Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: **«Дослідження та розробка методології фреймворку NextJS»**

Виконала студентка 4-року навчання
спеціальності 122 «Комп’ютерні науки»
Гвоздак Вікторія Миколаївна
Керівник: Борозенний С. О.
кандидат _____
Рецензент _____
(*прізвище та ініціали*)
Кваліфікаційна робота захищена
з оцінкою _____
Секретар ЕК _____
«___» _____ 2024 р.

Київ – 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ
Завідуючий кафедри мультимедійних систем,
доцент, кандидат наук
Жежерун О. П.
_____
(*підпис*)
«___» _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студентки 4 курсу факультету інформатики

Гвоздак Вікторії Миколаївни

Тема: Дослідження та розробка методології фреймворку NextJS

Зміст кваліфікаційної роботи:

- Вступ
- Розділ 1. Теоретичний огляд
- Розділ 2. Аналіз існуючих рішень
- Розділ 3. Дослідження підходів розробки на NextJS
- Розділ 4. Проектування власної методології
- Розділ 5. Аналіз результатів
- Висновки

- Список використаних джерел
- Додатки

Дата видачі «____» _____ 2024 р.

Керівник _____ (*підпис*)

Завдання отримала _____ (*підпис*)

Календарний план виконання кваліфікаційної роботи

Тема: Дослідження та розробка методології фреймворку NextJS

План виконання роботи:

№ з/п	ЕТАПИ РОБОТИ	ТЕРМІН ВИКОНАННЯ
1.	Вибір теми кваліфікаційної роботи	жовтень 2023
2.	Складання графіка роботи над темою	листопад 2023
3.	Збирання матеріалу. Опрацювання літератури.	листопад – грудень 2023
4.	Вибір програмного забезпечення	січень 2024
5.	Теоретичний огляд засобів розробки	лютий-березень 2024
6.	Написання практичної частини	квітень 2024
7.	Надання кваліфікаційної роботи на перевірку навчальному керівнику	травень 2024
8.	Остаточне оформлення текстової частини і презентації	травень 2024
9.	Захист кваліфікаційної роботи	30 травня 2024

Студентка Гвоздак Вікторія Миколаївна

Керівник Борозенний С. О.

“___” _____

ЗМІСТ

АНОТАЦІЯ

Ця кваліфікаційна робота зосереджується на дослідженні та розробці методології використання фреймворку NextJS, який є передовим інструментом для створення високоефективних веб-додатків. В роботі проведено глибокий аналіз архітектури та основних функціональних можливостей NextJS, включаючи серверний рендеринг (SSR), статичну генерацію сторінок (SSG) та оптимізацію веб-ресурсів.

Основна увага приділяється розробці методології, яка дозволяє ефективно інтегрувати NextJS у процеси веб-розробки, щоб максимально використати його потенціал для підвищення продуктивності, швидкості завантаження та оптимізації пошукової системи. Аналізуються різні сценарії використання NextJS, включаючи створення великих масштабованих додатків і персоналізованих користувацьких інтерфейсів.

Ця робота забезпечує цінний ресурс для розробників, які бажають глибше зрозуміти та ефективніше використовувати можливості NextJS в своїх проєктах. Завдяки теоретичним дослідженням та практичним рекомендаціям, робота сприяє підвищенню якості та ефективності веб-розробки.

ВСТУП

У сучасному світі веб-розробки швидкість, ефективність та надійність стають домінуючими вимогами до проєктів. Через це вибір найбільш підходящих і ефективних інструментів серед величезного різноманіття технологій для розробки веб-застосувань постає надзвичайно гостро. У цьому контексті, NextJS, як один із провідних фреймворків для розробки на React, займає особливе місце завдяки своїм можливостям.

Вибір теми "Дослідження та розробка методології фреймворку NextJS" обґрунтовується стабільною актуальністю веб-розробки і зростаючим попитом на високопродуктивні веб-застосування, створення яких займає мінімум часу і зусиль.

Фреймворк NextJS, побудований на основі популярної JavaScript бібліотеки – React, швидко набуває популярності серед розробників завдяки своїм передовим можливостям у веб-розробці, зокрема забезпечення оптимальної продуктивності додатків, SEO-оптимізацію та швидкість розробки. Однак для того, щоб уповні використовувати весь його потенціал, потрібно детально розібратися в функціональності та принципах роботи технології, оскільки вони вагомо відрізняються від класичних інструментів frontend розробки, таких як React, Angular або Vue.

Зокрема остання версія NextJS 14, яка вийшла зовсім нещодавно, надає можливість frontend частині отримувати інформацію напряму із бази даних, що, при правильній побудові додатку, зменшує необхідність використання додаткових backend фреймворків та бібліотек у повноцінних проектах і збільшує швидкість роботи додатків відповідно. [1]

Тож об’єктом цієї кваліфікаційної роботи є full-stack фреймворк NextJS, а предметом дослідження є методологія його застосування у розробці сучасних веб-

сайтів. Це дослідження акцентується на аналізі архітектурних підходів, інтеграції з іншими технологіями та оптимізації процесу розробки реальних застосунків із врахуванням можливого масштабування.

Метою дослідження є вивчення та удосконалення методології застосування фреймворку NextJS для підвищення ефективності розробки веб-проектів. Конкретні завдання дослідження включають: аналіз сучасних практик застосування NextJS, розробку рекомендацій щодо оптимізації процесів розробки, та проведення експерименту з реалізації тестового проекту з використанням додаткових бібліотек та особливостей вивченого фреймворку.

Огляд літератури показує, що більшість джерел висвітлює великий потенціал NextJS у контексті інтеграції з різноманітними API та удосконалення швидкості роботи веб-сайтів. Також вони вказують на значні переваги в SEO-оптимізації, що критично важливо для комерційних веб-проектів.

Розроблена методологія дозволить розробникам максимально швидко та ефективно використовувати можливості фреймворку для створення високоякісних веб-додатків. Вона створена з урахуванням аналізу основних принципів роботи фреймворку, його переваг та недоліків, а також вивчення найкращих практик використання як у серверній частині, так і клієнтській.

Це дослідження сприятиме глибшому розумінню специфіки використання NextJS і внесе вагомий вклад у сферу методологій веб-розробки, що в свою чергу підвищить якість та ефективність веб-проектів написаних із фреймворком NextJS у проєктах різної складності.

РОЗДІЛ 1. ТЕОРЕТИЧНИЙ ОГЛЯД

1.1. Огляд сучасних веб-фреймворків.

Веб-фреймворк - це набір інструментів, бібліотек і правил, які спрощують розробку веб-додатків, забезпечуючи шаблони, структури та готові компоненти для реалізації різних функцій. Для кращого розуміння варто детально розглянути особливості найпопулярніших зараз frontend-фреймворків.

Ці фреймворки представляють собою різні підходи до веб-розробки, кожен з яких має свої переваги та недоліки. Дослідження та порівняння різних напрямків може бути вельми корисним для розробників, щоб обирати найбільш підходящі для їхніх потреб інструменти.

1.1.1. React. React.js, розроблений компанією Facebook (тепер Meta), є одним із найпопулярніших frontend-фреймворків в індустрії веб-розробки. Його популярність пояснюється декількома факторами, які роблять його привабливим вибором для розробників різних рівнів кваліфікації.

Однією з головних переваг бібліотеки React є широка підтримка та активна спільнота розробників. Завдяки великій кількості досвідчених спеціалістів, які використовують React, компаніям легше знаходити кваліфікованих працівників, а командам – ефективно налагоджувати співпрацю. Ця спільнота постійно надає нові можливості для навчання та обміну знаннями.

Також React має широкий вибір навчальних ресурсів, включаючи офіційну документацію та багато посібників для різних рівнів навчання. Це допомагає розробникам оволодіти фреймворком без великих зусиль та застосовувати найкращі практики у своїх проектах.

Компонентна архітектура React сприяє модульності та легкості управління кодом. Це дозволяє створювати компоненти, які можливо використовувати повторно, і забезпечує покращення продуктивності та стабільності додатків.

Окрім цього, React є гнучким та сумісним з різними бібліотеками та інструментами, що дає можливість використовувати його для різноманітних видів проектів: від невеликих сайтів до складних корпоративних додатків. [2]

1.1.2. Angular. Angular, який створений компанією Google, є одним із провідних учасників на ринку frontend-фреймворків. Його велика спільнота забезпечує міцну основу для підтримки, співпраці та обміну досвідом найкращих підходів.

Останнім часом Angular оновив всі свої офіційні документації та навчальні матеріали, і його популярність знову зросла. Для Angular доступно безліч високоякісних уроків, онлайн-курсів та змісту, створеного спільнотою, які призначені для різних рівнів експертності.

Angular є MVC-фреймворком, який пропонує структурований підхід до розробки додатків. У нього є багато вбудованих функцій, таких як впровадження залежностей, двостороннє зв'язування даних та маршрутизація.

Для новачків може бути складним почати вивчення Angular, особливо через його широкий функціонал та необхідність розуміння таких понять, як TypeScript та RxJS. Однак, якщо розробник уже знає TypeScript та зустрічався із об'єктно-орієнтованим програмуванням, Angular забезпечить зручне середовище.

Angular добре поєднується з різними технологіями на сервері та може використовуватися з різними фреймворками на backend частині застосунків. Однак, інтеграція певних сторонніх бібліотек може вимагати більше зусиль, порівняно з більш гнучкими фреймворками, такими як React. [2]

1.1.3. Vue. Vue.js теж має своє місце у створенні веб-сайтів через його простоту використання та відмінну сумісність з іншими інструментами. Багато людей по всьому світу використовують його, а спільнота продовжує зростати.

Багато людей вносять свій вклад у поліпшення Vue.js, що є однією з причин його популярності. Одним з найкращих аспектів Vue.js є його зрозумілі інструкції та керівництва, ідеальні для початківців.

Vue.js має багато додаткових інструментів та бібліотек, які роблять його ще кращим для різних видів веб-проектів. Він дуже гнучкий – ви можете почати використовувати його невеликою частиною і додавати більше функцій по мірі необхідності. У Vue.js є власні інструменти для управління маршрутами веб-сторінок та відстеження даних.

Цей фреймворк розроблений так, щоб код був легким для читання та організації, що значно спрощує розробку масштабних проектів. Спосіб, яким Vue.js дозволяє писати додаток у одному файлі, робить все організованим.

Vue.js також цікавий тим, що його можна поєднувати з іншими інструментами для кодування або впроваджувати його в уже існуючі проекти. Це зручно, якщо потрібно поступово переходити з іншого фреймворку або впроваджувати Vue.js в різноманітні проектні середовища. [2]

1.2. Визначення поняття фреймворку Next.js

Next.js – потужний фреймворк для веб-розробки, який спрощує процес створення швидких та інтерактивних додатків. В основі фреймворку – React, популярна JavaScript бібліотека. Next.js пропонує додаткову структуру та функціонал, такі як серверний рендеринг та статична генерація. Next.js – frontend-

фреймворк, який базується на можливостях React, універсальної JavaScript бібліотеки, що використовується для створення інтерфейсів користувача. Основна перевага Next.js полягає в його здатності спрощувати складні аспекти веб-розробки, надаючи повний набір інструментів та функціоналу.

Одна з ключових функцій, яку пропонує Next.js, – серверний рендеринг, який значно покращує продуктивність та оптимізацію для пошукових систем (SEO) веб-додатків. Це означає, що веб-сайти, побудовані з використанням Next.js, завантажуються швидше і мають кращий рейтинг у пошукових результатах.

Ще одна відмінна особливість Next.js – це статична генерація. Ця функція передбачає попереднє генерування файлів під час побудови, що дозволяє їм бути обслуговуваними заздалегідь без очікування на завантаження даних. Файли потім розповсюджуються глобально, що забезпечує ще швидше завантаження.

Next.js також пропонує інкрементальну статичну регенерацію, яка поєднує статичну генерацію з динамічними оновленнями для елементів, які не були згенеровані статично або змінилися з моменту першої генерації. [3]

У підсумку, цей потужний інструмент надає розробникам все необхідне для створення повноцінних інтерактивних, динамічних та продуктивних веб-додатків.

1.3. Опис архітектури та основних особливостей Next.js

Архітектура Next.js базується на компонентах, які називаються "сторінками" (pages). Кожна сторінка може бути файлом JavaScript, TypeScript, або JSX, що містить React-компонент, який відповідає за відображення цієї сторінки.

У основі Next.js знаходяться кілька ключових функцій, які відрізняють його від інших фреймворків для веб-розробки. Одна з таких функцій - інноваційний

підхід до рендерингу, що пропонує як серверний рендеринг, так і генерацію статичних сайтів. Серверний рендеринг дозволяє отримувати дані під час запиту, покращуючи продуктивність і можливості SEO. З іншого боку, генерація статичних сайтів дозволяє створювати файли під час компіляції та обслуговувати їх заздалегідь, покращуючи швидкість завантаження.

