МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра електроніки і комп'ютерної техніки

«До захисту допущено»
Завідувач кафедри
_____ Анатолій ОПАНАСЮК
_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавра

зі спеціальності 172 «Телекомунікації та радіотехніка»,
освітньо-професійної програми «Мережеві та інтернет технології»
На тему:
Проектування системи керування веб-контентом на базі Firebase

Здобувача групи ТК-91 Загорулька Владислава Сергійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Владислав ЗАГОРУЛЬКО

Керівник, старший викладач, кандидат технічних наук, доцент,
Олексій ГОРЯЧЕВ _____

_____ Олексій ГОРЯЧЕВ

Суми – 2023

АНОТАЦІЯ

Кваліфікаційна робота бакалавра містить проектування системи керування веб-контентом на базі Firebase з використанням фреймворку Nuxt.js.

Метою кваліфікаційної роботи є дослідження основних аспектів створення проекту на базі Firebase, включаючи структуру бази даних, архітектуру проекту та функціональні можливості адмін панелі.

У наш час веб-сайти стали ключовими інструментами для комунікації, розповсюдження інформації та надання послуг. Для ефективного управління веб-сайтами потрібна потужна система керування веб-контентом, яка дозволяє легко додавати, редагувати та видаляти контент без потреби великих знань програмування. Дана робота є важливим кроком у напрямку розробки інструментів для керування веб-контентом та допоможе покращити якість та доступність веб-сайтів у цифрову епоху.

Кваліфікаційна робота містить 41 сторінку, 21 рисунок, 1 таблицю, 8 джерел літератури.

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА ЕЛЕКТРОНІКИ І КОМП'ЮТЕРНОЇ ТЕХНІКИ
Факультет ЕліТ Спеціальність 6.172 "Телекомунікації та радіотехніка"

ЗАТВЕРДЖУЮ
Зав. кафедрою Опанасюк О.А.
_____
«___» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА
Загорулька Владислава Сергійовича

1. Тема роботи: «**Проектування системи керування веб-контентом на базі Firebase**»

Затверджена наказом по університету від "31" березня 2023 р. № 0316 - VI

2. Термін здачі студентом закінченої роботи: 05.06.2023 р;

3. Вихідні дані до роботи:

4. Зміст розрахунково-пояснювальної записки (перелік питань, що належить розробити):

1. Огляд літератури та поставлення задачі роботи; 2. Детальний огляд платворми Firebase; 3. Детальний огляд фронтенд фреймворків; 4. Створення веб-сайту на базі Firebase з використанням Nuxt; 5. Тестування та оптимізація веб-сайту.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

_____

_____

6. Дата видачі завдання: 31.03.2023 р;

Керівник _____ _____Горячев О. Є._____
(підпис) (прізвище, ім'я, по-батькові)

Завдання прийняв до виконання _____ _____Загорулько В. С._____
(підпис) (прізвище, ім'я, по-батькові)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів курсового проекту	Термін виконання етапів проекту	Примітка
1	Огляд літератури згідно з темою дослідження	20.04.2023	
2	Узагальнення інформації та опис Firebase і фронтенд фреймворків	27.04.2023	
3	Створення веб-сайту на базі Firebase з використанням Nuxt	25.05.2023	
4	Тестування та оптимізація веб-сайту	27.05.2023	
5	Узагальнення результатів та оформлення остаточної роботи	31.05.2023	
6	Підготовка кваліфікаційної роботи до захисту	05.06.2023	

Студент Загорулько В. С.

Керівник Горячев О. Є.

ЗМІСТ

ВСТУП

У сучасному світі інтернет є невід'ємною частиною нашого повсякденного життя, а веб-сайти стали ключовими інструментами для комунікації, розповсюдження інформації та надання послуг. Для ефективного управління веб-сайтами потрібна потужна система керування веб-контентом, яка дозволяє легко додавати, редагувати та видаляти контент без потреби великих знань програмування.

Актуальність цієї теми полягає у тому, що багато бізнесів, організацій та індивідуальних користувачів шукають зручні та ефективні способи керування своїми веб-сайтами. Firebase, який є платформою розробки мобільних та веб-додатків, надає широкі можливості для створення потужних систем керування веб-контентом.

Метою цієї роботи є розробка та реалізація системи керування веб-контентом на базі Firebase, яка буде забезпечувати зручність та ефективність управління веб-сайтом. Для досягнення цієї мети будуть виконані наступні завдання:

- Вивчення актуальних тенденцій у веб-розробці та керуванні веб-контентом.
- Аналіз існуючих систем керування веб-контентом та їх переваг і недоліків.
- Визначення можливостей та переваг Firebase як платформи для реалізації системи.
- Проектування архітектури системи та структури бази даних.
- Розробка користувальницького інтерфейсу для керування веб-контентом.
- Імплементація функціональності для додавання, редагування та видалення веб-контенту.
- Тестування та оцінка ефективності розробленої системи.

Результати цієї роботи будуть корисними для розробників веб-сайтів, бізнесів та організацій, які шукають зручне та ефективне рішення для

керування своїм веб-контентом. Розроблена система на базі Firebase надасть їм можливість швидко та просто оновлювати та модифікувати контент, покращуючи їхню присутність в Інтернеті та забезпечуючи кращу взаємодію з користувачами.

Дослідження включатиме аналіз літературних джерел, порівняльний аналіз існуючих систем, проектування архітектури системи та реалізацію функціональності. Результати цього дослідження сприятимуть подальшому розвитку та вдосконаленню систем керування веб-контентом на базі Firebase та сприяють збільшенню ефективності веб-розробки. Таким чином, ця робота є важливим кроком у напрямку розробки інструментів для керування веб-контентом та допоможе покращити якість та доступність веб-сайтів у цифрову епоху.

1. ОГЛЯД ЛІТЕРАТУРИ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Сучасні підходи та методи керування веб-контентом

Керування веб-контентом є ключовим аспектом розробки та управління веб-сайтами. Воно включає в себе процес створення, редагування, оновлення та управління веб-контентом, який включає тексти, зображення, відео, аудіо та інші елементи, що розміщуються на веб-сторінках.

Веб-контент відіграє важливу роль у залученні та утриманні відвідувачів на веб-сайтах. Ефективне керування веб-контентом дозволяє забезпечити свіжість, релевантність та якість веб-сайту, покращує користувацький досвід і сприяє досягненню мети веб-проекту. Одним з основних понять у керуванні веб-контентом є контентний менеджмент. Контентний менеджмент охоплює планування, створення, редагування, публікацію та архівування веб-контенту. Він включає у себе організацію та структурування контенту, його тегування та категоризацію, а також керування версіями контенту.

У сучасній сфері керування веб-контентом існує ряд нових підходів та методів, які спрямовані на поліпшення ефективності та зручності управління веб-контентом. Ці підходи і методи враховують зростаючі потреби веб-розробників та адміністраторів, а також нові тренди у веб-розробці.

Найпростішим підходом до створення, редагування та публікації веб-контенту без необхідності глибоких знань програмування є різні системи керування веб-контентом (CMS), такі як WordPress, Drupal, Joomla та інші. Вони надають інтерфейс для зручного керування веб-сайтом, включаючи можливість додавати нові сторінки, редагувати існуючі, керувати зображеннями та медіафайлами, налаштовувати права доступу користувачів тощо.

Headless CMS та хмарні платформи є двома різними підходами до керування веб-контентом, що вимагають знання програмування та надають свої унікальні переваги та можливості. Headless CMS (Content Management System) – тип системи керування вмістом, яка розділяє контент від його

представлення, надаючи API для зберігання, управління та постачання вмісту, незалежно від фронтенду, на якому він буде відображений. У відміну від традиційних CMS, які мають вбудовану функціональність для відображення вмісту на веб-сторінках, Headless CMS фокусується на наданні структурованого та доступного контенту через API. Хмарні платформи, у свою чергу, надають зручне та масштабоване середовище для розробки та управління веб-контентом. Вони пропонують інфраструктуру хмарних обчислень, бази даних в реальному часі, зберігання файлів, аутентифікацію користувачів, аналітику та інші функціональні можливості, які спрощують розробку та підтримку додатків.

У таблиці 1 наведено порівняльний аналіз цих двох підходів.

Таблиця 1 – Порівняння хмарних платформ та Headless CMS

Фактор	Headless CMS	Хмарні платформи
Архітектура та розділення функцій	Надають API для отримання та редагування контенту, тоді як відображення цього контенту на веб-сторінках або мобільних додатках залежить від фронтенд розробників.	Надають комплексний набір готових інструментів та сервісів для розробки та управління веб-контентом.
Гнучкість	Надають розробникам більшу гнучкість та контроль над тим, як вони відображають контент. Однак, при розробці власного фронтенду потрібно більше зусиль та часу.	Надають готові інструменти та сервіси, які спрощують розробку веб-контенту, що забезпечує швидкість розробки та зменшує зусилля, але може обмежувати гнучкість та можливості налаштування.
Управління інфраструктурою	Не надають готової інфраструктури для розміщення веб-сайту чи додатку. Розробники самостійно відповідають за інфраструктуру, таку як хостинг, масштабування та безпеку.	Надають готову інфраструктуру, що включає хостинг, зберігання даних, безпеку та інші сервіси. Це зменшує витрати та зусилля, пов'язані з управлінням інфраструктурою.

Продовження таблиці 1

1	2	3
Розширення та масштабованість	Може бути більш гнучким для розширення та масштабування, оскільки розробники можуть використовувати будь-які фреймворки та інструменти для розширення функціональності.	Надають готові інструменти для розширення та масштабування системи. Вони можуть легко обробляти зростаюче навантаження та забезпечувати високу доступність.

1.2. Вимоги та очікування щодо керування веб-контентом

Вимоги та очікування користувачів щодо керування веб-контентом можуть бути різними в залежності від їхніх потреб та контексту використання. Однак, деякі загальні вимоги та очікування можуть бути такими:

1. Легкість використання: Користувачі очікують, щоб система керування веб-контентом була легкою у використанні, навіть для тих, хто не має технічних навичок. Інтерфейс користувача повинен бути зрозумілим, інтуїтивно зрозумілим і добре організованим, щоб дозволити користувачам легко створювати, редагувати та публікувати контент на веб-сайті.

2. Гнучкість та налаштування: Користувачі очікують, щоб система керування веб-контентом надавала їм можливість гнучко налаштовувати вигляд, структуру та функціональність свого веб-сайту. Вони мають можливість визначати шаблони, дизайн, меню, розташування вмісту та інші аспекти, щоб відповідати їхнім унікальним потребам та бренду.

3. Ролевий доступ та права користувачів: Користувачі очікують, щоб система керування веб-контентом надавала можливість управління рівнями доступу та правами користувачів. Вони можуть мати потребу встановлювати різні рівні доступу для адміністраторів, редакторів, авторів та інших користувачів залежно від їхніх ролей та обов'язків.

4. Масштабованість: Користувачі можуть мати вимоги щодо масштабованості системи керування веб-контентом, особливо якщо вони планують розширювати свій веб-проект у майбутньому. Вони можуть очікувати, щоб система була здатною обробляти великий обсяг вмісту, підтримувала розширення та інтеграцію з іншими сервісами та мала високу продуктивність.

