НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

на тему: «Модуль веб-системи «Конференції/Семінари»»

Виконала:
студентка IV курсу, групи ТВ-71
Конюк Анастасія Володимирівна _____

Керівник:
доцент, канд. техн. наук
Діденко Олексій Олександрович _____

Консультант :
_____

Рецензент:
_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2021р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Конюк Анастасії Володимирівні
_____
(прізвище, ім’я, по батькові)

1. Тема роботи «Модуль веб-системи «Конференції/Семінари» керівник роботи Діденко Олексій Олександрович , к.т.н., доцент
затверджена наказом вищого навчального закладу від ”24” травня 2021р. № **1267с**

2. Строк подання студентом роботи 08.06.2021
3. Вихідні дані до роботи Мова програмування HTML,Java, редактор вихідного коду Visual Studio Code, API навчального порталу
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Провести дослідження сучасних навчально-інформаційних веб-сайтів зі схожою функціональністю сторінки. Провести порівняльний аналіз існуючих програмних рішень. Визначити головні функціональні переваги програмного рішення, що були знайдені в аналогах. Спроектувати та розробити унікальний дизайн користувацького інтерфейсу. Створити діаграму класів та діаграму прецедентів. Обрати технології для реалізації програмного продукту. Розробити необхідні модулі програмної системи та об’єднати їх. Розробити функціональні можливості програмної системи. Імплементувати розроблений дизайн. Об’єднати усі складові застосунку. Протестувати готовий програмний продукт, виправити помилки та покращити функціональні можливості

застосунку.

5. Перелік ілюстративного матеріалу: Титульна сторінка, Актуальність Структура бази даних,Алгоритм редагування інтерфейсу,Діаграма прецедентів ,Структурна схема системи ,Діаграма класів, Засоби розробки, Схема аналізу відношень, Файлова структура проекту, ,Інші сторінки застосунку, Навігація, Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”___”_____202___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	***Затвердження теми роботи***		
2.	Вивчення та аналіз задачі		
3.	Розробка архітектури та загальної структури системи		
4.	Розробка структур окремих підсистем		
5.	Програмна реалізація системи		
6.	Оформлення пояснювальної записки		
7.	***Захист програмного продукту***	12.05.2021	
8.	***Передзахист***	25.05.2021	
9.	Захист	14.06.2021	

Студент _____ Конюк А.В
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Діденко О.О.
(підпис) (прізвище та ініціали,)

ВІДГУК

керівника дипломної роботи

освітньо-кваліфікаційного рівня „бакалавр”

виконаної на тему: «Модуль веб-системи «Конференції/Семінари»

студенткою Конюк Анастасією Володимирівною

(прізвище, ім’я, по батькові)

Для покращення кваліфікаційного рівня спеціалістів потрібно удосконалювати їх знання в предметних областях, а для цього потрібні якісні та досконалі системи для дистанційної освіти. Розробка навчальних порталів потребує аналізу методів наочного подання знань. Бакалаврська дипломна робота Конюк А.В. присвячена розробці програмної навчальної сторінки функціональними можливостями якої є перегляд конференцій та ефективне постачання інформації.

Студентка Конюк А.В. під час роботи над дипломною роботою продемонструвала високий рівень професійних навичок та знань з проектування мобільних систем, розробки користувацьких інтерфейсів, а також показала вміння працювати з науково-технічною літературою та аналізувати її. Студентка проявила самостійність та відповідальність під час розробки програмного забезпечення.

Розроблена веб-система повністю вирішує поставлену задачу. Користувацький інтерфейс створено з використанням адаптивного, зручного та зрозумілого дизайну.

Робота відповідає всім вимогам, які ставляться до бакалаврської дипломної роботи. Студентка Конюк Анастасія Володимирівна заслуговує присвоєння ступеня бакалавра та кваліфікації професіонала в галузі програмування з напряму підготовки 6.050103 “Програмна інженерія”.

Керівник дипломної роботи

к.т.н., доцент _____ Діденко О.О

(посада, вчені звання, ступінь) (підпис) (ініціали, прізвище)

РЕЦЕНЗІЯ

на здобуття ступеня бакалавра,
виконаний на тему «Модуль веб-системи «Конференції/Семінари»
студенткою Конюк Анастасією Володимирівною

Бакалаврська дипломна робота студентки Конюк А.В. присвячена розробці навчальної системи, основними функціями якої була б швидка передача актуальної інформації на тему конференцій та семінарів.

Перший розділ містить постановку задачі автоматизованої побудови лінків на конференції та розклад в системі. У другому розділі проаналізовано проблему створення розкладу та актуальних сповіщень про конференції . У третьому розділі описано методи розробки навчальної системи. У четвертому розділі описано алгоритмічні основи системи. П'ятий розділ описує архітектуру системи, засоби програмної реалізації, діаграми, які відображають логіку роботи програмного додатку та продемонстровано макети користувацького інтерфейсу. Шостий розділ містить опис сценаріїв взаємодії користувача з системою.

До недоліків програмного продукту слід віднести те, що система інтегрована тільки для локального використання .

Робота відповідає всім вимогам, які ставляться до бакалаврської дипломної роботи. Студентка Конюк Анастасія Володимирівна заслуговує оцінки «відмінно» та присвоєння ступеня бакалавра і кваліфікації професіоналу в галузі програмування з напряму підготовки 6.050103 «Програмна інженерія».

Рецензент

_____	_____	_____
(посада, вчені звання, ступінь)	(підпис)	(ініціали, прізвище)

АНОТАЦІЯ

Дипломну роботу виконано на 60 аркушах, вона містить 3 додатки та перелік посилань на використані джерела з 15 найменувань. У роботі наведено 37 рисунків.

Метою даної дипломної роботи є розробка веб-сторінки в системі з використанням розкладу і актуальних сповіщень про конференції та семінари.

У роботі проведено аналіз існуючих рішень поставленої задачі — варіантів сповіщень та видів подання актуальної інформації. Виконано їх порівняння з погляду актуальності , проаналізовано переваги та недоліки. Для розв'язання задачі та вирішення недоліків існуючих програмних рішень розроблено веб-сторінку в системі .

Веб-сторінка містить перелік конференцій та семінарів , доступних користувачу. Розроблено систему, що автоматично створю розклад актуальних конференцій . Виконано тестування розробленої системи.

Основні положення дипломної роботи опубліковано у вигляді тез доповідей на Міжнародній науково-практичній конференції молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики». Також було опубліковано статтю на Міжнародній науково-практичній онлайн конференції «Недінські читання».

Ключові слова: конференції,семінари,розклад, веб-система, поняття, теза, інформаційно-навчальний портал, онтологічне моделювання.

ABSTRACT

Thesis is done on 60 sheets, it contains 3 appendices and a list of references to the sources used with 15 items. The work shows 37 figures.

The purpose of this thesis is to develop a web page in the system using the schedule and current announcements about conferences and seminars.

The paper analyzes the existing solutions to the problem - notification options and types of presentation of relevant information. Their comparison in terms of relevance, analyzed the advantages and disadvantages. A web page in the system has been developed to solve the problem and solve the shortcomings of the existing software solutions.
The web page contains a list of conferences and seminars available to the user. A system has been developed that will automatically create a schedule of current conferences. The developed system has been tested.

The main provisions of the thesis were published in the form of abstracts at the International Scientific and Practical Conference of Young Scientists and Students "Modern Problems of Scientific Support of Energy". An article was also published at the International Scientific and Practical Online Conference "Nedin Readings".

Keywords: conferences, seminars, schedule, web system, concept, thesis, information-educational portal, ontological modeling

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

MVC – Model-View-Controller, архітектурний шаблон що використовується під час проєктування та розробки програмного забезпечення.

RSS – Rich Site Summary, сімейство XML-форматів, що слугує для постачання та публікації інформації, що часто змінюється.

HTTPS – Hyper Text Transfer Protocol Secure, протокол HTTP з додатковим шаром шифрування.

CMS – Content Management System, система управління контентом веб-сайту.

БД – база даних.

СКБД – система керування базами даних.

CSS – Cascading Style Sheets, каскадні таблиці стилів.

ECMAScript – стандарт мови програмування, затверджений міжнародною організацією ECMA.

HTML – Hyper Text Markup Language, стандартизована мова розмітки документів в мережі.

JVM – Java Virtual Machine, віртуальна машина, що виконує скомпільований байткод.

ACID – Atomicity, Consistency, Isolation, Durability, набір властивостей, що гарантують надійну роботу транзакцій баз даних.

SQL – мова структурованих запитів, що застосовується для управління даними в реляційних базах даних.

UNIX – сімейство переносних, багатозадачних і багатокористувацьких операційних систем.

Авторизація – керування рівнями та засобами доступу до ресурсу залежно від ідентифікатора і пароля користувача.

Автентифікація – процедура встановлення належності користувачеві певної інформації за допомогою наданого ним ідентифікатора.

HTTP – Hyper Text Transfer Protocol, прикладний протокол передачі даних у комп'ютерних мережах.

REST – Representational State Transfer, підхід до архітектури мережевих протоколів що забезпечують доступ до інформаційних ресурсів.

XML – стандарт побудови мов розмітки для даних що мають ієрархічну структуру.

JSON – текстовий формат обміну даними між комп'ютерами, що використовується переважно для передачі даних через мережу.

ВСТУП

З розвитком інтернету змінилося відношення до інформації. Смартфон є практично у кожного з нас, і в будь-який момент ми можемо знайти в мережі те, що нас цікавить. Змінився й сам підхід до інформації, немає необхідності зберігати інформацію на окремих носіях, набагато важливіше знати, де в Інтернеті можна знайти необхідну актуальну інформацію.

В даний час майже кожна організація має свій власний веб-сайт, на якому представлена її діяльність в мережі, і якщо його немає, це викликає сумнівні думки і знижує довіру до організації. Тобто сьогодні веб-сайт - це, по суті, лице компанії,яке справляє перше враження . І дуже важливо, щоб це враження було позитивним. Важливо також, щоб сайт мав необхідну функціональність та вміст, для якого його відвідують. Якщо користувачеві щось не подобається на веб-сайті, це може викликати розчарування , через що ці відчуття починають асоціюватись з організацією, і зменшують лояльність .

Сучасна статистика становить ,що близько 50% людей, використовують дизайн веб-сайтів як головний фактор при прийнятті рішень, запропонованих організацією. Майже 30% населення припинить користуватися сайтом, оскільки їх вміст буде непривабливим. Також 50% очікують ,що швидкістьзавантаження сторінки буде менше двох секунд, а 35% перестанутьпрацювати з веб-сайтом , якщо завантаження буде тривати понад шести секунд . Близько двох третин користувачів переглядають веб-сайти з мобільних пристроїв. Близько 30% сприймають відсутність дизайну як відсутність піклування про клієнтів, що зменшує їхню довіру до закладу. Близько 35% зазначили, що обрали б інший сайт у пошуку, якщо той, на який вони перейшли, не пристосований для перегляду на мобільних пристроях.

Основна мета дипломного проекту - створення нової сторінки сайту кафедри програмного забезпечення комп'ютерних систем Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського».

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

1.1. Аналіз предметної області

Функціонування веб-сайтів наукових підрозділів Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» регулюється статутом університету, чинним законодавством, наказами та розпорядженнями ректора, положеннями про інформаційно-освітній портал НТУУ «КПІ ", внутрішні розпорядження структурного підрозділу університету, накази проректора (перспективний розвиток). Згідно нормативних документів основними завданнями сайту кафедри є:

- обмін інформацією між різними кафедрами університету;
- вирішення наукових, інноваційних та освітніх завдань кафедри та університету за допомогою використання сучасних інформаційних технологій;
- оперативне та об’єктивне інформування викладачів, студентів, аспірантів, працівників, докторантів, випускників, здобувачів, ділових партнерів, а також інших зацікавлених сторін про різні аспекти діяльності кафедри;
- створення позитивного іміджу кафедри як університетського підрозділу, здатного конкурувати на міжнародному ринку послуг у галузі науки та освіти.

Поточна версія сайту не повністю відповідає вимогам, описаним у нормативних документах. Також дизайн та технічні характеристики веб-додатку не відповідають очікуванням користувачів, які звикли користуватися веб-додатками, зробленими з урахуванням сучасних тенденцій у галузі веб-розробки.

Метою даного дипломного проєкту є розробка модулю «Конференції та семінари» веб-сайту кафедри Національного технічного університету, яка покликана виправити наявні недоліки та привести роботу веб-сайту у відповідність із вищезазначеними нормативними документами.

1.2. Аналіз аналогічних веб-сторінок

З метою аналізу аналогів були розглянуті веб-сторінки різних закладів України та світу.

1.2.1. Веб-сторінка Ягеллонського університету у Кракові

Структура веб-сторінки [1] залежить від обраної мови інтерфейсу. Наступні розділи доступні для локалізації англійською мовою:

- про університет;
- навчання;
- персонал;
- дослідницька діяльність;
- міжнародне співробітництво.

Також на головній сторінці веб-сторінки, показаної на рис. 1, є посилання на сторінки університету в соціальних мережах. Якість медіа-контенту, представленого на сторінках веб-сторінки, висока.

До переваг цього рішення належать сучасний дизайн, використання безпечного протоколу передачі даних, можливість зміни розміру шрифту, можливість переходу на більш контрастну версію інтерфейсу, локалізація інтерфейсу для кількох мов, особистий кабінет, висока швидкість , мобільна підтримка, RSS-канал тощо.

Серверна частина написана використовуючи мову програмування Java. Клієнтська частина виконана використовуючи HTML, CSS та різних бібліотек JavaScript. До веб-додатку підключений модуль аналітики Google Analytics.

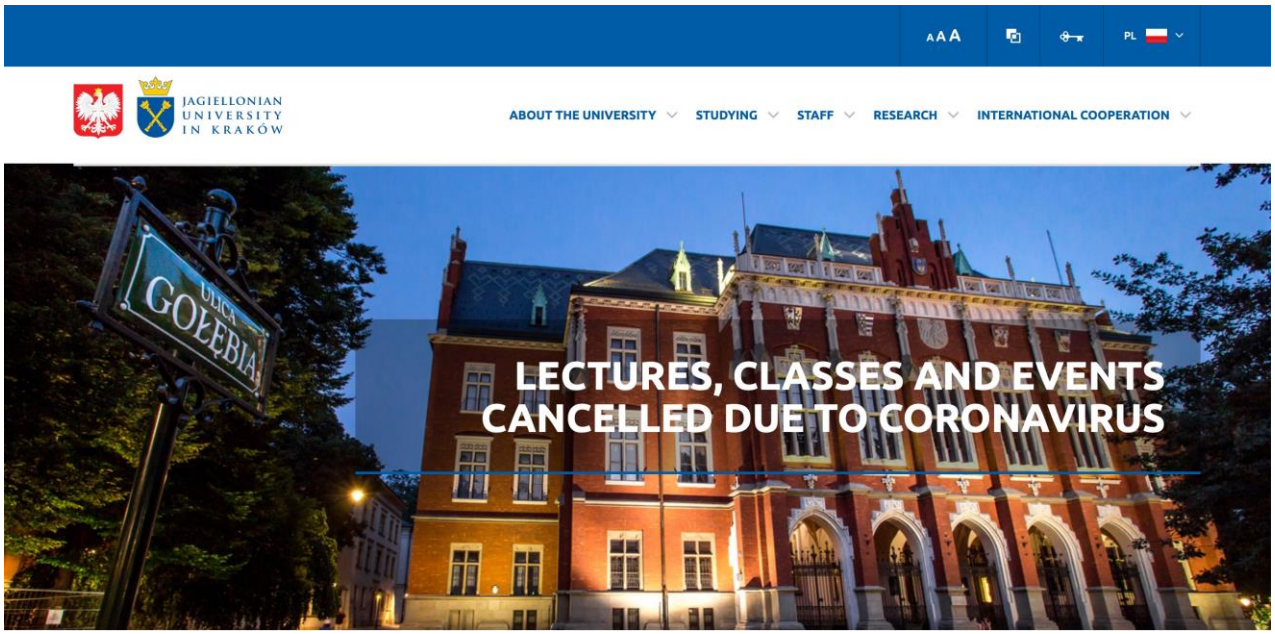


Рис. 1. Головна сторінка веб-додатку Ягеллонського університету

1.2.2. Веб-додаток Технічного університету Мюнхена

Структурно веб-додаток [2] поділено на такі розділи:

- інформація про університет;
- науковий розділ;
- новинки;
- навчання студентів;
- життя студентів;
- міжнародні зв'язки з студентами.

