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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка інтернет-магазину Modern Store з використанням бібліотек React і Redux та програмного інтерфейсу IndexedDB

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Граб Д.В. (прізвище та ініціали)

Керівник (підпис) Готович В.А. (прізвище та ініціали)

Нормоконтроль (підпис) Литвиненко Я.В. (прізвище та ініціали)

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

Рецензент (підпис) Гладьо Ю.Б. (прізвище та ініціали)

Тернопіль
2023

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

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

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

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

Студенту Грабу Дмитру Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інтернет-магазину Modern Store з використанням бібліотек React і Redux та програмного інтерфейсу IndexedDB

Керівник роботи Готович Володимир Анатолійович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «7» лютого 2023 року № 4/7-133

2. Термін подання студентом завершеної роботи 19.06.2023р.

3. Вихідні дані до роботи Перелік літературних та інтернет джерел

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Теоретичні відомості що до розробки інтернет-магазину. 1.1 Обґрунтування потреби в веб-сайті магазину «Modern Store». 1.2 Поняття веб-сайту. 1.3 Проектування веб-сайту. 1.4 Етапи розробки веб-сайту магазину. 1.5 Обґрунтування вибору системи управління вмістом. 1.6 Висновки до першого розділу. 2. Проектна частина розробки інтернет-магазину. 2.1 Загальна структура сайту. 2.2 Розробка клієнтської частини веб-сайту. 2.3 Розробка та підключення програмного інтерфейсу до веб-сайту. 2.4 Висновки до другого розділу. 3. Безпека життєдіяльності, основи охорони праці. 3.1 Актуальність безпеки життєдіяльності людини. 3.2 Загальні вимоги безпеки з охорони праці користувачів ПК. 3.3 Висновок до третього розділу. Висновки. Перелік джерел.

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., доцент кафедри МТ	05.06.2023	08.06.2023

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про розробку інтернет магазину	24.01.2023-26.01.2023	*Виконано*
3.	Опрацювання джерел про розробку програмного	27.01.2023-31.01.2023	*Виконано*
	забезпечення для розробки інтернет-магазину		
4.	Виконання дослідження щодо актуальності	01.02.2023-07.02.2023	*Виконано*
	технологій та підібраного методу розробки		
	інтернет-магазину		
5.	Оформлення розділу «Теоретичні відомості що до	08.02.2023-09.02.2023	*Виконано*
	розробки інтернет-магазину»		
6.	Оформлення розділу «Проектна частина розробки	10.02.2023-12.02.2023	*Виконано*
	інтернет-магазину»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	05.06.2023-06.06.2023	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	07.06.2023-08.06.2023	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	09.06.2023-11.06.2023	*Виконано*
10.	Нормоконтроль	12.06.2023-13.06.2023	*Виконано*
11.	Перевірка на плагіат	14.06.2023	*Виконано*
12.	Попередній захист кваліфікаційної роботи	15.06.2023	*Виконано*
13.	Захист кваліфікаційної роботи	19.06.2023	

Студент _____ Граб Д.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Готович В.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка інтернет-магазину «Modern Store» з використанням бібліотек React і Redux та програмного інтерфейсу IndexedDB. // Кваліфікаційна робота освітнього рівня «Бакалавр» // Граб Дмитро Віталійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2023 // С. 53, рис. – 15, табл. – 14, кресл. – , додат. – 0, бібліогр. – 34.

Ключові слова: веб-ресурс, інтернет-магазин, інтерфейс користувача, адаптивність, React, Redux, IndexedDB.

Кваліфікаційна роботи присвячена розробці програмного забезпечення для контролю реалізації товарів в інтернет-магазині.

Метою даної кваліфікаційної роботи є дослідження, проектування та розробка програмного забезпечення для контролю та підтримки продуктів в інтернет-магазині.

В першому розділі кваліфікаційної роботи розглянуто та проаналізовано методи та засоби для розробки інтернет-магазину. Здійснено підбір актуальних технологій для розробки інтернет-магазину.

В другому розділі кваліфікаційної роботи здійснено проектування моделі та розробку інтернет-магазину із використанням відповідних технологій.

В третьому розділі кваліфікаційної роботи розглянуто два актуальні питання для заданої теми кваліфікаційної роботи з безпеки життєдіяльності та охорони праці.

ANNOTATION

Development of an online store “Modern Store” using the React, Redux libraries and the IndexedDB software interface // Qualification work of the educational level "Bachelor" // Hrab Dmytro // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2023 // P. 53, fig. – 15, tabl. – 14, chair. – , annexes. – 0, references - 34.

Keywords: web resource, online store, user interface, adaptability, React Redux, IndexedDB, development.

The qualification work is devoted to the development of software for monitoring products in an online store.

The purpose of this qualification work is the research, design and development of software for the control and support of products in an online store.

In the first section of the qualification work, the methods and tools for developing an online store were considered and analyzed. The selection of relevant technologies for the development of an online store was carried out.

In the second section of the qualification work, the design of the model and the development of the online store for using appropriate technologies were carried out.

In the third section of the qualification work, two relevant issues for the given topic of the qualification work on life safety and occupational health are considered.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – Прикладний програмний інтерфейс.

BE (англ. Back end) – Серверна частина застосунка.

CMS (англ. Content Management System) – Система керування вмістом.

CSS (англ. Cascading Style Sheets) – Каскадні таблиці стилів.

DOM (англ. Document Object Model) – Об’єктна модель документа.

FE (англ. Front end) – Клієнтська частина застосунка.

HTML (англ. HyperText Markup Language) – Мова розмітки гіпертексту.

HTTP (англ. Hypertext Transfer Protocol) – Протокол передачі гіпертексту.

HTTPS (англ. Hypertext Transfer Protocol Secure) – Безпечний протокл передачі гіпертексту.

MVC (англ. Model-View-Controller) – Модель-Вид-Контроллер.

PWA (англ. Progressive Web Application) – Поступовий вебзастосунок.

SW (англ. Service Worker) – Обробник подій.

JS (англ. JavaScript) – Мова програмування.

JSON (англ. JavaScript Object Notation) – запис об’єктів в JavaScript.

ПК – персональний комп’ютер.

ЗМІСТ

ВСТУП

Актуальність теми. В сучасному світі електронна комерція стала невід'ємною частиною нашого повсякденного життя. Інтернет магазини набули великого попиту серед споживачів, завдяки своїй зручності та доступності. Вони дозволяють людям замовляти товари та послуги зручно, безпечно та ефективно, не виходячи із комфортного для них місця. Розробка та впровадження інтернет магазину вимагають комплексного підходу, технічної експертизи та знань в галузі електронної комерції.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є створення інтернет-магазину.

Під час виконання даної кваліфікаційної роботи поставлено наступні задачі:

1. Розглянути сучасні технології та методи, які можуть бути використані для створення високоякісного та функціонального інтернет магазину;
2. Проаналізувати поточні практики у сфері розробки інтернет магазинів;
3. Здійснити проектування структури та функціоналу інтернет-магазину;
4. Здійснити вибір та налаштування бази даних для зберігання товарів, замовлень та інших необхідних для функціонування інтернет-магазину даних.

Об'єкт дослідження – інструментальні засоби та методи розробки інтернет-магазину для продажу товарів.

Предмет дослідження – процеси проектування та розробки програмного забезпечення для створення інтернет-магазину.

Практичне значення отриманих результатів полягає в створені програмного забезпечення та інтерфейсу користувача інтернет-магазину із використанням бібліотек React і Redux та програмного інтерфейсу IndexedDB.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ ЩО ДО РОЗРОБКИ ІНТЕРНЕТ-МАГАЗИНУ

1.1 Обґрунтування потреби в веб-сайті магазину «Modern Store»

Бурхливий розвиток інформаційних технологій, а також стрімке поширення мобільних пристроїв призвело до створення та розвитку ринку мобільних додатків. Останнім часом мобільні програми є одним з головних трендів у світі IT-технологій. Мобільні програми полегшують та урізноманітнюють життя користувачам. Спостерігається постійне зростання кількості програмістів, котрі займаються розробкою додатків, пов'язані з збільшення попиту компаній розробку власних додатків, оскільки у час ці докладання дуже цікаві користувачам. Цей перспективний ринок відкриває великі можливості IT-компаніям.