5. Безпека та захист: Користувачі мають очікування щодо безпеки свого веб-контенту. Вони очікують, що система керування веб-контентом буде забезпечувати захист від несанкціонованого доступу, зберігання даних у безпечному форматі та можливість резервного копіювання та відновлення даних.

6. Підтримка та документація: Користувачі очікують наявності якісної технічної підтримки та документації з боку розробників системи керування веб-контентом. Це допомагає їм вирішувати технічні питання, отримувати відповіді на запитання та швидко реагувати на можливі проблеми.

Загалом, користувачі очікують, що система керування веб-контентом буде простою у використанні, гнучкою, масштабованою, безпечною та забезпечуватиме їхні потреби щодо створення та управління вмістом на веб-сайті.

1.3. Огляд об’єкта проектування

Завданням кваліфікаційного проекту є розробка веб-сайту з платформою для керування веб-контентом, забезпечуючи зручність управління та надійність зберігання інформації.

Дане завдання вимагає наявності клієнтської частини веб-сайту, а також інтерфейсу адміністратора, який буде дозволяти користувачеві з легкістю керувати контентом сайту. Використання Firebase, потужної хмарної платформи розробки, забезпечить швидкий та безперебійний обмін даними

між веб-клієнтом та сервером, а також забезпечить потужні можливості зберігання та синхронізації даних.

Крім того, система адміністрування повинна бути забезпечена механізмами аутентифікації та авторизації, щоб забезпечити безпеку та обмежені права доступу для різних користувачів. Це дозволить контролювати доступ до важливих функцій та забезпечить безпеку даних. Розробка даного веб-сайту повинна враховувати оптимізацію для пошукових систем, що дозволить покращити видимість сайту у результатах пошуку та привернути більше цільового трафіку. Для цього можуть використовуватися правильна структура URL-адрес, метатеги, розділи для ключового контенту та інші SEO-оптимізаційні практики.

Загалом, розробка цього веб-сайту з інтерфейсом адміністратора, використовуючи Firebase, вимагатиме глибоких знань веб-технологій, баз даних, а також уміння враховувати потреби адміністраторів та користувачів. Результатом буде потужна та функціональна платформа, яка забезпечить ефективне керування веб-контентом та надійність його зберігання.

2. ОГЛЯД ПЛАТФОРМИ FIREBASE

2.1. Основні сервіси та інструменти Firebase

Firebase надає широкий набір сервісів та інструментів, які допомагають розробникам створювати та керувати веб-контентом. Основні сервіси та інструменти Firebase включають наступне:

Firebase Authentication: надає безпечну та просту у використанні систему аутентифікації користувачів для веб-додатків. Він підтримує різні методи аутентифікації, включаючи електронну пошту та пароль, облікові записи Google, Facebook, Twitter та інші. Цей сервіс дозволяє розробникам швидко додавати функціонал авторизації до своїх додатків безпосередньо з Firebase.

Firebase Realtime Database: гнучка база даних в реальному часі, яка дозволяє синхронізувати та обмінюватися даними між користувачами в режимі реального часу. Вона зберігає дані у вигляді дерева JSON та автоматично оновлює їх на всіх підключених пристроях. Це робить Firebase Realtime Database ідеальним рішенням для веб-додатків, де необхідно отримувати оновлення даних відразу, наприклад, чати, спільна робота над документами тощо.

Firebase Cloud Firestore: гнучка, швидка та масштабована база даних, яка дозволяє зберігати та синхронізувати дані для веб-додатків. Вона працює на основі документів та колекцій, що дозволяє легко організувати та оптимізувати дані. Firestore підтримує режим офлайн доступу, дозволяючи користувачам працювати з додатком, навіть коли вони не мають доступу до Інтернету.

Firebase Storage: надає можливість зберігати та керувати файлами веб-додатка в хмарі. Він простий у використанні та надійний, дозволяючи завантажувати, зберігати та отримувати доступ до файлів з будь-якого пристрою. Firebase Storage забезпечує автоматичне масштабування та безпеку файлів, дозволяючи зосередитися на розробці вмісту веб-додатку.

Firebase Hosting: надає швидке та безкоштовне хостинг веб-додатків, що дозволяє швидко розгорнути та публікувати ваш веб-контент. Ви можете використовувати власний домен або отримати безкоштовний піддомен firebaseapp.com. Firebase Hosting автоматично надає SSL-сертифікати для забезпечення безпеки передачі даних.

Firebase Cloud Functions: дозволяє розробникам писати та запускати функції на серверному боці безпосередньо з Firebase. Це дає можливість реагувати на події, які створюються в додатку, такі як створення нового користувача, завантаження файлу тощо. Функції можуть використовуватися для автоматизації завдань, обробки даних, взаємодії з іншими сервісами та багатьох інших сценаріїв.

Firebase надає розробникам потужні та гнучкі інструменти для керування веб-контентом, спрощуючи розробку, розгортання та підтримку веб-додатків. Його широкий набір сервісів дозволяє ефективно керувати аутентифікацією користувачів, зберігати та синхронізувати дані, зберігати файли, розгортати додатки та виконувати серверні функції на основі подій.

2.2. Опис можливостей Firebase

Firebase надає широкі можливості зберігання, синхронізації даних та керування користувачами, що робить його потужним інструментом для розробки веб-додатків.

Зберігання даних є однією з основних функцій Firebase, яка надає розробникам потужні інструменти для зберігання та керування даними в їх веб-додатках. Firebase пропонує дві основні бази даних – Realtime Database та Firestore, кожна з яких має свої особливості та переваги.

Firebase Realtime Database: Firebase Realtime Database є NoSQL базою даних, яка дозволяє зберігати дані у вигляді дерева JSON. Ця база даних працює в режимі реального часу, що означає, що зміни, внесені до бази даних одним користувачем, негайно синхронізуються з усіма підключеними

клієнтами. Це забезпечує миттєву оновлення даних на всіх пристроях і створює реактивний досвід для користувачів.

Realtime Database дозволяє розробникам зберігати дані в структурованому форматі, де вузлами можуть бути об'єкти JSON з ключами та значеннями. За допомогою простих операцій читання та запису, розробники можуть отримувати доступ до даних та вносити зміни в реальному часі. Firebase також надає можливість фільтрування та сортування даних, що полегшує отримання необхідної інформації з бази даних.

Firebase Firestore: Firebase Firestore є багатофункціональною базою даних, яка пропонує більш розширені можливості порівняно з Realtime Database. Firestore використовує колекції, документи та колекції в документах для зберігання та організації даних. Ця база даних підтримує багато запитів та операцій, які дозволяють розробникам створювати складні структури даних та виконувати потужні запити до бази даних.

Firestore надає можливість створювати запити, які фільтрують дані за певними умовами, сортувати дані за різними полями, а також об'єднувати запити для отримання більш складної інформації. Крім того, Firestore підтримує можливість підписки на зміни даних, що дозволяє реагувати на зміни бази даних в режимі реального часу.

Firebase також надає можливості для резервного копіювання та відновлення даних, що забезпечує захист інформації в разі випадкового видалення або втрати даних. Розробники можуть легко налаштувати автоматичне резервне копіювання даних та використовувати його для відновлення даних в разі потреби.

Синхронізація даних є не менш важливою функцією. Firebase забезпечує потужні засоби для синхронізації даних між різними пристроями та платформами. Завдяки механізму реального часу, зміни, внесені до бази даних, негайно синхронізуються між всіма підключеними клієнтами. Це дозволяє користувачам бачити оновлені дані без необхідності оновлення сторінки або виконання запитів до сервера. Крім того, Firebase також підтримує режим

роботи без підключення до Інтернету, що дозволяє працювати з даними навіть тоді, коли немає зв'язку з сервером. У режимі офлайн розробники можуть зберігати дані локально на пристрої та вносити зміни без доступу до мережі. Коли підключення відновлюється, Firebase автоматично синхронізує локальні зміни з сервером, забезпечуючи цілісність та консистентність даних.

Аутентифікація та керування користувачами є важливими аспектами Firebase, які забезпечують безпеку та контроль доступу до веб-додатків. Firebase надає розробникам потужні інструменти для аутентифікації користувачів, а також для керування їх правами та обмеженнями доступу до ресурсів додатка.

Аутентифікація користувачів: Firebase пропонує набір автентифікаційних методів, які дозволяють користувачам увійти до веб-додатка зі своїми обліковими записами. Це включає такі методи, як електронна пошта та пароль, облікові записи Google, Facebook, Twitter, GitHub та інші соціальні мережі. Розробники можуть легко налаштувати ці методи автентифікації і забезпечити реєстрацію, вхід і вихід з облікових записів користувачів.

Управління ролями та доступом: Firebase дозволяє розробникам керувати правами та обмеженнями доступу користувачів до ресурсів додатка. Це досягається за допомогою ролей та прав доступу, які можна настроїти для кожного користувача або групи користувачів. Розробники можуть визначати, які дії користувачі можуть виконувати, такі як читання, запис або видалення даних, і встановлювати обмеження на доступ до конкретних ресурсів або функціональності додатка.

Сторонні інтеграції: Firebase також дозволяє інтегрувати сторонні сервіси аутентифікації, такі як OAuth або OpenID Connect, для розширення можливостей автентифікації користувачів. Це дозволяє розробникам використовувати існуючі системи аутентифікації, які вже використовуються користувачами, і спрощує процес реєстрації та входу до додатка.

Підтримка анонімної аутентифікації: Firebase також надає можливість анонімного входу, коли користувачі можуть використовувати додаток без обов'язкової реєстрації. Це особливо корисно для додатків, які не вимагають постійного збереження даних або взаємодії з іншими користувачами.

Загалом, Firebase надає потужні та гнучкі можливості зберігання, синхронізації даних та керування користувачами. Чи використовуючи Realtime Database або Firestore, розробники можуть легко зберігати та синхронізувати дані у реальному часі, виконувати запити до бази даних та керувати інформацією та гарантує злагоджений досвід для користувачів. Аутентифікація та керування користувачами в Firebase дозволяють розробникам створювати безпечні та захищені веб-додатки, забезпечуючи доступ до ресурсів тільки авторизованим користувачам і керуючи їхніми правами. Ці можливості дозволяють розробникам швидко розробляти веб-додатки з синхронізацією у реальному часі, високою масштабованістю та безпекою, задовольняючи потреби користувачів і забезпечуючи зручний та злагоджений досвід використання.

2.3. Інтеграція Firebase з веб-додатками та зовнішніми сервісами

Інтеграція Firebase з веб-додатками та зовнішніми сервісами надає розробникам широкі можливості для розширення функціональності та взаємодії їх додатків з іншими сервісами. Основні способи інтеграції Firebase з веб-додатками та зовнішніми сервісами включають:

1. Інтеграція з веб-фреймворками: Firebase може легко інтегруватися з популярними веб-фреймворками, такими як React, Angular, Vue.js тощо. Він надає бібліотеки та модулі, які спрощують роботу з Firebase в рамках вибраного фреймворку. Це дозволяє розробникам ефективно використовувати функціональні можливості Firebase у своїх веб-додатках.

2. Інтеграція зі сторонніми сервісами: Firebase надає API та документацію, що дозволяють легко інтегрувати його зі сторонніми сервісами.