Next.js також відмінно справляється з розділенням коду, що забезпечує завантаження лише необхідного коду для кожної сторінки, роблячи додатки швидшими та ефективнішими. Ця функція значно підвищує час відповіді, що призводить до кращого користувацького досвіду.

Ще одна важлива функція Next.js - це маршрутизація на основі файлів, яка дозволяє розробникам здійснювати маршрутизацію файлів більш легко і ефективно. Next.js дозволяє відображати URL-адреси безпосередньо як файли у каталозі проекту, що дозволяє зробити файли маршрутами та прискорити навігацію.

Ще однією перевагою є Hot Module Replacement (HMR), який дозволяє змінювати код в реальному часі без повного перезавантаження. Це прискорює процес розробки та робить його ефективнішим для розробників.

Next.js також має вбудовану обробку помилок, що дозволяє розробникам створювати власні сторінки помилок. Коли розробник стикається з аналогічною помилкою, Next.js відображає відповідну сторінку помилки з рішенням, що зберігає час і зусилля розробника.

Також Next.js пропонує підтримку API маршрутів, яка спрощує створення бекенд API за допомогою вбудованої підтримки для API маршрутів. Це дозволяє розробникам швидко та ефективно налаштовувати API для взаємодії з клієнтською частиною додатку.

Крім того, Next.js підтримує як CSS модулі, так і бібліотеки CSS-in-JS, що надає гнучкість у налаштуваннях стилів. Це означає, що розробники можуть використовувати різні підходи до стилізації веб-сайтів, забезпечуючи при цьому зручність у роботі зі стилями та підтримку чистоти коду. [3]

Завдяки цим особливостям, Next.js стає потужним інструментом для розробки веб-додатків, який дозволяє створювати швидкі, масштабовані та ефективні проекти з мінімальними зусиллями.

РОЗДІЛ 2. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1. Приклади використання Next.js

Next.js є універсальним фреймворком, здатним вирішувати різноманітні потреби веб-розробки в різних галузях промисловості. Одним із найбільш поширених є його використання у комерційних сферах. Завдяки швидкому рендерингу та потужним можливостям SEO, він надає оптимальне рішення для створення високопродуктивних онлайн-магазинів, які добре індексуються в пошукових системах.

Блоги та маркетингові веб-сайти також часто використовують технологію Next.js. Його функції серверного рендерингу та статичної генерації контенту дозволяють швидко змінювати вміст, що робить його ідеальним для платформ, які часто оновлюють свій контент.

Додатки новин та веб-сайти документації користуються можливістю Next.js ефективно обробляти оновлення у реальному часі. Так само розважальні платформи та веб-додатки, створені спільнотою, можуть використовувати його можливості для надання інтерактивних користувацьких інтерфейсів з високою продуктивністю.

Додатки для бронювання, аукціонні сайти, інформаційні центри – усі ці типи застосунків можуть бути покращені за допомогою Next.js через його швидкість, універсальність та масштабованість.

2.2. Дослідження проектів, які використовують Next.js

Для проведення аналізу існуючих проектів, які використовують фреймворк Next.js, було проведено огляд різноманітних веб-додатків та сайтів, розроблених з використанням цього фреймворку. Було розглянуто різноманітні додатки такі, як Twitch [4], Spotify [5], Nike [6] та Solana [7]. Дослідження включало аналіз наступних аспектів:

Функціональність та складність проектів: Було досліджено широкий спектр проектів, від невеликих інформаційних сайтів до складних корпоративних веб-додатків: серед яких e-commerce та платформи типу соціальних мереж. Аналізувалися основні функціональні можливості кожного проекту та його рівень складності.

Продуктивність та швидкодія: Оцінювалася продуктивність та швидкодія проектів, включаючи час завантаження сторінок, реактивність інтерактивних елементів та загальну відповідь сервера.

SEO-оптимізація: Проводилася перевірка наявності та якості SEO-оптимізації в проектах, так як Next.js забезпечує можливості для покращення індексації в пошукових системах.

Розширюваність та підтримка: Аналізувалася здатність кожного проекту до розширення та масштабування, а також наявність активної підтримки спільноти та оновлень.

Дизайн та користувацький досвід: Оцінювалася естетика та зручність використання інтерфейсу кожного проекту з точки зору дизайну та користувацького досвіду.

На основі проведеного дослідження зроблено висновки щодо ефективності та придатності фреймворку Next.js для реалізації різних типів веб додатків та сайтів, а також виявлені можливості для подальшого вдосконалення та розвитку.

2.3. Визначення сильних та слабких сторін фреймворку

Сильні сторони фреймворку Next.js:

По-перше, ця технологія чудовий вибір для власників e-commerce бізнесу, ціль яких збільшення продажів. Використання Next.js допомагає досягти цієї мети, оскільки він надає повний контроль над остаточним дизайном веб-сайтів, онлайн-магазинів, додатків та інших цифрових продуктів. Крім того, немає обмеження щодо тем або плагінів, присвяченим конкретній платформі електронної комерції або системі управління контентом (CMS).

Інші переваги NextJS для бізнесу включають:

Безпека даних. Веб-сайти, створені за допомогою NextJS, є статичними, що означає відсутність прямого підключення до бази даних, залежностей, даних користувачів або іншої конфіденційної інформації. Це гарантує безпеку даних.

Швидкість виходу на ринок. NextJS - це чудовий спосіб створити Мінімально життєздатний продукт (MVP) якомога швидше завдяки великій кількості готових компонентів. Такий спосіб побудови дозволяє отримувати швидкий зворотний зв'язок і вдосконалювати свій продукт без витрати часу та грошей.

Повна всеканальність. Веб-сайти та додатки, створені за допомогою NextJS, доступні з будь-якого пристрою, тому є можливість продавати продукти та послуги через різні канали продажу.

Короткий час завантаження сторінок. Статичні веб-сайти є швидкими за своєю природою, тому відвідувачі та клієнти будуть задоволені продуктивністю веб-сайтів і веб-додатків NextJS.

Покращена SEO-оптимізація: Фреймворк надає зручні можливості для вдосконалення SEO-параметрів, що сприяє підвищенню видимості проектів у пошукових системах.

Підтримка. Популярність послуг розробки на NextJS, а також самого фреймворку NextJS, зростає, так само, як і кількість розробників. Через це за необхідності буде легко знайти агентство або фрілансера для внесення змін.

Переваги використання Next.js для розробників:

Стрімінг елементів додатка користувачу. За допомогою React Server Components, елементи додатка можуть бути відображені користувачам миттєво, щоб вони могли взаємодіяти з додатком без очікування повного завантаження.

Покращений швидкий рендеринг. Next.js дозволяє швидкий рендеринг додатків за допомогою статичного рендерингу, що покращує продуктивність і відповідь додатка.

Вища ефективність розробки. Next.js забезпечує вищу ефективність розробки завдяки центрованій на сервері маршрутизації, що дозволяє легше керувати маршрутами та покращує організацію додатку.

Built-in Image Component та Automatic Image Optimization: Ця функція автоматично оптимізує зображення і тепер підтримує формат AVIF, що дозволяє зменшити розмір зображень на 20% порівняно з форматом WebP.

Інкрементне статичне відтворення: Це дозволяє веб-розробникам оновлювати існуючі сторінки, перерендерюючи їх у фоновому режимі при вході трафіку. Таким чином, статичний контент може стати динамічним.

Rust-based compiler SWC: SWC забезпечує трансформацію і мініфікацію коду JavaScript для продукції. Next.js включає зовсім новий компілятор Rust, який оптимізує збирання та компіляцію.

Гнучкість та розширюваність: Next.js дозволяє легко розширювати та масштабувати проекти, а також має активну спільноту та підтримку, що дозволяє вирішувати проблеми та вдосконалювати фреймворк.

Хоча Next.js швидко розвивається і надає багато функціональностей, все ж в нього є слабкі сторони:

Відсутність вбудованого менеджера стану. Тому, якщо є необхідність використовувати менеджер стану, також потрібно установлювати Redux, MobX або щось подібне.

Мала кількість плагінів. Порівняно з Gatsby.js, не можна використовувати багато простих плагінів.

Велика кількість варіантів конфігурації: Для початківців або тих, хто не має досвіду у роботі з фреймворками, велика кількість конфігурацій може стати перешкодою та ускладнити процес розробки.

Вивчення нових концепцій: Для повного використання можливостей Next.js розробникам потрібно освоїти нові концепції, такі як серверний рендеринг, що може зайняти певний час та зусилля.

Ризик перенавантаження: При неправильному плануванні та розробці складних проектів може виникнути перенавантаження та ускладнення у підтримці коду.

Незважаючи на деякі обмеження, фреймворк Next.js є потужним інструментом для розробки різних типів веб-додатків та сайтів. Його сильні сторони, такі як швидкодія, SEO-оптимізація та гнучкість, роблять його популярним серед розробників та бізнесів, які прагнуть створити високоякісні та ефективні цифрові рішення. [9] [10]

2.4. Порівняння Next.js з іншими популярними веб-фреймворками

Порівняння Next.js з іншими популярними веб-фреймворками може допомогти визначити найкращий вибір для конкретного проекту. Тепер давайте порівняємо Next.js з деякими популярними фронтенд-фреймворками та побачимо, як вони виправдовують очікування.

React.js, базова бібліотека Next.js, пропонує компонентний підхід до побудови інтерфейсів користувача. Хоча React.js фокусується на рівні представлення, Next.js розширює свої можливості, додавши можливості серверного рендерингу та маршрутизації. Це дозволяє Next.js забезпечувати кращу продуктивність та оптимізацію для пошукових систем порівняно з традиційним застосуванням React.js.

Angular - це повноцінний фронтенд-фреймворк, який забезпечує міцну інфраструктуру для побудови складних додатків. Він пропонує повнофункціональне середовище розробки, включаючи потужні інструменти для зв'язку даних, впровадження залежностей та тестування одиниць. Однак складне налаштування Angular роблять його менш вдалим вибором для невеликих та середніх проектів, де більше підходить простота використання та зручність.

Vue.js - це легкий та універсальний фронтенд-фреймворк, який поєднує найкращі функції React.js та Angular. Він пропонує поступове вивчення, відмінну продуктивність та багату екосистему плагінів і бібліотек. Хоча Vue.js забезпечує велику гнучкість та простоту, Next.js вирізняється, коли мова йде про серверний рендеринг та генерацію статичних сайтів. [8]

Загалом, кожен з цих фреймворків має свої переваги та недоліки, і вибір між ними буде залежати від конкретних потреб проекту. Next.js вирізняється тим, що забезпечує ефективний досвід розробки для проектів з різноманітними вимогами.

Підсумовуючи, якщо є необхідність у продуктивності, оптимізації для пошукових систем та простоті, то Next.js стає першим вибором для розробки веб-застосування.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ ПІДХОДІВ РОЗРОБКИ НА NEXTJS

3.1. Розгляд підходів до розробки за допомогою Next.js

Next.js - це потужний фреймворк, який може бути використаний для розробки різноманітних веб-додатків. Існують різні підходи до розробки за допомогою Next.js, включаючи такі популярні методології:

3.1.1. Статичне генерування (Static Site Generation, SSG).

Статичне генерування є одним з основних підходів до розробки на Next.js. Цей підхід дозволяє створювати статичні файли HTML на підставі даних, що зберігаються в головному файлі проекту. При використанні статичного генерування, при першому запуску проекту або при зміні даних, Next.js генерує статичні файли, які потім можуть бути розгорнуті на сервері. Цей підхід дозволяє зменшити навантаження на сервер і покращити швидкість завантаження сторінок. Основною перевагою статичного генерування є можливість кешування результатів, що забезпечує швидкий доступ до сторінок. Однак, недоліком цього підходу є необхідність редагувати файли та перезавантажувати сервер при зміні даних. Приклади використання статичного генерування включають створення статичних сайтів, блогів, новинних порталів та інших статичних сторінок. [5]

Основними принципами статичного генерування на Next.js є збереження даних в головному файлі проекту у вигляді реактивних компонентів React, що дозволяє автоматично відслідковувати зміни даних і оновлювати статичні файли при необхідності. Також важливо передбачити можливість динамічної генерації статичних файлів за допомогою параметрів запиту, що дозволяє створювати різні версії сторінок залежно від вхідних даних. Крім того, обробка запитів на стороні

сервера може доповнити статичну генерацію та надати більше можливостей для взаємодії з клієнтом.

Статичне генерування на Next.js має свої переваги та недоліки. Перевагами є висока швидкість відображення сторінок, зменшення навантаження на сервер та можливість кешування результатів. Це дозволяє забезпечувати швидкий доступ до статичних файлів та покращувати користувацький досвід. Однак, недоліком є необхідність редагувати файли та перезавантажувати сервер при зміні даних. Це може бути не зручною процедурою під час активної розробки або при великій кількості даних. Також, статичне генерування недостатньо гнучке для динамічних змін на стороні клієнта. Тому перед використанням цього підходу слід ретельно оцінити вимоги проекту та вибрати оптимальний варіант.

