

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: « Підсистема « Модуль « Публікації » »

Виконав:

студент IV курсу, групи ТВ-71

Лелет Євгеній Олегович _____

Керівник:

Діденко Олексій Олександрович _____

Рецензент:

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

Студент (-ка) _____

Київ – 2021 року

ЗМІСТ

ВСТУП.....	5
1. ПОСТАНОВКА ЗАДАЧІ.....	7
2. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ.....	8
2.1 Аналіз предметної області.....	13
2.2 Аналіз аналогічних застосунків.....	13
2.2.1 Вебсервіс Мюнхенського технічного університету.....	13
2.2.2 Вебсервіс Ягеллоського університету у Кракові.....	13
2.2.3 Вебсервіс Національного університету ім. О.О. Богомольця.....	13
2.3 Аналіз аналогічних вебсервісів.....	13
2.4 Актуальність розробки вебсервісу.....	13
3.ОПИС ПОБУДОВАНОЇ СИСТЕМИ.....	15
3.1 Загальний опис системи.....	20
3.2 Архітектура системи.....	20
3.3 Особливості реалізації.....	21
3.4 Висновки до розділу.....	27
4. ОПИС ПРОГРАМНИХ ЗАСОБІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ.....	24
4.1 Обґрунтування вибору мови програмування.....	29
4.2 Обґрунтування вибору бази даних.....	34
4.2.1 Огляд видів баз даних.....	13
4.2.2 Аналіз аналогічних вебсервісів.....	39
4.2.3 Аналіз СКБД PostgreSQL.....	40
4.3 Обґрунтування вибору фреймворку для серверної частини.....	41
4.3.1 Аналіз Node.js.....	41
4.3.2 Аналіз Bootstrap.....	43
4.3.3 Аналіз Express.js.....	44
4.3.4 Аналіз бібліотеки React.....	45
4.3.5 Аналіз Angular.....	46
4.4 Висновки до розділу.....	47

5. ВИПРОБУВАННЯ РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ.....	43
5.1 Аналіз розробленого вебсервісу.....	13
5.2 Тестування вебсервісу	49
5.3 Порівняння даного вебсервісу з його аналогами.....	50
5.4 Рекомендації на рахунок подальшого вдосконалення	51
5.5 Аналіз аналогічних вебсервісів	53
ВИСНОВОК.....	54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	56
ДОДАТОК А	53
ДОДАТОК Б.....	55
ДОДАТОК В.....	66
ДОДАТОК Г	76

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення.

ПК – персональний комп'ютер.

IDE – Integrated Drive Electronics.

API – Application Programming Interface.

HTTP – HyperText Transfer Protocol.

JSON – JavaScript Object Notation.

DOM – Document Object Model.

JS – JavaScript.

ВСТУП

Оптимізація процесів онлайн навчання є актуальною проблемою для закладів та сервісів у сфері освіти. Конкуренція в розробці динамічних інструментів для навчання стає підґрунтям для технологічних пошуків для великих та гнучких корпорацій, соціальних мереж та державних установ.

Сучасні практики створення гнучких вебсервісів дозволяють створювати достатньо якісні інструменти. Змінився й сам підхід до інформації, більше немає потреби зберігати інформацію на індивідуальних носіях, значно важливіше знати де можна знайти потрібну актуальну інформацію в мережі інтернет. Зазначені раніше компанії зможуть ефективно інтегрувати алгоритмічну систему в існуючі маркетингові стратегії лише за наявності інтуїтивно зрозумілого інтерфейсу користувача, що слідує класичним методикам розроблення ПЗ. Навчання існуючого персоналу заданим інструкціям по використанню вебдодатку повинна бути зведена до мінімуму.

Для початку, акумуляція онлайн навчання є витратною в часі й ресурсах задачею, оскільки інформація дисимільована на різних, взаємо не пов'язаних між собою сервісах із відмінними рейтинговими системами, фільтрами для сортування тощо. Також, навіть за наявності системи збору та динамічного відстеження даних постає проблема їх автоматизованого аналізу. Сьогодні таку обробку мануально виконують працівники відділень досліджень. Створення комплексу, що включає в себе як збір, так і аналіз, у формі автоматизованої класифікації, вимагає великого досвіду команди розробників у імплементації алгоритмів обробки природніх мов та машинного навчання. Також важливо щоб сайт містив необхідні функціональні можливості та контент заради якого люди його відвідують. Очевидно, що процес створення такої системи вимагатиме суттєві часові витрати.

Слід зазначити що створення описаної вище системи не є гарантією комерційного успіху та фактом усунення проблем дистанційного навчання, оскільки вимагає мануального адміністрування та аналізу отриманих результатів.

Основною метою дипломного проєкту є створення нового вебсервісу кафедри автоматизації проектування енергетичних процесів і систем теплоенергетичного факультету Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського”.

1. ПОСТАНОВКА ЗАДАЧІ

Метою даної дипломної роботи розробити серверну та клієнтську частину вебсервісу кафедри автоматизації проектування енергетичних процесів і систем теплоенергетичного факультету Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського” .

Завдання:

- проаналізувати нормативні документи, що регламентують роботу сайтів підрозділів університету та існуючі програмні аналоги й визначити вимоги до розроблюваного вебсервісу;
- розробити серверну частину вебсервісу;
- розробити клієнтську частину вебсервісу;
- налаштувати взаємодію серверної та клієнтської частини;
- протестувати роботу системи;

Програмне забезпечення повинно відповідати наступним вимогам:

- забезпечити безперервний доступ користувачів;
- обмеження доступу до приватних ресурсів на сервері;
- коректна обробка будь-яких запитів користувачів;
- використання захищеного протоколу передачі даних;
- забезпечення можливості редагування інформації;
- наявність працюючого RSS;
- резервне копіювання даних;
- відмовостійкість;
- масштабованість;

2. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1. Аналіз предметної області

Функціонування сайтів наукових підрозділів Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського” регламентується статутом університету, чинним законодавством, наказами та розпорядженнями ректора, положенням про інформаційно-освітній портал НТУУ “КПІ”, внутрішніми розпорядженнями структурного підрозділу університету, розпорядженнями проректора.

Згідно нормативних документів основними завданнями сайту кафедри є:

- Вирішення освітніх та науково-інноваційних задач кафедри та університету використовуючи сучасні технології;
- Створення позитивного іміджу кафедри, як університетського підрозділу, що здатен конкурувати на міжнародному ринку послуг у сфері освіти та науки;
- Об’єктивне й швидке інформування студентів та викладачів, а також інших зацікавлених осіб про різні аспекти діяльності кафедри ;
- Обмін інформацією між різними підрозділами університету;

Чинна версія вебсервісу відповідає описаним вище вимогам не у повному обсязі. Але технічні характеристики вебсервісу та дизайн відповідають очікуванням користувачів, звикшим користуватися вебсервісами виконаними з використанням сучасних технологій у сфері розробки.

Метою цього дипломного проєкту є розробка серверної та користувацької частини вебсервісу кафедри автоматизації проектування енергетичних процесів і систем теплоенергетичного факультету Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського” , яка повинна додавати можливість публікувати наукові статті.

2.2 Аналіз аналогічних застосунків

З метою огляду та аналізу аналогічних систем було розглянуто вебсервіси різних університетів України та світу. Було визначено їх переваги й недоліки.

2.2.1. Вебсервіс Мюнхенського технічного університету

Вебсервіс структурно поділяється на такі підрозділи:

- Про навчання;
- Про міжнародні зв'язки;
- Про студентське життя;
- Про університет;
- Про науку;
- Про інновації;

В якості переваг можна позначити мінімалістичний сучасний дизайн без зайвих елементів інтерфейсу, використання захищеного протоколу передачі даних, доволі зручну навігацію по вебсервісу, стрічку новин RSS, високу швидкість.

Серед недоліків можна виділити відсутність особистого кабінету користувача.

На сторінках вебсервісу медіа контент представлений у високій якості. На рис.1, на головній сторінці, знаходяться посилання на сторінки університету у соціальних мережах.

Клієнтська частина розроблена з використанням HTML, CSS. Для розробки серверної частини використали JavaScript.



Рисунок 1. Головна сторінка вебсервісу Мюнхенського технічного університету

2.2.2. Вебсервіс Ягеллонського університету у Кракові

Структура даного вебсервісу залежить від обраної мови інтерфейсу. Для англійської мови доступні такі розділи:

- Про навчання;
- Про університет;
- Про наукову діяльність;
- Про кадровий склад;
- Про міжнародну співпрацю;

Серед переваг даного вебсервісу сучасний дизайн, локалізація для декількох мов, висока швидкість, захищений протокол передачі даних, можливість перемикається до більш контрастної версії інтерфейсу та змінювати розмір шрифту, є стрічка новин RSS, підтримка мобільних пристроїв.

На головні сторінці вебсервісу, на рис. 2, є посилання на сторінки університету в соціальних мережах. Також якість медіа контенту на сторінках вебсервісу доволі висока.

Клієнтська частина розроблена з використанням HTML, CSS та різних бібліотек JavaScript. Серверна частина виконана з допомогою мови програмування PHP.

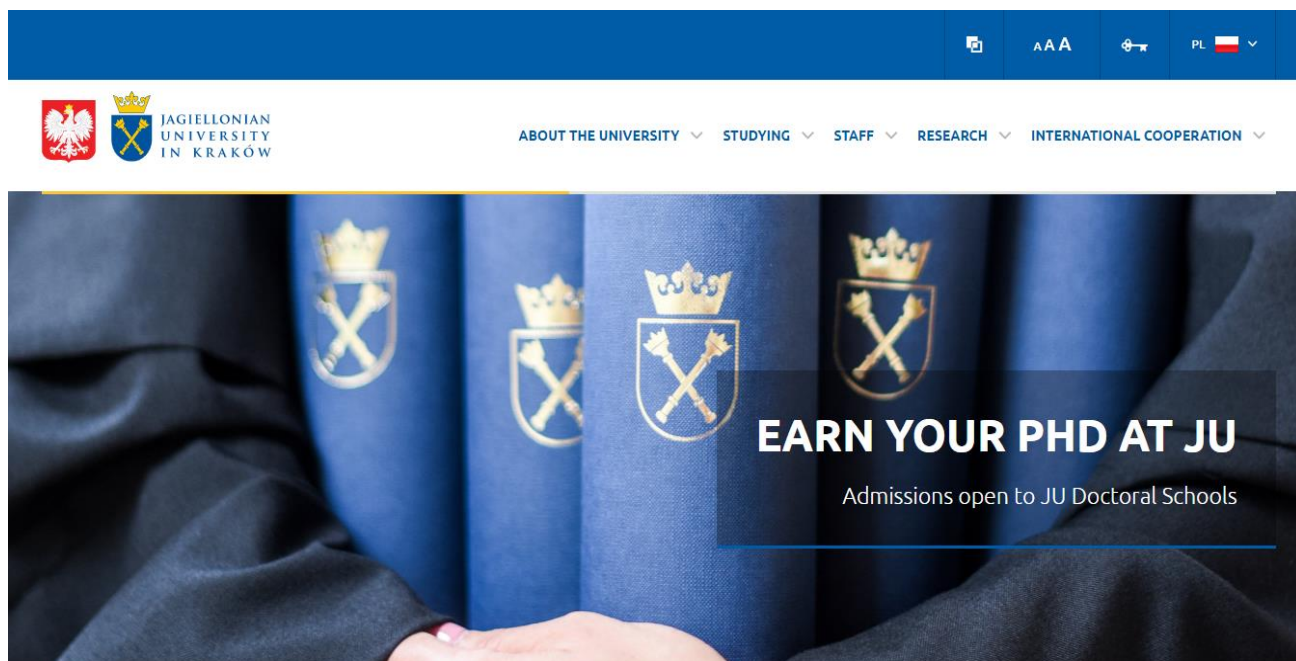


Рисунок 2. Головна сторінка вебсервісу Ягеллонського університету

2.2.3. Вебсервіс Національного медичного університету ім. О.О. Богомольця

Вебсервіс містить наступні розділи:

- Загальна інформація;
- Про науку;
- Інформація для студентів;
- Інформація для вступників;
- Інформація для випускників;

Переваги даного вебсервісу це зручна навігація, мінімалістичний, сучасний дизайн сайту, висока швидкість, використання захищеного протоколу передачі даних, відсутність зайвих елементів інтерфейсу, стрічка новин RSS.

Серед недоліків можна відмітити відсутність підтримки мобільних пристроїв. Хоч на головній сторінці сайту, зображеній на рис. 3, присутні посилання на сторінки

університету в соціальних мережах, більшість посилань не працює або працює не коректно.