Наприклад, ви можете інтегрувати Firebase з сервісами електронної пошти, платіжними шлюзами, соціальними мережами та багатьма іншими сервісами. Це дозволяє розширити функціональність веб-додатка та забезпечити його інтеграцію з іншими платформами та сервісами.

3. Інтеграція з серверними функціями: Firebase Cloud Functions дозволяє створювати та виконувати серверні функції, які можна інтегрувати з веб-додатком. Це дає можливість розробникам автоматизувати завдання, обробляти дані, взаємодіяти з іншими сервісами та виконувати специфічні операції на серверному боці. Інтеграція Firebase з серверними функціями дозволяє створити потужні та гнучкі веб-додатки з багатофункціональною логікою обробки даних.

4. Інтеграція зі зовнішніми API: Firebase дозволяє легко взаємодіяти з зовнішніми API через використання HTTP-запитів або сторонніх бібліотек. Це дає можливість отримувати та передавати дані з інших сервісів та платформ, що розширює можливості веб-додатка. Завдяки цій інтеграції, веб-додаток може взаємодіяти з зовнішніми джерелами даних, розширюючи його функціональність та забезпечуючи більш повну картину інформації для користувачів.

Загалом, інтеграція Firebase з веб-додатками та зовнішніми сервісами надає розробникам гнучкість та можливості розширення функціональності їх додатків. Це дозволяє створювати потужні та високоефективні веб-додатки, які взаємодіють з різними сервісами та платформами для надання більш повного та зручного досвіду користувачів.

2.4. Створення проекту Firebase

Створення проекту в Firebase може бути виконано у кілька етапів, які включають підготовку середовища розробки, створення проекту Firebase та налаштування необхідних сервісів. Нижче наведено детальний опис кожного етапу:

1. Створення проекту Firebase: Необхідно відкрити консоль Firebase (https://console.firebase.google.com/) і натиснути на кнопку "Створити проект". Ввести назву проекту та вибрати країну/регіон (для хостингу). Важливо не забути обрати налаштування безпеки проекту, які найбільше підходять для нашого проекту.

2. Налаштування сервісів Firebase: Firebase надає різноманітні сервіси, які були описані раніше та можуть бути використані в нашому проекті. Необхідно вибрати потрібні сервіси, активувати їх та дотримуватись інструкцій з їх налаштування.

3. Підключення до проекту Firebase: Після налаштування проекту Firebase ми отримаємо конфігураційні дані, такі як ключ проекту та інші параметри, які потрібні для підключення до Firebase з вашого веб-додатка. Ці дані можна знайти у вкладці "Налаштування проекту" в консолі Firebase. Необхідно використовувати ці дані для налаштування з'єднання нашого веб-додатка з Firebase.

4. Розробка та тестування: На цьому етапі ми можемо розробляти наш веб-додаток, використовуючи сервіси та інструменти, які надає Firebase. Важливо регулярно перевіряти та тестувати наш веб-додаток, щоб переконатися, що він працює належним чином з Firebase.

5. Розгортання виробленого рішення: Після успішного розроблення і тестування нашого веб-додатку, необхідно розгорнути його, щоб зробити його доступним для користувачів. Firebase надає можливості розгортання та хостингу веб-додатків, такі як Firebase Hosting, проте ми використаємо інший сервіс.

Кожен з цих етапів вимагає уваги до деталей та правильного налаштування, але з використанням Firebase ми можемо забезпечити швидке створення та розгортання потужних веб-додатків з різноманітними функціями та сервісами.

3. ВИБІР ОПТИМАЛЬНОГО ФРОНТЕНД ФРЕЙМВОРКУ

3.1. Огляд фреймворків для розробки веб-додатків

Фронтенд фреймворки є невід'ємною складовою сучасного веб-розроблення. Вони надають зручні і ефективні інструменти для створення користувацького інтерфейсу веб-додатків, які працюють в браузері. Задачею фронтенд фреймворків є спрощення процесу розробки шляхом надання стандартизованої структури, компонентного підходу і набору інструментів для взаємодії зі звичайними елементами веб-сторінок, такими як HTML, CSS і JavaScript. Розглянемо кілька найпопулярніших фронтенд фреймворків:

1. Angular є одним з найпотужніших фронтенд фреймворків, розроблених компанією Google. Він базується на TypeScript, статично типізованій версії JavaScript, що сприяє покращенню безпеки і підвищенню продуктивності. Angular пропонує широкий спектр функцій і можливостей, таких як компонентний підхід, залежності від ін'єкції, маршрутизація і керування станом додатка. Він також має вбудовану підтримку для розробки мобільних додатків і можливості тестування. Angular є потужним інструментом для створення великих, складних веб-додатків з багатими функціональними можливостями.

2. React є одним з найпопулярніших фронтенд фреймворків, розроблених компанією Facebook. Він базується на JavaScript і використовує концепцію компонентів для побудови користувацького інтерфейсу. Одна з особливостей React полягає у використанні віртуального DOM (Document Object Model), що дозволяє оптимізувати швидкість рендерингу і покращити продуктивність додатка. React також підтримує перевикористання компонентів, що дозволяє розробникам створювати масштабовані і підтримувані додатки. Він також має велику екосистему з додатковими бібліотеками і інструментами, що сприяють розширенню його можливостей.

3. Vue.js є ще одним популярним фронтенд фреймворком, який набуває значного популярності в останні роки. Він базується на JavaScript і

пропонує легку і просту у використанні синтаксичну структуру. Vue.js володіє потужним механізмом зв'язування даних, що дозволяє створювати ефективні і динамічні веб-додатки. Він також має компонентну архітектуру, подібну до React, що сприяє перевикористанню коду і забезпечує легку розширюваність. Vue.js має активну спільноту розробників, а також розширювальну екосистему з різноманітними плагінами і інструментами для полегшення розробки.

Angular, React і Vue.js є три найпопулярніші фронтенд фреймворки, які надають потужні інструменти для розробки веб-додатків. Кожен з них має свої особливості і переваги, і вибір між ними залежить від потреб розробника і вимог проекту. Незалежно від вибору, фронтенд фреймворки є незамінними інструментами для розробки сучасних веб-додатків.

3.2. Критерії вибору фреймворку для проекту на базі Firebase

При виборі фреймворку для проекту на базі Firebase слід враховувати кілька ключових факторів, таких як продуктивність, простота використання, розширюваність та спільнота розробників.

1. Продуктивність. При виборі фреймворку для проекту на базі Firebase важливо враховувати продуктивність. Це включає швидкодію фреймворку, його здатність обробляти завдання ефективно та масштабовано, а також оптимізацію шаблонів, даних та запитів. Продуктивність фреймворку впливає на відгук користувача, швидкість завантаження сторінок та реактивність додатку.

2. Простота використання. У процесі вибору фреймворку для проекту на базі Firebase важливо враховувати простоту використання. Це означає наявність зрозумілої документації, дружнього інтерфейсу користувача та зручного API. Простота використання сприяє швидкій і безпроблемній розробці, зменшує час на навчання і дозволяє ефективно використовувати можливості фреймворку.

3. Розширюваність. Розширюваність є важливим фактором при виборі фреймворку на базі Firebase. Фреймворк повинен мати гнучкість і можливості для розширення, дозволяючи додавати власні функціональні можливості та інтегрувати зовнішні бібліотеки. Розширюваність дозволяє адаптувати фреймворк під конкретні потреби проекту і забезпечує його майбутню розширеність та сумісність з новими технологіями.

4. Спільнота розробників. При виборі фреймворку на базі Firebase слід враховувати наявність активної та підтримуваної спільноти розробників. Це включає форуми, блоги, соціальні мережі та інші ресурси, де розробники обговорюють та надають підтримку один одному. Мати доступ до такої спільноти допомагає вирішувати проблеми, знаходити рішення, навчатися та отримувати оновлення і покращення для фреймворку.

Загальною метою вибору фреймворку для проекту на базі Firebase є забезпечення ефективного, стабільного і легко розширюваного додатку. З урахуванням перелічених вище факторів можна зробити обгрунтований вибір фреймворку, який найкраще задовольняє потреби конкретного проекту на базі Firebase.

3.3. Огляд фреймворку Nuxt.js

Nuxt.js є фреймворком для розробки універсальних (Universal) і статичних (Static) веб-додатків на базі Vue.js. Він надає розробникам потужні інструменти для створення високопродуктивних додатків зі складними вимогами щодо рендерингу та маршрутизації. Основні особливості та переваги Nuxt.js включають:

Універсальність (Universal): Nuxt.js дозволяє створювати універсальні веб-додатки, що рендеруються як на серверній стороні, так і на клієнтській стороні. Це дозволяє отримати переваги як з точки зору SEO (пошукова оптимізація), так і з точки зору швидкості завантаження сторінок.

Універсальний підхід Nuxt.js забезпечує покращення відгуку користувача та забезпечує більш гнучкий контроль над рендерингом.

Статичний рендеринг (Static): Nuxt.js також підтримує статичний рендеринг, де веб-сторінки генеруються попередньо та можуть бути кешовані для швидкого відгуку. Це особливо корисно для сторінок зі статичним контентом, таких як блоги або сторінки з інформацією про продукт. Статичний рендеринг дозволяє ефективно використовувати ресурси сервера та забезпечує швидкість завантаження сторінок для кінцевих користувачів.

Автоматична маршрутизація (Automatic Routing): Nuxt.js надає автоматичну маршрутизацію на основі структури папок і файлів проекту. Це дозволяє розробникам зосередитися на створенні компонентів і сторінок, не хвилюючись про складність налаштування маршрутів. Автоматична маршрутизація значно спрощує розробку і підтримку додатків, особливо при роботі з великими проектами з багатьма сторінками.

Пакетування і оптимізація: Nuxt.js має вбудовані інструменти для пакетування та оптимізації веб-додатків. Він автоматично генерує оптимізовані CSS та JavaScript файли, здійснює мініфікацію та компресію ресурсів, що сприяє зменшенню розміру завантажуваних файлів та покращенню продуктивності додатків.

Розширюваність: Nuxt.js є розширюваним фреймворком, що дозволяє використовувати сторонні плагіни та модулі для розширення можливостей. Це дозволяє розробникам використовувати готові рішення, спільнотні модулі та плагіни для швидкого розгортання нових функцій та інтеграції з іншими інструментами.

В цілому, Nuxt.js є потужним фреймворком для розробки універсальних і статичних веб-додатків на базі Vue.js. Він надає розробникам різноманітні інструменти, що сприяють зручності використання, продуктивності та розширюваності проектів. Перелічені особливості Nuxt.js роблять цей фреймворк привабливим варіантом для проекту на базі Firebase.

4. РЕАЛІЗАЦІЯ ПРОЕКТУ FIREBASE ТА ОРГАНІЗАЦІЯ БАЗИ ДАНИХ

4.1. Створення проекту Firebase

Створення проекту Firebase представляє собою важливий крок у розробці сучасних веб-додатків, який забезпечує зручну та ефективну інфраструктуру для зберігання даних, автентифікації користувачів та реалізації різноманітних функцій. Firebase, як мобільна та веб-платформа, розроблена компанією Google, надає розробникам набір інструментів, які спрощують процес розробки, прискорюють внедрення та забезпечують масштабованість додатків. Притримуючись описаної раніше методології створимо проект Firebase

Google розробили Firebase максимально зрозумілим та простим для користувача. Для створення проекту необхідно тільки ввести його назву (рис. 4.1) та якщо не включати сервіси аналітики Google, підтвердити створення проекту натиснувши відповідну кнопку (рис. 4.2).