Фото контент у високій якості представлений на сторінках веб-сторінки. Також на головній вкладці, показаній на рис. 2, містяться посилання на університетські соціальні мережі

Серед переваг - мінімалістичний сучасний дизайн, відсутність зайвих елементів інтерфейсу, зручна навігація по сайту, використання безпечного протоколу передачі даних, висока продуктивність та стрічка новин RSS.

До недоліків можна віднести відсутність особистого кабінету користувача.

Рис. 2. Головна сторінка веб-додатку Мюнхенського технічного університету

1.2.2. Веб-додаток Національного медичного університету ім. О.О. Богомольця

Розділи структури сайту[3]:

- загальні відомості про університет;
- абітурієнська інформація;

- Інфорграфіка студентів;
- Короткі відповіді на питання;
- Навчання .

Серед переваг - наявність пошуку по сайту і наявність RSS-стрічок новин.

До недоліків можна віднести застарілий дизайн, повільну швидкість завантаження сторінки, відсутність підтримки мобільних пристроїв, відображення минулих авторських прав, відсутність окремого значка, деякі зображення взагалі не завантажуються.

Поділ доступу до інформації в залежності від мови носить суперечливий характер, з одного боку, дозволяє іноземним студентам економити час, не переглядаючи інформацію, яка їх не стосується, з іншого боку, сам інтерфейс не змінюється, а посилання на закриті посилання є тільки дратівливий.

Хоча на головній сторінці веб-додатки, показаної на рис.3, є посилання на сторінки вузу в соціальних мережах, деякі посилання працюють некоректно, перенаправляючи на головну сторінку ресурсу, а не на сторінку вузу. установа. Крім того, деякі посилання ведуть на ресурси, які перестали працювати або заблоковані в Україні.

Серверна частина розроблена на CMS WordPress з використанням мови програмування PHP і СУБД MySQL. Клієнтська частина розроблена з використанням HTML і CSS. Модуль Google Analytics підключений до веб-додатку.

Рис. 3. Головна сторінка веб-додатку НМУ ім. О.О. Богомольця

Результати аналізу наведено у табл. 1.

Таблиця 1

Порівняльна таблиця програмних рішень

Критерії / Програмні рішення	Сучасний дизайн	Швидкодія	Підтримка мобільних пристроїв	Захищене підключення
Ягеллонський університет	+	+	+	+
Мюнхенський технічний університет	+	+	–	+
НМУ ім. Олександра Богомольця	–	–	–	+

Дивлячись на результати аналізу приходимо до висновку ,що веб-сайт Яггелонського є найкращим серед усіх порівнюваних варіантів. Він перемагає завдяки сучасному дизайну, адаптивному інтерфейсу зі спеціальними покращеннями для людей з обмеженими фізичними можливостями.

1.3. Актуальність розробки веб-сторінки

На сьогоднішній день існує проблема комунікації та взаємодії між викладачами та студентами в умовах дистанційного навчання. І наразі, для вирішення цієї проблеми вони вимушені використовувати різне програмне забезпечення проведення навчального процесу: сервіси для спілкування, для поширення матеріалів курсу та для введення поточного контролю.

Для вирішення цієї проблеми може бути запроваджена веб система з наступним її інтегруванням у сайт кафедри чи факультету.

Така система дистанційної комунікації між студентами та викладачами є актуальною для будь-якого вищого навчального. Її головними перевагами є централізований доступ до сервісу через веб-браузер, зручний спосіб спілкування, можливість поширювати матеріали лекцій та переглядати студентами поточні оцінки.

Данна підсистема веб-застосунку дозволятиме студентам обмінюватися повідомленнями у чатах викладач-група, переглядати і завантажувати файли курсу на відведеній сторінці, переглядати розклад та поточний контроль.

Для реалізації серверної частини веб застосунку було вирішено використовувати фреймворк Node.js . Його головними перевагами є асинхронність, швидке підняття серверу, мінімальне використовування оперативної пам’яті та велика кількість інструментів для створення швидких і функціональних мікросервісів. Для зберігання даних будем використовувати реляційну систему управління базами даних PostgreSQL.

Для легко масштабування та для легкої інтеграції в мікросервісну архітектуру

вона є однією з найкращих СУБД з відкритим вихідним кодом. Так як під час розробки головного функціоналу мікросервісу оптимізація запитів до БД не є важливою, то для зручно зчитування/збереження об’єктів дана система об’єктно-реляційного відображення є кращим вибором. Для подальшої оптимізації роботи с СУБД будемо використовувати асинхронну бібліотеку.

Взаємодія між екземплярами мікросервісів буде розроблена з використанням програмного брокера Docker.

Для швидкої реалізації клієнтської частини з можливістю створювати гнучкий інтерфейс буде використано бібліотека React яка базується на мові програмування JavaScript.

1.4. Висновки до розділу

Портал повинен давати змогу учням та студентам обмінюватися навчальними матеріалами, повідомленнями (студент – студент, студент – викладач, викладач - група) та новинами про публічні заходи кафедри АПЕПС.Web-портал розрахований на 3 категорії користувачів: не зареєстровані користувачі, користувачі студенти, користувачі викладачі.

Відповідно до категорій розподіляється відповідний функціонал. Не Зареєстровані користувачі можуть лише переглядати новини кафедри АПЕПС, а доступ до чатів та можливість обмінюватися повідомленнями є тільки у зареєстрованих користувачів. Викладачі також можуть додавати та змінювати інформацію про навчальні матеріали.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Обґрунтування вибору мови програмування

Через те що ,вибраний створений програмний продукт повинен бути в вигляді сайту то вибір мови програмування був здійснений згідно най розповсюджених мов програмування, які використовуються для створення сайтів.

Вибрана мова програмування відповідає таким вимогам :

- Функціональність в розробці веб-сайту;
- наявність великої кількості бібліотек. Бібліотек доступних для використання;

Мова Javascript

Мова програмування Javascript є реалізацією стандарту ECMAScript. Належить до мов з слабкою типізацією. В основному призначений для створення динамічного вмісту в браузерах користувачів. В середовищі Node.js стало можливим використовувати цю мову на стороні сервера, що дозволило професіоналам розробляти повноцінні веб-сторінки з використанням однієї мови програмування. JavaScript підтримує різні парадигми програмування.

До переваг JavaScript належать:

• надзвичайно швидкий розвиток;

• можливість вставляти код JavaScript у розмітку HTML;

• можливість виконувати код не тільки в браузері, але і на сервері;

• широкий вибір готових бібліотек.

Основним недоліком використання мови програмування Java є динамічне введення тексту.

На момент свого створення він набув широкої популярності завдяки проривній концепції віртуальної машини, яка дозволила досягти крос-платформенності. Код Java компілюється в байт-код, а потім може використовуватися на будь-якій платформі, для якої реалізована віртуальна машина. Раніше його часто використовували для написання програм для різних мобільних пристроїв та побутової техніки, зараз він використовується в основному в корпоративному сегменті для створення та обслуговування високонадійних систем. Хоча Java є багато парадигматичною мовою програмування, деякі парадигми відрізняються

від їхньої загальноприйнятої форми і тому широко не використовуються.

До переваг Java належать:

• стабільність роботи та доступність перевірених часом рішень;

• повна зворотна сумісність розроблених додатків;

• лише незначні зміни в синтаксисі протягом існування мови.

До недоліків Java часто відносять «багатослівний» синтаксис, а також недостатню продуктивність без додаткових налаштувань віртуальної машини, та довгу компіляцію коду, що сповільнює розробку.

HTML

HTML не є мовою програмування, тобто вона не має можливості створювати динамічні функції. Замість цього вона дозволяє організовувати і форматувати документи, аналогічно Microsoft Word.

При роботі з HTML ми використовуємо прості структури коду (теги і атрибути), щоб розмітити сторінку веб-сайту. Наприклад, ми можемо створити абзац, помістивши додається текст у вихідний тег <p> і закриває </ p>.

В цілому, HTML - це мова розмітки, яка дуже проста в освоєнні навіть для початківців в створенні сайтів.

Visual Studio Code – редактор вихідного коду

Visual Studio Code - це редактор вихідного коду, розроблений Microsoft для Windows, Linux та macOS. Він включає вбудований Git та підтримку налагодження, виділення синтаксису, інтелектуальне завершення коду, фрагменти та рефакторинг коду. Він гнучко налаштовується, що дозволяє користувачам змінювати тему, комбінації клавіш, налаштування та встановлювати розширення, що додають додаткову функціональність. Вихідний код - вільний та відкритий, випущений згідно з ліцензією MIT. Скомпільовані двійкові файли безкоштовні для будь-якого використання.

Обрана мова програмування

В результаті аналізу вище прелічених мов були визначені критичні недоліки що змушують відмовитись від їх використання для розроблення даної системи:

- розробка на мові C# вимагає використання програмного забезпечення з комерційною ліцензією

• Python, хоча і допомагає стрімко реалізувати програмний продукт ,але тільки для справжніх знавців мови.

При використанні цієї мови, враховуючи те, що розробники цього продукту не мають такого досвіду, вищезазначена перевага повністю компенсується, враховуючи, що, як і низько швидкісний Python, це не найкращий вибір для розробки;

• Мова програмування Java - не найкращий вибір через його відносно надлишковий синтаксис і повільну швидкість.

Тому, беручи до уваги попередній аналіз та вимоги до веб-додатку, Javascript був обраний мовою програмування розробленого програмного забезпечення. Ця мова має простий і зрозумілий синтаксис, розвивається дуже швидко, має широкий вибір бібліотек і фреймворків і може використовуватися для розробки як клієнтської, так і серверної частин.

2.2. Обґрунтування вибору бази даних

2.2.1. Огляд бази даних та вибір моделі

Усі існуючі бази даних поділяються на два типи: реляційні, або, як їх ще називають, SQL, і нереляційні, або NoSQL. Різниця між ними полягає в тому, яку модель даних вони представляють: реляційну чи нереляційну. Реляційні бази даних представляють дані у вигляді таблиць, на відміну від нереляційних, де спосіб подання даних залежить від типу самої бази даних.

Нереляційні бази даних поділяються на:

• стовпчастий (стовпчастий);

• документальний;

• пара "ключ-значення";

• підраховує;

• об’єкт.

Реляційні бази даних зберігають дані, структура яких чітко визначена на момент створення бази даних. Для взаємодії з базою даних використовується спеціальна мова запитів SQL. Оператори цієї мови дозволяють вводити нові дані до бази даних, оновлювати та видаляти їх, а також читати. Зовнішні ключі використовуються для приєднання таблиць баз даних один до одного, що також вирішує проблему невідповідності даних. Хоча всі бази даних використовують SQL для доступу до бази даних, більшість із них мають стандартні можливості SQL, які дозволяють виконувати всі необхідні операції, просто без додаткової зручності.

Плюси реляційних бази даних [8]:

• зрозуміла структура даних;

• підтримка SQL;

• надійність ,оскільки вони новіші ніж нереаліційні.

• частота використання. Оскільки наразі всі користуються реляційними базами

даних ,то і спеці ластів в цій сфері набагато більше.

• постійна підтримка клієнтів.Безпечне використання.

• підтримка механізму транзакцій ,надійність забезпечується ACID;

• можливість збільшення ефективності роботи використовуючи індексацію;

• Потужні вбудовані засоби обробки даних, включаючи тригери та процедури.

Мінуси:

- • швидкість - для правильного представлення взаємозв'язку між сутностями даних у реляційних базах даних використовують процес, який називається нормалізацією,що через велику кількість таблиць зумовлює потребу їх з’єднання у запитах і призводить до пониження швидкості і закручування запитів для підвищення ефективності яких використовують зворотній процес денормалізації бази даних, що призводить до проблеми надмірності даних, що означає що при використанні реляційних БД часто виникає потреба шукати компроміс між нормалізацією та швидкодією;

  • масштабування - незважаючи на можливість вертикального масштабування, важко організувати горизонтальне масштабування реляційної бази даних;

  • складна зміна схеми бази даних - при використанні реляційних баз даних структура даних повинна бути визначена заздалегідь, а це означає, що чим складніше схема бази даних, тим складніше її змінити в майбутньому;

  • невідповідність реляційної моделі певному класу проблем - незважаючи на універсальність реляційної моделі, існують проблеми, для яких її використання створює більше проблем, ніж у випадку використання бази даних із відповідною моделлю представлення даних, серед яких можна виділити проблему може бути проблематичним через велику кількість з'єднань таблиць у запитах.

Бази даних NoSQL дозволяють зберігати неструктуровані або напівструктуровані дані. Ці дані моделюються засобами, які відрізняються від тих, що використовуються при побудові реляційних баз даних. Бази даних

NoSQL зазвичай використовуються для вирішення ряду конкретних проблем.

Сьогодні їхня популярність неухильно зростає через зростаючу складність систем програм. Хоча іноді рішення NoSQL можуть підтримувати подібні до SQL запити, більшість із них мають власні механізми обробки даних. Відсутність чітко визначеної структури полегшує дизайн бази даних та робить її більш гнучкою, що робить деякі операції швидшими, ніж їх аналоги SQL. Перепонами для більшого використання є відсутність стандартизації та наявність стрімко набираю сих популярність реляційних аналогів з активнішою підтримкою. Відсутність механізму транзакцій також не покращує їх становище.

Переваги NoSQL [9]:

• масштабованість - бази даних NoSQL підтримують звичне горизонтальне масштабування з механізмом заточування, що дешевше і доступніше, ніж вертикальне;

• швидке відновлення - механізм автоматичної реплікації забезпечує копіювання даних і швидке відновлення до збереженого стану у випадку збою системи незалежно від причини.

Мінуси NoSQL:

- вузька спеціалізація – використовуються зазвичай, не як загальне рішення, а для вирішення якоїсь специфічної задачі;
- відсутність стандартизації через молодий вік NoSQL рішень, а також той факт, що найчастіше такі БД є рішеннями з відкритим кодом що постійно змінюється;

2.1.2. Аналіз СКБД MySQL

MySQL - це система керування базами даних з відкритим кодом, заснована на базі даних mSQL. В даний час Oracle є власником продукту. База даних переноситься на всі сучасні платформи, включаючи Mac, Windows, Linux і Unix. Основна область застосування - розробка веб-додатків. Поточна версія - 8.0.17.

MySQL основана на моделі клієнт-сервер, де ядром є сервер MySQL, який обробляє всі запити бази даних і доступний в двох варіантах: утиліта окрема ,яка створенна для використання в становищі і як бібліотека, яка може бути використана в іншому додатку [10]. Також для цієї бази даних є клієнт з адаптивним графічним інтерфейсом споживача.

MySQL підтримує великі бази даних, а також різноманітні типи даних.Включаючи цілі числа, числа з плаваючою комою, різні формати дат і виконавчі дані. Також є підтримка типів рядків змінної і фіксованої довжини.

Для забезпечення безпеки MySQL підтримує систему для шифрування паролів та інформації в базі даних.

Переваги MySQL:

- популярність – одна з найпопулярніших систем, що означає відсутність дефіциту висококваліфікованих розробників з досвідом використання даної БД, а також наявність великої кількості документації та навчальних матеріалів;
- безпечність – MySQL повністю захищає вхідні данні користувачів,за допомогою обмеження доступу до даних.
- швидкість –можливості не були повністю реалізовані , що дозволило підвищити продуктивність.

Недоліки MySQL:

• взаємопов'язаність: з самого початку SDBD була розбита з акцентом на швидкість і простоту перемоги за рахунок невисокої важливості можливостей SQL;

• Платні функції - MySQL доступний з двома ліцензіями: відкритої і комерційної, а деякі функції доступні тільки за плату;

• Гарантія розвитку - через викуп продукції, частина розробки SKBD, вірячи в те, що виробництво товарів не може проводитися швидше, оскільки шлях розвитку здійснюється компанією;

• Зниження продуктивності при великому розмірі податку - при збільшенні кількості податків в одну сторону підвищення продуктивності за рахунок архівування індексу.

2.1.3. Аналіз СКБД PostgreSQL

PostgreSQL - це об'єктно-реляційна концепція управління основами відомостей сформована так само як open-source план у Каліфорнійському університеті в Філософ під 1986 р .. Спочатку створена з метою платформи UNIX, також в подальшому перенесена в інші поширені платформи .. PostgreSQL вважається планом зі цілком не закритим початковим кодом загальнодоступних близько ліцензією PostgreSQL. Дана СУБД виділяється стійкістю діяльність також легкістю допомоги.

Відмінними рисами PostgreSQL вважається ймовірність доповнювати особисті функції, створені в різних стилях програмування з метою розширення типових функціональних можливостей, але крім того ймовірність формування особистих видів відомостей, індексів також поліадельфіт.буква. [11].