Клієнтська частина розроблена на HTML та CSS. Серверна частина розроблена на мові програмування JavaScript та СКБД MySQL.

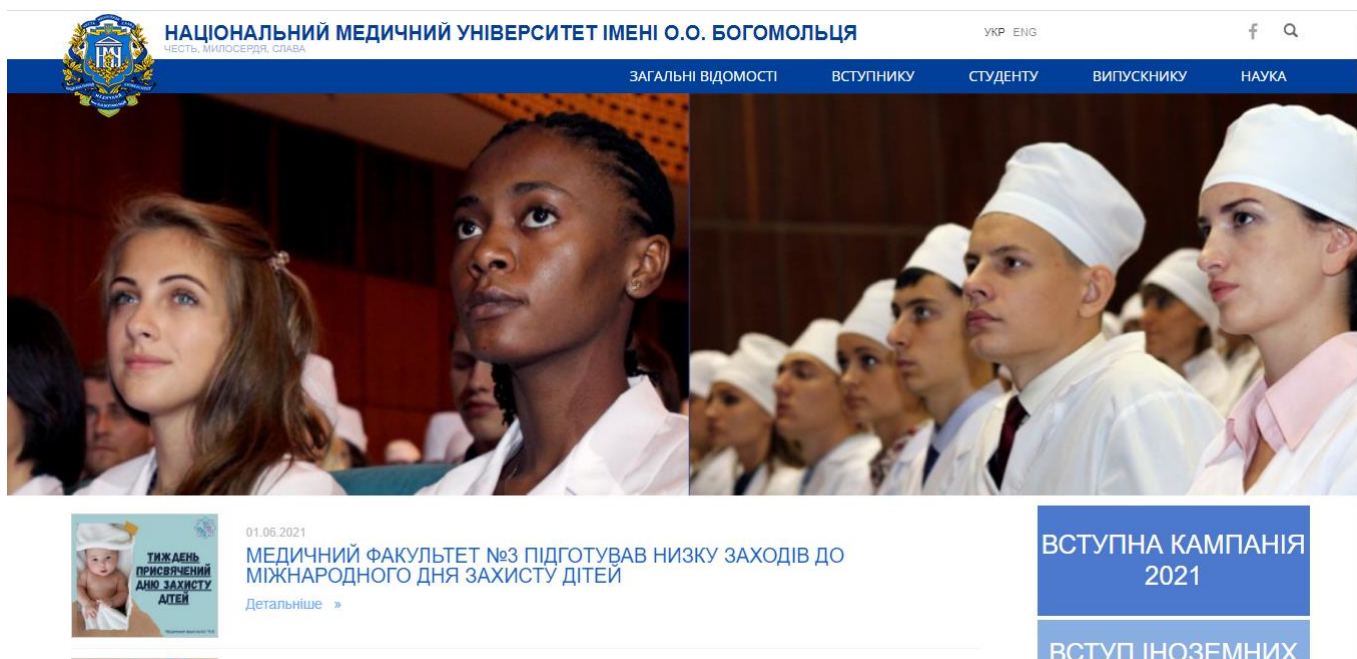


Рисунок 3. Головна сторінка вебсервісу НМУ ім. О.О. Богомольця

Таблиця 1

Порівняння програмних рішень

	Сучасний дизайн	Швидкодія	Підтримка мобільних пристроїв	Захищене підключення
Мюнхенський технічний університет	+	+	-	+
Ягеллонський університет	+	+	+	+
НМУ ім. Олександра Богомольця	+	+	-	+

В результаті проведеного аналізу було встановлено що серед протестованих вебсервісів найкращим є вебсервіс Ягеллонського університету. Він має усе що необхідно: сучасний дизайн, підтримка на мобільних пристроях, захищене підключення, висока швидкодія, тощо.

2.3 Актуальність розробки вебсервісу

У ході аналізу аналогічних застосунків були наведені у приклад вебсервіси вищих навчальних закладів Європи й України. Кожен аналог розроблений відповідно до вимог організації яку він представляє, хоч декі технічні характеристики в них і спільні.

Жоден з наведених аналогів не може бути використаним у силу відмінностей у наповненні та структурі сайту.

У результаті аналізу аналогічних систем було сформовано вимоги до клієнтської та серверної частини та до вебсервіса в цілому:

- сучасний та інтуїтивно зрозумілий користувацький інтерфейс ;
- відповідність структури вебсервісу вимогам описаним у положенні про типовий сайт кафедри НТУУ «КПІ»;
- висока швидкодія;
- захищений протокол передачі даних;
- резервне копіювання даних;
- коректна обробка запитів користувача;
- можливість редагування інформації;

Захист інформації має відбуватися за допомогою механізму авторизації. Також можливість редагування, публікації або видалення контенту має бути доступним лише авторизованим користувачам відповідно до їх рівня доступу.

2.4 Актуальність розробки вебсервісу

Так як вебсервіси є одним з головних ресурсів для пошуку інформації про навчальний заклад, люди змушені його використовувати за відсутності кращого варіанту. На сьогоднішній день більшість вебсервісів навчальних закладів України є застарілими. Застарілий дизайн та заплутаний, незрозумілий інтерфейс, відсутність підтримки на мобільних пристроях, погана швидкодія погіршують відношення людей до закладу який цей сайт представляє. Також користувачам важко знайти потрібну їм інформацію на такому сайті.

Розроблюваний вебсервіс кафедри автоматизації проектування енергетичних процесів і систем теплоенергетичного факультету Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського” має вирішити цю проблему відповідно до усіх представлених вимог.

3. ОПИС ПОБУДОВАНОЇ СИСТЕМИ

3.1. Загальний опис системи

Інформація що розміщена на даному вебсервісі повинна бути подана в зручному для користувача вигляді та відповідати визначеній структурі.

Користувачам з правами адміністратора має бути надана можливість публікації, редагування, та видалення інформації.

Вебсервіс має містити розділи для представлення інформації:

- загальна інформація;
- про освітні програми;
- про навчальний план;
- інформація для вступу на 1 курс;
- інформація для вступу на 5 курс;
- публікації наукових робіт;
- публікації викладачів та студентів;
- інформація про заходи що проводяться на кафедрі;
- інформацію про взаємодію кафедри із зарубіжними партнерами;
- інформацію про матеріально-технічне забезпечення;
- новини кафедри;

3.2. Архітектура системи

Для розробки вебсервіса було використано клієнт-серверну модель архітектури.

Клієнт-серверна архітектура – шаблон розробки програмного забезпечення, що базується на взаємодії серверу та клієнтів через мережу (рис. 4.). Сервер та клієнти представлені у вигляді програмних модулів що функціонують незалежно один від одного. Взаємодія виглядає так: клієнт робить запит на сервер, сервер отримує запит, обробляє його й повертає результат роботи клієнту. Одночасно сервер здатен обробляти запити від багатьох клієнтів. Як правило у ролі клієнта виступає браузер.

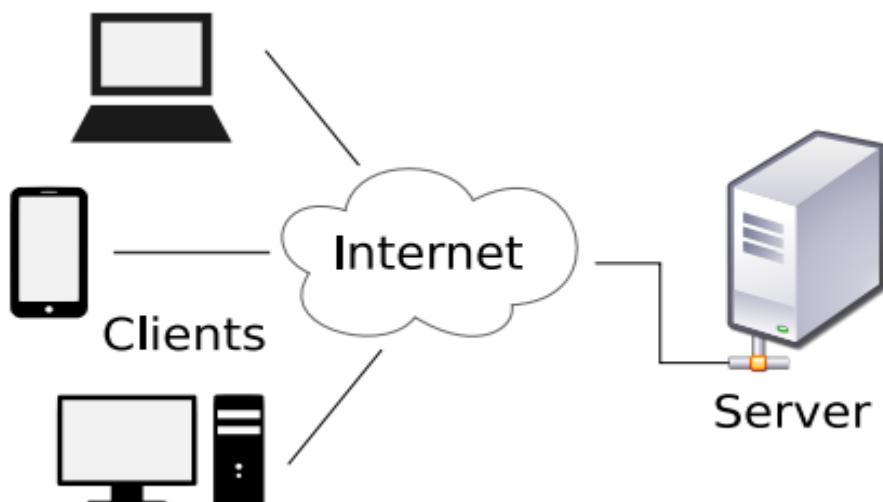


Рисунок. 4. Клієнт-серверна архітектура

3.3. Особливості реалізації

Серверна та клієнтська частина розроблені на основі шаблону Bootstrap 4 з використанням фреймворку Node.js.

Схема бази даних

Було створено вісім взаємодіючих між собою таблиць для реалізації створеного вебсервісу:

- users;
- news;
- curriculum;
- consultation timetable;
- exam timetable;
- staff;
- graduation papers;
- publication items;
- publications;

Проілюстровано базу даних на рис.5. Під час створення бази даних було досягнуто між швидкодією та нормалізацією.

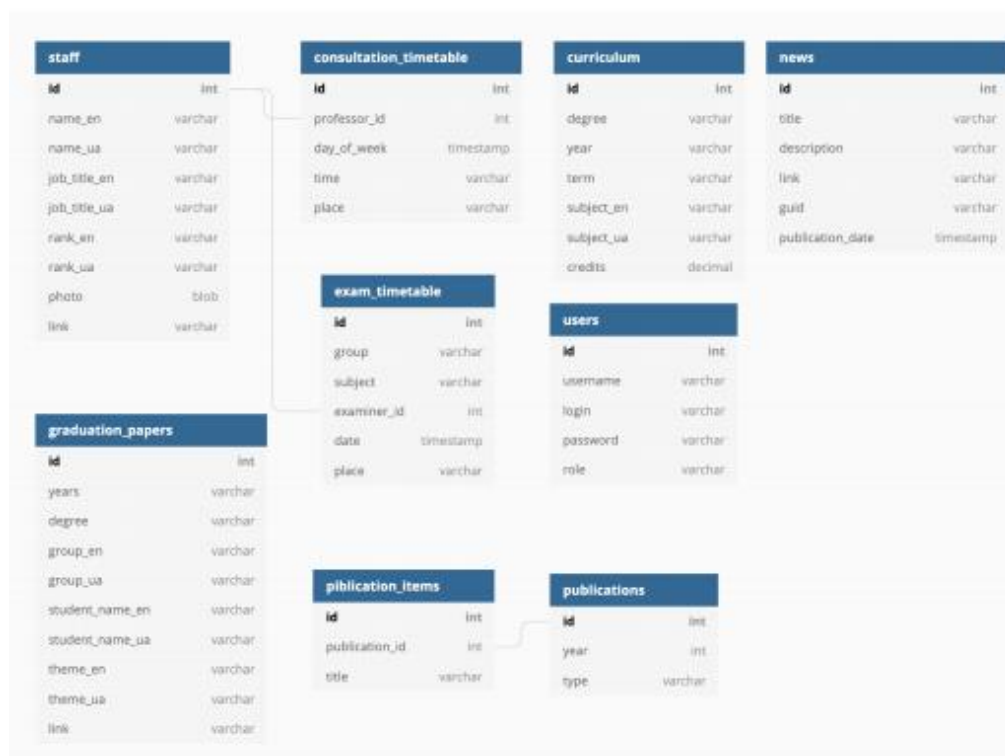


Рисунок. 5. Схема бази даних

Users

Тут міститься інформація про авторизованих користувачів. Детальніше у таблиці 2.

Таблиця 2

Поля таблиці Users

<i>Назва поля</i>	<i>Тип даних</i>
id	SERIAL AUTOINCREMENT
username	TEXT
login	TEXT
password	TEXT
role	TEXT

Consultation timetable

Тут міститься інформація про розклад викладачів. Детальніше у таблиці 3.

Таблиця 3

Поля таблиці Consultation_timetable

<i>Назва поля</i>	<i>Тип даних</i>
id	SERIAL AUTOINCREMENT
professor_id	INT (Foreign Key of Staff)
day_of_week	TEXT
time	TEXT
place	TEXT

Exam timetable

Тут міститься інформація про розклад екзаменів. Детальніше у таблиці 4.

Таблиця 4

Поля таблиці Exam_timetable

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

Staff

Тут міститься інформація про кадровий склад кафедри. Детальніше опис у таблиці 5.

Таблиця 5

Поля таблиці Staff

<i>Назва поля</i>	<i>Тип даних</i>
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

Publication items

Тут міститься інформація про конкретну публікацію. Таблиця має зв'язок з таблицею Publications. Детальніше у таблиці 6.

Таблиця 6

Поля таблиці Publication_items

<i>Назва поля</i>	<i>Тип даних</i>
id	SERIAL AUTOINCREMENT
publication_id	INT (Foreign Key of Publication)
title	TEXT

Graduation paper

Тут міститься інформація про випускні роботи студентів. Детальніше у таблиці 7.