Що ж таке мобільний додаток? Мобільний додаток – програма, яку розробляють під певну мобільну платформу (Android, Windows Phone, iOS). Мобільні програми можуть бути встановлені на мобільному пристрої або їх можна завантажити з онлайн магазинів, наприклад Google play market, App Store, Windows Phone Store. Розробка мобільних додатків спрямована на створення додатків, що передбачають споживчі уподобання, вирішення завдань, алгоритм яких невідомий. Програми можуть проводити аналіз інформації, отриманої з різних джерел, користувач може за допомогою програм контролювати різні процеси і приймати рішення витрачаючи менше часу та ресурсів.

Мобільні програми стають посередниками між користувачем та потоком різної інформації [1].

Програми бувають різної спрямованості:

– мультимедійні, які розширюють можливості мобільних пристроїв під час роботи з ауді та відео інформацією;

– навігаційні, які використовують GPS;

– системні програми, для використання додаткових опцій телефону та програмного забезпечення;

– ігри, які бувають не тільки для розваг але й можуть використовуватися для навчання та розвитку;

– інтернет-магазини для покупок онлайн;

– промо-додатки для реклами різних брендів;

– додатки служб;

– аналоги сайтів чи організацій;

– бізнес-додатки.

Перейдемо до розгляду поняття веб-сайту Progressive Web Apps.

1.2 Поняття веб-сайту Progressive Web Apps

Щоб зрозуміти, що таке Progressive Web Apps (PWA), достатньо уявити, що сайт взаємодіє з користувачем як додаток. Тобто користувач має можливість встановити його на будь-який гаджет, отримувати повідомлення і працювати з ним. Причому робота може продовжуватись навіть офлайн. Ця нова модель програми намагається поєднати функції, пропоновані більшістю сучасних браузерів, з перевагами мобільної роботи.

Progressive Web Apps – це гібрид сайту та програми. Створити його можна як на основі існуючого сайту, так і з нуля. PWA програма може працювати як мобільний додаток, так і як сайт на робочому столі комп'ютера. Одна з головних переваг PWA – його не потрібно створювати для кожної операційної системи. Тому технологія обходиться дешевше, ніж технологія мобільного додатка.

Також PWA може працювати без інтернету що збільшує можливості використання даної технології, хоч в нас час це малоймовірно, щоб користувач не мав доступу до інтернету але дана технологі дає таку можливість, також якщо із інтернетом погане з’єднання згадана технологія надає можливість справно працювати веб-додатку. В офлайні користувач може читати контент і відправляти форми. При підключенні до Інтернету ці форми надійдуть на сервер. PWA програма встановлюється миттєво, коли користувач заходить на сайт. Дані сайту зберігаються у кеші веб-сторінки, завдяки цьому у майбутньому сайт завантажується швидше [2–4]. Швидке завантаження важливе, оскільки за

статистикою 53% користувачів залишають сайт протягом 3 секунд, якщо він не завантажується [5].

Також PWA надає велику кількість можливостей використання на мобільному телефоні, оскільки він встановлюється як додаток. Ось декілька можливостей, які можна використовувати із PWA:

– можливість надсилати push-сповіщення;
– доступ до геоданих;
– доступ до файлів пристроя;
– взаємодія із мікрофоном та камерою.

Користувач може переходити на прогресивну програму під час перегляду веб-сторінок із посилань у соцмережах.

Розглянемо, як PWA встановлюється на смартфон або персональний комп’ютер за прикладом встановлення YouTube Music. Користувачу потрібно відкрити браузер на свому комп’ютері, далі необхідно перейти на відповідний веб-сайт, в нашому випадку це YouTube Music, який користувач би хотів встановити як додаток, звісно якщо потрібний веб-сайт підтримує дану технологію. Відкривши бажаний веб-сайт користувач може побачити в правому верхньому куті відповідну піктограму, яку зображено на риснку 1, яка довзолить встановити додаток даного сайту на девайс після підтвердження встановлення, яке також зображено на рисунку 1.1.

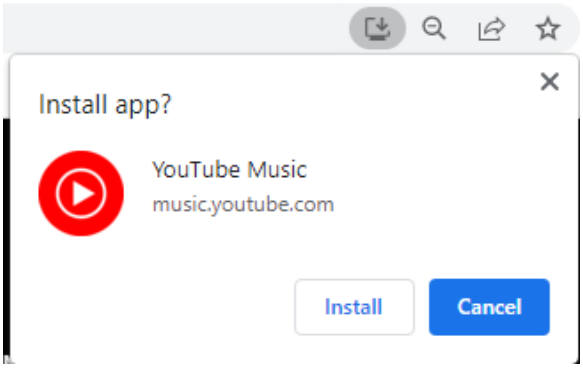

Рисунок 1.1 – Діалогове вікно підтвердження встановлення PWA

Коли користувач встановив бажаний PWA, сайт автоматично відкривається як додаток на вашому пристрої, також на головному екрані з’являється піктограма встановленого сайту.

Багато просунутих сайтів, таких як WebTelegram, AutoRia, Twitter, Facebook. використовують PWA додатки. Прогресивні програми не потрібно реєструвати в магазинах додатків, отже за них не потрібно платити в App Store або Google Play, а також підтримувати та просувати їх дешевше, ніж мобільні програми. У той же час за допомогою функції TWA (Trusted Web Activities) [6] PWA додаток можна додати Google Play, завдяки чому сайт отримує додатковий майданчик для розповсюдження.

Розглянемо перевагу PWA у порівнянні із мобільним додатком. Progressive Web Apps не може повністю замінити мобільні програми, оскільки можливостей у мобільної програми більше. Однак у багатьох випадках PWA-сайт стає дешевшою альтернативою додатку. До переваг PWA слід віднести:

– Прогресивна програма не вимагає оновлень.

– PWA - невеликі за розміром через те, що вони ефективно використовують можливості браузера.

– Швидші у розробці.

У прогресивних програмах легше редагувати контент. Не потрібно редагувати контент окремо у додатку та окремо на сайті.

Користуватися PWA-сайтом простіше, що підвищує

SEO – показники сайту PWA програми затребувані, багато брендів досягли позитивних результатів, скориставшись ними.

У Alibaba щорічна активність користувачів зросла майже на 100 мільйонів завдяки PWA. Користувачі OLX використовують більше мобільний додаток, якщо бути точнішим то 90% користувачів. Завдяки PWA компанія OLX збільшила кількість повернень на сайт у 250%. Всесвітньо відомий додаток для знайомств Tinder використовуючи технологію PWA зменшив об’єм даних мобільного додатку майже у 99% [7]. Було згадано тільки декілька прикладів як PWA впливає на сайт, але таких прикладів в інтернеті можна знайти незчисленну кількість.

За прогресивними додатками майбутнє вони можуть принести користь у різних сферах:

– інтернет-магазинах;

– державних служб;

– салонах краси;

– сервісах доставки.