Статичне генерування на Next.js знаходить застосування в різних типах проектів. Прикладами використання є створення статичних сайтів, документаційних порталів, блогів, новинних сайтів та інших статичних сторінок. Використання статичного генерування дозволяє забезпечувати швидку ініціалізацію проекту, а також зменшити навантаження на сервер при одночасному збереженні вигод динамічного контенту. Для статичного генерування на Next.js можна використовувати різні додаткові бібліотеки та інструменти, що розширюють його можливості і дозволяють створювати дійсно цікаві та потужні статичні сторінки. [6]

3.1.2. Серверний рендеринг (Server-Side Rendering, SSR).

Це процес створення та відправлення сторінки на сервері, до того як вона буде відображена на клієнтській стороні. Основним принципом серверного рендерингу в Next.js є використання сервера для обробки запиту клієнта та рендерингу сторінки перед тим, як вона буде доставлена до клієнта. Це дозволяє отримати швидкий перший вихід на екран та поліпшити визначення контенту сайту для пошукових систем. Більшість контенту генерується на сервері,

забезпечуючи високу продуктивність та кращі показники завантаження сторінки для користувачів.

Переваги серверного рендерингу (SSR) в Next.js включають швидкий перший вихід на екран, поліпшення SEO-показників завдяки повному контенту на сторінці при її завантаженні, кращий доступ користувачів до контенту на медленних пристроях та покращене індексування сторінок пошуковими системами. Однак, цей підхід може призвести до збільшення навантаження на сервер та затримок у відображенні динамічного контенту, особливо на великих проектах з великою кількістю даних. [6]

Одним з прикладів використання серверного рендерингу (SSR) в Next.js є інтернет-магазин, де потрібно швидко відображати товари та сторінки категорій. Використання SSR дозволяє швидко показувати користувачам сторінки з товарами та їхньою інформацією, що поліпшує користувацький досвід і збільшує імовірність покупок.

Інший приклад - новинний сайт або блог, де важливо, щоб статті відображалися швидко та були доступні для індексації пошуковими системами. За допомогою SSR, сторінки з новинами можуть бути побудовані на сервері та швидко відправлені користувачам для перегляду, що підвищує рейтинг сайту у пошукових системах і збільшує трафік.

Також, адміністративні панелі та панелі керування, де важлива швидкість та безпека обробки даних, можуть використовувати SSR для ефективного відображення та обробки інформації на сервері перед її надсиланням клієнтському браузеру. [5]

3.1.3. Гібридний підхід.

Next.js дозволяє комбінувати SSG та SSR для побудови гібридних додатків, де певні сторінки генеруються статично, а інші - динамічно на сервері за кожен

запит. Цей підхід дозволяє забезпечити оптимальну швидкодію та SEO-оптимізацію для всіх типів сторінок. [6]

3.1.4. SPA з використанням API (Single Page Application, SPA).

Хоча Next.js спеціалізується на SSR та SSG, його також можна використовувати для створення SPA, де весь контент завантажується асинхронно через API. Цей підхід може бути використаний для створення додатків зі складною взаємодією на клієнтському боці. [6]

3.2. Визначення оптимальних практик та стандартів коду

Для досягнення ефективності, зрозумілості та підтримки коду в проектах на основі Next.js, важливо дотримуватися оптимальних підходів та стандартів коду. Ось деякі рекомендації щодо цього:

- Використання сучасного стандарту JavaScript дозволяє використовувати нові функції та синтаксис, що полегшує розробку та робить код більш зрозумілим.
- Важливо створити логічну та чітку структуру проекту, розділивши його на компоненти, сторінки, сервіси та інші частини.
- Для досягнення найбільшої продуктивності створених додатків варто правильно використовувати серверні та клієнтські компоненти і оптимально розділити завдання між ними.
- Використання компонентів для організації UI та логічних частин додатку значно полегшує можливість використання коду повторно та забезпечує його модульність.
- Для оптимізації завантаження зображень на сайті потрібно використовувати влаштований у Next.js – next-image.

- Використання стандартних засобів Next.js для маршрутизації дозволяє створювати реактивні та SEO-оптимізовані додатки. Зокрема, файловий роутинг є найзручнішим і найбільш швидким методом у новій версії фреймворку.

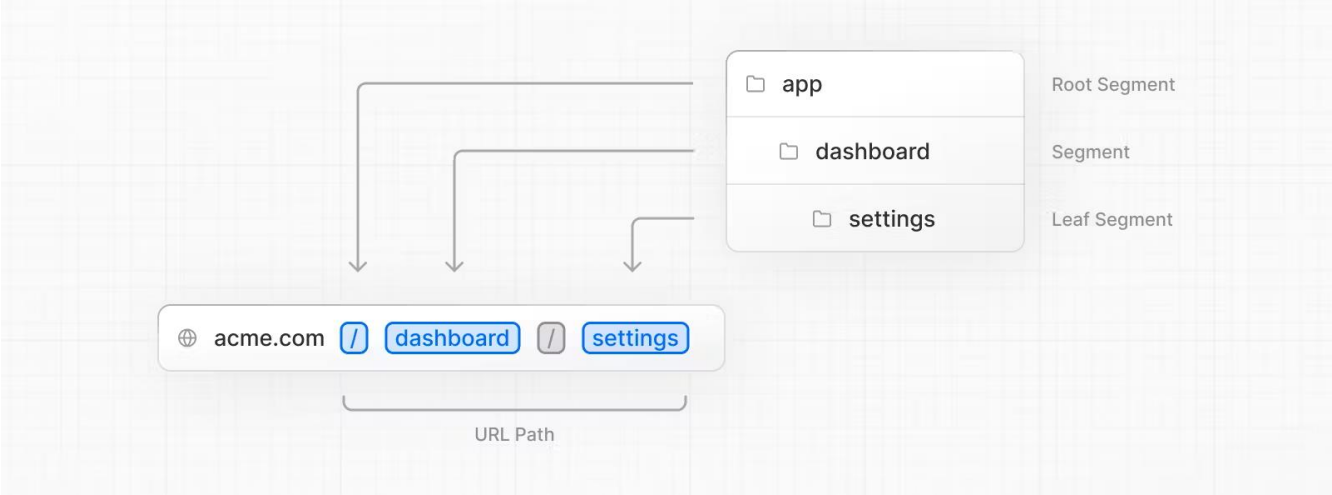


Рис. 1 - File-based Routing Schema [7]

- Використання TypeScript підвищує безпеку типів, що полегшує виявлення помилок та підвищує читабельність коду.
- Фреймворк Next.js є також відомий SEO-оптимізацією, однак для того, щоб отримати найоптимальніший результат, рекомендовано створити файл robots.txt, а також додати усі важливі мета-теги. Інакше – сайт не індексуватиметься пошуковими системами.
- Під час розробки потрібно звертати увагу на продуктивність додатку, оптимізовуючи його шляхи рендерингу, завантаження ресурсів та інші аспекти.
- Важливо забезпечити якісну документацію та коментування коду, що допоможе іншим розробникам легше розуміти наявний код та швидше внести в нього зміни.

Загалом, аналіз різних підходів до розробки на Next.js вказує на їх значний потенціал у покращенні не тільки технічної якості веб-додатків, але й загальної

ефективності розробки. Ці висновки підтверджують важливість інтеграції сучасних технологій та підходів у розробці веб-додатків та можуть слугувати основою для подальших досліджень і впровадження нових функціональних рішень у цій галузі.

РОЗДІЛ 4 . ПРОЕКТУВАННЯ ВЛАСНОЇ МЕТОДОЛОГІЇ

4.1. Розробка власної методології використання Next.js.

Розробка методології полягає у створенні готового шаблону NextJS проєкту, у якому враховано усі найкращі практики коду згадані раніше та налаштовано інструменти, необхідні для створення повноцінного додатку. Крім того, головною метою такого шаблону є простота і структурованість коду та можливість його адаптивного використання для швидкої розробки додатків на будь-яку тематику.

4.1.1. Конфігурація стилів та тем.

Для методології важлива зрозумілість та простота коду, а також легкість кастомізації, саме на це варто спиратись при виборі бібліотеки стилів для готових шаблонів. З документації відомо, що найпопулярнішими інструментами стилізації елементів у NextJS є глобальні CSS стилі та CSS-бібліотека Tailwind. Ці елементи можна з легкістю вбудувати у проєкт за лічені хвилини, однак є питання до читабельності CSS коду і налаштуванню тем. З цим завданням чудово справляється бібілотека React-компонентів Material UI, адже вона дозволяє створювати теми, які автоматично поширюються на всі елементи додатку. [11]

Зважаючи на ці переваги, проєкт буде зроблений на основі MUI. Для конфігурації цієї бібіліотеки в NextJS необхідно інсталювати потрібні пакети за допомогою команди “npm install @mui/material @emotion/react @emotion/styled”. Після цього створити файл theme.ts для створення стилів темної та світлої теми, у яких визначаються основні, другорядні, фонові, текстові кольори і так далі, а також додати функцію getDesignToken(). Ця функція приймає параметр mode (або 'light', або 'dark') і повертає відповідний об'єкт теми на основі вказаного режиму. Якщо mode - 'light', вона повертає об'єкт lightTheme. Якщо mode - 'dark', вона повертає об'єкт darkTheme. [11]

Наступним етапом є під’єднання цих стилів до всіх компонентів додатку. У проєктах NextJS існує файл layout.ts, у якому можна визначити дані, котрі мають бути присутні на усіх сторінках веб-застосування. Тому компонент RootLayout у цьому файлі є головним компонентом макету додатка Next.js, який відповідає зокрема і за підключення тем до всього додатку.

Для початку за допомогою useState створюється стан для зберігання режиму теми (mode), який початково встановлений як "light". З useMemo, створюється тема за допомогою createTheme, яка базується на дизайн-токенах, отриманих з функції getDesignTokens(mode). Відповідно, коли mode змінюється, тема також перетворюється. Потім додаток огортається у ThemeProvider, який і надає всім елементам задану тему. Додатково CssBaseline встановлює стилі базових елементів для вирівнювання візуального вигляду в різних браузерах.

Отже, весь додаток підключається до теми через ThemeProvider, відображаючи його з встановленою темою та базовими стилями. А за допомогою React хуків – useMemo та useState режим теми динамічно змінюється.

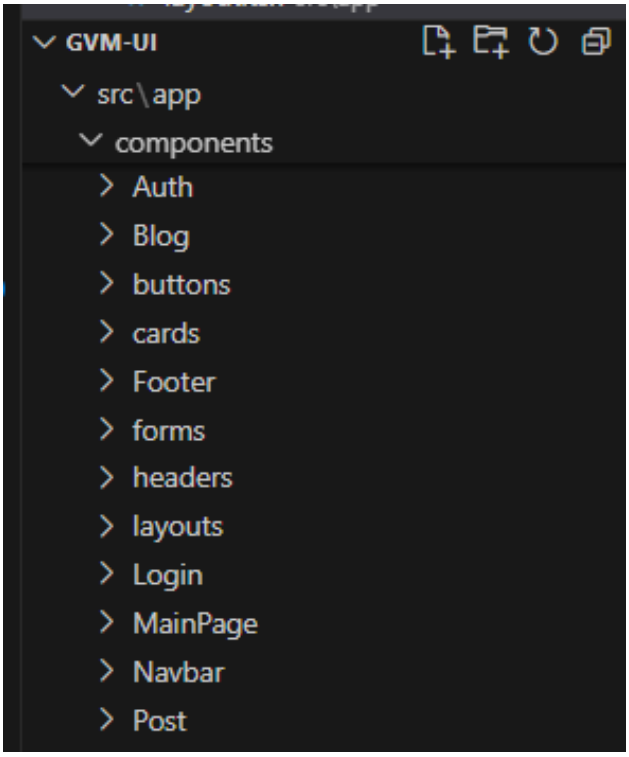


Рис. 2 - Бібліотека власних компонентів

4.1.2. Розробка власних reusable компонентів. Розробка власних повторно використовуваних компонентів є невід'ємною частиною сучасної веб-розробки, оскільки це дозволяє значно прискорити процес розробки та забезпечити більшу гнучкість та сумісність компонентів у різних частинах проекту. Одним із ключових аспектів цієї методології є використання утиліт, які зберігають дані компонентів, дозволяючи легко модифікувати їх відповідно до потреб користувача.

Для прикладу, розглянемо компонент Navbar, який використовує дані з файла navbarUtils.ts для генерації елементів навігаційної панелі. Це робить компонент гнучким і легко налаштовуваним, оскільки всі зміни можна проводити, вносячи корективи лише в один файл. Крім того, компонент включає різні режими відображення для різних розмірів екранів, використовуючи Drawer для мобільних версій і горизонтальне меню для більших екранів. Це забезпечує гарне візуальне представлення на всіх пристроях. Стилізація виконується за допомогою MUI тем і підтримується кастомізація через sx властивості, що дозволяє легко змінювати зовнішній вигляд елементів без зміни самого компонента.