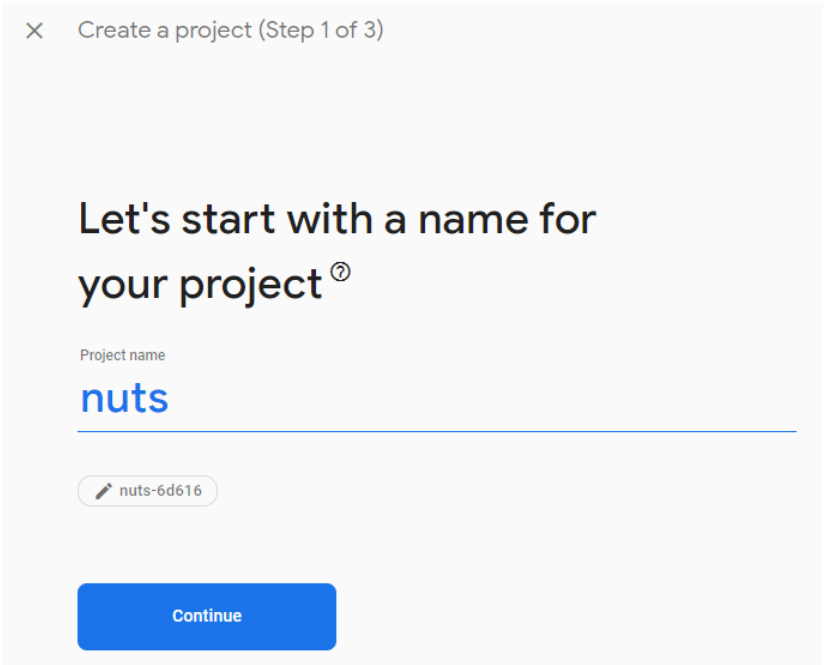

Рисунок 4.1 – Перший крок при створенні проекту Firebase

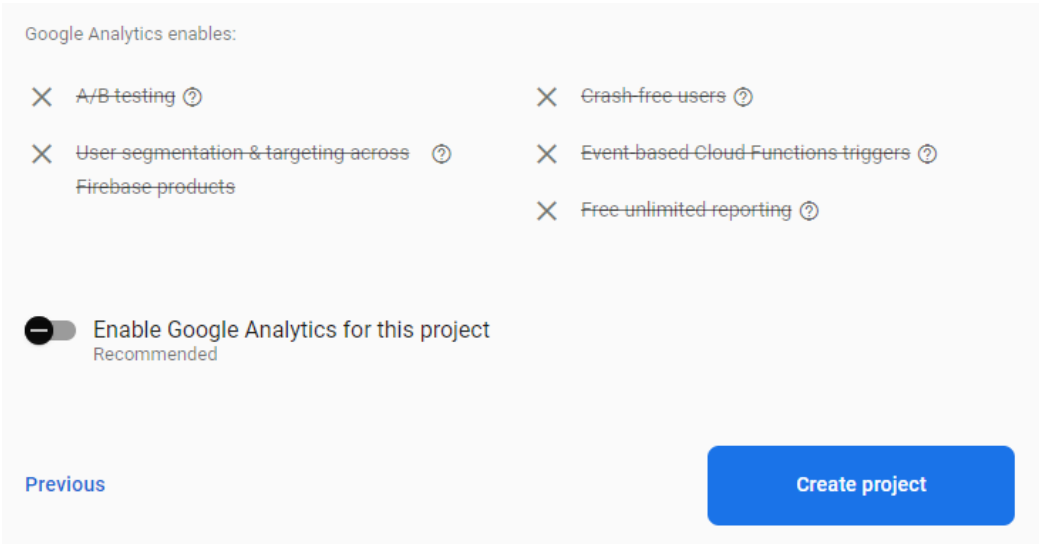

Рисунок 4.2 – Фінальний крок створення проекту Firebase

Для інтеграції Firebase з майбутнім веб-додатком, необхідно виконати декілька простих кроків. А саме, зареєструвати веб-додаток, назвавши його як завгодно (рис 4.3). Налаштування хостингу, пропонованого Firebase ми пропускаємо, оскільки в даному проекті ми використаємо інший сервіс. Фінальним кроком інтеграції є встановлення Firebase SDK в проект (рис. 4.4).

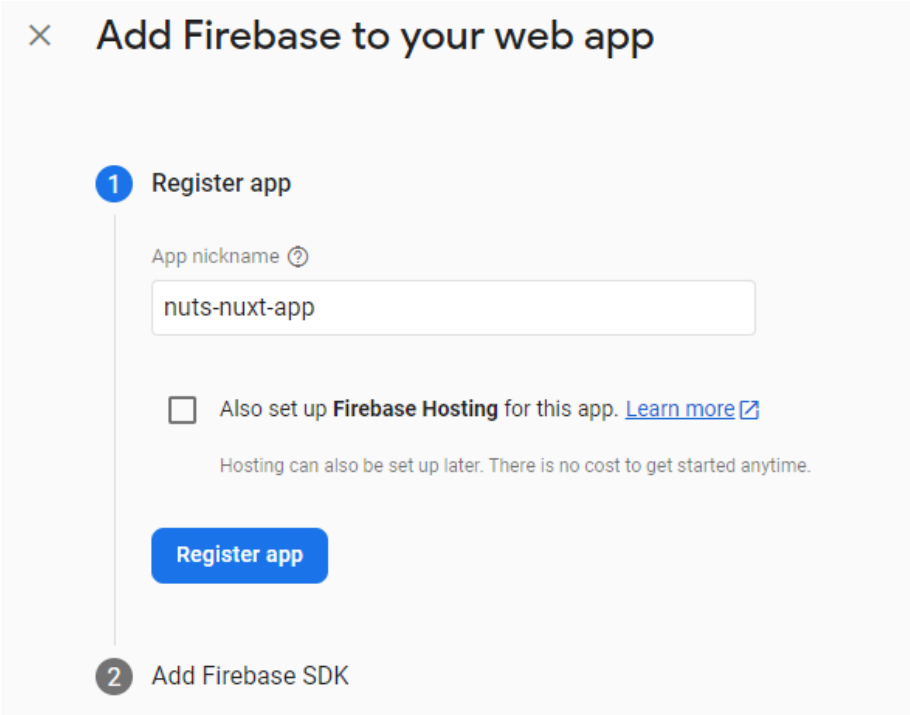

Рисунок 4.3 – Реєстрація веб-додатку в Firebase

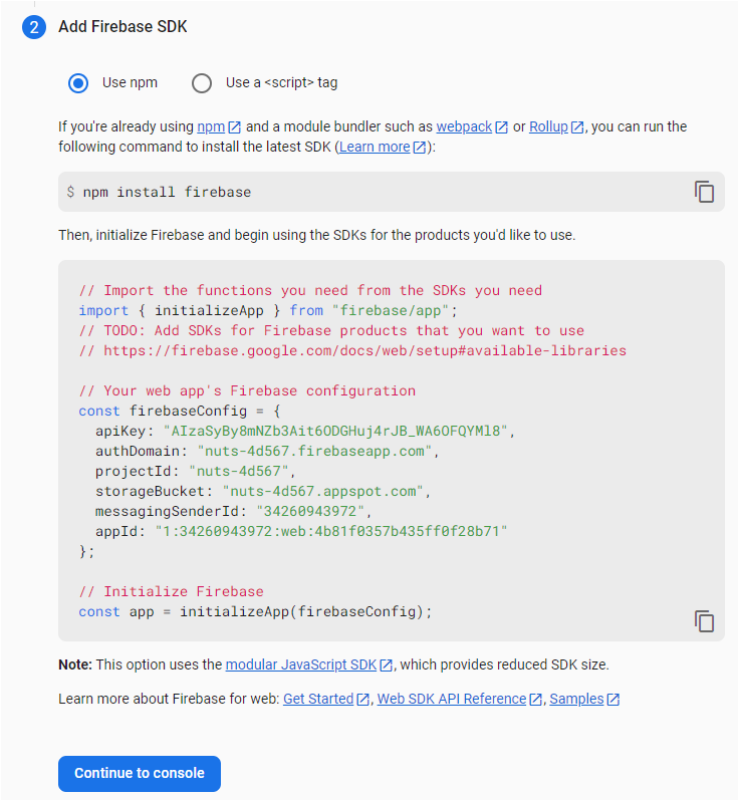

Рисунок 4.4 – Параметри конфігурації Firebase SDK

Також необхідно обрати та налаштувати описані в попередніх розділах сервіси Firebase (рис. 4.5). Загалом, Firebase пропонує чимало різних сервісів для використання, проте обмежимось вибором Authentication, Realtime Database та Storage для даного проекту.

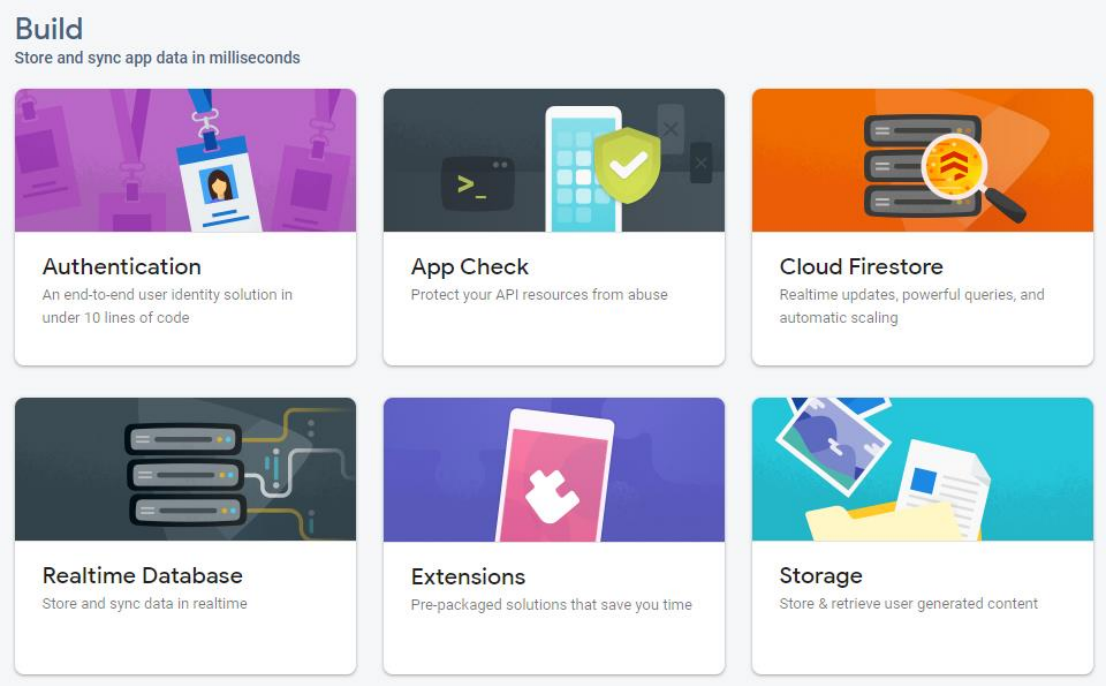

Рисунок 4.5 – Доступні сервіси Firebase

Для налаштування Authentication достатньо обрати провайдера для авторизації на майбутньому веб-сайті. Обираємо стандартний спосіб входу шляхом використання електронної адреси та паролю (рис. 4.6).