Таблиця 7

Поля таблиці Graduation_papers

<i>Назва поля</i>	<i>Тип даних</i>
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

Publications

Тут міститься інформація про елемент архіву публікацій. Детальніше у таблиці 8.

Таблиця 8

Поля таблиці Publications

<i>Назва поля</i>	<i>Тип даних</i>
id	SERIAL AUTOINCREMENT
year	INT
type	TEXT

Схема взаємодії клієнтської та серверної частини

Клієнтська та серверна частини взаємодіють за допомогою викликів REST API. Детальніше на рис. 6.

Взаємодія у фреймворку Node.js забезпечується функціями-обробниками HTTP запитів. Відбувається взаємодія через функцію що містить аргументи:

- req – HTTP запит, що приходить на сервер;
- res – відповідь сервера яку відправляє сервер у форматі JSON;

Механізм REST API мають право використовувати лише користувачі що авторизовані у системі. Іншим користувачам буде дана відповідь HTTP з кодом помилки 401.

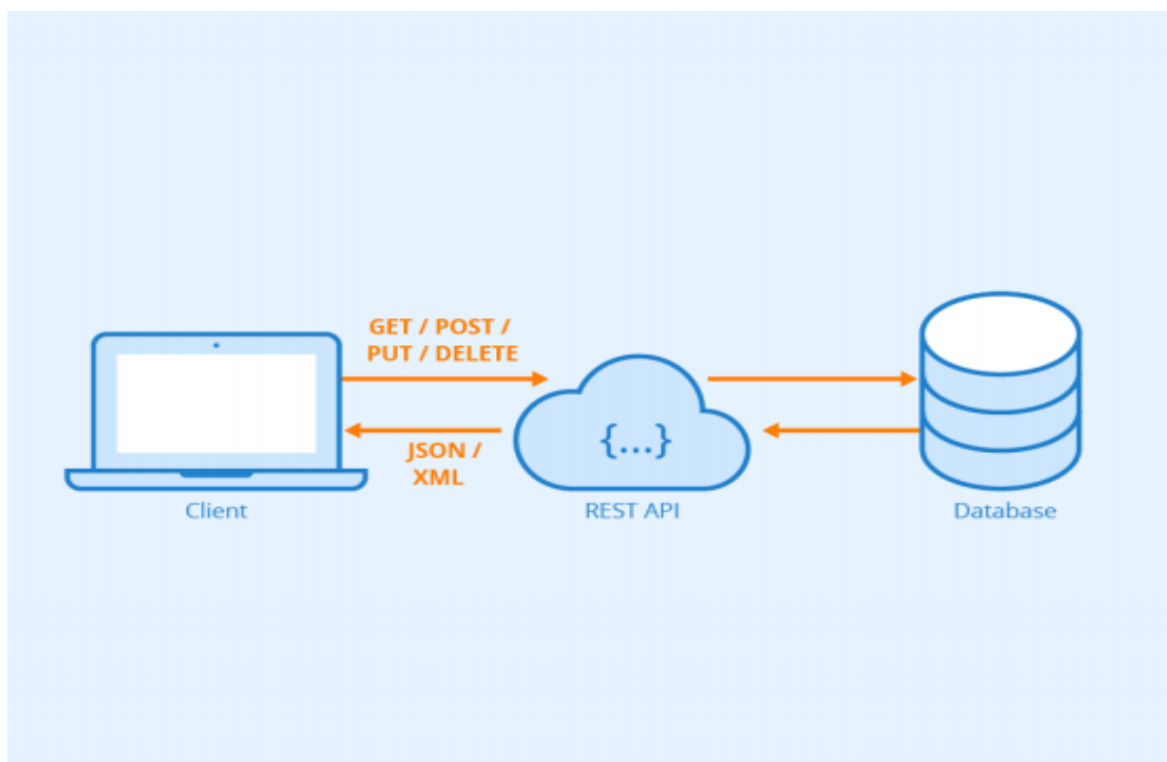


Рисунок 6. Робота REST API

Вхід у систему

Створена система має можливості автентифікації що відносяться до типу Session Based Authentication. Детальніше на рис.7.

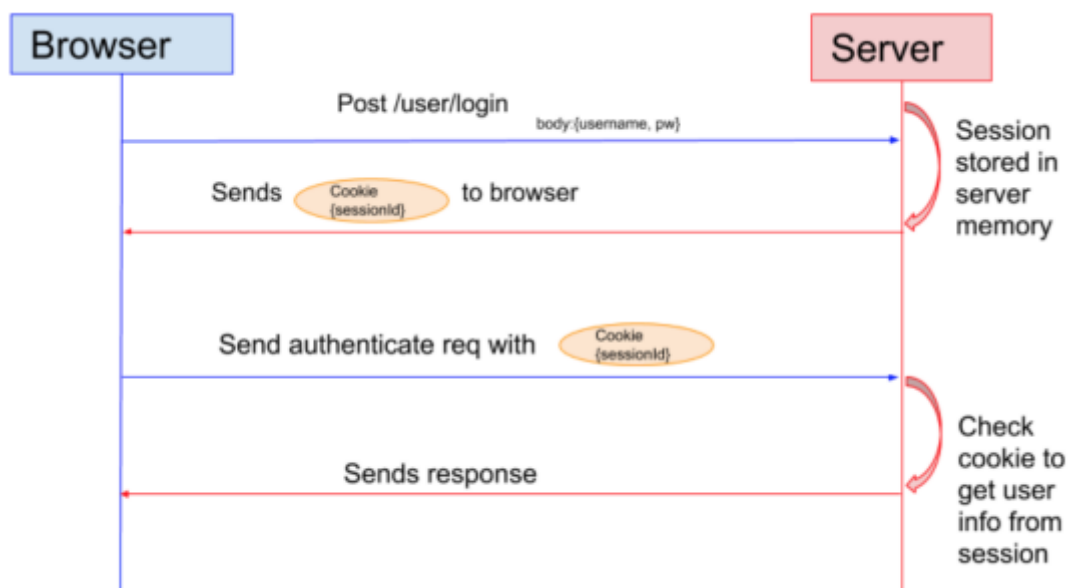


Рисунок 7. Session Based Authentication

Існують такі типи користувачів:

- Admin – користувач що має не тільки усі можливості звичайного користувача, але й можливість додавати та видаляти користувачів що не є адміністраторами;
- Guest – не авторизований користувач що може лише переглядати контент сайту, але не може вносити будь-які зміни;
- User – звичайний авторизований користувач для якого доступні не тільки перегляд контенту але й усі функціональні можливості вебсервісу;

Адміністратор відповідає за реєстрацію нових користувачів у системі.

3.4. Висновки до розділу

Було виконано загальний опис користувацької та серверної частини системи розроблюваного вебсервісу кафедри автоматизації проектування енергетичних процесів і систем теплоенергетичного факультету Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського”. Також було описано роботу бази даних та наведено схему її реалізації, детальний опис кожної її таблиці та полів. В цьому розділі описано як відбувається взаємодія

клієнтської та серверної частини через REST API, та особливості взаємодії в цій конкретній системі. Також було описано процес автентифікації в системі, типи користувачів в ній та їх можливості.

4. ОПИС ПРОГРАМНИХ ЗАСОБІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ

4.1. Обґрунтування вибору мови програмування

Так як розроблюваний програмний продукт має бути створений у вигляді вебсервісу, необхідно було обрати саме мову програмування для розробки даного типу програмного забезпечення.

Мова програмування має відповідати таким вимогам:

- підтримка сучасних технологій;
- підтримка усіх необхідних парадигм програмування;
- велика кількість фреймворків та бібліотек для розробки вебсервісів;
- швидкість та продуктивність;
- простота і раціональність застосування;

Мова програмування JavaScript

JavaScript – високорівнева мова програмування, що була створена для такого щоб оживити вебсторінки. Програми що написані на цій мові називають скриптами. Вони можуть вбудовуватися в HTML та виконуватися автоматично при завантаженні вебсторінки. Скрипти поширюються та виконуються як звичайний текст. Їм не потрібна спеціальна підготовка чи компіляція для запуску, тощо.

Основними особливостями є:

- незамінність для веброзробки;
- швидкість роботи та ефективність;
- потужна інфраструктура;
- простота та раціональність застосування;
- зручність користувацьких інтерфейсів;
- легкість засвоювання;
- велика кількість готових інструментів;

- велика спільнота розробників;

Наявність системи модулів та пакетів дозволяє краще організовувати та використовувати кодову базу цієї мови. JavaScript підтримує різні парадигми програмування. З створенням фреймворку Node.js стало можливим використовувати Javascript не тільки в розробці користувацької частини а й на стороні сервера.

До недоліків JavaScript можна віднести динамічну типізацію, що є суттєвим недоліком особливо при створенні великих систем та їх тестуванні, хоча при прототипуванні це можна віднести до переваг.

Мова програмування Java

Java – мова програмування зі строгою типізацією, підтримкою об'єктно орієнтованої парадигми програмування. Розроблена компанією Sun Microsystems.

Широкої популярності ця мова програмування набула завдяки інноваційній концепції віртуальної машини яка зробила її кросплатформною. Реалізовано це через компіляцію Java коду у байт-код з подальшим виконанням байт-коду віртуальною машиною на будь-якій операційній системі до якої віртуальну машину реалізовано. Використовується Java для розробки мобільних додатків, десктопних додатків та серверної частини вебсервісів.

Java має такі переваги:

- простота – перша технічна перевага Java;
- об'єктно орієнтований підхід до програмування;
- безпека – важливий критерій враховуючи використання Java в розробці мережових середовищ;
- продуктивність – нові версії динамічних компіляторів;
- надійність – програми на Java працюють стабільно в будь-яких умовах;
- незалежність від апаратної частини і ОС;
- динамічність та адаптивність;
- зручні та ефективні мережові можливості;

Серед недоліків Java можна визначити багатослівний синтаксис а ще погану продуктивність без додаткових налаштувань у віртуальній машині.

Мова програмування Python

Python – інтерпретована мова програмування, з одного боку це дозволяє значно спростити налагодження програми. З іншої – спричиняє відносно повільну швидкість виконання. Популярним Python зробили простий синтаксис і легкість навчання. Як правило Python використовують для створення серверної частини вебсервісів, обробки та аналізу даних й написання простих скриптів для взаємодії з операційною системою.

Серед переваг Python можна визначити:

- динамічна типізація;
- якісна підтримка модульності;
- вбудована підтримка Unicode;
- підтримка об'єктно-орієнтованого програмування;
- автоматичний збір сміття;
- інтеграція із C/C++;
- зрозумілий та лаконічний синтаксис;
- величезна кількість модулів;
- кросплатформність;

Серед недоліків Python можна визначити доволі низьку швидкість виконання що спричинена використанням інтерпритатора.

Мова програмування C#

C# - об'єкто-орієнтована мова програмування. Створена у 1998-2001 групою інженерів компанії Microsoft як мова для розробки додатків для платформи Microsoft. Проста, потужна, статично типізована мова. Входить в сімейство мов програмування із C – подібним синтаксисом. C# увібрав у себе кращі якості таких мов як C, C++, Modula, Object Pascal, спираючись на досвід їх практичного застосування. В багатьох

відношеннях мова C# дуже схожа на Java, ця схожість помітна у синтаксисі та основних поняттях цих мов програмування.

До переваг C# можна віднести:

- підтримка переважної більшості продуктів Microsoft;
- безкоштовність ряду інструментів для невеликих компаній;
- типи даних мають фіксований розмір;
- автоматичний збір сміття;
- велика кількість синтаксичного цукру;
- низький поріг входження;
- C – подібний знайомий синтаксис;
- наявність потужної IDE;

Серед недоліків C# можна виділити прив'язку до платформи .NET, що до недавніх пір була доступна лише для операційної системи Windows. Але на сьогоднішній день створено інструменти для розробки на C# під будь-які ОС.

Мова програмування PHP

PHP – скриптова мова програмування загального призначення що інтенсивно застосовується в вебсервісах. В даний час підтримується переважною більшістю хостинг-провайдерів і є одною із лідерів ринку серед мов що використовуються для створення динамічних вебсервісів .

Серед переваг PHP можна виділити:

- висока швидкість роботи і, відповідно, низьке використання ресурсів;
- бюджетність, економічність;
- простий синтаксис;
- відмінна сумісність та переносимість;
- набір тексту та редагування коду можна виконувати в будь-якому текстовому редакторі;

- висока гнучкість, ємкість та функціональність;
- багатозадачність та широкі можливості;

Серед недоліків PHP можна відмітити наявність дір в системі безпеки та деякі протиріччя у синтаксисі.

