Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

АНАЛІЗ ФУНКЦІОНАЛІВ СУЧАСНИХ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКІВ
(тема)

Виконав:
студентки 2 курсу, групи ІНФм-22-1
Пікуль І.С.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Гороховатський В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентці Пікуль Інні Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз функціоналів сучасних програмних засобів для розроблення вебзастосунків

затверджена наказом по університету від 3 листопада 2023 року № 1280Ст

2. Термін подання студентом роботи до екзаменаційної комісії. 23 грудня 2023 р.

3. Вихідні дані до роботи літературні джерела, теоретичні основи та сучасні програмні засоби для розроблення вебзастосунків.

4. Перелік питань, що потрібно опрацювати в роботі

1. Огляд сучасних програмних засобів для веброзробки.
2. Обґрунтування і вибір засобів для розроблення вебзастосунку.
3. Дослідження та розроблення вебзастосунків на прикладі Next.js

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми, постановка задачі, огляд сучасних програмних засобів для створення вебзастосунків, вимоги до вебзастосунку, обґрунтування і вибір інструментів та систем, переваги фреймворку, результати дослідження.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.11.2023	виконано
2	Аналіз завдання, підбір літератури	03.11.23-05.11.23	виконано
3	Аналіз літератури з досліджуваної проблеми	05.11.23-06.11.23	виконано
4	Огляд сучасних програмних засобів	06.11.23-07.11.23	виконано
5	Обґрунтування і вибір засобів	07.11.23-08.11.23	виконано
6	Реалізація застосунку	08.11.23-10.11.23	виконано
7	Оформлення пояснювальної записки	11.11.23-12.11.23	виконано
8	Перевірка на плагіат	10.12.2023	виконано
9	Рецензування	18.12.2023	виконано
10	Підготовка презентації та доповіді	20.12.2023	виконано
11	Занесення роботи в електронний архів	02.01.2024	виконано
12	Попередній захист кваліфікаційної роботи	02.01.2024	виконано

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Гороховатський В.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 60 с., 2 табл., 17 рис., 49 джерел.

ВЕБЗАСТОСУНОК, SEO, ВЕБСАЙТ, NEXT.JS, ВІДЛАГОДЖЕННЯ, БРАУЗЕР, ЕФЕКТИВНІСТЬ.

Об’єктом дослідження є функціонали для розробки вебзастосунків на прикладі Next.js.

Метою дослідження є оцінка можливостей та особливостей функціоналів, які пропонує Next.js для розробки вебзастосунків та їх вплив на продуктивність та розширюваність проєктів.

Здійснено аналіз функціональних можливостей Next.js. Розглянуто інструменти для розширення функціональності. Проведено аналіз документації та спільноти. Сформульовано висновки та рекомендації.

У результаті аналізу здійснена програмна реалізація застосунку за допомогою фреймворка Next.js.

WEB APPLICATION, SEO, WEBSITE, NEXT.JS, DEBUGGING, BROWSER, EFFICIENCY.

The object of the research is the functionalities for developing web applications using Next.js.

The goal of the research is to assess the capabilities and features of the functionalities offered by Next.js for the development of web applications and their impact on the performance and scalability of projects.

Analysis of the functional capabilities of Next.js has been conducted. Tools for extending functionality have been considered. Documentation and community analysis has been carried out. Conclusions and recommendations have been formulated.

As a result of the analysis, the application has been programmatically implemented using the Next.js framework.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDE – Integrated Development Environment (інтегроване середовище розробки)

SCM – Source Code Management (система управління вихідним кодом)

CI/CD – Continuous Integration and Continuous Delivery (неперервна інтеграція та неперервна доставка)

HTML – HyperText Markup Language (мова гіпертекстової розмітки)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

SQL – Structured Query Language (мова структурованих запитів)

API – Application Programming Interface (інтерфейс прикладного програмування)

SSR – Server-Side Rendering (серверна генерація сторінок)

SSG – Static Site Generator (генератор статичних сайтів)

ВСТУП

Веброзробка є важливою галуззю в інформаційних технологіях, що продукує значущий попит на сучасні програмні засоби. Існують численні інструменти та платформи, призначені для розробки вебзастосунків. Однак вибір правильного інструменту для конкретного проєкту може бути складним завданням [1–5].

В цілому дослідження функціоналів сучасних програмних засобів для розроблення вебзастосунків важливо для того, щоб розробники та бізнеси мали можливість вибирати інструменти, які відповідають їхнім потребам і дозволяють створювати високоякісні вебзастосунки у відповідь на зростаючий попит і конкуренцію [6–11].

В умовах постійних технологічних інновацій, дослідження функціоналів програмних засобів для веброзробки дозволяє виявити не тільки поточні тенденції, але й адаптуватися до майбутніх викликів. Розгляд різноманітних мов програмування, фреймворків, інтегрованих середовищ розробки, інструментів безпеки та тестування стає основою для обґрунтування оптимального вибору для конкретного вебзастосунку.

Дослідження функціоналів дозволить зрозуміти, як кожен інструмент відповідає вимогам ринку, а також визначити його переваги та обмеження. Це стане основою для обґрунтування вибору конкретного засобу для подальшої розробки вебзастосунку, забезпечуючи його ефективність, надійність та відповідність сучасним стандартам веброзробки [12–19].

Актуальність дослідження полягає у стрімкому розвитку інформаційних технологій, і це справедливо особливо для веброзробки. Нові мови програмування, фреймворки, технології та інструменти з’являються на ринку з великою швидкістю. Тому важливо аналізувати актуальність та конкурентоспроможність цих інструментів.

1 ОГЛЯД СУЧАСНИХ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ВЕБРОЗРОБКИ

1.1 Аналіз сучасних засобів та середовищ програмування

Мова програмування – це формальна система символів та правил, призначена для створення послідовностей інструкцій (команд) для комп’ютера або інших обчислювальних пристроїв. Вона дозволяє програмістам визначати логіку обчислень, створювати алгоритми, обробляти дані та керувати роботою обчислювальних систем. Мови програмування надають абстрактні засоби виразу, які дозволяють конкретизувати завдання для обчислювача [11].

Фреймворк (або framework) – це велика структура або набір коду, який надає основний функціонал для розробки програмного продукту або застосунку. Фреймворки допомагають розробникам значно спростити процес розроблення шляхом надання готових компонентів, бібліотек та структурних рішень для роботи над конкретною задачею чи типом застосунку [1].

Розробка програмного продукту або вебзастосунку включає в себе чималу кількість етапів, від проєктування і архітектури до тестування та впровадження. Використання мов програмування та фреймворків сприяє структуризації і полегшенню цього процесу, що робить його більш ефективним. Враховуючи особливості кожної мови та фреймворку, розробники можуть здійснювати оптимальний вибір для досягнення поставлених завдань.

У таблиці 1.1 наведено перелік сучасних мов програмування, які використовуються для створення вебзастосунків. Для кожної мови програмування вказані її основні переваги та недоліки, а також назви фреймворків, які часто використовуються з цією мовою [1–3].

Таблиця 1.1 – Порівняння мов програмування

№	Мова програмування	Фреймворк	Переваги	Недоліки
1	**2**	**3**	**4**	**5**
1	HTML	Bootstrap, Metro UI, Onsen UI, Ionic, Semantic UI	Простий у вивченні та використанні, велике поширення	Не дозволяє виконувати програмний код
2	CSS	Bootstrap, Tailwind CSS, Foundation, Bulma, Skeleton	Дає широкий спектр можливостей для стилізації вебсторінки, велике поширення	Не дозволяє виконувати програмний код
3	JavaScript	React.js, AngularJS, Vue.js, Svelte	Динамічний і потужний, широке поширення. JavaScript є основною мовою для веброзробки на клієнтському боці	Може бути складним для вивчення та використання
4	Python	Django, Flask, FastAPI, Tornado	Простота в вивченні та використанні, велика спільнота розробників, велика кількість бібліотек і фреймворків	Python використовує GIL, що робить неможливим використання багатоядерних обчислювальних ресурсів для однопотокових програм. Це може обмежити продуктивність в високонавантажених застосунках
5	PHP	Laravel, Symfony, Yii, CodeIgniter, Zend Framework	Велика спільнота розробників, велика кількість бібліотек і фреймворків	На порівняно великих навантаженнях PHP може бути повільнішим у порівнянні з мовами, які використовують JIT-компіляцію, такими як Java або Go
6	Go	Gin, Echo, Beego, Revel, Fiber	Швидкість, ефективність, масштабованість, простота в вивченні та використанні	Невелика спільнота розробників, небагато бібліотек і фреймворків

Продовження таблиці 1.1

1	2	3	4	5
7	Java	Spring Boot, JSF, Apache Struts, Play Framework	Безпека, масштабованість, велика спільнота розробників, велика кількість бібліотек і фреймворків	Java відома своєю великою вимогою до пам’яті та продуктивністю в порівнянні з деякими іншими мовами, особливо на початку роботи застосунку. Це може вплинути на швидкодію та вартість масштабування
8	C#	ASP.NET Core, ASP.NET Web Forms, Blazor	Безпека, масштабованість, велика спільнота розробників, велика кількість бібліотек і фреймворків	Відпочатку була розроблена для платформи Windows, і це може створювати обмеження для розробки платформонезалежних вебзастосунків
9	Ruby	Ruby on Rails (Rails), Sinatra	Простота в вивченні та використанні, велика спільнота розробників, велика кількість бібліотек і фреймворків	Може вимагати більше пам’яті в порівнянні з іншими мовами програмування. Це може призвести до обмеженості ресурсів на сервері та погіршити продуктивність високонавантажених вебзастосунків

IDE – це скорочення від Integrated Development Environment, що означає «інтегроване середовище розробки». Це програмне забезпечення, яке надає розробникам все, що їм потрібно для створення програм, включаючи текстовий редактор, компілятор, відладчик і засоби для тестування.

IDE часто бувають спеціально розроблені для певної мови програмування або типу програм. Наприклад, Visual Studio Code – це IDE, яка призначена для розробки програм на C# і Visual Basic.

IDE можуть бути дуже корисними для розробників, оскільки вони можуть заощадити час і зусилля. Вони також можуть допомогти розробникам створювати більш якісні програми.

Основні засоби IDE:

– текстовий редактор – IDE мають вбудований текстовий редактор, який можна використовувати для написання коду;

– компілятор – IDE використовують компілятор для перетворення коду в машинний код, який може бути виконаний комп’ютером;

– відладчик – IDE мають вбудований відладчик, який можна використовувати для виявлення і усунення помилок у коді;

– засоби для тестування – IDE мають вбудовані засоби для тестування, які можна використовувати для перевірки роботи програми [1, 4].

Visual Studio Code, JetBrains WebStorm і Eclipse – це популярні IDE, які допомагають розробникам створювати вебзастосунки. Вони мають різні можливості та зручності, які можуть вподобатися різним категоріям розробників.

Visual Studio Code – це безкоштовна та відкрита IDE, яка підтримує широкий спектр мов програмування, включаючи JavaScript, TypeScript, HTML і CSS. Вона має простий і інтуїтивно зрозумілий інтерфейс користувача, а також широкий спектр розширень, які можна використовувати для додавання додаткових функцій і можливостей [21].

JetBrains WebStorm – це платна IDE, яка спеціально розроблена для розробки вебзастосунків. Вона має широкий спектр функцій, які можуть бути корисними для розробників вебзастосунків, включаючи автозаповнення коду, інтелектуальне підсвічування синтаксису і відладку.

Eclipse – це безкоштовна і відкрита IDE, яка підтримує широкий спектр мов програмування, включаючи Java, JavaScript, HTML і CSS. Вона має розширений набір функцій, який може бути корисним для розробників вебзастосунків, включаючи підтримку різних фреймворків і технологій.

Вибір IDE в першу чергу залежить від мови програмування, яку використовуєте розробник, та його особистих уподобань [4].

1.2 Інструменти для версіонування коду

Системи контролю версій (source code management, SCM) – це програмне забезпечення, яке використовується для відстеження змін у коді. Вони дозволяють розробникам зберігати різні версії коду, а також відстежувати, хто вносить зміни і коли.

Git є найпоширенішим та найбільш ефективним SCM у сучасній розробці, але є інші інструменти, які також можуть бути корисними.

Git є дуже ефективною і швидкою системою керування версіями, здатним ефективно обробляти великі кодові бази та велику кількість різних гілок. Git має велику кількість інтеграцій з різними інструментами розробки вебзастосунків, такими як GitHub, GitLab, Bitbucket, і багатьма іншими. Ці інструменти дозволяють спростити спільну роботу над кодом, код-рев'ю, відстеження завдань та автоматизацію процесів CI/CD [5].

