Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Веб-система комплексного управління робочими завданнями, нотатками та проєктами

Виконав: студент VI курсу, групи СНнм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Вербіцький І.В.
(підпис) (прізвище та ініціали)

Керівник _____ Козбур Г.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Готович В. А
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Лещишин Ю.З.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_28_» __травня__ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Вербіцькому Івану Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Веб-система комплексного управління робочими завданнями, нотатками та проєктами.

Керівник роботи Козбур Галина Володимирівна, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_24_» _листопада_ 2023 року № _4/7-1100_

2. Термін подання студентом завершеної роботи 29 травня 2024р.

3. Вихідні дані до роботи Наукові публікації про розробку SPA-застосунків на MERN-стек. Наукова публікація на тему: «Адаптація систем управління до невизначеності: роль комунікацій, стратегічного планування та оцифрування».

4. Зміст роботи (перелік питань, які потрібно розробити)
ВСТУП. 1 АНАЛІЗ МЕТОДІВ КОНСПЕКТУВАННЯ. 1.1 Конспектування у цифрову епоху. 1.2 Роздаткові матеріали та прописні методи конспектування. 1.3 Цифрові нотатки. 1.4 Дослідження генеративного конспектування. 1.5 Висновки до першого розділу. 2 АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ. 2.1 Постановка задачі. 2.2 Інформаційні системи в контексті веб-застосунків. 2.3 MERN технологічний стек, комплексний підхід до веб-розробки. 2.4 Опис обраних технологій розробки веб-системи. 2.5 Висновки до другого розділу. 3 РОЗРОБКА ВЕБ-ЗАСТОСУНКУ КОМПЛЕКСНОГО УПРАВЛІННЯ РОБОЧИМИ ЗАВДАННЯМИ, НОТАТКАМИ ТА ПРОЄКТАМИ. 3.1 Перехід до рендерингу на стороні сервера. 3.2 Переваги використання Next.js. 3.3 Розробка бази даних для веб-застосунку ведення нотаток та управління проєктами. 3.4 Модель системи для ведення нотаток та управління проєктами. 3.5 Структуризація архітектури проєкту. 3.6 Висновки до третього розділу. 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. 4.1 Розробка раціональної діяльності та створення сприятливих і безпечних умов персоналу. 4.2 Вплив електромагнітного випромінювання комп'ютера на здоров'я користувача. 4.3 Організація цивільного захисту на житлових об`єктаз під час воєнного стану. 4.4 Висновки до четвертого розділу. ВИСНОВКИ. ПЕРЕЛІК ДЖЕРЕЛ. ДОДАТКИ

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульний. 2. Мета. 3. Конспектування у цифрову епоху. 4. Інформаційна система. 5. Аналіз технологій розробки. 6. SSR. 7. Проектування бази даних. 8. Модель системи. 9. Зображення застосунку. 10. Висновки. 11. Дякую за увагу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 24 листопада 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	04.12.2023	*Виконано*
2.	Підбір наукових джерел про розробку веб-застосунку	15.12.2023-31.11.2023	*Виконано*
	для ведення нотаток, та про тему конспектування		
3.	Опрацювання наукових публікацій та збір даних по	15.01.2024-25.02.2024	*Виконано*
	темі роботи		
4.	Виконання дослідження згідно мети кваліфікаційної	26.02.2024-07.04.2024	*Виконано*
	роботи		
5.	Оформлення розділу «Аналіз методів	15.04.2024-18.04.2024	*Виконано*
	конспектування»		
6.	Оформлення розділу «Аналіз технологій розробки	19.04.2024-25.04.2024	*Виконано*
	Веб-застосунку»		
7.	Оформлення розділу «Розробка веб-застосунку	26.04.2024-02.05.2024	*Виконано*
	комплексного управління робочими завданнями,		
	Нотатками та проєктами»		
8.	Виконання завдання до підрозділу «Охорона праці»	03.05.2024-07.05.2024	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	08.05.2024-10.05.2024	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	11.05.2024-14.05.2024	*Виконано*
11.	Нормоконтроль	15.05.2024-16.05.2024	*Виконано*
12.	Перевірка на плагіат	17.05.2024	*Виконано*
13.	Попередній захист кваліфікаційної роботи	21.05.2024	*Виконано*
14.	Захист кваліфікаційної роботи	29.05.2024	

Студент _____ Вербіцький І.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Козбур Г.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Веб-система комплексного управління робочими завданнями, нотатками та проєктами // Кваліфікаційна робота освітнього рівня «Магістр» // Вербіцький Іван Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2024 // С. 72, рис. – 6, табл. – 7, кресл. – 11, додат. – 2, бібліогр. – 25.

Ключові слова: веб-система, цифрові нотатки, управління проєктами, управління завданнями, веб-розробка, комплексне управління.

Кваліфікаційна робота присвячена розробці веб-застосунку для ведення нотаток, управління робочими завданнями та проєктами. Досліджено та проведено аналіз стосовно актуальності створення веб-платформи подібного формату в конкурентному середовищі.

В першому розділі кваліфікаційної роботи розглянуто ключові методи цифрового конспектування. Описані варіанти оптимізації та покращення методик управління проектами в академічному, професійному та дослідницькому середовищах.

В другому розділі кваліфікаційної роботи магістра проведено аналіз технологій розробки веб-системи. Поставлено задачі на створення клієнт-серверного веб-застосунку. Подано різні стеки розробки, здійснено порівняння переваг та недоліків. Описано обрані технології для створення веб-системи.

В третьому розділі кваліфікаційної роботи описано етапи розробки веб-застосунку для ведення нотаток, комплексного управління проєктами та завданнями. Сформульовано архітектуру створеного веб-застосунку та ключові сутності бази даних.

У кваліфікаційна роботі магістра в загальному зібрано комплексний аналіз, проектування та розробку веб-застосунку комплексного управління нотатками, проєктами та робочими завданнями.

ANNOTATION

Web system of comprehensive management of work tasks, notes and projects // The educational level "Master" qualification work // Verbitskyi Ivan // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNnm-61 group // Ternopil, 2024 // P. 72, fig. – 6, tables – 7, posters – 11, annexes – 2, ref. – 25.

Key words: web-based system, digital notes, project management, task management, web development, integrated management.

The qualification work is devoted to the development of a web application for note-taking, work task and project management. The relevance of creating a web-based platform of this format in a competitive environment has been researched and analyzed.

The first chapter of the qualification work discusses the key methods of digital note-taking. Options for optimizing and improving project management techniques in academic, professional, and research environments are described.

The second chapter of the master's thesis analyzes the technologies of web system development. The tasks of creating a client-server web application are set. Different development stacks are presented, and their advantages and disadvantages are compared. The selected technologies for creating a web system are described.

The third chapter of the qualification work describes the stages of developing a web application for note-taking, integrated project and task management. The architecture of the created web application and the key entities of the database are formulated.

The master's thesis contains a comprehensive analysis, design, and development of a web application for integrated note-taking, project, and task management.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SSR (англ. Server Side Rendering) – серверний рендеринг.

API (англ. Application Programming Interface) – це набір визначень взаємодії різнотипного програмного забезпечення.

SQL (англ. Structured Query Language) – декларативна мова програмування для взаємодії користувача з базами даних.

JS (англ. JavaScript) – динамічна, об'єктно-орієнтована, прототипна мова програмування.

DOM (англ. Document Object Model) – Модель побудови документу сторінки веб-застосунка.

MERN – Стек технологій MongoDB, Express, React, NodeJS.

БД – База даних, система для зберігання та організації даних.

СКБД – Система керування базами даних.

СУБД – Система управління базами даних.

ПЗ – Програмне забезпечення.

ПК – Персональний комп’ютер.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному академічному, професійному та дослідницькому середовищах ефективне управління завданнями, нотатками та проєктами є критично важливим для досягнення високої продуктивності та успіху. Розробка веб-системи для інтегрованого управління цими компонентами має важливе значення для студентів, викладачів та дослідників. Цю важливість можна оцінити за кількома ключовими параметрами: технологічним прогресом, потребами користувачів та перевагами інтеграції. Швидкий розвиток веб-технологій суттєво трансформував підходи до управління завданнями та проєктами. Сучасні веб-системи використовують хмарні обчислення, що забезпечують доступ у реальному часі та співпрацю з будь-якої точки, де є Інтернет. Це особливо важливо в академічному та дослідницькому середовищі, де співпраця між різними установами та географічними регіонами є звичайною практикою. Використання адаптивного дизайну і мобільної сумісності гарантує, що користувачі можуть управляти своїми завданнями і проєктами на різних пристроях, що підвищує гнучкість і доступність.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Магістр» є розробка веб-системи, яка інтегрує управління робочими завданнями, нотатками та проєктами для підвищення продуктивності, співпраці та ефективності в академічному та професійному середовищі. Веб-застосунок має на меті розширити обмеження існуючих інструментів шляхом надання централізованої платформи, яка задовольняє потреби студентів, викладачів, дослідників.

Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– проаналізувати стан досліджень в темі конспектування;

– проаналізувати методи конспектування в цифрову епоху;

– виконати порівняння існуючих застосунків для ведення нотаток та управління проєктами;

– розробити веб-застосунок, що буде вдосконаленою версією своїх конкуренти.

Об’єкт дослідження – методи цифрового конспектування, ведення нотаток, управління проєктами та робочими завданнями.

Предмет дослідження – методи розробки веб-застосунку для ведення нотаток, робочих записів та управління проєктами.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у тому, що впроваджено новітні методи ведення нотаток, управління проєктами у веб-системі, що дозволяє формувати переваги традиційного конспектування у цифровому форматі, за рахунок можливості створювати користувачу власні види нотаток.

Практичне значення одержаних результатів. Розроблено базову версію веб-застосунку ведення нотаток, з можливістю подальшого вдосконалення системи за рахунок створення нового функціоналу та розширення варіантів використання для користувача.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на XII Міжнародної науково-практичної конференції молодих учених та студентів «Актуальні задачі сучасних технологій» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 6-7 грудня 2023 року) і на XI Науково-технічній конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 13-14 грудня 2023 року).

Публікації. Основні результати кваліфікаційної роботи опубліковано у двох працях конференції (Див. додатки А, Б).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 25 найменувань та 2 додатків. Загальний обсяг кваліфікаційної роботи складає 80 сторінки, з них 72 сторінки основного тексту, який містить 6 рисунків та 7 таблиць.

1 АНАЛІЗ МЕТОДІВ КОНСПЕКТУВАННЯ

1.1 Конспектування у цифрову епоху

Конспектування є важливим аспектом формального навчання в академічному та професійному середовищі, і особи, які активно ведуть нотатки, зазвичай досягають вищих результатів. Однак на цей процес впливає низка змінних, серед яких структура лекції, знання майбутніх тестів, сприйняття актуальності матеріалу, наявність та типи роздаткових матеріалів. Через ці фактори дослідження щодо найефективніших методів конспектування і того, наскільки важливим є сам процес, залишаються неоднозначними. Конспектування є індивідуальним завданням, і врахування особистих відмінностей у здатності до нього ускладнює інтерпретацію літератури.

Дослідження конспектування триває протягом багатьох років, охоплюючи різноманітні варіанти і перспективи. Основною метою є розуміння факторів, які сприяють підвищенню продуктивності та успішності. Однак через численні варіації, пов'язані з конспектуванням, складно зробити однозначні висновки.

На додаток до традиційних змінних, нові фактори також впливають на процес ведення нотаток. Впровадження планшетів, додатків для нотаток та інших освітніх технологій змінює спосіб, у який фахівці фіксують, сприймають і обробляють інформацію. Цей технологічний прогрес створює нові виклики для викладачів, дослідників і професіоналів у веденні конспектів.

У наступних розділах буде коротко розглянуто основну літературу про конспектування, висвітлено останні тенденції та потенційний вплив мобільних технологій на цей процес, а також запропоновано альтернативні підходи до використання роздаткових матеріалів з акцентом на фактори, що мають відношення до сучасного цифрового середовища.

1.2 Роздаткові матеріали та прописні методи конспектування

Роздаткові матеріали є важливими допоміжними засобами для конспектування та навчання в академічному, професійному та дослідницькому середовищах. Основне питання стосується обсягу інформації та формату роздаткових матеріалів. Питання полягає в тому, чи краще матеріал засвоюється при записуванні лекцій власними словами або за допомогою підготовлених матеріалів, які дозволяють зосередитися на змісті лекції [1].

Здатність самостійно перекладати інформацію, отриману на лекції, в особисто значущі нотатки є важливою частиною навчального процесу. Конспектування сприяє кращому засвоєнню матеріалу, однак це є когнітивно складним процесом [2]. Це може бути пов'язано як з навичками конспектування, так і з манерою викладання лектора, наприклад, якщо він говорить нерозбірливо або занадто швидко. Проблема в тому, що робоча пам'ять може бути перевантажена, що заважає ефективно слухати лекцію, обробляти інформацію та записувати її.

Приймаючи рішення щодо роздаткових матеріалів, викладачі повинні збалансувати переваги самостійного конспектування з обмеженнями когнітивного навантаження. Одна зі стратегій полягає в наданні конспектів лекційного матеріалу замість повного набору слайдів або нотаток викладача. Таким чином, слухачі лекцій отримують когнітивний каркас, з якого можуть відштовхуватися, слухаючи лекцію і записуючи лише ключові деталі. Ті, хто отримує конспекти, демонструють кращі результати, ніж ті, хто конспектує самостійно.

Крім впливу стилю викладання та роздаткових матеріалів на конспектування, існують стандартизовані стратегії ведення нотаток. Раніше, коли лекції були основною формою подачі матеріалу, роздаткові матеріали використовувалися рідше, і акцент робився на запам'ятовуванні інформації. Для допомоги у записуванні лекційного матеріалу було розроблено низку методів конспектування, таких як Процедура формального конспектування,

Корнельський метод, Активний метод Бартуша та Метод дослівного конспектування [3].

Навчання систематичним методам конспектування може бути корисним для кодування та вивчення великих обсягів інформації. Незважаючи на ефективність, ці методи часто ігноруються через зміни у підходах до викладання та динаміці роботи. Останнім часом було проведено небагато досліджень цих методів, але вони можуть знову набувати актуальності в контексті формату "перевернутого класу", де необхідно робити нотатки з позааудиторних записаних лекцій для підготовки до активних занять.

1.3 Цифрові нотатки

1.3.1 Впровадження планшетних комп'ютерів

Впровадження мобільних комп'ютерних пристроїв в академічному, дослідницькому та професійному середовищах створило нові виклики для конспектування. Швидкість, розбірливість та можливість пошуку є трьома ключовими перевагами цифрового запису інформації. Завдяки цим властивостям можна надавати перевагу цифровому конспектуванню перед традиційним – рукописним.

Однак технологічні зміни не завжди мають позитивний вплив [4]. Основною проблемою використання ноутбуків під час лекцій є комп'ютерне відволікання [5]. Спокуса багатозадачності на цих пристроях може суттєво заважати розумінню матеріалу. Стосовно ефективності друкованих нотаток – дослідження Мюллера і Оппенгеймера показали, що ті, хто робив нотатки на ноутбуці, гірше запам'ятовували концептуальний зміст порівняно з тими, хто записував від руки, хоча вони однаково добре відповідали на запитання про фактичну інформацію [6]. Гіпотеза цієї різниці полягає в тому, що ті, хто друкує нотатки, схильні робити довші записи і занотовувати інформацію дослівно, а не перефразовувати. Особи, які робили рукописні нотатки, демонстрували кращі

результати, ніж ті, хто використовував цифрові засоби, як з фактичних, так і з концептуальних питань [7].

Планшетні пристрої значно сприяли переходу до цифрового зберігання нотаток. Однією з переваг планшетів для конспектування є можливість ручного запису нотаток, додавання малюнків та виділення тексту в цифровому форматі. У зв'язку з складністю та схематичністю сучасних курсів та джерел інформації, цифровий формат, який дозволяє як друкувати, так і малювати, може бути надзвичайно корисним [8]. Крім того, планшетні пристрої можуть зменшити або усунути витрати на друк, скоротити обсяг паперу, який потрібно носити з собою, знизити необхідність у транспортуванні важкого та громіздкого ноутбука, а також сприяти організації нотаток [9].

Особи, що користуються планшетами, значно рідше відволікаються на такі чинники, як електронна пошта, Facebook, месенджери або YouTube під час занять, порівняно з особами, що використовують ноутбуки. Планшетні пристрої також привабливі з інституційної точки зору, оскільки вони забезпечують відносно недорогий спосіб впровадження цифрових технологій у навчальний процес. Декілька університетів, зокрема Нотр-Дам, Пеппердайн і Стенфорд, розпочали програми, спрямовані на заохочення використання iPad в аудиторіях.

1.3.2 Застосунки для ведення нотаток

Поява планшетів і смартфонів стимулює розробку мобільних додатків, які забезпечують легкий доступ до різних функцій. Більшість доступної інформації про використання мобільних застосунків базується на дослідженнях у сфері технологій та онлайн-оглядах додатків. Швидка зміна ринку мобільних сервісів зумовлює велику кількість варіантів за функціональністю та ціною.

Вибір додатків для ведення цифрових нотаток у дослідницькому та академічному середовищах значною мірою залежить від рекомендацій колег. Поради, отримані на основі відгуків дослідників у мережі Інтернет, включають додатки StudyBlue Flashcards, Evernote Peek, Dropbox, а також Evernote, Notability і Penultimate для ведення нотаток [10]. Дослідження когнітивних функцій

конспектування свідчать, що привабливість додатків для конспектування залежить від кількох факторів. Щоб навчальні програми були ефективними, вони повинні підтримувати активний процес конспектування і дозволяти попередньо переглядати нотатки. Додатки, що дозволяють користувачами редагувати, узагальнювати та виділяти роздаткові матеріали для активної участі у процесі конспектування, є найбільш корисними.