Висновок щодо обраної мови

Проаналізувавши усі мови програмування перераховні вище було визначено їх основні переваги та недоліки перед їх використанням для побудови цільової системи:

- головний недолік мови програмування Python це низька швидкодія, хоча ця мова й сприяє швидкій розробці програмного забезпечення, але не є кращим вибором для реалізації проєкту;
- головний недолік мови програмування C# це необхідність придбання комерційної ліцензії, хоча ця мова й зручна у використанні для розробки програмного забезпечення, але не є кращим вибором для реалізації проєкту;
- головним недоліком мови програмування Java є низька швидкодія та надмірно ускладнений синтаксис, тому навіть враховуючи усі її переваги при розробці програмного забезпечення, не є кращим вибором для реалізації проєкту;
- головний недолік мови програмування Python це низька швидкодія, хоча ця мова й сприяє швидкій розробці програмного забезпечення, але не є кращим вибором для реалізації проєкту;
- основним недоліком мови програмування PHP це діри у системі безпеки та протиріччя у синтаксисі, тому ця мова не є кращим вибором для реалізації проєкту, хоча й має ряд незаперечних переваг;

4.2. Обґрунтування вибору бази даних

4.2.1. Огляд видів баз даних

Серед усіх баз даних що існують на сьогодні виділяють два типи: реляційні, також відомі як SQL, та нереляційні, також відомі як NoSQL. Відмінність між ними у тому що вони представляють різні моделі: реляційну та нереляційну. В нереляційних базах даних дані представляють залежно від типу самої бази. В той же час реляційні бази даних представляють дані у вигляді таблиць.

Типи нереляційних баз даних:

- ключ-значення (Key-value);
- документно-орієнтоване сховище;
- колоночне сховище;
- графове сховище;

Реляційна модель має на увазі логічне перетворення даних: таблиці, уявлення і індекси. Логічна структура відрізняється від фізичної структури зберігання. Така дія дає можливість адміністраторам керувати системою зберігання даних, не змінюючи даних, налаштовується в логічній структурі. Наприклад, зміна імені файлу бази даних не вплине на дані що зберігаються в таблиці.

Поділ між фізичними і логічним рівнем поширюється в тому числі на операції, які представляють собою чітко визначені дії з даними і структурами бази даних. Логічні операції дають можливість додаткам визначити вимоги до необхідного змісту, в той час як фізичні операції визначають спосіб доступу до даних і виконання завдання.

Щоб забезпечити точність і доступність даних, в реляційних базах повинні дотримуватися певних правил цілісності. Наприклад, в правилах цілісності можна заборонити використання дублікатів рядків в таблицях, щоб усунути ймовірність потрапляння неправильної інформації в базу даних.

Компанії всіх типів і розмірів використовують просту, але функціональну реляційну модель для обслуговування різноманітних інформаційних потреб. Реляційні бази даних застосовуються для відстеження товарних запасів, обробки торгових транзакцій через Інтернет, управління великими обсягами критично важливих даних замовників, тощо. Реляційні бази даних можна рекомендувати для будь-яких інформаційних потреб, де елементи даних пов'язані між собою і необхідно забезпечувати безпечне і надійне управління ними на основі правил цілісності.

Реляційна модель найбільш ефективно підтримує цілісність даних у всіх додатках і копіях (примірниках) бази даних. Наприклад, коли замовник кладе гроші на рахунок за допомогою банкомату, а потім перевіряє баланс на мобільному телефоні, він очікує, що кошти, що надійшли відразу ж відобразяться на рахунку. Реляційні бази даних відмінно підходять для забезпечення цілісності даних в різних телефонах бази в один і той же час.

Переваги реляційної системи бази даних:

- простота й доступність для розуміння користувача;
- єдиною використовуваною інформаційною структурою є таблиця;
- строгі правила проєктування;
- базування на математичному апараті;
- повна незалежність даних;
- зміни в програмному забезпеченні при зміні реляційної бази даних мінімальні;
- для організації запитів і написання прикладного програмного забезпечення немає необхідності знати конкретну організацію бази даних у зовнішній пам'яті;
- наявність потужних інструментів обробки даних;
- велика кількість фахівців та документації у зв'язку з широкою популярністю;
- підтримка стандарті SQL;
- мінімальні затрати пам'яті;

- для задач що допускають реалізацію за допомогою будь-якої з трьох моделей даних ієрархічна модель дозволяє отримати уявлення з мінімально необхідною пам'яттю;

Недоліки реляційних баз даних

- далеко не завжди предметна область може бути представлена у вигляді таблиць;
- в результаті логічного проєктування з'являється безліч таблиць;
- при великих об'ємах труднощі з розумінням структури даних;
- займає відносно багато зовнішньої пам'яті;
- відносно низька швидкість доступу до даних;
- трудомісткість розробки;
- некоректне уявлення сутностей реального світу;
- семантичне перенавантаження;
- слабка підтримка обмежень цілісності і корпоративних обмежень;
- однорідна структура даних;
- обмежений набір операцій;
- складності при обробці рекурсивних запитів;
- проблема неузгодженості типів даних;
- інші проблеми пов'язані з паралельним використанням, змінами схеми і нерозвиненими засобами доступу;

Бази даних NoSQL спеціально створені для певних моделей даних і мають гнучкі схеми, що дозволяє розробляти сучасні програми. Бази даних NoSQL набули широкого поширення в зв'язку з простотою розробки, функціональністю і продуктивністю при будь-яких масштабах. Бази даних NoSQL використовують різноманітні моделі даних для доступу до даних і управління ними. Бази даних таких типів оптимізовані для додатків, які працюють з великим обсягом даних, потребують низької затримки і гнучких моделей даних. Все це досягається шляхом пом'якшення жорстких вимог до несуперечності даних, характерних для інших типів БД. Бази

даних NoSQL добре підходять для багатьох сучасних додатків, наприклад мобільних, ігрових, вебсервісів, коли потрібні гнучкі масштабовані бази даних з високою продуктивністю і широкими функціональними можливостями, здатні забезпечувати максимальну зручність використання.

Переваги NoSQL:

- лінійна масштабованість – додавання нових вузлів в кластер збільшує загальну продуктивність системи;
- гнучкість, що дозволяє оперувати напівструктурованими даними, реалізуючи в тому числі повнотекстовий пошук по базі;
- можливість працювати з різними уявленнями інформації, в тому числі без завдання схеми даних;
- висока доступність за рахунок реплікації даних і інших механізмів відмовостійкості, зокрема, шаринга – автоматичного розподілу даних за різними вузлами мережі, коли кожен сервер кластера відповідає тільки за певний набір інформації, обробляючи запити на його читання і запис. Це збільшує швидкість обробки даних і пропускну здатність додатку;
- продуктивність за рахунок оптимізації для конкретних видів моделей даних (документованої, графової, колонкової або ключ - значення) і шаблонів доступу;
- широкі функціональні можливості – власні SQL – подібні мови запитів, RESTful - інтерфейси, API і складні типи даних, наприклад map, list і struct, що дозволяють обробляти величезну кількість значень;

Недоліки NoSQL:

- обмежена ємність вбудованої мови запитів, для вирішення цієї проблеми використовуються сторонні засоби трансляції класичних SQL – виразів в виконуваний код для конкретної нереляційної бази даних;
- складності в підтримці всіх ACID – вимог до транзакцій (атомарність, консистентність, ізоляція, довговічність) через те, що NoSQL – база даних замість CAP – моделі (узгодженість, доступність, стійкість до поділу)

скоріше відповідають моделі BASE (базова доступність, гнучкий стан і підсумкова узгодженість);

- сильна прив'язка додатку до конкретної СКБД через специфіку внутрішньої мови запитів і гнучкої моделі даних, орієнтованої на конкретний випадок;
- недостаток фахівців з NoSQL – баз в порівнянні з реляційними аналогами;

Підводячи підсумок опису основних аспектів нереляційних СКБД, варто відзначити деяку некоректність запиту в зв'язку з різними архітектурними підходами і прикладними завданнями, на які орієнтовані ці засоби. Традиційні SQL – бази відмінно справляються з обробкою строго типізованої інформації невеликої кількості. Наприклад, локальна ERP – система або хмарна CRM. Однак, в разі обробки великого обсягу напівструктурованих і неструктурованих даних, тобто Big Data, в розподіленій системі слід обирати з безлічі NoSQL – сховищ, з огляду на специфікацію самого завдання. Зокрема, для самостійних рішень інтернету речей, в тому числі промислового, відмінно підходить Cassandra. А в разі багаторівневої IT – інфраструктури на базі Apache Hadoop варто звернути увагу на HBase, яка дозволяє оперативно, практично в режимі реального часу, працювати з даними, що зберігаються в HDFS.

4.2.2. Аналіз СКБД MySQL

MySQL – це система керування базами даних, яка поширюється як вільне програмне забезпечення (користувачі мають право на необмежену установку, запуск, вільне використання). З переваг СКБД MySQL варто відзначити простоту використання, гнучкість, низьку вартість володіння (щодо платних СКБД), а також масштабованість і продуктивність.

Гнучкість СКБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиця InnoDB, що підтримують транзакції на рівні окремих записів. Є й інші типи таблиць, розроблені спільнотою.

СКБД MySQL з'явилася в 1995. Написана на C і C++, протестована на безлічі різних компіляторів і працює на різних платформах. З 2010 року розробку та підтримку MySQL здійснює корпорація Oracle. Продукт поширюється як під GNU GPL, так і під власною комерційною ліцензією. Однак за умовами GPL, якщо яка – небудь програма включає вихідні коди MySQL, то і ця програма теж повинна розповсюджуватися за ліцензією GPL. Для небажаючих відкривати вихідні тексти своїх програм якраз передбачена комерційна ліцензія яка на додаток до можливості розробки під закритою ліцензією забезпечує якісну сервісну підтримку. Спільнотою розробників MySQL створені різні розгалуження – Drizzle, OurDelta, Percona Server і MariaDB, всі ці розгалуження вже існували на момент отримання прав на MySQL корпорацією Oracle.

Переваги MySQL:

- багатопоточність, підтримка декількох запитів одночасно;
- оптимізація зв'язків з приєднанням багатьох даних за один прохід;
- записи фіксованої і змінної довжини;
- ODBC драйвер;
- гнучка система привілеїв та паролів;
- гнучка підтримка форматів чисел, рядків змінної довжини та міток часу;
- інтефейс з мовами C, Perl, PHP;
- швидка робота, масштабованість;
- сумісність із ANSI SQL;
- безкоштовна в більшості випадків;
- хороша підтримка з боку провайдерів послуг хостингу;
- швидка підтримка транзакцій через механізм InnoDB;

Недоліки MySQL:

- MySQL розроблялась з акцентом на швидкість та простоту використання через що у ній відсутні деякі можливості SQL;

- ціна – деякий функціонал з можливостей MySQL є доступними лише на комерційній основі;
- не надто швидкий розвиток – через це що цю СКБД часто перекупають співтовариство не може швидко вносити зміни;
- зниження швидкості при обробці великих об'ємів даних;

4.2.3. Аналіз СКБД PostgreSQL

PostgreSQL – це популярна вільна об'єктно-реляційна система управління базами даних. PostgreSQL базується на мові SQL і підтримує численні можливості. Функції в PostgreSQL є блоками коду, виконуваними на сервері, а не на клієнті бази даних. Хоча вони можуть писатися на чистому SQL, реалізація додаткової логіки, наприклад, умовних переходів і циклів, виходить за рамки власне SQL і вимагає використання деяких мовних розширень. Функції можуть писатися з використанням різних мов програмування. PostgreSQL допускає використання функцій, які повертають набір записів, який далі можна виконувати так само як і результат звичайного запиту. Функції можуть виконуватися як з правами їх творця, так і з правами поточного користувача. Іноді функції ототожнюються зі збереженими процедурами, проте між цими поняттями є різниця.

Переваги PostgreSQL:

- підтримка БД необмеженого розміру;
- потужні та надійні механізми транзакцій та реплікацій;
- розширювана система вбудованих мов програмування і підтримка завантаження C – сумісних модулів;
- наслідування;
- легка поширюваність;
- вкладені запити;
- уявлення;
- посилавна цілісність;
- складні блокування;

- типи визначені користувачем;
- перевірка сумісності версій;

Недоліки PostgreSQL:

- продуктивність – при простих операціях читання PostgreSQL може значно уповільнити сервер і бути повільніше своїх конкурентів, таких як MySQL;
- популярність – за своєю природою, популярністю ця СКБД похвалитися не може, хоча і має досить велику спільноту;
- хостинг – в силу названих вище чинників іноді досить складно знайти хостинг з підтримкою цієї СКБД;

Було виконано порівняння MySQL та PostgreSQL, розглянуто основні відмінності обох систем правління базами даних і здійснена спроба зрозуміти що краще MySQL чи PostgreSQL. В загальному результаті кращим за можливостями виходить PostgreSQL, але у даному проєкті доцільно застосувати саме MySQL.