Для кращого розуміння переваг та недоліків PWA над мобільним додатком порівняємо їх з допомогою таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз PWA з мобільним додатком

Функція	**PWA**	**Мобільний додаток**
Можливість використовувати на різних платформах	Має таку можливість	Потрібно розробляти під кожну платформу окремий додаток
Ціна розробки	Низька ціна	Висока ціна
Місце, яке займає додаток на пристрої	Надзвичайно мало місця	Займає значно більше місця
Можливість встановити додаток на девайс	Можна встановити на любий сучасний девайс	Можна встановити тільки на смартфон
Використання в офлайн режимі	Можна легко використовувати	Неможливо для додатків, яким потрібен зв’язок із сервером
Можливість залишити посилання	Легко можемо залишити посилання на інші сайти	Відсутня така можливість
Швидкий та інтерактивний інтерфейс користувача	Так	Так
Можливість використовувати датчики смартфона	Відсутня така можливість	Можемо із легкістю використовувати датчики

Для кращого розуміння переваг та недоліків PWA над веб-сайтом порівняємо їх, використовуючи ті самі параметри порівняння (табл 2.1).

Таблиця 1.2 – Порівняльний аналіз PWA та сайту

Функції	**PWA**	**Сайт**
Можливість використовувати на різних платформах	Має таку можливість	Тільки в браузері
Ціна розробки	Низька ціна	Низька ціна
Місце, яке займає додаток на пристрої	Надзвичайно мало місця	Не займає взагалі
Можливість встановити додаток на девайс	Можна встановити на любий сучасний девайс	Неможливо встановити
Використання в офлайн режимі	Можна легко використовувати	Неможливо використовувати в офлайн режимі
Можливість залишити посилання	Легко можемо залишити посилання на інші сайти	Легко можемо залишити посилання на інші сайти
Швидкий та інтерактивний інтерфейс користувача	Так	Так

Як було згадано в таблицях 1.1 та 1.2 під час порівняння PWA проти веб-сайту або мобільного додатку, можна замітити що PWA має багато чого спільного як із мобільним додатком так і веб-сайтом, що слугує дуже універсальним рішенням для багатьох компаній.

Також варто згадати що в PWA відбувається швидкий взаємообмін даними, оскільки додаток можна використовувати без підключення до інтернету що вимагає додаткових напрацювань, які дозволять користуватись додатком без підключення до сервера [8-9].

Щоб PWA функціонував справно потрібно підключити SW, який дозволить правильно комунікувати між FE та BE та даними які зберігаємо в кеші,

якщо простими словами то SW виступає посередником між FE та BE частинами. На рисунку 1.2 зображене розташування SW в архітектурі застосунку.

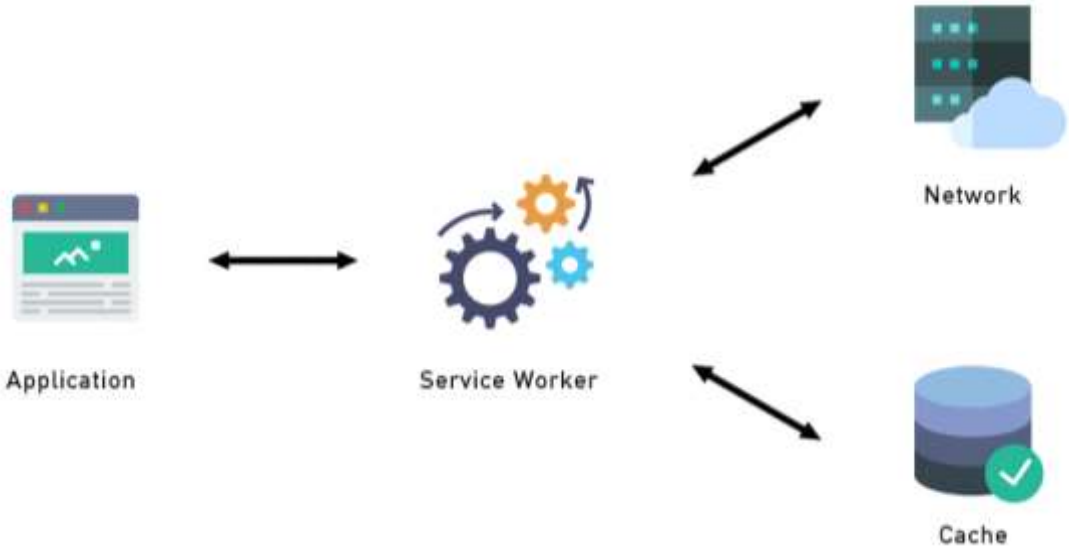

Рисунок 1.2 – Розташування елементів SW в архітектурі застосунка

Зазвичай під час створення веб-сайту розглядається тільки два шари імплементації: FE та BE частини додатку. Якщо потрібно створити PWA, який має бути використаний без доступу до мережі, то необхідно добавити новий шар між FE та BE це SW, який використовується як посередник. Завдяки використанню SW маємо легку можливість переходити із веб-сайту в PWA застосунок. Ось декільки переваг та недоліків SW. Серед переваг можна згадати:

– бездоганна робота навіть коли відсутнє підключення до мережі;
– надає надзичайне підвищення продуктивності завдяки кешуванню;
– використовує push сповіщення;
– може бути встановленим як PWA.

До недоліків SW можна віднести такі як:

– відсутній доступ до localStorage;
– відсутній доступ до DOM;
– відсутній доступ до window.

Як згадували раніше SW має доступ до кешу для web ресурсів але немає жодного доступу до localStorage, а таокж під час роботи без доступу до мережі у застосунка відсутній доступу до бази даних. Використовуючи IndexedDB для

зберігання даних на стороні клієнта, це надає велику свободу дій для логіки бізнесу.

IndexedDB – це низькорівневе API, що дозволяє зберігати значну кількість даних на стороні клієнта. Основна стуь використання IndexedDB полягає в тому що це об’єкто-орієнтована база даних на основі JS. Дана база даних дозволяє зберігати та отримувати об’єкти, які були проіндексовані. Щоб працювати із базою даних необхідно створити схему бази даних та підключитись до бази даних відкривши з’єднання після чого буде можливість отримувати та завантажувати потрібні нам дані [10].

SW дозволяє нам зберігати форми або запити за відсутністю інтернету і надсилати ці дані на BE для обробки після того як з’явиться підключення до мережі. Також варто згадати що SW дозволяє нам не тільки використовувати додаток офлайн, він ще надає такі можливості як:

- обробляє push сповіщення;
- виконання масивних обчислень в окремому потоці;
- може контролювати запити до нашої мережі;
- може модифікувати запити до нашої мережі;
- опрацьовувати запити, якщо відповідь зберігається в кеші;
- може повністю синтезувати відповідь на запит.

Під час використання SW із PWA можна не переживати за безпеку запитів, тому що запити в даному сервісі можуть бути виконані тільки в безпечному середовищі, тобто всі ресурси сайту передаються завдяки протоколу HTTPS, що важливо для безпеки.

Для даного додатку є дуже важливий принцип «offline first», який є найпопулярнішим патерном надання даних для користувача. Якщо потрібний користувачу ресурс є закешовано і він доступний офлайн варто повернути цей ресурс із кешу не звертаючись до сервера, якщо ресурсу немає то за даним методом звертаємось до сервера та кешуємо дані для подальшого використання [11-13].

Відмінність між HTTPS та HTTP полягає у використанні безпечного протоколу для передачі даних між клієнтом та сервером. На днаий момент для

шифрування використовується технології TSL (англ. Transport Layer Security), яка є сучаснішою за SSL (англ. Secure Sockets Layer), які дозволяють шифрувати дані для їх передачі між клієнтом та сервером. На рисунку 1.3 розглянуто різницю між HTTP та HTTPS [14].

Рисунок 1.3 – Відмінність HTTP та HTTPS

PWA має підготовлений спеціальний каркас на стороні клієнта для швидкого завантаження. Коли застосунок було запущено спочатку завантажується клієнтська частина включаючи Web App manifests, далі довантажується з сервера динамічна інформація, яка буде розміщена на сайті.

Web application manifest – це файл із JSON структурою, в якому надається інформація про PWA застосунок браузеру для коректного встановлення додатку на пристрій. Найчастіше в даному файлі надаються такі дані як:

- назва додатку;
- іконка;
- відображення додатку.