Apache Subversion (SVN) – ще одна система керування версіями з відкритим вихідним кодом, яка прагне бути найкращою широко використовуваною системою керування версіями. Це надійний варіант для цінних даних. Він безкоштовний з точки зору витрат на ліцензування, але стягує певну плату за деякі функції. Це система контролю версій програмного забезпечення та версій, яку розробники використовують для керування версіями файлів для вебсайтів. SVN також має інтеграції з різними інструментами, але їхні можливості не такі багатогранні, як у Git.

Mercurial – це безкоштовна розподілена система контролю версій для керування та відстеження змін, внесених у проєкти командою програмного забезпечення. Такі популярні організації, як Mozilla, World Web Consortium (W3) і Facebook, використовують Mercurial як свою систему контролю версій. Найкраща частина цієї системи контролю версій полягає в тому, що вона ефективно обробляє будь-який розмір проєкту та пропонує простий та інтуїтивно зрозумілий інтерфейс. Mercurial має інтеграції з різними

інструментами розробки, але вона може бути менш популярною та менш розширеною, ніж Git.

Team Foundation Version Control (TFVC) – це централізована система керування версіями, яка використовується головним чином в екосистемі Microsoft. Вона може бути ефективною для проєктів, що використовують інші інструменти Microsoft, такі як Visual Studio Code.

TFVC має глибоку інтеграцію з іншими продуктами Microsoft, але може бути менш сумісною з іншими інструментами.

Усі ці системи керування версіями мають свої власні переваги та недоліки, і вибір залежить від специфічних потреб проєкту та вподобань. Велика частина сучасних веброзробників використовує Git через його популярність та потужність, а також через широкий вибір інтеграцій з іншими інструментами розробки вебзастосунків [1, 5].

1.3 Системи управління базами даних

Система управління базами даних (СУБД) – це програмне забезпечення, яке дозволяє організовувати та управляти зберіганням, оновленням та отриманням даних з бази даних. Є кілька типів СУБД, і кожен з них має свої характеристики та сфери застосування.

Реляційні системи управління базами даних (РСУБД) – це тип баз даних, який використовує модель даних у вигляді таблиць (реляційних таблиць) для зберігання та обробки даних. Кожна таблиця складається з рядків і стовпців, а кожен рядок представляє собою запис даних.

Основні характеристики реляційних СУБД включають можливість встановлення відносин між таблицями, нормалізацію даних для забезпечення їх консистентності і уніфікації, а також використання мови структурованих запитів (зазвичай SQL) для взаємодії з базою даних.

Реляційні СУБД використовують табличну структуру для зберігання даних, де дані представлені у вигляді таблиць з рядками і стовпцями. Основними представниками є MySQL, PostgreSQL, Oracle, SQL Server тощо.

Реляційні СУБД мають драйвери для багатьох мов програмування та часто використовуються в вебзастосунках. Вони підходять для застосунків, де важлива структурованість та цілісність даних, такі як системи управління контентом, електронна комерція, інформаційні панелі тощо.

Популярні реляційні СУБД включають MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database та SQLite.

NoSQL СУБД призначені для роботи з неструктурованими, півструктурованими або нереляційними даними. Вони використовують різні моделі даних, такі як документ-орієнтована (MongoDB), ключ-значення (Redis), колоночна (Apache Cassandra) та інші. NoSQL СУБД підходять для вебзастосунків, де потрібна гнучкість у схемі даних або обробка великого обсягу даних. Вони часто використовуються для розробки вебзастосунків з нереляційною структурою даних, таких як соціальні мережі, ігри та аналітичні застосунки.

In-Memory СУБД зберігають дані в оперативній пам’яті (RAM), що забезпечує швидкий доступ до них. Приклади включають Redis та Memcached. In-Memory СУБД використовуються для вебзастосунків, де важлива швидкість доступу до даних, таких як кешування, сесійні сховища, та високонавантажені системи.

Колоночні СУБД орієнтовані на оптимізацію зберігання колоночних даних. Прикладами є Apache Cassandra та HBase. Колоночні СУБД підходять для вебзастосунків з великим обсягом даних та вимогами до розподіленості, таких як системи журналювання подій або аналітичні системи.

Графові СУБД спеціалізуються на зберіганні та оптимізації графових даних, де важливі відносини між об’єктами. Прикладами є Neo4j та Amazon Neptune. Графові СУБД використовуються для вебзастосунків, які

потребують обробки графових даних, таких як соціальні мережі, рекомендаційні системи або аналіз залежностей [6].

1.4 Інструменти для тестування та відлагодження

Тестування вебзастосунків виступає важливим етапом розробки, спрямованим на забезпечення якості та надійності програмного продукту. Цей процес включає в себе ряд дій, спрямованих на виявлення помилок, підтримку високої якості коду, забезпечення безпеки, оптимізацію продуктивності та відповідність вимогам користувачів.

Виявлення помилок в коді та їх виправлення перед впровадженням в продакшн сприяє стабільності та надійності вебзастосунку. Такий підхід гарантує ефективну роботу продукту та позитивний досвід використання для кінцевих користувачів.

Забезпечення якості коду є основним завданням тестування, оскільки висока якість дозволяє забезпечити оптимальну продуктивність та уникнути непередбачуваних проблем.

Тестування також допомагає виявити та виправити потенційні вразливості вебзастосунку, забезпечуючи його захист від можливих загроз та зловживань.

Оптимізація продуктивності через тестування дозволяє виявити та усунути проблеми, пов’язані з швидкістю завантаження сторінок та роботою застосунків, що призводить до покращення досвіду користувача.

Тестування також гарантує відповідність вебзастосунку вимогам та очікуванням користувачів, сприяючи розробці продукту, який задовольняє їхні потреби та очікування.

Інструменти для автоматизованого тестування вебзастосунків включають такі популярні засоби як Selenium, Cypress, Jest, Jenkins та Postman.

Selenium – це широко використовуваний інструмент для автоматизованого тестування вебзастосунків. Він надає можливість записувати і відтворювати тестові сценарії на різних мовах програмування, таких як Java, Python, C# та інші. Також підтримує різні браузери, включаючи Chrome, Firefox, Safari і інші.

Cypress – інструмент, спеціалізований на автоматизованому тестуванні вебінтерфейсів. Він відрізняється простим API та інтеграцією з іншими інструментами розробки. JavaScript є основною мовою програмування, яку використовує Cypress. Він підтримує браузери, такі як: Chrome, Firefox та Edge.

Jest – це фреймворк для автоматизованого тестування JavaScript-застосунків. Використовується для тестування фронтенду вебзастосунків, підтримує як JavaScript, так і TypeScript, і зазвичай використовується для тестування на рівні компонентів та функціональності.

Jenkins – це система для автоматизації процесів розробки, включаючи автоматизоване тестування. Вона допомагає запускати тести при кожному коміті коду і сповіщати про результати. Хоча Jenkins не обмежений конкретними мовами програмування, його часто використовують для автоматизації різних задач.

Postman – інструмент для автоматизованого тестування API. За допомогою Postman можна створювати, відправляти та перевіряти запити до API і перевіряти їх відповіді. Використовується для тестування API та не обмежується конкретними мовами програмування.

Відлагодження коду – це процес пошуку та усунення помилок у програмному коді. Існує безліч інструментів для відлагодження коду, які можуть допомогти розробникам знайти та усунути помилки.

Необхідність відлагодження:

– допомагає виявляти та усувати помилки та дефекти в програмному коді;

– забезпечує коректну роботу застосунку та відповідність функціональності специфікаціям;

– зменшує час, необхідний для розробки, оскільки дозволяє ефективно знаходити та виправляти помилки.

Інструменти для відлагодження коду включають інтегровані середовища розробки (IDE), такі як Visual Studio Code для різних мов програмування, включаючи C++, C#, Python, і інші. Eclipse є популярним для Java-розробки, а IntelliJ IDEA, хоча спеціалізований на Java, також підтримує інші мови.

Відладчики, такі як GDB для C і C++ або pdb для Python, допомагають відлагоджувати код, спостерігаючи за його виконанням. Visual Studio Debugger інтегрований у Visual Studio і підтримує відлагодження .NET-коду.

Логування подій корисне для відстеження виконання програми. Наприклад, Log4j та Logback використовуються для логування в Java, а Python Logging – для Python-застосунків.

Інструменти для профілювання, такі як VisualVM для Java або cProfile для Python, допомагають визначити проблемні ділянки коду, які можуть негативно впливати на продуктивність програми.

Інші корисні інструменти включають Sentry та Bugsnag для виявлення та відстеження помилок, ELK Stack (Elasticsearch, Logstash, Kibana) для аналізу логів, Wireshark для аналізу мережевого трафіку, Postman для тестування API та Git для відстеження змін в коді та спільної роботи над проєктами [4, 7, 8].

1.5 Засоби безпеки вебзастосунків

Безпека вебзастосунків є важливою для захисту даних користувачів, інформації про бізнес та довіри користувачів. Коли вебзастосунок не є

безпечним, користувачі можуть втратити свої особисті дані, такі як імена, адреси електронної пошти та номери кредитних карток.

Одним з найважливіших аспектів безпеки вебзастосунків є виявлення та виправлення вразливостей. Вразливості – це недоліки в програмному коді, які можуть бути використані зловмисниками для отримання несанкціонованого доступу до системи.

Нижче наведені інструментіи для виявлення вразливостей вебзастосунків.

Nmap – це інструмент для сканування мережі, який може використовуватися для виявлення вразливостей вебзастосунків. Вона використовується для визначення доступних хостів у мережі, виявлення відкритих портів і служб, визначення операційних систем і версій ОС, виявлення типів пакетних фільтрів/брандмауерів та ін.

Nmap використовує «сирі» IP-пакети для визначення стану мережі. Він генерує різні типи пакетів, щоб визначити, які хости доступні, які порти відкриті, які служби запущені та яка операційна система використовується.

Nmap має широкий набір опцій, які дозволяють користувачам налаштувати сканування відповідно до своїх потреб. Наприклад, можна вказати, які порти сканувати, яку операційну систему шукати та як уникнути виявлення скануванням.

Nikto – це безкоштовна та відкрита утиліта для сканування вебсайтів на наявність вразливостей. Він використовує базу даних відомих вразливостей, щоб сканувати вебсайти на наявність потенційних проблем безпеки.

Nikto використовує різні методи сканування, щоб виявити вразливості, включаючи: сканування HTTP-заголовків, сканування HTTP-кодів стану, сканування файлів і каталогів, сканування форм і скриптів.

OWASP ZAP (Zed Attack Proxy) – це безкоштовний та відкритий інструмент для тестування безпеки вебзастосунків. Він використовується для виявлення вразливостей у вебзастосунках, таких як Cross-Site Scripting (XSS), SQL Injection, Cross-Site Request Forgery (CSRF) та інші.

Ще одним важливим аспектом безпеки вебзастосунків є обмеження доступу до даних. Дані, які не є необхідними для виконання функції, повинні бути захищені від несанкціонованого доступу [1, 9].

Способи обмеження доступу до даних:

– використання ролей та дозволів (ролі та дозволи дозволяють надавати користувачам доступ лише до тих даних, які їм необхідні);

– шифрування даних (робить їх нечитабельними для несанкціонованих користувачів);

– використання брандмауера (може бути використаний для блокування несанкціонованого доступу до системи).

Крім виявлення та виправлення вразливостей та обмеження доступу до даних, існує безліч інших інструментів та практик, які можна використовувати для забезпечення безпеки вебзастосунків.

Використання сучасних протоколів і стандартів безпеки, таких як HTTPS і TLS, можуть допомогти захистити вебзастосунок від атак.

Використання безпечних методів розробки, таких як кодування за принципом «безпека по замовчуванню», можуть допомогти запобігти вразливостям.

Використання системи управління ризиками безпеки може допомогти організації визначити, оцінити та вжити заходів щодо ризиків безпеки [9].

1.6 Інтеграція та розгортання

У контексті сучасного середовища розробки, створення та підтримка вебзастосунків вимагають систематичного та ефективного підходу до процесів інтеграції коду та розгортання програмних засобів на серверах. Забезпечення стабільності, безпеки та швидкості впровадження нового функціоналу вимагає використання інструментів та практик автоматизації.

Інструмент Jenkins є одним із найпопулярніших інструментів для автоматизації процесу Continuous Integration (CI) та Continuous Deployment (CD). Він відрізняється широким спектром плагінів і можливостей для конфігурації та управління конвеєрами CI/CD. Jenkins надає розробникам засіб для автоматизації процесів тестування, збирання та розгортання [7].

Платформа Travis CI – це хмарний інструмент для автоматизації CI/CD, який інтегрується з репозиторіями на платформі GitHub. Завдяки простій конфігурації через файл .travis.yml, розробники можуть швидко налаштувати інтеграцію та автоматизувати тестування вебзастосунків.