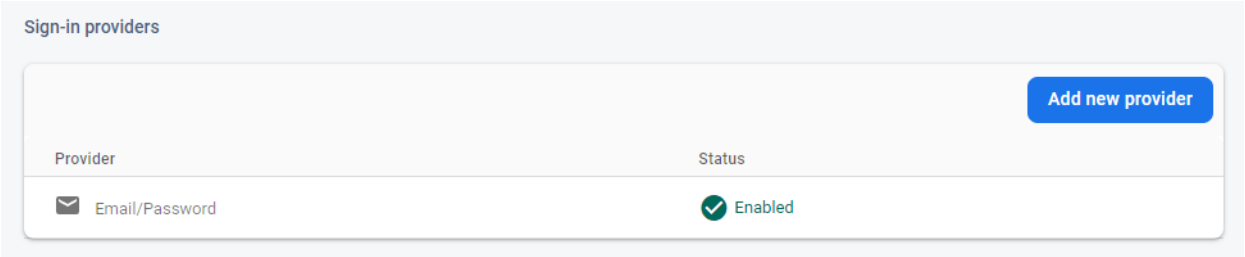

Рисунок 4.6 – Вибір провайдеру для авторизації

Для додавання бази даних в проект необхідно обрати розміщення найближчого до потенційних користувачів серверу (рис. 4.7), у нашому випадку це Європа, та встановлення правил безпеки, що ми пропустимо, оскільки даний проект створюється у навчальних цілях. Аналогічно додається Storage в проект Firebase.

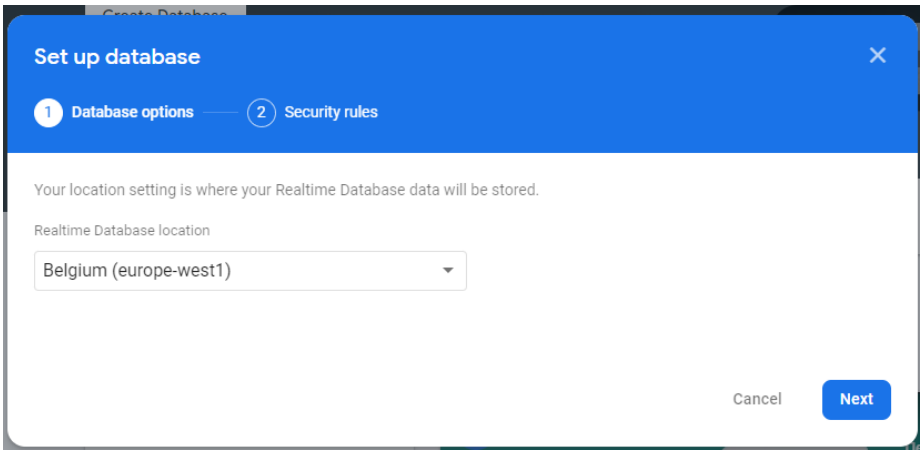

Рисунок 4.7 – Вибір розташування серверу Firebase

4.2. Проектування бази даних

Проектування бази даних для зберігання веб-контенту та даних користувачів є ключовим етапом у розробці сучасних веб-додатків. Від правильної організації бази даних залежить ефективність та надійність системи зберігання і обробки даних. Розглянемо основні принципи та кроки проектування бази даних для зберігання веб-контенту та даних користувачів.

1. Аналіз вимог: Першим кроком у проектуванні бази даних є аналіз вимог до системи. Це включає визначення типів даних, які потрібно зберігати, зв'язків між ними та функцій, які мають бути реалізовані.

2. Нормалізація даних: Нормалізація даних є процесом розбиття бази даних на окремі таблиці з метою уникнення дублювання даних та забезпечення консистентності. Цей процес включає в себе ідентифікацію первинних ключів, визначення залежностей між таблицями та розбиття даних на логічні сутності.

3. Визначення структури таблиць: Після нормалізації даних необхідно визначити структуру кожної таблиці. Це включає вибір полів (стовпців), їх типів даних та обмежень.

4. Встановлення зв'язків між таблицями: Якщо в базі даних існують залежності між таблицями, необхідно встановити відповідні зв'язки. Це забезпечить цілісність даних та можливість виконання операцій зв'язаних таблиць.

5. Визначення індексів: Індекси є структурами даних, які покращують швидкодію пошуку та фільтрації даних. Вони створюються на основі стовпців, які часто використовуються у запитах. Визначення індексів допоможе забезпечити ефективне виконання запитів до бази даних.

6. Оптимізація запитів: Оптимізація запитів є важливим етапом проектування бази даних. Вона включає в себе створення оптимальних запитів, використання індексів та врахування особливостей даних. Правильна оптимізація запитів допоможе забезпечити швидку та ефективну обробку даних.

Storage та Realtime Database значно спрощують та автоматизують створення надійної та ефективної системи зберігання даних, яка задовольнятиме потреби розробленого веб-додатку. Визначившись зі структурою та типами даних, які будуть використані в нашому проекті створимо базу даних (рис. 4.8) та папки в Storage (рис. 4.9) для майбутнього використання.

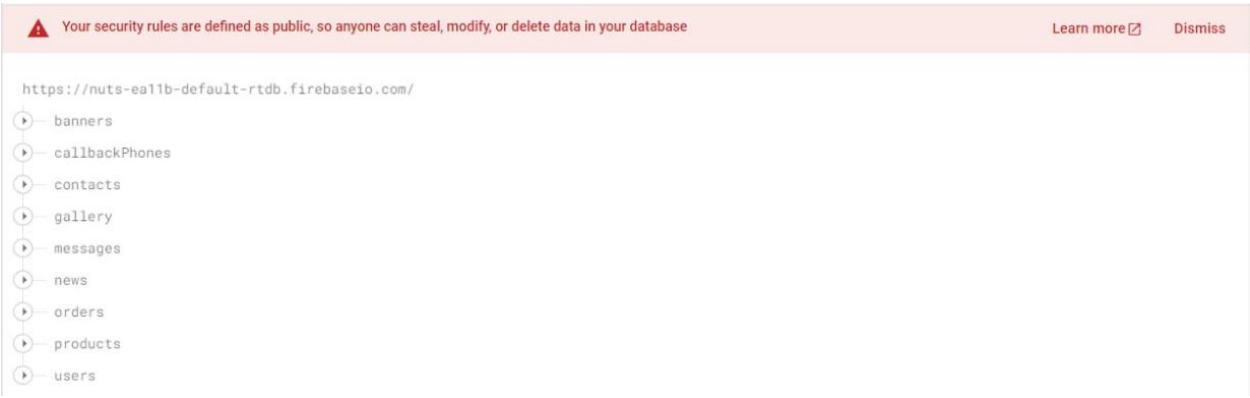

Рисунок 4.8 – Структура бази даних

Рисунок 4.9 – Структура папок Storage

5. РОЗРОБКА ІНТЕРФЕЙСУ АДМІНІСТРАТОРА

5.1. Вимоги до адмін панелі та її функціональні можливості

Адмін панель відіграє ключову роль у забезпеченні зручного та ефективного керування вмістом та користувачами додатку. Розглянемо основні вимоги та функціональні можливості, які необхідно врахувати при проектуванні адмін панелі.

1. Автентифікація та авторизація: Веб-сайт повинен забезпечувати механізми автентифікації та авторизації, щоб забезпечити безпеку та обмеження доступу до інтерфейсу адміністратора. Це включає можливість реєстрації та входу адміністратора, а також встановлення рівнів доступу та прав для керування вмістом та користувачами.

2. Керування веб-контентом: Адмін панель повинна надавати зручні інструменти для додавання, редагування та видалення веб-контенту. Це може включати можливість створювати та редагувати статті, категорії, теги, зображення та інші елементи веб-контенту. Крім того, можуть бути потрібні функції для управління медіа-файлами та іншими ресурсами.

3. Керування користувачами: Адмін панель має надавати можливості для керування користувачами, такі як перегляд, редагування, блокування або видалення облікових записів користувачів. Також можуть бути потрібні функції для надання адміністративних прав доступу та управління ролями користувачів.

4. Аналітика та звітність: Адмін панель може включати функціонал для збору та відображення аналітичних даних, таких як кількість відвідувачів, популярність сторінок, час витрачений на сайт та інші метрики. Також можуть бути потрібні звіти про активність користувачів, публікації та інші статистичні дані.

5. Налаштування та розширення: Важливо мати можливість налаштовувати адмін панель залежно від потреб проекту. Це може включати налаштування загальних параметрів, дизайну, мови, SEO-опцій та іншого.

Крім того, можуть бути передбачені механізми розширення функціональності адмін панелі за допомогою плагінів або модулів.

Усі ці вимоги та функціональні можливості повинні бути ретельно розглянуті при проектуванні веб-сайту з інтерфейсом адміністратора для керування веб-контентом та даними користувачів. Це допоможе забезпечити ефективну та зручну роботу адміністраторів, сприятиме безпеці та стабільності системи та підвищить задоволення користувачів від використання веб-додатку.

5.2. Розробка інтерфейсу адмін панелі

У нашому проекті необхідно розробити функціонал адміністративної панелі для редагування товарів, банерів, новин, галереї, інформації про клієнтів та контакти компанії. Також необхідно інтегрувати Firebase в проект для зберігання та редагування веб-контенту. Як було зазначено раніше, веб-сайт має підтримувати декілька мов, тому необхідно передбачити це, при додаванні контенту.

Для забезпечення механізму автентифікації, було створено форму, зображену на рис. 5.1. Вона є універсальною для авторизації як адміністратора, так і звичайного користувача.

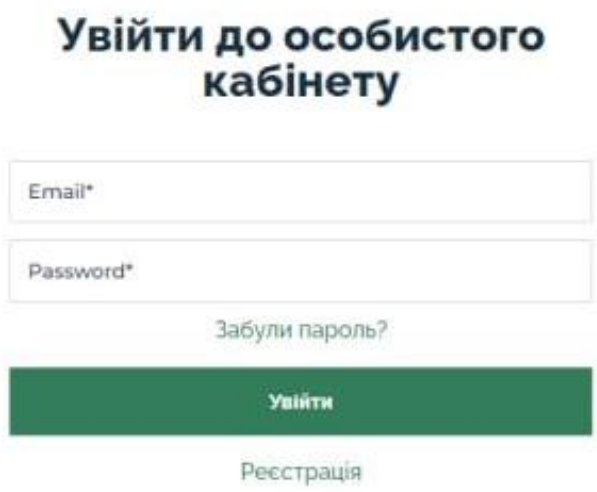

Рисунок 5.1 – Форма авторизації

Реєстрація користувача відбувається за наступною формою, зображеною на рисунку 5.2. Firebase Authentication дозволяє змінювати тривалість сесії входу, що є доволі зручною функцією для розробника.