Дослідження когнітивних функцій конспектування показують, що привабливість додатків для конспектування зумовлена низкою факторів. Для того, щоб робочі програми були ефективними, вони повинні підтримувати активний процес конспектування і дозволяти ефективно переглядати ці нотатки [11]. Застосунок, який дозволяє лише копіювати і вставляти заздалегідь написані нотатки без додавання власних визначень або роз'яснень, набагато менш ефективний, ніж той, який заохочує до написання власних текстів. Хоч дослівні нотатки можуть бути більш точними, але вони не дають переваги "процесу", а отже, зменшують вплив користувацького досвіду. Застосунки, які дозволяють особам використовувати, редагувати, узагальнювати та виділяти роздаткові матеріали таким чином, щоб забезпечити когнітивну участь у процесі конспектування, є найбільш корисними для ведення дослідницької та академічної діяльності [11].

Офіційне використання додатків для конспектування та навчання перебуває на стадії становлення. Одним із прикладів такої ініціативи є Християнська академія Абілена, яка видає кожному своєму студенту iPod Touch або iPhone для використання під час навчання [12]. Основна увага при використанні цих пристроїв приділяється програмам, які покращують досвід ведення нотаток та управління проектами [13]. На додаток до покращення візуального і тактильного професійного процесу, застосунки також можуть покращити процес конспектування, дозволяючи інтегрувати більш складні зображення та інші медіа в письмові посібники студентів. Використання цих пристроїв у класі допомагає навчити учнів користуватися різноманітними додатками та інструментами, які мають вирішальне значення для кар'єри в цифрову епоху.

1.4 Дослідження генеративного конспектування

Генеративне конспектування є важливою стратегією для роботи зі складними текстами, яка вимагає не тільки осмислення та відбору інформації, але й її активного продукування. Це дослідження фокусується на різних форматах конспектування тексту, прочитаного з метою узагальнення, та покращення як практичного, так і теоретичного розуміння цього процесу.

Методологія дослідження – у дослідженні взяли участь 103 студенти другого курсу факультету англійської мови Сілезького університету у Катовіце, Польща. Метою було вивчити, як ці студенти, як читачі іноземних мов, взаємодіють зі складними текстами, які стратегії конспектування використовують і як їх навчали вести нотатки [14].

Загалом, тільки 42 з 103 студентів отримали певну форму інструктажу з техніки ведення паперових нотаток або цифрових додатків, що полегшують процес конспектування.

В таблиці 1.1 наведено розподіл студентів за формою інструктажу з конспектування. Дослідження показало, що учні не змогли назвати більше чотирьох додатків для конспектування, які б сприяли формуванню цілісної інтерпретації прочитаного цифрового тексту [15].

Таблиця 1.1 – Розподіл студентів за формою інструктажу з конспектування

Категорія	**Кількість студентів**
Отримали інструктаж з паперового конспектування	42
Не отримали інструктажу	61

Таблиця демонструє розподіл студентів за різними формами інструктажу, що використовуються в процесі дослідження.

Графік розподілу студентів за формою інструктажу представлено на рисунку 1.1.

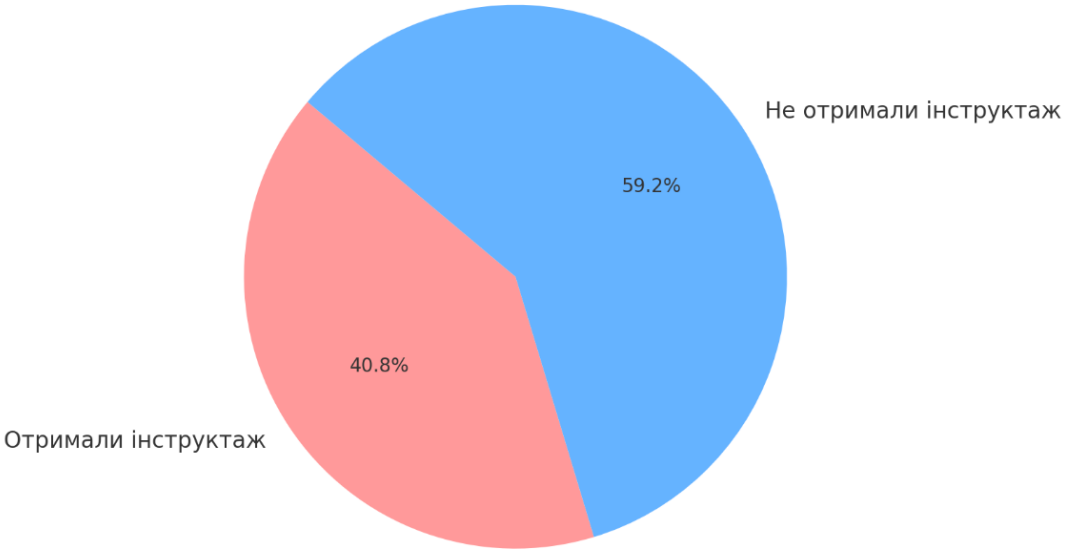

Рисунок 1.1 – Графік розподілу студентів за формою інструктажу

Аналіз також показав, що студенти не повністю перейшли від паперового до цифрового інтерфейсу, оскільки в їхніх читацьких звичках все ще проявляється ефект меншовартості екрану. Це свідчить про те, що перехід на цифрові методи конспектування ще не завершений і потребує подальшого дослідження та адаптації.

Зібрані дані свідчать про необхідність відкритого навчання конспектуванню як у паперовому, так і в цифровому режимі. Такий підхід створить більше можливостей для заохочення студентів, їхньої взаємодії та залучення до читання. Інструкції з конспектування мають бути адаптовані з урахуванням цифрових інструментів, щоб ефективніше використовувати формати конспектування, доступні для обробки текстів з цифровим інтерфейсом.

Дослідження підкреслює важливість інтеграції нових технологій у процес навчання та необхідність адаптації методів конспектування до цифрового середовища. Це може включати розробку нових інструкцій та навчальних програм, що враховують специфіку роботи з цифровими текстами.

1.5 Висновки до першого розділу

У першому розділі кваліфікаційної роботи магістра було досліджено та описано загальні методи ведення конспектів. Розкрито питання конспектування

в цифрову епоху. Створено порівняння традиційних методів ведення нотаток та сучасних методик, які надають цифрові інструменти.

Було описано передумови появи сфери цифрового конспектування. Представлено етапи розвитку технічних засобів, а саме планшетних комп’ютерів та мобільних пристроїв та чинники, що вплинули на створення цифрового конспектування в загальному.

Висвітлено дослідження серед студентів, яке показало готовність до сучасної технічної мобільності у плані ведення нотаток та навчання цьому. Подано графічне представлення результатів дослідження

2 АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ

2.1 Постановка задачі

Потрібно проаналізувати процес створення інформаційної системи веб-застосунку, при цьому використовуючи стек MERN.

Цей додаток ділиться на дві частини – серверну та клієнтську, між якими організована клієнт-серверна архітектура (див. рис. 2.1). Описати її основні засади та характеристики. Дослідити основні положення інформаційних систем та веб-застосунків.

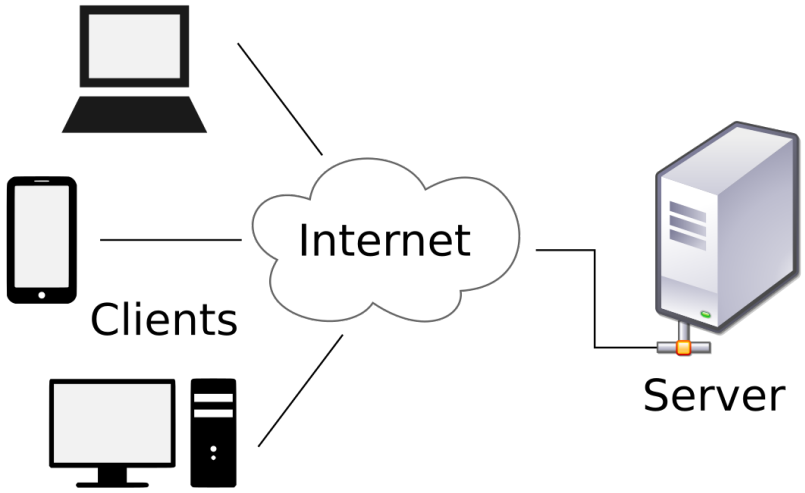

Рисунок 2.1 – Зображення клієнт-серверної архітектури

Етапи технічної розробки веб-застосунку:

- Створити серверну частину.
- Налаштувати роботу NodeJS та Express.
- Розробити структуру бази даних використовуючи MongoDB.
- Розробити інтерфейс використовуючи ReactJS.

Проаналізувати та описати отримані результати дослідження, визначити чому обрані технології найкраще підійдуть для створення інформаційної системи веб-застосунку.

2.2 Інформаційні системи в контексті веб-застосунків

Інформаційні системи є невід'ємною складовою сучасного управління та організації діяльності в різних сферах. Ці системи забезпечують ефективне зберігання, обробку та передачу даних, що сприяє прийняттю обґрунтованих рішень. Особливо важливою є роль інформаційних систем, реалізованих у вигляді веб-застосунків, які пропонують додаткові переваги завдяки використанню Інтернету.

Інформаційні системи являють собою інтегровані комплекси, що забезпечують автоматизацію процесів управління, обробки та аналізу інформації. Основними компонентами таких систем є апаратне забезпечення, програмне забезпечення, бази даних, мережеве обладнання та користувачі.

- Класифікація інформаційних систем може бути здійснена за різними критеріями, такими як:
- Функціональна ознака: управлінські, інформаційно-довідкові, аналітичні системи.
- Рівень управління: стратегічного, тактичного та оперативного рівня.
- Сфера застосування: бізнесові, освітні, медичні, державні тощо.

Для ефективного проектування інформаційних систем необхідно дотримуватися певних принципів:

- Дані, що вводяться в систему, повинні використовуватися багаторазово для вирішення різних завдань. Це дозволяє знизити ймовірність дублювання інформації та підвищити її достовірність.
- Автоматизація всіх процедур перетворення даних на всіх етапах роботи системи сприяє зменшенню витрат часу та ресурсів на обробку інформації.
- Забезпечення комплексного підходу до обробки даних дозволяє отримувати інформацію, необхідну для прийняття управлінських рішень.

Реалізація інформаційних систем у вигляді веб-застосунків відкриває нові можливості для користувачів. Далі описані основні переваги веб-застосунків.

Веб-застосунки доступні з будь-якого пристрою, підключеного до Інтернету, що робить їх зручними для користувачів з різних платформ.

Основою для таких систем є середовище зберігання даних. Використання хмарних рішень для зберігання даних забезпечує масштабованість та гнучкість управління даними.

Веб-застосунки надають інтуїтивно зрозумілі інтерфейси, доступні з будь-якого пристрою, що робить їх зручними навіть для користувачів, які не є експертами в галузі обчислювальної техніки.

Можливість реальної співпраці та обміну даними між користувачами, незалежно від їхнього географічного розташування, є великою перевагою для підприємств з розподіленими командами.

Централізоване оновлення веб-застосунків гарантує, що всі користувачі матимуть доступ до найновіших функцій та покращень безпеки без необхідності встановлювати оновлення вручну.

Інформаційні системи, реалізовані у вигляді веб-застосунків, відіграють важливу роль у сучасному управлінні, забезпечуючи ефективний обмін та обробку інформації. Дотримання принципів інтеграції, комплексності та системності є ключовими для створення ефективних інформаційних систем, що відповідають потребам користувачів і допомагають оптимізувати управлінські процеси. Веб-застосунки пропонують додаткові переваги доступності, співпраці в режимі реального часу та централізованого оновлення, що робить їх незамінним інструментом для підприємств у цифрову епоху.

2.3 MERN технологічний стек, комплексний підхід до веб-розробки

У сфері веб-розробки «стек» означає набір технологій, які використовуються разом для створення та підтримки застосунків. Кожен компонент стеку надає окремі функціональні можливості, які, будучи інтегрованими, дозволяють розробникам створювати надійні та масштабовані додатки. Вибір стеку впливає на продуктивність, масштабованість і ремонтопридатність додатку. Одним з найвідоміших стеків у сучасній веб-розробці є стек MERN, абревіатура від MongoDB, Express.js, React та Node.js.

Стек MERN складається з чотирьох ключових технологій:

- MongoDB – не реляційна база даних, яка використовує гнучкий JSON-подібний формат для зберігання даних.
- Express.js – легкий фреймворк для створення веб-додатків та серверної частини сайту на Node.js.
- React – бібліотека JavaScript для створення користувацьких інтерфейсів, зокрема односторінкових додатків, де дані можуть динамічно змінюватися.
- Node.js – середовище виконання, яке дозволяє запускати JavaScript на стороні сервера.

Візуалізація технології MERN зображена на рисунку 2.2.

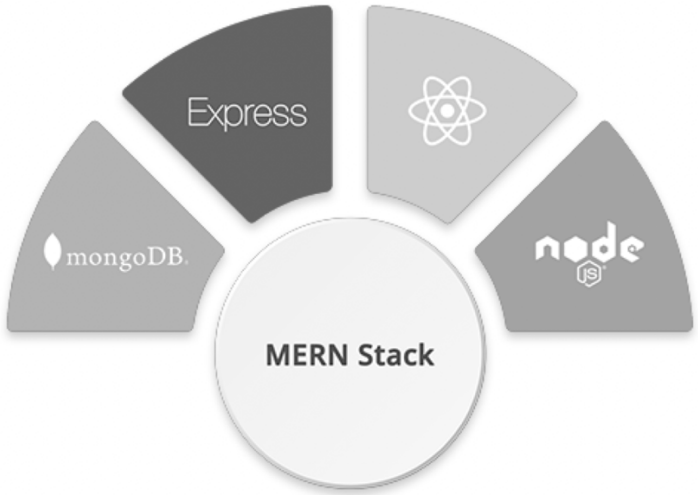

Рисунок 2.2 – Технологія MERN

Кожен з цих компонентів відіграє важливу роль у процесі розробки, дозволяючи створювати ефективні та масштабовані повностекові додатки.

Стек MERN має декілька переваг, які роблять його кращим вибором серед розробників:

Значною перевагою стеку MERN є використання JavaScript протягом усього процесу розробки. Як клієнтський, так і серверний код написані на JavaScript, що спрощує процес розробки. Така уніфікація підвищує продуктивність команди і скорочує час навчання для розробників, які переходять між різними частинами програми.

MongoDB забезпечує гнучкість у зберіганні та обробці даних. Її дизайн без схем дозволяє швидко змінювати структури даних, не вимагаючи значних модифікацій.

Відомий своєю подієво-керованою, неблокуючою моделлю вводу/виводу, Node.js має високу масштабованість і може ефективно обробляти численні одночасні з'єднання. Така масштабованість має важливе значення для додатків, які з часом планують зростати і обробляти великі навантаження.

Стек MERN прискорює розробку завдяки великій кількості готових компонентів і бібліотек. Ці ресурси дозволяють розробникам більше зосередитися на логіці додатку та користувацькому досвіді. Наприклад, компонентна архітектура React сприяє повторному використанню коду та спрощує розробку складних інтерфейсів користувача.

Екосистема навколо стеку MERN включає численні інструменти, які полегшують розробку, розгортання та підтримку. Такі інструменти, як PM2 для Node.js, спрощують розгортання та управління додатками, надаючи такі функції, як моніторинг процесів та автоматичний перезапуск. Такі інструменти значно зменшують операційні витрати і дозволяють розробникам зосередитися на створенні функціоналу.

Стек MERN отримує переваги від великої та активної спільноти розробників. Ця спільнота надає обширну документацію, навчальні посібники та форуми, де розробники можуть звернутися за допомогою та обмінятися знаннями. Активна підтримка спільноти гарантує, що розробники можуть швидко знаходити рішення проблем та бути в курсі найкращих практик та нових розробок.

Стек MERN являє собою комплексний та ефективний підхід до розробки веб-додатків. Використання JavaScript у всьому стеку в поєднанні з гнучкістю MongoDB, надійністю Node.js та потужними можливостями інтерфейсу користувача React роблять його чудовим інструментарієм для розробників. Швидкість розвитку стеку, що підтримується багатим набором інструментів та сильною спільнотою, ще більше підкреслює його популярність та ефективність у створенні високоякісних веб-додатків. Оскільки веб-розробка продовжує

розвиватися, стек MERN залишається ключовим гравцем, дозволяючи розробникам створювати масштабовані, підтримувані та продуктивні додатки.

2.4 Опис обраних технологій розробки веб-системи

2.4.1 Формування списку обраних технологій

У сучасному ландшафті розробки програмного забезпечення вибір технологій відіграє вирішальну роль в ефективності та успіху проекту. Не існує єдиної мови програмування або інструменту, який був би кращим у всіх контекстах; натомість, кожна технологія має унікальні переваги, які найкраще підходять для вирішення конкретних завдань. У цьому розділі буде детально розглянуто технології, обрані для розробки надійної веб-орієнтованої інформаційної системи комплексного управління цифровими нотатками, проєктами та робочими записами. Обраний стек технологій включає Visual Studio Code, JavaScript, React, Node.js, Express, Mongoose, MongoDB та Git. Кожна з цих технологій має свої переваги, які буде розглянуто.