4.3. Обґрунтування вибору фреймворку для серверної частини

4.3.1. Аналіз Node.js

Node.js – являє собою середу виконання коду на JavaScript, яка побудована на основі JavaScript Chrome V8, який дозволяє транслювати виклики на мові JavaScript в машинний код. Node.js перш за все призначений для створення серверної частини вебсервісів на мові програмування JavaScript. Хоча також існують проєкти по написанню десктопних додатків (Electron) і навіть зі створення коду для мікроконтролерів. Але перш за все ми говоримо про Node.js, як про платформу для створення вебсервісів. Вебсервіси написані слідуєчи клієнт / серверній архітектурі, працюють за такою схемою – клієнт запитує потрібний ресурс у сервера і сервер відправляє ресурс у відповідь. У цій схемі сервер, відповівши на запит, перериває з'єднання. Така модель ефективна оскільки кожен запит до сервера споживає ресурси. Для того щоб обробляти кожен наступний запит від клієнта, сервер повинен завершити обробку попереднього.

Переваги Node.js:

- можливість застосовувати одну мову на клієнті і сервері;
- при володінні мовою програмування JavaScript, буде легше вивчити надбудову ніж кардинально іншу технологію, розроблявся спеціально для web;
- вільно взаємодіє з найбільш популярними базами даних;
- допомагає отримати низькорівневий доступ;
- швидкість – створення робочого прототипу не відніме багато часу, перший етап коли програмис формує кістяк майбутнього продукту, проходить дуже швидко;
- якщо архітектура продумана – то в подальшому труднощів з тим, щоб розшиювати свій сайт на Node.js не винекне;
- велика та актина спільнота;
- відкритий код;
- синтаксис JavaScript – це заезпечує відносну простоту і доступність, при вивченні буде багато знайомого;
- вбудована бібліотека, кожним оновленням вона розширюється, але навіть у першій версії були досить широкі можливості ;
- постійний розвиток екосистеми, кількість готових модулів і звнішніх бібліотек безперервно зростає;
- двигун V8, він економічно витрачає пам'ять, непогано оптимізований, дає функціонал по реалізації процесора і пам'яті;
- технологія стрімко поліпшується, над нею працюють сотні спеціалістів з усього світу;
- із активною спільнотою технологія розвивається значно швидше і все завдяки відкритому коду;

4.3.2 Аналіз Bootstrap

Bootstrap – це найбільш популярний фреймворк HTML, CSS, JS для розробки в першу чергу мобільних проєктів в мережі інтернет. Однією зі сторін Bootstrap є те, що його можна використовувати виключно як набір вже готових елементів. Для цього не потрібно серйозних знань HTML і CSS – достатньо вміти створювати найпростіші сторінки і прочитати документацію. Наприклад для створення кнопки використовується два класи, один з яких відповідає за структуру кнопки, а інший за оформлення. Такий підхід називається об'єкто орієнтований CSS.

Процес розробки супроводжується безперервною роботою по додаванню або видаленню функціоналу на сторінці. Чим більше проєкт, тим частіше відбуваються ці процеси і швидкість реалізації не в останню чергу залежить від того, як скоро функціонал зовнішнім виглядом. Використовуючи готові компоненти і утиліти Bootstrap можна швидко ввести новий функціонал на сайт і дати його користувачам.

Переваги Bootstrap:

- кросбраузерність – компоненти Bootstrap враховують відмінності браузерів і оптимізовані таким чином, щоб мінімізувати відмінності між ними;
- адаптивність – Bootstrap має готову сітку, по якій можна розташовувати компоненти;
- сітка є найпопулярнішим компонентом Bootstrap і її використовують навіть ті, хто скептично ставиться до фреймворку, сітка використовує підхід при якому можливо відразу налаштувати адаптивність;
- використовуючи готові компоненти можна бути впевненим у тому що вони будуть добре відображатися на різних пристроях;
- доступність – розробники приділяють увагу на використанні сторінок з компонентами людьми з обмеженими можливостями;
- дозволяє приховати елемент для всіх пристроїв, крім скрінрідера, це важливо для правильної організації верстки форм на сайті;

- створення сторінки за компонентами не вимагає експертного знання HTML і CSS, це дозволяє створювати нові блоки не тільки верстальнику але й будь – якому працівнику компанії;

Недоліки Bootstrap:

- вага проєкту – Bootstrap дає не тільки набір класів але й інтерактивні компоненти, для їх роботи використовують кілька JavaScript бібліотек, які збільшують вагу файлів на сторінці;
- дизайн – стандартні компоненти накладають обмеження на дизайн, ці рамки виражені в тому, що компоненти можливо розширювати лише вглиб, вставляючи одні компоненти всередину інших, але навіть в таких обмеженнях створюються красиві і зручні сайти за рахунок роботи з UI;
- як і у випадку з дизайном, функціонал блоків визначений заздалегідь і зміна приводить до суттєвих доопрацювань, це стосується й інтерактивних елементів, які використовують JavaScript код, для таких елементів створено класи які допомагають змінити поведінку але цього не завжди буває достатньо;

4.3.3. Аналіз Express.js

Express являє собою популярний фреймворк для вебсервісів, написаний на JavaScript і працює всередині середовища виконання Node.js. Цей модуль висвітлює деякі ключові переваги цього фреймворка, встановлення середовища розробки і виконання основних завдань розробки і згортання.

Переваги Express.js:

- простота;
- гнучкість;
- хороша масштабованість;
- розвинена спільнота;
- детальні документація;
- широкий вибір модулів;

Недоліки Express.js:

- великий об'єм ручної роботи;
- використовує застарілий підхід callbacks функцій;

4.3.4. Аналіз бібліотеки React

React часто згадують в одному ряді з іншими JavaScript фреймворками, але суперечки не мають сенсу, тому що це не співставні речі. Наприклад Angular – це повноцінний фреймворк (що включає і рівень уявлення). React – ні. Ось чому React викликає стільки нерозуміння в світі повноцінних фреймворків – це тільки уявлення.

React надає мову шаблонів і деякі callback функції для відтворення HTML. Весь результат роботи React – це HTML. Ваші зв'язки HTML / JavaScript, що називаються компонентами, займаються тим, що зберігають свій внутрішній стан в пам'яті, але в підсумку вам просто повертає HTML.

Переваги React:

- ви завжди можете сказати, як ваш компонент буде відмальовувати, дивлячись на вихідний код. Це може бути вашою перевагою, хоча це нічим не відрізняється від шаблонів;
- зв'язування JavaScript і HTML в JSX робить компоненти простими для розуміння;
- є можливість рендерити React на сервері. Якщо ви розробляєте публічний сайт або додаток, і ви Рендерите все на клієнті, це є неправильний шлях. Клієнтський рендеринг це причина чому SoundCloud працює повільно;

Недоліки React:

- відсутність системи подій;
- відсутність роботи із AJAX;
- відсутність будь-якого шару даних;
- відсутність promises;
- це не фреймворк на усі випадки життя;
- відсутність будь-яких ідей як реалізувати усе перераховане вище;

- погана та незрозуміла документація;
- React досить великий враховуючи те, як мало ви від нього отримаєте, враховуючи погану кроссбраузерну підтримку;

4.3.5. Аналіз фреймворку Angular

Angular являє собою фреймворк від компанії Google для створення клієнтських додатків. Перш за все він націлений на розробку SPA - рішень (Single Page Application), тобто односторінкових додатків. В цьому плані Angular є спадкоємцем іншого фреймворку AngularJS. У той же час Angular це не нова версія AngularJS, а принципово новий фреймворк.

Angular надає таку функціональність, як двостороннє зв'язування, що дозволяє динамічно змінювати дані в одному місці інтерфейсу при зміні даних можелі в іншому, шаблони, маршрутизація і так далі.

Однією з ключових особливостей Angular є те, що він використовує в якості мови програмування TypeScript.

Але ми не обмежені мовою програмування TypeScript. При бажанні можемо писати програми на Angular за допомогою таких мов як Dart або JavaScript. Однак TypeScript все ще є основною мовою для Angular.

4.4. Висновки до розділу

Було виконано порівняння MySQL та PostgreSQL, розглянуто основні відмінності обох систем правління базами даних і здійснена спроба зрозуміти що краще MySQL чи PostgreSQL. В загальному результаті кращим за можливостями виходить PostgreSQL, але у даному проєкті доцільно застосувати саме MySQL.

Також було протестовано JavaScript фреймворк Node.js, визначено його переваги й недоліки і ми говоримо про Node.js, як про платформу для створення вебсервісів. Вебсервіси написані слідуючи клієнт / серверній архітектурі, працюють за такою схемою – клієнт запитує потрібний ресурс у сервера і сервер відправляє ресурс у відповідь. У цій схем сервер, відповівши на запит, перериває з'єднання. Така

можель ефективна оскільки кожен запит до сервера споживає ресурси. Для того щоб обробляти кожен наступний запит від клієнта, сервер повинен завершити обробку попереднього.

Було протестовано HTML, CSS, JS фреймворк Bootstrap, зважено його переваги й недоліки і зроблено висновок що його використання для даного проекту є доцільними та виправданим.

5. ВИПРОБУВАННЯ РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ

5.1. Аналіз розробленого вебсервісу

Було створено вебсервіс, з використання сучасних технологій та з урахуванням усіх вимог та тенденцій до розробки програмного забезпечення. Головною функцією вебсервісу є надання інформації у чіткому сучасному інтерфейсі.

Враховуючи поставлене технічне завдання було обрано наступні рішення для розробки цільового вебсервісу:

- була реалізована система додавання, редагування та видалення контенту вебсервісу;
- розроблений вебсервіс має високу швидкодію;
- також було реалізовано взаємодію клієнтської та серверної частини вебсервісу;
- до вебсервісу підключено базу даних;
- контент вебсервісу надійно захищений за допомогою захищеного протоколу передачі даних;
- було реалізовано збагачене зведення сайту;

Було реалізовано функцію автентифікації користувачів та публікації користувачів з відповідними правами. Авторизованим користувачам надана можливість:

- робити нові публікації;
- редагувати вже існуючі публікації;
- видаляти публікації;
- користуватися усіма функціональними можливостями даного вебсервісу;

Користувачі з правами адміністратора також мають можливість змінювати інформацію, редагувати, додавати та видаляти інших користувачів. Клієнтська та серверна частина даного вебсервісу взаємодіють на основі REST API.

Взаємодія даного вебсервісу із БД реалізована з використанням SQL – запитів.

Типи запитів до бази даних:

- SELECT – команда для зчитування даних з бази даних;
- INSERT – команда для додавання даних у базу даних;
- UPDATE – команда для оновлення наявних даних у базі даних;
- DELETE – команда для видалення наявних даних з бази даних;

5.2. Тестування вебсервісу

Тестування даного вебсервісу – це процес дослідження програмного забезпечення з метою виявлення помилок і визначення відповідності між реальною та очікуваною поведінкою програмного забезпечення який здійснюється на основі набору тестів, обраних певним чином. У більш широкому сенсі, тестування програмного забезпечення – це техніка контролю якості програмного продукту, що включає в себе проєктування тестів, виконання тестування і аналіз отриманих результатів.

Дуже часто сучасні програмні продукти розробляються в стислі терміни і при обмежених бюджетах проєктів. Програмування сьогодні перейшло з розряду мистецтва в розряд ремесел для багатьох мільйонів фахівців. Але на жаль в такому поспіху розробники часто ігнорують необхідність забезпечення захищеності своїх продуктів, піддаючи тим самим користувачів на невиправданий ризик. Контроль якості вважається важливим в процесі розробки програмного забезпечення, тому що забезпечує безпеку, надійність, зручність створюваного продукту. В даний час існує безліч підходів і методик до вирішення завдання тестування програмного забезпечення, але ефективне тестування складних програмних систем – процес творчий, що не зводиться до проходження суворим і чітким правилам.

Перевірка роботи клієнтської частини була проведена шляхом використання усіх можливостей користувача, в тому числі: публікування, редагування та видалення публікацій. Публікування інформації різних об'ємів, типів інформації, та в усіх випадках вебсервіс показав коректну роботу.