Але також в ньому можна вказати і багато інших функцій, такі як:

- фоновий колір додатку;
- масив знімків екрана, яки потрібний для відображення застосунка;
- скороченне ім’я;
- масив ключових слів, до яких може відноситись додаток;
- опис застосунка [15].

Також необхідно опрацювати сповіщення, які дозволяють відправляти з сервера дані на клієнтську сторону, навіть коли застосунок може бути закритим і користувач не працюватиме із ним в даний момент. Повідомлення можуть бути надіслані до SW, який забезпечує отримання повідомлень, навіть коли програма закрита. Сповіщення тут отримуються так само, як і у звичайних додатках, на системному рівні. Відбувається запит на дозвіл від користувача на надсилання повідомлень. Оповіщення мають бути персональними, повинні приходити вчасно, бути доречними та короткими. Робота повинна проводитись незалежно від доступу до мережі, сповіщення не повинні містити рекламу.

1.3 Проектування веб-сайту

Розглянемо найпопулярніші фреймворки та бібліотеки для розробки веб-сайтів. Кожен фреймворк має свої переваги та недоліки але вони всі застосовуються із метою зробити процес розробки швидшим та приємнішим. Розберемося чим відрізняються фреймворки та бібліотеки.

Бібліотеки представляють набір методів та функцій певної мови програмування, які зроблені іншим розробником. Зазвичай в бібліотеках містяться готові рішення, які можна використовувати для розробки власного додатку, що значно спрощує життя іншим розробникам.

Фреймворк потрібний для того щоб задати певну архітектуру під час розробки додатку, якої слід дотримуватись в процесі. Таокж фреймворк виступає як програмний каркас, який спрощує розробку програм та об'єднання компонентів, оскільки містить частину, що не змінюється не залежно від заданої конфігурації [16].

Створити сайт можна різними способами, розглянемо декілька із способів:

- написати код з нуля;
- використовуючи фреймворк написати вихідний код;
- встановлення готового CMS.

Написання повного, робочого та зрозумілого коду з нуля надає можливість реалізувати практично необмежений функціонал але сам процес дуже тяжкий, працезатратний процес, який вимагає багато часу та скурпульозного тестування.

Наступним кроком можна розглянути встановлення готового CMS для керування сайтом та його вмістом за допомогою набору скриптів розроблених в зручному інтерфейсі для легкого використання іншими програмістами. Ось декілька CMS систем для розробки сайтів:

– WordPress;
– Joomla;
– OpenCart;
– Drupal;
– WooCommerce [17].

Цей варіант використовують люди, які погано знають веб-розробку та мови програмування, які необхідні для створення веб-сайтів. Даний підхід дозволяє швидко створити сайт, проте при створенні можна зіткнутися з великою кількістю обмежень.

Використання фреймворку в даному випадку є хорошим варіантом між написанням коду з нуля та використанням CMS. У ньому є каркас, який задає основу для розробки додатку та є функціональна гнучкість, яка дозволяє розробити потрібний нам додаток.

Розглянемо деякі фреймворки які дозволяють взаємодіяти із HTML та CSS та їх особливості:

– Bootstrap;
– UIkit;
– Django;
– Flask;
– Ruby on Rails;
– Spring;
– Angular;
– React;
– Vue;
– Ember [18-19].

Bootstrap це найбільш відомий CSS-фреймворк. Основні інструменти, якого є:

– сітки;
– шаблони;
– типографіка;
– медіа;
– таблиці;
– форми;
– навігація.

Головна перевага даного фреймворку це адаптивність, що дозволяє створити сайт із чуйним дизайном, що динамічно підлаштовується під екран, із забезпеченням правильного зображення сайту на різних пристроях. Сумісний з усіма браузерами, але у старих версіях браузерів виникають проблеми. Він простий у використанні, значно економлять час шаблони та стилі, які вже закладені в фреймворку. Використовує Sass та Less мови [20].

Uikit фреймворк, який має легку та модульну структуру. Відрізняється від інших синтаксичним підсвічуванням для HTML і полегшеної мови розмітки даних. Колекція компонентів має уніфіковану систему найменувань, що забезпечує безконфліктність. Зміни таблиці сітки налаштовуються легко, завдяки компоненту Gril, який заснований на технології Flexbox. Має добре структурований та відкритий код, тому розробник може модернізувати будь-яку частину коду під себе. Розроблено на препроцесорах Sass та Less [21].

Django є найпопулярнішим фреймворком на мові Python. Має велику кількість бібліотек та плагінів. Відмінною особливістю Django є його принцип DRY, який розшифровується як «Don't repeat yourself». Тобто розробникам не слід повторювати рядки коду, які вони вже використовували, тому вихідний код виглядає лаконічно і зрозуміло. До недоліків можна віднести поведінку деяких компонентів непередбачуваною, що не підходить для невеликих проектів [22].

У Flask закладено лише необхідний функціонал, який можна розширювати до рівня, необхідного для даного проекту. Доступна велика кількість розширень, яка може вирішити багато завдань [23].

Фреймворк Ruby on Rails є найбільш популярним на мові програмування Ruby. Фреймворк базується на макимальному використанні коду, який може

повторюватись та на мінімізації коду, який дублюється в додатку. Використовує архітектурний шаблон MVC. Застосунок не повинен створювати свою власну архітектуру, оскільки використовується готовий каркас MVC, де кожна частина знання повинна мати єдине та авторитетне подання у рамках системи [24].

Spring представляє собою універсальний фреймворк для платформи Java. Spring завдяки широкій функціональності дозволяє вирішити безліч завдань, що стоять перед розробниками. Він має розширення потрібні для створення складних бізнес-додатків, але і може використовуватися для створення невеликих проектів. Фреймворк має власну ахрітектуру MVC для веб-додатків. Spring дозволяє забезпечити проект гарною масштабованістю, можливістю простого тестування та спрощеною інтеграцією з іншими фреймворками. У фреймворку присутня модель розробки, яка базується на кращих стандартах індустрії [25].

AngularJS є одним із найпопулярніших фреймворків у світі та на мові програмування JS. Основне призначення даного фреймворку полягає у розробці односторінкових додатків. До переваг AngularJS слід виділити декларативний стиль коду, який дозволяє зробити код легковажнішим. Покращує тестування коду та спрощує відділення DOM та маніпуляції із ним та логікою додатку. Дозволяє паралельно вести розробку над клієнтською та серверною частинами. Фреймворк дозволяє використовувати директиви, які дозволяють приділити особливу увагу логіці та підвищити читання коду. Можна використовувати готові модулі або створювати програми із залежних або автономних модулів [26].

React є найпросунутішою бібліотекою JS, яка використовується для створення веб-додатків. Особливо її добре застосовувати під час створення односторінкових додатків. Використання віртуального DOM дозволяє підвищити продуктивність. Ізоморфні програми використовують один і той же код у серверній частині та клієнтській, тому відпадає дублювання одного і того ж функціоналу. Код, написаний для створення сайту, можна використовувати для створення мобільного додатка [27].

Vue для розробки веб-додатків використовує паттерн MVVM (англ. Model-View-ViewModel), який полегшує розробку інтерфейсу користувача завдяки

розділенню розробки графічного інтерфейсу від розробки логіки на якій формуються окремі частини застосунку та сам застосунок [28].

Ember.js призначений для спрощення створення масштабованих односторінкових програм. Основним принципом даного фреймворка є відпвідність до кожного із існуючих маршрутів певної моделі. У моделях міститься інформація про поточний стан програми. Основною перевагою є відсутність необхідності писати допоміжний код також наявність хорошого оброблювача шляхів та модуля для роботи з даними є великим плюсом [29].