Travis CI є платформою для неперервної інтеграції та неперервної доставки (CI/CD), яка дозволяє автоматизувати процеси тестування та розгортання програмного забезпечення. Вона легко інтегрується з репозиторіями на платформах для управління версіями коду, таких як GitHub та GitLab, і надає зручні інструменти для виконання тестів.

Однією з основних функцій Travis CI є неперервна інтеграція, що дозволяє автоматично запускати тести при кожній зміні в коді. Це допомагає розробникам виявляти помилки та проблеми раніше, сприяючи покращенню якості коду та ефективності розробки.

Платформа також підтримує неперервну доставку, автоматично розгортаючи програмне забезпечення після успішного проходження тестів. Це дозволяє швидше впроваджувати зміни та надавати нові версії програми, сприяючи швидкому циклу розробки.

Однією з переваг Travis CI є можливість конфігурації у вигляді коду. Розробники можуть використовувати файл конфігурації, який легко включається в репозиторій, для налаштування роботи CI/CD процесів з урахуванням потреб конкретного проєкту.

Також, платформа підтримує різні мови програмування та середовища розробки, що робить її гнучкою і придатною для різноманітних проєктів та команд розробників. Завдяки Travis CI, розробники можуть автоматизувати

тестування та розгортання, спрощуючи та прискорюючи процес розробки програмного забезпечення.

CircleCI – це інша платформа для неперервної інтеграції та неперервної доставки (CI/CD), яка забезпечує автоматизацію процесів розробки програмного забезпечення. Як і Travis CI, вона інтегрується з репозиторіями на таких платформах, як GitHub та Bitbucket, надаючи розробникам можливість автоматизувати тести та процес розгортання.

Основна функціональність CircleCI включає в себе неперервну інтеграцію, що дозволяє автоматично виконувати тести при кожній зміні в коді. Це сприяє виявленню помилок та недоліків на ранніх етапах розробки, полегшуючи їх виправлення.

Крім того, CircleCI підтримує неперервну доставку, автоматично розгортаючи програмне забезпечення після успішного проходження тестів. Це сприяє швидкому впровадженню змін та видачі нових версій програми.

CircleCI дозволяє використовувати конфігурацію у вигляді коду, що полегшує налаштування та управління процесами CI/CD. Це дає розробникам більше гнучкості та контролю над автоматизованими робочими процесами.

Загалом, CircleCI є інструментом, який допомагає розробникам прискорити розробку програмного забезпечення та покращити якість коду через автоматизацію тестів та розгортання [24].

Використання контейнеризації, наприклад, за допомогою Docker, дозволяє ізолювати програмні засоби та їх залежності, що спрощує процес збирання та розгортання.

Інтеграція інструментів автоматизації з системами керування версіями, такими як Git, надає можливість спостерігати за версіями коду, забезпечуючи єдність розробки та процесу розгортання.

Практика конфігурації інфраструктури через код (Infrastructure as Code, IaC) дозволяє визначити інфраструктуру серверів, баз даних та інших компонентів у вигляді коду, що підвищує швидкість та надійність розгортання.

Використання інструментів, таких як Prometheus або ELK Stack, дозволяє збирати та аналізувати дані про роботу систем. Ці інструменти можуть бути використані для моніторингу продуктивності, виявлення проблем та оптимізації систем.

Prometheus – це система моніторингу, яка збирає дані у вигляді метрик. Метрики – це ключові показники ефективності (KPI), які вимірюють стан системи. ELK Stack – це стекове програмне забезпечення для керування журналами, яке збирає, зберігає та аналізує журнали. Журнали – це записи подій, які відбуваються в системі.

Використання автоматизованих тестів та механізмів валідації дозволяє перевіряти працездатність та відповідність застосунку визначеним вимогам. Це забезпечує впевненість у якості коду та його здатності до безперебійної роботи в різних умовах.

Використання інструментів для автоматизованого аналізу коду на вразливості та визначення найкращих практик є важливою складовою практики CI/CD з точки зору безпеки.

Однією з основних цілей цієї практики є максимальне скорочення часу від написання коду до впровадження його в продакшн. Автоматизація всіх етапів розробки, тестування та розгортання допомагає прискорити цей цикл та забезпечити швидку реакцію на зміни [24, 29].

1.7 Постановка задачі дослідження

Аналіз предметної області показав, що розроблення вебзастосунків є актуальним і розвивається в значних масштабах. Сучасні програмні засоби для розроблення вебзастосунків надають широкий спектр функціоналів, який допомагає розробникам створювати ефективні та масштабовані вебзастосунки для різних галузей впровадження.

Об’єктом дослідження є функціонали для розробки вебзастосунків на прикладі Next.js.

Метою дослідження є оцінка можливостей та особливостей функціоналів, які пропонує Next.js для розробки вебзастосунків та їх вплив на продуктивність та розширюваність проєктів.

Для досягнення мети необхідно вирішити такі завдання:

– здійснити аналіз функціональних можливостей Next.js;

– створити простий застосунок на Next.js;

– розглянути інструменти для розширення функціональності;

– здійснити аналіз документації та спільноти;

– сформулювати висновки та рекомендації щодо впровадження.

2 ОБҐРУНТУВАННЯ І ВИБІР ЗАСОБІВ ДЛЯ РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ

2.1 Вимоги та вибір засобів для розроблення вебзастосунків

У даному розділі кваліфікаційної роботи фокус дослідження ставиться на визначенні основних характеристик та параметрів, які будуть визначальними для успішного розроблення вебзастосунку за допомогою Next.js.

Починаючи з загальних принципів, таких як зручність взаємодії та адаптивний дизайн, розглянуто конкретні вимоги, спрямовані на використання Next.js у контексті розроблення вебзастосунку. Вимоги будуть визначати технічні, функціональні аспекти, необхідні для створення вебзастосунку, який буде відповідати високим стандартам ефективності, користувацького досвіду та зручності управління контентом.

Загальні сучасні вимоги до вебзастосунку:

– має бути розроблений на сучасних інструментах та технологіях;

– має відповідати сучасним стандартам веброзробки;

– має бути безпечним та надійним;

– має бути адаптивним до різних пристроїв.

Конкретні прикладні вимоги до вебзастосунку:

– має бути промо-сайтом для нового продукту на базі Next.js, відповідно до завдань дослідження та розроблення в рамках кваліфікаційної роботи;

– використання можливостей Next.js для SEO оптимізації;

– використання можливостей Next.js для динамічного завантаження контенту, що дозволить показувати актуальну та змінну інформацію;

– має містити головну сторінку з інформацією про продукт, його переваги та можливості;

– має містити сторінку для відображення списку публікацій;

– має містити сторінку для детального перегляду окремої публікації за ідентифікатором;

– має містити сторінку для відображення списку користувачів та їх інформації;

– має містити сторінку з контактною інформацією.

На рисунку 2.1 схематично зображена структура промо-сайту.

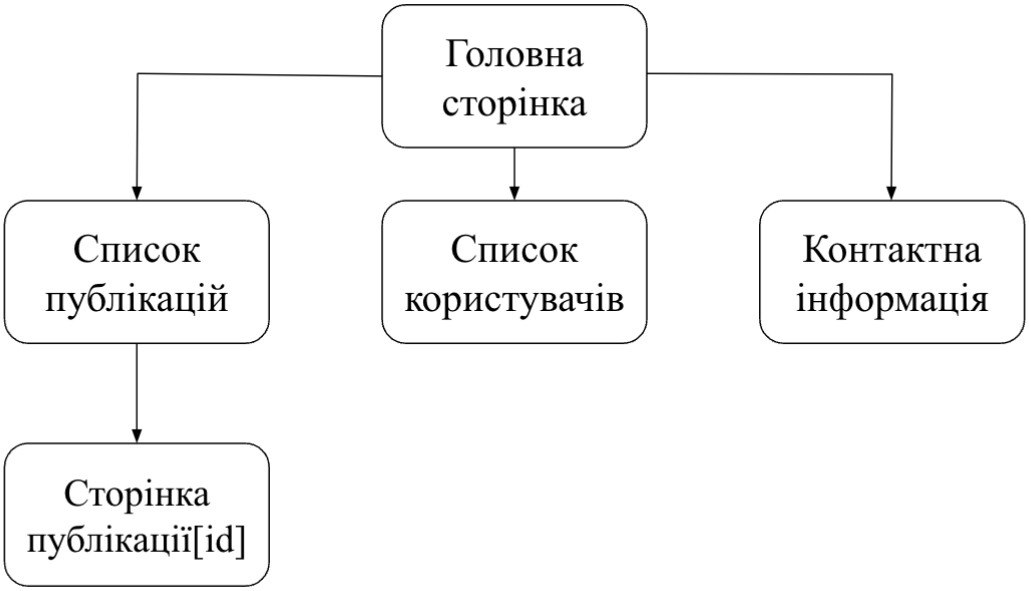

Рисунок 2.1 – Схема промо-сайту

У цьому розділі розглядається вибір технологічного стеку для розроблення вебзастосунку. В ньому було проаналізовано різні аспекти та обґрунтовано вибір конкретних інструментів, мов програмування, та інших технологій, які найбільше відповідають потребам проєкту.

Технічний стек – це набір програмного забезпечення, мов програмування, фреймворків та інструментів зберігання даних, які розробники можуть використовувати для створення та запуску однієї програми. На рисунку 2.2 зображено компоненти технічного стеку типового вебзастосунку [14].

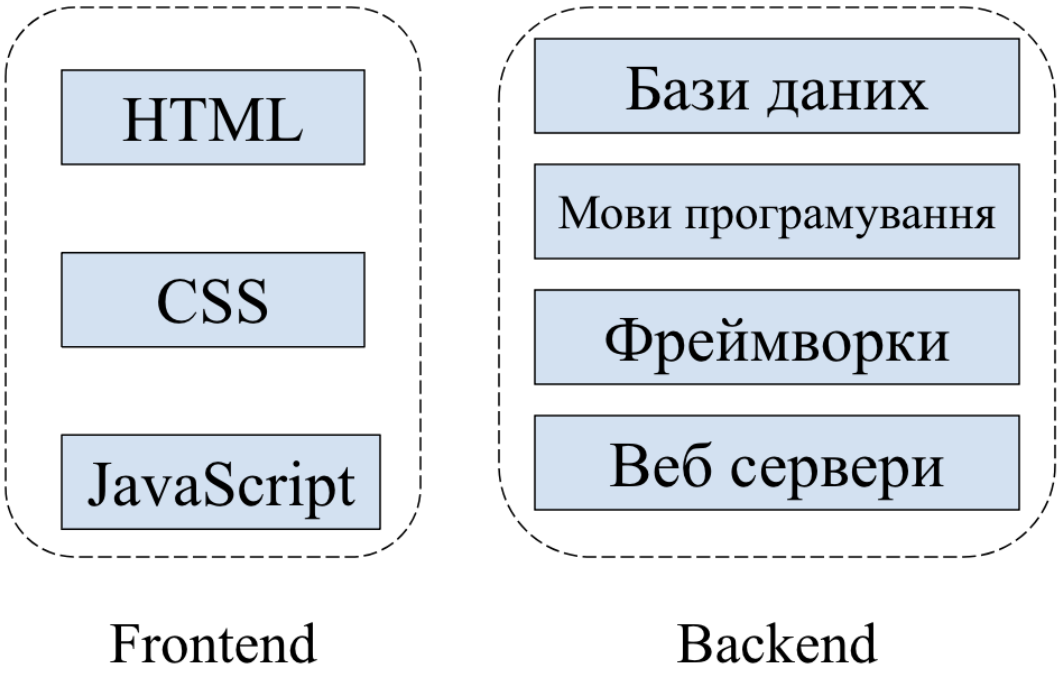

Рисунок 2.2 – Компоненти технічного стеку вебзастосунку

Клієнтська сторона програми представлена fronend технічним стеком. Все, що користувач може переглядати або з чим взаємодіяти на екрані, називається клієнтським. Основна мета fronend стеку – створити високу якість користувацького досвіду, плавний інтерфейс користувача та прості внутрішні структури. Іншими словами, він відповідає за дизайн, форматування та навігацію вебсайтів і вебзастосунків.

Основними компонентами frontend стеку:

– HTML – використовується для створення та відображення вебсторінок;

– CSS – відповідає за дизайн та організацію вебсторінок;

– JavaScript – для створення інтерактивних вебсторінок.

Стек backend технологій у розробці програмного забезпечення є серверним. Він описує внутрішню роботу вебсайту або програми.