Переваги PostgreSQL:

• синхронні також оптимізовані вимоги;• надання елементів блокування;

- підтримка індексів: btree, hash, тощо;
- розподілені таблиці;

• мінливість способів реплікації: одночасна також асинхронні повторення, логічна також фізіологічна, неповна також поліадельфіт.буква .;

• допомога SSL;

• допоміжні види відомостей - крім типових видів властивих будь-якої реляційної основі, але крім того певних додаткових видів своєрідних з метою PostgreSQL, можливо формувати власні особисті;

• присутність драйверів з метою багатьох нинішніх платформ також стилів програмування;

• ймовірність повнотекстового розшуку;

• open-source - проект має цілком публічний початковий код також підтримується суспільством, що захоплюється його допомогою, вирішує напрям формування та відповідає за створення документації.

Недоліки PostgreSQL:

• популярність - історичне запізнення популярності від MySQL, обумовлюється відсутністю фірм власників які просували результат, привела до відносно найменшого числа сторонніх приладів але крім того найменшого числа експертів з навиком адміністрування цієї СУБД;

• знижена ефективність із-за відділення додаткової пам'яті на будь-яке під'єднання до бази.

Беручи до уваги також недоліки перерахованих вище висновків з метою дослідження цього дипломного плану була підібрана СУБД PostgreSQL через великі функціональні можливості, але крім того ,що вона є цілком відкритою.

2.2. Обгрунтування вибору фреймворку серверної частини

2.2.1. Аналіз Note.js

Node.js створений під впливом таких систем як Event Machine в Ruby або Twisted в Python. Node.js використовує подієву модель значно ширше, він приймає цикл подій (event loop) за основу оточення, замість того, щоб використовувати його в якості бібліотеки. В інших системах завжди стається блокування виклику, щоб запустити цикл подій.

Зазвичай поведінка визначається через функції зворотнього виклику на початку скрипта і в кінці запускає сервер через блокуючий виклик, як от EventMachine::run(). В Node.js немає нічого подібного на виклик початку циклу подій. Node.js просто входить в подієвий цикл після запуску скрипта на виконання. Node.js виходить з подієвого циклу тоді, коли не залишається зареєстрованих функцій зворотнього виклику. Така поведінка схожа на поведінку браузерного JavaScript: подієвий цикл прихований від користувача.

Те що Node.js спроектований без багатопоточності, не означає, що ви не можете використовувати можливості кількох ядер у вашому середовищі. Ви можете створювати дочірні процеси, якими легко керувати з допомогою API child_process.fork(). Модуль cluster побудований на цьому інтерфейсі і дозволяє вам ділитись сокетами між процесами та розподіляти навантаження між ядрами.

Цей фреймворк стабільно потрапляє в списки найбільш популярних, перспективних і використовуваних javascript фреймворків і отримує такі звання як кращий фреймворк корпоративного рівня і кращий фреймворк для особистих проектів.

2.2.2. Дослідження Meteor.js

Meteor.js [13] - інтернет-фреймворк із не закритим початковим кодом створений на JavaScript зі застосуванням Node.js. Створений фірмою Meteor Development Group в 2011 р .. Meteor дає можливість стрімко формувати крос платформні інтернет-додатки також додатки для мобільних прикладів. Фреймворк застосовує акт розподілених даних та зразок проектування publish-subscribe для того щоб автоматом освіжати відомості на клієнті та виключити потреби записувати відповідний код з метою синхронізації.

Беручи до уваги вищевказані властивості з метою дослідження серверної частки був обраний фреймворк Note.js через значну популярність, що в свою чергу означає присутність найбільшого числа оброблених бібліотек також значну допомогу суспільства.

2.3. Висновки до розділу

У цьому розділі було прокладено отримання вимог по технологіям дослідження, але безпосередньо мови програмування, бази данних та фреймворка з метою дослідження серверної частини зі урахуванням особливостей розроблюваного програмного забезпечення.

Було здійснено дослідження найбільш поширених мов програмування з метою дослідження інтернет-додатків: Python, JavaScript, Java, C #. Існували встановлені їхні переваги також недоліки.

Був обраний мова Javascript за комфорт її застосування з метою комплексної розробки серверної та частини споживача, присутність великої кількості бібліотек, крос-платформність.

Здійснено дослідження SQL також NoSQL баз данних, у слідстві якого було встановлено, що з метою дослідження цього правильніше застосовувати реляційну основу відомостей. Уже після цього було проаналізовано СУБД MySQL також PostgreSQL, так як більш поширених представників БД цього виду, встановлені їхні переваги також недоліки. У слідстві розгляду з метою дослідження цього інтернет-доповнення було підібрано СУБД PostgreSQL.

Переглянуто інтернет-фреймворки Note.js також Meteor.js з метою форсування дослідження серверної частки. З Метою дослідження був підібраний перший через більшу популярність .

3. РОЗРОБЛЕННЯ ВЕБ-ДОДАТКУ

3.1. Загальний опис системи

Інформація розміщена на веб-сторінці має бути подана у комфортному для читання вигляді . Адміністратору має бути надана функціональна можливість редагування інформації.

Веб-сторінка містить розділи з наступною інформацію:

- Вся інформація про лекторів;
- актуальний розклад;
- графік частоти відвідування;
- освітні програми;
- посилання на минулі конференції;
- перелік тем конференцій та семінарів;
- сторінка запису на конференцію чи семінар
  Веб-сторінка повинна мати працездатне RSS

3.2. Архітектура розробленої системи

Клієнт-серверна модель архітектури була підібрана для реалізації програмного коду .

Клієнт-серверна архітектура – шаблон ПЗ, побудований на дії клієнтів та серверів за допомогою мережі (рис. 4). Клієнти та сервери представлені у вигляді програмних модулів що активні незалежно від роботи один одного. Дія має наступний вигляд: клієнт відправляє запит серверові, сервер читає його і повертає відповідь обробки клієнту. Зазвичай у ролі клієнта є веб-браузер [15].

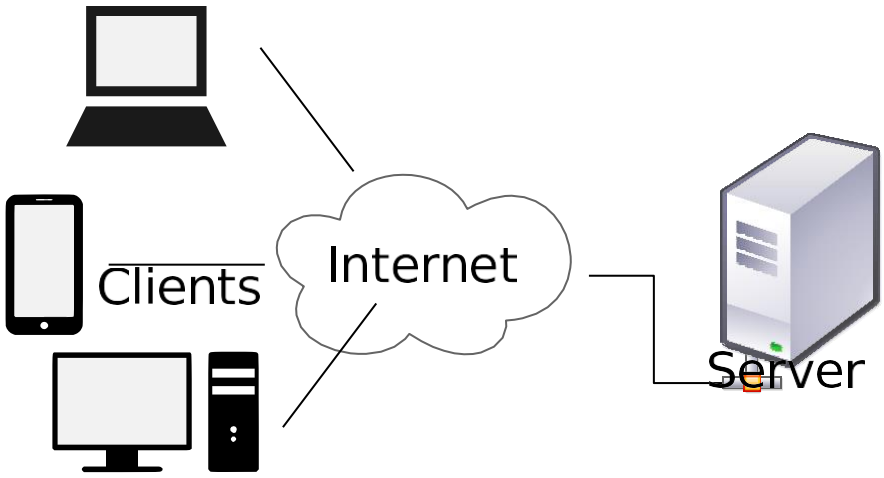


Рис. 4. Клієнт-серверна архітектура

3.3. Особливості реалізації

Реалізація серверної частини виконана на основі шаблону проєктування MVC (Model-View-Controller) з використанням фреймворку Note.js.

Модель бази даних

З метою реалізації розроблюваного застосунку було написано 8 взаємозв’язанихміж собою табличок:

- Staffi;
- Exam_timetabli;
- Consultationi_timetable;
- Publicationsi;
- Publicationi_items;
- Graduationi_papers;
- Curriculum;
- Usersi.

Схему зображено на рис. 5.

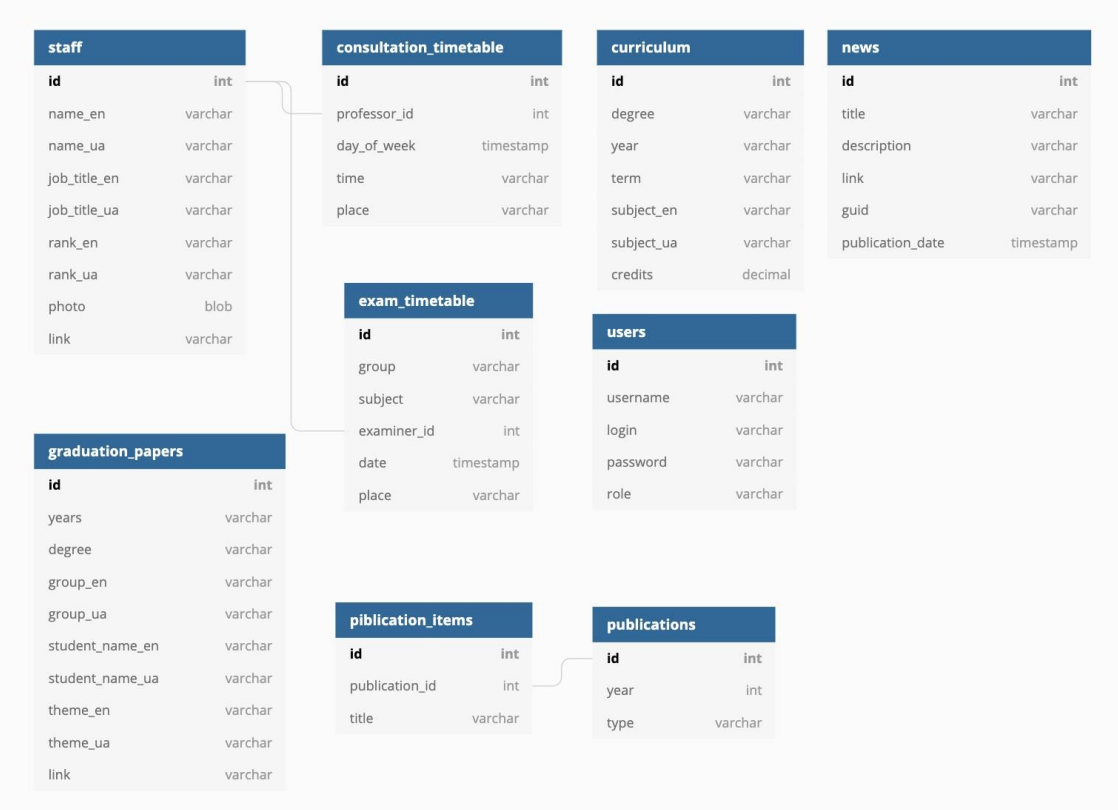


Рис. 5. Схема бази даних

Staff

Таблиця містить у собі інформацію про склад лекторів. Обширний опис полів приведено у табл. 2.

Таблиця 2

Поля таблиці Staff

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
name_en	TEXT
name_ua	TEXT
job_title_en	TEXT
job_title_ua	TEXT
rank_en	TEXT
rank_ua	TEXT
photo	BLOB
link	TEXT

Exam_timetable

Дана таблиця представляє інформацію про розклад семінарів та конференцій . Деталізований опис назв та типів полів у табл. 3.

Таблиця 3

Поля таблиці Exam_timetable

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
group	TEXT
subject	TEXT
examiner_id	INT (Foreign Key of Staff)
date	TIMESTAMP
place	TEXT

Consultation_timetable

Дана таблиця несе в собі інформацію про розклад конференцій та семінарів лекторів. Детальний опис у табл. 4.

Таблиця 4

Поля таблиці Consultation_timetable

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
professor_id	INT (Foreign Key of Staff)
day_of_week	TEXT
time	TEXT
place	TEXT

Publications

Ця таблиця візуалізує загальну інформацію про елемент архіву публікацій.

Таблиця 5

Поля таблиці Publications

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
year	INT
type	TEXT

Publication_items

Дана таблиця заповнена конкретною інформацією про публікацію. Пов’язана з таблицею Publications у відношенні один до багатьох.

Таблиця 6

Поля таблиці Publication_items

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
publication_id	INT (Foreign Key of Publication)
title	TEXT

Graduation_papers

Таблиця 7

Поля таблиці Graduation_papers

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
years	TEXT
degree	TEXT
group_en	TEXT
group_ua	TEXT
student_name_en	TEXT
student_name_ua	TEXT
theme_en	TEXT
theme_ua	TEXT
link	TEXT

Curriculum

У таблиці учбовий план для різних науковихступенів.

Таблиця 8

Поля таблиці Curriculum

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
degree	TEXT
year	INT

Продовження табл. 8

term	INT
subject_en	TEXT
subject_ua	TEXT
credits	DECIMAL (2,2)

Users

Інформація про усіх авторизованих споживачів системи міститься у цій таблиці . Детальний опис назв та типів полів наведено у табл. 9.

Таблиця 9

Поля таблиці Users

Назва поля	***Тип даних***
id	SERIAL AUTOINCREMENT
username	TEXT
login	TEXT
password	TEXT
role	TEXT

Взаємодія з клієнтською частиною

Взаємодія між частиною клієнта та сервера виконується за допомогою викликів REST API. Схема взаємодії наведена на рис. 6.

Для забезпечення сумісності в рамках Express існують спеціальні обробники функцій HTTP-запитів, які називаються проміжним програмним забезпеченням. Вони беруть функцію, яка містить два типи аргументів:

• Запит - містить усі дані про HTTP-запит, що надійшов на сервер;

• Відповідь - містить відповідь від сервера , надіслану клієнту.

Лише користувачі, авторизовані в системі, можуть використовувати механізми REST API. Виклик користувачів, які не пройшли процедуру авторизації, поверне відповідь із статусом HTTP 401 (Не авторизовано).

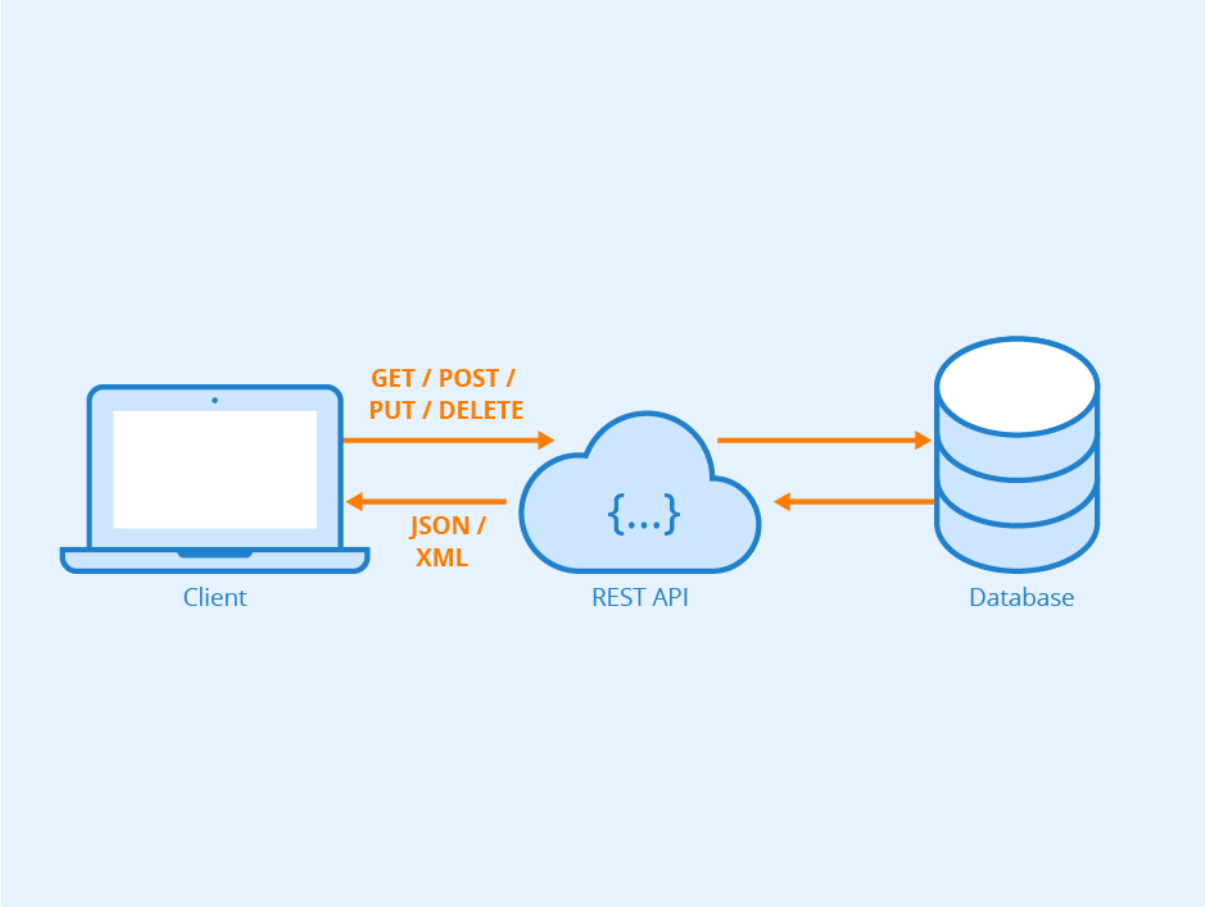


Рис. 6. Схема взаємодії через REST API

Авторизація та автентифікація

Розроблена система містить можливості авторизації та автентифікації що належать до виду Session Based Authentication, принцип дії якої зображено на рис. 7.