Обрані технології для реалізації веб-системи ведення нотаток, комплексного управління робочими завданнями та проєктами представлені у таблиці 2.1.

Таблиця 2.1 – Обрані засоби реалізації веб-застосунку

№	Засіб
1	Visual Studio Code
2	JavaScript
3	NodeJS
4	ExpressJS
5	MongooseJS
6	MongoDB
7	ReactJS
8	Git

Visual Studio Code (далі VS Code) – це потужний редактор коду з відкритим вихідним кодом, розроблений компанією Microsoft. Він широко відомий завдяки

своїй продуктивності, крос-платформенній підтримці та розгалуженій екосистемі розширень. VS Code підтримує різні мови програмування та надає такі функції, як IntelliSense, налагодження та інтегрований контроль Git, що робить його ідеальним вибором для сучасної веб-розробки.

JavaScript – це універсальна та динамічна мова програмування, яка є фундаментальною для веб-розробки. Вона використовується для створення інтерактивних елементів на веб-сайтах і сумісна з усіма основними браузерами. Завдяки тому, що JavaScript керується подіями і не блокується, вона ідеально підходить для розробки адаптивних користувацьких інтерфейсів і додатків у реальному часі. Широке розповсюдження та велика підтримка спільноти гарантують розробникам доступ до численних ресурсів та бібліотек.

React – це популярна JavaScript-бібліотека для створення користувацьких інтерфейсів, зокрема односторінкових додатків, де дані змінюються з часом. Розроблена компанією Facebook, React дозволяє розробникам створювати багаторазові компоненти інтерфейсу, які керують власним станом. Його віртуальний DOM підвищує продуктивність, оновлюючи лише необхідні частини інтерфейсу, що призводить до швидшого та ефективнішого рендерингу [16].

Node.js – це середовище виконання, яке дозволяє розробникам запускати JavaScript на стороні сервера. Воно побудоване на основі JavaScript-рушія Chrome V8, що робить його високопродуктивним і масштабованим. Node.js використовує модель вводу/виводу, керовану подіями, що не блокується, що робить його придатним для створення масштабованих мережевих додатків, таких як веб-сервери та API. Його здатність обробляти багато паралельних з'єднань з високою пропускною здатністю є значною перевагою для веб-додатків.

Express – це мінімальний і гнучкий фреймворк веб-додатків Node.js, який надає надійний набір функцій для створення веб- і мобільних додатків. Він спрощує процес розробки серверних додатків, надаючи набір утиліт HTTP і проміжного програмного забезпечення. Це дозволяє розробникам створювати надійні API та ефективно обробляти різні HTTP-запити.

Mongoose – це бібліотека об'єктного моделювання даних (ODM) для MongoDB та Node.js. Вона надає просте, засноване на схемах рішення для моделювання даних додатків. Mongoose має вбудовані засоби приведення типів, валідації, створення запитів та хуки бізнес-логіки, які спрощують взаємодію з MongoDB та забезпечують цілісність і узгодженість даних.

MongoDB – це документно-орієнтована NoSQL-база даних, яка зберігає дані у гнучких JSON-подібних документах. Це дозволяє зберігати складні структури даних і підтримує динамічний дизайн схем. Масштабованість і продуктивність MongoDB роблять її кращим вибором для додатків, які потребують зберігання і пошуку великих обсягів даних, таких як системи управління контентом і аналітичні платформи в реальному часі.

Git – це розподілена система контролю версій, яка відстежує зміни у вихідному коді під час розробки програмного забезпечення. Вона дозволяє декільком розробникам одночасно співпрацювати над проектом, зберігаючи історію змін і полегшуючи перегляд та злиття коду. Можливості розгалуження та об'єднання Git забезпечують ефективний робочий процес для управління розробкою функцій, виправленням помилок та випуском релізів.

Поєднання цих технологій – VS Code, JavaScript, React, Node.js, Express, Mongoose, MongoDB та Git створює потужний та ефективний стек розробки. Кожна технологія має унікальні сильні сторони, які, будучи інтегрованими, дозволяють розробляти масштабовані, підтримувані та високопродуктивні веб-додатки. Цей стек технологій підтримує безперервний процес розробки від написання та тестування коду до розгортання та управління додатками у виробництві.

2.4.2 Visual Studio Code, визначення та переваги

Visual Studio Code – засіб для створення і редагування сучасних веб-застосунків. Доступний у версіях для платформ Windows, Linux і Mac OS.

Інтерфейс програми зображений на рисунку 2.3.

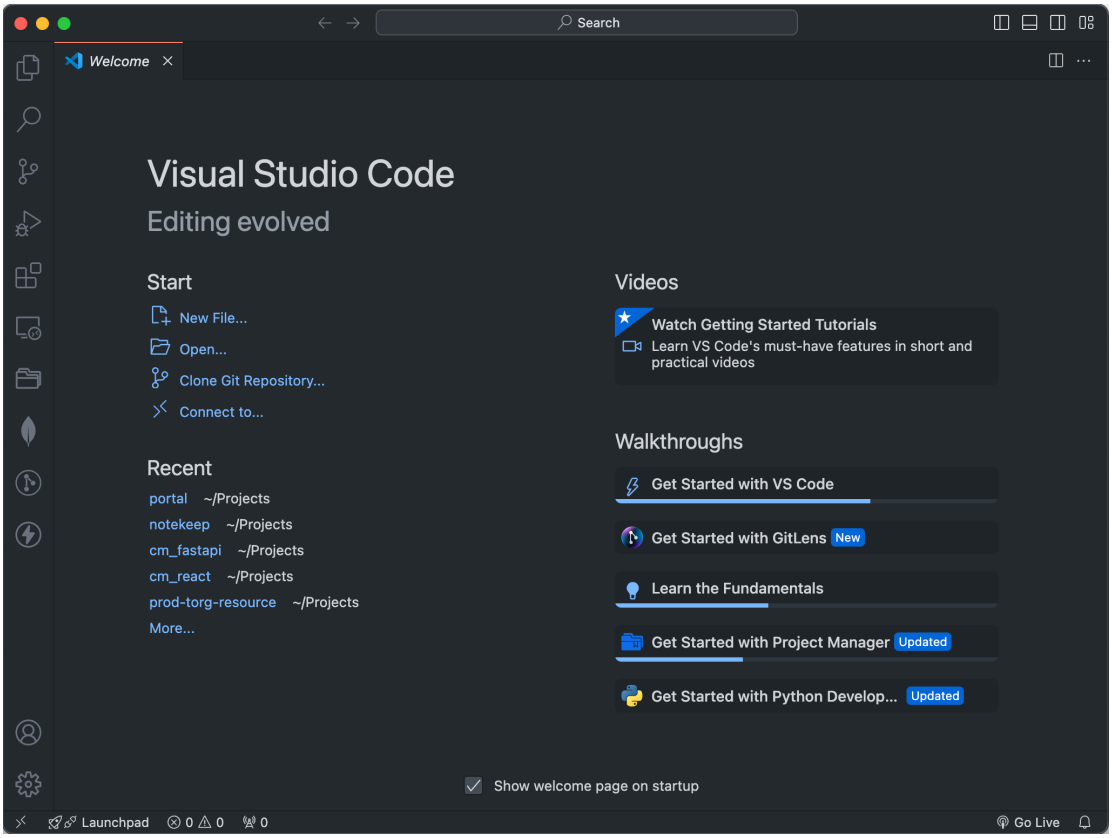

Рисунок 2.3 – Інтерфейс програми Visual Studio Code

Visual Studio Code - це безкоштовний редактор коду, який став популярним серед розробників завдяки своїм функціям та можливостям.

Деякі з переваг Visual Studio Code:

- Кросплатформенність. Редактор підтримує різні операційні системи, такі як Windows, macOS та Linux.
- Розширюваність. Visual Studio Code має велику кількість розширень, що дозволяє розробникам налаштувати редактор під свої потреби.
- Вбудована підтримка Git. Visual Studio Code має вбудовану підтримку системи контролю версій Git, що дозволяє розробникам більш зручно працювати зі своїми проєктами.
- Підтримка різних мов програмування. Visual Studio Code має підтримку багатьох мов програмування, таких як JavaScript, Python, C++, Java, PHP та багато інших.

- Відладка коду. Редактор дозволяє відлажувати код прямо в редакторі, що дозволяє розробникам ефективно вирішувати проблеми в своєму коді.
- Швидкість та продуктивність. Visual Studio Code має швидку роботу та забезпечує розробникам продуктивну роботу зі своїми проєктами.

Ці переваги роблять Visual Studio Code популярним серед розробників. Він є відмінним вибором для тих, хто шукає редактор коду з великою кількістю функцій та можливостей, який є легким у використанні та налаштуванні.

2.4.3 Мова програмування JavaScript

JavaScript – це універсальна мова програмування, яка в основному використовується для веб-розробки. Вона забезпечує динамічні та інтерактивні веб-сторінки, що робить її основою сучасної веб-розробки [17].

JavaScript, часто скорочено JS, є інтерпретованою мовою програмування високого рівня. Спочатку розроблена для сценаріїв на стороні клієнта у веб-браузерах, але зараз вона перетворилася на багатопарадигмальну мову, що підтримує імперативний, функціональний та об’єктно-орієнтований стилі програмування. JavaScript широко використовується для створення інтерактивного та динамічного веб-вмісту [18].

JavaScript дозволяє створювати динамічні та інтерактивні веб-сторінки. Це дає можливість розробникам маніпулювати елементами DOM (англ. Document Object Model), реагуючи на дії користувача в режимі реального часу, не перезавантажуючи всю сторінку. Це покращує взаємодію та залучення користувачів.

JavaScript не обмежується веб-браузерами. Його можна використовувати на стороні сервера (Node.js), для розробки мобільних додатків (React Native, Ionic) і навіть для настільних додатків (Electron). Його універсальність робить його популярним вибором для розробки повного стека.

Має величезну екосистему бібліотек і фреймворків, таких як React, Angular і Vue.js, які спрощують і прискорюють процес розробки. Ці інструменти

пропонують рішення для різноманітних завдань, починаючи від дизайну інтерфейсу користувача до управління станом, сприяючи швидкому розвитку.

Асинхронна природа JavaScript дозволяє виконувати неблокуючі операції вводу/виводу, забезпечуючи ефективну обробку одночасних завдань. Це особливо корисно для веб-додатків, які вимагають оперативності та масштабованості, оскільки запобігає зависанню інтерфейсу користувача під час виконання трудомістких операцій.

JS підтримується всіма основними веб-переглядачами, що робить його кросплатформною мовою. Код, написаний на JavaScript, може працювати на різних операційних системах і пристроях без змін, забезпечуючи широку сумісність.

Виконання JavaScript може відрізнятися в різних веб-браузерах, що призводить до проблем із сумісністю. Розробникам часто доводиться писати додатковий код або використовувати полізаповнення, щоб забезпечити узгоджену поведінку в різних браузерах, що може збільшити час і складність розробки.

Через свою клієнтську природу код JavaScript видимий для користувачів і чутливий до підробки. Це наражає програми на вразливості безпеки, такі як атаки міжсайтового сценарію (XSS) і витоки даних. Належні заходи безпеки, такі як перевірка введених даних і безпечне кодування, є важливими для пом’якшення цих ризиків.

Незважаючи на те, що двигуни JavaScript значно підвищили продуктивність протягом багатьох років, вони все ще можуть мати проблеми з завданнями, що потребують обчислень. Операції, пов’язані з процесором, можуть спричинити вузькі місця продуктивності, особливо у складних веб-додатках. Такі оптимізації, як скорочення коду та кешування, можуть певною мірою пом’якшити ці проблеми.

Однопотокова модель виконання JavaScript може створювати проблеми для пов’язаних із центральним процесором (далі ЦП) завдань і паралельного програмування. Оскільки JavaScript працює в одному циклі подій, важкі обчислення або операції блокування можуть заблокувати потік, що призведе до

погіршення взаємодії з користувачем. Такі методи, як веб-воркери, можуть пом’якшити це обмеження, розвантаживши інтенсивні завдання на окремі потоки.

JavaScript – це мова з динамічним типом, що означає, що типи змінних визначаються під час виконання. Хоча ця гнучкість може бути корисною для швидкого прототипування, вона також може призвести до помилок під час виконання та проблем з налагодженням, особливо у великих кодових базах. TypeScript, наднабір JavaScript, який додає статичний тип, усуває це обмеження, надаючи розширену перевірку типів і підтримку інструментів.

JavaScript – це потужна та універсальна мова програмування, яка широко використовується для веб-розробки. Його переваги включають інтерактивність на стороні клієнта, універсальність, багату екосистему, асинхронне програмування та кросплатформну сумісність. Однак JavaScript також має обмеження, такі як залежність від браузера, ризики безпеки, обмеження продуктивності, однопотокове виконання та відсутність введення. Розуміючи ці переваги та недоліки, розробники можуть ефективно використовувати JavaScript, пом’якшуючи його обмеження в проектах веб-розробки.

2.4.4 Бібліотека React.js

React.js бібліотека JavaScript для створення інтерфейсів користувача, набула широкої популярності у веб-розробці. Буде представлено науковий аналіз переваг React.js, зосереджуючись на його ключових функціях і принципах. Завдяки аналізу його віртуальної DOM (англ. Document Object Model), компонентної архітектури, багаторазового використання, оптимізації продуктивності та підтримки екосистеми ця стаття дає зрозуміти, чому розробники віддають перевагу React.js для сучасних проектів веб-розробки.

React.js, розроблений Facebook, є декларативною, ефективною та гнучкою бібліотекою JavaScript для створення інтерфейсів користувача. Його популярність пояснюється здатністю з легкістю створювати динамічні та інтерактивні веб-додатки.

Однією з відмінних особливостей React.js є його віртуальна DOM. Віртуальний DOM є спрощеним представленням фактичного DOM у пам’яті. Коли в інтерфейс користувача вносяться зміни, React.js оновлює віртуальну DOM, а не безпосередньо маніпулює DOM браузера. Цей підхід мінімізує маніпуляції з DOM і покращує продуктивність завдяки пакетному оновленню та ефективному повторному рендерингу лише необхідних компонентів. Наукові дослідження показали, що ця стратегія значно скорочує час візуалізації та покращує швидкість реагування веб-додатків.

React.js підтримує архітектуру на основі компонентів, де користувальницький інтерфейс розбивається на багаторазово використовувані та складові компоненти. Кожен компонент інкапсулює власну логіку та елементи інтерфейсу користувача, що полегшує модульну розробку та повторне використання коду. Наукові дослідження показали, що компонентний підхід веде до кращої зручності обслуговування, масштабованості та співпраці в проектах програмного забезпечення. Акцент React.js на компонентах узгоджується з усталеними принципами розробки програмного забезпечення, такими як поділ проблем та інкапсуляція, що призводить до більш надійних і зручних для обслуговування кодових баз.

Звертаючись повторно до компонентів, що перевикористовуються, React.js дозволяє розробникам створювати складні інтерфейси користувача з мінімальним дублюванням коду. Компоненти можна легко компонувати для створення нових елементів інтерфейсу користувача або цілих макетів програм. Це сприяє повторному використанню коду та прискорює процес розробки. Орієнтація React.js на багаторазове використання узгоджується з найкращими галузевими практиками розробки програмного забезпечення, покращуючи якість коду та зручність обслуговування.

React.js містить різні оптимізації продуктивності, щоб забезпечити швидке та ефективне відтворення компонентів інтерфейсу користувача. Віртуальний DOM, алгоритм розбіжностей і процес узгодження оптимізують процес рендерингу, мінімізуючи оновлення DOM. Крім того, React.js надає такі

інструменти, як shouldComponentUpdate і PureComponent, для оптимізації відтворення компонентів.

React.js отримує переваги від динамічної екосистеми інструментів, бібліотек і фреймворків, які розширюють його можливості та спрощують завдання розробки. Такі фреймворки, як Next.js і Gatsby.js, покращують React.js такими функціями, як рендеринг на стороні сервера, генерація статичного сайту та маршрутизація. Такі бібліотеки, як Redux і GraphQL, надають рішення для управління станом і отримання даних, які бездоганно інтегруються з React.js. Науковий аналіз показав, що процвітаюча екосистема сприяє інноваціям, співпраці та обміну знаннями в спільноті розробників, сприяючи прийняттю та успіху React.js у веб-розробці [19].

Підсумовуючи, React.js пропонує численні переваги для створення сучасних веб-додатків, що базуються на його віртуальній DOM-компонентній архітектурі, повторному використанні, оптимізації продуктивності та підтримці екосистеми. Наукові дослідження та емпіричні дані підтверджують ефективність React.js у підвищенні продуктивності розробки, якості коду та продуктивності додатків. Використовуючи React.js і його принципи, розробники можуть створювати адаптивні веб-програми, які можна масштабувати та підтримувати, які відповідають вимогам сучасного цифрового середовища.

2.4.5 Технології серверної частини – NodeJS, Express

Node.js і Express.js є ключовими технологіями в екосистемі JavaScript для розробки на стороні сервера. Node.js обговорюється в контексті його неблокуючої моделі вводу-виводу та керованої подіями архітектури, тоді як Express.js аналізується на його роль як мінімалістичної веб-платформи для програм Node.js.

Node.js і Express.js трансформували розробку на стороні сервера за допомогою JavaScript, уможливлюючи створення масштабованих і ефективних веб-додатків. Node.js – це середовище виконання, яке дозволяє запускати

JavaScript на стороні сервера, а Express.js – це фреймворк веб-додатків, створений на основі Node.js.