Компоненти backend стеку:

– мови програмування (Java, PHP, Python);

– фреймворки (Ruby on Rails, Django, Laravel);

– вебсервери (Apache, Nginx, Microsoft Internet Information Server);

– бази даних (MySQL, PostgreSQL, MongoDB).

Ці мови програмування, фреймворки, технології, сервери та бази даних накладаються зверху одна на одну в процесі веброзробки. Звідси і термін «технічні стеки».

У процесі розгляду та вибору мови програмування для реалізації вебзастосунку були враховані низка факторів, що визначають успішну інтеграцію з фреймворком Next.js та відповідність вимогам проєкту. При цьому було приділено увагу як загальним, так і конкретним вимогам, які включають у себе необхідність аналізу можливостей функціоналу Next.js, оскільки проєкт на даному етапі представляє собою простий промо-сайт та має мету швидкої реалізації прототипу для подальшого використання у кваліфікаційній роботі.

З урахуванням зазначених вимог та особливостей проєкту прийнято рішення використовувати комбінацію мов програмування, таких як JavaScript, TypeScript, HTML, CSS разом з CSS-фреймворком Tailwind. Ці технології вже давно є стандартними для розробки вебсайтів і визначають оптимальний підхід до швидкої та ефективної реалізації всіх необхідних функцій.

JavaScript обрано як основну мову сценаріїв через його високу ефективність у додаванні інтерактивності до вебсторінок. Він дозволяє реалізувати анімацію, обробку подій та взаємодію з користувачем, відповідаючи конкретним вимогам проєкту.

TypeScript, як розширення JavaScript, використовується для покращення читабельності та продуктивності коду за рахунок статичного типування. Однією з основних переваг TypeScript є можливість визначати типи для змінних, параметрів функцій та інших об’єктів. Це дозволяє виявляти помилки ще до того, як код виконається, що полегшує виявлення та виправлення помилок у ранніх етапах розробки. TypeScript сприяє удосконаленню кодової бази та робить розробку більш надійною.

CSS та HTML є основними складовими для визначення структури та стилю вебсторінок. CSS відповідає за форматування зовнішнього вигляду елементів, таких як колір, шрифт та розмір, тоді як HTML використовується для структурування контенту сторінки. Використання Tailwind CSS додає додатковий рівень ефективності до процесу стилізації, прискорюючи його завдяки готовим класам, які визначають широкий спектр стилів без необхідності написання власного CSS коду. Це дозволяє розробникам швидко та зручно стилізувати елементи і фокусуватися на інших аспектах розробки, підвищуючи продуктивність та покращуючи якість коду [11].

Однією з основних переваг обраного стеку технологій є активне використання JavaScript/TypeScript у Next.js, що гарантує оптимальну інтеграцію та використання всіх можливостей фреймворку. Такий підхід відповідає вимогам проєкту, забезпечуючи оптимізовану реалізацію SEO та можливостей динамічного завантаження контенту [23].

2.2 Вибір системи контролю версій

Вибираючи систему керування версіями (SCM), важливо враховувати функції, які вона пропонує. Загальні функції, на які слід звернути увагу, включають розгалуження та злиття, що дозволяє створювати паралельні версії коду для різних цілей, а потім об’єднувати їх назад у основну гілку. Розв’язання конфліктів також корисно для вирішення розбіжностей або помилок під час об’єднання коду з різних джерел або гілок. Крім того, функції історії та відкату дозволяють переглядати історію змін коду та за потребою повертатися до попереднього стану. Теги та мітки дозволяють позначати конкретні версії коду ідентифікаторами, такими як номери випусків, дати або етапи, а функції перегляду коду та співпраці полегшують спілкування та зворотний зв’язок між розробниками.

Оцінюючи систему керування версіями (SCM), необхідно враховувати її сумісність з іншими інструментами та платформами, які використовуються в проєкті. Деякі SCM розроблено для певних мов або фреймворків, забезпечуючи кращу підтримку або функціональність. Крім того, деякі мають плагіни або розширення, які дають змогу отримувати до них доступ і використовувати їх безпосередньо з бажаного IDE або редактора. Крім того, деякі SCM можуть автоматизувати або оптимізувати процеси тестування, створення та розгортання коду в різних середовищах або платформах. Нарешті, деякі SCM можуть пов’язувати або синхронізувати зміни коду з інструментами управління проєктом або спілкуванням, такими як трекери проблем, диспетчери завдань або програми для чату.

Важливо знайти SCM із чіткою, вичерпною та оновленою документацією та підтримкою, такою як підручники, посібники, поширені запитання, форуми чи блоги. Крім того, інтерфейс користувача та команди мають бути зручними та інтуїтивно зрозумілими. Необхідно шукати SCM із великою та активною спільнотою та популярністю, щоб забезпечити надійність, стабільність і доступ до додаткових ресурсів і допомоги, якщо це необхідно [1, 5].

Після вивчення різних систем керування версіями вирішено використовувати Git для системи контролю версій у проєкті.

Git є легким у використанні і не вимагає значного завантаження конфігурацій для початку роботи. Можливість створення та об’єднання гілок в Git стане корисною при індивідуальному експериментуванні та розробці різних аспектів проєкту. З урахуванням невеликого розміру проєкту, швидкість роботи Git дозволить швидко виконувати операції з контролю версій. Git легко інтегрується з різними інтегрованими середовищами розробки, надаючи можливість ефективно використовувати його у роботі [5].

Такий вибір гарантує зручність управління версіями коду та відповідає особливостям індивідуального розроблення у рамках проєкту.

На рисунку 2.3 відображено інтеграцію системи контролю версій Git та інтегрованого середовища розробки Visual Studio Code. Ця інтеграція дозволяє ефективно взаємодіяти з Git-репозиторієм безпосередньо в середовищі розробки, спрощуючи процес роботи з версіями коду та забезпечуючи безпечне та зручне ведення історії змін.

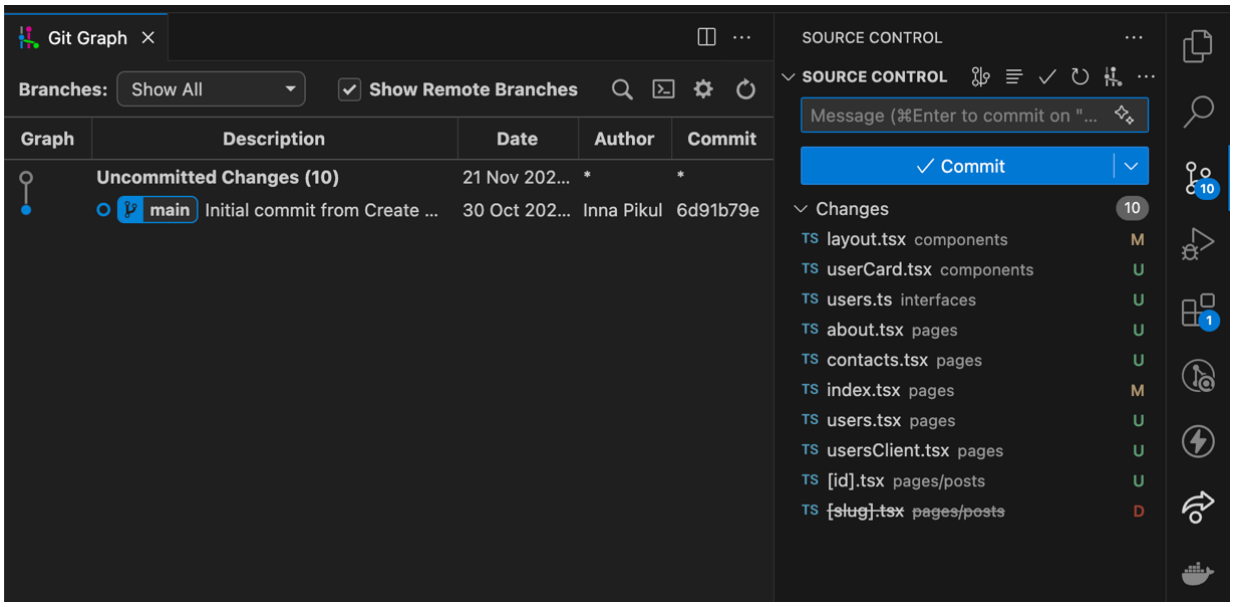

Рисунок 2.3 – Інтеграція Git в середовищі розробки Visual Studio Code

2.3 Вибір IDE

Веброзробка вимагає від розробників ефективних інструментів та інтегрованих середовищ розробки (IDE), які спрощують процес створення вебзастосунків.

Одним із найважливіших факторів, який слід враховувати при обранні IDE, є мова програмування, яку обрано або планується використовувати в проєкті. Різні IDE підтримують різні мови, і деякі можуть пропонувати більше можливостей і функцій для певних мов, ніж інші.

Наприклад, при розробці вебзастосунку з використанням HTML, CSS і JavaScript, слід звертати увагу на IDE, у якому реалізовано підсвічування

синтаксису, автозавершення коду, інструменти налагодження та попередній перегляд для цих мов. З іншого боку, при роботі зі скомпільованою мовою, такою як C++, може знадобитися IDE, в якому інтегровано компілятор, компонувальник і налагоджувач. Треба також перевірити, чи підтримує обране IDE фреймворки, бібліотеки та розширення, які планується використовувати в проєкті.

Ще одним фактором, який слід враховувати при обранні IDE, є операційна система, яку обрано або націлено на конкретний проєкт. Деякі IDE є кросплатформними, тобто вони можуть працювати на кількох операційних системах, таких як Windows, Linux і Mac OS. Однак деякі IDE є специфічними для однієї операційної системи або можуть мати різні функції та продуктивність в залежності від операційної системи.

Наприклад, при розробці нативної програми для iOS, може знадобитися використовувати Xcode, який є IDE, розробленою лише для MacOS. Крім того, якщо розробляється кросплатформна програма для мобільних пристроїв, можна використовувати IDE, яку підтримано для кількох платформ, наприклад, Visual Studio Code або Flutter.

Залежно від обсягу та масштабу проєкту може бути використано IDE, яка працює з великими файлами, кількома файлами та складними залежностями. Наприклад, якщо розробляється велика корпоративна програма з численними модулями, компонентами і рівнями, можна використати IDE з такими функціями, як навігація кодом, рефакторинг, контроль версій, тестування та розгортання. З іншого боку, при роботі над невеликим чи простим проєктом, який має кілька файлів і залежностей, можна віддати перевагу легкій, швидкій та простій у використанні IDE.

Ще одним фактором, який слід враховувати при обранні IDE, є особисті переваги та стиль. Різні IDE мають різні інтерфейси користувача, макети, теми та налаштування, які можуть сподобатися різним користувачам. Наприклад, деяким користувачам може сподобатися IDE з мінімалістичним і чистим дизайном, тоді як інші можуть віддати перевагу IDE з великою

кількістю вкладок, панелей і меню. Деякі користувачам може сподобатися IDE, що має багато функцій і параметрів, тоді як інші можуть вважати їх приголомшливими та відволікаючими. Деяким користувачам може сподобатися IDE, що має багато ярликів і команд, тоді як інші можуть віддати перевагу IDE, яка має більш інтуїтивно зрозумілий і графічний інтерфейс. Зрештою, потрібно вибрати IDE, яка відповідає робочому процесу, звичкам і вподобанням.

Різні IDE мають різні витрати та вимоги, які можуть вплинути на рішення. Деякі IDE є безкоштовними та мають відкритий вихідний код, тобто можна завантажувати та використовувати їх без сплати будь-яких комісій чи ліцензій. Однак деякі IDE є платними та пропрієтарними, тобто їх потрібно придбати або підписатися на них, щоб отримати доступ до повних функцій і можливостей. Деякі IDE також потребують більше ресурсів, ніж інші, тобто вони можуть споживати більше пам’яті, дискового простору та обчислювальної потужності комп’ютера. Треба порівняти витрати та переваги різних IDE і визначити, яка з них відповідає бюджету та ресурсам.

Останнім фактором, який слід враховувати при обранні IDE, є відгуки та огляди інших користувачів і експертів. Можна багато чому навчитися з досвіду та думок інших розробників, які використовують різні IDE для своїх проєктів. Можна прочитати онлайн-огляди, рейтинги, порівняння та рекомендації різних IDE і дізнатися, що вони мають сказати про свої сильні та слабкі сторони, переваги та недоліки. Також можна приєднатися до онлайн-форумів, спільнот і груп розробників, які використовують схожі з вами мови, фреймворки та платформи, і запитати у них поради та пропозиції. Також можна випробувати різні IDE для себе та побачити, яка з них найкраще підходить для конкретних потреб [1, 25].

Visual Studio Code має широкий набір розширень для роботи з різними мовами програмування. Підтримує інтеграцію з Git для ефективного керування версіями.