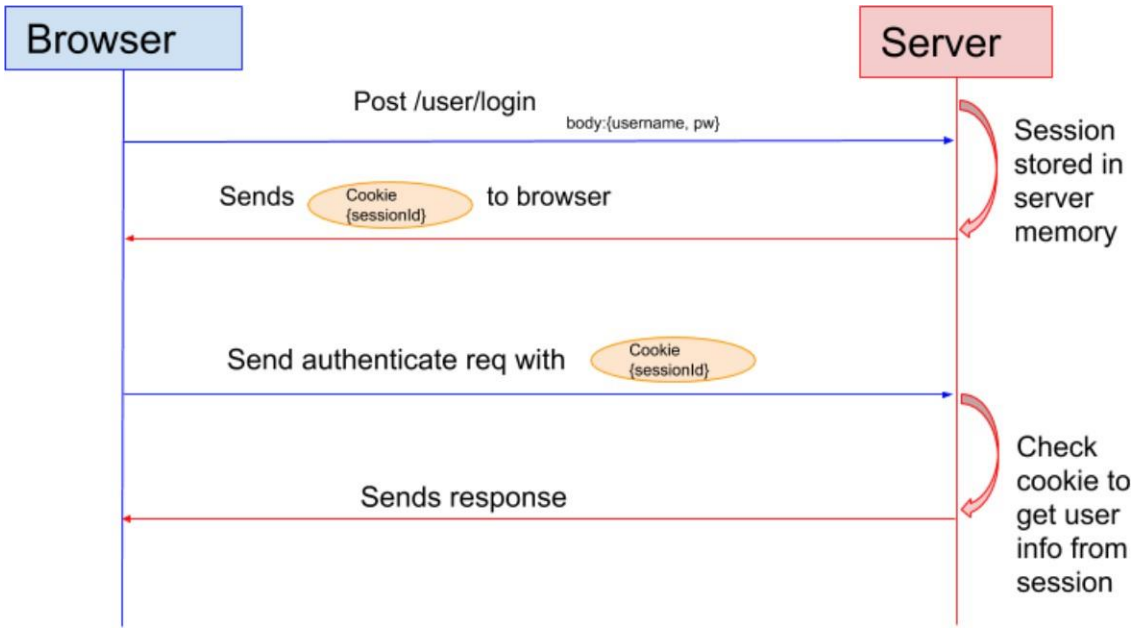


Рис. 7. Session Based Authentication

Робота авторизації та автентифікації оформлена за допомогою модуля passport.js.

Існує три типи користувачів:

• Відвідувач - будь-який користувач сайту, який не ввійшов у систему. Такі користувачі можуть лише переглядати вміст веб-програми, але не змінювати його;

• Користувач - уповноважений користувач, для якого доступна функціональність модифікації вмісту веб-програми;

• Адміністратор - адміністратор, який має всі можливості звичайного користувача та може додавати та видаляти інших користувачів, які не є адміністраторами.

Реєстрація нових користувачів проводиться через адміністратора.

Стрічка новин RSS

Стрічка новин або RSS-стрічка надає користувачам нову інформацію, що з'явилася на сайті, а також посилання на повну версію. Майже всі актуальні веб-браузерів можуть працювати з RSS-каналами. Крім того, існують спеціальні програми (агрегатори RSS), які збирають та обробляють дані з RSS-каналів.

Модуль подачі npm був використаний для реалізації стрічки RSS. Приклад елемента стрічки новин показано на рис. 8.

Рис. 8. Елемент стрічки новин RSS

```
<link>http://pzks.fpm.kpi.ua/news/bf86345f-a178-4a51-805f-34c25cab3527</link>
<guid>bf86345f-a178-4a51-805f-34c25cab3527</guid>
```

3.4. Висновки до розділу

У цьому розділі наведено загальний опис серверної частини веб-сайту, що розробляється. Цей розділ містить схему бази даних, детальний опис імен та типів полів таблиці та взаємозв'язків між ними. Описано принцип взаємодії між клієнтською та серверною частинами за допомогою REST API, а також особливості реалізації цієї взаємодії під час розробки цієї системи. Реалізація автентифікації на основі сеансу була описана для забезпечення авторизації та автентифікації користувачів системи, що використовують модуль passport.js. Також було описано реалізаціюстрічки новин RSS, за допомогою модуля feed.

4. АНАЛІЗ РОЗРОБЛЕНОГО ВЕБ-ДОДАТКУ

4.1. Аналіз реалізованого веб-сторінки

Результатом розробки є веб-сторінки, виконаний з урахуванням сучасних тенденцій розробки ПЗ. Основною функцією веб-сторінки є представлення інформації про конференції та семінари.

Відповідно до завдань та вимог сервера були представлені наступні рішення:

• використовується закритий протокол передачі даних

• впроваджено RSS;

• запроваджено систему редагування вмісту;

• дані були передані зі старої версії веб-програми;

• забезпечується взаємодія з базою даних;

• реалізована взаємодія з клієнтською частиною;

• досягається висока швидкість нанесення.

Користувачі також можуть змінювати інформацію про інших користувачів системи. Взаємодія частин клієнта та сервера реалізується за допомогою REST API. На рис. 9. показує приклад типового RESTful API для модуля новин.

```
app.get('/api/news', db.getNews);
app.get('/api/news/:id', db.getNewsById);
app.post('/api/news', db.createNews);
app.put('/api/news/:id', db.updateNews);
app.delete('/api/news/:id', db.deleteNews);
```

Рис. 9. RESTful API для модулю новин

Програмний інтерфейс для інших модулів слід аналогічним сценарієм. Взаємодія з базою даних відбувається за допомогою SQL-запитів.

За попереднім записом запити бувають наступних видів:

• SELECT - використовується для читання даних з бази даних;

• INSERT - використовується для додавання нових даних в базу даних;

• ОНОВЛЕННЯ - використовуються для поновлення даних, наявних в базі даних;

• ВИДАЛИТИ - використовується для видалення даних з бази даних.

4.2. Тестування веб-додатки

Невідємною частиною розробки ПЗ процесу є тестування розробки і проводиться на кожному етапі і допомагає виявити помилки і недоробки на ранніх етапах, коли вартість їх усунення не надто висока. Тестування із залученням майбутніх споживачів важливо облік відгуків користувачів допомагає більш точно визначити потреби ринку і розставити пріоритети в розробці елементів системи. Основні цілі тестування - знайти помилки, перевірити зручність використання графічного інтерфейсу і перевірити правильність роботи програми в цілому.

При розробці веб-додатки тестування проводилося після впровадження кожного модуля системи, що не дозволяло усувати помилки відразу після виявлення, що не дозволяло їм впливати на подальший процес розробки.

Після ручного тестування інтерфейсу користувачі внесли деякі зміни в дизайн веб-програми, що допомогло покращити сприйняття інформації та зовнішній вигляд інтерфейсу в цілому.

Щоб перевірити правильність запитів, надісланих на сервер, а також перевірити відповіді сервера, ми використали браузер Google Chrome та його вбудовані інструменти розробника.

Перевірка правильності запису, оновлення та видалення даних у базі даних проводилася за допомогою інтерфейсу консолі, що входить до дистрибутивного пакету PostgreSQL.

Для перевірки функціональності серверної сторони були написані автоматизовані тести; у майбутньому це полегшить рефакторинг та внесення змін, не впливаючи на роботу інших компонентів системи.

В результаті тестування була отримана цінна інформація, яка допомогла знайти та виправити проблеми та недоліки у роботі веб-програми, а також інформація, яка допомогла визначити шляхи подальшого вдосконалення системи.

4.2. Порівняння розробки з існуючими аналогами

У підрозділі 1.2 дипломного проекту були розглянуті веб-заявки вищих навчальних закладів України та Європи. Проведено аналіз переваг та недоліків подібних програм. Кожен з них розроблений відповідно до норм установ, які вони представляють. Функціональність подібних додатків також варіюється залежно від потреб навчальних закладів.

Основними критеріями оцінки були:

• сучасний дизайн - після опитування користувачів було встановлено, що дизайн веб-додатку не виглядає застарілим;

• швидкодія - після тестування було визначено, що за швидкодією розроблене ПЗ не поступається, а іноді і перевершує аналоги;

• зручний інтерфейс - інтерфейс веб-додатки розроблений таким чином, щоб користувачі могли інтуїтивно розуміти, де знайти потрібну їм інформацію;

• наявність захищеного протоколу передачі даних - розроблений програмний продукт підтримує HTTPS, що захищає користувачів від втрати їх персональних даних;

• кроссплатформенность - розроблене програмне забезпечення виконано у вигляді веб-додатки, що дозволяє використовувати його з будь-якого пристрою, що має веб-браузер з підтримкою JavaScript;

• Доступність RSS - розроблене програмне забезпечення містить новинну стрічку RSS, яка є поширеним форматом поширення новин для різних веб-додатків, включаючи веб-додатки освітніх установ.

Тому, порівнюючи аналоги в їх поточної реалізації, можна сказати, що розроблений сайт відповідає вимогам і є гідним конкурентом.

4.2. Рекомендації щодо подальшого поліпшення

Були визначені наступні шляхи подальшого поліпшення:

• впровадження форми зворотного зв'язку, яка допоможе поліпшити комунікацію між користувачами і адміністраторами веб-додатки;

• впровадження архіву матеріалів, який дозволить зберігати всю матеріально-технічну базу в одному місці і вирішити проблему пошуку студентами необхідних матеріалів;

• Удосконалення систем редагування вмісту для адміністраторів веб-додатків шляхом введення більш зручного інтерфейсу, який пришвидшить процес та уникне помилок модифікації;

• запровадження особистого кабінету користувача, який допоможе організувати навчальний процес та спілкування між співробітниками кафедри та студентами через веб-сайт. Наприклад, сайт включатиме завдання, які повинні виконати студенти. Виконані завдання завантажуються на сайт студентами, викладачі перевіряють їх і ставлять оцінки або відправляють завдання на доопрацювання;

• розширення можливостей інтерфейсу для людей із вадами зору: зміна розміру шрифту, контрастності та колірної схеми елементів інтерфейсу тощо;

• впровадження модуля статистики, який допоможе відстежувати трафік веб-додатків;

• додати систему захисту від DoS / DDoS-атак;

• забезпечення автоматичного розширення для витримки великих навантажень. У поточній реалізації максимальне навантаження на систему залежить від потужності машини, на якій працює серверне програмне забезпечення. За необхідності ви можете оновити систему таким чином, що коли поточне навантаження на машину наближається до межі, копія служби запускається на іншій машині і частина запитів користувачів перенаправляється на неї;

• Забезпечити заточування та реплікацію бази даних, щоб підвищити її стійкість та полегшити відновлення у разі несправності, а також збільшити навантаження, яку база даних може витримати;

• створення інформаційних сторінок кафедри в різних соціальних мережах та надання інтерфейсних посилань на них, а також посилання на веб-сайти співробітників кафедри, які безпосередньо пов’язані з організацією навчального процесу.

4.2. Висновки до розділу

У цьому розділі аналіз впровадженого програмного забезпечення проводився у формі веб-сайту кафедри програмного забезпечення для комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут Ігоря Сікорського». Після порівняння кінцевого продукту з аналогами було встановлено, що він якісно перевершує більшість вітчизняних аналогів і майже не поступається закордонним аналогам. Були визначені шляхи вдосконалення сайту, серед яких є варіанти різної складності та часу виконання, аж до перетворення сайту в повноцінну освітню платформу. Деякі шляхи вдосконалення ще неможливі через те, що поточне навантаження на систему не є великим, але в міру збільшення навантаження актуальність цих удосконалень також зростає. Процес тестування, проведений на кожному етапі розробки, також був описаний і містить як ручне, так і автоматичне тестування із зазначенням інструментів, які використовувались для тестування певних частин системи.

ВИСНОВКИ

Метою даного дипломного проєкту було створення веб-сторінки модуля конференції та семінари. Перед розробкою був проведений аналіз нормативних документів, що регламентують роботу веб- додатків підрозділів університету, а також порівняльний аналіз веб-сторінок інших сайтів. На основі проведеного аналізу були визначені вимоги до розроблюваного програмного забезпечення.

У даному проєкті проаналізовано та вибрано програмні засоби реалізації. Для реалізації серверної частину обрано мову програмування JavaScript та веб-фреймворк Note.js, реляційну базу даних MySQL.

У дипломному проєкті розроблено архітектуру серверної частини веб-додатку, структуру бази даних, стрічку новин RSS. За рахунок впровадження системи авторизації та автентифікації реалізовано розділення функціональних можливостей залежно від типу користувачів, а також захист від несанкціонованого доступу.

Розробка програмного забезпечення виконана у відповідності до поставлених вимог у повному обсязі. Тестування системи проведено згідно зі затвердженою програмою та методикою тестування.

Пояснювальна записка також містить опис реалізованої системи та рекомендації щодо її вдосконалення.

Впровадження результатів розробки допоможе:

- покращити обмін інформацією між студентами та викладачами;

- вирішити науково-інноваційні та освітні задач кафедри і університету за допомогою використання сучасних інформаційних технологій;
- оперативно та об'єктивно інформувати викладачів, студентів, аспірантів, співробітників, докторантів, випускників, абітурієнтів, ділових партнерів, а також інших зацікавлених осіб про різні аспекти діяльності кафедри;
- створити позитивний імідж кафедри, як університетського підрозділу, що здатен конкурувати на міжнародному ринку послуг у сфері науки та освіти.