Node.js розроблено з керованою подіями неблокуючою моделлю вводу-виводу, яка підвищує ефективність і масштабованість. Ця архітектура дозволяє Node.js обробляти декілька з'єднань одночасно, не блокуючи потік виконання, що робить його придатним для додатків із великими обсягами введення-виведення, таких як веб-додатки реального часу, API та мікросервіси.

Node.js використовує двигун Google V8 JavaScript, який компілює код JavaScript у машинний код, що забезпечує швидке виконання. Методи оптимізації двигуна V8, такі як своєчасна компіляція, сприяють високій продуктивності програм Node.js.

Node.js працює в однопоточному циклі подій, який керує асинхронними операціями. Ця конструкція мінімізує перемикання контексту та накладні витрати на ресурси, пов'язані з багатопотоковістю, таким чином зберігаючи низьку затримку та високу пропускну здатність. Наукові тести показали, що Node.js чудово справляється з ефективною обробкою великої кількості одночасних підключень.

Express.js – це мінімалістичний фреймворк веб-додатків для Node.js, який забезпечує основні функції для створення веб-застосунків і мобільних додатків. Його простота та невимушений дизайн дозволяють розробникам структурувати свої програми за потреби, забезпечуючи гнучкість і легкість у використанні.

Express.js використовує архітектуру проміжного програмного забезпечення, де функції проміжного програмного забезпечення можуть обробляти запити та відповіді в конвеєрі. Цей модульний підхід спрощує розробку складних додатків, дозволяючи багаторазово використовувати та компонувати проміжне програмне забезпечення для таких завдань, як журналювання, автентифікація та обробка помилок.

Express.js пропонує надійні можливості маршрутизації та методи утиліти HTTP, полегшуючи обробку різноманітних запитів і відповідей HTTP. Його механізм маршрутизації дозволяє визначати маршрути для різних методів HTTP та шаблонів URL-адрес, що полегшує створення RESTful API.

Node.js і Express.js разом забезпечують потужну платформу для серверної розробки JavaScript. Керована подіями неблокуюча модель вводу-виводу Node.js і однопотоковий цикл подій сприяють її масштабованості та ефективності, тоді як мінімалістична структура та архітектура проміжного ПЗ Express.js спрощують розробку веб-додатків. Разом вони дозволяють створювати високопродуктивні, масштабовані та підтримувані програми на стороні сервера, що відповідають сучасним потребам веб-розробки.

2.4.6 Redux

Redux – це бібліотека керування станом для програм JavaScript, яка широко використовується з React. Він забезпечує передбачуваний контейнер стану, що полегшує централізоване та послідовне керування.

Redux, створена Деном Абрамовим та Ендрю Кларком, є популярною бібліотекою керування станом. Хоча зазвичай використовується з React, він не залежить від фреймворків і може бути інтегрований з іншими бібліотеками чи фреймворками.

Redux підтримує весь стан програми в єдиному незмінному дереві станів або сховищі. Ця централізація спрощує керування станом, полегшуючи відстеження та налагодження змін стану.

Стан у Redux незмінний, тобто його не можна змінити безпосередньо. Зміни до стану вносяться за допомогою дій, які описують тип змін, які необхідно внести. Ця незмінність забезпечує передбачуваність і послідовність.

Redux використовує чисті функції, які називаються редукторами, щоб визначити, як змінюється стан у відповідь на дії. Редуктори приймають поточний стан і дію як вхідні дані та повертають новий. Цей чисто функціональний підхід забезпечує передбачуваність змін стану та відсутність побічних ефектів.

Забезпечуючи суворий односпрямований потік даних і незмінність, Redux гарантує, що зміни стану передбачувані та відстежуються. Така передбачуваність спрощує налагодження та тестування.

Єдине джерело істини, надане Redux, централізує стан програми, полегшуючи керування та підтримку. Ця централізація особливо корисна для великих програм із складною взаємодією стану.

Передбачувані зміни Redux дають змогу розширеним інструментам налагодження, таким як Redux DevTools, які дозволяють розробникам перевіряти кожну зміну стану, подорожувати в часі через історію станів і відтворювати дії, покращуючи досвід розробки.

Redux є дуже гнучким і може використовуватися з різними бібліотеками інтерфейсу користувача та фреймворками поза React, такими як Angular або Vue.js. Його система проміжного програмного забезпечення дає розширення, як асинхронні дії та журналювання.

Redux має сильну спільноту та багату екосистему проміжного програмного забезпечення, інструментів розробника та бібліотек, які розширюють його функціональні можливості та спрощують інтеграцію в різні програми.

Redux надає надійне рішення для керування станом у програмах JavaScript. Основні принципи єдиного джерела істини, незмінності та чистих функцій забезпечують передбачувані та керовані зміни стану. Завдяки перевагам централізованого управління, покращеного налагодження, гнучкості та спільноти підтримки Redux залишається цінним інструментом для розробників, які створюють складні веб-застосунки.

2.4.7 MongoDB

MongoDB – популярна система керування базами даних NoSQL, відома своєю гнучкістю, масштабованістю та простотою використання.

MongoDB – це система керування базою даних, яка зберігає інформацію в гнучких документах, схожих на JSON. Він широко використовується в сучасній веб-розробці через здатність обробляти великі обсяги даних і підтримку динамічних схем.

MongoDB зберігає дані в колекціях документів, подібних до JSON, що забезпечує гнучкий дизайн схем і вкладених структур даних. Цей документоорієнтований підхід спрощує представлення та пошук даних, що робить його придатним для обробки різноманітних типів даних.

MongoDB розроблено для горизонтальної масштабованості, що дозволяє розподілено розгортати на кількох серверах або кластерах. Його функції сегментування та реплікації забезпечують високу доступність і відмовостійкість, забезпечуючи оптимальну продуктивність і масштабованість у міру зростання обсягів даних.

Надає потужні можливості надсилання запитів із підтримкою спеціальних запитів, діапазону та конвеєрів агрегації. Функції індексування оптимізують продуктивність запитів, полегшуючи ефективний пошук і фільтрацію даних.

Динамічна схема MongoDB забезпечує гнучку розробку та ітерацію, оскільки схеми можуть розвиватися з часом, не вимагаючи попередньо визначених структур або міграцій. Ця гнучкість прискорює цикли розробки та полегшує швидке створення прототипів.

MongoDB має динамічну екосистему з вичерпною документацією, навчальними посібниками та ресурсами для розробників. Він пропонує офіційні драйвери для багатьох мов програмування та інтегрується з популярними фреймворками та інструментами, сприяючи широкому застосуванню та підтримці спільноти.

Безсхемний дизайн MongoDB дозволяє динамічно змінювати схему та враховує нові вимоги до даних, зменшуючи складність розробки та забезпечуючи гнучкі методи.

Розподілена архітектура MongoDB і вбудовані можливості шардингу підтримують плавну масштабованість, дозволяючи програмам легко обробляти зростаючі навантаження даних і трафік користувачів.

Ефективне індексування, оптимізація запитів і можливості горизонтального масштабування MongoDB забезпечують високу продуктивність, що робить його придатним для великомасштабних програм із високим навантаженням.

Інтуїтивно зрозуміла мова запитів MongoDB, гнучка схема та вичерпна документація підвищують продуктивність розробника, забезпечуючи швидші цикли розробки та спрощені завдання керування даними.

MongoDB отримує переваги від надійної спільноти розробників, співавторів і користувачів, а також розгалуженої екосистеми бібліотек, інструментів і інтеграцій, що полегшує впровадження та співпрацю в спільноті розробників.

2.5 Висновки до другого розділу

У другому розділі магістерської роботи було здійснено ґрунтовний аналіз технологій для розробки веб-застосунків. У розділі описано кожну технологію, наведено їхні переваги та недоліки, що дозволило зробити обґрунтований вибір найбільш відповідних інструментів для реалізації проєкту.

Також було визначено сутність терміну інформаційної системи, описано технології розробки інформаційних систем.

Визначено поняття стеку технологій, а саме на основі MERN стеку подано поєднання різноманітних технологій, які працюють задля успішного функціонування в комплексі веб-застосунку.

3 РОЗРОБКА ВЕБ-ЗАСТОСУНКУ КОМПЛЕКСНОГО УПРАВЛІННЯ РОБОЧИМИ ЗАВДАННЯМИ, НОТАТКАМИ ТА ПРОЄКТАМИ

3.1 Перехід до рендерингу на стороні сервера

Серверний рендеринг – це популярна техніка у веб-розробці, коли HTML-контент генерується на сервері та надсилається клієнту, замість того, щоб покладатися на клієнтський JavaScript для рендерингу контенту. Next.js, фреймворк React, отримав значну популярність у спільноті розробників завдяки своїй надійній підтримці server-side rendering (далі SSR). У цій статті ми розглянемо переваги та недоліки SSR за допомогою Next.js, надамо огляд самого Next.js та окреслимо етапи розробки повноцінних веб-додатків з використанням цієї технології.

Server-Side Rendering – це технологія веб-розробки, за якої вміст веб-сторінки відображається на сервері, а не в браузері клієнта. Основна перевага SSR полягає в тому, що вона значно покращує взаємодію з користувачем за рахунок швидшого переходу між сторінками та швидкого завантаження. Це дозволяє користувачам взаємодіяти з веб-сторінкою без необхідності чекати на завантаження файлів JavaScript або CSS, створюючи в цілому більш плавний перегляд.

SSR оптимізує сторінки для пошукових систем і соціальних мереж, полегшуючи індексацію. Він також пропонує переваги при використанні з сучасними JavaScript-фреймворками, такими як Vue.js, дозволяючи серверам рендерити компоненти в HTML-рядки, які потім надсилаються безпосередньо в браузер.

Однак важливо не тільки зосереджуватися на перевагах, але й пам'ятати про деякі компроміси порівняно з клієнтським рендерингом (CSR). Впровадження SSR може вимагати складніших налаштувань збірки та більшого навантаження на сервер. Крім того, ефективне кешування може стати проблемою, оскільки HTML кожної сторінки відрізняється.

SSR також може легко інтегруватися з гнучкими системами управління контентом, такими як Sanity, що дозволяє редагувати в режимі реального часу на різних платформах, забезпечуючи при цьому значне підвищення продуктивності в парі.

У життєвому циклі веб-розробки SSR відіграє важливу роль у скороченні часу на створення контенту, пропонуючи уніфіковану ментальну модель для розробників і покращуючи SEO. Цей метод особливо корисний для додатків, де швидка доставка контенту має вирішальне значення.

Однак застосування SSR пов'язане з певними компромісами, які розробники повинні враховувати на етапах проектування та впровадження. Це вносить додаткові міркування, такі як управління збільшеним навантаженням на стороні сервера та вирішення більш складних налаштувань та розгортання. Працюючи з фреймворками, які підтримують як клієнтський рендеринг, так і SSR, розробники повинні ретельно зважити ці фактори, щоб вибрати оптимальний підхід.

При плануванні готових до виробництва додатків використання таких фреймворків у поєднанні з можливостями редагування Sanity в реальному часі на різних платформах може значно підвищити продуктивність без шкоди для користувацького досвіду або SEO-переваг. Таким чином, хоча включення SSR у ваш життєвий цикл розробки може вимагати деяких додаткових кроків на початковому етапі, його переваги часто переважають складнощі, пов'язані з ним.

Впровадження серверного рендерингу (SSR) у процес веб-розробки не є самостійним завданням. Воно часто вимагає певного рівня попередніх знань і розуміння конкретних інструментів і технологій, а також встановлення та налаштування відповідних залежностей.

По-перше, знання фреймворків JavaScript, таких як Vue.js або React, є дуже важливим, оскільки ці бібліотеки часто складають основу додатків, що використовують SSR. Розуміння того, як працюють ці фреймворки, може значно підвищити вашу здатність успішно впроваджувати SSR.

Крім того, вам може знадобитися налаштувати та керувати додатковими залежностями, необхідними для коректної роботи SSR. Не варто забувати, що

вирішення потенційних проблем сумісності між сторонніми бібліотеками та інструментами є ще одним важливим аспектом підготовки до впровадження SSR. Таким чином, розробники повинні бути готові вирішувати будь-які проблеми, що виникають під час цього процесу.

Підсумовуючи, можна сказати, що хоча впровадження SSR може забезпечити значні переваги з точки зору підвищення продуктивності та SEO-оптимізації, воно має свій власний набір передумов і залежностей, які розробники повинні ретельно врахувати, перш ніж розпочати цей шлях.

Вплив рендерингу на стороні сервера (SSR) виходить за рамки простого покращення продуктивності веб-сайту та покращення користувацького досвіду. Він відіграє ключову роль у формуванні цифрового ландшафту, впливаючи на способи доставки контенту, доступу до нього та взаємодії з ним в Інтернеті.

Одним із важливих наслідків SSR є його вплив на SEO. Завдяки рендерингу веб-сторінок на стороні сервера пошукові системи можуть ефективніше індексувати контент, покращуючи таким чином видимість сайту в результатах пошуку. Така оптимізація може призвести до збільшення веб-трафіку і потенційно сприяти зростанню бізнесу.

При інтеграції з гнучкими системами управління контентом, такими як Sanity, SSR може докорінно змінити можливості редагування в режимі реального часу на різних платформах. Така інтеграція не лише підвищує продуктивність, але й дозволяє краще контролювати сторітелінг та узгодженість повідомлень у різних регіонах.

Однак, як уже згадувалося, варто зазначити, що вплив SSR не завжди є універсально позитивним; він супроводжується власним набором викликів, таких як складність в обслуговуванні та потенційні проблеми сумісності зі сторонніми бібліотеками. Ці фактори необхідно враховувати, оцінюючи доцільність використання SSR у процесі веб-розробки.

3.2 Переваги використання Next.js

Next.js - популярний фреймворк React, який дозволяє розробникам створювати багатофункціональні, масштабовані веб-додатки. Він забезпечує потужне поєднання можливостей серверного рендерингу (SSR) та статичної генерації сайтів (SSG), що робить його чудовим вибором для створення динамічних, високопродуктивних веб-сайтів.

Далі буде проведено ключові переваги та особливості обраного фреймворку.

Однією з головних переваг Next.js є його здатність підвищувати продуктивність веб-сайтів. Використовуючи рендеринг на стороні сервера, Next.js мінімізує час завантаження веб-сторінки. Це особливо важливо для великих веб-сайтів зі складним контентом, оскільки забезпечує безперебійну роботу користувачів. Крім того, Next.js включає автоматичне розділення коду, що дозволяє сайту завантажувати лише необхідний JavaScript для кожної сторінки, що ще більше оптимізує продуктивність.

Next.js розроблено з урахуванням пошукової оптимізації (SEO). Завдяки рендерингу на стороні сервера пошукові боти можуть легко сканувати та індексувати вміст сайту, що призводить до кращої видимості на сторінках пошукової видачі. Надаючи клієнту повністю відрендеризовані HTML-сторінки, Next.js гарантує, що вміст веб-сайту буде легко доступний пошуковим системам, що підвищує його шанси на вищі позиції в результатах пошуку. Це важливий аспект для веб-застосунку управління робочими завданнями та нотатками.

Next.js спрощує процес розробки, забезпечуючи впорядкований робочий процес. Вбудована підтримка React-компонентів та маршрутизації дозволяє легко створювати та керувати інтерфейсом веб-сайту. Крім того, Next.js підтримує гарячу заміну модулів, що дозволяє розробникам миттєво бачити зміни без перезавантаження всієї сторінки. Next.js також містить вбудований сервер розробки, що усуває необхідність додаткового налаштування сервера.

Розгортання додатків Next.js також є простим. Його можна розмістити на будь-якому сумісному сервері, при цьому доступні як варіанти рендерингу на

стороні сервера, так і статичної генерації сайту. Next.js автоматично оптимізує веб-сайт для виробництва, використовуючи передові методи, такі як розділення коду та стиснення, щоб забезпечити швидке завантаження.

Next.js пропонує чудовий досвід для розробників, що робить його привабливим вибором для веб-розробки. Він надає ряд функцій, які спрощують виконання типових завдань, таких як маршрутизація URL-адрес, стилізація та отримання даних. Next.js також підтримує TypeScript з коробки, що дозволяє розробникам писати безпечний за типом код і відловлювати помилки під час компіляції. Документація фреймворку обширна і містить вичерпні приклади, які допоможуть розробникам швидко розпочати роботу.

Next.js - це універсальний фреймворк, який можна використовувати для різних типів веб-додатків. Незалежно від того, чи створюєте ви простий блог, чи складну платформу для електронної комерції, Next.js забезпечує гнучкість, щоб задовольнити ваші вимоги. Він підтримує інтеграцію різних баз даних, API-сервісів та сторонніх бібліотек, що дозволяє створювати надійні та масштабовані додатки. Завдяки можливостям рендерингу на стороні сервера та статичної генерації сайтів, Next.js може обробляти великі обсяги трафіку без шкоди для продуктивності.

Next.js - це неймовірно потужний фреймворк, який поєднує в собі найкраще з серверного рендерингу та статичної генерації сайтів. Серед його переваг - покращена продуктивність, SEO-дружність, простота розробки та розгортання, чудовий досвід розробників, а також можливість працювати з універсальними та масштабованими проєктами. Використовуючи Next.js, розробники можуть створювати привабливі веб-додатки, які забезпечують винятковий користувацький досвід, зберігаючи при цьому оптимальну продуктивність.

3.3 Розробка бази даних для веб-застосунку ведення нотаток та управління проєктами

Розробка ефективної бази даних є ключовим аспектом створення веб-застосунку для ведення нотаток та управління проєктами. Цей розділ описує процес проектування та реалізації бази даних, яка забезпечить надійне зберігання та ефективний доступ до даних користувачів, нотаток, завдань та проєктів.