WebStorm – спеціалізована IDE для веброзробки з підтримкою JavaScript, TypeScript, HTML та CSS. Має інтегрований відлагоджувач та інструменти для роботи з Node.js.

Atom. Відкритий та налаштовуваний текстовий редактор, розроблений для легкої веброзробки. Підтримує велику кількість плагінів та розширень.

Sublime Text Легкий та швидкий текстовий редактор, особливо популярний серед розробників. Має велику спільноту та широкий вибір плагінів [4, 7, 21].

Після докладного огляду різних IDE для веброзробки вирішено використовувати Visual Studio Code для проєкту.

Visual Studio Code має розгалужений екосистему розширень, що дозволяє адаптувати робочий процес до потреб розробника. Підтримка багатьох мов програмування зокрема JavaScript і TypeScript, робить його ідеальним для проєкту, який базується на цих технологіях. Підтримка Git дозволяє ефективно керувати версіями коду та спрощує спільну роботу над проєктом. Visual Studio Code має активну спільноту розробників і постійно оновлюється, що забезпечує підтримку та розвиток інструменту. Вибір Visual Studio Code визначено не лише його функціональністю та продуктивністю, але й можливістю найкраще відповісти конкретним потребам проєкту.

2.4 Вибір системи управління базами даних

Вибір системи управління базами даних є важливим етапом у процесі розробки вебзастосунку, оскільки від цього вибору залежить ефективність роботи з даними та подальша розширюваність системи. Розглянемо основні різновиди СУБД.

Relational Database Management System (RDBMS) використовує модель даних у яких дані впорядковані в таблиці рядків, кожна таблиця пов’язана з іншими. Кожен блок даних, який має своє фіксоване місце та значення в базі

даних, забезпечує просту та чітку структуру для розробників програмного забезпечення. Призначений для проєктів, де важлива структурованість та надійність даних. До таких СУБД належать: PostgreSQL, MySQL.

NoSQL (Not Only SQL) бази даних пропонують гнучкий підхід до зберігання та витягування даних, не обов'язково використовуючи SQL мову. Ідеальний вибір для проєктів з різнорідними та змінними даними. До таких СУБД належать: MongoDB, Redis.

Graph Database спеціалізовані у зберіганні та операціях з графовими даними, які мають взаємозв'язки. Використовується для проєктів, де важливо відтворення взаємозв'язків між об'єктами. До таких СУБД належить: Neo4j, ArangoDB, Dgaph [6, 28].

Після аналізу різновидів систем управління базами даних (СУБД) та урахування специфіки проєкту, було обрано PostgreSQL як оптимальний варіант для вебзастосунку. Зазначений вибір ґрунтується на відповідності PostgreSQL основним критеріям, що визначені вимогами до бази даних даного проєкту.

Однією з основних вимог є структурованість даних, PostgreSQL відповідає цьому критерію, забезпечуючи організацію і зберігання даних у вигляді таблиць та дозволяючи ефективну роботу з ними через мову запитів SQL. Додатково, база даних підтримує виразні індекси, геопросторові запити та інші розширені функціональності, що забезпечує можливість оптимізації та розширення функціональності в майбутньому.

Використання PostgreSQL є безкоштовним, а також має велику активну спільноту розробників, що надає можливість отримання підтримки. При цьому, враховуючи її здатність до ефективного масштабування, PostgreSQL гарантує оптимальну продуктивність бази даних із зростанням обсягу даних.

Підтримка Node.js бібліотеки PostgreSQL відображається в його гармонійній інтеграції з середовищем розробки Next.js, що базується на Node.js. Це забезпечує зручність та ефективність взаємодії між вебзастосунком та базою даних [26].

Отже, обрана база даних PostgreSQL відповідає вимогам проєкту, забезпечуючи структурованість, ефективність та гнучкість для подальшого розвитку вебзастосунку.

2.5 Вибір інструментів для тестування, відлагодження, безпеки, інтеграції та розгортання

У контексті вимог до тестування та відлагодження вебзастосунку, обираючи інструменти, враховувалися основні аспекти безпеки та функціональності.

Першою вимогою було забезпечення безпеки коду. Для цього був обраний Jest – інструмент для модульного тестування JavaScript коду. Використовуючи Jest, можна проводити тестування різних частин вебзастосунку, зокрема, забезпечуючи перевірку безпеки коду та виявлення можливих помилок [4].

Другою важливою вимогою було тестування API. Для цього був вибраний Supertest, що дозволяє перевіряти коректність обробки запитів та відповідей сервера, а також забезпечує можливість тестування без авторизації.

Третьою вимогою стало енд-ту-енд тестування. В цьому випадку вибором став Cypress, що забезпечує ефективне енд-ту-енд тестування вебзастосунку, включаючи безпеку та функціональність, і надає інтерфейс для відлагодження [4].

Visual Studio Code надає вбудовані засоби для відлагодження коду. Зокрема, використовується вбудований дебагер для JavaScript, що дозволяє відстежувати виконання коду та ідентифікувати можливі помилки [21].

Додатково, для аналізу коду та виявлення потенційних проблем безпеки були обрані ESLint та Prettier. Ці інструменти допомагають не лише аналізувати, але і форматувати код, покращуючи загальний рівень безпеки.

У разі відлагодження коду на ранніх етапах розробки можна використовувати вбудовані інструменти розробника браузера (Chrome DevTools), що забезпечує зручний інтерфейс відлагодження та аналіз роботи вебзастосунку. Обрані інструменти спрямовані на ретельне тестування різних аспектів вебзастосунку, враховуючи вимоги до безпеки та функціональності.

Отже, загальні вимоги включають ефективний захист від різних видів атак, таких як SQL-ін’єкції, XSS атаки, CSRF та інші. Конкретні вимоги до безпеки вебзастосунку обумовлені його функціональністю та особливостями.

Враховуючи ці вимоги, вибір інструментів для забезпечення безпеки має на меті забезпечити надійний захист від атак та збереження конфіденційності та цілісності даних. Отже, використання захищеної передачі даних за допомогою протоколу HTTPS та застосування механізмів, таких як унікальні CSRF-токени, є основними елементами.

Зокрема, обрані інструменти включають Jest для модульного тестування JavaScript коду, Supertest для тестування API, Cypress для енд-ту-енд тестування, Helmet.js для управління HTTP-заголовками та забезпечення безпеки, а також вбудовані інструменти розробника браузера для відлагодження коду на ранніх етапах розробки.

Отже, вибір цих інструментів спрямований на забезпечення базового рівня безпеки, включаючи захист від атак та забезпечення конфіденційності передачі даних. При цьому враховуються принципи простоти, використання надійного фреймворка Next.js, правильний вибір інструментів та постійна підтримка чистоти кодової бази, забезпечать тим самим безпеку та надійність програмного забезпечення в довгостроковій перспективі [9].

Під час роботи над проєктом Next.js застосування ефективних практик розробки та впровадження безперервної інтеграції (CI) має важливе значення для безперебійного роботи застосунку.

Безперервна інтеграція (CI) – це практика розробки, яка зосереджена на регулярній інтеграції змін коду від кількох розробників у спільне сховище. Основна мета CI – виявити проблеми інтеграції на ранній стадії, забезпечити

якість коду та швидкий зворотний зв’язок. Проєкти NextJS можуть отримати значну користь від впровадження робочого процесу CI. Основні аспекти CI:

– автоматизоване збирання та тестування;

– перевірка якості коду;

– безперервне розгортання;

– співпраця та зворотній зв’язок;

– контроль версій і розгалуження;

– моніторинг і звітність [29].

Для описаних вимог можна обрати Vercel та Docker, як інструменти для інтеграції та розгортання вебзастосунку на основі Next.js.

Vercel обрано для керованого розгортання вебзастосунків, побудованих з використанням Next.js. Платформа Vercel надає автоматичне конфігурування та високу продуктивність, що відповідає вимогам безпечного, швидкого та гнучкого розгортання. Її особливості включають високу доступність та простоту використання. Використання Vercel гарантує оптимальну конфігурацію для Next.js застосунків та спрощує весь процес розгортання.

Для самостійного розміщення та забезпечення переносимості та консистентності середовища обрано Docker. Використання Docker дозволяє створити контейнер, який включає всі необхідні залежності для правильної роботи Next.js застосунку. Docker забезпечує ізольоване середовище, що гарантує, що всі компоненти застосунку працюють однаково на будь-якій системі, в якій відбувається розгортання [25, 27].

Отже, обрані інструменти забезпечують простоту використання та відповідають вимогам щодо безпеки, швидкості, та адаптивності.

3 ДОСЛІДЖЕННЯ ТА РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКІВ НА ПРИКЛАДІ NEXT.JS

3.1 Переваги та функціонали Next.js у контексті дослідження

Next.js – це фреймворк на основі React для створення Full Stack вебзастосунків [23].

Next.js є одним із найпопулярніших фреймворків для розробки вебзастосунків, і має велику та активну спільноту розробників. Це робить його привабливим об’єктом дослідження, оскільки велика кількість користувачів та спільнота розробників сприяють активному розвитку та підтримці фреймворку. На рисунку 3.1 зображено результат порівняння Full Stack фреймворків за кількістю завантажень за 2023 рік з огляду на це можна сказати, що попит на застосування фреймворка постійно збільшується.

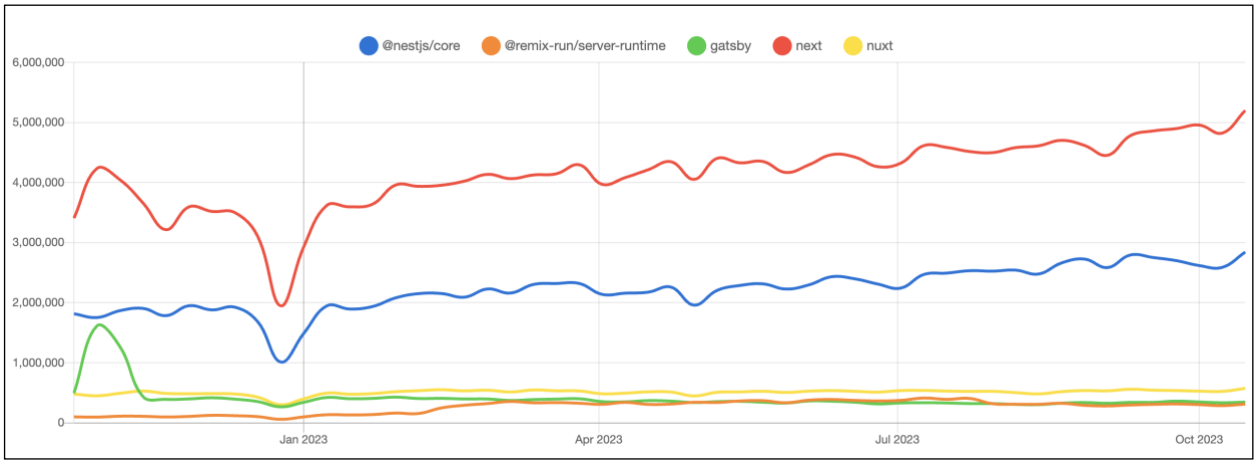

Рисунок 3.1 – Результат порівняння Full Stack фреймворків за кількістю завантажень за 2023 рік [18]

Next.js надає розширену функціональність для розробки вебзастосунків, включаючи Server-Side Rendering (SSR), Static Site Generation (SSG), маршрутизацію, систему компонентів, оптимізацію зображень та інші

можливості. Ця різноманітність функцій дозволяє проводити детальний аналіз різних аспектів розробки.

Це вебфреймворк з відкритим вихідним кодом, що дозволяє докладно досліджувати його роботу та вносити зміни за потребою. Також, Next.js має добре структуровану документацію, яка часто оновлюється, що спрощує роботу з ним та аналіз його можливостей. Документація написана зрозумілою мовою, містить достатньо деталей та прикладів коду.

Велика кількість популярних вебзастосунків та сайтів побудовані на базі Next.js (Netflix Jobs, Notion, Typeform, Nike, Western Union та ін.), що робить його релевантним об’єктом дослідження [17]. Аналізуючи Next.js, можна отримати важливі відкриття щодо найкращих практик розробки.

Next.js відомий своєю продуктивністю та можливістю оптимізації вебзастосунків. Це дозволяє розробникам створювати швидкі та ефективні вебзастосунки, що стає важливою темою для дослідження.

Server-Side Rendering (SSR). Однією з основних функцій Next.js є можливість використовувати Server-Side Rendering. Це означає, що сторінки вебзастосунку генеруються на сервері під час запиту користувача, а не в клієнтському браузері. Це дозволяє створювати динамічний вміст, який може бути індексований пошуковими системами, покращуючи SEO. Для використання SSR в Next.js, ви можете використовувати функції getServerSideProps та getInitialProps, які дозволяють підготувати дані для сторінок на сервері перед їх відправленням клієнту.

