаптивність шаблонів;
- зручний інтерфейс візуального редактора, з яким зможе швидко розібратися навіть новачок;
- магазин додатків з двома сотнями віджетів;
- величезна база знань, яка допомагає розібратися з інтерфейсом конструктора;
- можливість додавання коду HTML на сторінки;
- єдина панель управління інтернет-магазином, через яку можна відстежувати замовлення, отримувати платежі і управляти доставкою;
- повна статистик по сайту — кількість переглядів сторінок, джерела трафіку, позиція в пошуковій видачі і т.д.

Якщо повернутися до зовнішньої складової конструктора, то можна ще раз відзначити що шаблонів на Wix дійсно багато. Вони грамотно розподілені за категоріями і типами сайтів, плюс підтримується пошук по ключевим словам і фільтри «Нові», «Популярні», «Порожні».

Перш ніж вибрати макет, можна подивитися, як він буде виглядати на моніторі та на екрані мобільного пристрою. Якщо жоден з варіантів дизайну не

підійшов, можна відфільтрувати пропозиції, використовуючи розділ «Порожні», і створити власний шаблон з нуля. На Wix можна створювати односторінкові сайти. Готові шаблони для Лендінзі знаходяться в одноймен-ному розділі.

Головна претензія до Wix - високі ціни і незрозумілий поділ преміум-планів. Connect Domain, після оплати якого на сайті зберігається реклама конструктора, навряд чи комусь потрібен; eCommerce для створення інтернет-магазину не пропонує нічого надприродного, а ціну має таку, ніби сам збере сайт за день.

Проблемою для веб-майстрів початківців може стати перевантажений інтерфейс. Його легко пояснити функціональністю конструктора, але на освоєння сервісу знадобиться не один день: треба буде не тільки розібратися з тим, що де знаходиться, але й навчитися правильно застосовувати інструменти. На Wix можна зіпсувати шаблон, тому що можливостей для редагування у користувача дуже багато.

Малоінформативним залишається розділ «Підтримка». Опис того, як виконуються в редакторі основні дії знайти можна, але без скріншотів буває важко зрозуміти, куди натискати. Можливо, це марна зачіпка, але якщо сервіс позиціонується як конструктор для новачків, то логічно очікувати максимально зрозумілого представлення інформації.

1.5.4. 1С-Бітрікс

«1С-Бітрікс» — популярна комерційна CMS від російських розробників. Створювалася в першу чергу для високонавантажених проектів: інформаційних порталів, інтернет-магазинів, сайтів великих компаній і державних організацій. Найбільшого поширення набула в електронній комерції.

В архітектурі продукту реалізований принцип MVC, поділ логіки від подання. Управління структурою відбувається за допомогою Інфоблоки, вони порівнянні з базою даних, кожен інфоблок це якась коробка яку можемо налаштовувати саме під ту інформацію яку хочемо в ній зберігати. Вся

візуальна частина знаходиться окремо що дає гнучкість в управлінні дизайном сайту.

Розширення в 1С-Бітрікс як безкоштовні, так і платні. Всі розширення знаходяться в магазині Маркеплейс: купивши рішення ми отримуємо підтримку розробників і зворотний зв'язок.

Технічна підтримка Бітрікс оперативно вирішує завдання, є документація і навчальний матеріал з управління двигуном як для користувача, так і розробника. 1С-Бітрікс має свою спільноту розробників, форуми, блоги, де допоможуть вирішити ті чи інші завдання.

Високий ступінь захисту самої платформи забезпечується за рахунок вбудованого модуля проактивного захисту. В неї входить двохетапна авторизація, захист сесії, обмеження по IP і багато додаткових фішок, які не дадуть просто так зламати сайт.

Розділи і сторінки сайту мають зрозумілу структуру, реалізовано за принципом папок і файлів, як на комп'ютері. Завдяки технології Ермітаж є можливість редагувати сторінки, змінювати налаштування сайту прям через візуальну частину сайту, не заходячи в адміністративний розділ, що у багато разів полегшує її адміністрування.

Переваги 1С-Бітрікс

- Коробкова версія включає всі необхідні модулі для роботи з сайтом. У кожній редакції продукту свій набір модулів — чим дорожче, тим ширше функціонал.
- Зручна і інтуїтивно зрозуміла адміністративна панель дозволяє легко управляти контентом сайту.
- Система гнучка і кастомізуєма — можна створити і підтримувати проект практично будь-якої складності і будь-яких масштабів. Регулярні оновлення забезпечують стабільну роботу і відкривають нові можливості.
- Є магазин готових рішень «Маркетплейс 1С-Бітрікс», де можна знайти безліч розробок, які не просто розширюють штатні можливості системи,

так і дозволяють створити готовий сайт в найкоротші терміни без залучення технічних фахівців.

- Модуль «Пошукова оптимізація» відкриває широкі можливості для просування в пошукових системах.
- Інтеграція з «1С: Підприємство». Можна відстежувати всі покупки через сайт компанії, а статус замовлення — в особистому кабінеті. Також можлива синхронізація каталогу через «1С» - вся інформація про товари буде оновлюватися автоматично [10].
- Сайти, створені на платформі проходять моніторинг якості. Проходження моніторингу хоч і не дає стовідсоткової гарантії, але дозволяє переконатися, що в процесі розробки не було допущено критичних помилок.
- Система «Композитний сайт» — унікальна, запатентована технологія, яка прискорює завантаження ресурсу за рахунок оптимізації процесу доставки контенту і створення кешованих копій сторінок.

Головні недоліки

- Вимогливість до ресурсів. Сервер повинен бути досить потужним. Однак багато хостерів пропонують доступні тарифи для «1С-Бітрікс», що знімає проблему.
- Надмірність коду і складна архітектура. Для підтримки сайту і доробок потрібен програміст, який вивчив систему.
- Для впровадження і зв'язку з «1С: Підприємство» необхідно залучати професіоналів.
- Вартість розробки і підтримки. Крім вартості самого «движку», необхідно врахувати, що вартість роботи програмістів на «1С-Бітрікс» зазвичай вище середнього.
- Платне оновлення системи. Термін оновлення лише 1 рік.

- На «1С-Бітрікс» добре працюють складні, великі і високонавантажені проекти. Для маленького сайту краще вибрати CMS простіше — це дозволить заощадити гроші на розвиток продукту і сервісу.

1.5.5. Порівняння систем-аналогів

Порівнянння систем аналогів наведені у таблиці 2. Вимоги, за якими проводолось порівняння CMS систем:

- Легкість установки — система встановлена так, що навіть людина далека від програмування має змогу з легкістю установити програму. Не пред'являє особливих вимог до хостингу та працює на більшості ОС.
- Підтримка розробників — при певних умовах (наприклад, при певному пакеті) система надає можливість скористатися допомогою та отримати більш професійну розробку сайту. У майбутньому це забезпечить сайт замовленими оновленнями без втручання в код користувача.
- Безпека — гарантує надійність при користуванні системою: може включати в себе фаєрвол, антивірус та захист від DDOS атак.
- Простота освоєння системи — мануали та інструкції знаходяться у відкритому доступі, що не викликає особих проблем з освоєнням системи.
- Має безкоштовний пакет — система пропонує безкоштовний набір інструментів для створення простого WEB-сайту.
- Не потребує багато ресурсів — не обов'язково сервер має бути потужним для підтримки сайту у мережі Інтернет, не витрачає багато пам'яті.
- Підтримка HTML/CSS — можливість для користувачів із знаннями HTML/CSS редагувати та доповнювати шаблони для отримання необхідного результату.
- Можливість використовувати шаблони — готові приклади дизайну на різні тематики, які можна використати у своєму сайті.
- Багатофункціональність — можливості сайта можна розширити, скориставшись можливістю підключення плагінів.

- Простота використання — зручний та інтуїтивно зрозумілий інтерфейс, який дозволяє легко користуватися можливостями системи.

Табл. 2. Порівняння систем-аналогів

Системи **Вимоги**	**WordPress**	**Joomla**	**Wix**	**1С-Бітрікс**
Простота установки	+	+	+	—
Підтримка користувачів	+	+	—	+
Безпека	+	—	+	+
Простота освоєння системи	+	—	—	+
Безкоштовність	+	+	+	—
Не потребує багато ресурсів	+	—	+	—
Підтримка HTML/CSS	+	+	+	+
Можливість використовувати шаблони	+	+	+	+

Багатофункціо-нальність	+	—	+	+
Простота використання	+	+	—	—
Вартість	**до 1000$**	**до 1000$**	**до 600$**	**до 3533$**

WordPress — задовольняє більшість вимог, але не забезпечує WEB-сайт безпекою, що є одним із головних критерієв вибору системи.

Joomla — дана система є досить дешевою, але має багато мінусів, які дають зрозуміти, що саме ця система нам не підходить; найголовніший мінус — це незабезпечення безпекою від спроб злому та DDOS атак.

Wix — хоч дана система і надає можливість створити сайт за мінімальну плату, та для повноцінного WEB-сайту треба купляти підписку, яка матиме необхідні інструменти, а це коштує досить неоправдано дорого. Також багато часу піде на вивчення системи, що збільшить час створення WEB-сайту.

1С-Бітрікс — оптимальна система для використання, але має занадто високу вартість при недотриманні деяких вимог.

1.6. Обґрунтування доцільності проектування й розроблення WEB-сайту кафедри «ЕБОП» НУХТ

Як було зазначено у п 1.3, існуюча панель управління кафедри «ЕБОП» НУХТ не має ті важливі критерії, які необхідні для того, щоб бути повноцінної системою управління контентом. З наведених у п.1.5 результатів порівняння існуючих альтернативних систем CMS видно, що вони є або занадто дорогими для кафедри, або не відповідають вимогам, які необхідні для створення WEB-сайту, перелік яких наведено у п.1.4.

Отже, проектування та розробка нової панелі управління кафедри «ЕБОП» НУХТ є доцільними, адже його створення передбачає вирішення основних проблем, які впливають на пошукову оптимізцію, зручність у подачі та форматуванні контентенту для кінцевого користувача.

1.7. Концептуальна модель

Проаналізувавши недолікі та сформувавши задачі на покращення цих процесів, було прийнято рішення додати деякі підпроцеси.

По перше, була додана можливість віднесення новини до певної категорії, що покращить структурованість та полегшить пошук новин за тематикою кінцевому користувачу веб-сайту.

По друге, додалась валідація даних прямо на формі, що дозволить перевіряти правильність даних, без відправки на сервер, так покращить якість контенту.

По третє, з’явиться повторна валідація даних на сервері, перед записом до БД, що дозволить трансформувати дані в більше кращий формат, для показу на веб-сайті, та провести оптимізацію зображент, якщо це потрібно. Ці правила можуть бути іншими, оскільки на формі ми можемо вказати дані в різних форматах, як наприклад розширення зображень, але при валідації на сервері ми можемо трансформувати в якийсь один формат, який буде найбільш підходящим для веб-сайту.

1.8. Постановка задачі

1.8.1. Призначення та цілі створення панелі управління

Основними користувачами панелі будуть користувачі, які мають спеціальне програмне забезпечення — «браузер», доступ до мережі Інтернет та матимуть відповідні права, які дозволятимуть доступ до панелі. В ній вони будуть мати можливітсть додавати, редагувати, видаляти контент та елементи сайту.

Призначення панелі управління кафедри «ЕБОП» НУХТ полягає у покращенні якості донесення інформації до користувачів (студентів, абітурієнтів, викладаців тощо) сучасними методами.

Основними цілями створення панелі управління є:

- спрощення управлінням контентом веб-сайту;
- покращення форм введення даних, які будуть на веб-сайті, спеціально для потреб кафедри;
- більш простий дизайн панелі, тільки потрібний функціонал.

1.8.2. Вимоги до створюваної панелі управління

Для повного функціонування панель управління повинна виконувати такі вимоги:

- Повинна мати форму авторизації, для надання доступу тільки користувачам які мають права на вхід.
- Повинна працювати на всіх сучасних браузерах та всіх ОС.
- Повинна працювати на ПК з слабким процесором та малою оперативною пам’яттю.
- Відносно швидко завантажуватися при повільному інтернет з’єднанні.
- В якості СУБД використовувати Firebase Realtime Datebase.
- Передача та отримання даних із серверу повинна бути у форматі JSON та по протоколу HTTP.
- Елементи інтерфейсу повинні мати простий вигляд та уніфіковану функціональність.
- Сторінки панелі повинні мати лаконічну структуру та бути не перенаватаженими контентом.

- Дизайн форм та процесів управління контентом має бути інтуітивним.
- Користувачі панелі повинні мати базове розуміння про HTML верстку.

1.8.3. Функції, які повинна виконувати панель управління

- Додавати, редагувати, видаляти новини.
- Додавати, редагувати, видаляти сторінки.
- Додавати, редагувати, видаляти зображення в галареї.
- Додавати, редагувати, видаляти вакансії.
- Додавати, редагувати, видаляти події.
- Додавати, редагувати, видаляти компанії.
- Додавати, редагувати, видаляти категорії для новин, сторінок, вакансій, зобарежень, подій.
- Додавати, редагувати, видаляти користувачів веб-сайту.
- Заблоковувати та розблоковувати користувачів веб-сайту.
- Сортування, фільтрацію та пошук новин, сторінок, зображень, вакансій, компаній, подій.

1.8.4. Вхідні та вихідні дані панелі управління

Вхідні дані:

- Логін та пароль користувача.
- Інформація про новини.
- Інформація про сторінки.
- Інформація про вакансії.
- Інформація про події.
- Інформація про компанії.
- Зображення для галереї.
- Інформація про категорії новин, сторінок, подій, вакансій, зображень.

Вихідні дані:

- Новини.
- Сторінки.
- Події.
- Вакансії.
- Компанії.
- Зображення.
- Категорії для новин, сторінок, подій, зображень, вакансій.
- Користувачі.

Розділ 2.

2.1. Перелік використаних технологій

- **CASE- засіб ALLFusion Process Modeler -** для побудови функціональної моделі.
- **CASE-засіб ALLFusion Erwin Data Modeler -** для побудови логічної та фізичної моделей бази даних та генерування SQL-кодів створення таблиць.
- **HTML** – мова розмітки для веб-сторінок.
- **CSS** – мова для створення таблиць стилів для HTML елементів.
- **JavaScript** – мова для програмування логіки клієнтської частини веб-сайту. За допомогою неї пишуться функції запитів до сервера, дії з інтерфейсом та анімацій.
- **Vue.js** – фреймворк для спрощення розробки елементів клієнтської частини веб-сайту.
- **Firebase Realtime Database** – база даних у форматі JSON, для збереження даних.
- **Firebase Authentication** – набір програмних інструментів для створення безпечного створення адміністраторів, які отримають доступ до панелі управління веб-сайтом.
- **Firebase Lambda Functions** – платформа яка дозволяє створювати функції маніпулювання з базою даних. Є основною частиною backend логіки веб-сайту.
- **GIT** – система контроля версій, яка дозволяє полегшити процес розробки програмного продукту

2.2. Проектування бази даних

Опис таблиць:

User (користувач):

Таблиця відповідає за користувачів веб-сайту. Користувач, має більше можливостей при роботі з веб-сайтом, та, якщо має спеціальні права доступу, може керувати контентом в панелі адміністратора.

- **user_id** – ідентифікатор користувача;
- **email** – електронна пошта користувача;
- **password** – пароль користувача;
- **is_blocked** – чи заблокований користувач;
- **role_id** – ідентифікатор ролі;
- **name** – нікнейм або ім’я користувача.

User_role:

Таблиця відповідає за опис ролей.

- **role_id** – ідентифікатор ролі;
- **name** – назва ролі.

Operation_grant:

Таблиця описує які операції може здійснювати роль над таблицею.

- **role_id** – ідентифікатор ролі;
- **operation_type** – тип операції яка може здійснюватися: CREATE, READ, UPDATE, DELETE;
- **table_name** – назва таблиці до якої ця операція може бути здійснена.

Дана структура має переваги в тому сенсі, що кожен користувач, може взяти на себе деяку роль. Наприклад один користувач може писати новини, інший публікувати вакансії. Розмежування на ролі дає користувачу деяку область панелі адміністратора, в якій він може працювати, тим самим не заважаючи іншим. Або, як варіант, користувач з лихим наміром не зможе, наприклад, видалити весь контент веб-сайту, що підвищує безпеку.