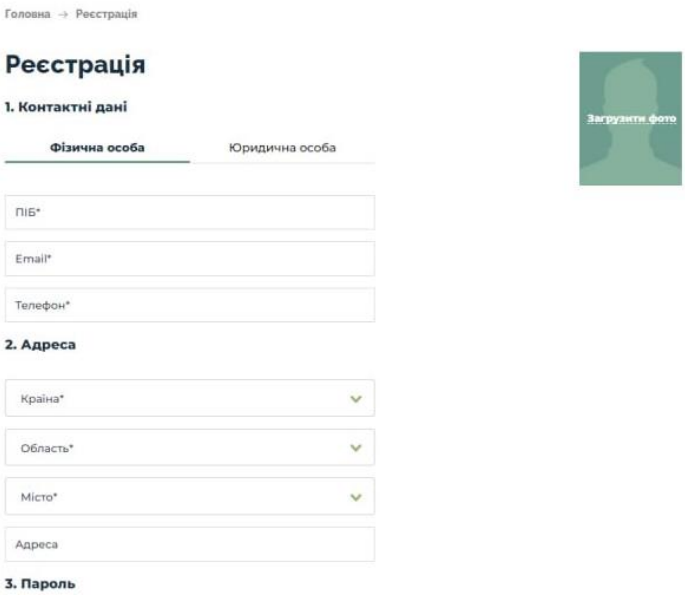

Рисунок 5.2 – Форма реєстрації

Банери є елементом веб-сайту, які дають змогу ознайомитися з завантаженням в Firebase текстових даних та зображень, а також встановлення зв’язку між ними. Вигляд форми завантаження банеру можемо бачити на рисунку 5.3. Вона дає змогу вибрати зображення та заповнити заголовок, текст та посилання на відео банеру.

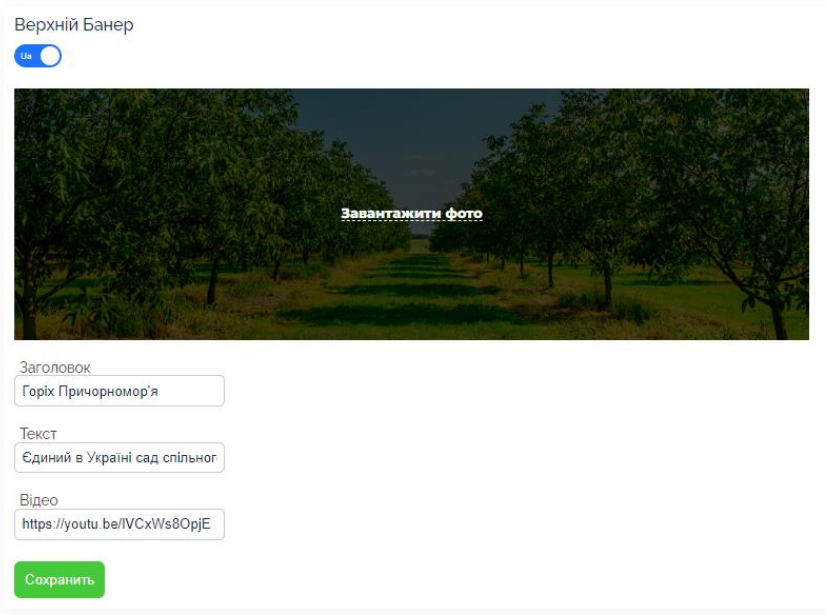

Рисунок 5.3 – Форма завантаження банерів

Створення та товару (рис. 5.4) є аналогічним компонентом з більшою кількістю полів та має зручну таблицю зі списком всіх товарів з можливістю редагування та видалення конкретного товару (рис. 5.5). Користувачі та новини мають подібні таблиці та структуру сторінки. Адміністратори також мають можливість редагування користувачів (рис. 5.6) за необхідності, проте в рамках навчального проекту видалення та блокування користувачів не було реалізовано.

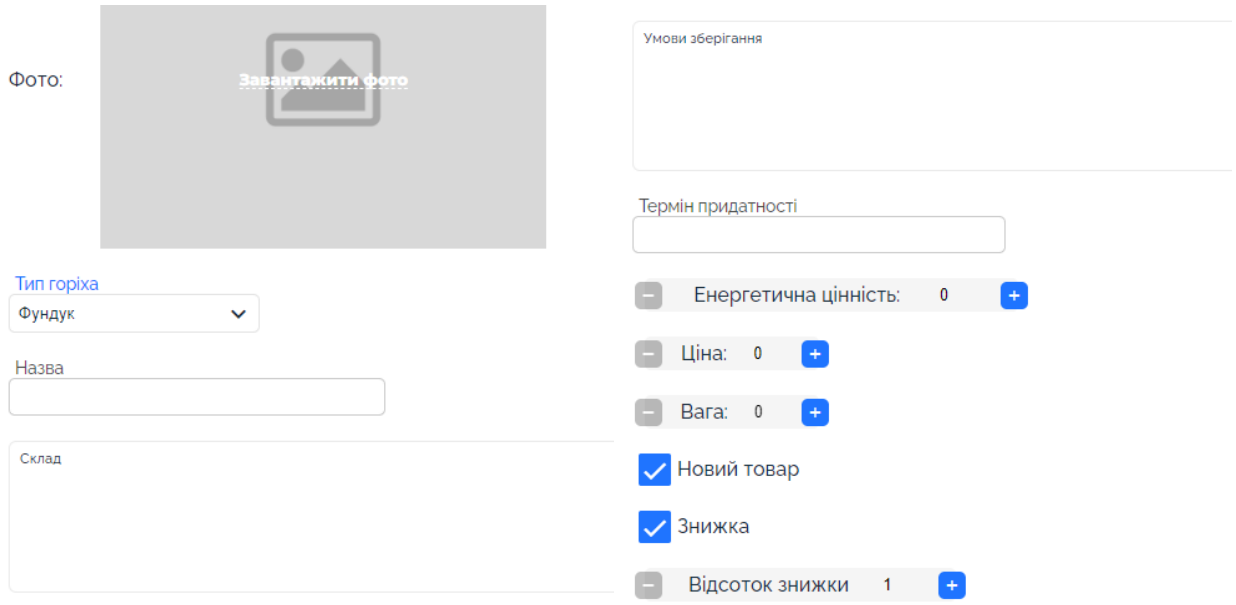

Рисунок 5.4 – Сторінка створення товару

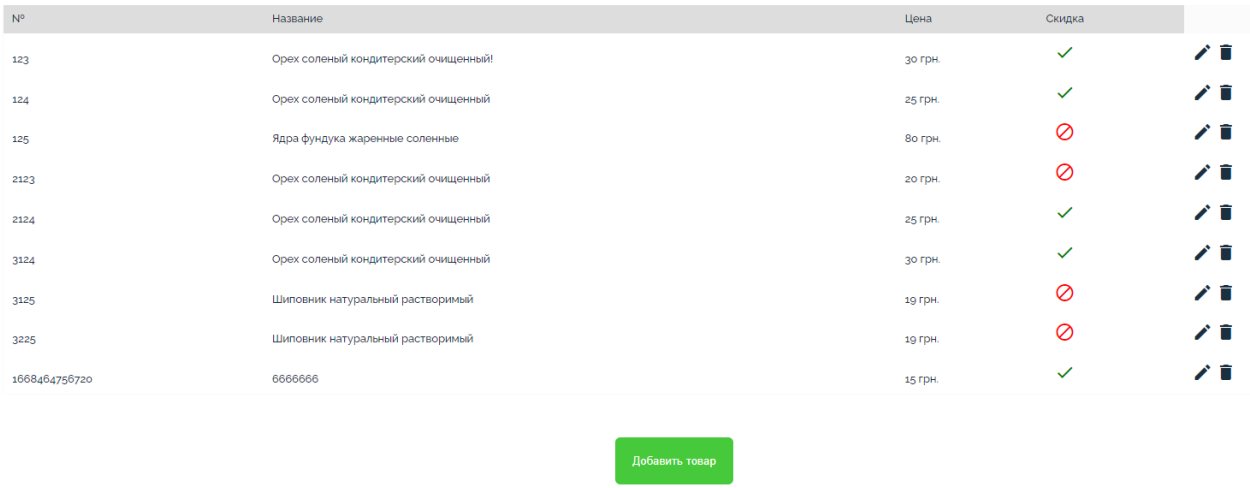

Рисунок 5.5 – Таблиця товарів

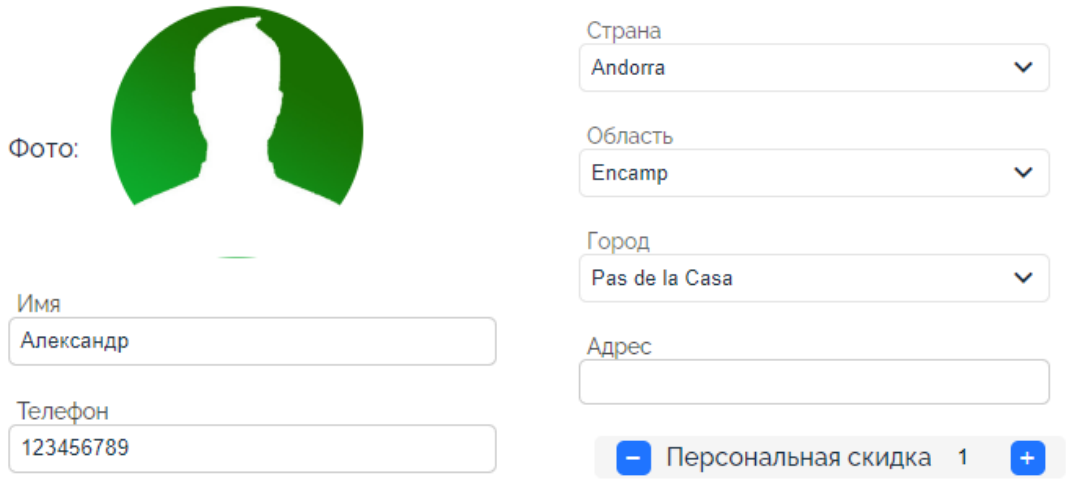

Рисунок 5.6 – Редагування користувача

Щодо аналітики та звітності, у даному проекті було реалізовано можливість переглядати статистику продажу товарів. Покупка товару реалізована шляхом запису замовлення в базу даних, що дає змогу адміністраторам переглядати список замовлень та їх деталі (рис. 5.7).