Static Site Generation (SSG). Next.js також підтримує Static Site Generation, яке дозволяє попередньо генерувати сторінки в HTML-файли під час збірки застосунку. Це спрощує процес розгортання та дозволяє використовувати кешування та розподілення контенту через CDN. За допомогою функції getStaticProps, надає можливість визначити, які сторінки мають бути згенеровані на стадії збірки.

Маршрутизація та робота з маршрутами. В Next.js існує вбудований маршрутизатор, що спрощує створення маршрутів та навігацію в застосунку.

Він надає можливість визначити динамічні маршрути з параметрами, що дозволяє створювати різні сторінки на основі даних. Це робить роботу з маршрутами в Next.js дуже зручною. Використовуються модулі Link та useRouter для навігації та роботи з маршрутами.

Система компонентів. Next.js сприяє використанню компонентів як основних будівельних блоків для створення інтерфейсу вебзастосунку. Компоненти можуть бути перевикорстані в різних частинах застосунку і організовані в дерево компонентів для структурування сторінок.

Оптимізація зображень. Оптимізація зображень – це важливий аспект веброзробки. Next.js надає вбудовану підтримку для оптимізації зображень, що дозволяє автоматично оптимізувати та змінювати розмір зображення для покращення продуктивності та швидкості завантаження сторінок.

Маніпуляція статичними файлами. В Next.js є можливість робити статичні файли доступними через вбудований сервер. Це дозволяє легко розміщувати файли, такі як стилі, зображення або файли JSON, як статичні ресурси, без необхідності написання додаткового серверного коду.

Інтеграція з сервером та API. Next.js дозволяє легко інтегрувати застосунок з сервером та іншими API. Ви можете використовувати функції getServerSideProps, getStaticProps, getInitialProps та інші для отримання та відправлення даних на сервері та взаємодії з іншими послугами, такими як бази даних або зовнішні API.

Однією з особливостей Next.js є його гнучкість, яка виражається в можливості легко інтегрувати фреймворк з іншими технологіями. Це дозволяє розробникам створювати не лише прості вебсторінки, але і складні, унікальні вебзастосунки, задовольняючи різноманітні потреби та вимоги проєктів.

Next.js надає можливість легко інтегруватися з різними системами керування базами даних, такими як MongoDB, MySQL, PostgreSQL та інші. Інтеграція з базами даних дозволяє забезпечити ефективне зберігання та обробку даних в реальному часі.

Next.js спрощує процес аутентифікації, включаючи реалізацію різних стратегій, таких як OAuth, локальна аутентифікація тощо. Використання бібліотеки next-auth забезпечує легке налаштування аутентифікації через провайдерів, таких як Google чи Facebook. Крім аутентифікації, Next.js дозволяє легко реалізувати систему авторизації, контролюючи доступ користувачів до конкретних ресурсів чи функцій застосунку.

Next.js взаємодіє з іншими фронтенд фреймворками, щоб забезпечити універсальний та гнучкий підхід до розробки. Next.js дозволяє легко інтегрувати функції Прогресивного Вебу (PWA), такі як сервісні робітники (Service Workers) чи можливості офлайн-роботи. Це важливо для забезпечення ефективності та доступності Застосунку в умовах обмеженої мережевої доступності.

Next.js забезпечує розширені можливості інтеграції з інструментами для тестування та забезпечення якості, спрощуючи процес валідації та підтримки високого стандарту коду. Для юніт-тестування та інтеграційних тестів, використовуються популярні бібліотеки, такі як Jest та Testing Library. Інтеграція з інструментами тестування React-компонентів, такими як Enzyme чи Testing Library, дозволяє перевіряти роботу компонентів. За допомогою інструментів для визначення стилевих та синтаксичних помилок, таких як ESLint та Prettier, можна забезпечити чистоту та послідовність коду. Крім того, Next.js дозволяє проводити тести верифікації відповіді сервера, що є важливим для перевірки правильності роботи API та забезпечення надійності бекенду [22].

У таблиці 3.1 порівняно функціональні можливості Next.js з Nuxt та Gatsby. Основна різниця між Next.js, Nuxt і Gatsby полягає в їхніх підходах до розробки вебзастосунків та способах досягнення серверного рендерингу та статичної генерації.

Таблиця 3.1 – Порівняння функціональних можливостей Next.js з іншими популярними фреймворками

Параметр	**Next.js**	**Nuxt**	**Gatsby**
Основна мета	Динамічний вебсайт, статичний вебсайт	Динамічний вебсайт, статичний вебсайт	Статичний вебсайт
Server-Side Rendering (SSR)	Має вбудовану підтримку	Має вбудовану підтримку для Vue.js	Не має вбудованої підтримки
Static Site Generation (SSG)	Має вбудовану підтримку	Має вбудовану підтримку для Vue.js.	Має вбудовану підтримку
Маршрутизація та робота з маршрутами	Має вбудований маршрутизатор і дозволяє створювати динамічні маршрути.	Має вбудований маршрутизатор для Vue.js.	Має вбудований маршрутизатор для SSG
Система компонентів	Підтримує роботу з компонентами, але не надає готових шаблонів компонентів	Має підтримку компонентів для Vue.js.	Використовує React для створення компонентів і має доступ до багатьох бібліотек React компонентів
Оптимізація зображень	Має вбудовану підтримку оптимізації зображень	Має підтримку оптимізації зображень для Vue.js	Має вбудовану підтримку оптимізації зображень для SSG
Маніпуляція статичними файлами	Дозволяє легко розміщувати статичні файли на сервері.	Має підтримку для роботи зі статичними файлами для Vue.js.	Добре підтримує статичну генерацію файлів, включаючи стилі, зображення та інші ресурси.
Інтеграція з сервером та API	Має вбудовану підтримку для роботи з сервером та API	Має вбудовану підтримку для роботи з сервером та API для Vue.js	Має підтримку роботи з сервером та API через плагіни і функції.

Next.js – це фреймворк для React, який надає інструменти для створення універсальних застосунків з можливістю Server-Side Rendering

(SSR) та Static Site Generation (SSG). Next.js використовує React як основу та дозволяє розробникам створювати реактивні та інтерактивні вебзастосунки з вбудованою підтримкою SSR та SSG для поліпшення продуктивності та SEO [16].

Nuxt – це фреймворк для Vue.js, який надає подібні можливості до Next.js. Він також підтримує SSR та SSG, але для Vue.js. Nuxt спеціалізується на зручності розробки для Vue.js і має вбудований маршрутизатор, оптимізацію зображень та інші функції для спрощення процесу розробки. Хоча Nuxt і Next.js є досить схожими, Nuxt потребує більше конфігурації, що не завжди є погано. У файлі конфігурації Nuxt можна визначити шаблони, глобальні плагіни компоненти, маршрути тощо, тоді як з Next.js потрібно робити це за стандартами React [20].

Gatsby – це фреймворк з відкритим кодом, який спеціалізується на Static Site Generation (SSG). Він побудований на основі React, але покладає основний акцент на генерацію статичних сторінок. Gatsby підходить для створення вебсайтів з високою швидкістю завантаження і продуктивністю. Він має велику кількість плагінів для оптимізації зображень, стилів та інших ресурсів [19].

Отже, основна різниця полягає в тому, що Next.js та Nuxt надають інструменти для розробки вебзастосунків з SSR та SSG на основі React та Vue.js відповідно, тоді як Gatsby спеціалізується на SSG на основі React і має більшу акцент на статичні сайти.

3.2 Результат розроблення вебзастосунку

Для реалізації вебзастосунку було використано останню стабільну версію Next.js – 13.5.

Розробка проводилась в інтегрованому середовищі розробки Visual Studio Code версії 1.83.1. Visual Studio Code – це легкий, але потужний

редактор вихідного коду, доступний для Windows, macOS і Linux. Він має з вбудовану підтримкою JavaScript, TypeScript і Node.js та багату екосистему розширень для інших мов програмування (таких як C++, C#, Java, Python, PHP, Go, .NET) [23].

Для розробки логіки застосунку було використано мови програмування JavaScript та TypeScript. JavaScript є загальною мовою програмування для розробки вебзастосунків, і TypeScript надає переваги статичної типізації, яка полегшує розробку та підвищує надійність коду.

Для стилізації компонентів було використано CSS та Tailwind. CSS та Tailwind використовуються для стилізації та оформлення компонентів вебзастосунку. CSS (Cascading Style Sheets) є основною мовою стилізації вебсторінок і дозволяє визначити зовнішній вигляд різних елементів на сторінці, таких як текст, колір, розмір шрифту та розташування.

Tailwind CSS, з іншого боку, є CSS-фреймворком, який надає набір готових до використання класів для швидкого та ефективного стилізації. Замість написання власного CSS-коду, розробники можуть використовувати класи Tailwind для застосування стилів. Це спрощує процес розробки, прискорює час виконання завдань та сприяє однорідності в оформленні вебзастосунку.

Таким чином, CSS та Tailwind виконують функцію визначення зовнішнього вигляду та стилізації елементів вебзастосунку, забезпечуючи користувачам приємний та естетичний інтерфейс.

Для управління залежностями та пакетами було використано пакетний менеджер npm. Цей пакетний менеджер надав зручність у встановленні необхідних пакетів та зберіганні їхніх версій у файлі package.json. npm є потужним інструментом для керування залежностями проєкту та дозволяє ефективно управляти пакетами.

Тестування проєкту проводилось у браузері Google Chrome версії 118.0.5993.117, який працює локально в операційній системі macOS Ventura версії 13.4.1.

Розробка промо-сайту почалася зі створення порожнього Next.js застосунку (лістинг 3.1).

Лістинг 3.1 Команда для створення порожнього Next.js застосунку:

npx create-next-app@latest <promo-site>

У каталозі pages/ доступний файл index.tsx, який представляє головну сторінку вебзастосунку, яку користувачі побачать відвідавши кореневу сторінку промо-сайту. Локально дана сторінка доступна за адресою http://localhost:3000.

Для створення нової сторінки з контактною інформацією знадобилось додати новий файл contacts.tsx в каталог pages/ (рис. 3.2).

```
11  // /pages/contacts
12
13  export default function Contacts() {
14    return (
15      <Layout>
16        <Container>
17          <h1>Контакти</h1>
18        </Container>
19      </Layout>
20    )
21  }
```

Рисунок 3.2 – Демонстрація сторінки контактів

На рисунку 3.3 продемонстровано використання Server-Side Rendering (SSR) у вебзастосунку. Функція getServerSideProps використовується для виконання запиту до JSONPlaceholder API та отримання даних на сервері при кожному запиті. Отримані дані передаються на сторінку Users через props. Цей підхід особливо корисний для сторінок, де дані можуть змінюватися

часто або вимагають динамічних запитів до серверу при кожному оновленні сторінки.

Використання SSR гарантує актуальні дані для користувача під час завантаження сторінки. Це дозволяє підтримувати сторінку у поточному стані та надавати користувачам оновлену інформацію при кожному відвідуванні сторінки. Такий підхід дозволяє створювати динамічні сторінки з оновленою інформацією, зберігаючи при цьому продуктивність та швидкість завантаження.

```
export default function Users({ users }: Props) {
  return (
    <Layout>
      <Container>
        <div>
          {users.map(item => {
            return <UserCard key={item.id} user={item} />})
          }
        </div>
      </Container>
    </Layout>
  )
}

export async function getServerSideProps() {
    const users = await fetch('https://jsonplaceholder.typicode.com/users')
      .then(response => response.json())

    return { props: { users } }
  }
```

Рисунок 3.3 – Застосування Server-Side Rendering

Застосування Static Site Generation(SSG) у Next.js дозволяє генерувати статичні сторінки під час збірки проєкту, що покращує продуктивність та час завантаження сторінок. На рисунку 3.4 наведено приклад створення статичної сторінки за допомогою SSG у Next.js. SSG особливо корисний для сторінок, які не змінюються часто, оскільки він дозволяє кешувати статичний контент і покращити продуктивність вебзастосунку.

Для оптимізації швидкості завантаження сторінок можна використовувати вбудовані інструменти Next.js для оптимізації зображень та їхньої попередньої обробки. На рисунку 3.5 створено компонент UserCard, в якому знаходиться оптимізоване за допомогою компонента Image зображення. Нижче в коді було додано звичайний JSX-елемент img для порівняння продуктивності обох компонентів.

```
export default function Contacts({ data }: Props) {
  return (
    <Layout>
      <Container>
        <div>{data}</div>
      </Container>
    </Layout>
  )
}

export async function getStaticProps() {
  const data = await fetch('https://jsonplaceholder.typicode.com/posts/1')
  .then(response => response.json())
  return {
    props: {
      data,
    },
  };
}
```

Рисунок 3.4 – Застосування Static Site Generation

Розробники можуть легко використовувати компонент Image без необхідності ручного оброблення та розміщення різних версій зображень.