News (новини):

Дана таблиця відповідає за новини, які будуть опубліковані на веб-сайті.

- **title** – назва новини;
- **content** – HTML розмітка контенту новини, є основним контентом який буде показаний на сторінці новини;
- **image** – посилання на зображення, яке буде використане як прев'ю новини;
- **user**_id – ідентифікатор користувача, який додав цю новину;
- **category_id** – ідентифікатор категорії, до якої відноситься новина;
- **isVisible** – чи опублікована новина на веб-сайті. Це значення потрібне для позначення новини як чорновик. В деяких випадках новину потрібно публікувати не відразу, або вона ще не до кінця написана, якщо позначка стоїть, новина буде показана в панелі управління новинами, але не на веб-сайті;
- **created** – дата та час, яка показує коли буде створена новина.

Comments:

Таблиця містить в собі коментарі, під новинами.

- **comment_id** – ідентифікатор коментаря**;**
- **text –** текст коментаря;
- **datetime** – дата та час публікації коментаря**;**
- **news_id –** ідентифікатор новини, до якої відноситься коментар**;**
- **author_name –** ім'я автора. У випадку якщо коментар хоче залишити незареєстрований користувач, йому буде запропоновано ввести його ім'я, якщо користувач зареєстрований та увійшов під своїм логіном, це поле буде пусте**;**
- **user_id –** ідентифікатор користувача, який додав коментар. Якщо коментар буде залишений не зареєстрованим користувачем, це поле буде пусте**.**

News_category (категорії новин):

Таблиця містить в собі категорії новин.

- **category_id –** ідентифікатор категорії**;**
- **name –** назва категорії**.**

Page (сторінки):

Таблиця сторінки, які будуть показані на веб-сайті.

- **page_id** – ідентифікатор сторінки**;**
- **content –** контент сторінки;
- **isVisible** – видимість сторінки, якщо мітка увімкнена, то сторінка опублікована на сайті і всі можуть її переглядати, якщо вимкнена, сторінка доступна як чорновик в панелі управління веб-сайтом**;**
- **created –** дата та час публікації сторінки**;**
- **user_id –** ідентифікатор користувача, який її створив**;**
- **category_id –** ідентифікатор категорії сторінки**.**

Page_category (категорії сторінок):

Таблиця містить в собі категорії сторінок.

- **category_id –** ідентифікатор категорії**;**
- **name –** назва категорії**.**

Gallery (галерея зображень):

Таблиця галереї, в якій зберігаються зображення.

- **image_id** – ідентифікатор зображення**;**
- **description –** опис зображення;
- **user_id –** ідентифікатор користувача, який її завантажив**;**
- **category_id –** ідентифікатор категорії зображення**.**

Gallery_category (категорії зображень):

Таблиця містить в собі категорії зображень з галереї.

- **category_id –** ідентифікатор категорії**;**

- **name –** назва категорії**.**

Event (події):

Таблиця подій. В ній зберігаються події які відбуваються на кафедрі. Наприклад, ярмарка вакансій, день факультету і тд. і тп.

- **event_id** – ідентифікатор події**;**
- **title –** назва події;
- **image_url –** посилання на зображення події**;**
- **location –** місце проведення події**;**
- **start_datetime –** дата і час початку**;**
- **end_datetime –** дата і час закінчення**;**
- **category_id –** ідентифікатор категорії**;**
- **user_id –** ідентифікатор автора який опублікував подію**.**

Event_category (категорії подій):

Таблиця містить категорії подій.

- **category_id –** ідентифікатор категорії**;**
- **name –** назва категорії**.**

Vacancy (вакансії):

Таблиця з вакансіями для студентів.

- **vacancy_id** – ідентифікатор вакансії**;**
- **title –** назва вакансії;
- **content –** опис вакансії**;**
- **image_url –** зображення вакансії**;**
- **created –** дата і час публікації вакансії**;**
- **contact_email –** контактна електронна пошта**;**
- **contact_phone –** контактний телефон**;**
- **company_id –** ідентифікатор компанії, від якої прийшла вакансія**;**

- **category_id –** ідентифікатор категорії**;**
- **user_id –** ідентифікатор автора який опублікував вакансію**.**

Vacancy_category (категорії зображень):

Таблиця містить категорії вакансій.

- **category_id –** ідентифікатор категорії**;**
- **name –** назва категорії**.**

Company (вакансії):

Таблиця з вакансіями для студентів.

- **company_id** – ідентифікатор компанії**;**
- **name –** назва компанії;
- **image_url –** посилання на зображення чи логотип компанії**;**
- **description –** опис компанії**;**
- **contact_email –** контактна електронна пошта**;**
- **contact_phone –** контактний телефон**;**
- **address –** адреса компанії**;**
- **user_id –** ідентифікатор автора який опублікував компанію**.**

Структура *Компанія – Вакансія – Категорія вакансії* дає гарні можливості по зручній навігації по вакансіями, Що полегшує пошук потрібної вакансії від потрібної компанії і навпаки. Логічну та фізічну модель бази даних наведено в додатку Б на рис. Б1 та Б2.

На основі створеної моделі генеруємо базу даних в Firebase Realtime Database (використавши спеціальний плагін для генерації у правильному для цієї бази даних форматі JSON), перед цим створивши порожню базу даних. Генерація структури БД на основі створеного відбувається після натиснення кнопки Generate. Після цього ми отримали SQL код для створення бази даних. Оскільки наша база даних представляє собою JSON структуру і таблиця може

мати мати будь-яку структуру нам потрібно зробити спеціальні JavaScript класи які будуть мати правильну структуру та валідувати дані. Для цього можно використати спецільний інструмент, який конвертує SQL код в JavaScript клас. Приклад формування стурктури наведено у додатку Б на рис. Б3.

Формуємо все інші класи за таким же принципом.

Потрібно розуміти, що такий тип бази даних не ставить ніяких обмежень на структуру даних, все залежить від налаштувань класу. Всю валідацію на тип ми будемо робити перед занесенням до бази даних на сервері.

2.3. Створення інтерфейсу користувача

При розробці інтерфейсу користувача спочатку створимо спеціалний файл з розширенням .vue, який є спеціальним форматом файлу розмітки компонентів фреймворка Vue.js, та внесемо в нього дизайн сторінки управління новинами в панелі адміністратора. Засобами HTML та Vue.js створимо розмітку для сторінки. Приклад розмітки наведено у додатку В на рис В1.

Після цього ми отримаємо перший результат сторінки. Вигляд наведено у додатку В на рис В2. Поки що ніяких даних на ній немає, оскільки для цього ми повинні створити запит на отримання даних з БД.

Для того, щоб ми мали можливість додавати дані до БД потрібно запрограмувати кнопку «ДОДАТИ НОВИНУ». Знаходимо кнопку в розмітці і додаємо їй подію натискання на неї. Приклад додавання обробника події натискання наведено у додатку В на рис. В3.

Функція *openDialogForNewRecord* буде запущена в момент натискання на кнопку. Після цього нам потрібно створити та запрограмувати логіку виконання програми після натискання. Приклад готової функції наведено в додатку В на рис. В4.

В коді функції ми ставимо мітку що нам потрібно саме додати нову новину. Надалі ми будемо використовувати іншу мітку, вона буде

символізувати що ми редагуємо новину. Це потрібно для того щоб використовувати одну форму для введення даних, що зменшує обсяг коду та розмітки для її створення.

Також робимо програмну очистку мітки, як відповідає за те чи чорновий варінт новини чи ні. Після цього відкриваємо вікно з формою.

Далі нам потрібно створити саме вікно та форму в ньому куди будуть додаватися дані. Приклад розмітки вікна та форми наведено у додатку В на рис. В5 та В6.

Ми створили форму в окремому файлі, ми буде вважати що це окремий компонент, оскільки він буде використаний як форма для додавання новини, так і для її редагування. Компонент використовується в двух місцях, що звільнює нас від написання додаткового компонента з дуже схожим кодом та розміткою.

Після створення ми повинні підключити даний компонент до нашої сторінки, для його використання.

```
import { NavTabsCard, NavTabsTable } from "@/comp
import DialogWindow from "./DialogWindow";
import { MiniLoading, MainLoading } from "../../c
import { News, Gallery, Categories } from "@/serv
```

Рис. 3. Імпорт компонента з формою

Оскільки ми використовуємо фреймворк Vue.Js, ми можемо використовувати цей компонент як HTML-тег. Тому вставляємо прямо в розмітку.

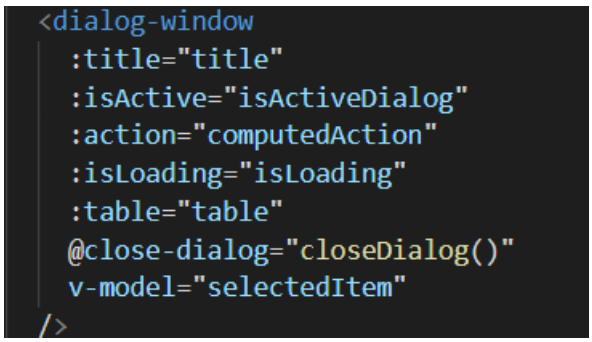

Рис. 4. Приклад доданого в розмітку вікна з формою

Раніше в функції openDialog, ми присвоїли змінній *isActiveDialog* значення *true*, що означає, вікно з формою відкрите. Передаємо цю змінну в наш компонент, щоб він з'явився на єкрані. Приклад відкритого вікна наведено у додатку В на рис В7.

Після цього нам потрібно запрограмувати кнопку додавання новини. Знаходимо кнопку «ЗБЕРЕГТИ» в коді, та додаємо їх обробник натискання.

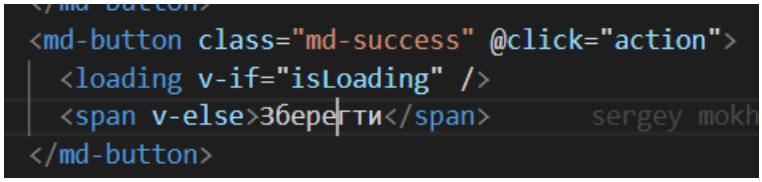

Рис. 5. Приклад доданого обробника подій на натискання

Оскільки наша форма буде використана як для додавання так і для редагування новини, назвемо метод який буде запущено після натискання: *action*. Цю функцію ми передамо в наш компонент. Приклад передачі методу був на зображенні де ми підключали компонент в розмітку.

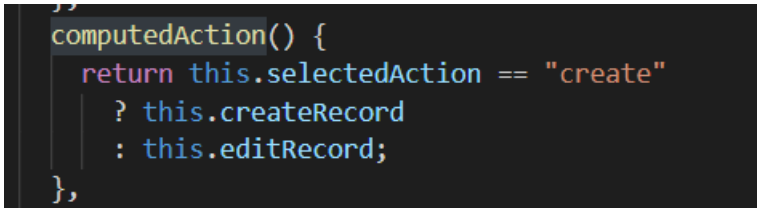

Рис. 6. Приклад методу який повертає функцію яка буде запущена при натисканні кнопки зберегти

Як бачимо на зображенні, якщо ми натиснули на кнопку додати новину, то метод для збареження буде *createRecord*, якщо на кнопку редагування новини, буде запущений метод *editRecord*. Приклад коду методу *createRecord* наведено у додатку В на рис. В8.

Спочатку беремо кожні значення з форми, та передаємо ці значення в функцію яка відправить запит до серверу на додавання.

Якщо новина додана то покажемо користувачу повідомлення що новина була успішно додана, якщо виникла якась помилка, покажемо повідомлення з помилкою.

Фреймворк Vue.js автоматично валідує введені дані на формі, тому користувач буде відправляти валідні дані, використовуючи інтерфейс який ми створили. Але для повної безпеки, повторна валідації є на стороні сервера.

Спочатку ми формуємо правильну структуру для запиту. Для цього над потрібен токен користувача, заголовки з типом даних, та в тіло запиту ми вставляємо наші дані з форми. Приклад коду функції наведено у додатку В на рис. В9.

Токен користувача створюється коли користувач входить під своїм логіном та паролем, та зберігається в нього на комп’ютері. Це звичайний набір символів який є унікальним для кожної сесії користувачів та має розмір від 32 до 256 символів. Він потрібен для того, щоб тільки користувач міг додавати новини до бази даних. Якщо у користувача неправильний токен або його немає, запис до бази даних не відбудеться так само як і доступ до панелі адміністратора.

Після формування правильної структури запиту, ми робимо HTTP POST[4] запит на адресу, які відповідає за роботу з таблицею новин. Приклад функції яка опрацьовує HTTP POST[4] запит наведено у додатку В на рис. В10.

Після запуску функції на стороні серверу, ми відразу перевіряємо токен, на те вірний він чи ні, та чи є в цього користувача права на таку операцію з базою даних. Приклад коду функції наведено у додатку В на рис. В11.

Якщо права дозволяють виконати операцію, ми отримуємо дані з запиту, та викликаємо функцію яка додасть їх в базу даних, приклад функції, яка додає валідує дані та додає до бази даних наведено у додатку В на рис. В12.

Функція отримує дані, валідує для їх типи, щоб вони співпадали з типами полів в базі даних та робить запит.

Для створення підключення до бази даних нам потрібно написати дві команди.

```
const admin = require('firebase-admin');
admin.initializeApp();
```

Рис. 7. Приклад підключення до бази даних

Наша база даних, та функції знаходяться на одній платформі, отже нам не потрібно прописувати логін, пароль, та інші дані для підключення як це було в SQL базах даних. Після ініціалізації підключення до БД звернувшись до змінної admin (в якій будуть функції для роботи з БД) ми зможемо робити будь-які запити до бази.

Після виконання функції ми дані будуть записані до БД, приклад доданих даних наведено у додатку В на рис. В13.

Після успішного запису функція повертає результат до браузера користувача. Далі на сторінці новин буде показане повідомлення про успішне додавання новини та сама новина. Приклад показу новини наведено у додатку В на рис. В14.

Після додавання, ми можемо редагувати, позначувати як чорновик, та видаляти новини. Для прикладу, додамо ще декілька новин.

Припустимо, що додана новина про показникі питної води має неповну назву, відредагуємо її, натистувни на кнопку редагування з іконкою олівця:

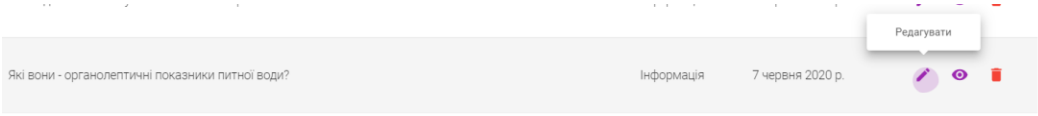

Рис. 8. Кнопка редагування новини

Після натискання буде відкрите вікно з формою в якій будут заповнені дані про новину. Приклад вікна з даними наведено у додатку В на рис. В15.

Змінюємо назву у новини, як показано на зображенні нижче:

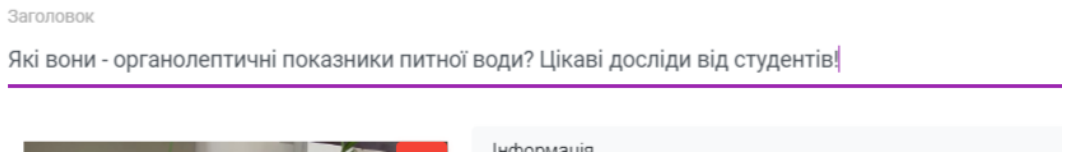

Рис. 9. Приклад зміненої назви новини

Після натискання кнопки «ЗБЕРЕГТИ», приклад фукнції наведено у додатку В на рис. В16., та відправляємо запит на сервер по тому ж самому алгоритму, щ обув описаний для додавання новини, але на цей раз робимо на HTTP POST[4] запит, а HTTP PUT[4], який створений для запитів на редагування даних, приклад коду функції наведено у додатку В на рис. В17.