Перед початком проектування бази даних необхідно визначити основні вимоги до неї:

- Зберігання користувачів: інформація про користувачів, включаючи імена, електронні пошти, паролі (захищені хешуванням) та інші особисті дані.
- Нотатки: текстові та мультимедійні нотатки, створені користувачами, з можливістю організації їх за категоріями та тегами.
- Завдання та проєкти: структури для управління завданнями та проєктами, включаючи дедлайни, статуси, пріоритети та взаємозв'язки між завданнями та проєктами.
- Спільна робота: можливість спільного доступу до нотаток та проєктів між користувачами, з контролем доступу та правами редагування.
- Історія змін: зберігання історії змін для відстеження редагувань та повернення до попередніх версій.

Для даного проєкту було обрано реляційну СУБД PostgreSQL, зважаючи на її наступні переваги:

- Підтримка складних запитів та транзакцій: PostgreSQL забезпечує високу продуктивність при обробці складних запитів та транзакцій.
- Надійність та масштабованість: PostgreSQL відома своєю надійністю та здатністю обробляти великі обсяги даних.
- Підтримка розширень: можливість використання різних розширень для додаткової функціональності.

Проектування бази даних включає визначення основних таблиць, їх полів та взаємозв'язків між ними.

У таблиці 3.1 представлено вигляд сутності «Користувач», її поля, типи даних та опис.

Таблиця 3.1 – Таблиця «Users» (користувачі)

Поле	Тип даних	Опис
id	SERIAL PRIMARY KEY	Унікальний ідентифікатор
name	VARCHAR(100)	Ім'я користувача
email	VARCHAR(100) UNIQUE	Електронна пошта
password	VARCHAR(255)	Хешований пароль
created_at	TIMESTAMP	Дата створення запису

У таблиці 3.2 представлено вигляд сутності «Нотатки», її поля, типи даних та опис.

Таблиця 3.2 – Таблиця «Notes» (нотатки)

Поле	Тип даних	Опис
id	SERIAL PRIMARY KEY	Унікальний ідентифікатор
user_id	INTEGER REFERENCES Users(id)	Ідентифікатор користувача
title	VARCHAR(200)	Назва нотатки
content	TEXT	Вміст нотатки
tags	VARCHAR(255)	Теги, розділені комами
created_at	TIMESTAMP	Дата створення запису
updated_at	TIMESTAMP	Дата останнього оновлення

У таблиці 3.3 представлено вигляд сутності «Projects», її поля, типи даних та опис.

Таблиця 3.3 – Таблиця «Projects» (проєкти)

Поле	Тип даних	Опис
id	SERIAL PRIMARY KEY	Унікальний ідентифікатор
user_id	INTEGER REFERENCES Users(id)	Ідентифікатор користувача
name	VARCHAR(200)	Назва проєкту
description	TEXT	Опис проєкту
created_at	TIMESTAMP	Дата створення запису
updated_at	TIMESTAMP	Дата останнього оновлення

У таблиці 3.4 представлено вигляд сутності «Завдання», її поля, типи даних та опис.

Таблиця 3.4 – Таблиця «Tasks» (завдання)

Поле	Тип даних	Опис
id	SERIAL PRIMARY KEY	Унікальний ідентифікатор
project_id	INTEGER REFERENCES Projects(id)	Ідентифікатор проєкту
title	VARCHAR(200)	Назва завдання
description	TEXT	Опис завдання
status	VARCHAR(50)	Статус завдання (наприклад, "виконується", "завершено")
priority	INTEGER	Пріоритет завдання
due_date	TIMESTAMP	Термін виконання
created_at	TIMESTAMP	Дата створення запису
updated_at	TIMESTAMP	Дата останнього оновлення

У таблиці 3.5 представлено вигляд сутності «Співпраця», її поля, типи даних та опис.

Таблиця 3.5 – Таблиця «Collaborations» (співпраця)

Поле	Тип даних	Опис
id	SERIAL PRIMARY KEY	Унікальний ідентифікатор
note_id	INTEGER REFERENCES Notes(id)	Ідентифікатор нотатки
user_id	INTEGER REFERENCES Users(id)	Ідентифікатор користувача
permission	VARCHAR(50)	Права доступу (наприклад, "читання", "редагування")
created_at	TIMESTAMP	Дата створення запису

Взаємозв'язки між таблицями:

- Користувачі та нотатки: Один користувач може мати багато нотаток, але кожна нотатка належить лише одному користувачу. Це виражається зв'язком один-до-багатьох між таблицями Users та Notes.
- Користувачі та проєкти: Один користувач може мати багато проєктів, але кожен проєкт належить лише одному користувачу. Це також зв'язок один-до-багатьох між таблицями Users та Projects.
- Проєкти та завдання: Один проєкт може містити багато завдань, але кожне завдання належить лише одному проєкту. Це зв'язок один-до-багатьох між таблицями Projects та Tasks.
- Нотатки та співпраця: Одну нотатку можуть редагувати або переглядати декілька користувачів, і один користувач може мати доступ до багатьох нотаток. Це виражається зв'язком багато-до-багатьох між таблицями Notes та Users, реалізованим через таблицю Collaborations.

Після проектування структури бази даних необхідно реалізувати її, створивши відповідні таблиці та налаштувавши необхідні індекси для покращення продуктивності запитів. У лістингу 3.1 приклади SQL-запитів для створення таблиць.

Лістинг 3.1 – Код SQL-запитів для створення таблиць

```
CREATE TABLE Users (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100),
  email VARCHAR(100) UNIQUE,
  password VARCHAR(255),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Розробка бази даних для веб-застосунку ведення нотаток та управління проєктами є критичним етапом, який забезпечує надійне зберігання даних та ефективний доступ до них.

3.4 Модель системи для ведення нотаток та управління проєктами

Розробка моделі системи для веб-застосунку ведення нотаток та управління проєктами включає опис архітектури та визначення основних функціональних компонентів. У цьому розділі ми опишемо модель такої системи, включаючи варіанти використання та взаємодію компонентів.

Архітектура системи передбачає розподіл на фронтенд та бекенд частини, що взаємодіють з базою даних. Ми використовуємо фреймворк Next.js для реалізації фронтенду та бекенду з підтримкою Server Side Rendering (SSR). База даних реалізується на основі PostgreSQL.

Основні компоненти системи:

Фронтенд: інтерфейс користувача, розроблений з використанням React і Tailwind CSS для швидкого та зручного створення адаптивних інтерфейсів. Next.js для рендерингу сторінок на сервері (SSR) та клієнтської частини (CSR).

Бекенд: серверна частина, розроблена на Node.js з використанням Express.js для обробки запитів. Next.js API маршрути для реалізації логіки бізнесу та обробки запитів до бази даних.

База даних: реляційна база даних PostgreSQL для зберігання даних користувачів, нотаток, проєктів та завдань. Використання ORM (Object-Relational Mapping), такого як Prisma або Sequelize, для спрощення взаємодії з базою даних.

Модель даних системи включає наступні основні сутності:

- Користувачі (Users)
- Нотатки (Notes)
- Проєкти (Projects)
- Завдання (Tasks)
- Співпраця (Collaborations)

Взаємозв'язки між сутностями:

- Користувачі можуть мати багато нотаток та проєктів.
- Проєкти можуть містити багато завдань.
- Нотатки можуть мати багато співавторів через сутність співпраця.

Для кращого розуміння можливостей та функціоналу системи розглянемо кілька типових варіантів використання.

Реєстрація та аутентифікація користувачів

Реєстрація: Новий користувач може створити обліковий запис, надавши необхідну інформацію (ім'я, електронна пошта, пароль).

Аутентифікація: Існуючий користувач може увійти в систему, надавши свої облікові дані.

Взаємодія компонентів: користувач заповнює форму реєстрації на фронтенді. Фронтенд надсилає запит до API на бекенді. Бекенд перевіряє дані та створює запис у таблиці Users. Користувач отримує підтвердження реєстрації та може увійти в систему.

Створення та редагування нотаток

Створення нотаток: Користувач може створити нову нотатку, надавши заголовок та вміст.

Редагування нотаток: Користувач може редагувати існуючу нотатку, змінюючи її заголовок або вміст.

Взаємодія компонентів: користувач заповнює форму створення або редагування нотатки на фронтенді. Фронтенд надсилає запит до API на бекенді з необхідними даними. Бекенд обробляє запит, вносить зміни у базу даних та повертає оновлені дані. Фронтенд оновлює інтерфейс відповідно до отриманих даних.

Створення та управління проєктами

Створення проєктів: Користувач може створити новий проєкт, вказавши його назву та опис.

Управління проєктами: Користувач може додавати завдання до проєкту, змінювати їх статус та пріоритет.

Взаємодія компонентів: користувач заповнює форму створення проєкту на фронтенді. Фронтенд надсилає запит до API на бекенді з даними проєкту. Бекенд обробляє запит, створює запис у таблиці Projects та повертає підтвердження. Користувач може додавати завдання до проєкту через інтерфейс фронтенду.

Фронтенд надсилає відповідні запити до API для додавання або редагування завдань. Бекенд обробляє запити та оновлює дані у базі даних.

Спільна робота над нотатками

Надання доступу до нотаток: Користувач може надати іншим користувачам доступ до своїх нотаток з правами читання або редагування.

Спільне редагування: Користувачі можуть одночасно працювати над однією нотаткою, бачити зміни в реальному часі.

Взаємодія компонентів: користувач вибирає нотатку для надання доступу та вказує користувачів, яким хоче надати доступ. Фронтенд надсилає запит до API на бекенді для створення запису в таблиці Collaborations. Бекенд обробляє запит та створює відповідний запис у базі даних. Запрошені користувачі отримують сповіщення та можуть переглядати або редагувати нотатку залежно від наданих прав. Всі зміни в нотатці синхронізуються в реальному часі за допомогою WebSockets або іншої технології для забезпечення спільного редагування.

Розробка моделі системи для ведення нотаток та управління проєктами включає визначення архітектури, моделі даних та варіантів використання. Вибір реляційної бази даних PostgreSQL та фреймворку Next.js для фронтенду та бекенду забезпечує надійне та ефективне функціонування системи. Реалізація функціональності, такої як реєстрація користувачів, створення та редагування нотаток і проєктів, а також спільна робота над нотатками, дозволяє створити потужний інструмент для організації особистих та командних завдань.

Графічне зображення моделі системи для ведення нотаток та управління проєктами зображено на рисунку 3.1. Ця ER-діаграма ілюструє основні сутності та їх взаємозв'язки:

- Users (Користувачі) мають багато Notes (Нотаток).
- Users (Користувачі) мають багато Projects (Проєктів).
- Projects (Проєкти) містять багато Tasks (Завдань).
- Notes (Нотатки) можуть бути спільними з іншими користувачами через Collaborations (Співпраця).

• Users (Користувачі) можуть співпрацювати над нотатками через Collaborations (Співпраця).

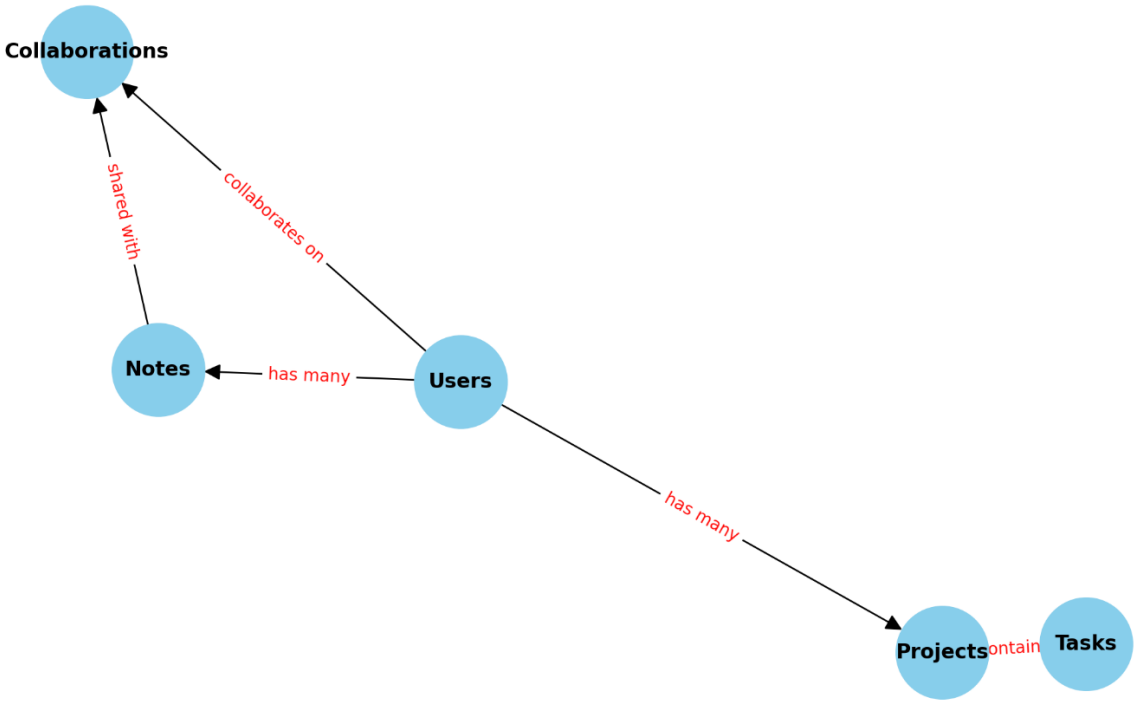

Рисунок 3.1 – Модель системи для ведення нотаток та управління проєктами

Пояснення діаграми:

• Users (Користувачі): зберігають інформацію про користувачів.

• Notes (Нотатки): містять текстові та мультимедійні нотатки користувачів.

• Projects (Проєкти): організовують роботу користувачів у більші завдання.

• Tasks (Завдання): конкретні завдання в межах проєктів.

• Collaborations (Співпраця): забезпечують спільний доступ до нотаток між користувачами.

Ця діаграма допомагає візуально зрозуміти структуру бази даних та взаємодію між різними компонентами системи.

3.5 Структуризація архітектури проєкту

В останні роки Next.js стає все більш популярним в екосистемі React. Зараз це де-факто фреймворк для створення React-додатків.

Популярність цього фреймворку значною мірою пояснюється такими можливостями, як першокласна підтримка широкого спектру інструментів та бібліотек, використання модулів CSS для стилізації, використання TypeScript для безпеки типів, оптимізації зображень та багато іншого. Ці можливості дозволяють створювати масштабовані додатки.

Отже, опишемо, як з нуля створити архітектуру Next.js додатку, який буде масштабуватися без проблем у міру зростання проєкту.

Почнемо з установки проєкту Next.js. У терміналі потрібно ввести наступну команду:

```
npx create-next-app --typescript notekeep
```

Буде створено шаблон Next.js з назвою notekeep. Використання TypeScript у цьому Next.js-додатку завжди є гарною ідеєю, оскільки це забезпечує кращу безпеку типів - те, що повинні мати при створенні сучасного, масштабованого, повностекового додатку.

У цьому прикладі було використано npm як менеджер пакетів. Можна використовувати yarn/pnpm відповідно до потреб.

Після успішного встановлення відкриємо nextjs-архітектуру у VS Code або будь-якій іншій IDE і виконуємо команду:

```
npm run start
```

Ця команда запустить локальний сервер розробки для nextjs-архітектури. За замовчуванням Next.js обслуговує свій додаток на порту 3000. Тому потрібно перейти за адресою localhost:3000, щоб побачити, що шаблонний код успішно працює.

Цей проєкт Next.js використовує версію 14.1.0 і на локальному сервері буде виглядати так, як зображено на рисунку 3.2

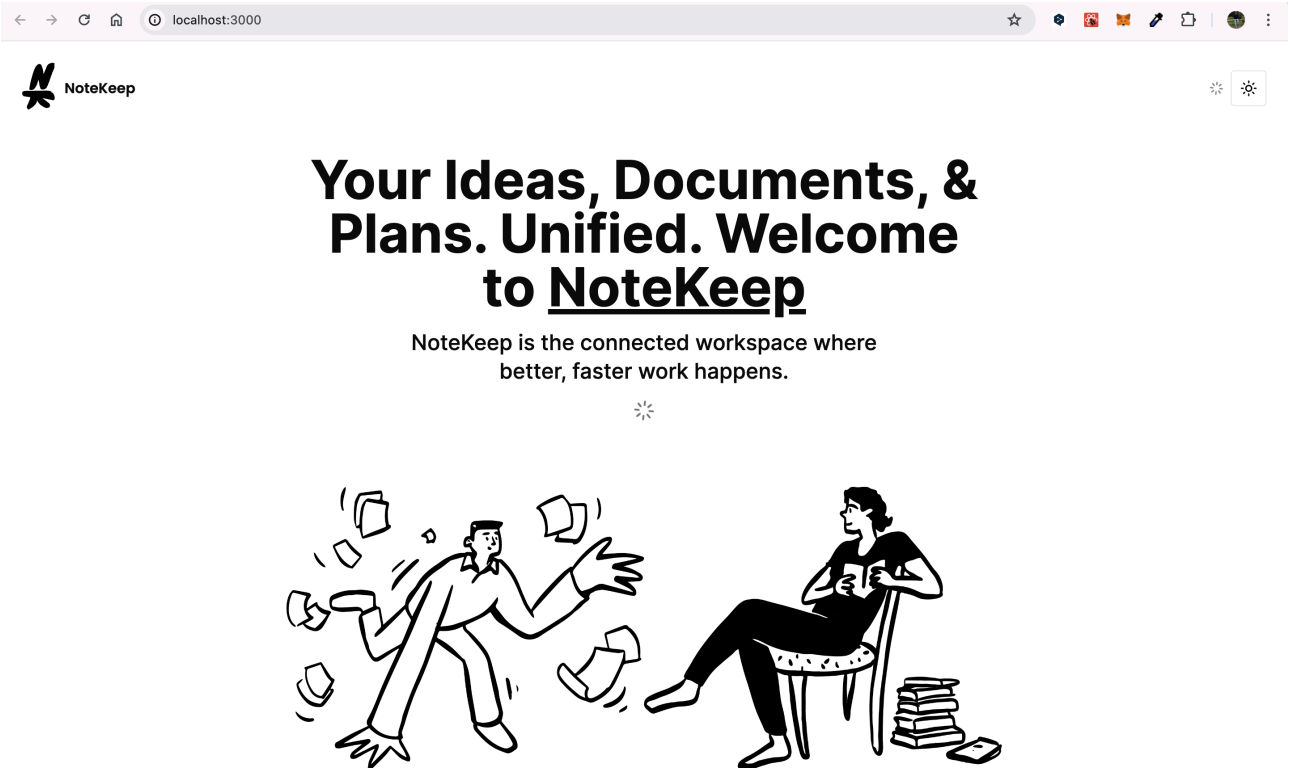

Рисунок 3.2 – Вигляд початкового вікна застосунку

Якщо у проєкті беруть участь кілька членів команди, то повинні переконатися, що всі вони використовують одну і ту ж версію Node.js. Використання різних версій призведе до неузгодженості версій пакунків і згодом може спричинити помилки.

Щоб запобігти будь-яким проблемам, викликаним неузгодженістю між різними версіями Node, можемо додати файл .nvmrc на кореневому рівні проєкту і додати номер версії Node.js, яку хочемо, щоб цей проєкт використовував. У цьому випадку Node встановлено на v18: 18.17.0

Також слід налаштувати менеджер пакунків - в даному випадку npm - для суворого управління використанням залежностей для членів команди. Створено новий файл з назвою npmrc і додано до нього наступний код:

```
engine-strict=true
```

Тепер потрібно перейти до файлу package.json і додати нову пару ключ-значення:

```
"engines": { "node": ">=18.17.0", "npm": "please-use-npm" },
```

Це гарантує, що для запуску проєкту буде потрібно Node.js версії 16 і вище, і в цьому випадку також змусить використовувати npm як менеджер пакунків. Встановлення пакунків за допомогою yarn/pnpm призведе до помилки у цьому проєкті

Ці заходи допоможуть забезпечити сумісність і стабільність при масштабуванні, коли Next.js проєкт зростатиме і змінюватиметься.

На цьому етапі було створено базовий шаблонний код для Next.js, а також налаштовано деякі конфігурації. Зараз доречно додати систему контролю версій для спільної роботи з іншими членами команди над проєктом.

Репозиторій контролю версій є важливим для будь-якого проєкту, особливо для тих, які планується масштабувати. Він дозволяє розробникам легко відстежувати зміни, керувати версіями проєкту та повертатися до попередніх версій за необхідності, що полегшує співпрацю з членами команди та максимізує час безвідмовної роботи програми.

Для цього проєкту буде використовуватися GitHub, але можна також обрати GitLab, Bitbucket або інші платформи для розміщення репозиторію контролю версій відповідно до потреб проєкту.

Спочатку необхідно додати SHA-ключ GitHub на локальну машину. Потім слід створити порожній репозиторій на GitHub і назвати його nextjs-architecture.

Необхідно перейти до терміналу VS Code і внести зміни до порожнього репозиторію за допомогою наступних команд:

```
git add .
git commit -m "initial commit"
```

Для першого коміту може знадобитися перенести його до віддаленого сховища за допомогою таких команд:

```
git remote add origin git@github.com:<name>/nextjs-architecture.git
git branch -M main
git push -u origin main
```

Перед штовханням коміту необхідно переконатися, що у проєкті додано файл .gitignore на кореневому рівні. За замовчуванням Next.js надає шаблонний код.

Файл .gitignore гарантує, що певні папки, такі як node_modules, або файли, що містять ключі безпеки або змінні оточення, не будуть випадково включені в репозиторій. Ці компоненти генеруються "на льоту" під час виштовхування та розгортання додатку на хостинговому сервері у виробничому середовищі.

Можна також скопіювати три вищезгадані команди з самого GitHub під час ініціалізації порожнього сховища. Для перевірки успішного перенесення змін необхідно перейти до репозиторію GitHub і оновити сторінку, де мають відобразитися внесені зміни.

Тепер можна перейти до визначення архітектури коду додатку. Ані React, ані Next.js не нав'язують конкретної структури додатку. Однак, враховуючи файлову маршрутизацію Next.js, доцільно дотримуватися відповідної структури.

Можна почати зі створення такої структури каталогів, як зображено у лістингу 3.2.

Лістинг 3.2 – Вигляд структури застосунку