При виборі фреймворку спираємося на такі факти:

– архітектура PWA додатків має на увазі, що додаток повинен працювати без підключення до інтернету;

– додаток повинен працювати як десктопний додаток

– таокж потрібно, щоб на мобільних пристроях та на комп'ютері використовувався один і той самий клієнт.

Щоб задовільнити наші умови, будемо використовувати веб-застосунок та PWA. Щоб у браузері була можливість српавно працювати без підключення до інтернету потрібно використовувати SW, що буде виступати як посередник між сервером та клієнтом. Також будемо використовувати локальні сховища даних та IndexedDB, що дозволить нам зберігати дані на стороні клієнта. Як результат буде розроблено громіздкий клієнт та мінімальна взаємодія із сервером, якщо не повністю відсутня. Це потрібно для того щоб більшість операцій могли виконуватись на стороні клієнта навіть за відсутності інтернету, одним із кращих варіантів для реалізації архітектури із мінімальним використанням сервера служать хмарні послуги. Вони представляють можливість роботи з базою даних через вже налаштований API і лямбда функції, які допомагають реалізовувати візуальні ефекти.

Провайдером хмарного сервісу було вирішено вибрати Firebase, тому що є можливість легкої інтеграції з додатком та присутні зручні інструменти розробки та налагодження з’єдання між сервісом та клієнтом, а також збирає всю необхідну статистику користувачів та аналітику критичних помилок, які стались в програмі на комп’ютері користувача.

Так як ми використовуємо архітектуру із мінімальною взаємодією із сервером, нам не підходять фреймворки, які використовують PHP, Ruby, Python та Java. Також нам не підходять рішення із використанням CMS системи, тому що нам потрібна більш просунута у функціональному плані програма.

Залишається вибір між фреймворками, написаними на JS. Серед них найбільш популярні на сьогоднішній день є:

– React;
– Angular;
– Vue.

Розділимо їх на 2 категорії. Фреймворк представляє інструменти і нав'язує їх нам, тоді як бібліотека компонентів представляє зручний високорівневий інтерфейс, обробку під чистими JS, HTML та CSS та надає свободу вибору інструментарію.Для нашого застосування не потрібна вся міць Angular, тому що він найважчий з усіх і розмір його збірки в середньому в 2 інколи і в 3 рази більший, ніж в наявних аналогів. Отже, його можна сміливо відкинути. У той же час перевага React та Vue є віртуальний DOM. Віртуальний DOM дозволяє значно оптимізувати роботу з браузером та із його DOM деревом за рахунок зменшення кількості перемалювання малюнків та зменшення зміннення DOM вузлів які є HTML елементами завдяки механізму реконсиляції. Між React та Vue було вирішено вибрати React, через більшу спільноту і те, що React розробляється величезною командою Facebook, коли Vue розробляється здебільшого одним розробником.

1.4 Етапи розробки веб-сайту магазину

Серед згаданих раніше фреймворків та відповідних їм мов програмування для написання PWA найбільше підходить JS із використанням типізації за допомогою TypeScript, що дозволить нам уникнути великої кількості проблем через не строго типізований JS.Також це хороший вибір технологій оскільки вони підтримуються у всіх барузерах, а однією із особливостей програм PWA є можливість працювати в більшості існуючих середовищ.

Із існуючих фреймворків для створення веб-сайту із можливістю переходу у PWA додаток було вибрано React для створенню інтерфейсу користувача. Також для контролю стану застосунка було вибрано Redux бібліотеку. Redux є дуже вдалою бібліотекою для контролю стану застосунка інтернет-магазину, оскільки нам потрібний високу продуктивність, що даний менеджер стану цілком надає. Серед плюсів бібліотеки Redux можна згадати:

– велика можливість передбачити стан сховища;
– легкий у виправлені;
– запобігає перерендерингу компонент;
– оптимізована продуктивність;
– легкий у процесі пошуку проблем або виконанні налаштувань.

Серед недоліків бібліотеки Redux можна згадати наступні:

– відсутня інкапсуляція;
– обмежена дизайн структура за рахунок жорсткої архітектури;
– обширне використання пам’яті;
– підвищена складність маніпуляцією стану та методами, які його змінюють або отримують;
– часозатратний для початківців, оскільки потребує глибокого розуміння як працює даний менеджер [30].

Також великою перевагою є можливість тільки читати дані із сховища, що не дозволить модифікувати значення звернувшись до нього на пряму. Щоб змінити значення в сховищі під щас використання даної бібілотеки потрібно стоврити відповідні чисті функції, які дозволять змінити значення в сховищі

Для зберігання масивів даних продуктів інтернет-магазину було вирішено використовувати IndexedDB оскільки вона працює без підключення до інтернету. Серед плюсів варто згадати:

– надає велику продуктивність в пошуку заданого елемента;
– обширна підтримка браузерів;
– використовує асинхроне API, тому загалом має хорошу продуктивність
– може оброблти складні дані.

До недоліків варто віднести складну роботу із API даної бази даних, тому що вона вимагає чіткої структуризації даних за таблицями та сховищами даних. Також IndexedDB надає можливість швидко і легко оперувати великими масивами даних за рахунок індексації та вимогливої структуризації також завдяки індексації можна легко діставати дані для PWA застосунка [31].

1.5 Обґрунтування вибору системи управління вмістом

Вибраний тип програми, а саме PWA вимагає роботи без інтернет-з'єднання, тому нам потрібно передбачити такі моменти, як повноцінна робота без зв'язку з сервером, можливість синхронізації з базою даних при відновленні інтернет-з'єднання. Для того, щоб програма могла працювати без сервера нам потрібно перенести більшу частину, якщо не всю, бізнес логіку додатки на бік клієнта, тим самим зробивши клієнт громістким тасамостійним. У той же час нам необхідно налаштувати механізм синхронізації даних програми з базою даних.

Для цього нам доведеться зберігати історію змін даних на стороні клієнта до тих пір, доки відсутнє з'єднання із сервером, щоб потім була можливість синхронізації даних клієнта з даними в хмарі. Зберігати всю історію все ж таки не найкращий варіант, тому було прийнято рішення про введення додаткової оптимізації для даної функціональності, яка дозволить зберігати не всю історію змін, а лише різницю даних клієнта та бази даних. Це виявилося досить сильним ускладненням клієнтської частини програми, що вплинуло на рішення відмовитися від сервера зовсім, тому що в результаті майже всі операції довелося б перенести на бік клієнта і дублювати їх на сервері, що є не найкращою ідеєю. Також можна виділити, що відмова від сервера є хорошою ідеєю, оскільки через перебої із мережою можуть виникнути проблеми передачі даних. Здебільшого на сьогоднішній день користувачі використовують мобільні пристрої для пошуків бажаних продуктів, що означає що більшість користувачів будуть використувати мобільну версію застосунку. На рисунку 1.4 зображено архітектуру додатку із мінімальним використанням серверної частини.