На стороні сервера створена функція яка реагує на цей запит та знову ж таки, перевіряє чи вірний токен та права, витягує дані із запиту та викликає функцію яка валідує дані та записує їх до бази даних. Приклад фукнцій наведений у додатку В на рис. В18 та В19.

Коли запит відбувся, та дані були успішно записані до БД, повертаємо результат до браузера користувача та показуємо повідомлення про успішне виконання операції. Приклад повідомлення та відредагованої новини наведено у додатку В на рис. В20.

Для видалення новини, натискаємо на кнопку видалення, як показано на зображенні нижче:

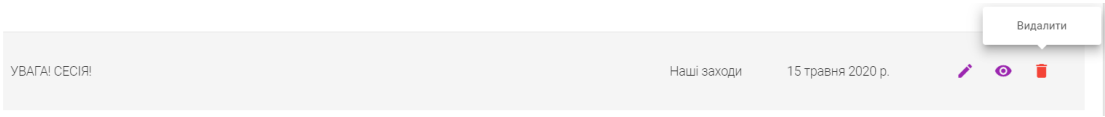

Рис. 10. Приклад кнопки видалення новини

Після натискання на кнопку видалення запускаємо функції яка покаже вікно із підтвердженням видалення, та у разі згоди відправить HTTP DELETE[4] запит на сервер, який відповідає за видалення даних. Після успішного виконання запиту на видалення буде показане повідомлення про успішну операцію, або у разі помилки буде показане повідомлення з помилкою. Приклади коду та результату наведені у додатку В на рис. В21 – В24.

Якщо новин багато, і потрібно видалити, позначити як чорновик або редагувати, дуже корисним буде можливість пошуку новин. Для цього на сторінці додамо поле для пошуку. Приклад коду ромізтки та вигляду поля наведено у додатку В на рис. В25 та В26.

Для створення пошуку по новинам, нам потрібно зробити функцію, яка буде слідкувати за контентом в полі пошуку, якщо користувач починає туди вводити слово, нам потрібно відфільтрувати новини в яких є це слово.

Для цього на допомогу знову приходить фреймворк Vue.js, який дає готовий набір функціональності для такої задачі. Приклад коду, який слідкує за змінами в полі наведено у додатку В на рис. В27.

Після того як користувач почне щось вводити, функція буде запущена автоматичко засобами фреймворку, та надішле HTTP GET[4] запит на сервер за параметрами для фільтрування. Приклад коду функції та запиту наведено у додатку В на рис. В28 та В29.

Для зменшення наватаження на браузер та ПК користувача, всі «важкі» дії з даними відбуваються на сервері. Відправляється запит на сервер з словом по якому потрібно відфільтрувати новини, там отримуємо готовий набір відфільтрованих даних. Якщо таку дію по фільтрації виконувати на стороні браузера, це може негативно сказатися на швидкодії панелі управління та ПК

користувача. Приклад функцій на отримання запиту, обробки та запису до БД наведені у додатку В на рис. В30 – В32.

Кожен запит на отримання, фільтрацію, додавання, зміну, видалення перевіряється на помилки та будь-які винятки обробляються. Для кожного винятку було написано окреме повідомлення яке буде показано. Приклад повідомлень наведено у додатку В на рис. В33.

2.4. Інструкція користувача

2.4.1 Форма авторизації

Перейшовши за посиланням, за яким буде знаходитися панель адміністратора, користувачу буде запропонова увійти під його логіном та паролем. В нашому випадку, логіном виступає електронна пошта при реєстрації на сайті.

В базі даних буде створений користувач з правами *SUPER_ADMIN[7]*, який відразу буде мати всі права в панелі адміністратора. Він зможе створювати інші акаунти, які зможуть мати доступ до панелі адміністратора.

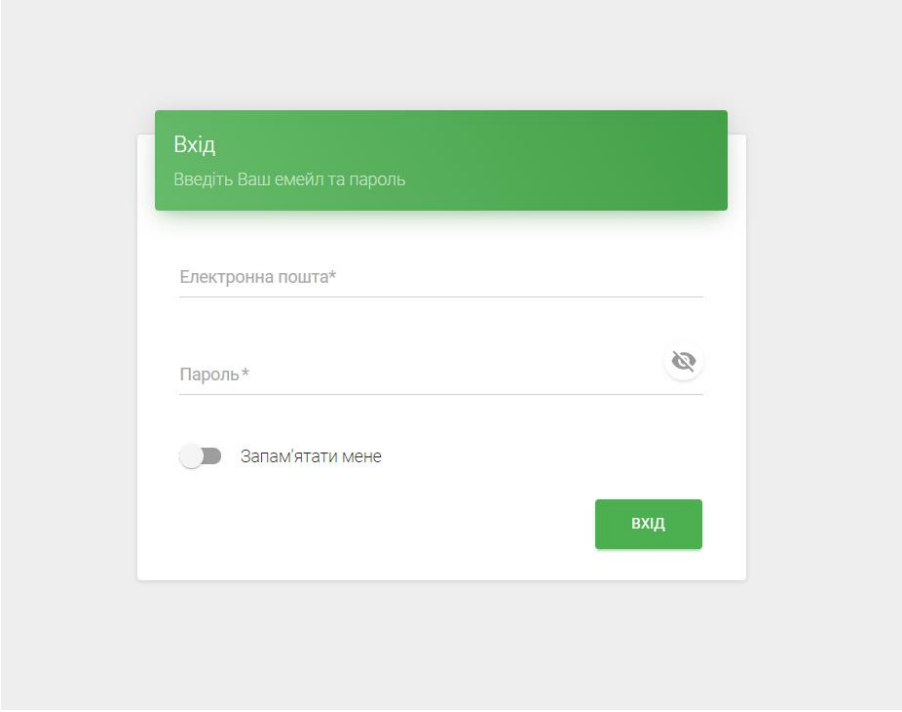

Рис.11. Приклад форми авторизації

На формі присутні два поля: електронна пошта та пароль. Ці дані адміністратор указує при додаванні нового користувача, який буде мати доступ до панелі адміністратора. Далі йде перемикач, яки дозволяє зберігати сесію користувача після закриття вікна чи браузера. Якщо він вимкнений, після закриття браузера користувач буде повинен знову увійти до панелі знову.

Якщо користувач ввів неправильні дані, йому буде показане повідомлення, що дані неправильні. Після успішного входу його буде перенаправлено на головну сторінку панелі управління. Після цього йому буде доступне меню з посиланнями на сторінки налаштувань контенту веб-сайту.

2.4.2 Управління новинами

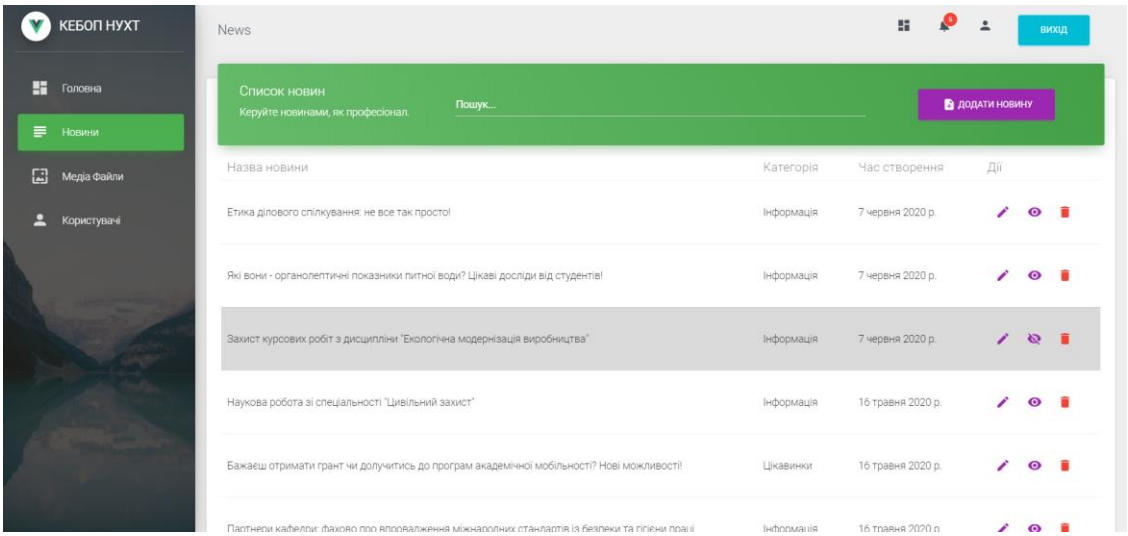

Рис. 12. Приклад сторінки з управління новинами

На сторінці можно побачити список новин, якщо новина позначеня як чорновик, вона буде підсвічена сірим кольором, це означае що вона не опублікована на веб-сайті. Для зняття позначки чорновкика, потрібно натиснути на іконку зачеркнутого ока, після цього новина буде опублікована на сайті. Натистувши другий раз на цю іконку, новина буде позначена як чорновик знову. Приклад повідомлення про успішне зняття позначки чорновка наведено у додатку Г на рис. Г1.

Для додаваня новини потрібно натиснути на кнопку «ДОДАТИ НОВИНУ», після чого відкриється вікно з формою додавання новини. Приклад форми додавання новини наведено у додатку Г на рис. Г2.

На формі ми бачимо поле Заголовок, це назва новини, яка буде відображатися на веб-сайті.

Кнопка «ОБРАТИ ЗОБРАЖЕННЯ» відповідає за завантаження зображення, яке буде показане як основне зображення новини.

Під кнопкою завантаження зображена є перемикач, який пропонує зберегти новину як чорновик.

Справа ми бачимо випадаючий список для вибору категорії новини.

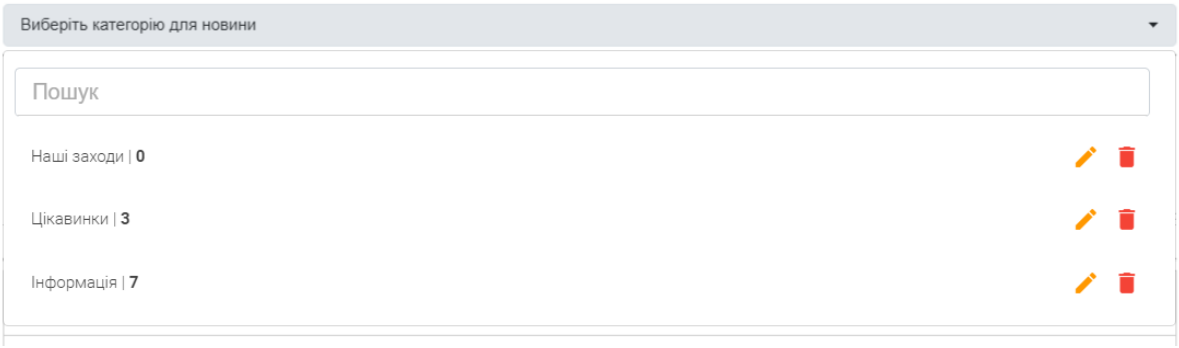

Рис. 13. Приклад відкритого списку з категоріями

Якщо потрібної категорії немає в цьому списку, ми можемо додати нову, для цього потрібно в поле пошуку ввести назву категорії яку ми хочемо додати:

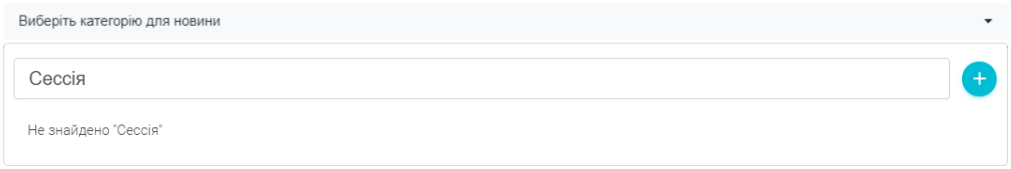

Рис. 14. Приклад введеня нової категорії

Після чого ми бачимо, що такої категорії не знайдено, але в з’явилась кнопка, натиснувши на яку, категорія з такою назвою буде додана до списку категорій. Приклад доданої категорії наведено в додатку Г на рис. Г3.

Як бачимо, категорія була додана до списку. Номери справа, показують кількість новин в яких указана ця категорія. Після цього ми можемо вибрати нову категорію.

Далі ми бачимо поле, в якому ми можемо ввести короткий опис новини, який буде показаний на веб-сайті в списку новин.

Далі йде візуальний редактор в якому ми будемо писати весь контент новини. Приклад вигляду редактора наведено в додатку Г на рис. Г4.

В ньому ми можемо форматувати текст, вставляти посилання, зображення, відео ті цитати. Редактор є мінімалістичним та включає в себе тільки основні функції для роботи з форматуванням, що не перенавантажує інтерфейс.

Після того як ми заповнили новину, ми можемо її зберегти, натиснувши на кнопку «ЗБЕРЕГТИ». Тепер новина з’явилась в списку новин.

Припустимо що ми написали неповну назву, та вказали неправильні терміни в тексті новини, для зміни новини, нам потрібно натиснути на кнопку редагування, з іконкою олівця.

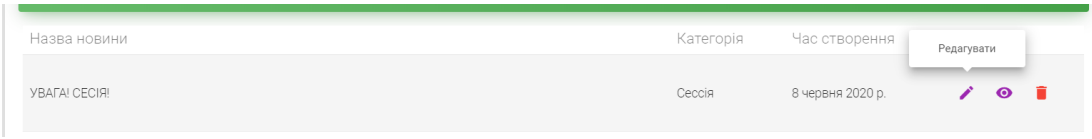

Рис. 15. Приклад іконки редагування

Натиснувши на неї, відкриється вікно редагуванни новини:

Змінивши назву новини, там дати проведення заліков з іспитами ми можемо зберегти зміни, натиснувши на кнопку «ЗБЕРЕГТИ»

Після цього ми можемо побачити змінену назву новини, та повідомлення про успішне оновлення запису в базі даних.

Якщо новина неправильна, або її потрібно прибрати, ми можемо її видалити:

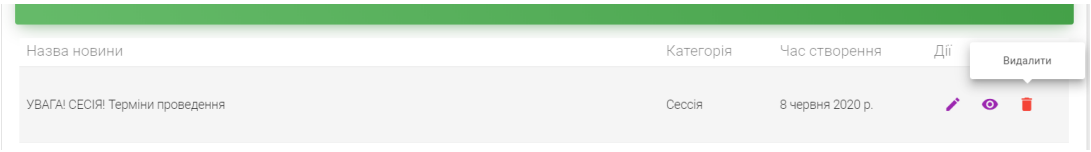

Рис. 16. Приклад кнопки видалення новини

Натиснувши на кнопку видалити, буде відкрите вікно з підтвердженням дії. Приклад вікна наведено у додатку Г на рис. Г6.

Після підтвердження видалення, ми отримаємо повідомлення про успішне видалення та новина буде видалені зі списку новин.

Якщо новин дуже багато, а потрібно видалити або редагувати, ми можемо скористатися пошуком. Поле пошуку знаходиться зліва від кнопки додати новину. Приклад поля пошуку наведено у додатку Г на рис. Г8.

Почавши введеня слова чи фрази для пошуку, новини відразу почнуть шукатися. Приклад знайдених новин наведено в додатку Г на рис. Г9.

Ввівши слово для пошуку, ми відразу отримуємо новини, в яких це слово є. Якщо ввести слово чи фразу, якого немає нівяких новинах, нам буде показане повідомлення, що нічого не знайдено. Приклад повідомлення що пошук не дав результатів наведено в додатку Г на рис. Г10.

2.4.3 Управління галереєю

Перейшовши на сторінку управління галереєю, ми побачимо всі зображення які були завантажені до неї. Для завантаження зображень, натискаємо на кнопку «ЗАВАНТАЖИТИ ЗОБРАЖЕННЯ».

Після натискання відкриється вікно, в якому нам запропонують вибрати зображення з ПК. Це можно зробити натиснувши на кнопку «ВИБЕРІТЬ ЗОБРАЖЕННЯ».

Після того як ми вибрали зображеннями, ми побачимо їх зменшені версії у вікні. Під кожним зображенням є поле, в якому можно додати опис.