```
src >
  > app
  > components
  > utils
  > hooks
```

Тека “app” відповідатиме за створення файлової маршрутизації у додатку.

Папка “components” призначена для зберігання файлів компонентів на основі React, таких як компоненти карток, слайдерів, вкладок тощо.

Тека “utils” може використовуватися для різноманітних багаторазових ресурсів, як-от бібліотеки, фіктивні дані або утиліти.

У папці “hooks” можна створювати кастомні хуки React, які не надаються з коробки.

Створення всіх підпапок всередині теки “src” не є обов’язковим. Можна розмістити підпапки безпосередньо в кореневому каталозі. Під час встановлення нового додатку Next.js пропонує обрати структуру на основі “src” — це рішення залежить від особистих уподобань.

3.6 Висновки до третього розділу

В третьому розділі кваліфікаційної роботи освітнього рівня «Магістр» було детально розглянуто реалізацію веб-застосунку для комплексного управління нотатками, проєктами та робочими записами. В цьому розділі висвітлено наступні аспекти.

Архітектура застосунку. Було представлено загальну архітектуру застосунку, яка забезпечує ефективну взаємодію між різними його компонентами. Показано, як фронтенд, бекенд та база даних взаємодіють між собою для забезпечення користувачів зручним та інтуїтивно зрозумілим інтерфейсом.

Побудова бази даних. Описано структуру бази даних, включаючи її сутності та взаємозв'язки між ними. Було створено модель даних, яка підтримує зберігання та обробку інформації про нотатки, проєкти та робочі записи, забезпечуючи при цьому цілісність та узгодженість даних.

Варіанти використання застосунку. Розглянуто різні сценарії використання веб-застосунку, які демонструють його функціональні можливості [20]. Описано, як користувачі можуть створювати, редагувати та видаляти нотатки, організовувати проєкти та відстежувати робочі записи, а також як ці функції сприяють підвищенню продуктивності та організації робочого процесу.

Модель застосунку. Представлено модель застосунку, яка забезпечує модульність та масштабованість системи. Було показано, як різні компоненти

застосунку інтегруються між собою, забезпечуючи гнучкість та можливість подальшого розширення функціональності.

Таким чином, третій розділ магістерської роботи надав вичерпний огляд процесу розробки веб-застосунку для комплексного управління нотатками, проєктами та робочими записами. Було показано, як ретельно спланована та реалізована архітектура, база даних та модель застосунку можуть забезпечити надійне та ефективне рішення для організації робочих процесів.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Розробка раціональної діяльності та створення сприятливих і безпечних умов персоналу

Тема кваліфікаційної роботи освітнього рівня «Магістр» присвячена розробці веб-системи комплексного управління робочими завданнями, нотатками та проєктами. Тому доцільно розглянути питання розробки середовища для ефективної та сприятливої роботи працівників, які будуть проводити багато часу, сидячи за комп'ютерами.

У будь-якому робочому середовищі забезпечення безпеки та добробуту персоналу має першочергове значення. У цьому розділі зосереджено увагу на розробці раціональної діяльності та створенні сприятливих і безпечних умов для персоналу в контексті охорони праці та безпеки в надзвичайних ситуаціях.

Розвиток раціональної діяльності та створення сприятливих і безпечних умов для персоналу є важливими складовими ефективного управління охороною праці та технікою безпеки. Впроваджуючи профілактичні заходи та розвиваючи культуру безпеки, організації можуть зменшити ризики, запобігти нещасним випадкам і захистити здоров'я та добробут своїх працівників.

Першим кроком у створенні безпечних умов для персоналу є проведення комплексної оцінки ризиків та виявлення потенційних небезпек на робочому місці. Це передбачає оцінку фізичного середовища, обладнання, процесів і людських факторів, які можуть становити загрозу для безпеки персоналу [21].

Організації повинні розробити чітку та стислу політику безпеки та процедури, що окреслюють очікування безпечної поведінки та протоколи реагування на надзвичайні ситуації. Ці документи слід регулярно переглядати, оновлювати та доводити до відома всього персоналу для забезпечення відповідності та обізнаності.

Забезпечення персоналу належним навчанням і освітою щодо техніки безпеки, дій у надзвичайних ситуаціях та поінформованості про небезпеку має

важливе значення для сприяння культурі безпеки. Програми навчання повинні бути адаптовані до конкретних потреб різних ролей і відділів в організації.

Роботодавці повинні забезпечити відповідне обладнання безпеки, інструменти та засоби контролю, щоб зменшити ризики та захистити персонал від потенційних небезпек. Це може включати засоби індивідуального захисту (ЗІЗ), інженерний контроль та адміністративні заходи для мінімізації впливу небезпеки [22].

Регулярні перевірки та технічне обслуговування обладнання, приміщень і робочих зон необхідні для швидкого виявлення та усунення потенційних загроз безпеці. Звичайні перевірки та процедури профілактичного обслуговування допомагають забезпечити дотримання стандартів безпеки та їх відповідність [23].

Організації повинні розвивати культуру безпеки, коли персонал заохочується віддавати пріоритет безпеці в усіх аспектах своєї роботи. Це передбачає сприяння відкритому спілкуванню, заохочення звітування про проблеми з безпекою, а також визнання та винагороду за поведінку, що піклується про безпеку.

Персонал повинен брати активну участь у розробці та впровадженні заходів безпеки. Запрошуючи співробітників на думку, організації можуть отримати цінну інформацію про потенційні небезпеки та визначити можливості для вдосконалення.

Ефективна готовність до надзвичайних ситуацій і планування реагування мають вирішальне значення для пом’якшення впливу непередбачуваних подій і мінімізації шкоди для персоналу. Організації повинні проводити регулярні тренування, моделювання та тренування, щоб забезпечити готовність і майстерність у надзвичайних ситуаціях [24].

Управління безпекою – це безперервний процес, який вимагає постійного оцінювання, зворотного зв’язку та вдосконалення. Організації повинні регулярно переглядати свою політику безпеки, процедури та показники ефективності, щоб визначити області для покращення та запровадити коригувальні дії, якщо це необхідно.

Розвиток раціональної діяльності та створення сприятливих і безпечних умов для персоналу є засадами ефективного управління охороною праці та технікою безпеки. Завдяки оцінці ризиків, політиці безпеки, навчанню, обладнанню та профілактичним заходам організації можуть сприяти розвитку культури безпеки, мінімізувати ризики та забезпечити благополуччя своїх працівників як у звичайних, так і в екстрених ситуаціях.

4.2 Вплив електромагнітного випромінювання комп'ютера на здоров'я користувача

Повсюдне використання комп'ютерів у сучасному суспільстві викликає занепокоєння щодо потенційного впливу на здоров'я електромагнітного випромінювання, яке випромінюють ці пристрої. Завдяки аналізу результатів досліджень, включаючи епідеміологічні дослідження, експериментальні дослідження та теоретичні моделі, у цій статті досліджуються потенційні ризики для здоров'я, пов'язані з тривалим впливом комп'ютерного випромінювання. Крім того, тут обговорюються стратегії пом'якшення наслідків і рекомендації щодо мінімізації впливу та захисту здоров'я користувачів [25].

У сучасну епоху цифрових технологій комп'ютери стали невід'ємною частиною повсякденного життя, слугуючи основними інструментами для роботи, спілкування, розваг та навчання. Однак із зростанням залежності від комп'ютерів зростає й занепокоєння щодо їх потенційного впливу на здоров'я людини. Однією з сфер особливого занепокоєння є електромагнітне випромінювання, яке випромінюють комп'ютери, що викликало запитання щодо його потенційного впливу на здоров'я. Ця стаття спрямована на огляд наявних досліджень і пролиття світла на зв'язок між електромагнітним випромінюванням комп'ютера та здоров'ям користувача.

Комп'ютери випромінюють електромагнітне випромінювання в широкому спектрі частот із різними джерелами, що сприяють загальному впливу. Надзвичайно низькочастотне (ELF) випромінювання генерується джерелами живлення та дисплеями, тоді як радіочастотне (RF) випромінювання

випромінюється компонентами бездротового підключення, такими як Wi-Fi і Bluetooth. Розуміння типів і джерел радіації має важливе значення для оцінки їхнього потенційного впливу на здоров'я та впровадження відповідних стратегій пом'якшення.

Вплив електромагнітного випромінювання комп'ютера на здоров'я користувача є складним і багатогранним питанням, яке є предметом значних наукових досліджень. Епідеміологічні дослідження зв'язку між використанням комп'ютера та наслідками для здоров'я дали неоднозначні результати. У той час як деякі дослідження повідомляли про зв'язок між тривалим використанням комп'ютера та такими проблемами зі здоров'ям, як напруга очей, головні болі та розлади опорно-рухового апарату, інші виявили непереконливі або незначні ефекти. Експериментальні дослідження біологічних ефектів електромагнітного випромінювання показали докази окислювального стресу, пошкодження ДНК і потенційної канцерогенності на клітинних і тваринних моделях. Однак перенесення цих висновків на здоров'я людини залишається складним через відмінності в рівнях впливу та механізмах реакції.

Щоб мінімізувати потенційні ризики для здоров'я, пов'язані з електромагнітним випромінюванням комп'ютера, можна застосувати кілька стратегій і рекомендацій. Оптимізація ергономіки робочої станції, зменшення відблисків екрана, регулярні перерви, використання екрануючих пристроїв і обмеження впливу функцій бездротового підключення є одними з заходів, які можуть допомогти зменшити вплив і захистити здоров'я користувачів. Крім того, регуляторні органи та організації можуть відігравати певну роль у створенні вказівок щодо безпечного використання комп'ютера та встановлення обмежень на електромагнітне випромінювання від електронних пристроїв.

Незважаючи на те, що вплив комп'ютерного електромагнітного випромінювання на здоров'я користувача залишається предметом постійних досліджень і дискусій, дані свідчать про те, що тривалий вплив певних типів випромінювання може становити потенційний ризик. Окремим особам, організаціям і контролюючим органам важливо зберігати пильність і вживати запобіжних заходів для мінімізації впливу та захисту здоров'я користувачів.

Потрібні подальші дослідження, щоб з’ясувати механізми, що лежать в основі впливу комп’ютерного випромінювання на здоров’я, і розробити ефективні стратегії пом’якшення пов’язаних з цим ризиків [26].

Майбутні дослідження в цій галузі мають бути зосереджені на усуненні наявних прогалин у знаннях і вдосконаленні нашого розуміння впливу комп’ютерного електромагнітного випромінювання на здоров’я. Лонгітюдні дослідження з більшим розміром вибірки та вдосконаленими методологіями можуть забезпечити більш чітке розуміння зв’язку між використанням комп’ютера та наслідками для здоров’я. Крім того, міждисциплінарна співпраця між дослідниками в таких галузях, як інженерія, медицина та охорона здоров’я, може сприяти розробці інноваційних рішень для пом’якшення впливу та зміцнення здоров’я користувачів у епоху цифрових технологій.

4.3 Організація цивільного захисту на житлових об`єктаз під час воєнного стану

4.3.1 Створення відповідної інфраструктури для безпеки населення під час воєнного стану

Під час воєнного стану забезпечення безпеки мешканців житлових будинків стає першочерговим. Зусилля цивільної оборони мають вирішальне значення в організації та здійсненні заходів щодо захисту цивільного населення, підтримання порядку та ефективного реагування на надзвичайні ситуації. У цьому розділі розглядається організація цивільного захисту житлових приміщень під час воєнного стану з акцентом на планування, координацію та механізми реагування.

Органи цивільної оборони проводять комплексну оцінку ризиків і аналіз вразливості, щоб визначити потенційні загрози та небезпеки для житлових приміщень. Це включає оцінку ризиків від стихійних лих, технологічних аварій, тероризму та громадянських заворушень.

На основі оцінки ризиків розробляються плани реагування на надзвичайні ситуації та адаптуються до конкретних потреб житлових приміщень. Ці плани окреслюють процедури евакуації, укриття, медичної допомоги, зв’язку та координації із зовнішніми агентствами [27].

Персонал цивільної оборони та жителі проходять навчання щодо процедур реагування на надзвичайні ситуації, включаючи вправи з евакуації, першої допомоги, пожежної безпеки та самозахисту. Для перевірки ефективності планів реагування та підвищення готовності проводяться регулярні навчання та моделювання.

Установи цивільної оборони співпрацюють з місцевою владою, правоохоронними органами, екстреними службами, громадськими організаціями та асоціаціями мешканців для координації заходів щодо готовності та реагування. Встановлюються чіткі лінії зв’язку та протоколи координації для забезпечення ефективної співпраці під час надзвичайних ситуацій.

Мешканці беруть активну участь в ініціативах цивільного захисту через програми роботи з громадою, інформаційні кампанії та навчальні сесії. Їх заохочують брати участь у програмах спостереження за сусідами, волонтерських групах реагування на надзвичайні ситуації та заходах зі зміцнення стійкості громади.

Житлові будинки обладнані системами оповіщення про надзвичайні ситуації, включаючи сирени, гучномовці та цифрові сповіщення, щоб розповсюджувати попередження та інструкції мешканцям під час надзвичайних ситуацій.

У разі надзвичайної ситуації, що потребує евакуації, визначаються призначені маршрути евакуації та пункти збору, а процедури евакуації виконуються впорядкованим чином. Для розміщення переміщених мешканців створюють тимчасові притулки, які забезпечують такі необхідні послуги, як їжа, вода, медичне обслуговування та консультації [28].

Правоохоронні органи підтримують безпеку та забезпечують виконання норм воєнного стану для забезпечення громадської безпеки та порядку в

житлових приміщеннях. Заходи можуть включати комендантську годину, контрольно-пропускні пункти, обмежений доступ і патрулювання для запобігання злочинності та підтримки контролю.

Після того, як надзвичайна ситуація минула, групи з оцінки збитків оцінюють вплив на житлові об’єкти та інфраструктуру, визначаючи пріоритетність заходів з відновлення, щоб відновити основні послуги та полегшити повернення мешканців до своїх домівок.

Агентства цивільної оборони розробляють плани стійкості, щоб забезпечити безперервність основних послуг і операцій у житлових приміщеннях під час надзвичайних ситуацій. Це включає резервне електропостачання, системи зв’язку в надзвичайних ситуаціях і плани на випадок надзвичайної ситуації для критичної інфраструктури.

Довгострокові заходи впроваджуються для зменшення ризику майбутніх надзвичайних ситуацій, такі як удосконалення будівельних норм, зміцнення інфраструктури, підвищення готовності до надзвичайних ситуацій і сприяння стійкості громади через освітні та просвітницькі програми.

Організація цивільного захисту в житлових будинках під час воєнного стану потребує ретельного планування, координації та взаємодії зацікавлених сторін. Інвестуючи в засоби готовності, навчання, зв’язку та реагування, органи влади можуть ефективно захистити мешканців, зменшити ризики та забезпечити безперервність надання основних послуг під час кризи. Залучення громади та зусилля з підвищення стійкості відіграють вирішальну роль у збільшенні спроможності житлових приміщень протистояти надзвичайним ситуаціям і відновлюватися після них.

У період воєнного стану організація цивільного захисту на об’єктах житлового фонду стає найважливішим аспектом забезпечення безпеки та благополуччя мешканців. У цьому розділі розглядається конкретна організаційна структура та стратегії, що застосовуються для управління зусиллями цивільного захисту в житлових приміщеннях у періоди воєнного стану.

Житлові заклади можуть створювати комітети або ради мешканців, яким доручено координувати зусилля цивільного захисту на місцевому рівні. Ці комітети служать зв'язком між мешканцями, адміністрацією та органами цивільного захисту, сприяючи комунікації, поширюючи інформацію та організовуючи громадські ініціативи.

Кожен житловий заклад може призначати групи реагування на надзвичайні ситуації, що складаються з навченого персоналу, відповідального за виконання процедур надзвичайних ситуацій, проведення евакуації, надання першої допомоги та управління притулками для надзвичайних ситуацій. Ці групи проходять регулярне навчання та тренування, щоб забезпечити готовність та ефективність реагування на кризи.

Адміністратори закладу проводять оцінку ризику, щоб визначити потенційні небезпеки та вразливі місця, характерні для житлового середовища. Для зменшення ризиків і підвищення готовності впроваджуються такі заходи пом'якшення, як забезпечення безпеки інфраструктури, впровадження протоколів протипожежної безпеки та встановлення систем зв'язку в екстрених ситуаціях.

Розробляються комплексні плани на випадок надзвичайних ситуацій, у яких описано процедури евакуації, укриття, медичної допомоги та зв'язку під час надзвичайних ситуацій. Ці плани розроблено з урахуванням унікальних характеристик і потреб кожного житлового закладу та регулярно переглядаються, оновлюються та доводяться до відома мешканців і персоналу.

Житлові приміщення обладнані системами оповіщення про надзвичайні ситуації для швидкого розповсюдження попереджень та інструкцій мешканцям під час надзвичайних ситуацій. Ці системи можуть включати сирени, гучномовці, оповіщення текстовими повідомленнями та цифрові сповіщення для забезпечення широкого та своєчасного зв'язку.

Встановлено кілька каналів зв'язку для полегшення зв'язку між адміністрацією об'єкта, групами екстреного реагування, мешканцями та зовнішніми органами влади. Ці канали можуть включати спеціальні гарячі лінії,

екстрені радіостанції, платформи соціальних мереж і дошки оголошень для спільнот.

Чіткі маршрути евакуації та пункти збору визначені та розміщені на видному місці в житлових приміщеннях. Мешканці знайомляться з процедурами евакуації та беруть участь у регулярних тренуваннях щодо безпечної евакуації та збирання у визначених місцях.

Спеціально призначені притулки для надзвичайних ситуацій у житлових закладах обладнані для розміщення переміщених мешканців і надання основних послуг, таких як їжа, вода, медичне обслуговування та тимчасове житло. Ці укриття готові працювати автономно протягом тривалого періоду, якщо це необхідно.

Для захисту житлових приміщень під час воєнного стану впроваджено посилені заходи безпеки, зокрема посилене спостереження, обмежені точки доступу та патрулювання периметра. Співробітники служби безпеки співпрацюють з органами цивільної оборони для підтримки порядку, запобігання несанкціонованому доступу та оперативного реагування на загрози безпеці.

Доступ до житлових приміщень може бути обмежений або врегульований у період воєнного стану для забезпечення безпеки мешканців. Точки входу та виходу контролюються, і можуть проводитися ідентифікаційні перевірки, щоб запобігти проникненню в приміщення сторонніх осіб.

Вживаються заходи для підвищення стійкості критичної інфраструктури в житлових приміщеннях, включаючи резервне електропостачання, водопостачання та системи зв’язку. Ці заходи забезпечують безперервність надання основних послуг і мінімізують збої під час надзвичайних ситуацій.

Адміністратори закладу впроваджують стратегії управління ресурсами, щоб забезпечити доступність основних засобів, включаючи їжу, воду, медичні засоби та обладнання для екстреної допомоги. Запаси регулярно поповнюються і забезпечуються для задоволення потреб мешканців та заходів з реагування на надзвичайні ситуації.

Організація цивільного захисту житлових будинків під час воєнного стану – це багатопланова справа, яка потребує ефективної координації, комунікації та

готовності. Встановлюючи чіткі організаційні структури, впроваджуючи комплексні Плани на випадок надзвичайних ситуацій і надання пріоритету безпеці та добробуту мешканців, житлові заклади можуть ефективно зменшити ризики, реагувати на надзвичайні ситуації та забезпечити безперервність надання основних послуг у періоди воєнного стану. Співпраця між адміністрацією закладу, мешканцями, групами реагування на надзвичайні ситуації та органами цивільної оборони має важливе значення для створення стійкого та безпечного житлового середовища під час кризи.

4.3.2 Евакуаційні плани та зони під час воєнного стану

У період воєнного стану впровадження ефективних планів евакуації та визначення зон евакуації є ключовими складовими стратегії цивільного захисту, спрямованої на захист цивільного населення від різних загроз і надзвичайних ситуацій. У цьому розділі розглядаються принципи, процеси та міркування, пов’язані з розробкою та виконанням планів евакуації та зон у контексті воєнного стану.

У часи підвищеної безпеки або громадянських заворушень потреба в організованих і скоординованих зусиллях з евакуації стає необхідною для захисту життя та добробуту цивільних. Плани та зони евакуації сприяють упорядкованому та ефективному переміщенню людей із зон небезпеки чи нестабільності під час воєнного стану.

Плани евакуації ґрунтуються на комплексній оцінці ризику, яка визначає потенційні загрози, небезпеки та вразливі місця в певних географічних зонах. Ці оцінки враховують такі фактори, як стихійні лиха, технічні аварії, громадянські заворушення, терористичні загрози та військові операції.

Плани евакуації враховують демографічні показники та щільність населення постраждалих районів, включаючи кількість жителів, підприємств, шкіл, закладів охорони здоров’я та іншої критичної інфраструктури. Під час планування евакуації особлива увага приділяється вразливим групам населення, таким як діти, люди похилого віку, інваліди та особи з особливими потребами.

Ефективна комунікація є важливою для передачі інструкцій щодо евакуації, попереджень і вказівок мешканцям і зацікавленим сторонам. Плани евакуації включають комунікаційні стратегії та канали, які охоплюють різноманітну аудиторію, включаючи публічні оголошення, системи оповіщення про надзвичайні ситуації, соціальні медіа та громадські ініціативи.

Плани евакуації розробляються завдяки співпраці між кількома установами, зокрема органами цивільної оборони, екстреними службами, правоохоронними органами, місцевою владою, постачальниками медичних послуг, транспортними службами та громадськими організаціями. Міжвідомча координація гарантує, що плани евакуації є комплексними, добре скоординованими та узгодженими з існуючими протоколами реагування на надзвичайні ситуації.

Плани евакуації включають визначені маршрути евакуації, пункти збору та транспортні ресурси для сприяння безпечному та впорядкованому переміщенню евакуйованих. Ці маршрути ретельно плануються, щоб уникнути заторів, вузьких місць і потенційних небезпек, беручи до уваги такі фактори, як пропускна здатність доріг, потік транспорту, доступність і альтернативні маршрути.

Плани евакуації включають положення щодо тимчасового притулку, медичної допомоги та допоміжних послуг для евакуйованих, включаючи їжу, воду, санітарію, медичну допомогу та психосоціальну підтримку. Місця укриття визначаються заздалегідь і обладнуються для розміщення евакуйованих із предметами першої необхідності та основними послугами.

Зони евакуації визначаються на основі рівня ризику та загрози, пов’язаної з різними небезпеками та надзвичайними ситуаціями. Райони високого ризику, схильні до стихійних лих, промислових аварій або громадських заворушень, позначаються як основні зони евакуації, тоді як вторинні зони можуть охоплювати райони з меншим ризиком або служити плацдармами для евакуації.

Зони евакуації визначаються на основі географічних кордонів, топографічних особливостей, характеристик інфраструктури та щільності населення. Такі фактори, як близькість до небезпечних об’єктів, заплави,

сейсмічні зони та військові об’єкти, впливають на розмежування зон евакуації та розподіл ресурсів.