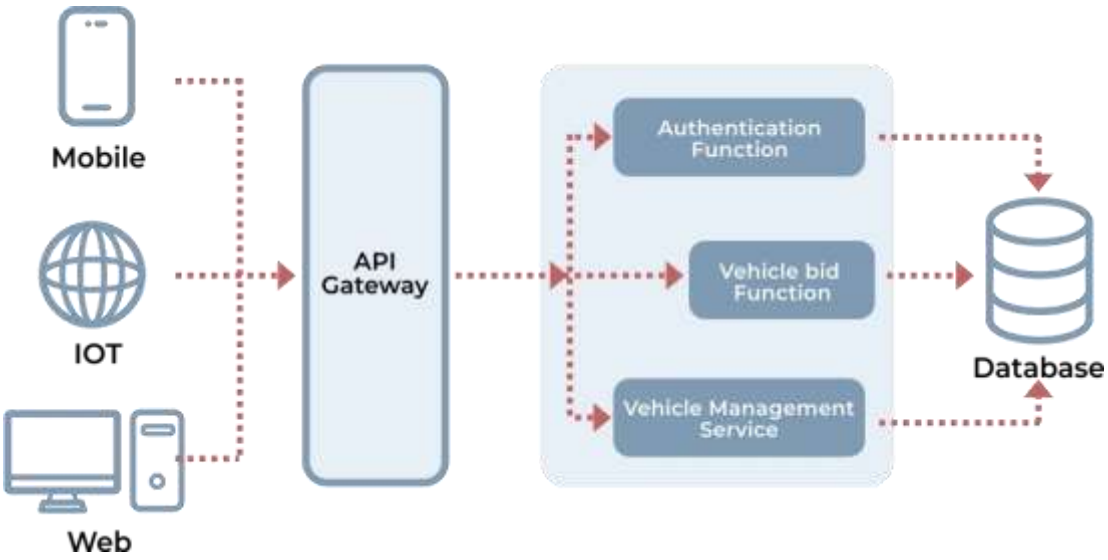

Рисунок 1.4 – Архітектура додатку із мінімальним використанням серверної частини

Хмарні сервіси допомагають зберігати дані в хмарі і надають зручний API для взаємодії з базами даних, що допомагає при розробці додатків і дозволяє відмовитися від сутності сервера в архітектурі. Візуалізація принципу роботи хмарного сервісу представлена рисунку 1.5.

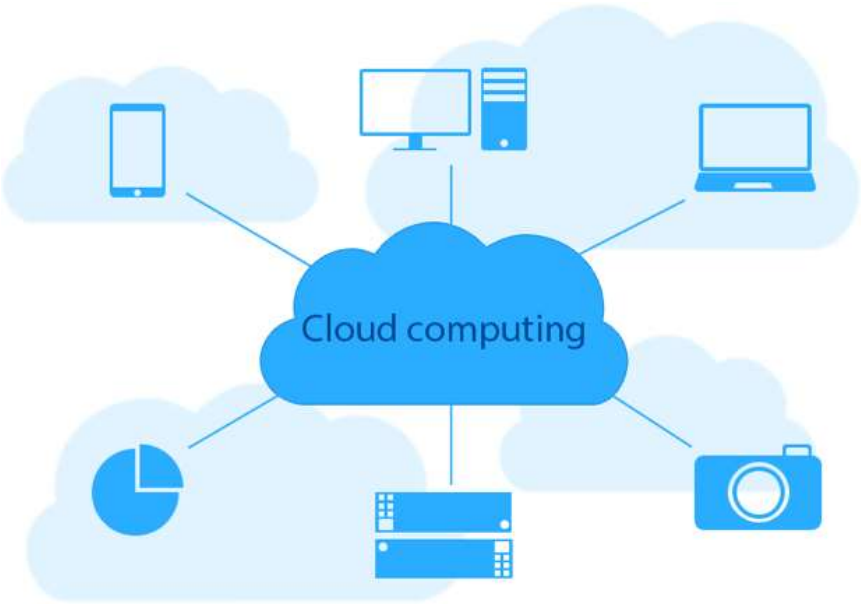

Рисунок 1.5 – Візуалізація роботи хмарного сервісу

За допомогою хмарних сервісів, які легкі у використанні можна значно оптимізувати роботу додатку.

1.6 Висновки до першого розділу

В першому розділі кваліфікаційної роботи було вивчено ряд сучасних технологій та методів, які можемо використати для створення високоякісного та функціоанального веб-сайту. Також проведено ряд аналізів взаємодії між різних технологій за якими краще розробляти інтернет-магазин.

Таокж розглянуто різні архітектурні взяємодії PWA із SW та серверною чатсино та як вони взаємодіють між собою для кращого розуміння суті додатку та як його правильніше розробити, щоб отримати ефективний, простий та зручний додаток.

Розглянули технології та методи, які на даний момент є популярними та широко застосовуються для розробки інтернет-магазинів також розглянули різні варіанти мов програмування та фреймворків для них, за допомогою яких можна розробити веб-додаток. Було вибрано потрібні комбінації мов програмування та фреймворків для реалізації поставленого завдання.

РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА РОЗРОБКИ ІНТЕРНЕТ-МАГАЗИНУ

2.1 Загальна структура сайту

Програмна частина структури інтернет-магазину сприймається як серверна частина.

В адмініструванні будуть утримуватися основні налаштування інтернет-магазину, такі як:

- редагування каталогу товарів;
- управління оформленими замовленнями;
- керування відгуками;
- незавершені замовлення.

У клієнтській частині архітектури розробляється:

- зручний інтерфейс;
- доступні та зрозумілі діалогові вікна;
- зручна система оплати товарів.

Важливим фактором є зворотний зв'язок, що дозволяє користувачеві залишити свою думку, відгук про товар або якість обслуговування магазину в цілому.

При розробці інтернет-магазину важливо одразу побудувати правильну структуру бази даних для коректної роботи системи. Сформувавши правильно структуру бази даних ми зможемо:

- ефективно виконувати запити і швидко отримувати результат;
- легко додавати новий функціонал до застосунку;
- гарантувати цілісність і надійність інформації;
- швидко та точно вносити зміни в базу даних;
- зручно підтримувати та налаштовувати базу даних, а в разі непередбачуваних ситуацій дозволить легко відновлювати дані;

Діаграму бази даних зображено на рисунку 2.1.

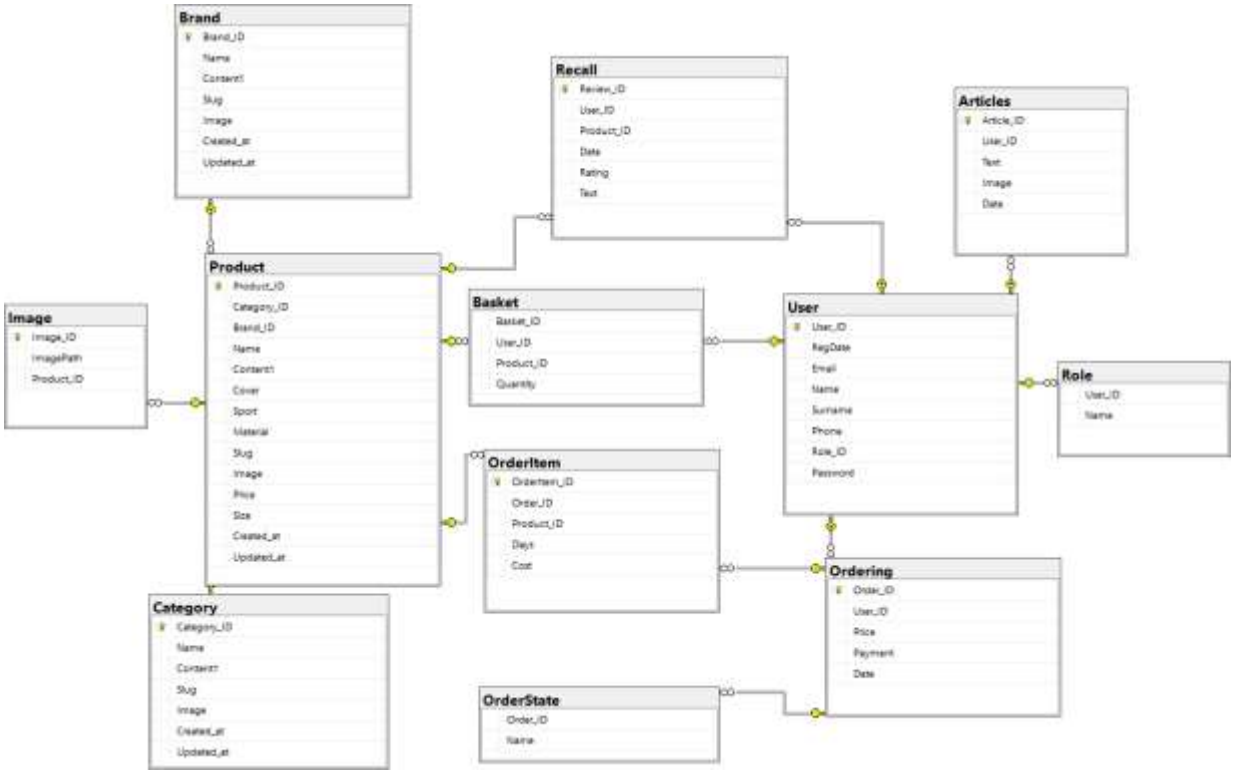

Рисунок 2.1 – Логічна модель бази даних

Для коректного функціонування веб-сайту потрібно створити модель користувача. Вибрано декілька головних полів для користувача, а саме:

– індивідуальний ідентифікатор користувача;

– електронна пошта;

– пароль.