Тепер ми можемо натиснути на кнопку завантажити, і зображення будуть завантажені до галереї. Приклад завантежння зображень та додання описів наведено в додатку Г на рис. Г11 – Г13. Після успішного завантаження зображення будуть показані в галереї.

Для видалення зображення, треба натиснути на кнопку з іконкою на кожному зображенні. Після натискання на кнопку, зображення буде видалено.

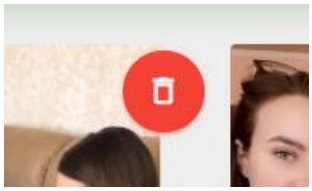

Рис. 17. Приклад кнопки видалення зображення

2.4.4 Управління користувачами

Вигляд сторінки з користувачами дуже схожий на вигляд сторінки з управлянням новинами. У нас є таблиця з списком користувачів, їхніми ролями, та діями. Для додавання нового користувача натискаємо на кнопку «+ НОВИЙ КОРИСТУВАЧ». У вікні ми можемо створити користувача, та дати йому роль. Приклад вікна з додавання користувача наведено у додатку Г на рис. Г14. Додавши користувача він буде в списку користувачів.

Якщо користувача потрібно заблокувати, ми можемо натиснувши на кнопоку «Заблокувати»

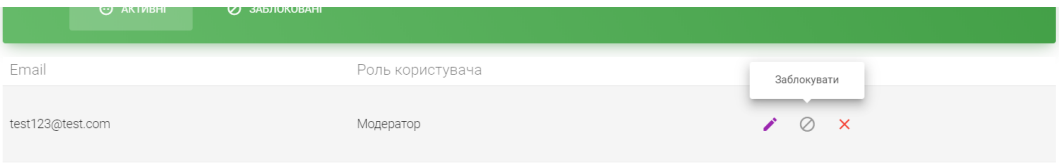

Рис. 18. Кнопка «заблокувати» користувача

Після цього з’явиться повідомлення про те, що користувач заблокований. Всіх заблокованих користувачів можна переглянути у вкладці «Заблоковані». Приклад вкладки наведений в додатку Г на рис. Г15.

Розблокувати користувача можно у вкладці «Заблоковані», натиснувши на цю саму кнопку. Для редагування користувача, ми можемо натиснути на кнопку редагування, вона така сама яка була і в новинах.

Для видалення користувача, також є така сама кнопка як і в новинах.

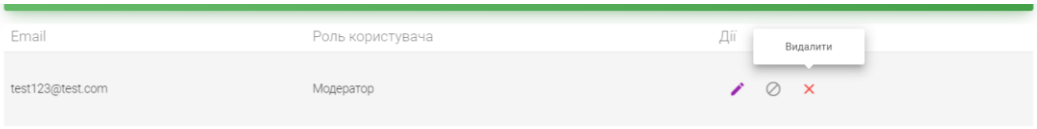

Рис. 19. Кнопка видалення користувача

2.4.5 Управління користувачами

Для розділів «Вакансії», «Сторінки» та «Події» процес такий же самий як і для новин.

2.4.5 Техніко-економічне обґрунтування

Вид системи*:* веб-сайт

Стадії:

- ескізний проект;
- ТЗ, технічний проект;
- робочий проект;
- впровадження.

Ступінь новизни розроблюваної задачі: В

Група складності алгоритму за їх характеристикою - 2

Вид інформації:	**Позначення**	**К-сть наборів даних**
Кількість видів змінної інформації	ЗІ	M=7
Кількість видів нормативно-довідкової інформації	НДІ	N=3
Кількість баз даних	БД	P=1
Обробка в режимі реального часу	РЧ	Так

Визначення витрат часу:

Вид системи	Стадія розробки системи	
Управління відвідуванням	Ескізний проект Т1	Технічне завдання Т2
	В	В
	Т1=70	Т2=43

Визначення витрат часу для стадії "технічний проект"(Т3)

Дані для визначення:

- Кількість форм вхідної інформації: В1 = 2
- Кількість форм вихідної інформації В2 = 5
- Кількість баз даних В3 = 1
- Базове значення витрат часу для стадії "технічний проект" ТБ3 = 72
- Базове значення витрат часу для стадії "робочий проект" ТБ4 = 221

- Базове значення витрат часу для стадії "впровадження" ТБ5 = 73

Розрахунок витрат часу для стадії "Технічний проект"(Т3)

$$Т3 := ТБ3 \cdot Кп \cdot К0$$

Кп - коефіцієнт трудомісткості робіт на стадії "технічний проект"

$k1 := 1 \quad k2 := 0.7 \quad k3 := 2.08$

$B1 := 2 \quad B2 := 5 \quad B3 := 1$

$$Кп := \frac{k1 \cdot B1 + k2 \cdot B2 + k3 \cdot B3}{B1 + B2 + B3} = 0.96$$

Вид використаної інформаії	Ступінь новизни
	В
k_1 (ЗІ)	1.0
k_2 (НДІ)	0.72
k_3 (БД)	2.08

К0 - коефіцієнт ступеню новизни проекту

$$Т3 := ТБ3 \cdot Кп \cdot К0 = 87.091$$

Розрахунок витрат часу для стадії "Робочий проект"

$$Т4 := ТБ4 \cdot Кп \cdot К0 \cdot Кс$$

$ТБ4 := 221 \quad Кс := 1.16$

$k1 := 1.1 \quad k2 := 0.58 \quad k3 := 0.48$

$К0 := 1.32$

$$Кп := \frac{k1 \cdot B1 + k2 \cdot B2 + k3 \cdot B3}{B1 + B2 + B3} = 0.697$$

$$Т4 := ТБ4 \cdot Кп \cdot К0 \cdot Кс = 236.031$$

Розрахунок витрат часу для стадії "Впровадження"

$ТБ5 := 73 \quad К0 := 1.21$

$$Т5 := ТБ5 \cdot Кп \cdot К0 \cdot Кс = 71.468$$

Визначення загальних витрат часу на розробку системи

$$T = T_1 + T_2 + T_3 + T_4 + T_5 = 507.59$$

Визначення чисельності виконавців

$$Ф = 75 : М = 3$$

$$Ч = Т / Ф = 507.59 / 75 = 6.768$$

Оплата праці виконавців

$$V1 := Ч \cdot М \cdot ЗП = 101517.931$$

Розрахунок річного фонду часу роботи ПК в годинах

$$Т_{пк} = 2000 - (6 * 8 + 5 * 12) = 1892$$

$$Т`_{пк} = 1892 * (450 / 2000) = 425.7$$

Поточні витрати на експлуатацію V"1

$$V''_1 = З_{ОП} + З_{АМ} + З_{ЕЛ} + З_Р + З_{МАТ}$$

Амортизаційні відрахування:

$$З_{АМ} = \frac{Ц_{ПК}}{H_А}$$

Балансова вартість ПК:

$$Ц_{пк} = 10000 * (1 + 0.12) = 11200 \text{ (грн)}$$

$$Н_А = 5$$

$$З_{ам} = 11200 / 5 = 2240 \text{ (грн)}$$

Витрати на електроенергію :

Потужність пк: $Р_{пк}$ = 0.25 кВт. Фонд корисного часу роботи ПК: $Т'_{пк}$ = 425.7 год. Вартість 1 кВт електроенергії для підприємств: $Ц_{ел}$ = 1.2 грн\кВт. Коефіцієнт інтенсивного використання ПК: А = 0.9

$$Зел = Р_{пк} * Т'_{пк} * Ц_{ел} * А = 0.25 * 425.7 * 1.2 * 0.9 = 114.939 \text{ (грн)}$$

Витрати на поточний ремонт і технічне обслуговування ПК:

$$Зр = Ц_{пк} * 0.06 = 11200 * 0.06 = 672 \text{ (грн)}$$

Непрямі витрати, пов'язані з експлуатацією ПК:

$$З_{мат}=Ц_{пк}*0.05=11200*0.05=560 \text{ грн}$$

Заробітна плата обслуговуючого персоналу $З_{оп}$ = 3500 грн

Поточні витрати на експлуатацію

$$V''1 = 3500 + 2240 + 114.939 + 672 + 560 = 7086.939 \text{ (грн)}$$

Загальні витрати на розробку програмного забезпечення:

$$V1 = V`1 + V''2 = 101517.931 + 7086.939 = 108604.87 \text{ (грн)}$$

Витрати на придбання і установку ПК

$$V2=Ц_{пк}=11200 \text{ грн}$$

Витрати на підготовку приміщення і навчання персоналу

Приміщення підходить, тому V3=0 грн та навчання персоналу V4=1000 грн

Загальна вартість розробки і впровадження системи

$$V = V1 + V2 + V3 + V4 = 108604.87 + 11200 + 0 + 1000 = 120804.87 \text{ (грн)}$$

Враховуючи норму амортизаційних втрат:

$$V_р = 120804.87 / 5 = 24160.974$$

Коефіцієнт економічної ефективності розробки:

Річний прибуток за рахунок збільшення кількості замовлень становить приблизно 202900

$$К_{еф} = П_р / V_р = 202900 / 24160.974 = 8.4$$

Термін окупності системи:

$$Т_{ок} = 1 / К_{еф} = 1 / 8.4 = 0.12 (~2 місяці)$$

Розділ 3. Охорона праці та правила безпеки

3.1. Правила безпечної роботи за комп'ютером

Працюючи за комп'ютером, рекомендовано дотримуватися правил тривалості роботи, постави, розміру шрифтів та зображень, вимог до приміщення тощо. Деякі принципи правильної роботи наведені нижче:

- у робочому приміщенні чи офісі, потрібно час від часу виконувати вологе прибирання;
- потрібно провітрювати щогодини приміщення;
- після кожного часу роботи рекомендується робити десяти хвилинну перерву, яку зручно суміщати з провітрюванням;
- необхідно постійно слідкувати за станом екрану монітора: він має бути чистим, без плям та пилу. Крім того, обов'язково слідкуйте за чистотою окулярів – комп'ютерних чи звичайних;
- постава: ноги стоять на підлозі чи на підставці; стегна розташовані під прямим кутом до тулуба, а гомілки – під прямим кутом до стегон; сидіти потрібно прямо або злегка нахилившись вперед; пальці рук знаходяться на рівні зап'ястків або трохи нижче – у такому положенні вони найбільш рухливі; плечі мають бути розслаблені та вільно опущені, що сприяє розслабленню рук; відстань від очей до екрану монітора – не менше 55-60 см; центр екрану має знаходитися на рівні очей чи трохи нижче; рекомендується хоча б раз на день виконувати гімнастику для очей;
- щоб попередити „синдром сухого ока", моргайте кожні 3-5 секунд;
- не використовувати телевізор у якості монітора;
- у процесі роботи за комп'ютером обов'язково звертайте увагу на дихання: воно має бути рівномірним, без затримок;
- при роботі з текстом рекомендується, щоб колір шрифту був темним, а колір фону – світлим (ідеальний варіант – чорний шрифт на білому фоні);

- якщо шрифт занадто мілкий, то потрібно збільшити масштаб документу (наприклад, до 120% чи більше);
- при наборі текстів з паперів чи книг рекомендується помістити джерело якомога ближче до монітору;

3.3. Правила захисту даних

Паролі від пошти, акаунтів, додатків, соцмереж, банкінгу в жодному разі не публікуйте на сайті – вже краще в блокнот записати, паперовий, і під подушку покласти. Однаковий пароль «на все» – дуже ризиковано, думаю, не потрібно пояснювати, чому.

Перегляньте уважно галерею – там не має бути фото документів (паспорта, ІПН, прав на авто).

3.2. Правила публікації

За загальним правилом фотографії, на яких зображено фізичну особу, можуть бути публічно показані, відтворені, розповсюджені лише за згодою цієї особи (ст.308 Цивільного кодексу). Інше питання, якщо такі фото були відкриті лише для певного кола користувачів. У такому випадку публічного розповсюдження таких фото не було, а отже, використання такого фото без згоди особи є неправомірним.

При цьому, для уникнення неправомірного втручання у приватне життя особи, варто утриматись від публікації відверто особистих фотографій, якщо тільки це безпосередньо не стосується питання публічного інтересу, яке не має ототожнюватись зі звичайною цікавістю чи жагою до сенсацій. Щодо використання інформації, яка розміщується особою у соціальній мережі (наприклад, пости у facebook), можна цитувати її у матеріалах, але із обов’язковим зазначенням автора.

Висновки

У даній роботі було розроблено панель управління веб-сайтом кафедри «ЕБОП» та бекенд частину сервера. Основна задача якої покращити та спростити процес управління контентом веб-сайту.

За допомогою розробленої панелі управління керування новинами, сторінками, галереєю стало простіше, якість контенту та його структура дозволила швидкий пошук по контенту веб-сайту.

Основними покращеннями нової панелі управління стали категорії для новин, сторінок та галереї. Збільшились можливості по керуванню користувачами, та надання прав доступу до певних сторінок панелі. Простий та лаконічний дизайн не перенавантажує інтерфейс непотрібними функціями. Процес додавання, редагування та видалення став більш інтуїтивним та швидшим.

Завдяки використанню сервісу Firebase, витрати на хостинг та сервер значною мірою зменшились, чим меньше дій виконано в панелі, ти менша ціна за місяць користування, що є великим плюсом. Окупованість даної системи не перевищує двух місяців, що є гарним показником.

Панель була розроблена на сучних технологіях та з можливістю простого розширення функціональності, що дозволяє адаптуватися під любі потреби. Простий інтерфейс доступу до даних дозволяє отримувати інформація не тільки на веб-сайті чи в панелі, але й на будь-яких пристроях які мають доступ до інтернету та можуть робити запити за допомогою HTTP протоколу.

Список використаних джерел

1. Кафедра екологічної безпеки та охорони праці | НУХТ – [Електронний ресурс]. – Режим доступу: https://nuft.edu.ua/fakultet-btek/kafedra-ebop/
2. Справочник по HTML | htmlbook.ru – [Електронний ресурс]. – Режим доступу: http://htmlbook.ru/html
3. Справочник по CSS | htmlbook.ru – [Електронний ресурс]. – Режим доступу: http://htmlbook.ru/css
4. Introduction — Vue.js – [Електронний ресурс]. – Режим доступу: https://vuejs.org/v2/guide/
5. Методы HTTP запроса - HTTP | MDN – [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/
6. Cloud Functions for Firebase – [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/functions
7. Firebase Realtime Database – [Електронний ресурс]. – Режим доступ: https://firebase.google.com/docs/database
8. Firebase Authentication – [Електронний ресурс]. – Режим доступ: https://firebase.google.com/docs/auth
9. Современный учебник JavaScript– [Електронний ресурс]. – Режим доступу: https://learn.javascript.ru/