Для перевірки коректної роботи серверної частини вебсервісу були використані автоматичні тести, завдяки яким у майбутньому тестувати систему буде значно швидше та легше.

5.3. Порівняння даного вебсервісу із його аналогами

Було проведено аналіз вебсервісів кращих учбових закладів Європи і України. Їх було порівняно між собою там визначено їх недоліки й переваги. Вони були розроблені з урахуванням технічного завдання учбових закладів що вони представляють. Можливості даних вебсервісів відрізняються відповідно до потреб учбових закладів.

Основними критеріями для оцінки даних вебсервісів було визначено:

- за допомогою опитування було виявлено що одним з основних показників є сучасність дизайну;
- за допомогою тестування було виявлено що швидкодія вебсервісу не гірше, або навіть краще ніж у його аналогів;
- також важливим показником для вебсервісу у наші дні є зручність інтерфейсу;
- також важливо відмітити використання захищеного протоколу передачі даних;
- за допомогою тестування різними шляхами була перевірена зручність даного вебсервісу;
- важливим показником також є крос-платформність програмного забезпечення;
- наявність збагаченого зведення сайту також є важливим для даного вебсервісу;

В результаті порівняння даного вебсервісу з його аналогами можна стверджувати що розроблений вебсервіс відповідає усім вимогам та може конкурувати з іншими аналогами. Публікування інформації різних об'ємів, типів інформації, та в усіх випадках вебсервіс показав коректну роботу.

5.4. Рекомендації на рахунок подальшого вдосконалення

Для удосконалення даного вебсервісу було визначено такі шляхи:

- створення функції зворотнього зв'язку що допоможе користувачам висловити свою думку по мікросервіс;
- створення сховища даних що допоможе при подальшій підтримці вебсервісу;
- покращення інтерфейсу;
- збільшення кількості можливостей;
- створення кабінету користувача;
- необхідно розробити автоматичне програмне забезпечення для того щоб вебсервіс зміг витримувати більші навантаження;
- основне дзеркало і єдиний формат URL – завдяки добре продуманою URL – адресою люди і пошукові системи можуть відразу зрозуміти про що цільова сторінка;
- перевірка файлу robots.txt і sitemap.xml – простий текстовий файл robots.txt може сильно вплинути на те як пошукові системи сканують ваш сайт. Цей текстовий файл не є обов'язковим, але містить інструкції для пошуку систем, про те як сканувати сайт і підтримується всіма основними пошуковими системами. Однак цей протокол носить чисто рекомендаційний характер і може бути проігнорований роботами скануючими вебсторінки якщо вони того бажають;
- перевірка правильності HTTP – заголовків які повертає сервер;
- перевірка оформленої сторінки 404-ї помилки;
- перевірка битих посилань – непрацюючі посилання дратують і погано відзначаються на вашому професіоналізмі. Вони не тільки утримують

відвідувачів від того, щоб вони залишалися а сайті і поверталися на нього, але також можуть негативно вплинути на загальний рейтинг в пошукових системах;

- Meta Title I Meta Description – теги заголовка відображаються на сторінках результатів пошукової системи у вигляді клікабельного заголовка для даного результату і важливі для зручності використання та соціального обміну. Тег заголовку вебсторінки призначений для точного і короткого опису вмісту сторінки. Google зазвичай відображає перші 50 – 60 символів тега заголовка. Якщо отримати заголовок завдовжки до 60 символів, дослідження показують, що можна очікувати що блзько 90% заголовків будуть відображатися правильно;
- оптимізація заголовків – теги заголовків надають структуру і контекст для статті. Кожен заголовок повинен дати читачеві уявлення про інформацію, яку вони можуть почерпнути з тексту абзацу, який слідує нижче;
- перевірка редиректорів на сайті – перенаправлення вебсервісу буде приймати адресу одного сайту і вказувати на інший. Коли хто-небудь вводить або натискає на цей оригінальний URL, він потрапляє на нову сторінку або сайт;
- перевірку сайту в Google PageSpeed Insights (PSI) – повідомляє про продуктивність як на мобільних так і на настільних пристроях, а також пропонує рекомендації щодо поліпшення цієї сторінки;
- перевірка мікророзмітки – це семантичний словник тегів який ви можете додати в свій HTML, щоб поліпшити спосіб, яким пошукові системи чіають та представляють вашу сторінку в пошуковій видачі;
- перевірка сторінок пагінації – це процес, який ділить контент на ряд сторінок. Це поширена й широко використовувана техніка для вебсервісів, що використовує нумерацію сторінок для поділу списків статей або продуктів в зручний формат;

- зайвий код на сторінках сайту – це перше що сканують боти пошукової системи. Провести роботу над видаленням непотрібних рядків коду над основним текстом, виводячи код JavaScript та CSS, який заважає змісту ключових слів;
- дубльовані елементи на сайті – це контент, який з’являється в інтернеті більш ніж в одному місці, це місце визначається, як місце розташування з унікальною адресою вебсервісу, тому, якщо один і той же контент з’являється на кількох веб адресах, ви отримуєте дубльований контент;

5.5. Висновки до розділу

У цьому розділі було проведено аналіз розробленого ПЗ що являє собою вебсервіс кафедри автоматизації проектування енергетичних процесів і систем теплоенергетичного факультету Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського”.

Було порівняно розроблений програмний продукт з існуючими аналогами та зроблено висовок що даний вебсервіс відповідає усім вимогам а також нічим не поступається своїм аналогам. У якості аналогів були обрані сайти одних з кращих навчальних закладів світу. Також було проведено тестування розробленого вебсервісу різними шляхами, як ручне так і авоматичне, з описом інструментів що були застосовані.

ВИСНОВКИ

Мета даного дипломного проєкту була досягнута, було створення вебсервісу кафедри автоматизації проектування енергетичних процесів і систем теплоенергетичного факультету Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського”.

З метою створення якісного програмного продукту попередньо було проведено детальний аналіз нормативних документів що регламентують роботу вебсервісів університету, а також було проведено детальний аналіз інших вебсервісів одних з кращих навчальних закладів України та Європи з метою визначення технічного завдання для розроблюваного вебсервісу. Після детального аналізу усього вище перерахованого було визначене технічне завдання для розроблюваного програмного продукту.

Також у ході виконання даного проєкту було виконано детальний аналіз програмних інструментів з метою визначення який саме з них найкраще підходить для реалізації даного вебсервісу. Було обрано такі інструменти розробки як мова програмування JavaScript, фреймворк для розробки серверної частини - Node.js, фреймворк для розробки клієнтської частини – Bootstrap, та система керування базами даних MySQL.

У даному дипломному проєкті було створено архітектуру клієнтської та серверної частини вебсервісу. Шляхом впровадження системи авторизації було реалізовано розділення функціональних можливостей користувачів, а також захист даних від сторонніх користувачів.

В ході розробки даного вебсервісу було виконано:

- а) досліджено літературу за темою дипломної роботи;
- б) досліджено можливості мови програмування JavaScript та фреймворку Node.js;
- в) здійснено проектування макетів застосунку в Adobe Photoshop;
- г) здійснено роботу з API;
- д) реалізовано фронт-енд та бек-енд;

- е) розроблено технічну специфікацію;
- ж) розроблено інструкції користувача у роботі з системою;
- з) побудовано архітектуру вебсервісу;
- и) протестовано розроблену систему.

Таким чином, під час виконання дипломної роботи було розроблено готовий вебсервіс, отримано нові навички й покращено старі в розробці вебсервісів, проектування дизайну, побудові архітектури та роботі з API.

СПИСОК ВИКОРИСТАНИХ 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. Eric Matthes. Python crash course. No starch press, 2016.
5. Mark J.Price. C# 8.0 and .NET Core 3.0.
6. Cay S. Horstmann. Core Java Volume I—Fundamentals, 2018.
7. Eloquent JavaScript: A Modern Introduction to programming
8. Advantages And Disadvantages of RDBMS [Електронний ресурс]. — Режим доступу до ресурсу: <http://tiny.cc/4mo37y>.
9. An Overview of SQL and NoSQL with it's Pros and Cons [Електронний ресурс]. — Режим доступу до ресурсу: <http://tiny.cc/4mo37y>.
10. Andrew Comeau. MySQL Explained: Your Step By Step Guide to Database Design. 2nd Edition, 2015
11. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems [Електронний ресурс]. — Режим доступу до ресурсу: <http://tiny.cc/ptp37y>.
12. Di Battista, Giuseppe, Eades, Peter; Tamassia, Roberto, Tollis, Ioannis G. (1994), "Algorithms for Drawing Graphs: an Annotated Bibliography", Computational Geometry: Theory and Applications.
13. Thomas M. J. Fruchterman, Edward M. Reingold (1991). Graph drawing by force-directed placement. , 21(11), 1129–1164. doi:10.1002/spe.4380211102
14. Client Server Architecture [Електронний ресурс]. ³/₄ 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.

ДОДАТОК А

Підсистема « Модуль « Публікації »

Специфікація

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_TV71108_21А 1

Аркушів 2

Київ – 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ"КПІ"_ТЕФ_АПЕПС_ТІ628 7_20Б 2	Записка Лелет Є.О. ТІ-71.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ"КПІ"_ТЕФ_АПЕПС_ТІ628 7_20Б 2-1	sb-admin-2.js	Компонент побудови
УКР.НТУУ"КПІ"_ТЕФ_АПЕПС_ТІ628 7_20Б 2-2	chart-area-demo.js	Компонент аналізу відношень
УКР.НТУУ"КПІ"_ТЕФ_АПЕПС_ТІ628 7_20Б 2-3	chart-bar-demo.js	Компонент допоміжної панелі
УКР.НТУУ"КПІ"_ТЕФ_АПЕПС_ТІ628 720Б 2-4	publications.html	Компонент керування публікаціями

ДОДАТОК Б

Підсистема « Модуль « Публікації »

Лістинг програми

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_TV71108_21Б 2

Аркушів 11

sb-admin-2.js

```

import 'package:concept_maps/controllers/force_directed_controller.dart';
import 'package:concept_maps/models/graph_entities/map_model.dart';
import 'package:concept_maps/models/graph_entities/node.dart';
import 'package:concept_maps/providers/app_provider.dart';
import 'package:concept_maps/views/paint_graph.dart';
import 'package:concept_maps/views/widgets/drawer_menu.dart';
import 'package:concept_maps/views/widgets/search_app_bar.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter/physics.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:provider/provider.dart';
import 'package:concept_maps/views/bottom_sheet_annel.dart';
import 'package:concept_maps/models/graph_entities/node.dart';
import 'package:vector_math/vector_math_64.dart';
import 'dart:math';

import 'bottom_sheet_annel.dart';

class ForceDirected extends StatefulWidget {
  @override
  _ForceDirectedState createState() => _ForceDirectedState();
}

class _ForceDirectedState extends State<ForceDirected>
  with TickerProviderStateMixin {
  //об'явлення змінних
  AnimationController animationController;
  Animation<Matrix4> animation;

  Animation<double> graphAnimation, graphCurve;
  AnimationController graphAnimationController;

  Node focusNode;
  bool graphFlag;
  int count;
  Vector2 temp;
  MapModel map;
  ForceDirectedController controller;
  var flag;
  Offset frame;
  TransformationController transformationController =
    TransformationController();
  //логіка графової анімації
  void runGraphAnimation(Duration d){
    graphAnimationController.duration = Duration(microseconds: d.inMicroseconds);
    graphCurve =
      CurvedAnimation(parent: graphAnimationController, curve: Curves.easeInOut);

    graphAnimationController.forward(from: 0.0);
  }

  void runAnimation(Offset position, double scale) {
    final size = MediaQuery.of(context).size;
    Matrix4 matrix = Matrix4.copy(transformationController.value);
    matrix.row0 = Vector4(scale, 0, 0, -position.dx * scale + size.width / 2);
    matrix.row1 =
      Vector4(0, scale, 0, -position.dy * scale + size.height * 0.2);
    matrix.row2 = Vector4(0, 0, scale, 0);
    //matrix.row3 = Vector4(position.dx, position.dy, 0, 1);

    animation = animationController.drive(
      Matrix4Tween(begin: transformationController.value, end: matrix));

    const spring = SpringDescription(
      mass: 15,
      stiffness: 1,
      damping: 1,
    );

    final simulation = SpringSimulation(spring, 0, 1, -5);

    animationController.animateWith(simulation);
  }
}