Плани евакуації окреслюють протоколи та тригери для активації евакуації на основі попередньо визначених критеріїв, таких як неминучі загрози, ескалація ризиків або офіційні директиви від цивільних органів влади. Процедури активації включають видачу наказів про евакуацію, мобілізацію ресурсів та ініціювання кампаній з підвищення обізнаності населення для інформування мешканців та зацікавлених сторін.

Операції з евакуації координуються та контролюються призначеними командними центрами, центрами аварійних операцій або командними пунктами надзвичайних ситуацій, де основні зацікавлені сторони наглядають і керують діяльністю з евакуації. Ефективна координація забезпечує впорядковане розгортання ресурсів, зв’язок з евакуйованими та моніторинг ходу евакуації в реальному часі.

Плани та зони евакуації відіграють вирішальну роль у захисті цивільного населення та мінімізації ризиків у періоди воєнного стану. Співпраця, комунікація та ефективне виконання, органи влади можуть розробити та впровадити стратегії евакуації, які захистять життя, збережуть інфраструктуру та підвищать стійкість громади до надзвичайних ситуацій. Постійний перегляд, оновлення та тренування гарантують, що плани евакуації залишаються адаптивними та реагують на зміну загроз та обставин.

4.3.3 Підготовка населення до дій під час воєнного стану

Підготовка населення до дій під час воєнного стану є важливою для забезпечення громадської безпеки, підтримання порядку та ліквідації наслідків надзвичайних ситуацій. У цьому розділі обговорюються стратегії та ініціативи, спрямовані на освіту, навчання та розширення можливостей населення для ефективного реагування на ситуації воєнного стану.

Урядові установи та органи цивільної оборони проводять інформаційні кампанії для інформування населення про положення про воєнний стан, порядок

надзвичайних ситуацій, а також про їхні ролі та обов'язки під час криз. Ці кампанії використовують різні канали зв'язку, зокрема телебачення, радіо, соціальні медіа та громадські ініціативи, щоб охопити широку аудиторію.

Програми громадської освіти надають громадянам знання та навички, необхідні для реагування на ситуації воєнного стану. Навчальні заняття охоплюють такі теми, як готовність до надзвичайних ситуацій, процедури евакуації, надання першої допомоги, самозахист і знання громадянських прав. Ці програми дають людям змогу вживати активних заходів, щоб захистити себе, свої родини та громади.

Місцева влада організовує зустрічі громади, семінари та сесії ратуші, щоб залучити жителів до заходів щодо підготовки до воєнного стану. Ці форуми надають можливість для діалогу, зворотного зв'язку та співпраці між мешканцями, урядовцями, правоохоронними органами та організаціями громадянського суспільства.

Програми сусідського спостереження створені для сприяння пильності, співпраці та взаємної підтримки серед мешканців у моніторингу та повідомленні про підозрілу діяльність під час воєнного стану. Мешканці проходять навчання із запобігання злочинності, методів спостереження та протоколів спілкування для підвищення безпеки та безпеки громади.

Громадян заохочують до участі в групах реагування на надзвичайні ситуації, завданням яких є підтримка зусиль цивільної оборони під час воєнного стану. Ці команди можуть надавати допомогу в різних сферах, зокрема в контролі над натовпом, управлінні дорожнім рухом, пошуково-рятувальних операціях і розподілі товарів для надзвичайних ситуацій.

Органи цивільної оборони набирають і навчають допоміжних цивільних осіб для посилення офіційних можливостей реагування та надання додаткової робочої сили під час надзвичайних ситуацій. Допоміжні особи проходять навчання щодо дій у надзвичайних ситуаціях, протоколів зв'язку та роботи з громадськістю, що дозволяє їм допомагати в евакуації, укритті та операціях з надання допомоги.

Організації громадянських прав і правозахисні групи проводять кампанії «Знай свої права», щоб ознайомити населення з їхніми законними правами та обов'язками в умовах воєнного стану. Громадян інформують про конституційний захист, належні процесуальні права та процедури оскарження свавільного затримання чи незаконних дій органів влади.

Організації правової допомоги пропонують безкоштовні юридичні послуги та допомогу особам, які постраждали від воєнного стану, включно з тими, хто зіткнувся з судовими оскарженнями, затриманням або порушенням прав людини. Громадян заохочують звертатися за юридичною консультацією та підтримкою, щоб захистити свої права та отримати компенсацію за скарги.

Підготовка населення до дій під час воєнного стану – це багатогранний процес, який потребує співпраці, залучення та розширення можливостей громадян. Заохочуючи громадську обізнаність, освіту, залучення громади та волонтерство, влада може підвищити стійкість громад, сприяти участі громадян і сприяти ефективному реагуванню на надзвичайні ситуації. Розширення можливостей громадян зі знаннями, навичками та ресурсами дозволяє їм відігравати активну роль у захисті власного добробуту та сприяти колективним зусиллям із забезпечення громадської безпеки під час воєнного стану.

4.4 Висновки до четвертого розділу

В третьому розділі кваліфікаційної роботи описані важливі питання охорони праці та безпеки в надзвичайних ситуаціях. Спершу подано методики розробки раціональної діяльності та створення сприятливих і безпечних умов персоналу.

Описано вплив електромагнітного випромінювання комп'ютера на здоров'я користувачів. Подано ідеї як мінімізувати негативні наслідки.

Розглянуто важливу тему організації цивільного захисту на житлових об'єктах під час воєнного стану, де описано створення відповідної інфраструктури для безпеки населення під час воєнного стану. Досліджено евакуаційні плани та підготовку населення у цей період.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи магістра було проведено ознайомлення з методиками досліджень процесів та явищ розробки стартапу веб-застосунку для побудови систем комплексного управління та ведення нотаток. Було вивчено, як працювати з програмними засобами, необхідними для проведення досліджень.

Вивчено методи аналізу та обробки результатів досліджень з використанням комп'ютерної техніки, правила оформлення наукових звітів, статей, матеріалів для патентування, формування свідоцтв авторського права, уміти виконувати бібліографічні дослідження та патентний пошук.

Виконано експериментальні та теоретичні дослідження за обраною темою магістерської роботи, а саме розробку стартапу веб-застосунку для побудови систем комплексного управління та ведення нотаток.

Було вивчено, як застосовувати методи обробки результатів досліджень з використанням комп'ютерної техніки та підготовлено матеріали для звіту, тези, наукової статті чи авторського свідоцтва за матеріалами дослідженнями.

Було описано та проаналізовано дослідження методів конспектування у навчанні, наведено основні види та техніки. Показано досвід студентів та викладачів на реальних дослідах, та сформовано актуальність обраної тематики магістерської роботи.

Описано ключові етапи розробки веб-застосунку, використовуючи MERN стек. Досліджено та описано ключові технології розробки додатків, визначено їх переваги, та методи застосування.

Для розробки обрано такі технології, як: JavaScriot, ReactJS, Redux, NodeJS, ExpressJS, MongoDB, Doker та інші.

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр» здійснено аналіз поняття конспектування, проведено дослідження видів, форм та методів конспектування серед студентів. Подано дослідницькі матеріали з інформацією про цифрові формати ведення нотаток.

У другому розділі магістерської дисертації проведено ретельний аналіз технологій, що застосовуються для розробки веб-застосунків. В цьому розділі детально описано кожну технологію, вказано їх переваги та недоліки, що дозволило зробити обґрунтований вибір найбільш відповідних інструментів для реалізації проєкту. Також проведено аналіз поняття інформаційної системи та описано технології, які використовуються для її розробки. Було визначено концепцію стеку технологій, і зокрема, на прикладі MERN стеку, викладено поєднання різноманітних технологій, які спільно працюють для успішного функціонування веб-застосунку.

У третьому розділі кваліфікаційної роботи було докладно розглянуто створення веб-застосунку для комплексного управління нотатками, проєктами та робочими записами. В даному розділі висвітлено наступні питання.

У четвертому розділі розкрито важливі питання охорони праці та безпеки в надзвичайних ситуаціях, де описано методи мінімізації впливу випромінювання на здоров’я працівника та різні ситуації та дії під час них в межах воєнного стану.

ПЕРЕЛІК ДЖЕРЕЛ

1 Мосій, Л.Є., Струтинська, І.В., & Козбур, Г.В. (2023). Цифрова трансформація: успішний досвід Європи. У Матеріали LII Науково-технічної конференції підрозділів ВНТУ (стор. [вказати номери сторінок]). Вінниця: ВНТУ.

2 Piolat A, Olive T, Kellogg RT. Cognitive effort during note-taking. Appl Cognit Psychol. 2005;19:291-312.

3 Мосій, Л.Є., Струтинська, І.В., & Козбур, Г.В. (2023). Роль комп'ютерно-інформаційних технологій у цифровій трансформації економіки. У Матеріали XII Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій “ (стор. 432-434). Видавець: ФОП Паляниця В.А.

4 Мосій, Л.Є., Козбур, Г.В., & Мосій, О.Б. (2023). Цифрова трансформація: стратегії та інструменти. Журнал "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення" (випуск 78), 69. Retrieved from [http://www.konferenciaonline.org.ua/data/downloads/file_1693298685.pdf#page=69]

5 Cain J, Bird ER, Jones MK. Mobile computing initiatives within pharmacy education. Am J Pharm Educ. 2008;72(4):Article 76.

6 Mueller PA, Oppenheimer DM. The pen is mightier than the keyboard: Advantages of longhand over laptop note-taking. Psychol Sci. 2014;25(6):1159-1168.

7 Kozbur, H. V., & Romanets, A. (2022, December 7). Проблеми аутентифікації акаунтів у соцмережах. У Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “Тернопільського національного технічного університету імені Івана Пулюя (с. 44-44). ТНТУ.

8 Романець, А., & Козбур, Г.В. (2022). Безпека соцмережі під час аутентифікації користувача. У Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології“ Тернопільського національного технічного університету імені Івана Пулюя (стор. 45-45). Тернопіль: ТНТУ.

9 Mang CF, Wardley LJ. Effective adoption of tablets in post-secondary education: Recommendations based on a trial of iPads in university classes. J Inf Technol Educ: Innov Pract. 2012;11:301-317.

10 Ventola CL. Mobile devices and apps for health-care professionals: Uses and benefits. Pharm Ther. 2014;39(5):356-64.

11 Bauer A, Koedinger K. Pasting and encoding: Note-taking in online courses. Presented at: Sixth International Conference on Advanced Learning Technologies, 2006; July 5-7, 2006; Kerkrade, Netherlands. [Електронний ресурс] – Режим доступу до ресурсу: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1652559. Accessed August 18, 2014.

12 Williams JA, Pence HE. Smartphones, a powerful tool in the chemistry classroom. J Chem Educ. 2011;88:683-686.

13 Moore B, Keenan N. Student success strategies and study skills. University of Kentucky College of Pharmacy. Lexington, KY. Sept 8, 2014. Student Convocation Presentation.

14 O'Neill K, Holmer H, Greenberg SLM, Meara JG. Applying surgical apps: Smartphone and tablet apps prove useful in clinical practice. Bulletin of the American College of Surgeons. [Електронний ресурс] – Режим доступу до ресурсу: http://bulletin.facs.org/2013/11/applying-surgical-apps/. Published November 1, 2013. Accessed August 18, 2014.

15 Diemer TT, Fernandez E, Streepey JW. Student perceptions of classroom engagement and learning using iPads. J Teach Learn Technol. 2012;1(2):13-25.

16 Gackenheimer C. Introduction to React // Introduction to React. 2015.

17 Freeman A. Pro JavaScript for Web Apps // Pro JavaScript for Web Apps. Apress, 2012.

18 Fielding J. Beginning Responsive Web Design with HTML5 and CSS3 // Beginning Responsive Web Design with HTML5 and CSS3. 2014.

19 Benefits of JSX in React [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/React_(JavaScript_library)#JSX.

20 Dror IE. Technology-enhanced learning: The good, the bad, and the ugly. Pragmatics Cognit. 2008;16:215-223.

21 Franko OI, Tirrell TF. Smartphone app use among medical providers in ACGME training programs. J Med Systems. 2012;36(5):3135-3139

22 Aungst TD. Medical applications for pharmacists using mobile devices. Ann Pharmacother. 2013;47 (7-8):1088-95.

23 Bjork RA. (1994). Memory and metamemory considerations in the training of human beings. In: Metcalfe J, Shimamura A, eds. Metacognition: Knowing About Knowing. Cambridge, MA: MIT Press; 1994: 185–205.

24 Створення сприятливих умов праці на виробництві. Охорана праці і пожежна безпека : веб-сайт. URL: https://oppb.com.ua/news/stvorennya-spryyatlyvyh-umov-praci-na-vyrobnyctvi (дата звернення: 10.12.2023).

25 Організація цивільного захисту на житлових об’єктах під час воєнного стану: теоретичні та практичні аспекти. LIGA 360 : веб-сайт. URL: https://ips.ligazakon.net/document/view/ji09215g?an=2 (дата звернення: 10.12.2023).

26 Забезпечення безпеки населення на житлових об’єктах під час воєнного стану. WikiLegalAid : веб-сайт. URL: https://wiki.legalaid.gov.ua/ (дата звернення: 10.12.2023).

27 Особливості організації цивільного захисту на житлових об’єктах у сучасних умовах. На Урок : веб-сайт. URL: https://naurok.com.ua/prezentaciya-organizaciya-civilnogo-zahistu-v-suchasnih-umovah-49856.html (дата звернення: 10.12.2023).

28 Забезпечення безпеки населення на житлових об’єктах під час воєнного стану: моніторинговий звіт. WikiLegalAid : веб-сайт. URL: https://wiki.legalaid.gov.ua/ (дата звернення: 10.12.2023).

ДОДАТКИ

Додаток А

Теза наукової конференції

УДК 004.04
Вербіцький І. – ст. гр. СНнм-51
Тернопільський національний технічний університет імені Івана Пулюя

РОЗРОБКА SPA-ЗАСТОСУНКІВ НА MERN-СТЕК

Науковий керівник: старший викладач Шимчук Г.

Verbitskyi I.
Ternopil Ivan Pul'uj National Technical University

DEVELOPMENT OF SPA APPLICATIONS ON THE MERN STACK

Supervisor: Senior Lecturer Shymchuk G.

Ключові слова: MERN, SPA, SPF, MongoDB, React, Express, Node.js, розробка
Key words: MERN, SPA, SPF, MongoDB, React, Express, Node.js, development

Сучасні веб-застосунки, особливо ті, що пропонують складні функції та взаємодії з користувачами, вимагають використання ефективних технологій та підходів розробки. Одним з таких підходів є розробка односторінкових застосунків (SPA) на MERN-стеці, що включає MongoDB, Express, React та Node.js [1]. Отже, розглянемо процес розробки SPA-застосунків на MERN-стеці, його переваги та особливості.

SPA-застосунки є веб-додатками, в яких взаємодія з користувачем відбувається без перезавантаження сторінки. Вони забезпечують більш швидку та зручну взаємодію з користувачами, оскільки вони завантажують лише необхідні дані та оновлюють відповідні елементи на сторінці. MERN-стек використовується для розробки SPA-застосунків із використанням JavaScript-технологій на кожному етапі [1].

Перший етап розробки SPA-застосунку на MERN-стеці - це розробка серверної частини. Для цього використовується Node.js - середовище виконання JavaScript на серверному боці. За допомогою фреймворка Express.js створюється сервер, який відповідає на HTTP-запити від клієнтів, обробляє їх та взаємодіє з базою даних MongoDB. MongoDB - це документ-орієнтована база даних, яка дозволяє зберігати дані у вигляді документів у форматі BSON (Binary JSON), що спрощує роботу з даними та їх збереження [2].

Другий етап - розробка клієнтської частини, яка відповідіає за взаємодією з користувачем. Для цього використовується фреймворк React - одна з найпопулярніших бібліотек JavaScript для розробки інтерфейсів користувача. React дозволяє створювати компоненти, які можна перевикористовувати, та оновлювати їх стан на основі взаємодії з користувачем без перезавантаження сторінки [3].

Останній етап - це забезпечення взаємодії між серверною та клієнтською частинами за допомогою API. Express.js дозволяє створювати RESTful або GraphQL API, які надають доступ до даних, збережених в базі даних MongoDB, та дозволяють виконувати різні операції, такі як створення, оновлення, видалення даних [2].

Переваги розробки SPA-застосунків на MERN-стеці очевидні. По-перше, вони забезпечують швидку відповідь та більш зручну взаємодію з користувачами, оскільки вони оновлюють лише необхідні елементи сторінки без перезавантаження всієї сторінки. По-друге, вони дозволяють перевикористовувати компоненти, що спрощує розробку та підтримку коду

[1]. По-третє, MERN-стек є повністю JavaScript-стеком, що дозволяє розробникам використовувати одну мову програмування на всіх етапах розробки, що спрощує взаємодію між різними командами розробників.

Однак, розробка SPA-застосунків на MERN-стеці також має свої особливості та виклики. Одним з них є відправка запитів на сервер та керування станом додатку на клієнтській стороні. Це може вимагати додаткової роботи з опрацюванням стану та синхронізації даних між сервером та клієнтом. Крім того, забезпечення захисту даних також є важливим аспектом розробки односторінкових застосунків, оскільки клієнтська частина взаємодії з сервером через відкритий Інтернет може бути вразливою до атак [1].

Розробка SPA-застосунків на MERN-стеці також вимагає ретельного планування та керування проєктом [2]. Враховуючи ряд технологій та компонентів, які використовуються, команда розробників повинна бути організованою та ефективною. Наявність якісної документації, що пояснює архітектуру, функціональність та взаємодію між компонентами, також є важливим фактором успіху розробки SPA-застосунків на MERN-стеці.

Узагалі, розробка з використанням MERN відкриває безліч можливостей для створення масштабованих, високопродуктивних та майбутньоорієнтованих додатків. Ця технологічна стекова комбінація забезпечує зручну розробку фронтенду та бекенду, забезпечуючи однорідність та ефективність розробки на всіх етапах проєкту. З правильним плануванням, документуванням та керуванням проєктом, розробка SPA-застосунків може бути успішним шляхом до створення сучасних та інноваційних веб-додатків.

Для створення односторінкових сайтів на MERN-стеці також важливо враховувати фактори оптимізації та продуктивності. Завантаження великих обсягів даних на клієнтську сторону може призвести до погіршення продуктивності застосунку [2]. Використання технік оптимізації завантаження даних, кешування, розподілення бізнес-логіки між клієнтом та сервером, а також мінімізація зайвих запитів до сервера є важливими кроками для забезпечення продуктивності SPA-застосунків на MERN-стеці.

Розробка також може включати реалізацію додаткових функціональностей, таких як аутентифікація та авторизація користувачів, робота зі сторонніми API, розширені можливості керування станом додатку та взаємодії зі сторонніми бібліотеками та фреймворками [1].

Однією з переваг розробки MERN є можливість використання ряду інструментів та бібліотек, що сприяють прискоренню розробки, таких як Mongoose, Express, React та Node.js. Ці інструменти дозволяють розробникам ефективно створювати складні SPA-застосунки, роблячи код більш організованим, модульним та керованим.

Література:

1. Vasan Subramanian (2019). Pro MERN Stack Development: Building Full-Stack React Apps with MongoDB, Express, and Node. Apress. ISBN 978-1484243909.
2. Shama Hoque (May 29, 2018). Full-Stack React Projects: Modern web development using React 16, Node, Express, and MongoDB. Packt Publishing. ISBN 978-1788835534.
3. Kirupa Chinnathambi (27 July 2018). Learning React: A Hands-On Guide to Building Web Applications Using React and Redux. Addison-Wesley Professional. ISBN 978-0134843551.

Додаток Б

Теза наукової конференції

УДК 004.04
Вербіцький І. – ст. гр. СНнм-61
Тернопільський національний технічний університет імені Івана Пулюя

АДАПТАЦІЯ СИСТЕМ УПРАВЛІННЯ ДО НЕВИЗНАЧЕНОСТІ: РОЛЬ КОМУНІКАЦІЙ, СТРАТЕГІЧНОГО ПЛАНУВАННЯ ТА ОЦИФРУВАННЯ

Науковий керівник: асистент кафедри Ковальчик О.

Verbitskyi I.
Ternopil Ivan Pul'uj National Technical University

ADAPTATION OF MANAGEMENT SYSTEMS TO UNCERTAINTY: THE ROLE OF COMMUNICATIONS, STRATEGIC PLANNING AND DIGITALIZATION

Supervisor: Assistant of the Department Kovalchyk O.

Ключові слова: Невизначеність, стратегічне планування, ефективна комунікація, адаптація, гнучкість, оцифрування систем керування, автоматизація
Key words: Uncertainty, strategic planning, effective communication, adaptation, flexibility, digitization of control systems, automation

Сучасний бізнес опиняється перед викликами та невизначеністю, що виникає внаслідок війни, а також швидкого темпу змін у технологіях, економіці та соціальному середовищі. Це ставить під загрозу стабільність та ефективність управління підприємством. У цьому контексті важливо розглядати аспекти адаптації систем управління для забезпечення стійкості та конкурентоспроможності.

Невизначеність у бізнесі: Невизначеність є невід'ємною частиною бізнес-середовища. Зміни в економіці, технологічні революції та соціокультурні фактори створюють ситуації, коли передбачення стає складним завданням. Управління бізнесом в умовах невизначеності вимагає специфічних стратегій та підходів.

Роль стратегічного планування: Стратегічне планування стає ключовим елементом управління в умовах невизначеності. Організації повинні розробляти гнучкі та адаптивні стратегії, спроможні враховувати обставини, що швидко змінюються. Аналіз ризиків та розробка альтернативних сценаріїв стають невід'ємною частиною стратегічного процесу.

Важливість комунікацій: Ефективна комунікація виступає як каталізатор для успішної адаптації систем управління. Забезпечення відкритого та ефективного обміну інформацією між різними рівнями управління та підрозділами допомагає уникнути затримок та невдач у реалізації стратегічних ініціатив.

Гнучкість та реалізація змін: Адаптівність та гнучкість в управлінні дозволяють підприємствам швидко реагувати на зміни у навколишньому середовищі. Здатність швидко впроваджувати зміни вимагає не лише стратегічного планування, але й готовності персоналу до інновацій та активної участі в процесі змін.

Онлайн системи комплексного керування проєктами та завданнями: Однією з ключових складових успішної адаптації є використання сучасних онлайн систем управління проєктами та завданнями. Ці системи надають змогу централізовано відстежувати прогрес робіт, розподіляти завдання та спілкуватися в реальному часі. Вони полегшують координацію команди та забезпечують доступ до інформації в будь-який час та з будь-якого пристрою.

Оцифрування систем керування: Оцифрування систем управління стає критично важливим елементом адаптації до невизначеності. Використання цифрових інструментів дозволяє автоматизувати багато рутинних завдань, зменшити людський фактор та підвищити швидкість прийняття управлінських рішень. Віртуальне зберігання даних і можливість отримання реального часу дозволяють збільшити точність та ефективність управління.

Отже, Адаптація систем управління до невизначеності є критично важливою для забезпечення стабільності та конкурентоспроможності підприємства. Врахування ролі стратегічного планування, ефективних комунікацій, використання онлайн систем управління та оцифрування допомагають створити гнучкі та адаптивні механізми, необхідні для успішного функціонування в умовах постійних змін у бізнес-середовищі.

Література:

1. Волкова О. Цифровізація систем управління: характеристика та принципи. XI Міжнародна науково-практична конференція «Сучасні проблеми управління: Трансформація публічного управління у постковідному світі». Київ, Україна, 2021. URL: https://doi.org/10.20535/spu2021.249172 (дата звернення: 15.11.2023).
2. Халіна В., Колмакова О., Устіловська А. АДАПТАЦІЯ ПІДПРИЄМСТВ УКРАЇНИ ДО УМОВ НЕВИЗНАЧЕНОСТІ ЯК ДЕТЕРМІНАНТ ЇХ ЕКОНОМІЧНОЇ БЕЗПЕКИ. Наукові перспективи (Naukovì perspektivi). 2023. № 10(40). URL: https://doi.org/10.52058/2708-7530-2023-10(40)-510-523 (дата звернення: 15.11.2023).
3. Горбаченко С. А., Чепурна О. Є., Слатвінська В. М. АДАПТАЦІЯ ПРОЄКТНОГО ПІДХОДУ ДО УПРАВЛІННЯ СТАРТАПАМИ. Трансформаційна економіка. 2023. № 4 (04). С. 24–28. URL: https://doi.org/10.32782/2786-8141/2023-4-5 (дата звернення: 15.11.2023).