Додатки

Додаток А – функціональні моделі

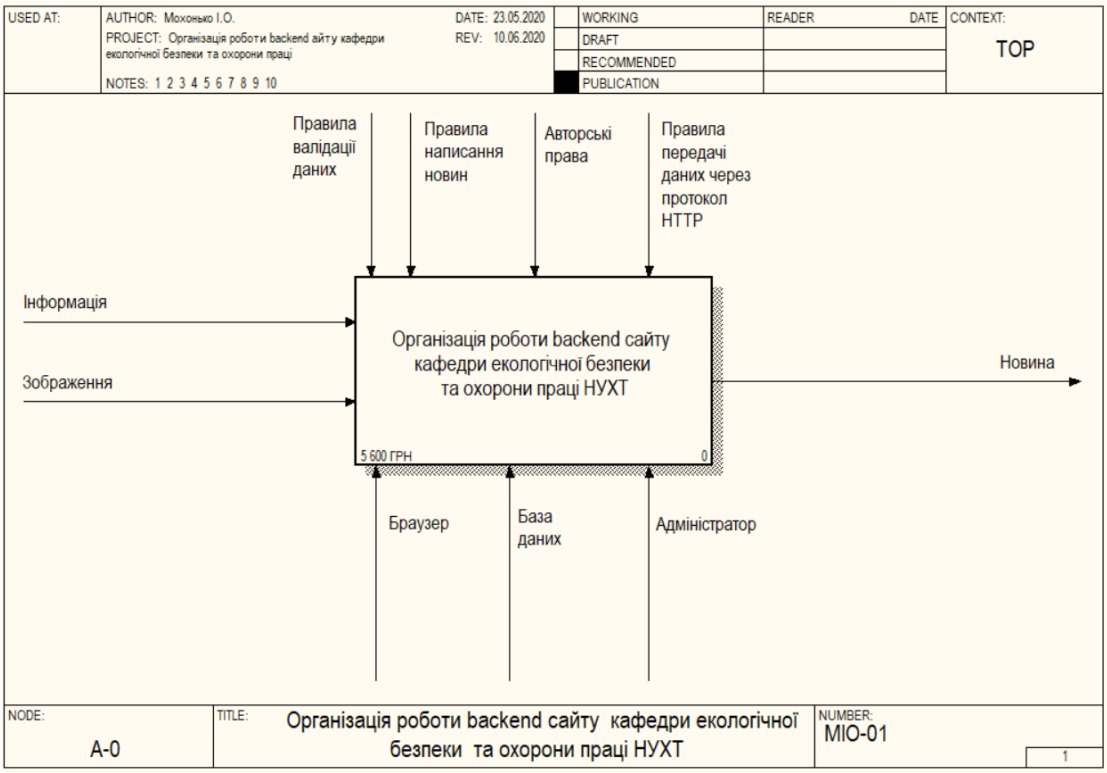

Рис. А1. Модель AS-IS, 1 рівень декомпозиції

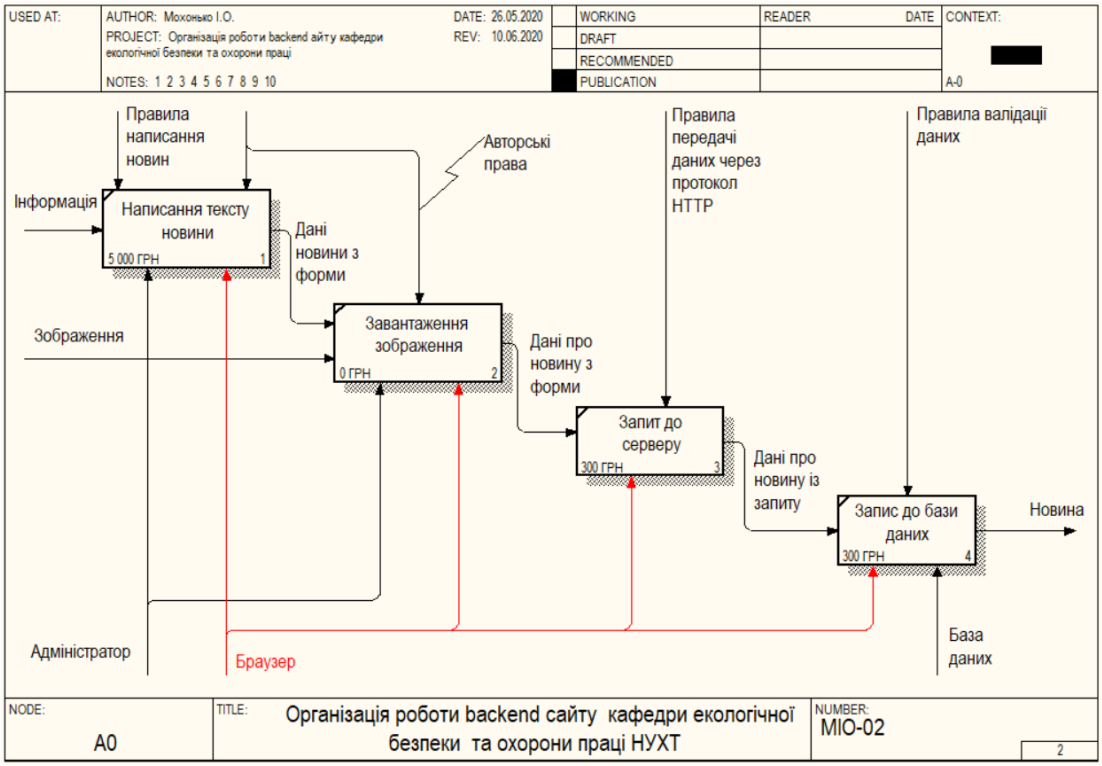

Рис. А2. Модель AS-IS, 2 рівень декомпозиції

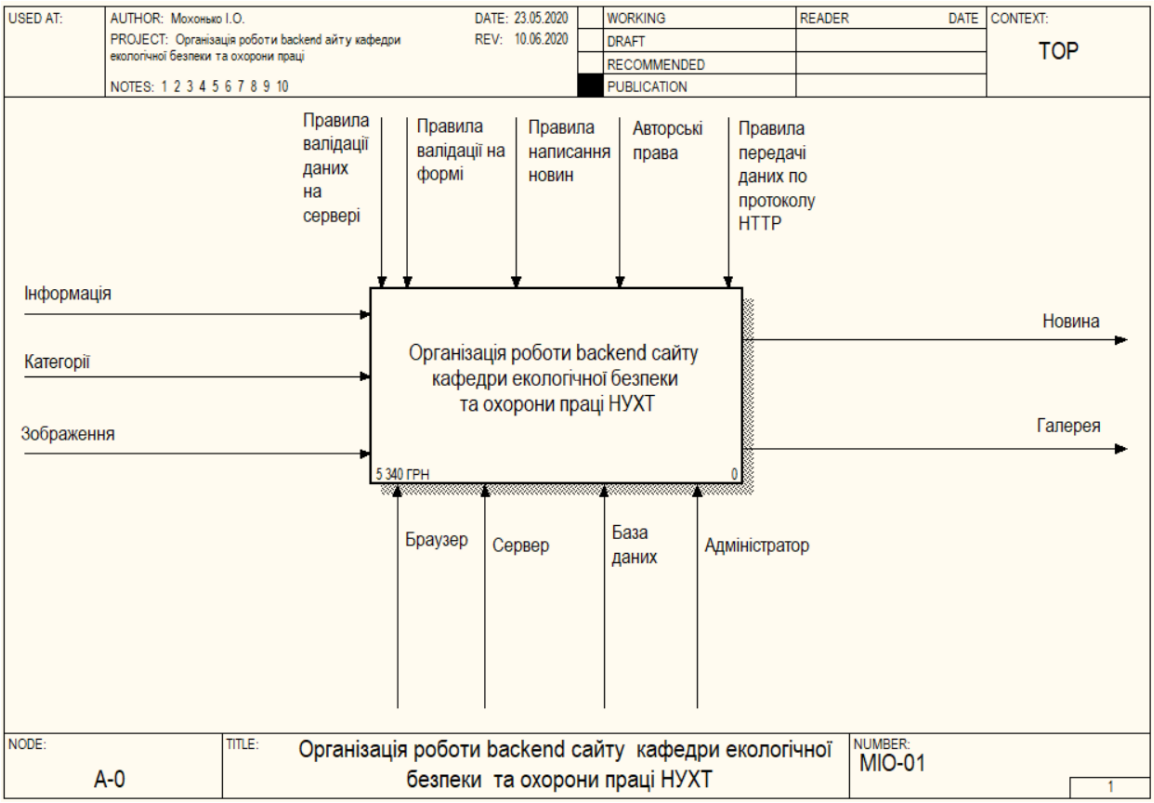

Рис. А3. Модель TO-BE, 1 рівень декомпозиції

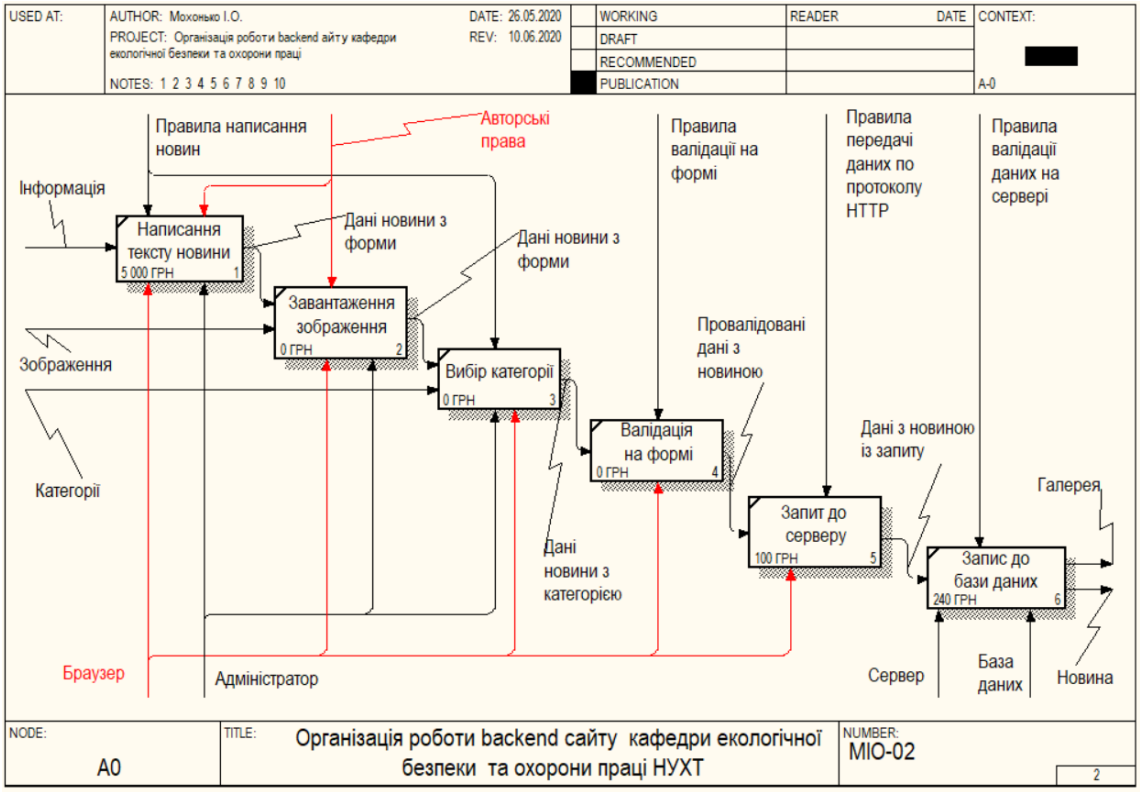

Рис. А4. Модель TO-BE, 2 рівень декомпозиції

Додаток Б – моделі БД

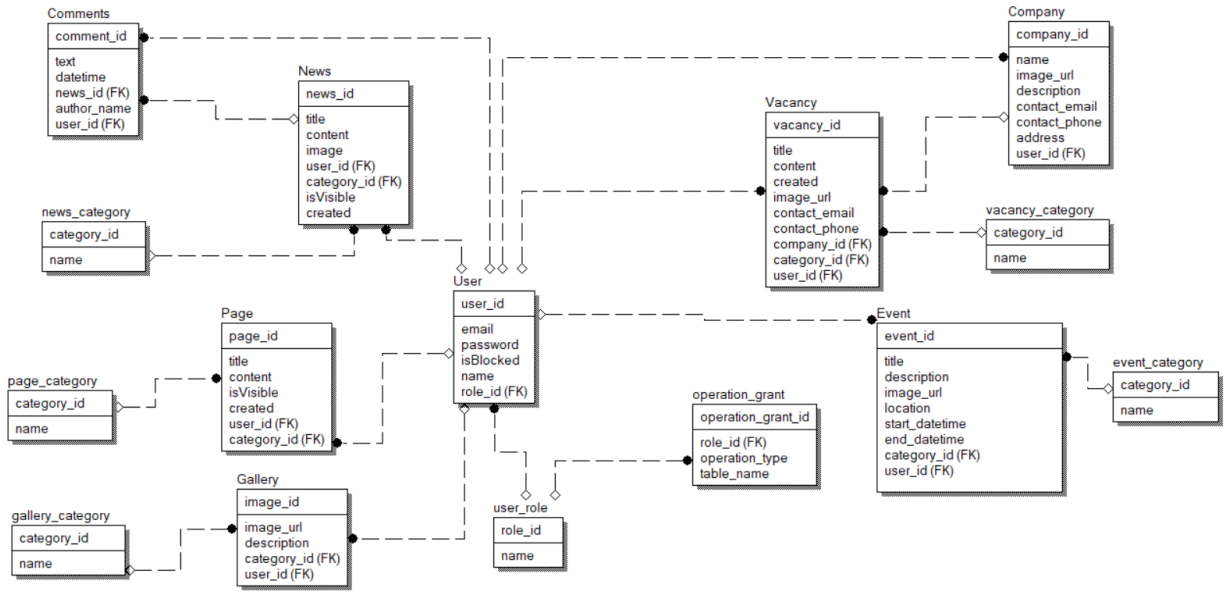

Рис. Б1. Приклад логічної моделі бази даних

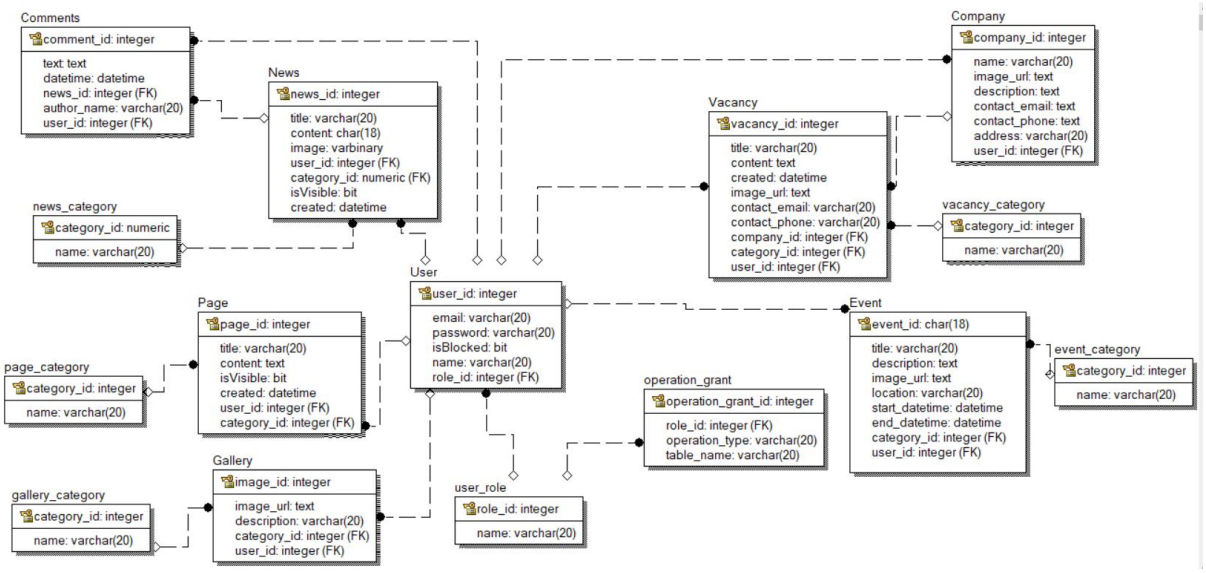

Рис. Б2. Приклад фізичної моделі бази даних

Рис. Б3. Приклад формування стуктури JavaScript класа для роботи з базою даних

Додаток В – фрагменти коду програми

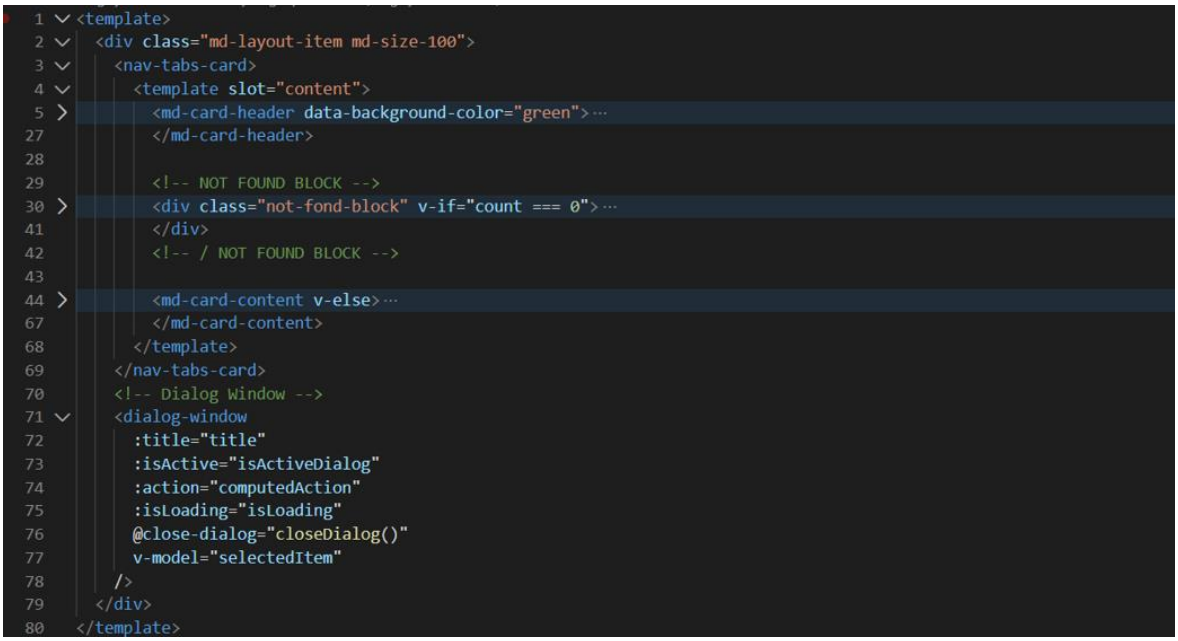

Рис. В1. Приклад розмітки сторінки управління новинами

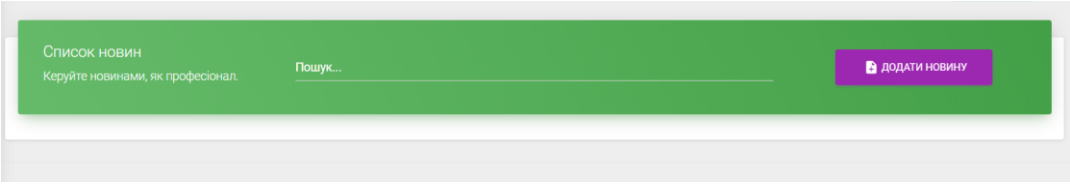

Рис. В2. Вигляд сторінки після написання розмітки

```
</div>
<div class="md-layout-item md-size-25 md-xsmall-size-100 allign">
  <md-button class="md-primary" @click="openDialogForNewRecord()">
    <md-icon> note_add</md-icon>
    Додати новину
  </md-button>
</div>
```

Рис. В3. Приклад розмітки кнопки

```
openDialogForNewRecord() {
  this.selectedAction = "create";
  this.selectedItem = {
    visible: true
  };
  this.openDialog();
},
```

Рис. В4. Приклад коду функції

```
<template>
  <dialog-window :showDialog="isActive" :title="title">
    <template slot="content">
      <md-field>
        <label>Заголовок</label>
        <md-input v-model="value.title"></md-input>
      </md-field>

      <div class="card-content">
        <!-- Upload File  -->
        <div class="row row-1">
          <div class="photo-contaier">
            <custom-files-upload-input :isMultiple="false" v-model="files" />
            <photo-preview v-model="files" :isDescription="false" />
          </div>
          <md-switch v-model="value.visible" class="md-primary">
            Відображати запис
          </md-switch>
        </div>

        <!-- Editor -->
        <div class="row row-2">
          <drop-down
            labelSearchPlaceholder="Пошук"
            labelTitle="Виберіть категорію для новини"
            labelNotFound="Не знайдено"
            textProp="title"
            :table="table"
            v-model="value.category"
          />

          <hr />

          <md-field>
            <label>Короткий опис новини</label>
            <md-textarea v-model="value.description"></md-textarea>
```

Рис. В5. Приклад розмітки вікна та форми (частина 1)

```
            <md-textarea v-model="value.description"></md-textarea>      Ivan Mokhonko, 8 days ago • feat(categories, description)
          </md-field>

          <hr />

          <vue-editor v-model="value.content" />
        </div>
      </div>
    </template>
    <template slot="actions">
      <md-button @click="$emit('close-dialog')">
        Закрити
      </md-button>
      <md-button class="md-success" @click="action">
        <loading v-if="isLoading" />
        <span v-else>Зберегти</span>
      </md-button>
    </template>
  </dialog-window>
</template>
```

Рис. В6. Приклад розмітки вікна та форми (частина 2)

Рис. В7. Приклад відкритого вікна з формою додавання новини

```
async createRecord() {      sergey mokhonko, a month ago • Feat: News. Create
  const {
    title,
    content,
    description,
    visible,
    category
  } = this.selectedItem;
  const catId =
    category !== undefined && category.hasOwnProperty("id")
      ? category.id
      : "";
  this.isLoading = true;
  let query;

  //Create record
  try {
    query = await createNews(
      {
        title,
        content,
        visible,
        category: catId,
        description
      },
      this.token
    );

    if (!query.success) {
      this.notifyVue(NEWSMESSAGES.REJECT.NOT_CREATED, "warning", "danger");
      this.isLoading = true;
      return;
    }
  } catch (err) {
    console.error(err);
    this.notifyVue(NEWSMESSAGES.REJECT.ERROR, "warning", "danger");
    this.isLoading = false;
```

Рис. В8. Приклад методу який буде запущений після натискання на кнопку «ЗБЕРЕГТИ»

```
createNews = async (data, token) => {
  const headers = createAuthHeader(token);
  const options = {
    headers,
    body: JSON.stringify(data)