Цей підхід дозволяє значно знизити час на додавання та налаштування нових елементів у навігаційну панель, забезпечуючи гнучкість та швидкість розробки великих проектів.

У контексті розробки власних повторно використовуваних компонентів, важливим аспектом є створення універсальних, налаштовуваних елементів, які

можна легко інтегрувати у різноманітні частини вашого додатку. Нижче розглянемо приклади кількох типів компонентів: кнопок, карток та макетів, які демонструють їхню користь та внесок у зручність та ефективність розробки.

GradientArrowButton - це кнопка з градієнтним фоном та анімацією, що зміщується при наведенні курсору. Вона використовує MUI Button для базової структури і дозволяє задавати динамічний фон, текст та інші стилізаційні параметри через пропси. Для розробників – це можливість швидко додати візуально привабливу кнопку з анімацією, у якій висока ступінь кастомізації забезпечує легку адаптацію до різних стилів і потреб проекту. Методологія містить кілька таких компонентів з різними типами стилізації і можливістю додаткової кастомізації.

Найкращим способом відображення інформації є використання різноманітних карток. У розробленій методології є шість видів компонентів карток з різною тематикою. Оскільки для більшості комерційних послуг важливо містити інформацію про цінові опції на сайті, то в методології створено також відповідний компонент. PriceCard - картка, яка відображає інформацію про ціноутворення. У ній використано елемент Card з MUI для структури та декілька внутрішніх компонентів для відображення ціни, заголовка, підзаголовка та переліку опцій.

Уніфікація представлення цінових планів забезпечує консистентність інтерфейсу, а для розробників це необхідний компонент, який легко інтегрувати із динамічними даними, такими як ціни та опції, що змінюються.

Для того, щоб усі сторінки додатку виглядали гармонійно і симетрично, до методології було додано різні макети, які задають позиціонування елементів та надають відмінну адаптивність. Наприклад, один із компонентів – StandardLayout – базовий макет, який включає заголовок та вміст, вирівнюваний згідно з параметрами. Це флексибільна обгортка, яка приймає додаткові компоненти як

children, дозволяючи легко створювати сторінки з заданими характеристиками заголовка та вмісту та різноманітним контентом.

Така стандартизація макетів сторінок забезпечує єдиний стиль та структуру, сприяючи більшій організованості та меншому часу на розробку. А можливість передачі будь-якого вмісту як children забезпечує високу гнучкість та перевикористання компоненту в різних контекстах.

Ці приклади демонструють, як належно створені і кастомізовані компоненти можуть поліпшити ефективність процесу розробки, забезпечуючи розробникам зручні інструменти для створення високоякісних веб-додатків.

4.1.3. Конфігурація бази даних.

Однією з найбільш важливих досягнень нової версії NextJS є можливість виконання частин коду на сервері. Саме ця опція надає можливість створювати full-stack додатки навіть без сторонніх backend технологій. Адже серверний компонент NextJS напряму “комунікує” з базою даних за допомогою Prisma ORM. [12]

Prisma — це сучасна ORM (Object-Relational Mapping), яка дозволяє розробникам взаємодіяти з базою даних за допомогою об'єктно-орієнтованого підходу. Вона спрощує процес написання запитів до бази даних шляхом надання потужного набору інструментів, які автоматизовують багато рутинних завдань, пов'язаних із керуванням базою даних. [12]

Основні особливості Prisma включають:

- Prisma Client: Автоматично генерований клієнт для виконання запитів до бази даних, який гарантує безпеку типів завдяки TypeScript підтримці.
- Prisma Migrate: Система міграцій, яка дозволяє керувати схемою бази даних через декларативні файли міграцій.

- Prisma Studio: Графічний інтерфейс для перегляду та управління даними в базі даних.

Тож використання Prisma в Next.js проекті сприяє швидкій розробці, оскільки Prisma забезпечує багато вбудованих рішень для типових завдань взаємодії з базою даних. Розробники можуть зосередитися на бізнес-логіці, а не на низькорівневому управлінні даними. Prisma підтримує багато популярних баз даних (наприклад, PostgreSQL, MySQL, SQL Server, SQLite), що робить її відмінним вибором для різноманітних проектів. Незалежно від вибраної бази даних, Prisma забезпечує однаково зручний досвід розробки. [12]

Тому найкращим варіантом буде використати саме цей інструмент для забезпечення зв’язку із даними в шаблоні. Сама ж база даних створена з PostgreSQL із хостингом на платформі Vercel.

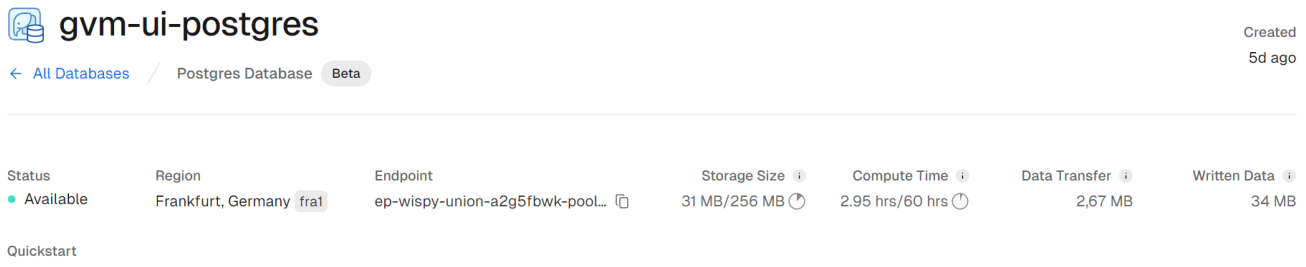


Рис. 3 - База даних на Vercel

Для інтеграції бази даних PostgreSQL за допомогою Prisma ORM у проект Next.js було виконано декілька ключових кроків. Перш за все це встановлення Prisma CLI та ініціалізація через команди npm, потім було використано команду prisma init для створення необхідних конфігураційних файлів у проекті Next.js.

Наступним кроком потрібно визначити конфігурацію клієнта Prisma (generator client) та налаштування джерела даних (datasource db) у файлі schema.prisma. Ось як виглядає файл конфігурації:

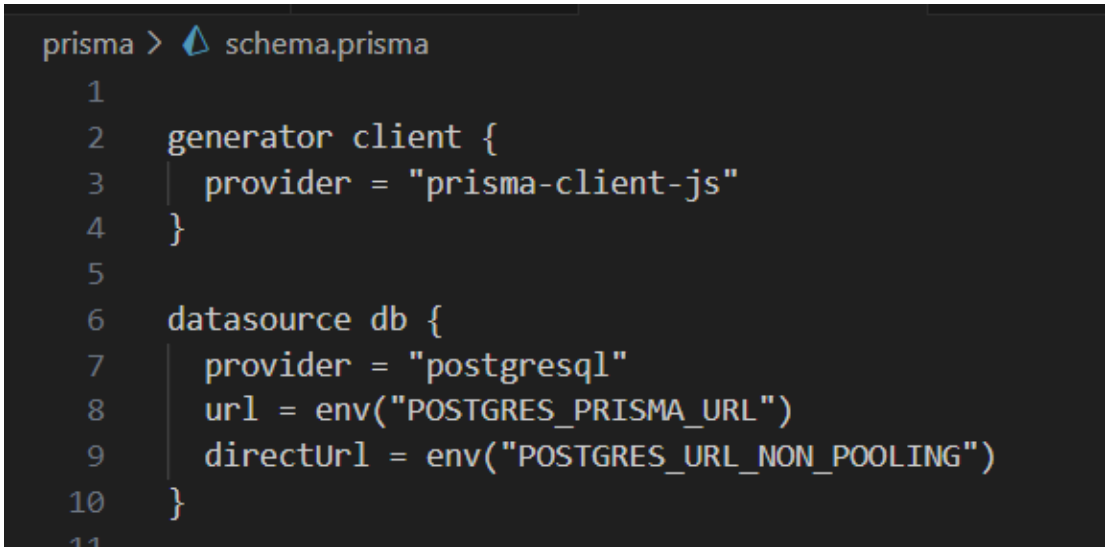


Рис. 4 - Налаштування БД

Далі створено синглтон PrismaClient для оптимізації використання підключення до бази даних у серверному середовищі Next.js:

Рис. 5 - Prisma Client

Цей код гарантує, що у виробничому режимі буде створено лише один екземпляр PrismaClient, запобігаючи надлишковому використанню ресурсів.

Для створення стандартної панелі адміна потрібно створити таблицю з даними адміна і, наприклад, пости, до створення яких лише він матиме доступ. Усі моделі записуються у файлі schema.prisma і після команди “prisma migrate dev” усі зміни зберігаються у базу даних на сервері.

prisma > schema.prisma

```
11
12  model Admin {
13    id        Int    @id @default(autoincrement())
14    nickname  String
15    email     String @unique
16    password  String
17    posts     Post[]
18  }
19
20  model Post {
21    id             Int      @id @default(autoincrement())
22    title          String
23    category       String
24    author         String
25    description    String
26    image          String
27    createdAt      DateTime @default(now()) @map(name: "created_at")
28    lastModifiedAt DateTime @updatedAt @map(name: "updated_at")
29    adminId        Int
30    admin          Admin    @relation(fields: [adminId], references: [id])
31  }
32
```

Рис. 6 - Моделі бази даних

Для взаємодії з базою даних у коді Next.js треба просто імпортувати раніше створений Prisma Client. Наприклад, отримати список постів можна, написавши такий рядок коду:

```
const posts = await prisma.post.findMany();
```

Цей код демонструє, як легко можна виконувати запити до бази даних, використовуючи Prisma. Однак важливо викликати ці дані лише всередині серверних компонент NextJS. [12]

Я вважаю, що включення Prisma і PostgreSQL у шаблон проекту Next.js може значно підвищити якість кінцевого продукту, спростити розробку та забезпечити стабільну та масштабовану основу для розробки веб-додатків. Адже, методологія, яка включає готові до використання налаштування бази даних, надає шаблону високий ступінь гнучкості. Розробники можуть легко налаштовувати та розширювати базовий шаблон, адаптуючи його під конкретні потреби проекту без потреби переписування коду доступу до даних. А Prisma Migrate дозволяє легко керувати змінами у структурі бази даних, що є важливим для підтримки консистентності даних у довгостроковій перспективі. Шаблон, який інтегрує ці можливості, допомагає забезпечити, що всі розробники працюють з останніми змінами бази даних.

4.1.4. Застосування можливостей бібліотеки NextAuth для авторизації.

При написанні веб-сайтів з фреймворком NextJS є можливість реалізувати різноманітні способи авторизації за допомогою бібліотеки NextAuth. [13]

NextAuth.js — це повноцінне відкрите рішення для автентифікації у застосунках Next.js. Ця бібліотека спеціально розроблена для підтримки Next.js та serverless-архітектур, пропонуючи гнучкість та легкість використання з будь-якою OAuth службою. Основні переваги включають підтримку безпарольної автентифікації, можливість використання станової автентифікації з будь-якою базою даних, підтримка як токенів JSON Web, так і сесій у базі даних, а також можливість роботи без бази даних. Це робить NextAuth.js цінним доповненням до будь-якого проекту на Next.js, який потребує надійної системи автентифікації. [13]

Додавання NextAuth у методологію NextJS дозволить підвищити захист даних користувачів та зберегти контроль над ними, пропонуючи використання власної бази даних, інтегрувати різні методи автентифікації, забезпечуючи гнучкість у виборі оптимального варіанту для кожного конкретного застосунку. Для розробленої методології бібліотеку NextAuth було використано для авторизації адміна, котрий може вносити зміни на сайт, і для авторизації різних користувачів за допомогою акаунта на Github. [13]

NextAuth інстальовується, як і всі базові пакети npm через команду npm install next-auth. Потім, у новій версії NextJS 14, конфігурацію бібліотеки можна налаштувати у файлі api/auth/[…next-auth]/routes.ts. Завдяки цьому файлу всі запити до шляху api/auth/signin, /signout, /callback будуть оброблені автоматично самою бібліотекою. Також у роуті задаються бажані способи авторизації у системі. Наприклад, щоб додати авторизацію через github, у об’єкт NextAuthOptions треба додати спеціальний провайдер з пакету “next-auth/provider” і створити OAuth застосунок, у якому містяться github ключі, які дозволяють відслідковувати авторизованих користувачів. [13]