Всі поля які присутні користувачу можна побачити в таблиці 2.1.

Таблиця 2.1 – Таблиця даних, які присутні в користувача

Поле	**Тип даних**	**Опис**
User_ID	Bigint	Ідентифікатор користувача
RagDate	Datetime	дата реєстрації
Email	Varchar(50)	Електронна пошта
Name	Varchar(50)	Ім'я
Surname	Varchar(50)	Прізвище
Phone	Bigint	Номер телефону
Password	Varchar(50)	Пароль
Role_ID	Bigint	Роль користувача

Також необхідно вибрати основні поля для категорій, що будуть визначати де розміщувати відповідний товар або групу товарів. Одними із найважливіших полів було вибрано наступні:

– індивідуальний ідентифікатор категорії;
– індивідуальний ідентифікатор батьківської категорії;
– назва.

Всі інші поля, що відносяться до категорій, можна побачити в таблиці 2.2.

Таблиця 2.2 – Таблиця даних, які присутні в категорії

Поле	**Тип даних**	**Опис**
Category_ID	bigint	Ідентифікатор у таблиці
Parent_ID	bigint	Ідентифікатор батьківської категорії
Name	Varchar(100)	Назва
Content	Varchar(200)	Опис
Image	image	Картинка
Created_at	datetime	Дата та час створення
Updated_at	datetime	Дата та час оновлення

В таблиці 2.3 представлено дані про торгову марку.

Таблиця 2.3 – Таблиця даних, які присутні в торговій марці.

Поле	**Тип даних**	**Опис**
Brand_ID	bigint	Ідентифікатор у таблиці
Name	Varchar(100)	Назва бренду
Content	Varchar(200)	Опис бренду
Image	image	Картинка
Created_at	datetime	Дата та час створення
Updated_at	datetime	Дата та час оновлення

В таблиці 2.4 міститься інформація про товар, який знаходиться в магазині.

Таблиця 2.4 – Таблиця даних, які присутні в продукті магазину

Поле	Тип даних	Опис
Product_ID	bigint	Ідентифікатор у таблиці
Category_ID	bigint	Ідентифікатор категорії товару
Brand_ID	bigint	Ідентифікатор у бренду
Name	Varchar(100)	Назва товару
Description	Varchar(200)	Опис товару
Image	image	Картинка
Price	bigint	Ціна
Created_at	datetime	Дата та час створення
Updated_at	datetime	Дата та час оновлення

Потрібно згадати що в корзину користувача потрапляють товари, які користувач тільки планує придбати. В таблиці 2.5 можна побачити детальну інформацію про поля корзини.

Таблиця 2.5 – Таблиця даних, для індивідуальної корзини користувача

Поле	Тип даних	Опис
Basket_ID	bigint	Ідентифікатор у таблиці
User_ID	bigint	Ідентифікатор користувача
Product_ID	bigint	Ідентифікатор товару
Quantity	int	Кількість товару у кошику

Також варто відмітити, що до даної категорії відносяться товари які вже були куплені користувачем. Також це немає ніякого відношення до особистої корзини. В таблиці 2.6 міститься інформація про замовлення, зроблені користувачем.

Таблиця 2.6 – Дані замовлень зроблених користувачем

Поле	Тип даних	Опис
Order_ID	bigint	Ідентифікатор замовлення
User_ID	bigint	Ідентифікатор користувача
Price	int	Вартість замовлення
Payment	bit	Оплата

На таблиці 2.7 міститься інформація про відгуки користувачів на певний товар.

Таблиця 2.7 – Дані, які відповідають відгукам на товар

Поле	Тип даних	Опис
Review_ID	bigint	Ідентифікатор відгуку
User_ID	bigint	Ідентифікатор користувача
Product_ID	bigintint	Ідентифікатор товару
Date	datetime	Дата розміщення
Rating	Int	Оцінка користувача
Text	Varchar(MAX)	Текст відгуку

На таблиці 2.8 міститься інформація про зображення, яке відповідає певному товару.

Таблиця 2.8 – Дані, які відповідають зображенню на відповідний товар

Поле	Тип	Опис
Image_ID	bigint	Ідентифікатор картинки
ImagePath	Varchar(MAX)	Шлях до файлу зображення товару
Product_ID	bigint	Ідентифікатор товару

Також доцільно згадати що потрібна окрема таблиця для товару, який замовив користувач. На таблиці 2.9 висвітлено відповідні поля, як необхідні для товарів, які замовили користувачі.

Таблиця 2.9 – Дані, які відповідають товару, який замовив користувач

Поле	Тип	Опис
Id	bigint	Ідентифікатор запису у таблиці
OrderId	bigint	Ідентифікатор користувача
ProductId	bigint	Ідентифікатор обраного товару
Cost	float	Вартість

На таблиці 2.10 міститься інформація про можливі статуси замовлень.

Таблиця 2.10 – Таблиця із даними, які відображають можливий стан замовлення

Поле	Тип	Опис
Order_ID	bigint	Ідентифікатор замовлення
Name	varchar(50)	Найменування стану замовлень

На таблиці 2.11 міститься інформація для розмежування прав доступу серед користувачів.

Таблиця 2.11 – Таблиця із даними, які відображають роль користувача

Поле	Тип	Опис
User_ID	bigint	Ідентифікатор користувача
Name	varchar(50)	Найменування ролі

В залежності від веб-сайту або додатка, структура бази даних буває надзвичайно різною, однак описані таблиці є основними, які будуть використовуватись під час розробки веб-сайту. Кожна із таблиць є унікальною та відповідає певній сутності даних, яка відображає той чи інший стан та містить дані продукту, категорії, бренду або користувача, що буде застосовано належним чином.

2.2 Розробка клієнтської частини веб-сайту

Архітектура проекту є дуже важливою частиною розробки любого додатку. Необхідно підібрати або створити архітектуру, яка буде відповідати розмірам та об’єму бажаного застосунка, що в майбутньому дозволить легко добавляти новий код, змінювати існуючий або видаляти непотрібний. Також основним завданням правильно створеної архітектури є розбиття проекту або додатка на окремі частини та відокремлення частин, які не взаємодіють або мало взаємодіють із додатком. Також архітектура вирішує те, як будуть взаємодіяти частини коду між собою.

Перш ніж перейти до прямої розробки клієнтської частини варто створити архітектуру проекту, яку можна легко масштабувати, підтримувати та видозмінювати в разі потреби. Розроблену архітектуру проекту зображено на рисунку 2.2.