  };      sergey mokhonko, a month ago • News.

  return await this._post("news", options);
};
```

Рис. В9. Приклад функція яка відправляє запит на сервер за даними які ми вказали на формі

Рис. В10. Приклад функції яка відправляє підготований запит з даними

Рис. В11. Приклад функції на стороні сервера

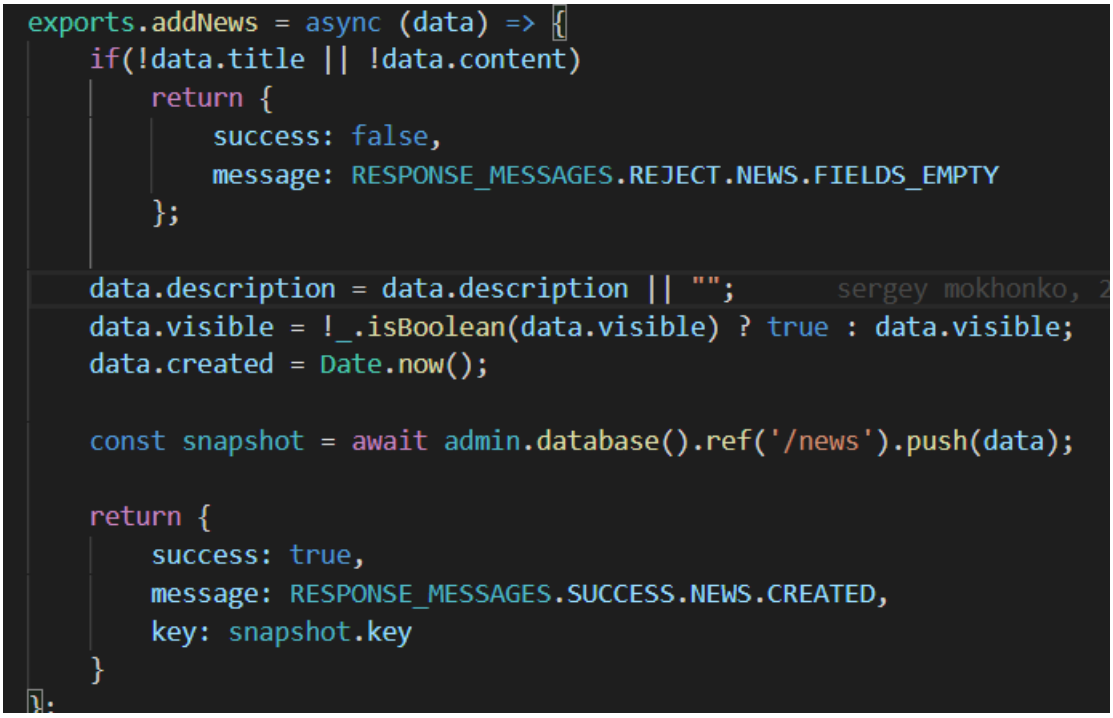

Рис. В12. Приклад функції яка записує дані до бази даних

Рис. В13. Приклад запису в базі даних після виконання функції

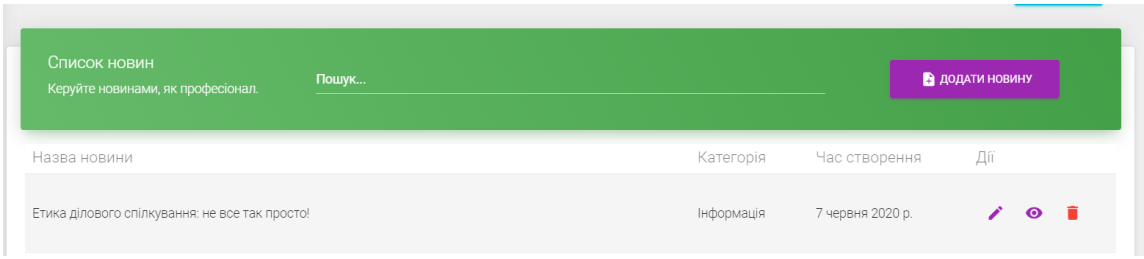

Рис. В14. Приклад доданої новини на сторінці

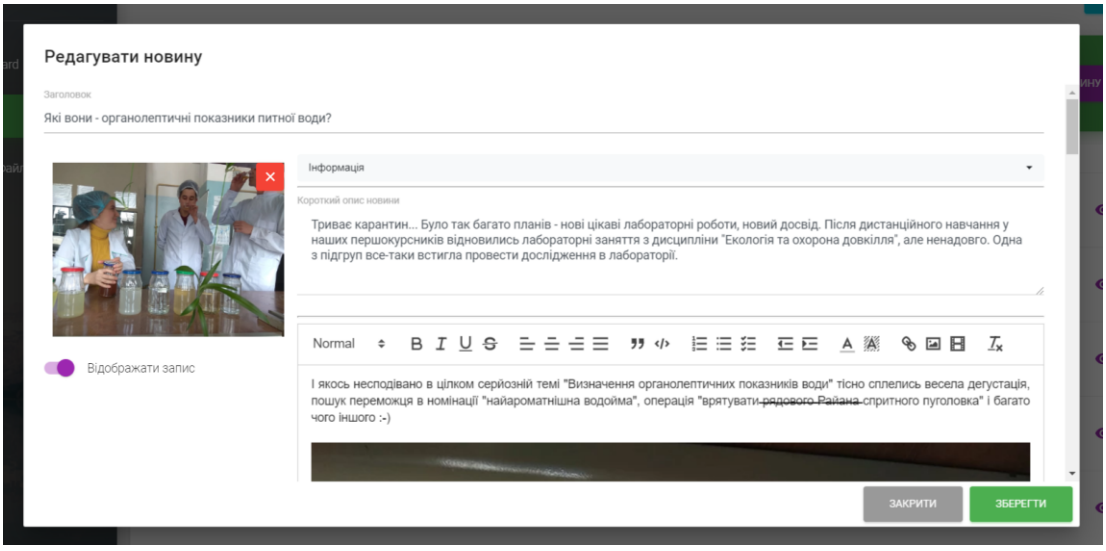

Рис. В15. Приклад відкрившогося вікня з новиною

```
async editRecord() {
  const { id, category } = this.selectedItem;
  //get ID of new category
  const catId =
    category !== undefined && category.hasOwnProperty("id")
      ? category.id
      : category;
  const newData = {
    ...this.selectedItem,
    category: catId
  };

  if (!isItemHasPhoto(newData) && isNewsHasPhoto) {
    await deleteImage(item);
  } else {
    isEqualUrl ? await deleteImage(item) : false;
  }

  //update post
  try {
    const query = await editNews(id, newData, this.token);
    //update image
    await updateImage(this.selectedItem);

    if (query.success) {
      const updatedNewsData = await getNews(index, 1);
      this.news.splice(index, 1, updatedNewsData.data[0]);
      this.notifyVue(NEWSMESSAGES.SUCCESS.UPDATED, "done", "success");
    } else {
      this.notifyVue(NEWSMESSAGES.REJECT.NOT_UPDATED, "warning", "danger");
    }
  } catch (err) {
    this.notifyVue(NEWSMESSAGES.REJECT.ERROR, "warning", "danger");
    console.error(err);
  }

  this.isLoading = false;
```

Рис. В16. Виконуємо функцію натискання на кнопку

```
editNews = async (id, newData, token) => {
  const headers = createAuthHeader(token);
  const data = JSON.stringify(newData);
  const options = {
    headers,
    body: data
  };

  return await this._put("news", { id, options });
};
```

Рис. В17. Створюємо запит та відпраляємо

```
news.put('/:key', async (req, res) => {
  const key = req.params.key;       Ivan Mokhonko, 2 months ago • feature(news): skeleton
  const { title, description, content, visible } = req.body;
  const isLoggedIn = await authenticate(req);
  const result = isLoggedIn.authenticated
    ? await editNews(key, { title, description, content, userIdUpdate: isLoggedIn.userID, visible })
    : isLoggedIn;
  res.set('Access-Control-Allow-Origin', '*');
  res.json(result);
});
```

Рис. В18. Запускаємо функцію на стороні сервера

```
exports.editNews = async (key, newData) => {
    const value = await checkValue(key, 'news');
    if(!value)
        return {
            success: false,
            message: RESPONSE_MESSAGES.REJECT.NEWS.ITEM_NOT_FOUND
        };

    if(!newData.title || !newData.content)
        return {
            success: false,
            message: RESPONSE_MESSAGES.REJECT.NEWS.FIELDS_EMPTY
        };

    newData.description = newData.description || "";
    newData.visible = !_.isBoolean(newData.visible) ? true : newData.visible;
    newData.updated = Date.now();

    await admin.database().ref(`/news/${key}`).update(newData);
        return {
            success: true,
            message: RESPONSE_MESSAGES.SUCCESS.NEWS.EDITED,
        };
};
```

Рис. В19. Ще раз валідуємо дані та оновлюємо базу даних

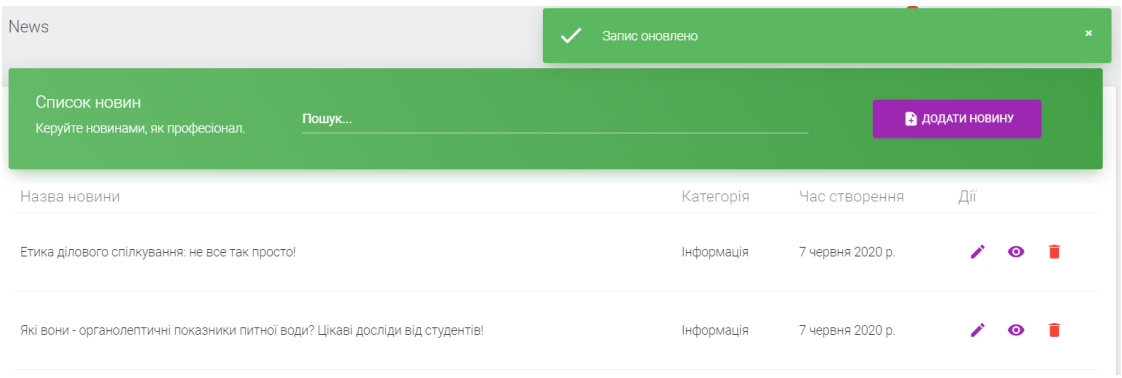

Рис. В20. Приклад оновленої новини

```
async deleteItem(id) {
  if (!window.confirm("Ви дійсно бажаєте видалити запис?")) return;

  const index = this.news.findIndex(el => el.id === id);
  const item = this.news[index];
  //find category
  const { category } = item;
  const catId =
    category !== undefined && category.hasOwnProperty("id")
      ? category.id
      : category;

  // delete image
  await deleteImage(item);
  // delete news from DB
  const result = await deleteNews(id, this.token);
  if (result.success) {
    this.news.splice(index, 1);
    this.count = this.count - 1;
    this.notifyVue(NEWSMESSAGES.SUCCESS.DELETED, "done", "success");
  } else {
    this.notifyVue(NEWSMESSAGES.REJECT.NOT_DELETED, "warning", "danger");
    return;
  }

  if (this.isMore) {
    const newsArr = await getNews(this.news.length, 1);
    this.createNewsData(newsArr);
  }
},
```