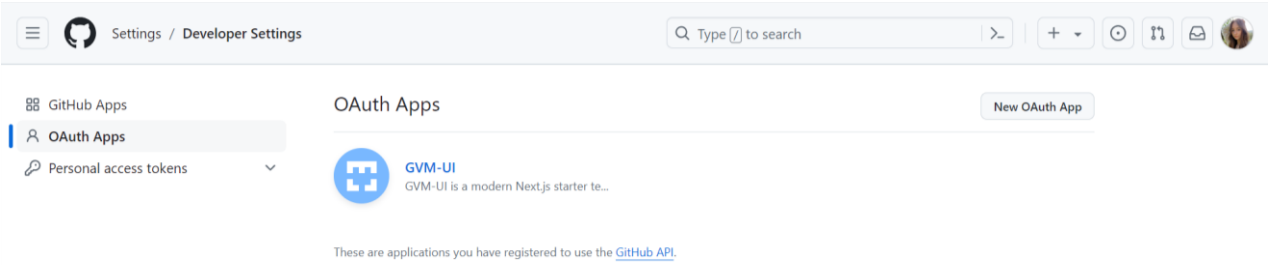


Рис. 7 - Github OAuth

```
export const authOptions: NextAuthOptions = {
  pages: {
    signIn: '/login',
  },
  providers: [
    GithubProvider({
      clientId: process.env.GITHUB_CLIENT_ID as string,
      clientSecret: process.env.GITHUB_SECRET_ID as string,
    }),
    CredentialsProvider({
      name: "Credentials",
      credentials: {
        email: {label: "Email", type: "text"},
        password: {label: "Password", type: "password"},
      },
      async authorize(credentials) {
        if (!credentials?.email || !credentials?.password)
          return null as any;

        try{
          const admin = await login(credentials.email, credentials.password);
          return admin;
        } catch (error) {
          console.log(error);
          return null;
        }
      }
    })
  ],
  secret: process.env.NEXTAUTH_SECRET as string,
};
```

Рис. 8 - Варіанти авторизації

Важливою опцією є авторизація адміністратора системи за паролем. Для реалізації такої можливості з NextAuth, необхідно додати Credentials провайдер з функцією authorize, котра порівнює правильність введених користувачем даних із інформацією збереженою у базі даних. [13]

4.1.5. SEO-оптимізація веб-додатку.

Однією з основних переваг створення додатків з допомогою фреймворку NextJS є відмінна індексація у пошукових системах. Однак для отримання найкращих показників SEO необхідно додати об’єкт метаданих в проєкт. Метадані не тільки покращують SEO, але й кастомізовують взаємодію з користувачами через соціальні мережі. Основні компоненти метаданих включають:

- Title: Заголовок веб-сторінки, який зазвичай відображається на вкладках браузера та у результатах пошукових систем.
- Description: Короткий опис сторінки, який зазвичай використовується пошуковими системами для відображення під заголовком у пошукових результатах.
- Keywords: Ключові слова, які допомагають пошуковим системам зрозуміти, для яких тем або запитань ваш сайт може бути релевантним.
- Open Graph: Серія метаданих, які використовуються соціальними мережами, як Facebook, для формування привабливих "карток" при поширенні посилання. Включає заголовок, опис, URL та тип (у цьому випадку "website"), що дозволяє краще контролювати як виглядатиме посилання, коли його поширюватимуть.
- Twitter Card: Спеціальні метадані для представлення сайту на Twitter.

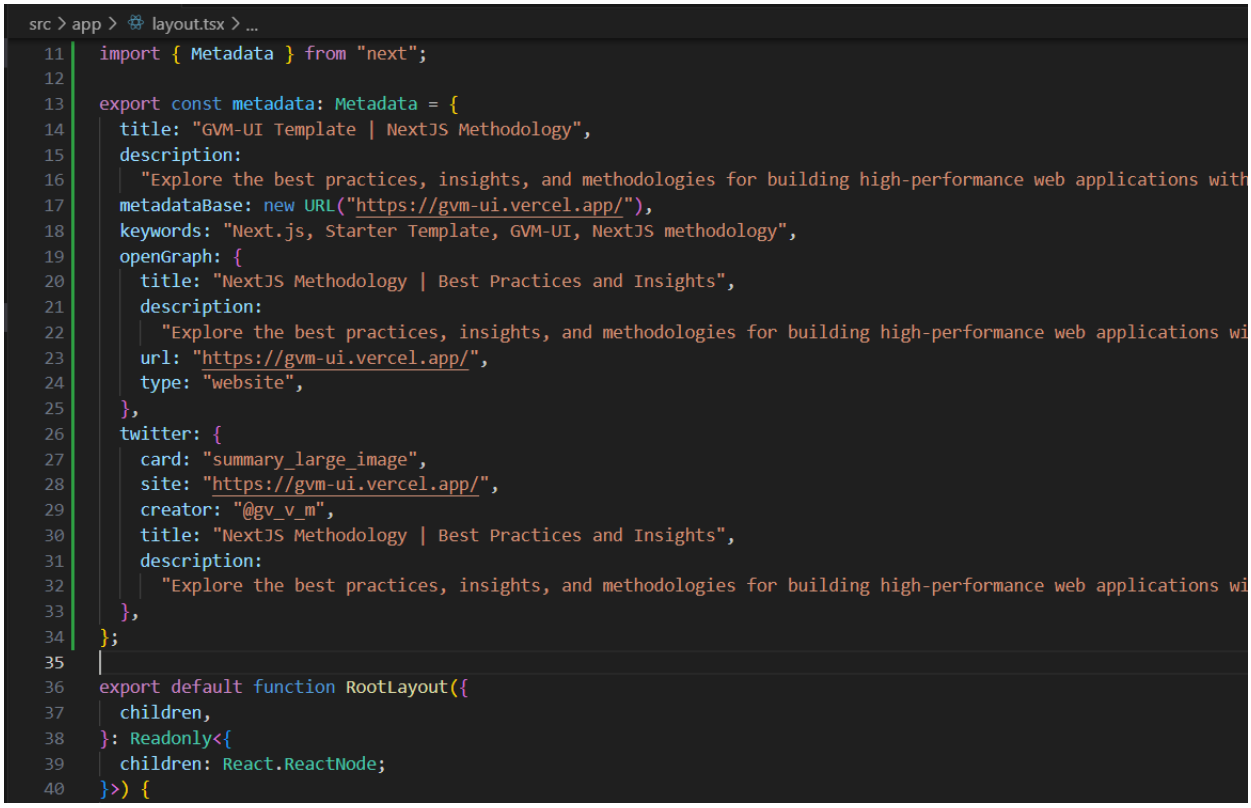


Рис. 9 - Об'єкт Metadata

Додатковим етапом оптимізації є генерація спеціального файлу robots.txt, котрий існує для управління поведінкою пошукових веб-роботів та систем при індексації веб-сайту. Він надає інструкції щодо того, які частини сайту можна сканувати та індексувати, а які слід ігнорувати. Наприклад, це дозволяє обмежити доступ до адміністративних сторінок, конфіденційних даних, тестових або тимчасових файлів та сторінок, які ще не готові до загального доступу. Обмеження індексації непотрібних сторінок допомагає пошуковим системам ефективніше використовувати свої ресурси, зосереджуючись на важливому вмісті сайту. Крім того, якщо у мережі є кілька URL-адрес, які ведуть до одного й того ж вмісту, robots.txt може допомогти запобігти індексації дублікатів, що теж негативно впливає на ранжування сайту в пошукових системах.

Отже, зазвичай у файлі потрібно вказати поле User-agent – перелік пошукових роботів або символ «*» для усіх, Allow – перелік шляхів сайту, які доступні пошуковим системам і Disallow – перелік шляхів, котрі не повинні

індексуватись пошуковими роботами. Можна також додати посилання на файл sitemap.xml з переліком усіх сторінок сайту.

Рис. 10 - Файл robots.txt

Файл sitemap.xml є спеціальним XML-документом, який містить перелік URL-адрес веб-сайту, що допомагає пошуковим системам ефективніше сканувати та індексувати сайт. Він надає важливу інформацію про структуру сайту і дозволяє швидше знаходити нові або оновлені сторінки.

Крім того, sitemap.xml може містити метадані про кожну URL-адресу, такі як дата останнього оновлення (lastmod), частота змін (changefreq) та пріоритетність (priority) сторінки щодо інших URL на сайті. Це допомагає пошуковим системам краще розуміти, як часто слід індексувати ці сторінки.

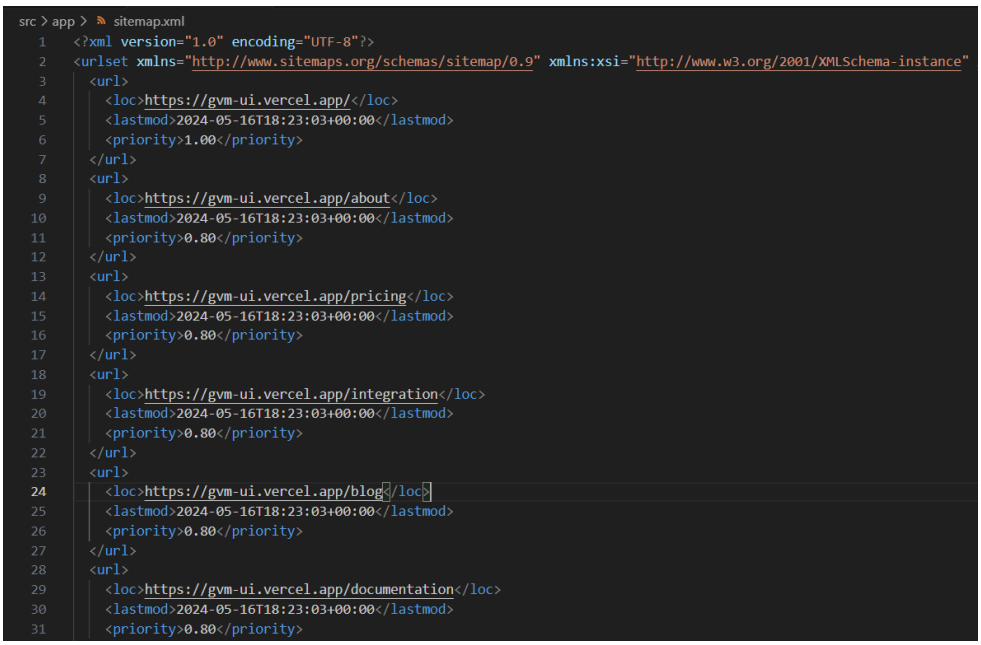


Рис. 11 - Файл sitemap.xml

А що, якщо користувач захоче додати нову сторінку в додатку або, припустимо, видалити блог чи документацію? У такому випадку при кожній зміні треба буде змінювати файл sitemap.xml. Однак можливості NextJS дозволяють зробити цей файл динамічним. Для цього створю новий sitemap.ts, котрий приймає масив маршрутів сайту і будує інформацію для індексації на його основі. Для цього усі сторінки сайту, звичайно, потрібно винести у окремий файл routesUtil.ts. Тут ті маршрути, що мають індексуватись пошуковою системою, додаються у масив:

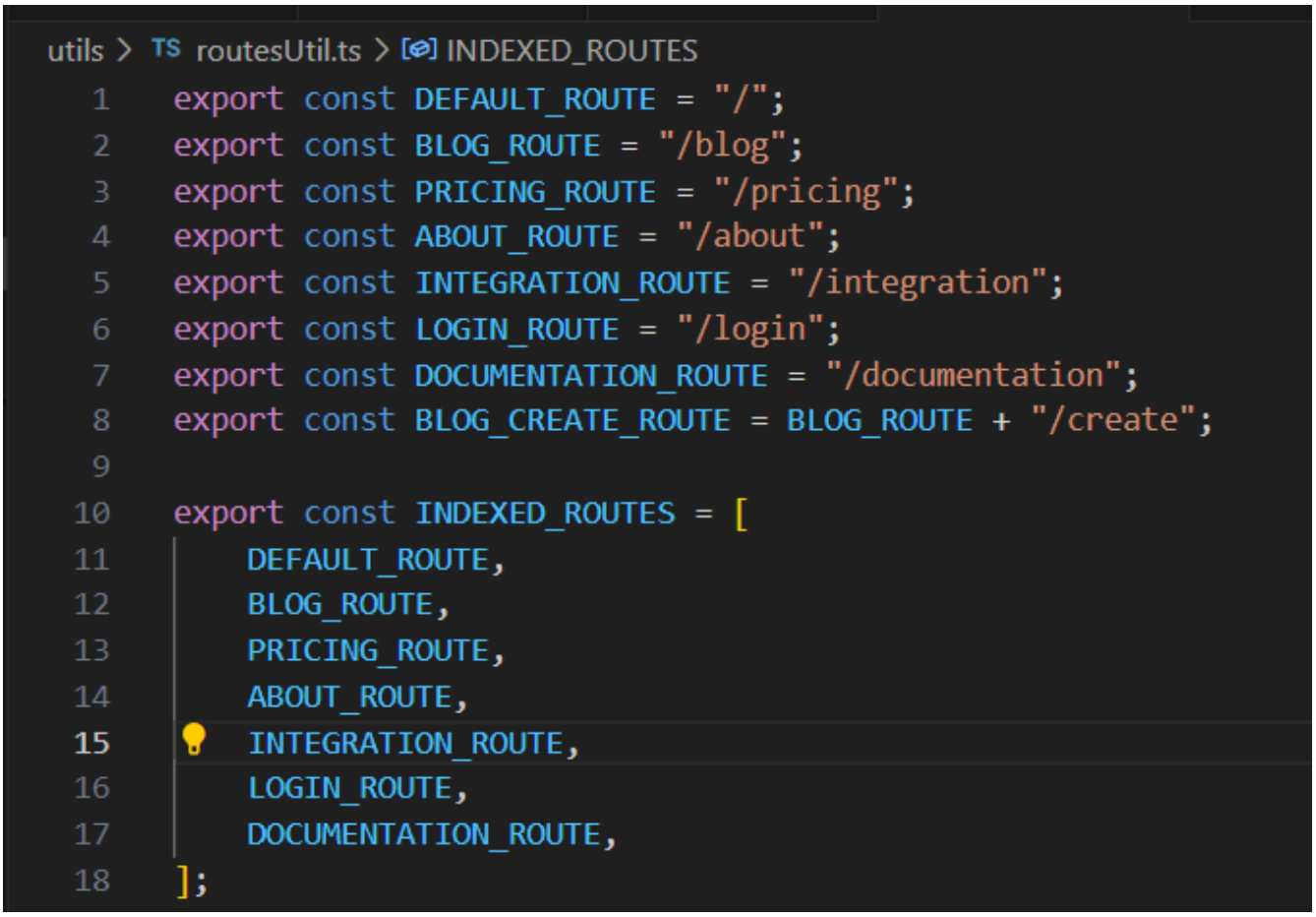


Рис. 12 - Маршрути сайту

Після того масив роутів перебирається у файлі sitemap.ts і надає відповідну пріоритетність сторінкам.

src > app > TS sitemap.ts > sitemap

```ts
import { INDEXED_ROUTES, DEFAULT_ROUTE } from "../../utils/routesUtil";

export default  function sitemap() {
    const baseUrl = process.env.NEXTAUTH_URL;
    const routes = INDEXED_ROUTES.map((route) => ({
        url: baseUrl + route,
        priority: route===DEFAULT_ROUTE ? 1 : 0.8,
    }))
    return routes;
}
```

Рис. 13 - Динамічний sitemap

Це додає певний рівень автоматизації у SEO-просуванні веб-сайту, адже немає необхідності кожен раз при внесенні змін у структуру веб-застосування змінювати sitemap.

4.2. Використання найкращих практик для розробки методології з використанням Next.js

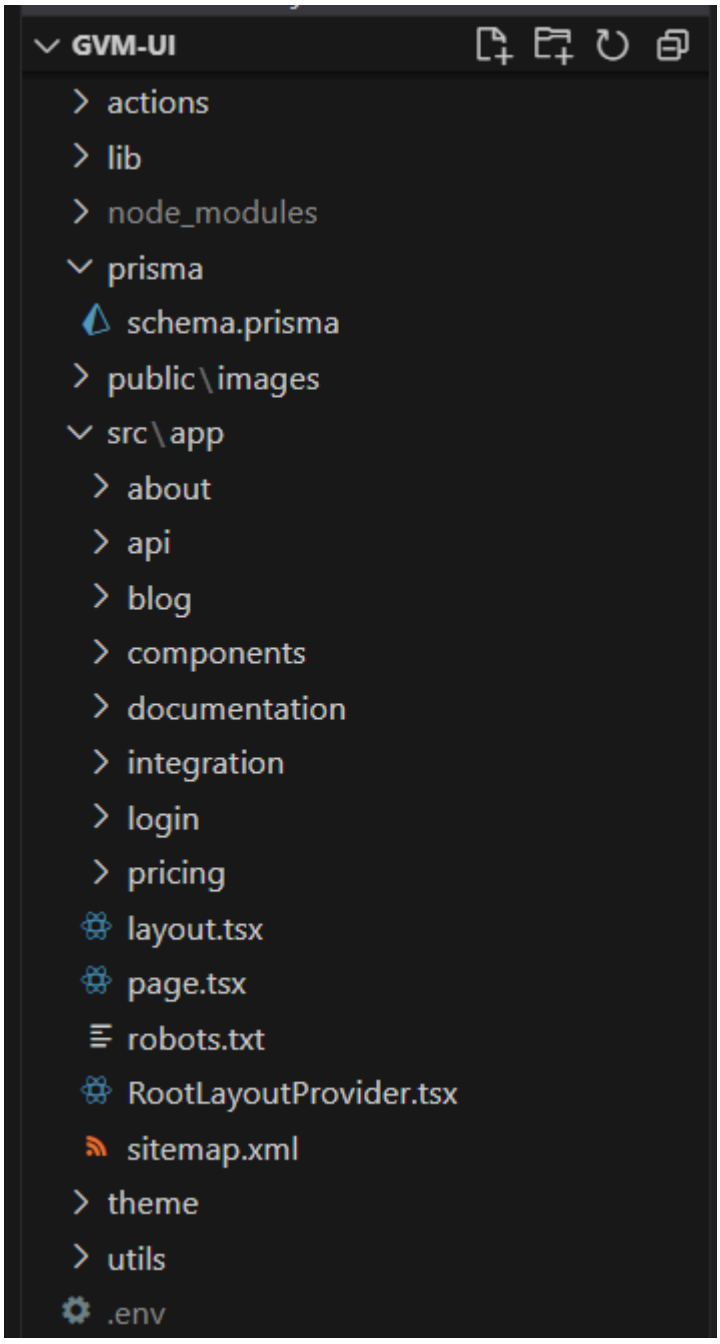


Рис. 14 - Структура проєкту методології

Загалом, окрім згаданих вище нюансів, при написанні додатків на NextJS важливо також дотримуватись наступних вимог. Найперше – структура проєктів повинна бути логічною та послідовною. Зокрема у розробленому шаблоні GVM-UI, кожна директорія та піддиректорія зосереджена на конкретному аспекті або функціональності, що сприяє модульності та легкості підтримки коду. Крім того, присутнє чітке розділення API логіки від UI компонентів (наприклад, api та components директорії) та використання динамічних маршрутів в Next.js (наприклад, [postId]) для створення параметризованих URL, що дозволяє ефективно обробляти контент на основі вхідних даних користувача.

Крім цього, у шаблоні створені різні варіанти готових UI компонентів, котрі легко змінювати під різні потреби користувача. А для роутингу сторінок більше не потрібно інстальовувати сторонні бібліотеки, адже тут використано нову функцію

NextJS – file-system based routing. Тому для створення сторінки з адресою “/about” достатньо всього лише створити у директорії src/app/ нову папку із назвою “about” і додати у неї файл page.tsx.

Отже, усі вищенаведені практики сприяють створенню чистого, організованого, і легкого для розуміння коду, що є особливо важливим для зручності використання готового шаблону NextJS у проєктах, які потребують постійної розробки та масштабування.

РОЗДІЛ 5. АНАЛІЗ РЕЗУЛЬТАТІВ

5.1. Приклади використання розробленої методології.

Найкраще перевірити ефективність та якість розробленої методології можливо лише створивши реальний проєкт з її використанням. Припустимо мені потрібен сайт власної компанії, а часу на його розробку обмаль. У такому випадку є сенс звернутись до методології, котра вже містить реалізовані найбільш ефективним чином сторінки сайту, стилі, панель адміна і так далі.

Оскільки методологія реалізована у вигляді програмного шаблону GVM-UI, то для її використання необхідно клонувати starter template з github на персональний комп’ютер. Після цього запустити у терміналі команду npm install, котра інсталює усі необхідні бібліотеки та пакети. [16]

Далі потрібно створити файл .env у кореневій директорії проєкту, там міститимуться необхідні змінні оточення для забезпечення правильної роботи додатку. Змінні повинні включати налаштування для авторизації через GitHub, конфігурацію для бази даних PostgreSQL, а також різні дані, специфічні для Vercel у разі розгортання проєкту. Кожну змінну необхідно відповідно заповнити власними значеннями:

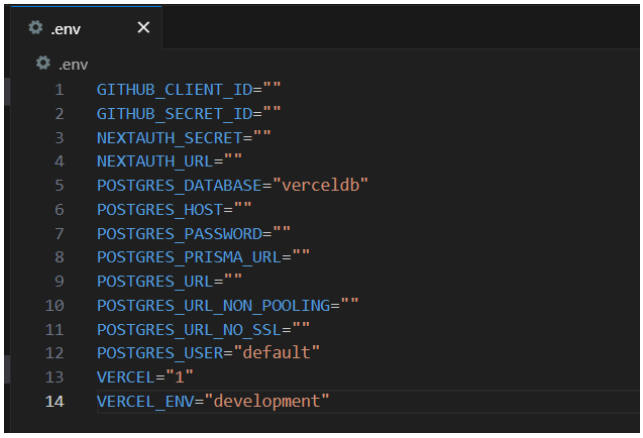


Рис. 15 - Файл із закритими змінними середовища

Після цього можна запустити додаток локально за допомогою команди npm run dev. Ця команда запустить Node.js сервер на самому комп'ютері, а додаток почне працювати на порті 3000.

Завершуючи налаштування, можна перейти у веб-браузері на адресу localhost:3000, щоб перевірити роботу та вигляд сайту. На цій сторінці буде доступний веб-інтерфейс, що включає розроблені сторінки та адмін-панель, яка дозволить керувати динамічним контентом сайту. Отже, за лічені хвилини отримано базовий сайт, який наповнений шаблонною інформацією. З базовим, включеним у шаблон, інтерфейсом можна ознайомитись у Додатку А. [17]

5.1.1. Шкільний сайт.

Безперечно, наступним важливим етапом є кастомізація цього шаблону з урахуванням потреб відповідної компанії/організації. Припустимо, виникла потреба змінити Hero секцію веб-застосунку, щоб вона підходила для сайту середньої загальноосвітньої школи. Щоб відредагувати головну сторінку сайту, треба знайти файл page.tsx у src/ директорії і почати редагування його компонентів. Єдине, що потрібно для цього зробити – змінити заголовки і наповнення у файлі Hero.tsx на відповідне. Немає необхідності самостійно налаштовувати адаптацію під різні розміри екранів чи додавати стилі тексту, усе вже враховано у макеті.

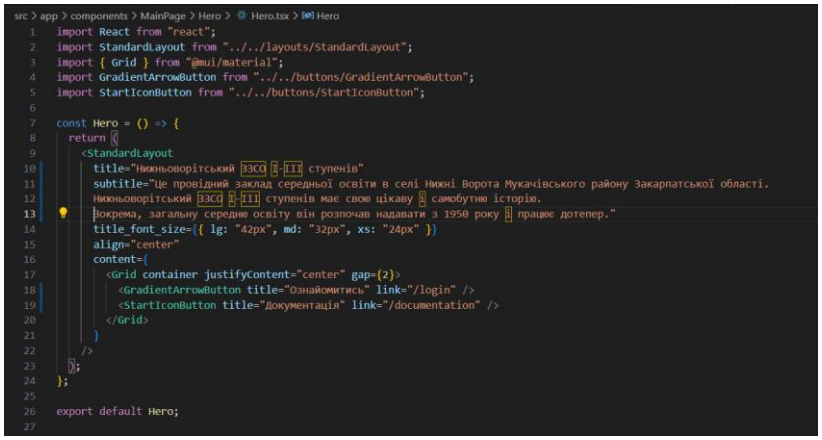


Рис. 16 - Код компоненти головної сторінки

Для зміни ж наповнення навігаційної панелі можна навіть не лізти у код, адже редагування сталої для усього сайту інформації, такої як навігаційна панель, підвал й інше, передбачене в папці utils. Тому знайшовши файл navbarUtils, достатньо наповнити список NAVBAR_OPTIONS потрібною інформацією. У додатку Б можна ознайомитись із інтерфейсом сайту, що вийшов у результаті внесених змін.

5.1.2. E-commerce платформа

Оскільки розроблена методологія є універсальною, вона може бути використана також для створення основи інтернет-магазину, що теж вагомо полегшує процес втілення такого проєкту розробникам. Для створення цього продукту, необхідно відредагувати наповнення платформи і дизайн (за бажанням) під задане технічне завдання (у цьому випадку умовний інтернет-магазин) і окрім цього внести зміни у базу даних. Для цього у кореневій папці проєкту потрібно знайти файл prisma/schema.prisma, у якому задані моделі бази даних за замовчуванням і змінити ці моделі на потрібні для збереження товарів інтернет-магазину. Після команди git push база даних автоматично модифікується відповідно до внесених змін.