```

```

void fillWidg() {
    var nodeSize = 100.0;
    controller.widgets.clear();
    controller.titles.clear();
    //задання початкових положень вершин
    controller.edges.forEach((element) {
        double a, c, angle = 0;
        if(element.v.position.y <= element.u.position.y && element.v.position.x <= element.u.position.x){
            a = Vector2(element.v.position.x - element.v.position.x, element.v.position.y - element.u.position.y).length;
            c = Vector2(element.v.position.x - element.u.position.x, element.v.position.y - element.u.position.y).length;
            angle = asin(a/c);
        } else if(element.u.position.y <= element.v.position.y && element.v.position.x <= element.u.position.x){
            a = Vector2(element.v.position.x - element.v.position.x, element.v.position.y - element.u.position.y).length;
            c = Vector2(element.v.position.x - element.u.position.x, element.v.position.y - element.u.position.y).length;
            angle = -asin(a/c);
        } else if(element.v.position.y <= element.u.position.y && element.u.position.x <= element.v.position.x){
            a = Vector2(element.v.position.x - element.v.position.x, element.v.position.y - element.u.position.y).length;
            c = Vector2(element.v.position.x - element.u.position.x, element.v.position.y - element.u.position.y).length;
            angle = -pi - asin(a/c);
        } else if(element.u.position.y <= element.v.position.y && element.u.position.x <= element.v.position.x){
            a = Vector2(element.v.position.x - element.v.position.x, element.v.position.y - element.u.position.y).length;
            c = Vector2(element.v.position.x - element.u.position.x, element.v.position.y - element.u.position.y).length;
            angle = -pi + asin(a/c);
        }
    });

    controller.widgets.add(
        Positioned(
            top: element.v.position.y - 25,
            left: element.v.position.x + 25,
            child: Transform.rotate(
                angle: angle,
                alignment: Alignment.centerLeft,
                child: InkWell(
                    enableFeedback: false,
                    canRequestFocus: false,
                    highlightColor: Color(0x0032f16f),
                    hoverColor: Color(0x0032f16f),
                    splashColor: Color(0x0032f16f),
                    onTap: () {
                        setState(() {
                            runAnimation(Offset((element.u.position.x + element.v.position.x)/2, (element.u.position.y + element.v.position.y)/2), 0.5);
                            context.read<AppProvider>().setFocusNode(controller.balloon.three[
                                controller.balloon.three
                                    .indexWhere((a) => a.id == element.u.id)));
                            context.read<AppProvider>().animationStart = false;
                            context.read<AppProvider>().isEdgeActive = true;
                            context.read<AppProvider>().focusEdge = element;
                            context.read<AppProvider>().focusTitle = element.u.fullTitle+ " — "+element.v.fullTitle;
                            context.read<AppProvider>().setBottomSheetFlag(true);
                        });
                    },
                ),
            ),
        ),
    );
}

//побудова карт понять
controller.vertices.forEach((element) {
    TextPainter textPainter = TextPainter(
        text: TextSpan(text: element.title),
        maxLines: 1,
        textDirection: TextDirection.ltr)
    ..layout(minWidth: 0, maxWidth: double.infinity);
    double textWidth = textPainter.width;

    controller.titles.add(Positioned(
        top: element.position.y + element.size / 2,
        left: element.position.x - textWidth,
        child: Text(
            element.title,
            //+"\\n"+element.attr.x.toString()+"\\n"+element.attr.y.toString()

```

```

//+"\n"+element.displacement.x.toString()+"\n"+element.displacement.y.toString(),
style: GoogleFonts.montserrat(
  fontSize: 22,
  fontWeight: FontWeight.w700,
),
));

controller.widgets.add(Positioned(
  top: element.position.y*graphAnimation.value + element.prevPosition.y*(1 - graphAnimation.value) - element.size / 2,
  left: element.position.x*graphAnimation.value + element.prevPosition.x*(1 - graphAnimation.value) - element.size / 2,
  child: GestureDetector(
    onPanDown: (details) {
      element.isOn = true;
    },
    onPanUpdate: (details) {
      count++;
      temp.x += details.delta.dx;
      temp.y += details.delta.dy;
      if(count>10){
        count = 0;
        setState() {
          graphFlag = false;
          element.prevPosition = Vector2(element.position.x, element.position.y);
          Duration d = controller.forceCalc(frame, 1, 1);
          runGraphAnimation(d);
          element.position.x += temp.x;
          element.position.y += temp.y;
          temp = Vector2(0, 0);
          fillWidg();
        };
      }
    },
    onPanEnd: (details) {
      element.isOn = false;
      setState() {
        fillWidg();
      };
    },
    onTap: () {
      setState() {
        runAnimation(Offset(element.position.x, element.position.y), 0.7);
        context.read<AppProvider>().setFocusNode(controller.balloon.three[
          controller.balloon.three
            .indexWhere((a) => a.id == element.id]));
        context.read<AppProvider>().isEdgeActive = false;
        context.read<AppProvider>().focusTitle = element.fullTitle;
        context.read<AppProvider>().animationStart = false;
        context.read<AppProvider>().setBottomSheetFlag(true);
      };
    },
    onDoubleTap: () {
      setState() {
        runAnimation(Offset(element.position.x, element.position.y), 0.3);
        context.read<AppProvider>().setFocusNode(controller.balloon.three[
          controller.balloon.three
            .indexWhere((a) => a.id == element.id]));
        context.read<AppProvider>().animationStart = false;
      };
    },
    child: Container(
      height: element.size,
      width: element.size,
      decoration: BoxDecoration(
        color: element.sideColor,
        border: Border.all(color: element.mainColor, width: 6),
        borderRadius: BorderRadius.circular(element.size)),
    ),
  ),
));

});
graphFlag = true;
}
//задання логіки при запуску віджета
@override

```

```

void initState() {

  Stopwatch s = Stopwatch();
  s.start();
  final map = context.read<AppProvider>().currentMap;
  map.age++;
  animationController =
//побора анимації
  AnimationController(duration: Duration(microseconds: 200), vsync: this);
  animationController.addListener() {
    setState() {
      transformationController.value = animation.value;
    };
  });

  graphAnimationController =
    AnimationController(vsync: this);
  graphAnimationController.addListener() {
    setState() {
    };
  });
  graphAnimation = animationController
    .drive(CurveTween(curve: Curves.easeInOut))
    .drive(Tween<double>(begin: 0.0, end: 1.0));

  flag = false;
  graphFlag = true;
  context.read<AppProvider>().bottomSheetFlag = null;
  context.read<AppProvider>().animationStart = false;
  count = 10;
  temp = Vector2(0, 0);
  controller = ForceDirectedController(map);
  controller.crToVE();
  frame = Offset(controller.vertices.length.toDouble() * 800,
    controller.vertices.length.toDouble() * 800);
  controller.setVerticesPos(frame);
  controller.forceCalc(frame, 50, 50);
  context.read<AppProvider>().setTree(controller.balloon.three);
  controller.setVerticesEdgesColors(controller.balloon.three);
  fillWidg();
  flag = true;
  force = 50;
  print(s.elapsedMilliseconds);
  super.initState();
}
//силовий алгоритм побудови карт понять
int force;

void d(){
  if(force > 0){
    Duration d = controller.forceCalc(frame, 1, force.toDouble());
    fillWidg();
    runGraphAnimation(d);
  }
  force--;
}

@override
void didChangeDependencies() {
  Stopwatch ss = Stopwatch()
  ..start();

  final size = MediaQuery.of(context).size;
  //d();
  if (Provider.of<AppProvider>(context).animationStart == true) {
    Vector2 v = controller
      .vertices[controller.vertices.indexWhere(
        (a) => a.id == Provider.of<AppProvider>(context).animationId)]
      .position;
    context.read<AppProvider>().isEdgeActive = false;
    context.read<AppProvider>().focusTitle = controller.vertices.firstWhere((a) => a.id == Provider.of<AppProvider>(context).animationId).fullTitle;
    runAnimation(Offset(v.x, v.y), 0.7);
    Provider.of<AppProvider>(context).bottomSheetFlag = true;
    context.read<AppProvider>().focusNode = controller.balloon.three[
      controller.balloon.three.indexWhere(
        (a) => a.id == Provider.of<AppProvider>(context).animationId)];
  }
}

```

```

} else if (Provider.of<AppProvider>(context).bottomSheetFlag == null) {
  context.read<AppProvider>().focusNode = Node(
    "",
    [],
    "",
    controller
      .vertices[controller.vertices.indexWhere(
        (element) => element.id == controller.rootId.toString())]
      .title);

  Vector2 v = Vector2.copy(controller
    .vertices[controller.vertices.indexWhere(
      (element) => element.id == controller.rootId.toString())]
    .position);

  transformationController.value = Matrix4(
    0.2,
    0,
    0,
    0,
    0,
    0.2,
    0,
    0,
    0,
    0,
    0.2,
    0,
    -v.x * 0.2 + size.width / 2,
    -v.y * 0.2 + size.height / 2,
    0,
    1);
}
print(ss.elapsedMilliseconds);
super.didChangeDependencies();
}

@override
void dispose() {
  animationController.dispose();
  graphAnimationController.dispose();
  super.dispose();
}
// верстка віджета, задання меню та нижньої панелі
@override
Widget build(BuildContext context) {
  return Scaffold(
    bottomSheet: BottomSheetPanel(),
    appBar: SearchAppBar(),
    drawer: DrawerMenu(),
    body: Container(
      child: InteractiveViewer(
        constrained: false,
        boundaryMargin: const EdgeInsets.all(double.infinity),
        minScale: 0.01,
        maxScale: 5.6,
        onInteractionUpdate: (a) {},
        transformationController: transformationController,
        child: Container(
          width: frame.dx,
          height: frame.dy,
          child: CustomPaint(
            painter:
              PaintGraph(controller.edges, controller.vertices, flag),
            child: Stack(
              children: [
                ...controller.widgets,
                ...controller.titles,
              ],
            ),
          ),
        ),
      ),
    ));
}

```

chart-area-demo.js

```
import 'package:auto_size_text/auto_size_text.dart';
import 'package:concept_maps/models/dedactic_relations_entities/concept_in_theses.dart';
import 'package:concept_maps/models/general_entities/concept.dart';
import 'package:concept_maps/models/general_entities/thesis.dart';
import 'package:concept_maps/providers/app_provider.dart';
import 'package:concept_maps/views/widgets/dots_indicator.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:concept_maps/views/text_templates/theses_builder.dart';
```

```
class RelatedConcepts extends StatefulWidget {
  @override
  _RelatedConceptsState createState() => _RelatedConceptsState();
}
```

```
class _RelatedConceptsState extends State<RelatedConcepts> {
  //об'явлення та ініціалізація змінних
  List<Concept> concepts = [];
  List<bool> dots = [];
  List<int> compareCinT = [];
  ConceptInTheses currentConceptsInTheses;
  ConceptInTheses nextConceptsInTheses;
  int conceptIndex;
  List<Thesis> thesis = [];
  String thesisData;

  // індикатор активних і неактивних понять
  calculateDots() {
    for (int i = 0; i < concepts.length; i++) {
      if (i == conceptIndex || i == conceptIndex - 1 || i == conceptIndex + 1) {
        dots.add(true);
      } else {
        dots.add(false);
      }
    }
  };
}

//отримання даних про пов'язані тези
Future<ConceptInTheses> fillCurrentCinT() async {
  context
    .read<AppProvider>()
    .fetchConceptInTheses(
      int.tryParse(context.read<AppProvider>().focusNode.id))
    .then((value) => currentConceptsInTheses = value);

  return currentConceptsInTheses;
}

Future<ConceptInTheses> fillNextCinT() async {
  context
    .read<AppProvider>()
    .fetchConceptInTheses(int.tryParse(concepts[conceptIndex].id))
    .then((value) => nextConceptsInTheses = value)
    .then((_) => compare());

  return nextConceptsInTheses;
}

void compare() {
  currentConceptsInTheses.innerReferences.forEach((a) {
    nextConceptsInTheses.innerReferences.forEach((b) {
      if (a.thesisId == b.thesisId) {
        if (!compareCinT.contains(a.thesisId)) {
          compareCinT.add(a.thesisId);
        }
      }
    });
  });
  nextConceptsInTheses.outerReferences.forEach((b) {
    if (a.thesisId == b.thesisId) {
      if (!compareCinT.contains(a.thesisId)) {
        compareCinT.add(a.thesisId);
      }
    }
  })
}
```

```

    });
  });
  currentConceptsInTheses.outerReferences.forEach((a) {
    nextConceptsInTheses.outerReferences.forEach((b) {
      if (a.thesisId == b.thesisId) {
        if (!compareCinT.contains(a.thesisId)) {
          compareCinT.add(a.thesisId);
        }
      }
    });
  });
  nextConceptsInTheses.innerReferences.forEach((b) {
    if (a.thesisId == b.thesisId) {
      if (!compareCinT.contains(a.thesisId)) {
        compareCinT.add(a.thesisId);
      }
    }
  });
});
}
//отримання наповнення тез
Future<String> getThesis() async {
  compareCinT.forEach((element) {
    print(element);
  });
  context
    .read<AppProvider>()
    .fetchTheses(compareCinT)
    .then((value) => thesis = value)
    .then((value) => value.forEach((element) {
      setState(() {
        print(thesisData);
        thesisData += element.data + "\n";
      });
    }));

  return thesisData;
}

Future<String> parseFuture() async {
  await fillCurrentCinT();
  await fillNextCinT();
  await getThesis();

  return thesisData;
}

//отримання пов'язаних з обраних понять та додання їх у список
parseDidacticAfter() {
  FutureBuilder(
    future: context
      .read<AppProvider>()
      .fetchConceptsDidacticAfter(
        int.tryParse(context.read<AppProvider>().focusNode.id))
      .then((a) => setState(() {
        a.forEach((element) {
          if (context.read<AppProvider>().focusNode.id != element.id) {
            concepts.add(element);
          }
        });
      })),
    builder: (BuildContext context, AsyncSnapshot snapshot) {
      return Center(child: CircularProgressIndicator());
    },
  );
}

//логіка при запуску віджету
@override
void initState() {
  conceptIndex = 0;
  super.initState();
}

@override