Рис. В21. Приклад функції яка запуститься після нажаття на кнопку видалення

```
deleteNews = async (id, token) => {
  const headers = createAuthHeader(token);
  const options = {
    headers
  };
  const query = await this._del("news", { options, id });
  if (query.success) return query;

  return {
    success: false,
    message: "Can't delete this item"
  };
};
```

Рис. В22. Формуємо запит на видалення

```
exports.deleteNews = async (key) => {
    const value = await checkValue(key, 'news');
    if(!value)
        return {
            success: false,
            message: RESPONSE_MESSAGES.REJECT.NEWS.ITEM_NOT_FOUND
        };

    await admin.database().ref(`/news/${key}`).remove();
    return {
        success: true,
        message: RESPONSE_MESSAGES.SUCCESS.NEWS.DELETED,
    };
};
```

Рис. В23. Видаляємо з бази даних запис

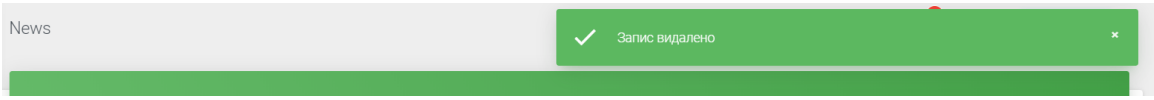

Рис. В24. Приклад повідомлення про видалений запис

```
<div class="md-layout-item md-size-50 md-xsmall-size-100">
  <md-autocomplete
    class="search__input"
    v-model="searchValue"
    :md-options="employees"
  >
    <label>Пошук...</label>
  </md-autocomplete>
</div>
```

Рис. В25. Приклад розмітки для поля пошуку

Рис. В26. Приклад поля на сторінці

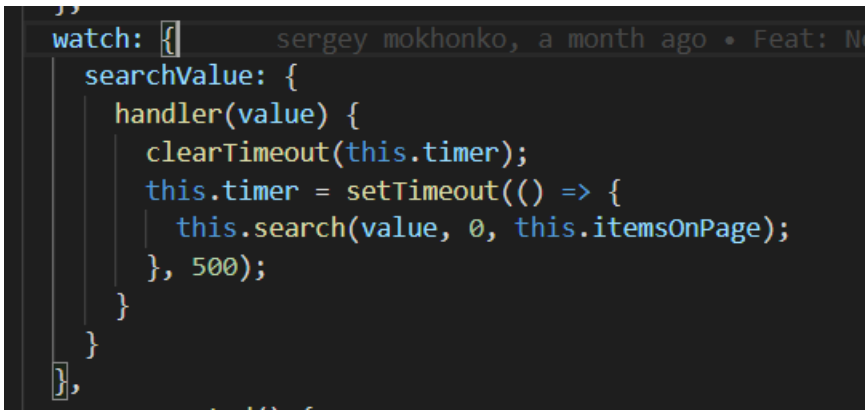

Рис. В27. Приклад коду який буде запускатиться коли користувач почне вводити щось в поле пошуку

```
async search(value, startAt, count) {
  this.news = [];
  this.isMainLoading = !this.isMainLoading;
  const query = await search(value, startAt, count);
  if (query.success) {
    this.isMainLoading = !this.isMainLoading;
    this.createNewsData(query);
    return;
  }

  this.isMainLoading = !this.isMainLoading;
  this.createNewsData({
    newsCount: 0,
    data: []
  });
```

Рис. В28. Приклад функція як виконує пошук по новинам

```
search = async (q, startAt, itemsOnPage) => {
  const params = `?startAt=${startAt}&itemsOnPage=${itemsOnPage}&q=${q}`;
  const options = {
    method: "GET"
  };
  return await this._get("news", { params, options });
};
```

Рис. В29. Приклад функції яка відправляє запит на сервер, з параметрами фільтрації

```
news.get('/', async (req, res) => {
  let { startAt, itemsOnPage, q } = req.query;
  let result;
  if(!_.isEmpty(q)){
    result = await searchNews(q, startAt, itemsOnPage);
  } else {
    result = await getNews(startAt, itemsOnPage);
  }

  const status = result.success ? 200 : 400;
  res.set('Access-Control-Allow-Origin', '*');
  res.status(status).json(result);
});
```

Рис. В30. Функція пошуку новин за ключовим словом

```
exports.getNews = async (startAt, count) => {
    if(_.isEmpty(startAt) || _.isEmpty(count))
        return{
            success: false,
            message: RESPONSE_MESSAGES.REJECT.NEWS.PARAMS_NOT_SET
        };

    try {
        const dbRecords =  await admin.database().ref('/news').once('value');
        const keys = Object.keys(dbRecords.val()).reverse();
        const key = keys[parseInt(startAt)]

        if(parseInt(startAt) >= keys.length)
            return {
                success: false,
                message: RESPONSE_MESSAGES.REJECT.NEWS.OUT_OF_RANGE
            };

        const query = admin.database().ref('/news').orderByKey().limitToLast(parseInt(cou
        const snapshot = await query.once('value');
        const data = snapshot.val();
        if(!data)
            return {···
            };

        const transformedData = transformData(data);

        return {
            success: true,
            data: transformedData.reverse(),
            newsCount: keys.length
        };

    } catch (err) {
        return {···
        }
```

Рис. В31. Приклад функції яка фільтрує новини, якщо вказане ключево слово, або повертає всі новини

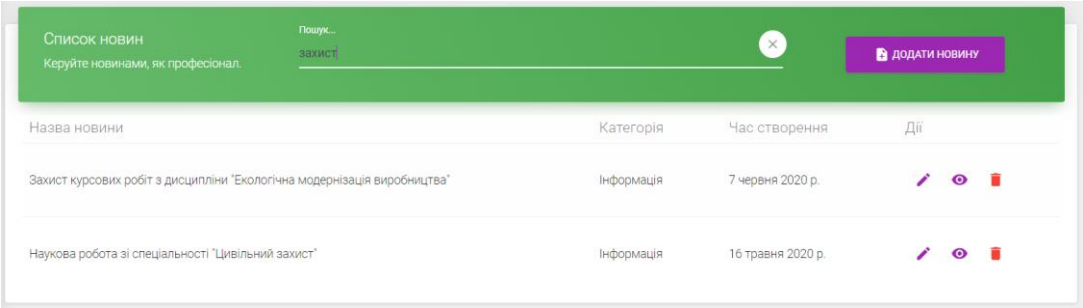

Рис. В32. Приклад знайдених новин за ключовим словом «захист»

```
const NEWSMESSAGES = {
  SUCCESS: {
    SAVED: "Новину збережено",
    UPDATED: "Запис оновлено",
    DRAFT: "Запис переміщено до чернеток",
    PUBLISHED: "Запис опубліковано",
    DELETED: "Запис видалено",
  },
  REJECT: {
    NOT_CREATED: "Неможливо зберегти запис",
    NOT_UPDATED: "Запис не оновлено",
    NOT_DRAFT: "Сталася помилка. Запис неможливо перемістити в чернетки",
    NOT_DELETED: "Неможливо видалити або запис не існує",
    ERROR: "Щось пішло не так. Спробуйте пізніше."
  }
};
```