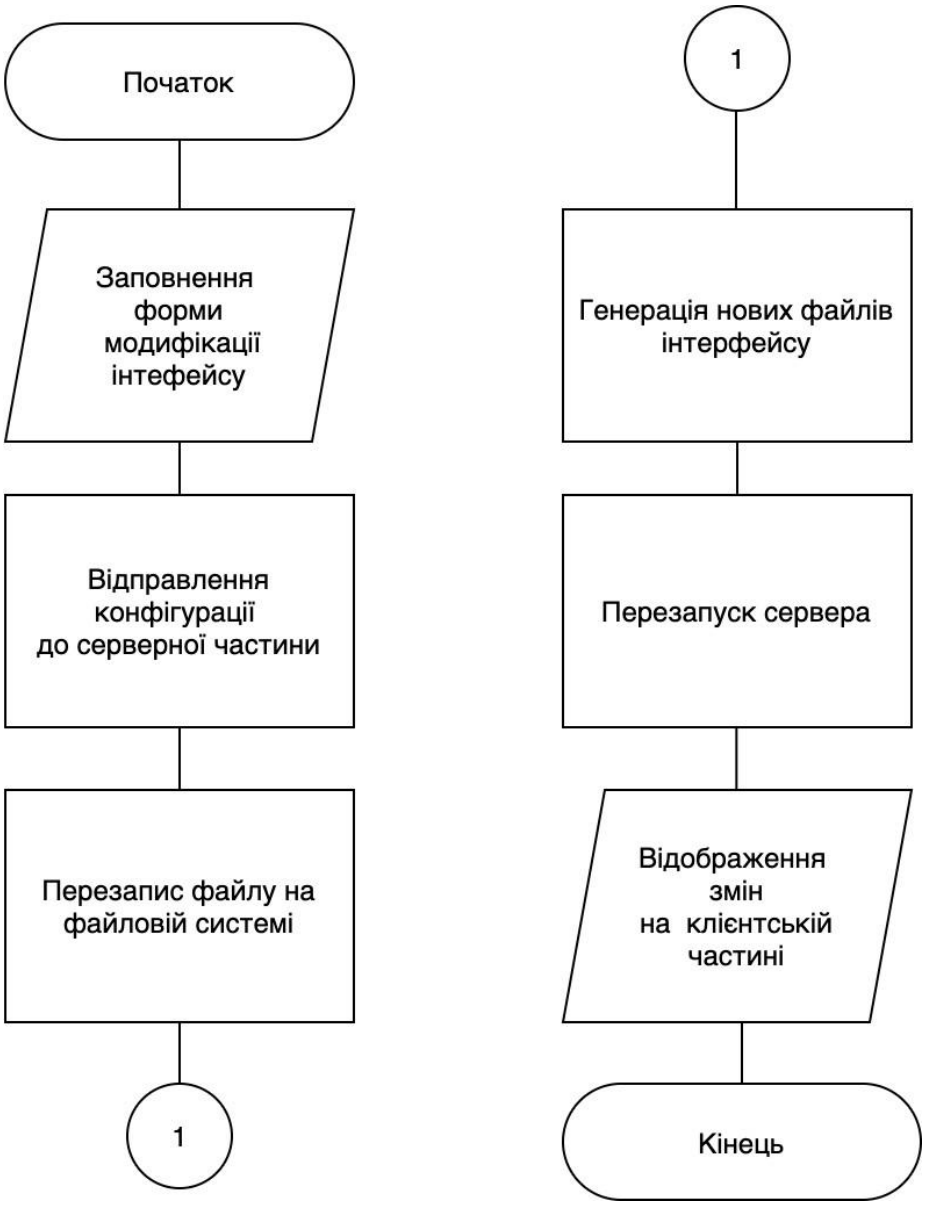
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Jagiellonian University – Homepage [Електронний ресурс]. — Режим доступу до ресурсу: https://en.uj.edu.pl.
2. Technical University of Munich – Homepage [Електронний ресурс]. — Режим доступу до ресурсу: https://www.tum.de.
3. НМУ ім. О.О. Богомольця – Головна [Електронний ресурс]. — Режим доступу до ресурсу: https://nmuofficial.com.
4. Python – Вікіпедія [Електронний ресурс]. — Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Python.
5. C# – Вікіпедія [Електронний ресурс]. — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/C_Sharp_(programming_language).
6. Java – Вікіпедія [Електронний ресурс]. — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Java_(programming_language).
7. JavaScript – Вікіпедія [Електронний ресурс]. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JavaScript.
8. Advantages And Disadvantages of RDBMS [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/0lo37y.
9. An Overview of SQL and NoSQL with it’s Pros and Cons [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/4mo37y.
10. MySQL [Електронний ресурс]. — Режим доступу до ресурсу: https://searchoracle.techtarget.com/definition/MySQL.
11. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/ptp37y.
12. Express.js – Вікіпедія [Електронний ресурс]. — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Express.js.

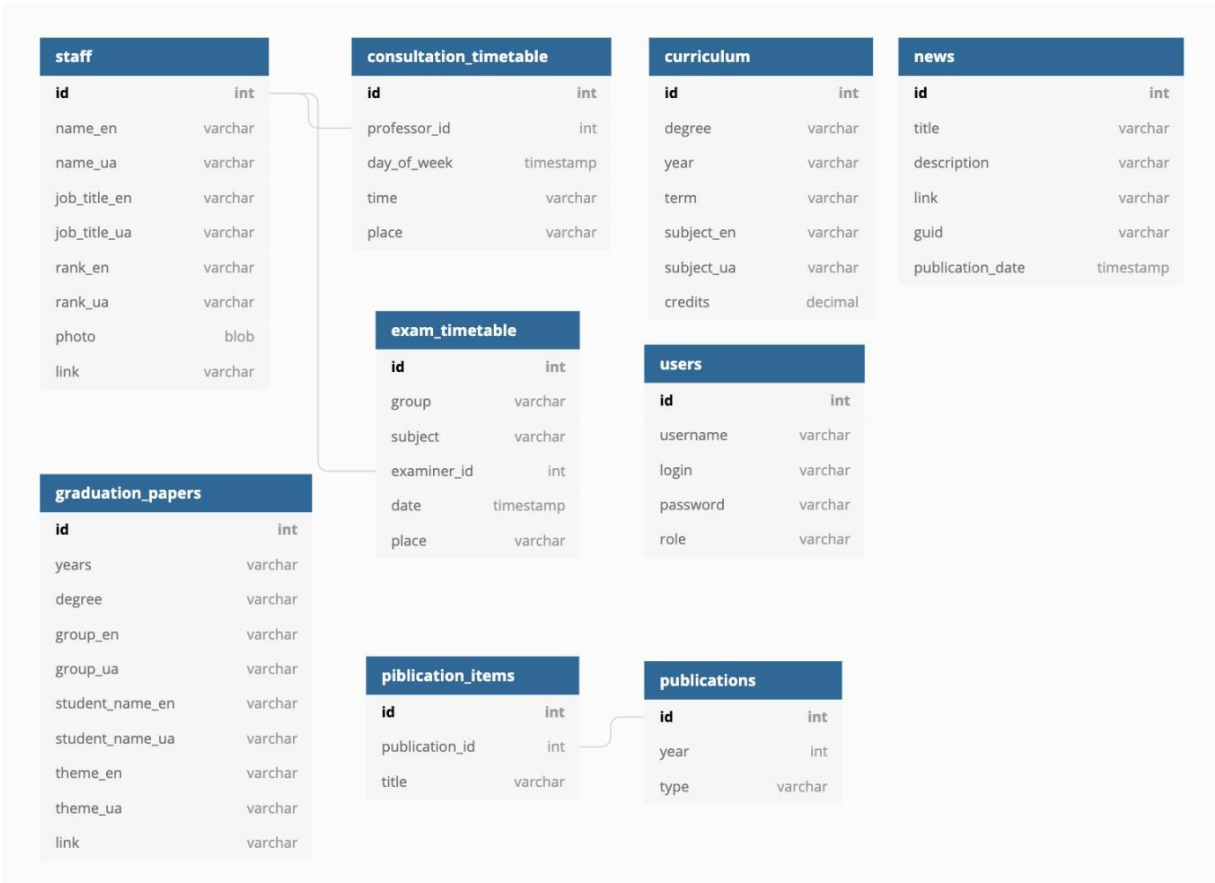
13. Meteor.js – Вікіпедія [Електронний ресурс]. — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Meteor_(web_framework).
14. Client Server Architecture [Електронний ресурс]. — Режим доступу до ресурсу: https://cio-wiki.org/wiki/Client_Server_Architecture.
15. Chris Northwood. The Full Stack Developer: Your Essential Guide to the Everyday Skills Expected of a Modern Full Stack Web Developer, 2018.
16. Hans-Jurgen Schonig. Mastering PostgreSQL 12: Advanced Techniques to Build and Administer Scalable and Reliable PostgreSQL Database Applications, 3rd Edition, 2019.
17. Ethan Brown. Web Development with Node and Express: Leveraging the JavaScript Stack, 2014.
18. Ben Hammersley. Developing feeds with RSS and Atom, 2005.

ДОДАТКИ

Додаток 1
Копії графічних матеріалів

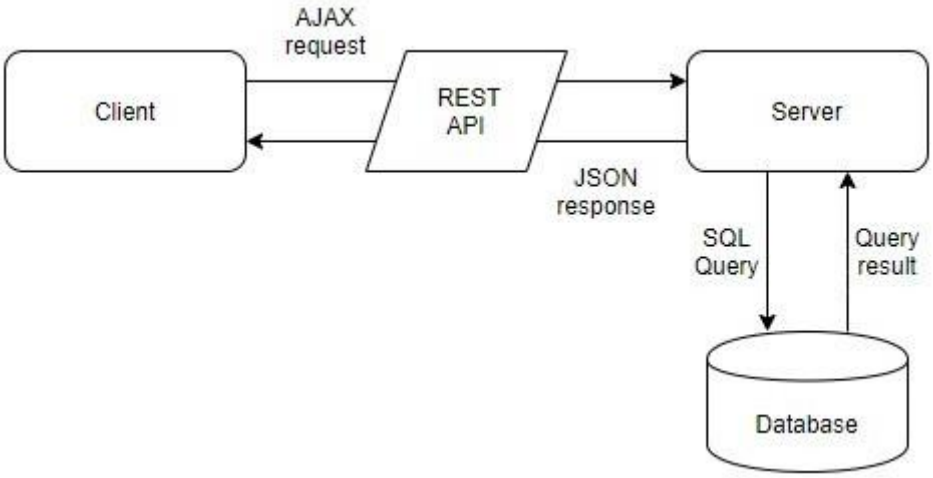


ДП.045440-06-99.
Веб-сайт кафедри програмного забезпечення комп'ютерних систем. Серверна частина. Алгоритм редагування інтерфейсу. Схема роботи алгоритму



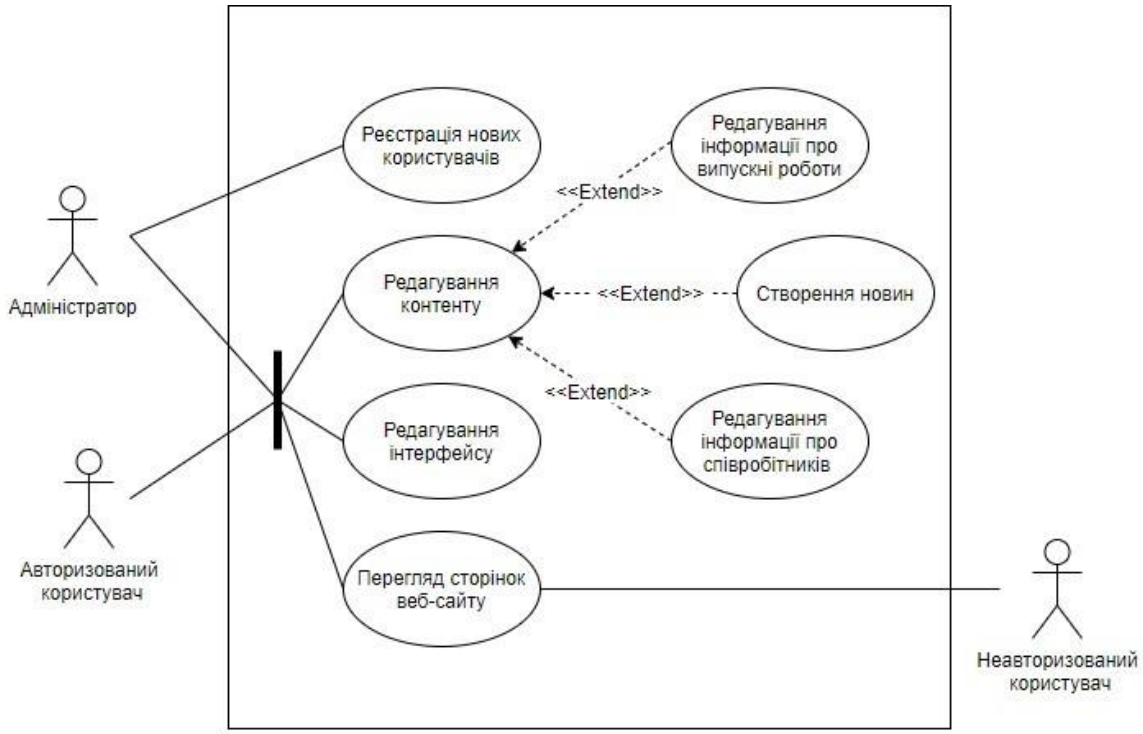
ДП.045440-07-99.
Веб-сайт кафедри програмного забезпечення комп'ютерних систем. Серверна частина. Структура бази даних. ERD-діаграма