Для демонстрації можливостей роутингу в Next.js в структуру проєкту було додано 2 файли index.tsx та users.tsx у папку pages (рис. 3.6). Кожен файл, розташований у папці pages, автоматично стає доступним як маршрут. Наприклад, файл pages/users.tsx доступний за адресою /users.

Для створення посилань між сторінками використовується компонент Link з пакета next/link (рис. 3.6). Він дозволяє створювати навігаційні посилання, які передбачають передзавантаження сторінки на клієнтському боці для забезпечення швидкості навігації.

```tsx
import User from "../interfaces/users"
import Image from "next/image";  15.8k (gzipped: 5.8k)

type Props = {
  user: User
}

const UserCard = ({ user }: Props) => {
  return (
    <div className="flex items-center mb-8">
      <Image
        src='/assets/blog/dynamic-routing/cover.jpg'
        width={500}
        height={300}
        alt="Cover picture 1"
      />
      <img
        src='/assets/blog/dynamic-routing/cover.jpg'
        width={500}
        height={300}
        className="ml-8"
        alt="Cover picture 2"/>
        <div>{user.name}</div>
    </div>
  )
}

export default UserCard
```

Рисунок 3.5 – Застосування next/image

```tsx
import Link from "next/link";  31.5k (gzipped: 9.2k)

const HomePage = () => {
  return (
    <div>
      <h1>Home Page</h1>
      <Link href="/users">Go to Users Page</Link>
    </div>
  );
};

export default HomePage;
```

```
node_modules
pages
  posts
  _app.tsx
  _document.tsx
  contacts.tsx U
  index.tsx M
  users.tsx U
public
styles
.gitignore
```

Рисунок 3.6 – Застосування маршрутизації

Вбудована система маршрутизації спрощує створення посилань та навігацію між сторінками. Розробники можуть легко керувати маршрутами та параметрами маршруту. Маршрутизація в Next.js допомагає створювати дружні для SEO URL-адреси та забезпечує швидку навігацію між сторінками без повторного перезавантаження сторінки.

Next.js також може обробляти динамічний контент на основі динамічного URL.

Для реалізації цієї можливості створено директорію з назвою posts (рис. 3.7). У папці posts додано файл [id].tsx, де id – це динамічний параметр у маршруті.

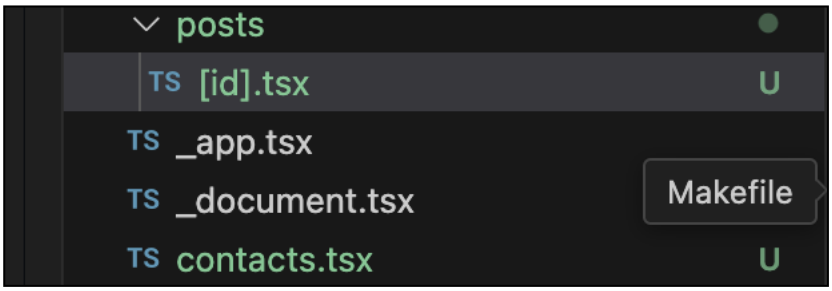

Рисунок 3.7 – Створення файлів для динамічного роутингу

У файлі [id].tsx визначено компонент, який буде відображати інформацію публікації (рис. 3.8). Компонент Post, який приймає дані про публікацію (наприклад, заголовок і текст) як властивість postInfo і відображає їх на сторінці. Цей компонент використовується для відображення окремої публікації на сторінці.

Для отримання даних про публікацію, використано getServerSideProps, який отримує context із параметрами маршруту, в даному випадку, id. Метод fetch використано для того, щоб виконати запит на сервері за інформацією про публікацію і передати ці дані через props для використання в компоненті Post.

```
export default function Post({ postInfo }: Props) {
  return (
    <Layout>
      <Container>
        <div>
          <h1>{postInfo.title}</h1>
          <p>{postInfo.body}</p>
        </div>
      </Container>
    </Layout>
  )
}

export async function getServerSideProps(context) {
  const { id } = context.params;

  const postInfo = await fetch(`https://jsonplaceholder.typicode.com/posts/${id}`)
    .then(response => response.json())

  return { props: { postInfo } }
}
```

Рисунок 3.8 – Приклад сторінки з динамічним маршрутом

3.3 Особливості роботи з функціоналом Next.js у вебзастосунку

Server-Side Rendering (SSR) виявився корисним для вебзастосунку. Однією з головних переваг було поліпшення швидкості завантаження сторінок. SSR надає можливість генерувати сторінки на сервері і відправляти готовий HTML користувачам, замість очікування, поки весь JavaScript завантажиться на клієнтському боці. Це призвело до миттєвого відображення вмісту для користувачів, що поліпшило загальне враження. Це особливо актуально для споживачів з обмеженими ресурсами, так як такі застосунки не вимагають багато робочої потужності на клієнтському боці.

Крім того, SSR дозволив оптимізувати сторінки для пошукових систем (SEO). Це дало змогу створювати статичні версії сторінок для пошукових роботів. Пошукові системи, такі як Google, краще індексують сторінки зі змістом, який генерується на стороні сервера.

На рисунку 3.9 та 3.10 зображено різницю між сторінками, які генеруються за допомогою Server-Side Rendering (SSR) і тих, які використовують клієнтський запит. Ця різниця демонструє, як SSR забезпечує наявність HTML-вмісту на сторінці до більшої SEO-ефективності.

На рисунку 3.9 зображено, що весь HTML-код сторінки міститься у відповіді від сервера. Це включає всі дані та вміст сторінки. Пошукові системи можуть легко індексувати цей контент.

На рисунку 3.10 видно, що сторінка спочатку має мінімальний HTML-код, який потім запитує дані з сервера, використовуючи клієнтський запит (fetch на клієнтському боці). Це означає, що вебсторінка спочатку завантажується з обмеженим вмістом, і весь контент додається динамічно після завантаження сторінки.

<!DOCTYPE html><html lang="en"><head><style data-next-hide-fouc="true">body{display:none}</style><noscript data-next-hide-fouc="true"><style>body{display:block}</style></noscript><meta charSet="utf-8"/><meta name="viewport" content="width=device-width"/><link rel="apple-touch-icon" sizes="180x180" href="/favicon/apple-touch-icon.png"/><link rel="icon" type="image/png" sizes="32x32" href="/favicon/favicon-32x32.png"/><link rel="icon" type="image/png" sizes="16x16" href="/favicon/favicon-16x16.png"/><link rel="manifest" href="/favicon/site.webmanifest"/><link rel="mask-icon" href="/favicon/safari-pinned-tab.svg" color="#000000"/><link rel="shortcut icon" href="/favicon/favicon.ico"/><meta name="msapplication-TileColor" content="#000000"/><meta name="msapplication-config" content="/favicon/browserconfig.xml"/><meta name="theme-color" content="#000"/><link rel="alternate" type="application/rss+xml" href="/feed.xml"/><meta name="description" content="A statically generated blog example using Next.js and Markdown."/><meta property="og:image" content="https://og-image.vercel.app/Next.js%20Blog%20Starter%20Example.png?theme=light&md=1&fontSize=100px&images=https%3A%2F%2Fassets.vercel.com%2Fimage%2Fupload%2Ffront%2Fassets%2Fdesign%2Fnextjs-black-logo.svg"/><meta name="next-head-count" content="14"/><noscript data-n-css=""></noscript>
<script defer="" nomodule="" src="/_next/static/chunks/polyfills.js?ts=1698942486387"></script><script src="/_next/static/chunks/webpack.js?ts=1698942486387" defer=""></script><script src="/_next/static/chunks/main.js?ts=1698942486387" defer=""></script><script src="/_next/static/chunks/pages/_app.js?ts=1698942486387" defer=""></script><script src="/_next/static/chunks/pages/users.js?ts=1698942486387" defer=""></script><script src="/_next/static/development/_buildManifest.js?ts=1698942486387" defer=""></script><script src="/_next/static/development/_ssgManifest.js?ts=1698942486387" defer=""></script><noscript id="__next_css__DO_NOT_USE__"></noscript></head><body> <main> Leanne Graham<!-- --> <!-- -->Sincere@april.biz<!-- --> <!-- --> <!-- --> <!-- -->1<!-- --> <!-- --> <!-- -->Bret<!-- --> <!-- --> <!-- -->92998-3874<!-- --> Ervin Howell<!-- --> <!-- --> <!-- -->Shanna@melissa.tv<!-- --> <!-- --> <!-- --> <!-- -->2<!-- --> <!-- -->Antonette<!-- --> <!-- --> <!-- -->90566-7771<!-- --> Clementine Bauch<!-- --> <!-- --> <!-- -->Nathan@yesenia.net<!-- --> <!-- --> <!-- --> <!-- -->3<!-- --> <!-- --> <!-- -->Samantha<!-- --> <!-- --> <!-- -->59590-4157<!-- --> Patricia Lebsack<!-- --> <!-- --> <!-- -->Julianne.OConner@kory.org<!-- --> <!-- --> <!-- --> <!-- -->4<!-- --> <!-- --> <!-- -->Karianne<!-- --> <!-- --> <!-- -->53919-4257<!-- --> Chelsey Dietrich<!-- --> <!-- --> <!-- -->Lucio_Hettinger@annie.ca<!-- --> <!-- --> <!-- --> <!-- -->5<!-- --> <!-- --> <!-- -->Kamren<!-- --> <!-- --> <!-- -->33263<!-- --> Mrs. Dennis Schulist<!-- --> <!-- --> <!-- -->Karley_Dach@jasper.info<!-- --> <!-- --> <!-- --> <!-- -->6<!-- --> <!-- --> <!-- -->Leopoldo_Corkery<!-- --> <!-- --> <!-- -->23505-1337<!-- --> Kurtis Weissnat<!-- --> <!-- --> <!-- -->Telly.Hoeger@billy.biz<!-- --> <!-- --> <!-- --> <!-- -->7<!-- --> <!-- --> <!-- -->Elwyn.Skiles<!-- --> <!-- --> <!-- -->58804-1099<!-- --> Nicholas Runolfsdottir V<!-- --> <!-- --> <!-- -->Sherwood@rosamond.me<!-- --> <!-- --> <!-- --> <!-- -->8<!-- --> <!-- --> <!-- -->Maxime_Nienow<!-- --> <!-- --> <!-- -->45169<!-- --> Glenna Reichert<!-- --> <!-- --> <!-- -->Chaim_McDermott@dana.io<!-- --> <!-- --> <!-- --> <!-- -->9<!-- --> <!-- --> <!-- -->Delphine<!-- --> <!-- --> <!-- -->76495-3109<!-- --> Clementina DuBuque<!-- --> <!-- --> <!-- -->Rey.Padberg@karina.biz<!-- --> <!-- --> <!-- --> <!-- -->10<!-- --> <!-- --> <!-- -->Moriah.Stanton<!-- --> <!-- --> <!-- --> <!-- -->31428-2261<!-- --> </main> <footer class="bg-neutral-50 border-t border-neutral-200"> <h3 class="text-4xl lg:text-[2.5rem] font-bold tracking-tighter leading-tight text-center lg:text-left mb-10 lg:mb-0 lg:pr-4 lg:w-1/2">Statically Generated with Next.js.</h3> Read DocumentationView on GitHub </footer> <script src="/_next/static/chunks/react-refresh.js?ts=1698942486387"></script><script id="__NEXT_DATA__" type="application/json">{"props":{"pageProps":{"users":[{"id":1,"name":"Leanne

Рисунок 3.9 – HTML-код сторінки з Next.js

Рисунок 3.10 – HTML-код сторінки без Next.js

Різниця між цими двома підходами полягає в тому, що сторінка, яка використовує SSR, має весь необхідний вміст у відповіді сервера, що поліпшує SEO та загальний вміст сторінки в очах пошукових систем.

Переміщення частини коду на сервер може забезпечити більшу безпеку для вебзастосунку. Такі дії, як керування файлами cookie, виклик приватних API і перевірка даних, відбуваються на сервері, тому особисті дані клієнта ніколи не будуть відкритими.

Static Site Generation був корисним для створення статичних сторінок, які не залежать від користувача і мають стабільний вміст. Наприклад, домашня сторінка або сторінка «Contacts» майже не змінюються, тому є можливість попередньо згенерувати їх і зберігати як статичні файли. Це дозволить ефективно кешувати ці сторінки та миттєво їх віддавати користувачам, навіть при великому навантаженні.