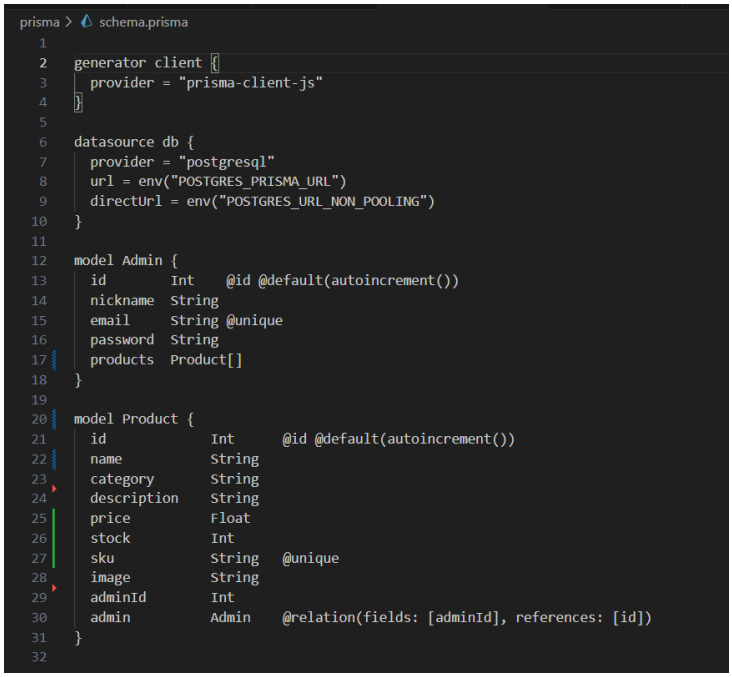


Рис. 17 - Модифіковані моделі БД

Щодо UI сторінки з товарами, то варто лише замінити наявні картки із постами блогу на готові картки ProductCard, а запит до бази даних – з post на product. Очевидно, що і властивості в картки теж мають задаватись відповідно до полів з бази даних. Приклад готової сторінки з товарами можна побачити у додатку В.

Використання такої методології дозволяє значно спростити процес розробки, оскільки більшість стандартних компонентів та функціональності вже реалізовано у вигляді шаблону, котрий створений з дотриманням певних вимог для отримання максимальної ефективності в роботі веб-застосування. А влаштована функція автентифікації й налаштовані теми додатку дозволяють зосередитись на унікальних аспектах власного проекту. Саме тому завдяки універсальності шаблону, користувач з будь-якими потребами може знайти цю розробку корисною.

5.2. Аналіз результатів та виправлення можливих проблем.

Отже, практичним результатом розробленої методології є початковий шаблон, який може бути корисним та адаптовуватись під проєкти різних потреб і масштабів, забезпечуючи влаштовані стилі, теми, автентифікацію, панель адміна, тощо. Основною вимогою до шаблону є висока ефективність і швидкість роботи сайту, а також високий показник SEO. Завдяки глибокому аналізу особливостей фреймворку NextJS та формуванню систематичного підходу до розробки на цій технології, очікується висока ефективність створеного проєкту.

Для того, щоб перевірити усі вище згадані показники, можна використати веб-плагін – Lighthouse. Це автоматизований інструмент для покращення веб-сайтів, розроблений компанією Google. Основна суть плагіну – перевірити задану веб-сторінку за чотирма критеріями. Зокрема, ефективність, доступність, оптимальність методів та оптимізацію пошукових систем (SEO) сайту. [14]

Щоб провести аудит розробленої методології, потрібно відкрити наявний шаблон (обов’язково на продакшині, не локально) і в інструментах розробника запустити потрібний плагін.

Як видно на скріншоті в Додатку Г, результати тестування наступні: ефективність сайту – 97, доступність – 90, оптимальність методів – 100 та SEO – 90. Це вважається високим результатом, тому можна вважати, що розроблена методологія фреймворку NextJS справді дає оптимальний підхід для впровадження майбутніх реальних проєктів.

Висновки

У цій роботі було проведено дослідження та розроблено методологію використання фреймворку Next.js для створення повноцінних веб-додатків. Під час аналізу сучасних frontend-фреймворків було виявлено, що Next.js вирізняється своєю потужною функціональністю, яка включає в себе Server-Side Rendering (SSR), Static Site Generation (SSG) та інші переваги.

У розділі про аналіз існуючих рішень було розглянуто проекти, які використовують Next.js, що підтверджує його популярність та придатність для створення різних типів додатків. А під час порівняння Next.js з іншими веб-фреймворками було визначено, що він має свої сильні та слабкі сторони, які варто враховувати при виборі для конкретного проекту.

Далі було розглянуто популярні підходи до розробки за допомогою Next.js, а також визначено оптимальні практики та стандарти коду для ефективної розробки проектів на цьому фреймворку.

На основі проведеного дослідження та аналізу була розроблена методологія використання Next.js, яка надає набір рекомендацій для отримання найбільш ефективних веб-застосувань та автоматизований шаблон, який створений з дотриманням усіх вимог та забезпеченням базових потреб стандартного проєкту: автентифікації, панелі адміна, налаштовуваних стилів та тем.

Отже, використання фреймворку Next.js з врахуванням розробленої методології може допомогти забезпечити успішну та ефективну розробку веб-додатків з високим рівнем якості та продуктивності.

Практичне застосування

Практичне застосування розробленої методології використання фреймворку Next.js може мати різноманітні напрямки в реальних проектах. Ось декілька прикладів, які варто розглянути:

Розробка веб-додатків різного масштабу: Методологія може бути успішно використана для створення веб-додатків будь-якого рівня складності, від невеликих сайтів-візитівок до великих корпоративних рішень.

Створення електронних комерційних платформ: Next.js може бути використаний для розробки онлайн-магазинів та інших електронних комерційних платформ. Використання методології допоможе забезпечити ефективну розробку та підтримку таких проектів.

Реалізація блогів та інформаційних сайтів: Для створення блогів, новинних сайтів та інших інформаційних ресурсів методологія також може бути корисною, оскільки вона дозволяє легко керувати контентом та оновлювати його.

Розробка веб-додатків зі складною взаємодією та аналітикою: Next.js може бути використаний для створення веб-додатків зі складною взаємодією, наприклад, аналітичних платформ, панелей керування тощо.

Незалежно від конкретного використання, розробка за допомогою розробленої методології може допомогти забезпечити ефективну, швидку та якісну розробку веб-додатків, що відповідають потребам та вимогам вашого проекту.

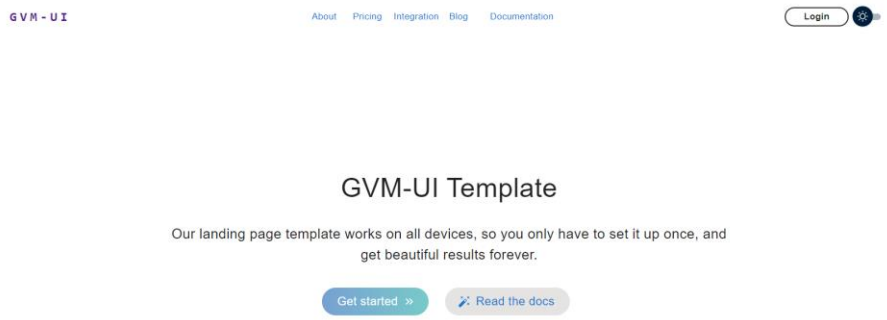
Список використаних джерел

1. Is Next.js frontend or backend framework? – URL: https://codedamn.com/news/nextjs/is-next-js-a-frontend-or-backend-framework#the_fullstack_nature_of_nextjs
2. Топ front-end фреймворків 2024. [Електронний ресурс] – URL: https://refine.dev/blog/best-front-end-frameworks-in-2024/
3. Що таке Next.js? [Електронний ресурс] – URL: https://www.sanity.io/glossary/next-js
4. https://www.twitch.tv/
5. https://open.spotify.com/
6. https://www.nike.com/hu/en/?_gl=1*psa1cm*_up*MQ..&gclid=Cj0KCQjw3ZayBhDRARIsAPWzx8o3xEBnskDhEi3KCAE_zbvVj-2pXOztH9Bxbn46yLjjEgGwMDO1-YUaAqrLEALw_wcB&gclsrc=aw.ds
7. Зображення file-based routing. [Електронний ресурс] – URL: https://nextjs.org/_next/image?url=%2Fdocs%2Flight%2Froute-segments-to-path-segments.png&w=3840&q=75
8. https://solana.com/
9. Next.js VS інші фреймворки. [Електронний ресурс] – URL: https://medium.com/@asiandigitalhub/next-js-vs-other-front-end-frameworks-55d7ef1d51f0
10. 5 причин використовувати Next.js для Front-End розробки. [Електронний ресурс] – URL: https://www.intuz.com/blog/5-reasons-why-you-should-use-next.js-for-your-front-end-development
11. Переваги та недоліки Next.js. [Електронний ресурс] – URL: https://pagepro.co/blog/pros-and-cons-of-nextjs/
12. Документація Material UI. [Електронний ресурс] – URL: https://mui.com/material-ui/

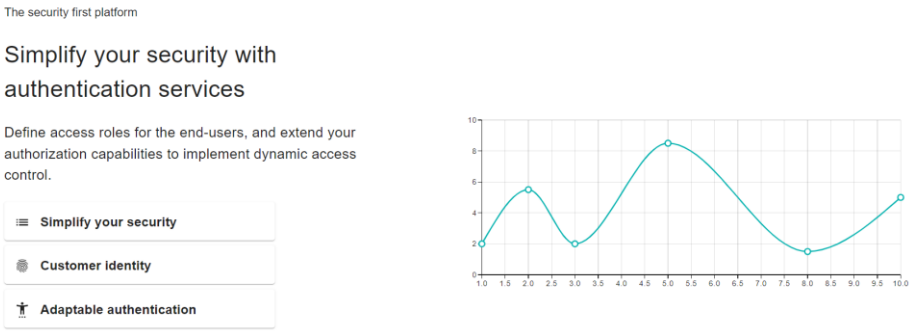
13. Що таке Prisma ORM? [Електронний ресурс] – URL: https://www.prisma.io/docs/orm/overview/introduction/what-is-prisma
14. NextAuth documentation. [Електронний ресурс] – URL: https://next-auth.js.org/getting-started/introduction
15. Плагін Lighthouse – документація. [Електронний ресурс] – URL: https://developer.chrome.com/docs/lighthouse/overview#:~:text=Lighthouse%20is%20an%20open%2Dsource,apps%2C%20SEO%2C%20and%20more
16. Посилання на розроблений шаблон. [Електронний ресурс] – URL: https://github.com/gviktoriia/gvm-ui
17. Посилання на демо-версію шаблону. [Електронний ресурс] – URL: https://gvm-ui.vercel.app/