Структурна схема системи



Півненко Олексій, група КП-61

Діаграма прецедентів



Півненко Олексій, група КП-61

Додаток 2
Лістинг програми

```
{
    "name": "pzks-website",
    "version": "1.0.0",
    "description": "FAM PZKS official website",
    "scripts": {
        "start:prod": "pm2 start npm --name \"server\" -- run
start:server",
        "start:dev": "webpack-dev-server --hot --open --mode development",
        "start:server": "node server.js",
        "build": "rm -rf dist && webpack --config
webpack.config.production.js ",
        "now-build": "npm run build",
        "test": "xo && stylelint './src/**/*.js'"
    },
    "main": "src/index.js",
    "repository": "https://github.com/xxczaki/styled-react-boilerplate",
    "author": "Egor Shchupakovskiy",
    "license": "MIT",
    "devDependencies": {
        "@babel/core": "^7.8.3",
        "@babel/preset-env": "^7.8.3",
        "@babel/preset-react": "^7.8.3",
        "@pieh/friendly-errors-webpack-plugin": "^1.7.0-chalk-2",
        "babel-eslint": "^10.0.3",
        "babel-loader": "^8.0.6",
        "babel-plugin-styled-components": "^1.10.6",
        "clean-css-loader": "^2.0.0",
        "css-loader": "^3.4.2",
        "eslint-config-xo-react": "^0.22.0",
        "eslint-plugin-react": "^7.18.0",
        "eslint-plugin-react-hooks": "^2.3.0",
        "extract-css-chunks-webpack-plugin": "^4.7.4",
        "file-loader": "^5.0.2",
        "html-webpack-plugin": "^3.2.0",
        "imagemin": "^7.0.1",
        "imagemin-mozjpeg": "^8.0.0",
        "imagemin-pngquant": "^8.0.0",
        "imagemin-svgo": "^7.0.0",
        "imagemin-webp": "^5.1.0",
        "img-loader": "^3.0.1",
        "json-loader": "^0.5.7",
        "script-ext-html-webpack-plugin": "^2.1.4",
        "style-loader": "^1.2.1",
        "stylelint": "^13.0.0",
        "stylelint-config-recommended": "^3.0.0",
        "stylelint-config-styled-components": "^0.1.1",
        "stylelint-processor-styled-components": "^1.9.0",
        "terser-webpack-plugin": "^2.3.2",
        "url-loader": "^3.0.0",
        "webpack": "^4.41.5",
        "webpack-cli": "^3.3.10",
        "webpack-dev-server": "^3.10.1",
        "webpack-nomodule-plugin": "^1.0.1",
        "webpack-pwa-manifest": "^4.1.1",
        "workbox-webpack-plugin": "^4.3.1",
        "xo": "^0.25.3"
    },
    "dependencies": {
        "@fortawesome/fontawesome-svg-core": "^1.2.28",
        "@fortawesome/free-solid-svg-icons": "^5.13.0",
        "@fortawesome/react-fontawesome": "^0.1.10",
        "@hot-loader/react-dom": "^16.11.0",
        "axios": "^0.19.2",
        "body-parser": "^1.19.0",
        "compression": "^1.7.4",
```

```
        "core-js": "^3.6.4",
        "express": "^4.17.1",
        "express-basic-auth": "^1.2.0",
        "express-history-api-fallback": "^2.2.1",
        "feed": "^4.2.0",
        "formik": "^2.1.4",
        "fs": "0.0.1-security",
        "history": "^4.10.1",
        "i18n": "^0.8.6",
        "i18next": "^19.4.1",
        "i18next-browser-languagedetector": "^4.0.2",
        "interweave": "^12.5.0",
        "lodash": "^4.17.15",
        "modern-normalize": "^0.6.0",
        "moment": "^2.26.0",
        "normalize.css": "^8.0.1",
        "pg": "^8.2.1",
        "react": "^16.13.1",
        "react-anchor-link-smooth-scroll": "^1.0.12",
        "react-dom": "^16.12.0",
        "react-ga": "^3.0.0",
        "react-hot-loader": "^4.12.19",
        "react-i18next": "^11.3.4",
        "react-json-view": "^1.19.1",
        "react-router-dom": "^5.1.2",
        "semantic-ui-css": "^2.4.1",
        "semantic-ui-react": "^0.88.2",
        "shelljs": "^0.8.4",
        "styled-components": "^5.0.0"
    },
    "xo": {
        "nodeVersion": ">=10",
        "parser": "babel-eslint",
        "envs": [
            "node",
            "browser"
        ],
        "extends": "xo-react",
        "settings": {
            "react": {
                "version": "16.12"
            }
        },
        "rules": {
            "import/no-unassigned-import": "off"
        }
    }
}
const Feed = require('feed').Feed;
var fs = require('fs');
var dirPath = __dirname + "/public/rss.xml";

const feedOptions = {
    title: "Кафедра програмного забезпечення комп'ютерних систем",
    link: "http://pzks.fpm.kpi.ua/",
    description : "",
    language: "UK",
};

var feed = new Feed(feedOptions);

const createNewFeed = () => {
    feed = new Feed(feedOptions);
}
```

```
const loadRss = () => {
    return fs.readFileSync(dirPath, 'utf8')
}

const addFeedItem = (item) => {
    feed.addItem(item);
}

const saveRss = () => {
  let rssdoc = feed.rss2();
  fs.writeFile(dirPath, rssdoc, function (err) {
    if (err) {
      throw err;
    }
  });
}


module.exports = {
    loadRss,
    saveRss,
    addFeedItem,
    createNewFeed
  }
const express = require("express");
const fs = require("fs");
const fallback = require("express-history-api-fallback");
const compression = require("compression");
const { exec } = require("child_process");
const basicAuth = require("express-basic-auth");
var bodyParser = require("body-parser");

const db = require("./queries");
const rss = require("./rss");

const app = express();
const port = process.env.PORT || 8080;

app.use(
  bodyParser.json({
    limit: "50mb",
  })
);

app.use(
  bodyParser.urlencoded({
    limit: "50mb",
    parameterLimit: 100000,
    extended: true,
  })
);

app.use(
  "/admin",
  basicAuth({
    challenge: true,
    authorizer,
    unauthorizedResponse: (req)  =>  {
      return `unauthorized. ip: ${req.ip}`;
    },
  })
);
function authorizer(username, password) {
  const userMatches = basicAuth.safeCompare(username, "admin");
  const passwordMatches = basicAuth.safeCompare(password, "admin");
```

```
  return userMatches & passwordMatches;
}

app.get("/api/news", db.getNews);
app.get("/api/news/:id", db.getNewsById);
app.post("/api/news", db.createNews);
app.put("/api/news/:id", db.updateNews);
app.delete("/api/news/:id", db.deleteNews);

app.get("/rss", (req, res) => {
  rssFeed = rss.loadRss();
  res.type("rss");
  res.send(rssFeed);
});

app.put("/api/admin", (req, res) => {
  const { content, type } = req.body;
  const data = JSON.stringify(content, null, 2);
  switch (type) {
    case "UA":
      fs.writeFileSync(__dirname + "/src/i18n/locales/ua.json", data);
      break;
    case "EN":
      fs.writeFileSync(__dirname + "/src/i18n/locales/en.json", data);
      break;
    case "curriculum":
      fs.writeFileSync(
        __dirname + "/src/Project/CurriculumInfo/curriculumInfoData.json",
        data
      );
      break;
    case "exam":
      fs.writeFileSync(
        __dirname +
"/src/Project/EducationalProcess/Schedules/exam/exams.json",
        data
      );
      break;
    case "consultation":
      fs.writeFileSync(
        __dirname +

"/src/Project/EducationalProcess/Schedules/consultation/consultation.json",
        data
      );
      break;
    case "graduationPaper":
      fs.writeFileSync(
        __dirname +

"/src/Project/EducationalProcess/GraduationPaper/graduationPapers.json",
        data
      );
      break;
    default:
      return;
  }
  exec("./restart_server.sh");
});

app.get("/api/isAdmin", (req, res) => {
  if (req.headers.authorization) {
    res.json({ isAdmin: true });
  } else {
```

```
    res.json({ isAdmin: false });
  }
});

app.use(compression());

app.use(express.static(`${__dirname}/dist`));

app.use(fallback(`${__dirname}/dist/index.html`));

app.listen(port, () => {
  console.log(`Server listening on port ${port}`);
});
      var t = e.active,
        n = e.children,
        a = e.className,
        o = e.collapsing,
        i = e.content,
        l = e.disabled,
        u = e.error,
        g = e.icon,
        h = e.negative,
        A =
        e.positive,M =
        e.selectable,v =
        e.singleLine,I =
        e.textAlign,
        N = e.verticalAlign,
        C = e.warning,
        j = e.width,
        k = s()(
          Object(d.a)(t, "active"),
          Object(d.a)(o, "collapsing"),
          Object(d.a)(l, "disabled"),
          Object(d.a)(u, "error"),
          Object(d.a)(h, "negative"),
          Object(d.a)(A, "positive"),
          Object(d.a)(M, "selectable"),
          Object(d.a)(v, "single line"),
          Object(d.a)(C, "warning"),
          Object(d.d)(I),
          Object(d.f)(N),
          Object(d.g)(j, "wide"),
          a
        ),
        _ = Object(p.a)(b, e),
        L = Object(f.a)(b, e);
      return m.a.isNil(n)
        ? c.a.createElement(L, r()({}, _, { className: k }), y.a.create(g),
i)
        : c.a.createElement(L, r()({}, _, { className: k }), n);
    }
    (b.handledProps = [
      "active",
      "as",
      "children",
      "className",
      "collapsing",
      "content",
      "disabled",
      "error",
      "icon",
      "negative",
      "positive",
      "selectable",
      "singleLine",
```

```
      "textAlign",
      "verticalAlign",
      "warning",
      "width",
    ]),
      (b.defaultProps = { as: "td" }),
      (b.propTypes = {}),
      (b.create = Object(A.e)(b, function(e) {
        return { content: e };
      }));
    var M = b;
    function v(e) {
      var t = e.children,
        n = e.className,
        a = e.content,
        o = e.fullWidth,
        i = s()(Object(d.a)(o, "full-width"), n),
        l = Object(p.a)(v, e),
        u = Object(f.a)(v, e);
      return c.a.createElement(
        u,
        r()({}, l, { className: i }),
        m.a.isNil(t) ? a : t
      );
    }
    (v.handledProps = ["as", "children", "className", "content",
"fullWidth"]),
      (v.defaultProps = { as: "thead" }),
      (v.propTypes = {});
    var I = v;
    function N(e) {
      var t = e.as,
        n = Object(p.a)(N, e);
      return c.a.createElement(I, r()({}, n, { as: t }));
    }
    (N.handledProps = ["as"]),
      (N.propTypes = {}),
      (N.defaultProps = { as: "tfoot" });
    var C = N;
    function j(e) {
      var t = e.as,
        n = e.className,
        a = e.sorted,
        o = s()(Object(d.e)(a, "sorted"), n),
        i = Object(p.a)(j, e);
      return c.a.createElement(M, r()({}, i, { as: t, className: o }));
    }
    (j.handledProps = ["as", "className", "sorted"]),
      (j.propTypes = {}),
      (j.defaultProps = { as: "th" });
    var k = j;
    function  _(e)  {
      var t = e.active,
        n =
        e.cellAs,a =
        e.cells,
        o = e.children,
        l = e.className,
        u = e.disabled,
        g = e.error,
        h = e.negative,
        A = e.positive,
        y =
        e.textAlign,
        b = e.verticalAlign,
        v = e.warning,
```

```
I = s()(
```

```
      Object(d.a)(t, "active"),
      Object(d.a)(u, "disabled"),
      Object(d.a)(g, "error"),
      Object(d.a)(h, "negative"),
      Object(d.a)(A, "positive"),
      Object(d.a)(v, "warning"),
      Object(d.d)(y),
      Object(d.f)(b),
      l
    ),
    N = Object(p.a)(_, e),
    C = Object(f.a)(_, e);
  return m.a.isNil(o)
    ? c.a.createElement(
        C,
        r()({}, N, { className: I }),
        i()(a, function(e) {
          return M.create(e, { defaultProps: { as: n } });
        })
      )
    : c.a.createElement(C, r()({}, N, { className: I }), o);
}
(_.handledProps = [
  "active",
  "as",
  "cellAs",
  "cells",
  "children",
  "className",
  "disabled",
  "error",
  "negative",
  "positive",
  "textAlign",
  "verticalAlign",
  "warning",
]),
  (_.defaultProps = { as: "tr", cellAs: "td" }),
  (_.propTypes = {}),
  (_.create = Object(A.e)(_, function(e) {
    return { cells: e };
  }));
var L = _;
function T(e) {
  var t = e.attached,
    n = e.basic,
    a  =  e.celled,
    o = e.children,
    l = e.className,
    u = e.collapsing,
    g = e.color,
    A =
    e.columns,y =
    e.compact,
    b = e.definition,
    M = e.fixed,
    v = e.footerRow,
    N = e.headerRow,
    j = e.headerRows,
    k = e.inverted,
      = e.padded,
    E = e.renderBodyRow,
    D = e.selectable,
    U = e.singleLine,
    w = e.size,
    S = e.sortable,
```

```
    x = e.stackable,
    O = e.striped,
    Y = e.structured,
    z  =  e.tableData,
    F  =
    e.textAlign,Q =
    e.unstackable,
    P = e.verticalAlign,
    V = s()(
      "ui",
      g,
      w,
      Object(d.a)(a, "celled"),
      Object(d.a)(u, "collapsing"),
      Object(d.a)(b, "definition"),
      Object(d.a)(M, "fixed"),
      Object(d.a)(k, "inverted"),
      Object(d.a)(D, "selectable"),
      Object(d.a)(U, "single line"),
      Object(d.a)(S, "sortable"),
      Object(d.a)(x, "stackable"),
      Object(d.a)(O, "striped"),
      Object(d.a)(Y, "structured"),
      Object(d.a)(Q, "unstackable"),
      Object(d.b)(t, "attached"),
      Object(d.b)(n, "basic"),
      Object(d.b)(y, "compact"),
      Object(d.b)(_, "padded"),
      Object(d.d)(F),
      Object(d.f)(P),
      Object(d.g)(A,
      "column"),"table",
      l
    ),
    H = Object(p.a)(T, e),
    R = Object(f.a)(T, e);
  if (!m.a.isNil(o))
    return c.a.createElement(R, r()({}, H, { className: V }), o);
  var W = { defaultProps: { cellAs: "th" } },
    B =
      (N || j) &&
      c.a.createElement(
        I,
        null,
        L.create(N, W),
        i()(j, function(e) {
          return L.create(e, W);
        })
      );
  return c.a.createElement(
    R,
    r()({}, H, { className: V }),
    B,
    c.a.createElement(
      h,
      null,
      E &&
        i()(z, function(e, t) {
          return L.create(E(e, t));
        })
    ),
    v && c.a.createElement(C, null, L.create(v))
  );
}
(T.handledProps = [
  "as",
```

```
    "attached",
    "basic",
    "celled",
    "children",
    "className",
    "collapsing",
    "color",
    "columns",
    "compact",
    "definition",
    "fixed",
    "footerRow",
    "headerRow",
    "headerRows",
    "inverted",
    "padded",
    "renderBodyRow",
    "selectable",
    "singleLine",
    "size",
    "sortable",
    "stackable",
    "striped",
    "structured",
    "tableData",
    "textAlign",
    "unstackable",
    "verticalAlign",
  ]),
    (T.defaultProps = { as: "table" }),
    (T.propTypes = {}),
    (T.Body = h),
    (T.Cell = M),
    (T.Footer = C),
    (T.Header = I),
    (T.HeaderCell = k),
    (T.Row = L);
  t.a = T;
},
function(e, t, n) {
  "use strict";
  var a = n(2),
    r = n.n(a),
    o = n(15),
    i = n.n(o),
    l = n(16),
    s = n.n(l),
    u = n(19),
    c = n.n(u),
    d = n(17),
    p = n.n(d),
    f = n(5),
    m = n.n(f),
    g = n(20),
    h = n.n(g),
    A = n(1),
    y = n.n(A),
    b = n(29),
    M = n.n(b),
    v = n(11),
    I = n.n(v),
    N = n(7),
    C = n.n(N),
    j = (n(10), n(0)),
    k = n.n(j),
```

```
  _ = n(42),
  L = n(152),
  T = n(153),
  E = n(6),
  D = n(238);
function U(e) {
  var t = e.children,
    n = e.className,
    a = e.content,
    o = C()(n, "description"),
    i = Object(L.a)(U, e),
    l = Object(T.a)(U, e);
  return k.a.createElement(
    l,
    r()({}, i, { className: o }),
    E.a.isNil(t) ? a : t
  );
}
(U.handledProps = ["as", "children", "className", "content"]),
  (U.propTypes = {}),
  (U.create = Object(D.e)(U, function(e) {
    return { content: e };
  }));
var w = U;
function S(e) {
  var t = e.children,
    n = e.className,
    a = e.content,
    o = C()("header", n),
    i = Object(L.a)(S, e),
    l = Object(T.a)(S, e);
  return k.a.createElement(
    l,
    r()({}, i, { className: o }),
    E.a.isNil(t) ? a : t
  );
}
(S.handledProps = ["as", "children", "className", "content"]),
  (S.propTypes = {}),
  (S.create = Object(D.e)(S, function(e) {
    return { content: e };
  }));
var x = S;
function O(e) {
  var t = e.children,
    n = e.className,
    a = e.content,
    o = e.description,
    i = e.floated,
    l = e.header,
    s = e.verticalAlign,
    u = C()(Object(_.e)(i, "floated"), Object(_.f)(s), "content", n),
    c = Object(L.a)(O, e),
    d = Object(T.a)(O, e);
  return E.a.isNil(t)
    ? k.a.createElement(
        d,
        r()({}, c, { className: u }),
        x.create(l),
        w.create(o),
        a
      )
    : k.a.createElement(d, r()({}, c, { className: u }), t);
}
(O.handledProps = [
```

```
      "as",
      "children",
      "className",
      "content",
      "description",
      "floated",
      "header",
      "verticalAlign",
    ]),
      (O.propTypes = {}),
      (O.create = Object(D.e)(O, function(e) {
        return { content: e };
      }));
    var Y = O,
      z = n(34);
    function F(e) {
      var t = e.className,
        n = e.verticalAlign,
        a = C()(Object(_.f)(n), t),
        o = Object(L.a)(F, e);
      return k.a.createElement(z.a, r()({}, o, { className: a }));
    }
    (F.handledProps = ["className", "verticalAlign"]),
      (F.propTypes = {}),
      (F.create = Object(D.e)(F, function(e) {
        return { name: e };
      }));
    var Q = F,
      P = n(150),
      V = n.n(P),
      H = n(156),
      R = (function(e) {
        function t() {
          var e, n;
          i()(this, t);
          for (var a = arguments.length, r = new Array(a), o = 0; o < a;
o++)
            r[o] = arguments[o];
          return (
            (n = c()(this, (e = p()(t)).call.apply(e, [this].concat(r)))),
            y()(m()(n), "handleClick", function(e) {
              n.props.disabled || I()(n.props, "onClick", e, n.props);
            }),
            n
          );
        }
        return (
          h()(t, e),
          s()(t, [
            {
              key: "render",
              value: function() {
                var e = this.props,
                  n = e.active,
                  a = e.children,
                  o = e.className,
                  i = e.content,
                  l = e.description,
                  s = e.disabled,
                  u = e.header,
                  c = e.icon,
                  d = e.image,
                  p = e.value,
                  f = Object(T.a)(t, this.props),
                  m = C()(
```

```
    Object(_.a)(n, "active"),
    Object(_.a)(s, "disabled"),
    Object(_.a)("li" !== f, "item"),
    o
  ),
  g = Object(L.a)(t, this.props),
  h = "li" === f ? { value: p } : { "data-value": p };
if (!E.a.isNil(a))
  return k.a.createElement(
    f,
    r()(
      {},
      h,
      {
        role: "listitem",
        className: m,
        onClick: this.handleClick,
      },
      g
    ),
    a
  );
var A = Q.create(c, { autoGenerateKey: !1 }),
  y = H.a.create(d, { autoGenerateKey: !1 });
if (!Object(j.isValidElement)(i) && V()(i))
  return k.a.createElement(
    f,
    r()(
      {},
      h,
      {
        role: "listitem",
        className: m,
        onClick: this.handleClick,
      },
      g
    ),
    A || y,
    Y.create(i, {
      autoGenerateKey: !1,
      defaultProps: { header: u, description: l },
    })
  );
var b = x.create(u, { autoGenerateKey: !1 }),M
  = w.create(l, { autoGenerateKey: !1 });
return A || y
  ? k.a.createElement(
      f,
      r()(
        {},
        h,
        {
          role: "listitem",
          className: m,
          onClick: this.handleClick,
        },
        g
      ),
      A || y,
      (i || b || M) && k.a.createElement(Y, null, b, M, i)
    )
  : k.a.createElement(
      f,
      r()(
        {},
```

```
                        h,
                        {
                          role: "listitem",
                          className: m,
                          onClick: this.handleClick,
                        },
                        g
                      ),
                      b,
                      M,
                      i
                    );
                },
              },
            ]),
            t
          );
        })(j.Component);
      y()(R, "handledProps", [
        "active",
        "as",
        "children",
        "className",
        "content",
        "description",
        "disabled",
        "header",
        "icon",
        "image",
        "onClick",
        "value",
      ]),
        (R.propTypes = {}),
        (R.create = Object(D.e)(R, function(e) {
          return { content: e };
        }));
      var W = R;
      function B(e) {
        var t = e.children,
          n = e.className,
          a = e.content,
          o = Object(L.a)(B, e),
          i = Object(T.a)(B, e),
          l = C()(Object(_.a)("ul" !== i && "ol" !== i, "list"), n);
        return k.a.createElement(
          i,
          r()({}, o, { className: l }),
          E.a.isNil(t) ? a : t
        );
      }
      (B.handledProps = ["as", "children", "className", "content"]),
        (B.propTypes = {});
      var Z = B,
        G = (function(e) {
          function t() {
            var e, n;
            i()(this, t);
            for (var a = arguments.length, r = new Array(a), o = 0; o < a;
o++)
              r[o] = arguments[o];
            return (
              (n = c()(this, (e = p()(t)).call.apply(e, [this].concat(r)))),
              y()(m()(n), "handleItemOverrides", function(e) {
                return {
```

Додаток 3
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ВЕБ-САЙТ КАФЕДРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ. СЕРВЕРНА ЧАСТИНА

Виконав: Півненко Олексій Володимирович

Керівник: старший викладач кафедри ПЗКС, к.т.н., Люшенко Л.А.