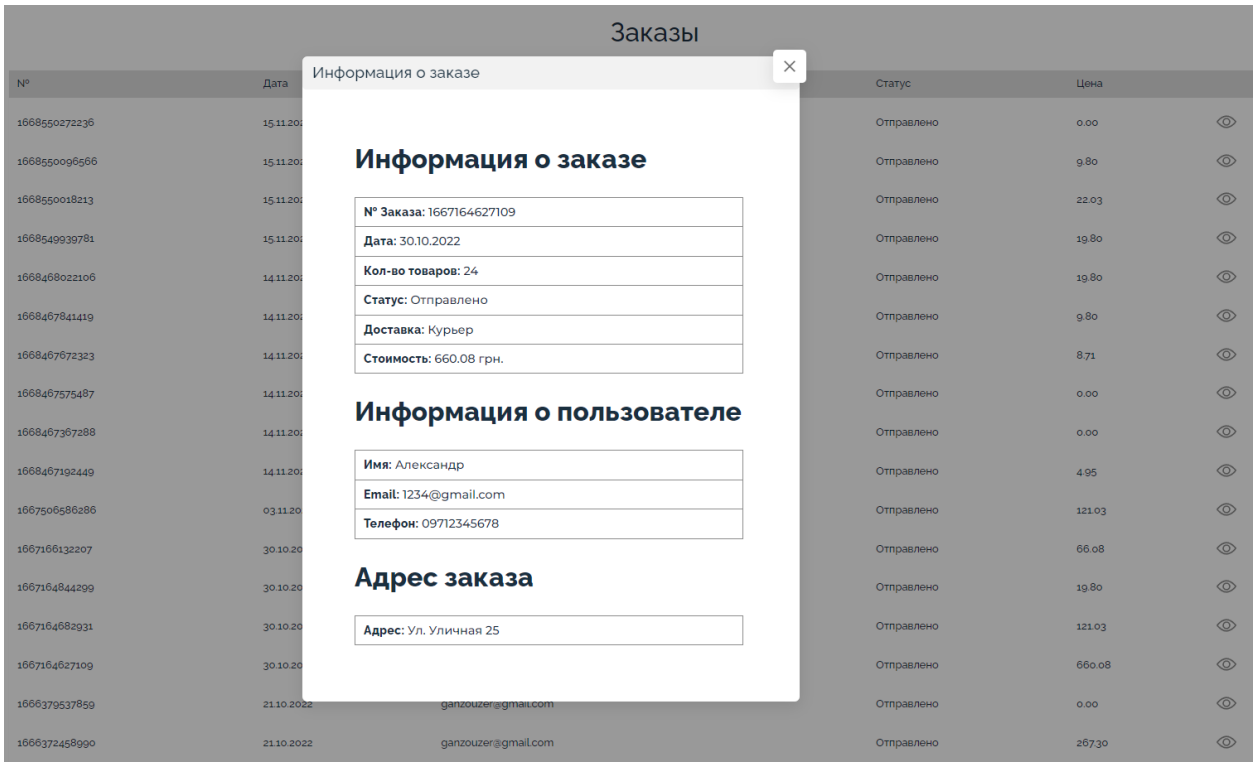

Рисунок 5.7 – Інформація про замовлення

Firebase забезпечує розробників усіма необхідними функціями, які детально описані в їх документації. Розробнику достатньо передати в функції необхідні дані та надіслати запит до Firebase.

6. РОЗРОБКА ІНТЕРФЕЙСУ КОРИСТУВАЧА

6.1. Функціональні вимоги до інтерфейсу користувача

Інтерфейс користувача визначає спосіб взаємодії між користувачем та системою, і його правильне проектування може суттєво покращити користувацький досвід та ефективність використання додатку. Розглянемо основні функціональні вимоги, які необхідно враховувати при проектуванні інтерфейсу користувача веб-додатку.

1. Навігація та структура: Інтерфейс користувача повинен мати зрозумілу та логічну структуру, яка дозволяє користувачеві легко орієнтуватись у додатку. Навігаційні елементи, такі як меню, панель інструментів та посилання, повинні бути чіткими та доступними, щоб допомогти користувачеві швидко знайти потрібну інформацію або функцію.

2. Взаємодія з елементами: Інтерфейс повинен надати користувачу зручні способи взаємодії з різними елементами, такими як кнопки, поля вводу, списки та інші. Елементи управління повинні бути інтуїтивно зрозумілими та легкими у використанні. Наприклад, кнопки мають мати зрозумілі підказки або піктограми, поля вводу мають мати валідацію та можливість введення даних у зручному форматі.

3. Відображення даних та вмісту: Інформація та вміст, що відображаються у додатку, повинні бути привабливими та зрозумілими для користувача. Елементи дизайну, такі як кольори, шрифти та макети, мають бути збалансованими та естетичними. Дані повинні бути представлені у зручному форматі, включаючи таблиці, графіки, списки та інші елементи візуалізації.

4. Функціональність та опції: Інтерфейс повинен надавати користувачу доступ до всіх функціональних можливостей додатку. Важливо забезпечити зручний доступ до різних опцій, налаштувань та дій, які користувач може виконувати. Наприклад, кнопки або меню повинні бути

розташовані таким чином, щоб користувачу було легко знайти та використовувати необхідні функції.

5. Відповідність принципам дизайну: Інтерфейс повинен відповідати сучасним принципам дизайну, таким як простота, чистота, консистентність та доступність. Це означає, що дизайн повинен бути мінімалістичним, зрозумілим для користувача, узгодженим у всьому додатку та пристосованим для різних пристроїв та платформ.

Врахування цих функціональних вимог при проектуванні інтерфейсу користувача веб-додатку допоможе забезпечити зручну та ефективну взаємодію між користувачем та додатком. Правильно розроблений інтерфейс сприятиме покращенню користувацького досвіду, забезпечить легку навігацію та швидкий доступ до функціональності, та сприятиме успішному використанню веб-додатку користувачами.

6.2. Розробка інтерфейсу користувача

Розглянемо розроблений інтерфейс користувача, що має відповідати наведеним у попередньому пункті вимогам. Даний проект має зрозумілу та логічну структуру, яка дозволяє користувачеві легко орієнтуватись по веб-сайту. На рисунку 6.1 можемо бачити, що навігаційні елементи, такі як меню, панель інструментів та посилання є чіткими та доступними, щоб допомогти користувачеві швидко знайти потрібну інформацію або функцію.

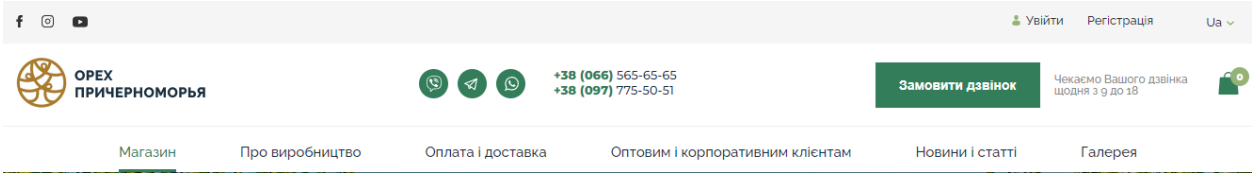

Рисунок 6.1 – Елементи навігації по веб-сайту

На рисунку 6.2 можемо бачити, що розроблений нами інтерфейс надає користувачу зручні способи взаємодії з різними елементами, такими як кнопки, списки та інші. На даному рисунку також бачимо, що додані в попередньому розділі товари коректно відображаються, кнопка придбати

додає товари до кошику (рис. 6.3), що відображає функціональні можливості вебсайту.

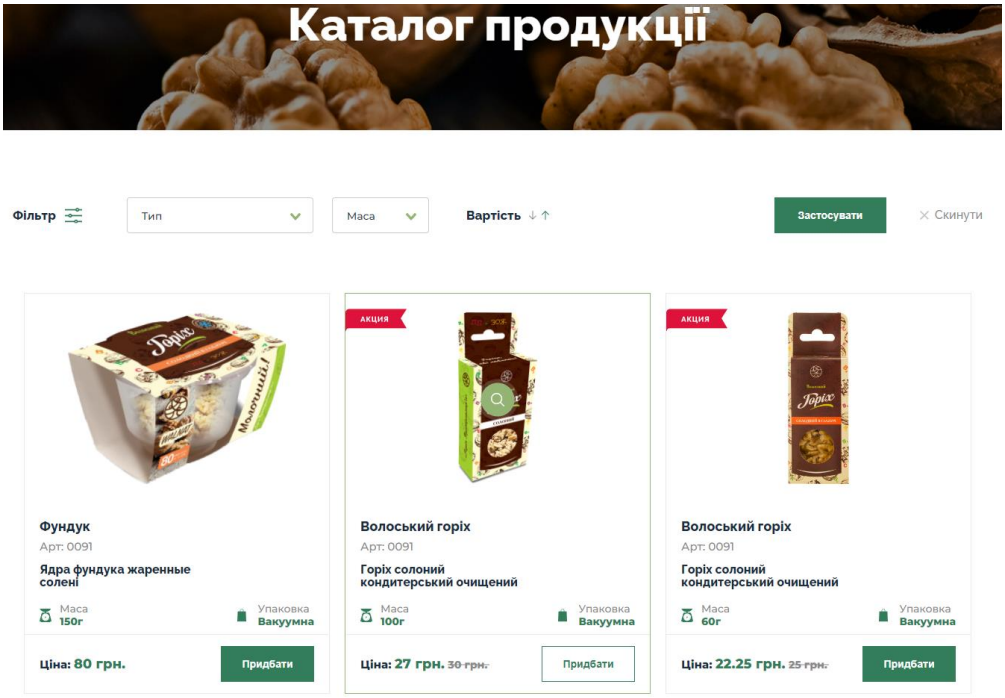

Рисунок 6.2 – Каталог продукції

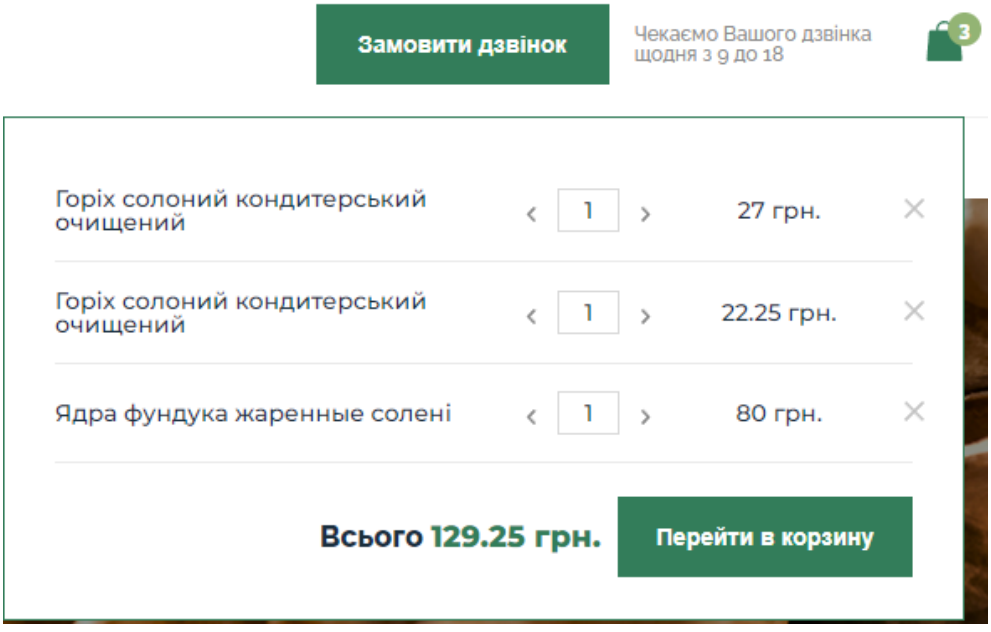

Рисунок 6.3 – Кошик з доданими товарами

Як ми можемо бачити інтерфейс відповідати сучасним принципам дизайну, таким як простота, чистота, консистентність та доступність, а також задовольняє всі зазначені у попередньому пункті умови, що забезпечує зручну та ефективну взаємодію між користувачем та додатком.

7. ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ ВЕБ-ДОДАТКУ

7.1. Опис методів тестування та оптимізації

Тестування та оптимізація веб-сайтів є важливою складовою процесу розробки, що дозволяє покращити якість та продуктивність веб-додатків. Розглянемо методи, які використовуються для тестування та оптимізації веб-сайтів.

1. Функціональне тестування: Функціональне тестування спрямоване на перевірку правильності роботи функцій та функціональності веб-сайту. Це включає тестування навігації, форм, кнопок та інших елементів управління, а також перевірку відповідності функцій заданим вимогам.

2. Тестування користувацького досвіду (UX): Тестування користувацького досвіду спрямоване на оцінку зручності та задоволеності користувачів веб-сайтом. Це може включати збір фідбеку від користувачів, аналіз їх поведінки на сайті та проведення тестування інтерфейсу користувача.

3. Тестування продуктивності: Тестування продуктивності включає перевірку швидкості завантаження веб-сторінок, відповіді сервера, обробки запитів та інших аспектів продуктивності веб-сайту. Це може включати використання спеціалізованих інструментів для вимірювання часу відгуку сервера, оптимізації розміру файлів та кешування ресурсів.

4. Тестування сумісності: Тестування сумісності включає перевірку відображення та функціональності веб-сайту на різних платформах, браузерах та пристроях. Це може включати перевірку сумісності з різними версіями браузерів, розмірів екрану та операційних систем.

5. Оптимізація продуктивності: Оптимізація продуктивності включає вдосконалення швидкості та ефективності веб-сайту. Це може включати мінімізацію файлів CSS та JavaScript, використання кешування, стиснення зображень та ресурсів, оптимізацію запитів до бази даних та використання кращих практик кодування.

6. Тестування безпеки: Тестування безпеки спрямоване на виявлення потенційних уразливостей веб-сайту та застосування заходів для їх запобігання. Це може включати тестування на вразливості введення даних, перехоплення сесій, захист від зламу та інших аспектів безпеки.

Застосування цих методів тестування та оптимізації веб-сайтів допоможе забезпечити високу якість, ефективність та безпеку вашого веб-додатку. Регулярне тестування та вдосконалення веб-сайту є важливим етапом у процесі розробки, що дозволяє задовольнити потреби та очікування користувачів.

7.2. Аналіз результатів тестування та виявлення можливих проблем

У нашому проекті було проведено функціональне тестування, тестування продуктивності, сумісності та безпеки. навігації, Форми, кнопки та інших елементів управління, а також функції авторизації, реєстрації оформлення замовлення пройшли перевірку. Безпека гарантується сервісами Firebase та за налаштування відповідних правил в базі даних відповідає вимогам.

Тестування продуктивності та сумісності було виконано за допомогою сервісу PageSpeed. Даний сервіс виконує перевірку швидкості завантаження веб-сторінок, відповіді сервера, обробки запитів та інших аспектів продуктивності веб-сайту. В результаті показує виміряні часу відгуку сервера, пропозиції щодо оптимізації розміру файлів та кешування ресурсів.

Результат тестування у зручному вигляді можемо бачити на рисунках 7.1 та 7.2. Загалом, спроектований нами веб-сайт має непогані показники, проте деякі є необхідність оптимізації продуктивності та швидкодії системи. Оптимізація зображень є основним фактором, що впливає на швидкодію системи. У сучасних проектах рекомендується використовувати зображення у форматі WebP і AVIF, адже вони часто стискаються краще, ніж PNG чи JPEG. Тому вони швидше завантажуються й використовують менше даних. Даний

підхід допоміг би заощадити близько 50 секунд на мобільних пристроях та 8 секунд на комп’ютерах, що дозволило б отримати показники ефективності близькі до 90-95%.

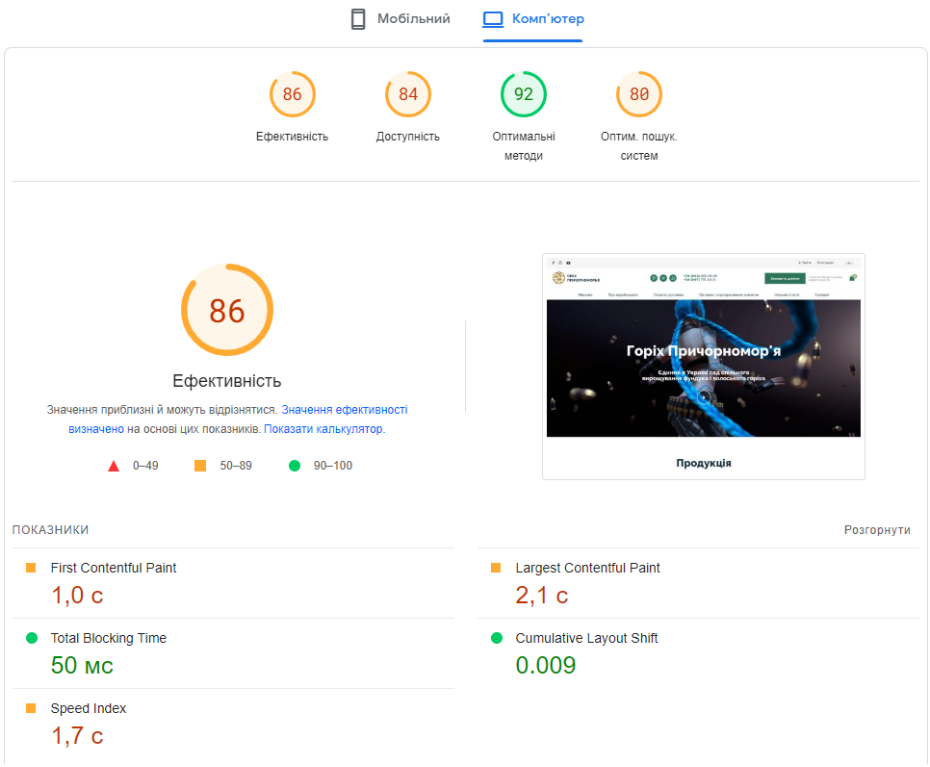

Рисунок 7.1 – Тестування веб-сайту для комп’ютерів

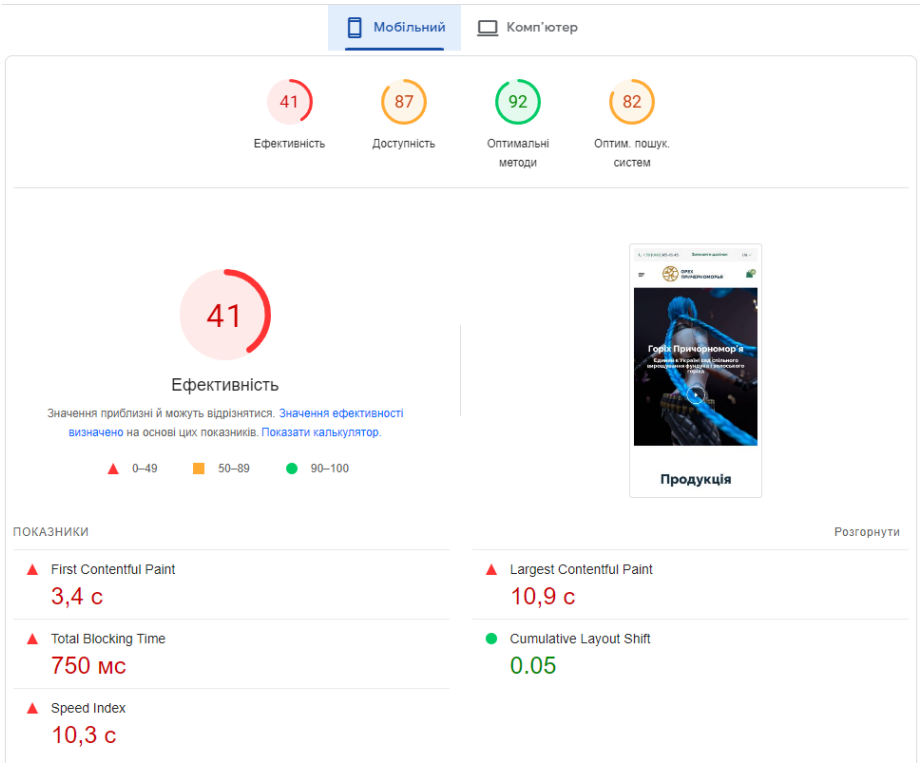

Рисунок 7.2 – Тестування веб-сайту для мобільних пристроїв

ВИСНОВКИ

У даній роботі було розглянуто проектування системи керування веб-контентом на базі Firebase з використанням фреймворку Nuxt.js. Були досліджені основні аспекти створення проекту на базі Firebase, включаючи структуру бази даних, архітектуру проекту та функціональні можливості адмін панелі.

Виявлено, що Firebase є привабливим варіантом для проекту на базі Nuxt.js завдяки його надійності, масштабованості та доступності. Firebase надає готові рішення для аутентифікації, зберігання даних та хостингу, що спрощує розробку веб-додатків і забезпечує швидке впровадження проекту.

Для ефективного використання Firebase в проекті було запропоновано оптимальну архітектуру, яка включає в себе розділення функціональності на клієнтську та серверну частини, використання Firebase Realtime Database для зберігання даних та використання Nuxt.js для створення універсальних або статичних веб-додатків.

Додатково, розглянуті були вимоги до адмін панелі, яка відповідає за керування веб-контентом та даними користувачів. Ці вимоги включали функціональні можливості, такі як управління контентом, керування користувачами, аналітика та звіти, а також забезпечення безпеки та захисту даних. Для забезпечення задоволення користувачів були визначені функціональні вимоги до інтерфейсу користувача, такі як зручна навігація, простота використання, естетичний дизайн та відповідність принципам дизайну.

У підсумку, розробка системи керування веб-контентом на базі Firebase з використанням Nuxt.js є перспективним напрямом, що дозволяє створити потужний та функціональний веб-додаток з використанням сучасних технологій. Це сприяє поліпшенню взаємодії з користувачами, забезпеченню швидкої та надійної роботи системи та задоволенню потреб та очікувань користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Developer documentation for Firebase. Firebase. URL: https://firebase.google.com/docs (date of access: 08.06.2023).
2. Documentation. Angular. URL: https://angular.io/docs (date of access: 08.06.2023).
3. Documentation. React. URL: https://react.dev/learn (date of access: 08.06.2023).
4. Documentation. Vue 2. URL: https://v2.vuejs.org/ (date of access: 08.06.2023).
5. Documentation. Nuxt 2. URL: https://v2.nuxt.com/ (date of access: 08.06.2023).
6. Рогачевський І.С. Створення веб-додатків з використанням Firebase. Київ: Видавничий дім "Слово", 2019.
7. Капралов Є.М., Балабан М.А., Лялька О.В. Створення мобільних додатків на платформі Firebase. Київ: ТОВ "Наш формат", 2020.
8. Lingras, P. Building Cross-Platform Mobile and Web Apps for Engineers and Scientists Using Firebase. Springer, 2017.