Рис. В33. Повідомлення про помилки під час виконання операцій

Додаток Г – скріншоти веб-сайту

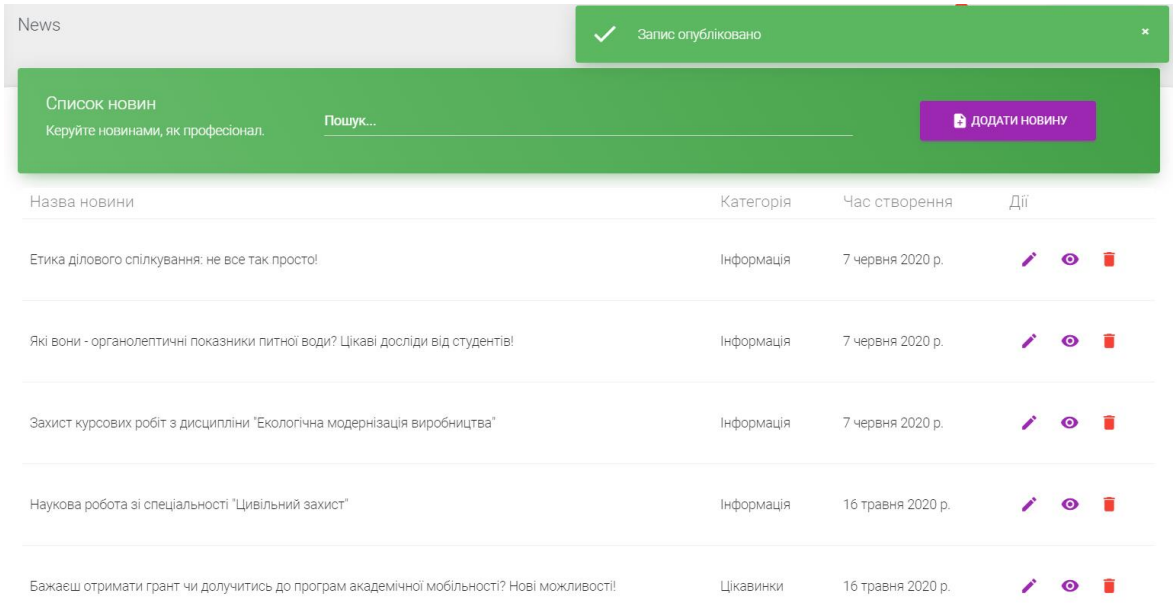

Рис. Г1. Приклад зняття позначки «чорновик» з новини

Рис. Г2. Приклад вікна з формою додавання новини

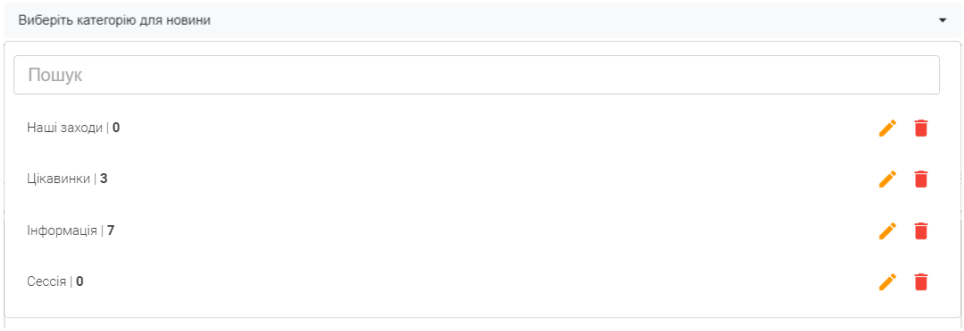

Рис. Г3. Приклад доданої категорії

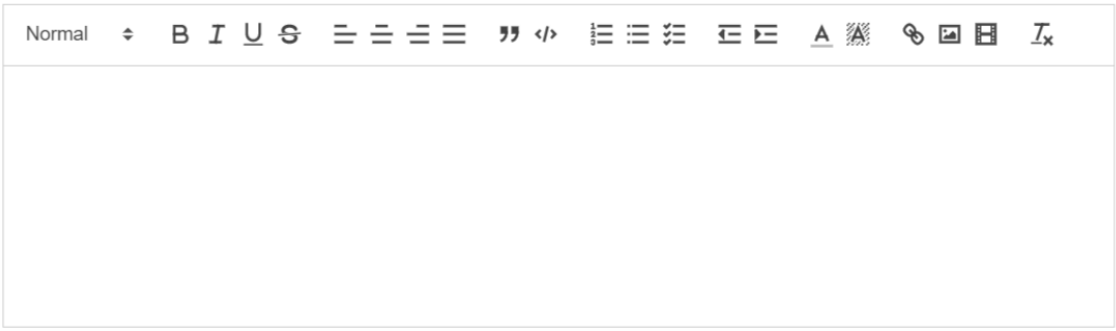

Рис. Г4. Приклад візульного редактора новини

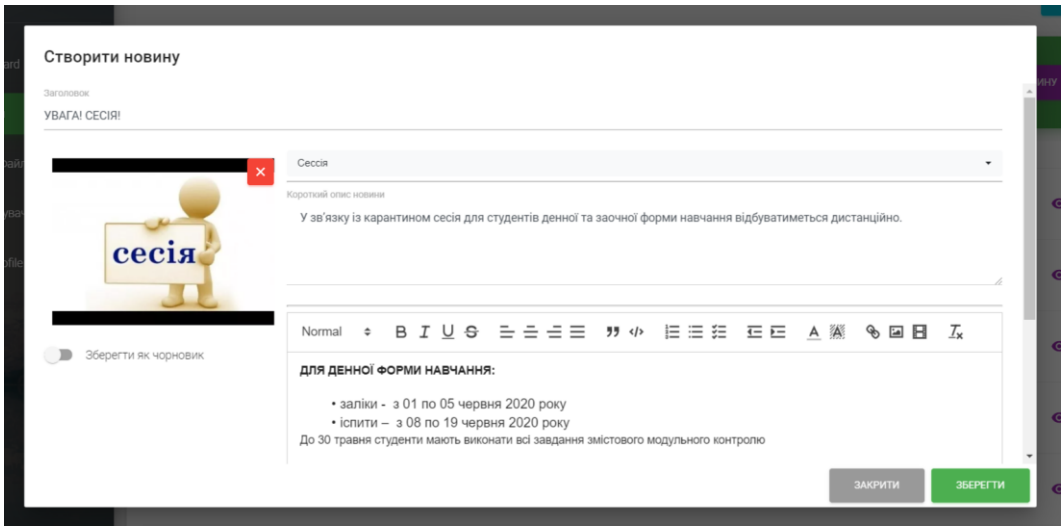

Рис. Г5. Приклад заповненої форми з новиною

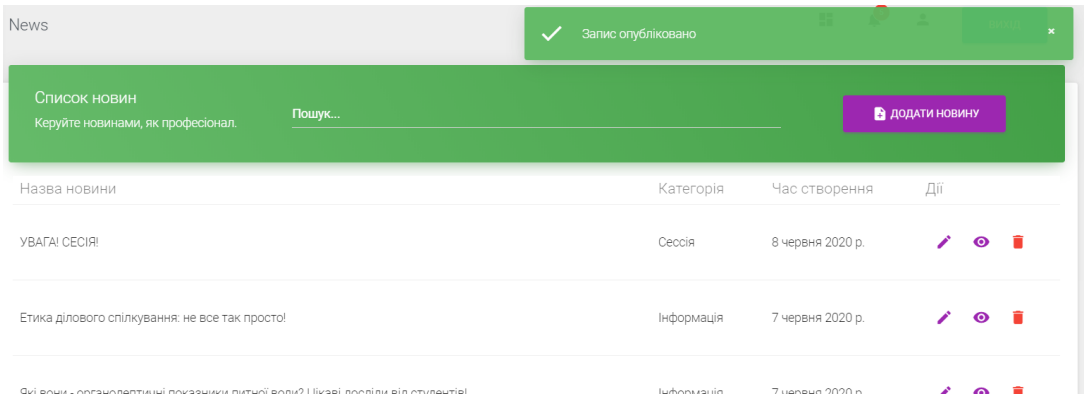

Рис. Г6. Приклад доданої новини

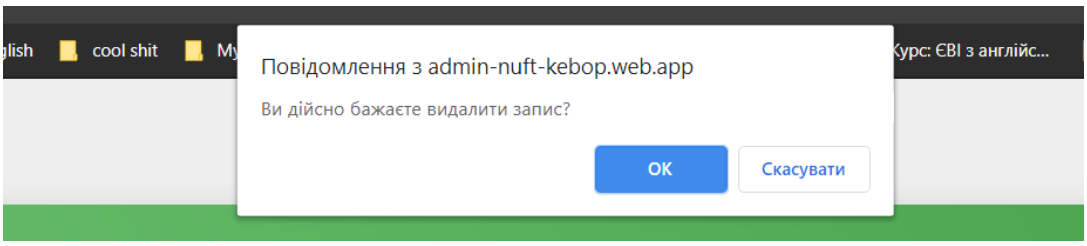

Рис. Г7. Приклад вікна з підвердженнями видалення новини

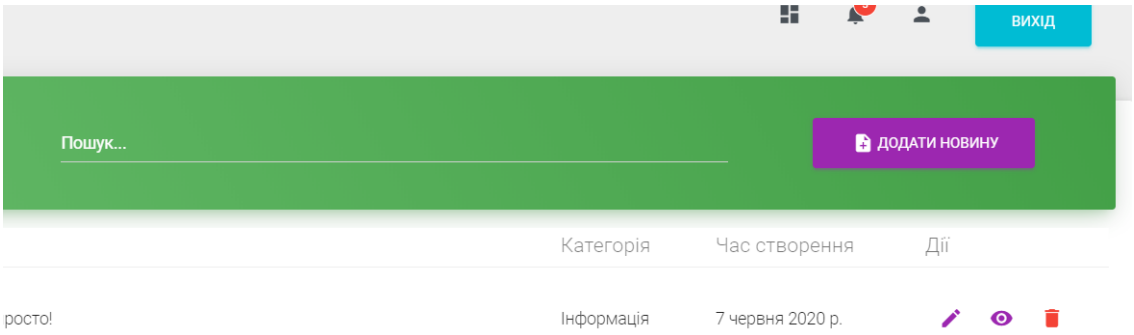

Рис. Г8. Приклад поля для пошуку новини

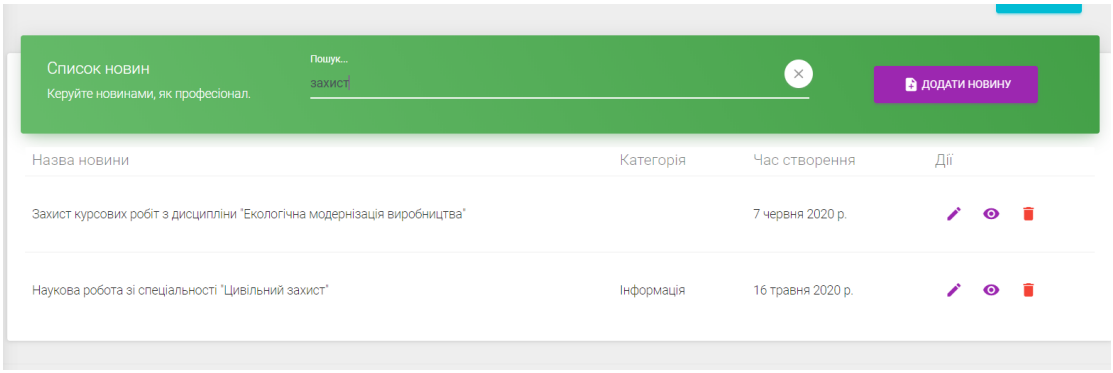

Рис. Г9. Приклад введеня слова для пошуку

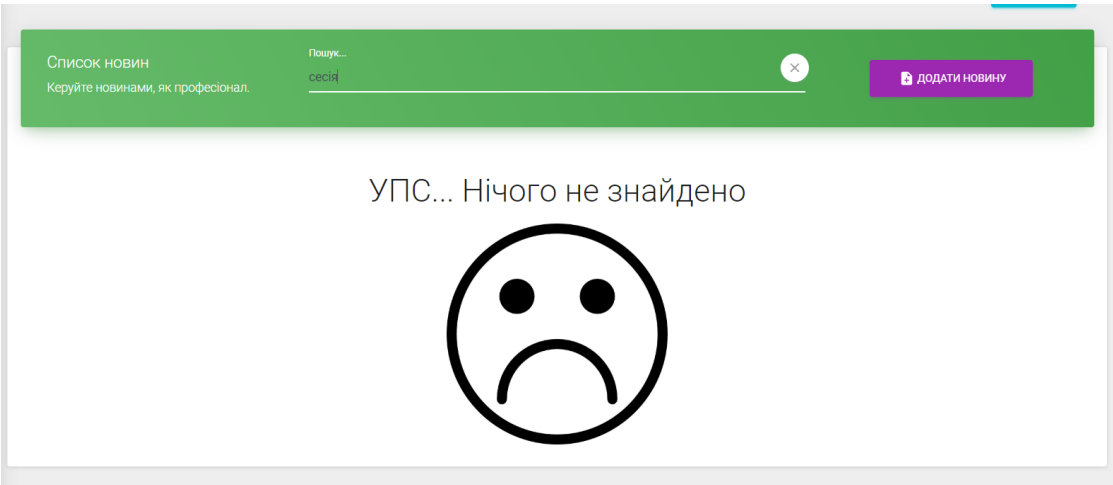

Рис. Г10. Приклад відсутності результатів пошуку

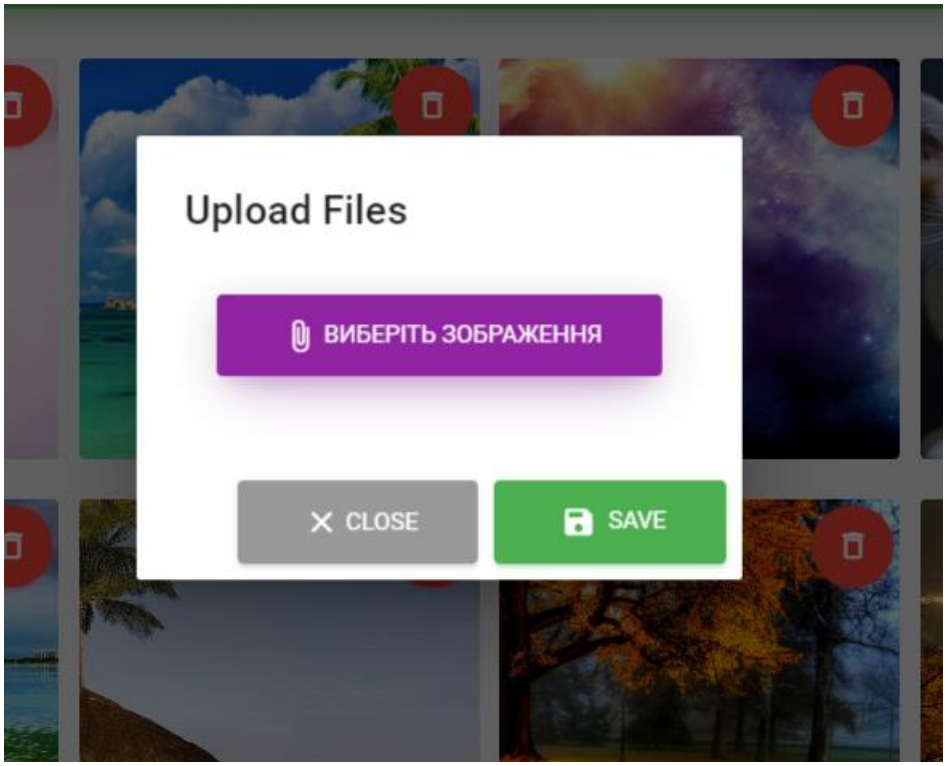

Рис. Г11. Приклад вікна з завантаженням

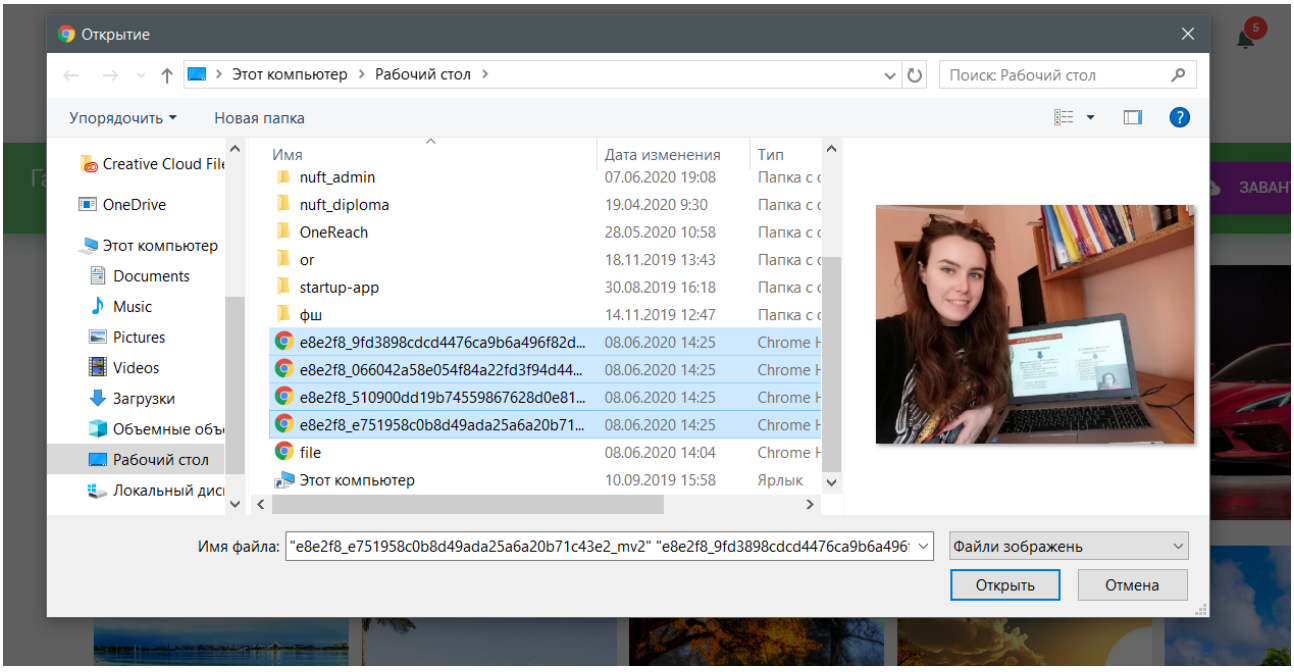

Рис. Г12. Вибір зображень для завантаження

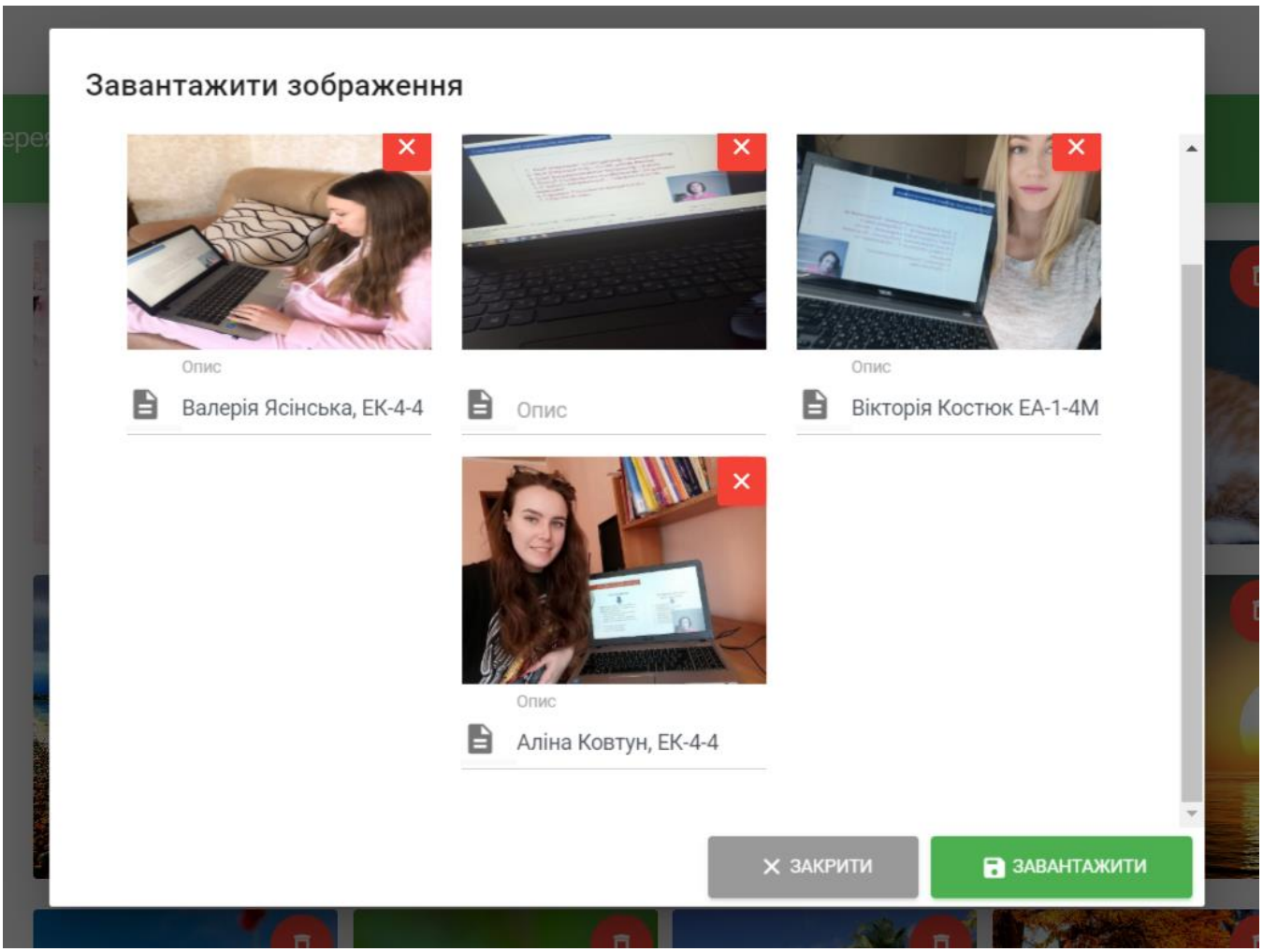

Рис. Г13. Приклад доданого опису

Рис. Г14. Приклад вікна з додавання користувача

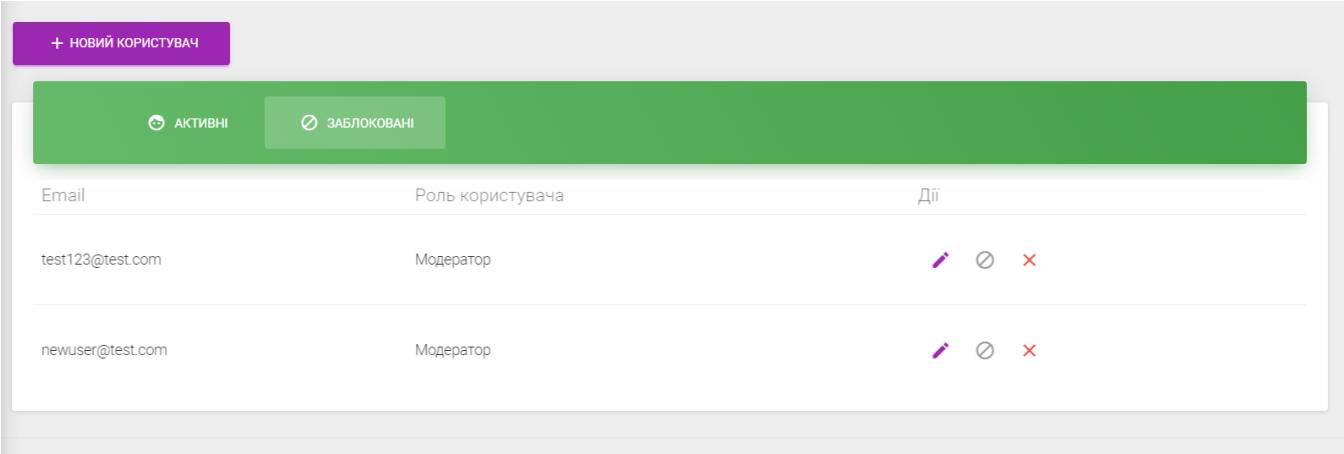

Рис. Г15. Вкладка з заблокованими користувачами