Київ – 2020

ПОСТАНОВКА ЗАДАЧІ

Мета проєкту: розробити серверну частину веб-сайту кафедри програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Завдання:

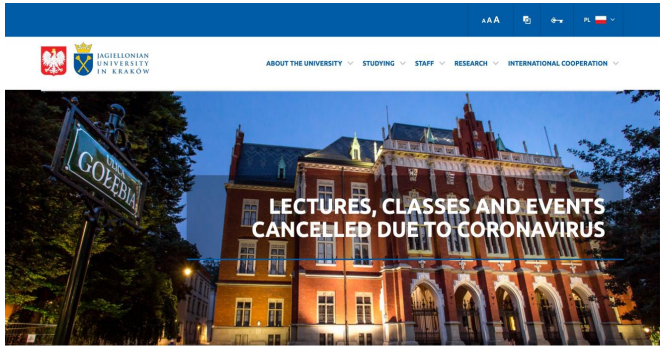
1. Проаналізувати нормативні документи, що регламентують роботу сайтів підрозділів університету, а також існуючі програмні аналоги та визначити вимоги до розроблюваного веб-сайту.
2. Розробити серверну частину веб-сайту.
3. Налаштувати взаємодію з клієнтською частиною.
4. Протестувати роботу системи.

АКТУАЛЬНІСТЬ

Поточна версія веб-сайту містить наступні недоліки:

- структура веб-сайту не відповідає вимогам описаним у положенні про типовий сайт кафедри НТУУ «КПІ» у повному обсязі;
- застарілий дизайн;
- використання незахищеного протоколу передачі даних;
- відсутність стрічки новин RSS.

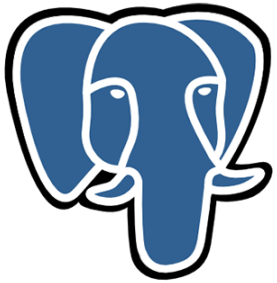
РОЗГЛЯНУТІ АНАЛОГИ



ВИМОГИ

- структура веб-сайту має бути розроблена згідно положення про типовий сайт кафедри НТУУ «КПІ»;
- забезпечення безперервного доступу користувачів;
- висока швидкість роботи;
- використання HTTPS;
- забезпечення можливості редагування інформації;
- коректна обробка будь-яких запитів користувача;
- наявність працюючого RSS.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



PostgreSQL

СХЕМА АРХІТЕКТУРИ СИСТЕМИ

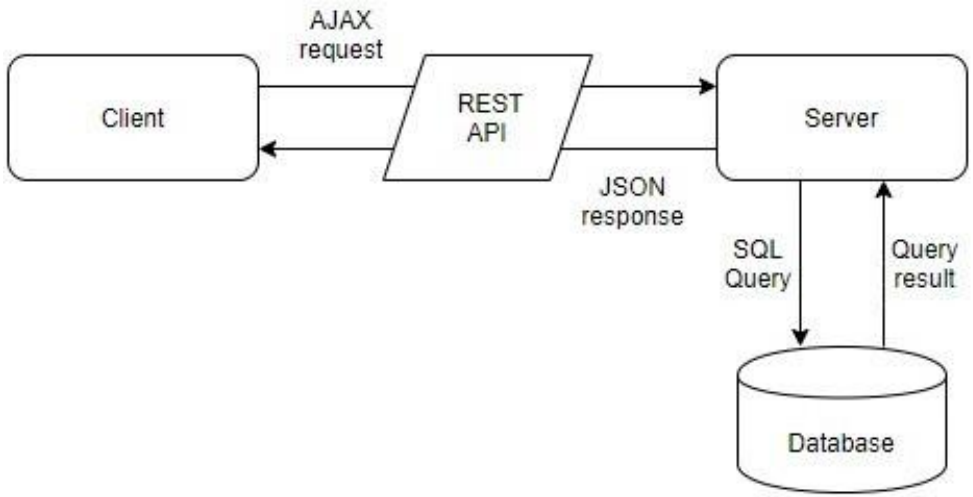
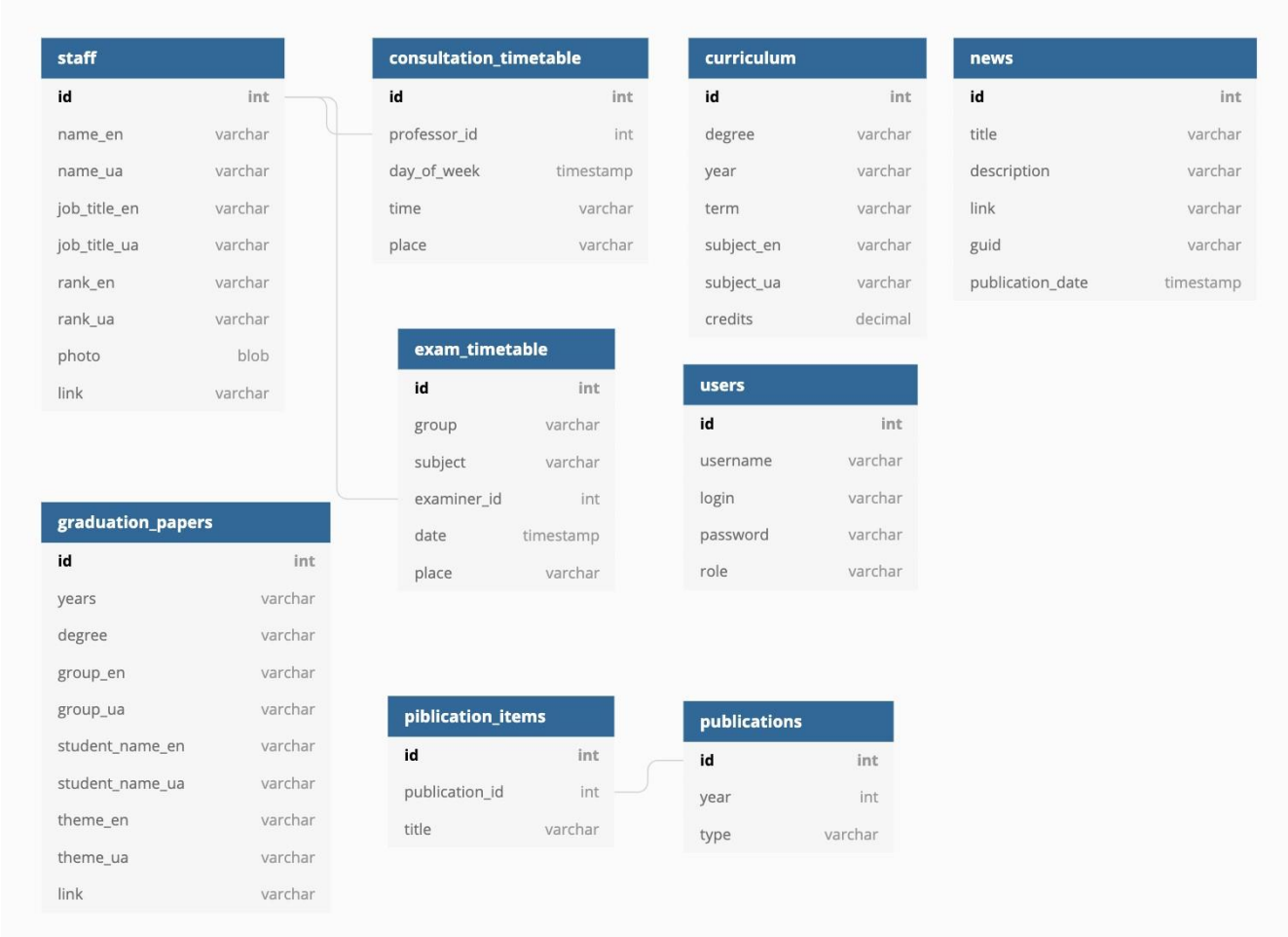
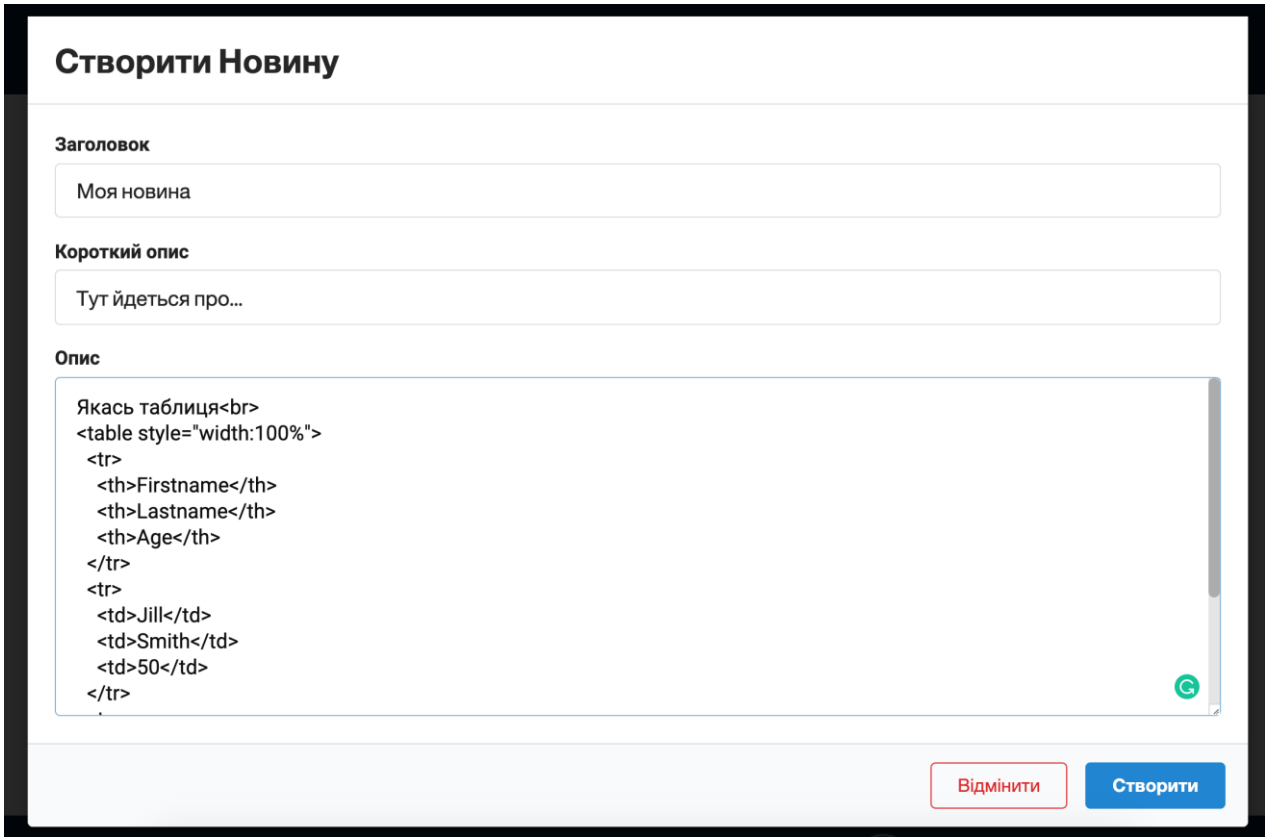


СХЕМА БАЗИ ДАНИХ



АЛГОРИТМ РЕДАГУВАННЯ ІНТЕРФЕЙСУ

СТВОРЕННЯ НОВИНИ



СТВОРЕННЯ НОВИНИ

Моя новина
17-06-2020
Тут йдеться про...