Додатки

Додаток А.1

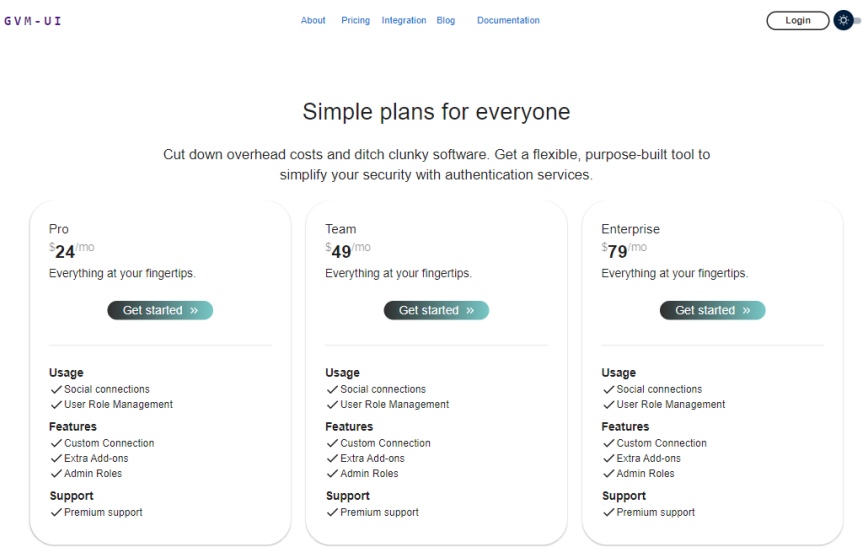


Додаток А.2

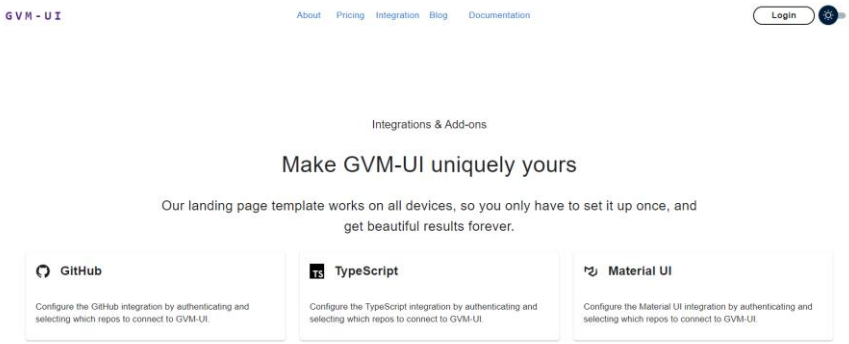


Додаток А.3

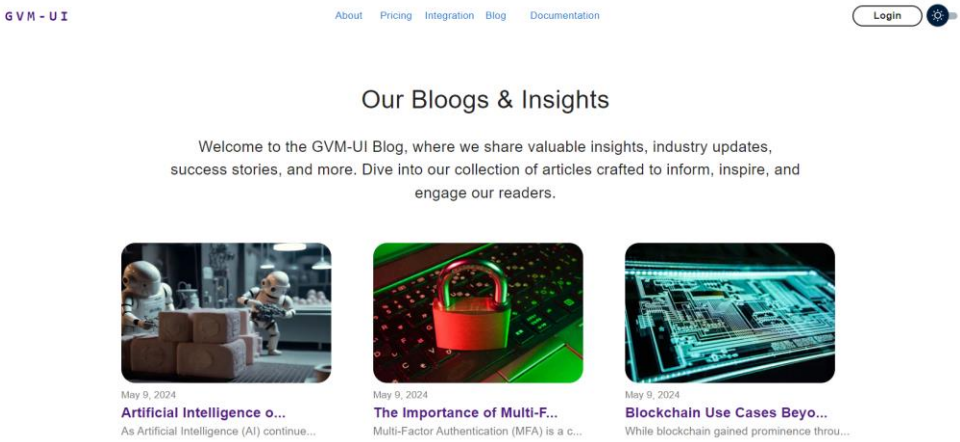
Додаток А.4



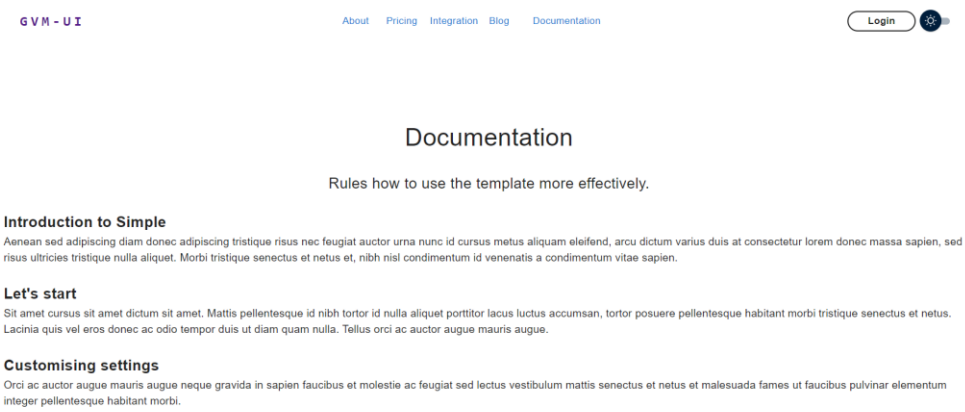
Додаток А.5



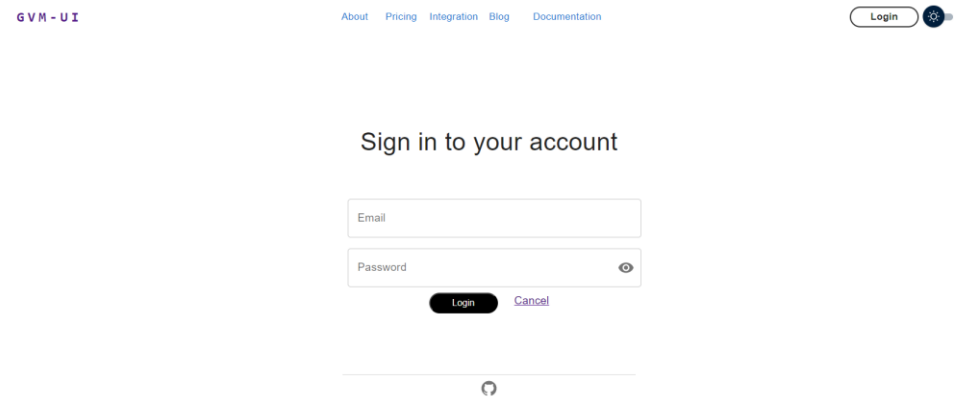
Додаток А.6



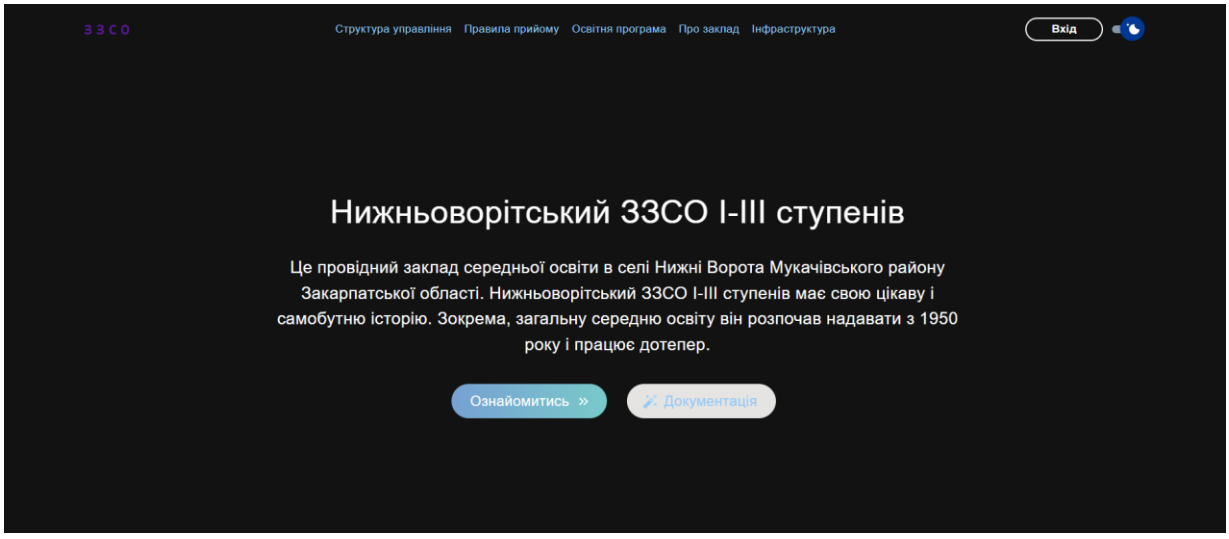
Додаток А.7



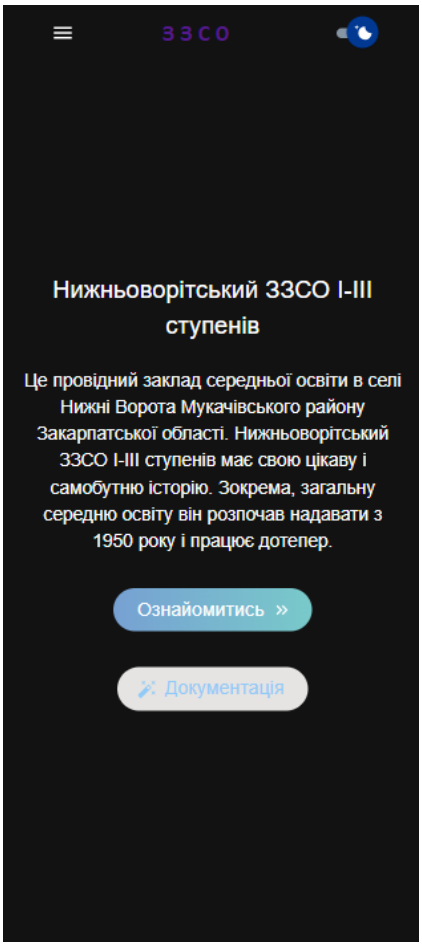
Додаток А.8



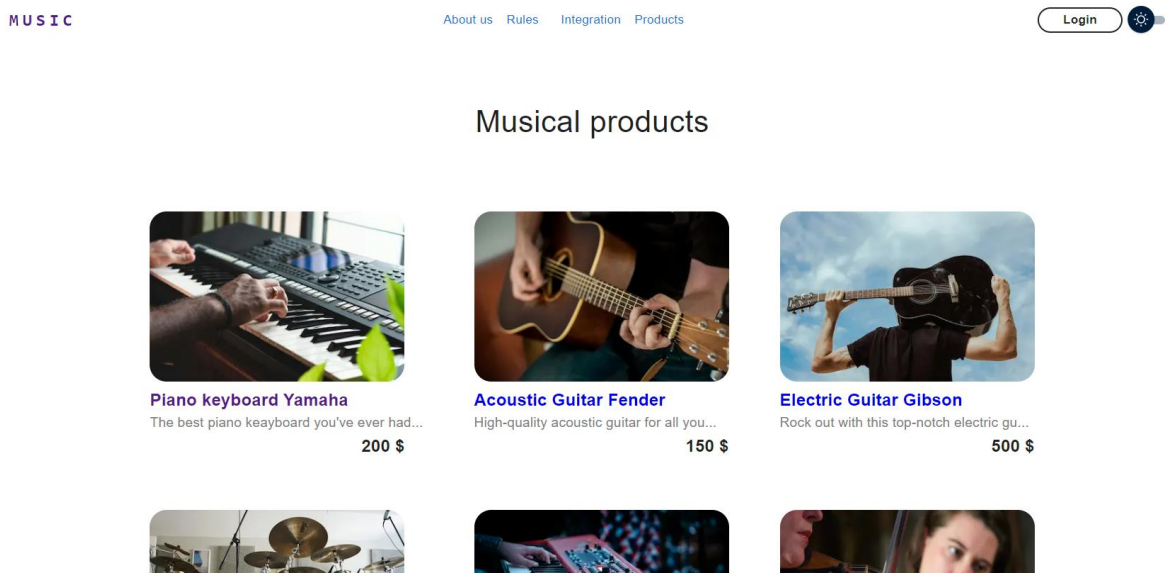
Додаток Б.1



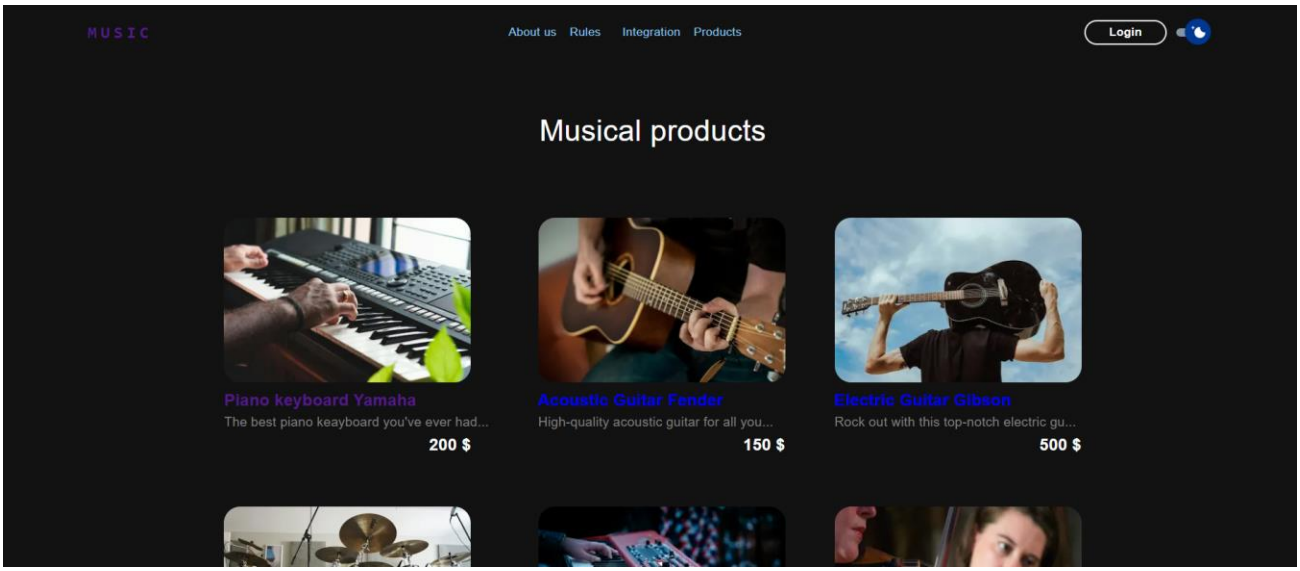
Додаток Б.2



Додаток В.1



Додаток В.2



Додаток Г