Система маршрутизації в Next.js дала можливість керувати навігацією в застосунку та створювати URL-адреси. Це надає змогу визначити маршрути та параметри, що зробило роботу з різними сторінками дуже простою.

Користувач може перейти за URL /posts/123, де 123 – ідентифікатор публікації, і система маршрутизації автоматично витягне цей параметр, що дозволяє завантажувати відповідну інформацію.

Однією з головних переваг Next.js є вбудована підтримка оптимізації зображень. У контексті тестового вебзастосунку цю функціональність

використано для покращення продуктивності та швидкості завантаження сторінок.

Next.js надає можливість автоматично оптимізувати зображення під час збирання проєкту. Наприклад, зображення для продуктів завантажуються великого розміру, але завдяки оптимізації Next.js, ці зображення автоматично обрізаються та стискаються до оптимального розміру, що дозволяє зекономити пропускну здатність та покращити швидкість завантаження сторінок.

На рисунку 3.11 наведено результат використання Image компонента в браузері. Після завантаження сторінки зображення має розмір 5,2 kB, тоді як стандартний JSX-елемент img має розмір 118kB (рис. 3.12). Можна зробити висновок, що розмір зображення, в даному випадку, зменшився у більш ніж 20 разів, при цьому на якість зображення це ніяк не вплинуло (рис. 3.13).
Оптимізація зображень у Next.js дозволяє зменшити обсяг даних, які користувач повинен був завантажити, що призводить до покращення швидкості завантаження сторінок та зменшення навантаження на сервер.
Компонент Image абстрагує складність оптимізації зображень і полегшує роботу з ними. Це означає, що розробник може зосередитися на основних видах діяльності, таких як усунення помилок і розробка функціональних можливостей застосунку. Компонент Image також забезпечив оптимальну якість зображень і продуктивність вебзастосунку, що є важливим для зручності користувачів та зниження витрат на хостинг.

Рисунок 3.11 – Image компонент в браузері

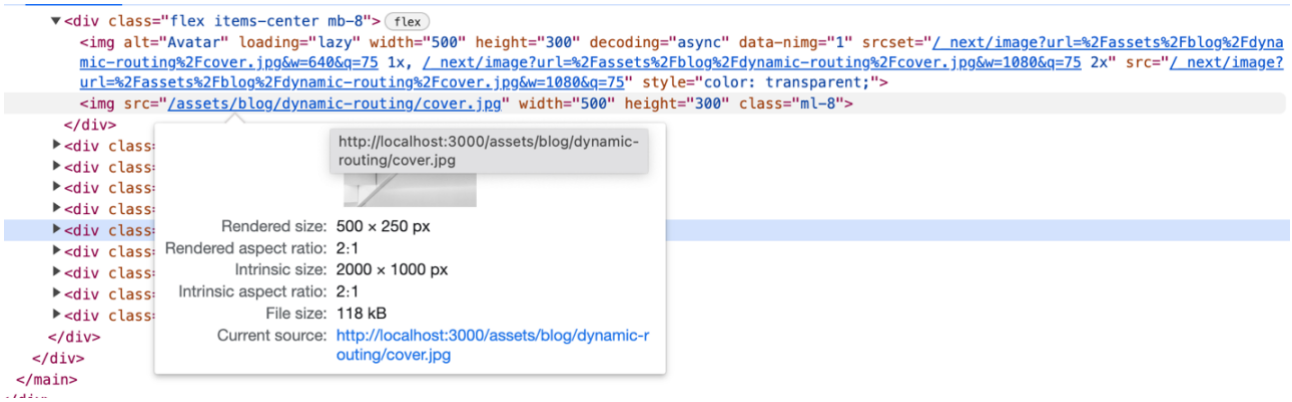

Рисунок 3.12 – JSX-елемент img в браузері

Рисунок 3.13 – Демонстрація якості зображень

ВИСНОВКИ

У рамках кваліфікаційної роботи проведено аналіз функціональних можливостей сучасних програмних засобів для розроблення вебзастосунків на прикладі Next.js. Аналіз функціоналів Next.js у складі технології визначив її потужності для розробки вебзастосунків. Створення простого застосунку дало змогу не лише практично вивчити фреймворк, але й перевірити його ефективність в реальному проєкті.

Інтеграція з вбудованими інструментами для аналізу зображень в Next.js дозволяє автоматично зменшувати їх розмір без втрати якості, що забезпечує ефективне використання ресурсів. Next.js надає простий та зручний спосіб робити маршрутизацію в застосунку. Динамічні роути дають можливість ефективно працювати з параметрами URL та створювати динамічний контент. Генерація статичних сайтів (SSG) і рендеринг на стороні сервера (SSR) гарантує, що пошукові системи можуть ефективно сканувати та індексувати сторінки.

У результаті виконання поставлених завдань отримані дані та висновки стануть основою для подальших кроків у розробці та вдосконаленні вебзастосунків з використанням даного фреймворку.

Дослідження може бути розширене для охоплення більшої кількості програмних засобів. Нові програмні засоби, які виникають на ринку, можуть бути включені до аналізу для оцінки їх функціональності.

Аналіз Next.js може бути продовжено шляхом додавання нових функцій у вебзастосунок, розгляд можливостей для інтеграції інших технологій чи фреймворків у комбінації з Next.js для поліпшення функціональності.

Результати дослідження апробовано у вигляді тез доповідей під час Міжнародної науково-практичної конференції «New ways of creating scientific ideas for implementation» [15].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Michael Scott (2015). Programming Language Pragmatics.
2. Cay S. Horstmann (2020). Modern JavaScript for the Impatient.
3. Jon Duckett (2011). HTML and CSS: Design and Build Websites.
4. Bruce Johnson (2019). Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers.
5. Scott Chacon і Ben Straub (2014). Pro Git.
6. Rick Silva (2021). MySQL Crash Course. A Hands-on Introduction to Database Development.
7. Arnon Axelrod (2018). Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects.
8. Kent Beck (2022). TDD: Test-Driven Development by Example. Third Edition.
9. Heather Adkins, Betsy Beyer, Paul Blankinship, Piotr Lewandowski, Ana Oprea, Adam Stubblefield (2020). Building Secure and Reliable Systems.
10. Michael Hartl (2022). Modular Web Development: Building Scalable and Reusable Components.
11. Jon Duckett (2022). Mobile Web Development: Building Mobile Websites and Apps with HTML, CSS, JavaScript, and PHP.
12. David M. Hersleb (2022). A Comparative Analysis of Web Development Frameworks.
13. David M. Hersleb (2022). A Survey of Web Development Frameworks.
14. Michael Hartl (2022). Building Modular Web Applications.
15. Пікуль І.С. (2023). Аналіз сучасних архітектурних рішень для створення вебзастосунків.
16. Next.js official documentation. URL: https://nextjs.org/docs (дата звернення 27.10.2023).

17. Showcase Next.js by Vercel. URL: https://nextjs.org/showcase(дата звернення 31.10.2023).

18. NPM trends. URL: https://npmtrends.com/@nestjs/core-vs-@remix-run/server-runtime-vs-gatsby-vs-next-vs-nuxt (дата звернення 27.10.2023).

19. Gatsby official documentation. URL: https://www.gatsbyjs.com/docs/(дата звернення 30.10.2023).

20. Nuxt official documentation. URL: https://nuxt.com/docs (дата звернення 30.10.2023).

21. Visual Studio Code official documentation. URL: https://code.visualstudio.com/docs (дата звернення 30.10.2023).

22. Flavio Copes (2019). Next.js Handbook.

23. Grace Huang (2023). DYNAMIC TRIO: Building Web Applications with React, Next.js & Tailwind.

24. Gene Kim, Jez Humble, Patrick Debois, та John Willis (2021). The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations.

25. Robert C. Martin Series (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design.

26. Mastering PostgreSQL 9.6: A Comprehensive Guide for PostgreSQL 9.6 Developers and Administrators (2017). Hans-Jürgen Schönig.

27. Ben Nadel (2019). Multi-Page Applications: A Complete Guide to Building and Deploying Modern Web Applications.

28. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom (2002). Database Systems: The Complete Book.

29. Development and Continuous Integration on a NextJS Project. URL: https://medium.com/@dickyarianugraha/development-and-continuous-integration-on-a-nextjs-project-5dc575c548ee (дата звернення 06.09.2023).

30. Гороховатський, В.О., Путятін, Е.П. (2008). Структурне розпізнавання зображень на основі моделей голосування ознак характерних точок. Реєстрація, зберігання і обробка даних, Т.10. №4, С. 75–85.

31. Гороховатский, В.А., Путятин, Е.П., Столяров В.С. (2017). Дослідження результативності структурних методів класифікації зображень, Радіоелектроніка, інформатика, управління, №3 (42), С. 78–85.

32. Гороховатський В.О., Пупченко Д.В., Солодченко К.Г. (2018) Аналіз властивостей, характеристик та результатів застосування новітніх детекторів для визначення особливих точок зображення. Системи управління, навігації та зв'язку. С. 93–98.

33. Gorokhovatskyi V., Gadetska S., Ponomarenko R. (2020) Recognition of Visual Objects Based on Statistical Distributions for Blocks of Structural Description of Image. Proc. of the XV Int. Scientific Conference "Intellectual Systems of Decision Making and Problems of Computational Intelligence" (ISDMCI'2019), Ukraine, May 21–25, 2019, pp. 501-512.

34. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing emotions using convolutional neural networks, International Journal of Academic Information Systems Research, 7(7), 25–36.

35. Гороховатський В.О. (2014) Структурний аналіз та інтелектуальне оброблення даних в комп'ютерному зорі: моногр., СМИТ, 316с.

36. Гороховатський, В. О., Передрій, О. О., Творошенко, І. С., & Марков, Т. Є. (2023). Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень.

37. Gadetska S., Gorokhovatskyi V., Stiahlyk N., Vlasenko N. (2022) Aggregate Parametric Representation of Image Structural Description in Statistical Classification Methods. In CEUR Workshop Proceedings: Computer Modeling and Intelligent Systems (CMIS-2022), 3137, pp. 68-77.

38. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., Vlasenko, N. (2023) Explanation of CNN Image Classifiers with Hiding Parts. In: J. Benois-Pineau, R. Bourqui, D. Petkovic, G. Quenot (eds), Explainable Deep Learning Artificial Intelligence, pp. 125-146, Academic Press, 346 p.

39. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description. Advances in Electrical and Electronic Engineering, 21(1), 19-27.

40. Gorokhovatskyi, V., Vlasenko, N. (2021). Редукція опису зображення у складі множини дескрипторів на основі метричного критерію інформативності. Advanced Information Systems, 5(4), pp. 10-16.

41. Гороховатський В., Творошенко І., Сидоренко Д. (2021) Класифікація зображень із використанням кластерного подання, Міжн. наук. симпозіум Інтелектуальні рішення-С. Обчислювальний інтелект. Теорія прийняття рішень: праці міжн. наук. симп. (Вересень 29, 2021). Київ – Ужгород, 44-45.

42. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium, Sept. 28, 2023, Kyiv-Uzhorod, Ukraine, pp. 25-27.

43. Tvoroshenko, I. S., & Tabashnyk, V. A. (2018). Development of a spatial model of geoinformation support for people with disabilities in wheelchairs in Kharkiv. Collection of scientific works of KhNUPS, 1(55), 122-128.

44. Pomazan V., Tvoroshenko I., Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, International Journal of Academic Engineering Research, 7(9), pp. 64-72.

45. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, International Journal of Academic and Applied Research, 7(9), pp. 57-70.

46. Gorokhovatsky, V.O. and Gadetska, S.V. (2019) Determination of Relevance of Visual Object Images by Application of Statistical Analysis of

Regarding Fragment Representation of their Descriptions, Telecommunications and Radio Engineering, 78 (3), pp. 211–220.

47. M. A. Ahmad, V. Gorokhovatskyi, I. Tvoroshenko, N. Vlasenko, S. K. Mustafa (2021) The Research of Image Classification Methods Based on the Introducing Cluster Representation Parameters for the Structural Description, Intern. Journal of Engineering Trends and Technology, 69(10), pp. 186-192.

48. Gorokhovatskyi V.A. (2018) Image Classification Methods in the Space of Descriptions in the Form of a Set of the Key Point Descriptors. Telecommunications and Radio Engineering, 77 (9), pp. 787-797.

49. Gadetska, S.V., Gorokhovatskyi, V. O., Stiahlyk, N. I., Vlasenko, N.V. Statistical data analysis tools in image classification methods based on the description as a set of binary descriptors of key points. Radio Electronics, Computer Science, Control, 2021, №4, pp. 58-68.т.