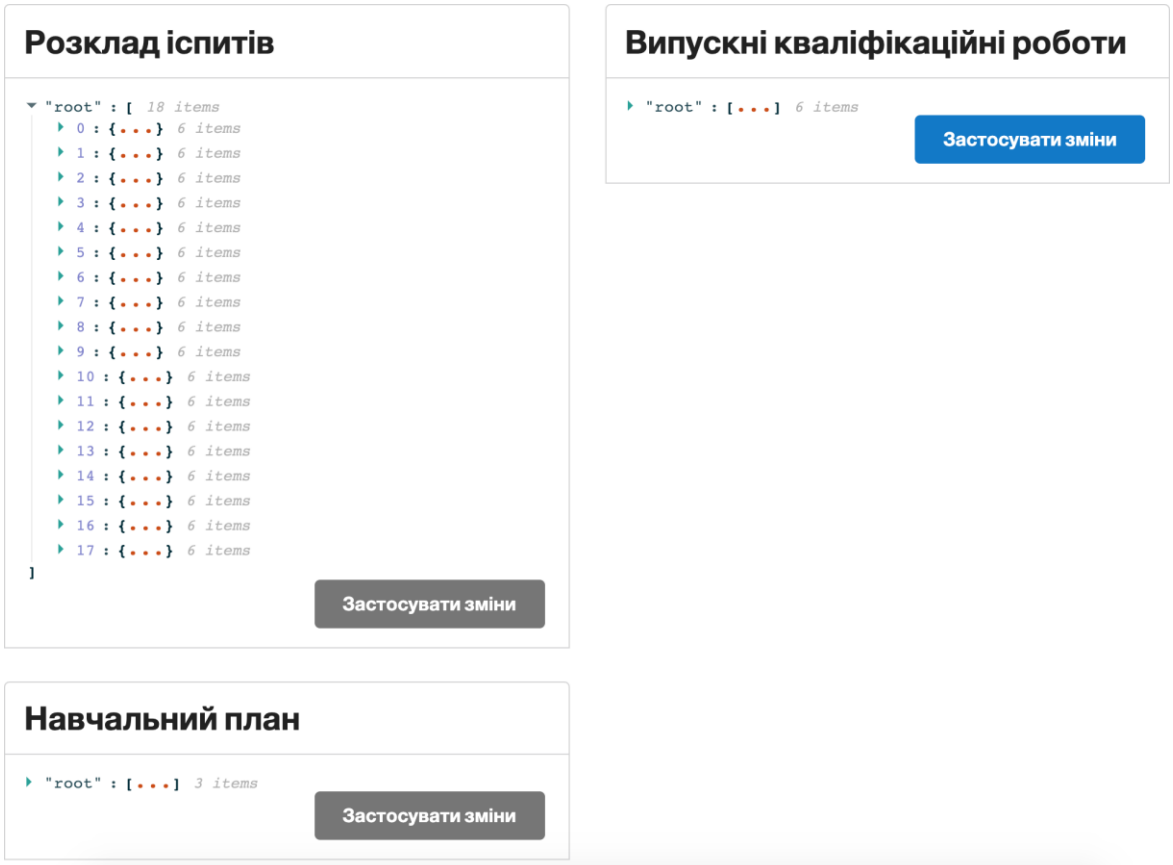
Детальніше

Моя новина

Якась таблиця

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94

ПАНЕЛЬ АДМІНІСТРАТОРА

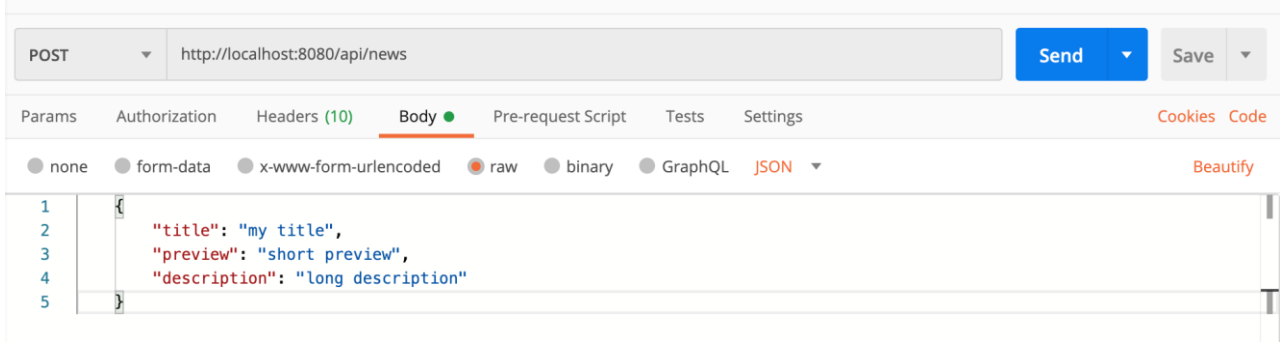


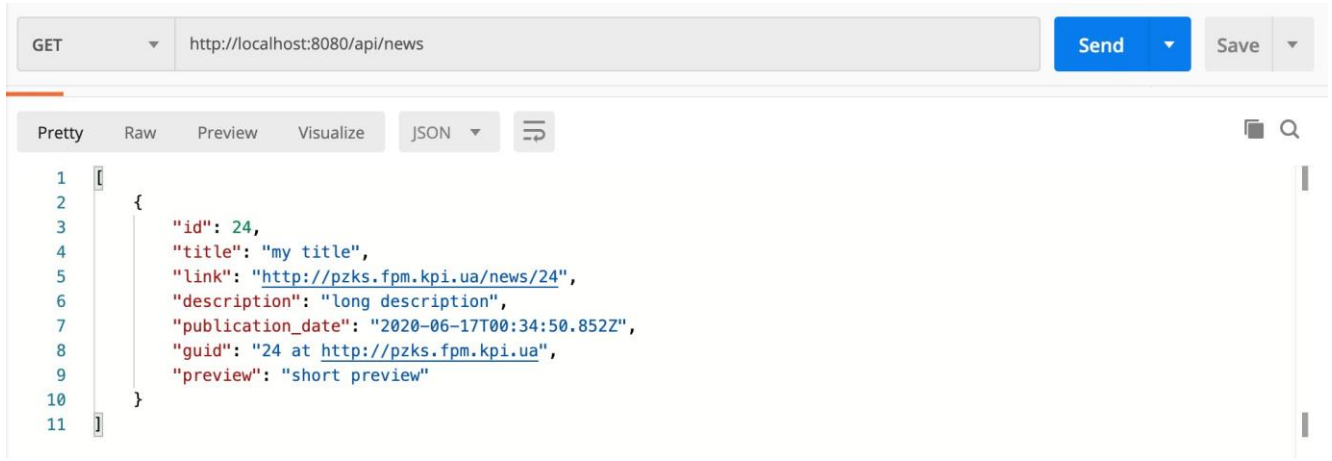
СТРІЧКА НОВИН RSS

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
    <channel>
        <title>Кафедра програмного забезпечення комп'ютерних систем</title>
        <link>http://pzks.fpm.kpi.ua/</link>
        <description></description>
        <lastBuildDate>Tue, 16 Jun 2020 23:59:46 GMT</lastBuildDate>
        <docs>https://validator.w3.org/feed/docs/rss2.html</docs>
        <generator>https://github.com/jpmonette/feed</generator>
        <language>UK</language>
        <item>
            <title><![CDATA[Моя новина]]></title>
            <guid>23</guid>
            <description><![CDATA[Якась таблиця<br>
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>]]></description>
        </item>
    </channel>
</rss>
```

ТЕСТУВАННЯ ВЕБ-САЙТУ

Приклад тестування утилітою Postman:





РЕКОМЕНДАЦІЇ ДЛЯ ПОКРАЩЕННЯ

- впровадження системи захисту від DoS/DDoS-атак;
- реалізація шардингу та реплікаціїї бази даних;
- реалізація модулю статистики;
- створення архіву матеріалів;
- реалізація форми зворотного зв’язку;
- впровадження особистого кабінету користувача.

ВИСНОВКИ

1. Проведено збір вимог до системи
2. Створено архітектуру серверної частини системи
3. Створено структуру бази даних веб-сайту
4. Реалізовано модуль взаємодії з БД
5. Реалізовано модуль REST API
6. Налаштовано взаємодію серверної та клієнтської частин
7. Проведено тестування веб-сайту
8. Розроблено документацію проєкту

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___»_____2019 р.

ВЕБ-САЙТ КАФЕДРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ. СЕРВЕРНА ЧАСТИНА

Програма та методика тестування

ДП.045440-04-51

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олексій ПІВНЕНКО

2019

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-сайт кафедри програмного забезпечення комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» розроблений згідно з архітектурним шаблоном SPA за допомогою мови JavaScript. Серверна частина реалізована за допомогою фреймворку Express.js.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Працездатність елементів сторінок веб-застосунку.
2. Коректність взаємодії з базою даних.
3. Стабільність роботи.
4. Забезпечення коректної обробки REST запитів.
5. Відповідність вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом White Box Testing. Таке тестування передбачає випадок, коли внутрішня структура застосунку відома. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1. Тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування).
2. Тестування продуктивності програмного забезпечення, зокрема Performance testing (тестування стабільності).
3. Тестування граничних значень.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Для тестування застосовується утиліта Postman консольний інтерфейс взаємодії з БД psql та найбільш популярні веб-браузери.

Працездатність веб-додатку перевіряється шляхом:

1. Динамічного ручного тестування – введенням граничних та недопустимих значень в поля запитів.
2. Динамічного ручного тестування на відповідність функціональним вимогам.
3. Статичного тестування коду.
4. Тестування веб-сайту в різних веб-браузерах.
5. Тестування при максимальному навантаженні.
6. Тестування стабільності роботи при різних умовах.
7. Тестування правильності запитів до бази даних.
8. Тестування REST API.
9. Тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___»_____2020 р.

ВЕБ-САЙТ КАФЕДРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ. СЕРВЕРНА ЧАСТИНА

Керівництво адміністратора

ДП.045440-05-34

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олексій ПІВНЕНКО

2020

ЗМІСТ

1. Процедура створення нової сторінки

Для створення нової сторінки потрібно відкрити проєкт у текстовому редакторі Після цього у дректорії «pzks_website/src/Project/» створити піддиректорію з назвою для майбутньої сторінки та скопіювати файл «pzks_website/src/Project/NewPageTemplate/index.js» у цю директорію. Далі потрібно змінити назву об’єктів виділених червоним (рис. 1) на назву новоствореної директорії. Далі всередині тегу «Semantic Container» потрібно створити розмітку нової сторінки. Для створення розмітки потрібно прибрати або модифікувати наявні елементи або додати нові в залежності від потреб. Для створення нових елементів потрібно використовувати HTML розмітку.

Після закінчення редагування файлу сторінки потрібно додати в інтерфейс посилання на неї. Для цього у файлі «pzks_website/src/Project/index.js» треба:

- додати імпорт нового об’єкту (рис. 2), де замість NewPageTemplate потрібно вставити назву новоствореного об’єкту а у лапках прописати шлях відносний шлях до файлу сторінки.
- додати новий тег Route, якому у якості атрибута path потрібно прописати частину URI яка буде використовуватися для посилання на сторінку, та замінити NewPageTemplate в атрибуті render на ім’я об’єкту (рис. 3).

Рис. 1. Створення об’єкту нової сторінки

Рис. 2. Імпорт об’єкту з іншого файлу

Рис. 3. Створення посилання на сторінку

Цих змін достатньо для того щоб після перекомпіляції проєкту нова сторінка була доступна за посиланням, що було вказано у атрибуті path, доданим до доменного імені веб-сайту.

Для того щоб додати посилання на вибрану сторінку до розділу меню потрібно зробити наступне:

1. Якщо пункт меню існує – перейти до наступного кроку, якщо ні – створити необхідний пункт меню (процедура створення пункту меню описана далі у цьому розділі).
2. знайти необхідний елемент меню у файлі «pzks_website/src/Project/Layout/Header/header.static.data.js» та додати інформацію про нове посилання до subMenuItems даного елементу (рис. 4), де:
   - title – текст посилання українською мовою;
   - titleEn – текст посилання англійською мовою;
   - path – посилання на сторінку без вказання доменного імені.

```
{
  id: 6,
  title: "Міжнародна діяльність",
  titleEn: "International Collaborations",
  subMenuItems: [
    {
      title: "Міжнародна діяльність",
      titleEn: "International Collaborations",
      path: "/international-collaborations",
    },
    {
      title: "Проект MEDIS",
      titleEn: "Project MEDIS",
      path: "/project-medis",
    },
    {
      title: "Проект PARIS",
      titleEn: "Project PARIS",
      path: "/project-paris",
    },
    {
      title: "Назва посилання українською",
      titleEn: "Liink title English",
      path: "/new-page",
    },
  ],
},
```

Рис. 4. Створення нового посилання у розділі меню

Для створення нового розділу меню потрібно у файлі «pzks_website/src/Project/Layout/Header/header.static.data.js» додати новий елемент за зразком (рис.5), де :

- id – ідентифікатор, поставити значення більше на 1 за значення у попередньому елементі;
- title – назва розділу українською мовою;
- titleEn – назва розділу англійською мовою;
- subMenuItems – посилання на сторінки, що належать до даного розділу меню

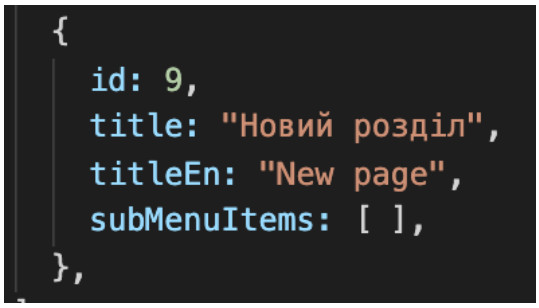


Рис. 5. Створення нового розділу меню

2. Процедура редагування контенту

Після проходження процедури авторизації, користувачу надається доступ до панелі адміністратора (рис. 6). Ця панель дозволяє редагувати присутню на сайті інформацію та інтерфейс веб-сайту. Дані представлені для редагування у форматі JSON. Принцип редагування однаковий для кожного розділу. Для запобігання загромадження інтерфейсу інформація у кожному розділі представлена у згорнутому вигляді. Для того щоб згорнути або розгорнути елемент використовують стрілки розміщені зліва від елементу (рис. 7).

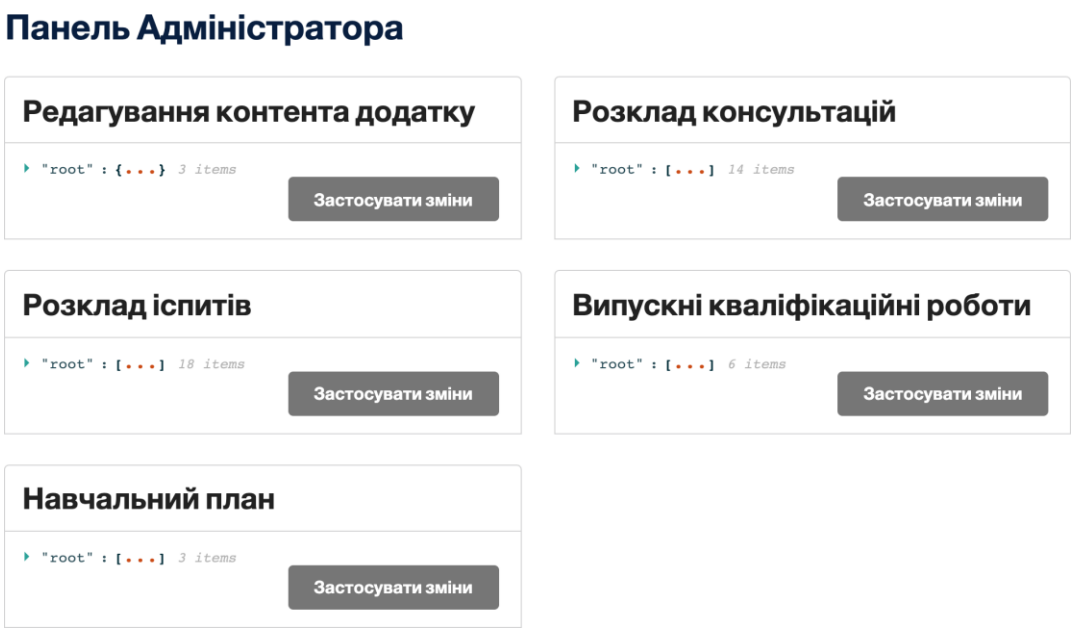


Рис. 6. Панель адміністратора

Рис. 7. Елементи згортання та розгортання блоків інформації

При наведенні курсору миші на поле об’єкту справа від нього з’являється дві кнопки (рис. 8). Перша з них (лівіша) дозволяє скопіювати значення поля до буферу обміну. Друга (правіша) дозволяє редагувати значення цього поля. Після натискання на кнопку редагування дані відображаються у текстовому полі і їх можна змінювати як звичайний текст.

Справа від поля з’являються дві кнопки (рис. 9):

- червоний хрестик – скасовує внесені зміни;
- зелена галка – підтверджує внесені зміни.

Рис. 8. Кнопки взаємодії з полем об’єкту

Рис. 9. Зміна значення поля

Після внесення змін кнопка «Застосувати зміни» стає активною, змінюючи свій колір з синього на сірий. Її натискання дозволяє зберегти внесені зміни у системі (рис. 10). Для застосування змін потрібно почекати деякий час, що приблизно дорівнює одній хвилині, доки проєкт перекомпілюється на сервері.

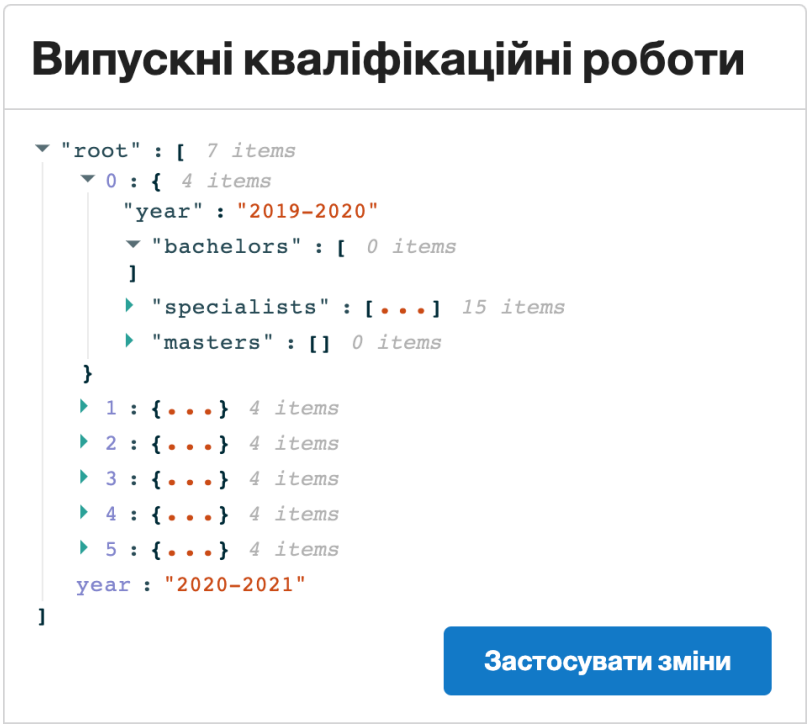


Рис. 10. Кнопка «Застосувати зміни»

3. Керування процесами сервера

Для керування процесами сервера використовується менеджер процесів pm2. Основні команди:

- pm2 list – виводить список процесів;
- pm2 start – запускає вказаний процес на виконання;
- pm2 stop – зупиняє вказаний процес;
- pm2 restart – перезапускає вказаний процес;
- pm2 delete – видаляє вказаний процес.

Використання цих команд напряму не є необхідним – для роботи з застосунком було створено окрему конфігурацію, що їх використовує (рис. 11).

```
"scripts": {
    "start:prod": "pm2 start npm --name \"server\" -- run start:server",
    "start:dev": "webpack-dev-server --hot --open --mode development",
    "start:server": "node server.js",
    "build": "rm -rf dist && webpack --config webpack.config.production.js ",
    "now-build": "npm run build",
    "test": "xo && stylelint './src/**/*.js'"
},
```

Рис. 11. Конфігурація скриптів для серверу

Для того щоб зміни на сервері були застосовані до веб-додатку потрібно:

- Якщо процес веб-додатку вже виконується, треба виконати bash script описаний у файлі «pzks_website/restart_server.sh»;
- Якщо процес веб-додатку ще не запущено, потрібно використовуючи командний рядок спочатку ввести *npm run build* для компіляції проєкту та *npm run start:prod* для запуску серверу.