```

```

void didChangeDependencies() {
    concepts = [];
    thesisData = "";
    parseDidacticAfter();
    super.didChangeDependencies();
}

//задання видимого користувачу віджету
@override
Widget build(BuildContext context) {
    final size = MediaQuery.of(context).size;
    final provider = context.read<AppProvider>();
    return Container(
        margin:
            EdgeInsets.only(left: 0, right: 0),
        child: SingleChildScrollView(
            scrollDirection: Axis.vertical,
            child: Column(
                children: [
                    Container(
                        padding: EdgeInsets.only(top: 10),
                        child: SingleChildScrollView(
                            scrollDirection: Axis.vertical,
                            //надання відповідного оформлення отриманій тези
                            child: FutureBuilder <List<Thesis>>({
                                future: provider.fetchEdgeTheses(int.parse(provider.focusNode.id), concepts[conceptIndex].iid),
                                builder: (context, snapshot) {
                                    if (snapshot.hasData) {

                                        final theses = snapshot.data;
                                        int k = -1;
                                        int difId = theses[0].conceptId;
                                        theses.asMap().forEach((key, value) {
                                            if(value.conceptId != difId)
                                            {
                                                k = key;
                                                difId = value.conceptId;
                                            }
                                        });
                                        final concept1 = provider.currentMap.concepts.firstWhere((a) => a.id == provider.focusNode.id);
                                        final concept2 = provider.currentMap.concepts.firstWhere((a) => a.id == concepts[conceptIndex].id);
                                        List<Thesis> thesesU = [];
                                        List<Thesis> thesesV = [];
                                        if(k == -1){
                                            if(difId.toString() == concept1.id){
                                                thesesV = theses;
                                            }else if(difId.toString() == concept2.id){
                                                thesesU = theses;
                                            }
                                        }else{
                                            thesesV = List.from(theses.sublist(0, k));
                                            thesesU = List.from(theses.sublist(k, theses.length));
                                        }
                                        final uTitle = concept1.concept;
                                        final vTitle = concept2.concept;
                                        return Column(
                                            children: [
                                                Align(
                                                    alignment: Alignment.centerLeft,
                                                    child: Container(
                                                        padding: const EdgeInsets.only(left: 15, right: 15),
                                                        margin: EdgeInsets.only(bottom: 5),
                                                        child: Text(
                                                            thesesU.isEmpty? "" : uTitle+" in "+vTitle+" theses",
                                                            style: GoogleFonts.montserrat(
                                                                fontSize: 18,
                                                                fontWeight: FontWeight.w800,
                                                            ),
                                                        ),
                                                    ),
                                                ),
                                                ),
                                                ),
                                                ),
                                                // вивід тез на екран
                                                ...thesesU.map<Widget>(<
                                                    (thesis) => ThesisViewBuilder(
                                                        thesis: thesis, conceptU: concept1, conceptV: concept2,
                                                    ),
                                                ),

```

```

        ).toList(),
        Align(
            alignment: Alignment.centerLeft,
            child: Container(
                padding: const EdgeInsets.only(left: 15, right: 15),
                margin: EdgeInsets.only(bottom: 5),
                child: Text(
                    thesesV.isEmpty? "" : vTitle+" in "+uTitle+" theses",
                    style: GoogleFonts.montserrat(
                        fontSize: 18,
                        fontWeight: FontWeight.w800,
                    ),
                ),
            ),
        ),
        ...thesesV.map<Widget>{(
            thesis => ThesisViewBuilder(
                thesis: thesis, conceptU: concept1, conceptV: concept2,
            ),
        )}.toList()
    ]
);

}

if (snapshot.hasError) {
    return Container();
    // return Text(
    //   snapshot.error.toString(),
    //   style: TextStyle(color: Colors.red),
    // );
}

return CircularProgressIndicator();
}),
),
height: (size.height - 115) * 0.47,
),
//верстка модуля аналізу відношень
Container(
    child: RadiantGradientMask(
        color1: provider.tree.firstWhere((a) => a.id == concepts[conceptIndex].id).sideColor,
        color2: provider.tree.firstWhere((a) => a.id == provider.focusNode.id).sideColor,
        child: Icon(
            Icons.keyboard_arrow_down_rounded,
            size: 45,
            color: Colors.white,
        ),
    ),
),
),
Container(
    //constraints: BoxConstraints(maxHeight: (size.height - 115) / 2),
    child: Row(
        children: [
            Container(
                margin: EdgeInsets.only(left: 15, right: 15),
                constraints: BoxConstraints(maxWidth: size.width * 0.4),
                padding: EdgeInsets.only(right: size.width * 0.02),
                //margin: EdgeInsets.only(top: size.height * 0.063),
                //color: Colors.red,
                child: AutoSizeText(
                    context.read<AppProvider>().focusNode.title,
                    style: TextStyle(
                        fontSize: 18, fontWeight: FontWeight.bold),
                    maxLines: 3,
                ),
            ),
            Expanded(
                child: Container(
                    height: 4,
                    decoration: BoxDecoration(
                        gradient: LinearGradient(
                            begin: Alignment.topRight,
                            end: Alignment.bottomLeft,
                            colors: [
                                provider.tree.firstWhere((a) => a.id == concepts[conceptIndex].id).sideColor,

```


ДОДАТОК В

Підсистема « Модуль « Публікації »

Опис програмного коду

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_TV71108_21В 3

Аркушів 10

АНОТАЦІЯ

Вебсервіс є навчальним застосунком на базі карт понять з функцією аналізу відношень між поняттями. Основними функціональними можливостями системи є:

- Доступ до курсів.
- Доступ до карти понять обраного курсу.
- Доступ до допоміжної панелі з інформацією про поняття.
- Доступ навігаційних інструментів.
- Доступ до модуля аналізу відношень між поняттями.

Мобільний застосунок розроблено в середовищі розробки Visual Studio, використовуючи мову програмування JavaScript та інструментальні засоби створення UI – фреймворк Node.js. Було використано дані з API навчального веб-порталу semantic-portal.net.

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ.....	74
2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ	75
3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ	76
4. ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ.....	77
5. ВИКЛИК І ЗАВАНТАЖЕННЯ.....	78
6. ВХІДНІ ДАНІ	79
7. ВИХІДНІ ДАНІ.....	80

ЗАГАЛЬНІ ВІДОМОСТІ

Додаток А містить специфікацію вебсервісу.

Додаток Б містить програмний код основних програмних модулів розробленої системи.

Додаток В містить опис основних класів розробленої програмної навчальної системи.

Для коректної роботи застосунку потрібен девайс з встановленою операційною системою. Також застосунком можна користуватися за допомогою емулятора.

Вебсервіс розроблений у середовищі розробки Visual Studio мовою програмування JavaScript, використовуючи фреймворк Node.js.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Вебсервіс навчальний призначений для студентів та має такі функціональні можливості:

1. Можливість входу в систему без авторизації.
2. Можливість вибору курсу.
3. Доступ до карти понять.
4. Доступ до нижньої допоміжної панелі.
5. Доступ до інформації про поняття – тез.
6. Можливість навігації по карті публікацій.
7. Можливість пошуку публікації.
8. Можливість навігації по курсу.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Згідно з архітектурою системи, вебсерві складається з двох головних модулів:

- Модуль побудови публікацій.
- Модуль аналізу відношень між публікаціями.

Модуль побудови публікацій складається з великої кількості класів, що містять сутності понять, їх нащадків та тез, а також допоміжні сутності для роботи силового алгоритму побудови. Також модуль складається з класів, в яких прописаний власне силовий алгоритм побудови.

Модуль аналізу відношень між публікаціями складається з класів, що слугують для отримання даних з API, та класів, що аналізують отримані дані та виводять на екран потрібну користувачу інформацію.

ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для користування застосунком користувач завжди має мати підключення до мережі інтернет для завантаження даних з API навчального порталу. Після відключення від мережі вже завантажені дані зберігаються і користувач має можливість користуватися завантаженою публікацією.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для користування вебсервісом потрібно перейти до нього у будь-якому браузері. Додаток можна використовувати на смартфоні з операційною системою Android. Перейти до нього можна за посиланням .

Для інсталяції програмного забезпечення потрібно запустити завантажений файл. Після завершення інсталяції програма готова до її використання.

ВХІДНІ ДАНІ

Вхідними даними для побудови карт понять в розробленій системі є дані з API навчального порталу semantic-portal.net, такі як id понять та тез, відношення між ними, форми назв та їх класи для візуалізації та виводу інформації користувачу.

Вхідними даними для модуля аналізу відношень є дані з API навчального порталу з розділу *Concept in Thesis*, що дозволяє отримати дані про входження назв понять у тези, що належать іншим поняттям.

ВИХІДНІ ДАНІ

Вихідними даними розробленої системи є доступний список вибору курсів, побудована система публікацій на основі отриманих з API даних, інформація про обрані поняття у вигляді тез, що йому належать, та результат аналізу відношень між публікаціями у вигляді списку з виділеними ключовими словами.

ДОДАТОК Г

Підсистема « Модуль « Публікації »

Публікації

УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_TV71108_21Г 4

Аркушів 2

Київ – 2021

ШЛЯХИ СТВОРЕННЯ ЯКІСНОГО ВЕБСЕРВІСУ

Оптимізація процесів онлайн навчання є актуальною проблемою для закладів та сервісів у сфері освіти. Конкуренція в розробці динамічних інструментів для навчання стає підґрунтям для технологічних пошуків для великих та гнучких корпорацій, соціальних мереж та державних установ.

Сучасні практики створення гнучких вебсервісів дозволяють створювати достатньо якісні інструменти. Змінився й сам підхід до інформації, більше немає потреби зберігати інформацію на індивідуальних носіях, значно важливіше знати де можна знайти потрібну актуальну інформацію в мережі інтернет. Зазначені раніше компанії зможуть ефективно інтегрувати алгоритмічну систем в існуючі маркетингові стратегії лише за наявності інтуїтивно зрозумілого інтерфейсу користувача, що слідує класичним методикам розроблення ПЗ. Навчання існуючого персоналу заданим інструкціям по використанню вебдодатку повинна бути зведена до мінімуму.

Для початку, акумуляція онлайн навчання є витратною в часі й ресурсах задачею, оскільки інформація дисимільована на різних, взаємо не пов'язаних між собою сервісах із відмінними рейтинговими системами, фільтрами для сортування тощо. Також, навіть за наявності системи збору та динамічного відстеження даних постає проблема їх автоматизованого аналізу. Сьогодні таку обробку мануально виконують працівники відділень досліджень. Створення комплексу, що включає в себе як збір, так і аналіз, у формі автоматизованої класифікації, вимагає великого досвіду команди розробників у імплементації алгоритмів обробки природніх мов та машинного навчання. Також важливо щоб сайт містив необхідні функціональні можливості та контент заради якого люди його відвідують. Очевидно, що процес створення такої системи вимагатиме суттєві часові витрати.

Слід зазначити що створення описаної вище системи не є гарантією комерційного успіху та фактом усунення проблем дистанційного навчання, оскільки вимагає мануального адміністрування та аналізу отриманих результатів.

Основною метою дипломного проєкту є створення нового вебсервісу кафедри автоматизації проєктування енергетичних процесів і систем

теплоенергетичного факультету Національного технічного університету України
“Київський політехнічний інститут імені Ігоря Сікорського” .