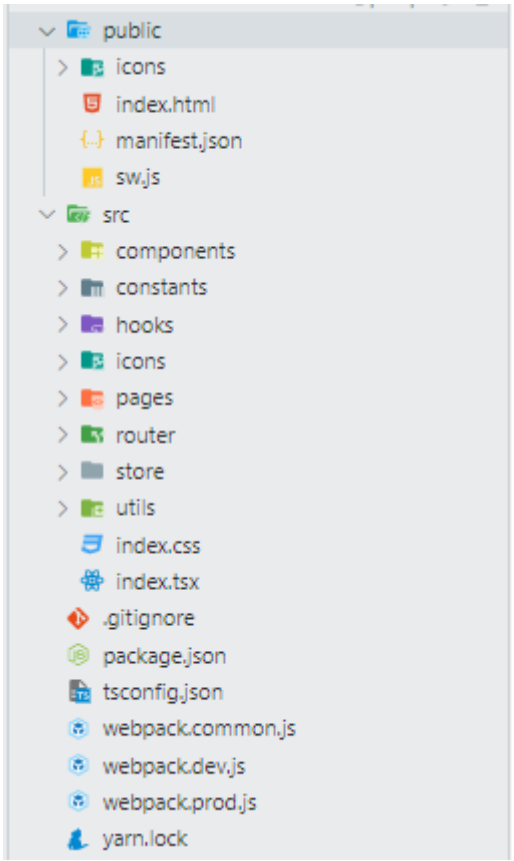

Рисунок 2.2 – Архітектура веб-застосунка

До розробленої архітектури зроблено таблицю 2.12, в якій додано детальний опис відповідних директорій та для чого і з якою метою вони були створені.

Таблиця 2.12 – Опис директорій та їх призначення

Файл/Директорія	**Призначення**
public	Містить публічні файли
Public/icons	Містить іконки, які можуть бути використані як піктограма додатку
src	Розміщує в собі основний застосунок із всім іншими даними
src/components	Розміщує всі необхідні компоненти для справного відображення додатку
src/constants	Містить статичні константи, які можуть існувати в даному застосунку
src/hooks	Містить так звані React hooks, що дозволяють легко та багаторазово використовувати код
src/icons	Містить всі можливі іконки, які використовуються в додатку
src/pages	Містить в собі сторінки, які існують у інтернет-магазині
src/router	Містить в собі засоби маршрутизації між сторінками
src/utils	Містить чисті функції, які можна використовувати в різних файлах застосунка

Для початку розглянемо заголовок нашого додатку, який зображений на рисунку 2.3.

Рисунок 2.3 – Загловок веб-сайту інтернет-магазину

Одразу можна побачити каталог товарів, який представлений як випадаючий список при натисканні на нього. Розгорнутий каталог товарів можна розглянути на рисунку 2.4.

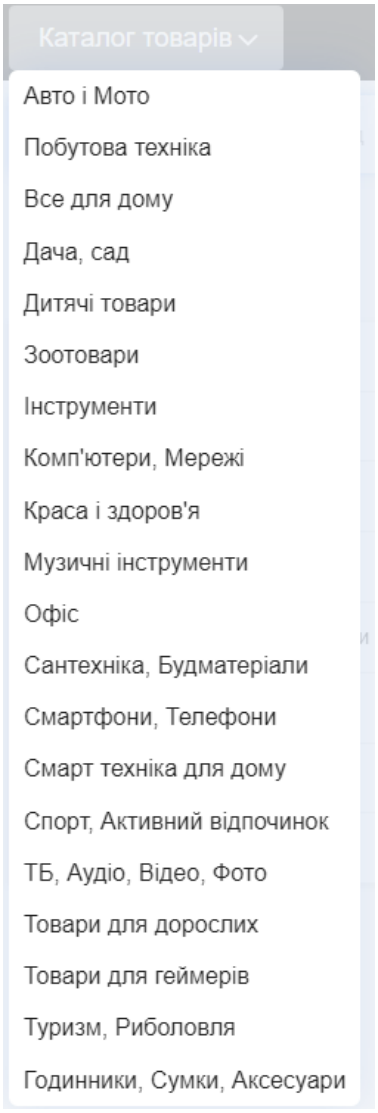

Рисунок 2.4 – Каталог товарів

Якщо попробуємо натиснути на піктограму кошика або користувача то нас направить на форму авторизації. Даний функціонал гарантує нам безпеку користувача, надаючи відповідний контроль доступу та забезпечує персоналізацію. Крім того, це дозволяє правильно керувати сесіями користувачів і зберігати інформацію про авторизацію для зручного та безпечного використання сайту. Форма реєстрації зображена на рисунку 2.5.

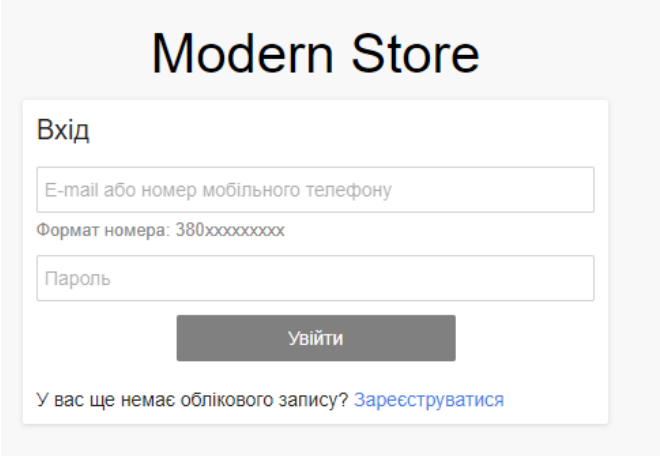

Рисунок 2.5 – Форма авторизації користувача

Код для авторизації в додаток знаходиться в лістингу 2.1.

Лістинг 2.1 – Вхід в додаток

```
export const Login = ({ onSubmit }) => {
  const userNameRef = useRef<HTMLInputElement>(null)
  const userPasswordRef = useRef<HTMLInputElement>(null);
  const handleSubmit = (event: React.FormEvent) => {
    event.preventDefault();
    onSubmit(userNameRef.current.value,
userPasswordRef.current.value);
  };

  return (
    <LoginSignUpWrap>
      <form onSubmit={handleSubmit}>
        <div>
          <div>Вхід</div>
          <input type="text" id="username" ref={userNameRef}
placeholder="E-mail або номер мобільного телефону" />
          <div>{NumberExample}</div>
        </div>
        <div>
          <input type="password" id="password"
placeholder="Пароль" ref={userPasswordRef} />
        </div>
        <button type="submit">Увійти</button>
        <div>
          У вас ще немає облікового запису?{" "}
          <Link to="/registration">Зареєструватися</Link>
        </div>
      </form>    </LoginSignUpWrap>
  );
};
```

Якщо користувач немає персонального аккаунта, він має можливість зареєструватися натиснувши на посилання нижче кнопки входу. Форму реєстрації можна побачити на рисунку 2.6.

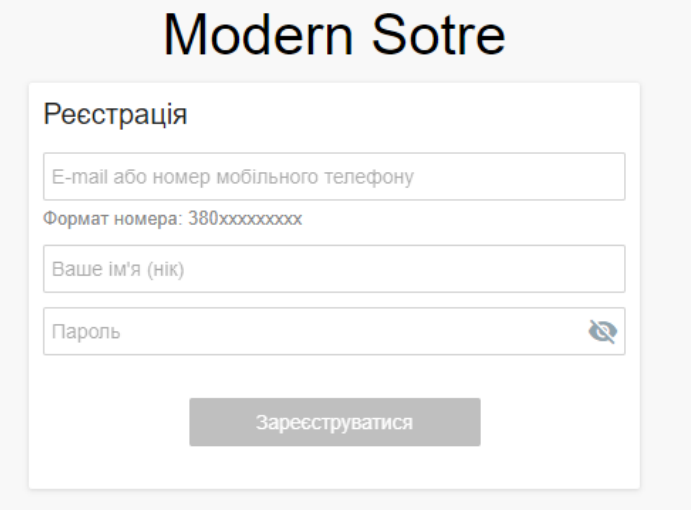

Рисунок 2.6 – Форма реєстрації користувача

Код форми реєстрації зображено в лістингу 2.2.

Лістинг 2.2 – Реєстрація користувача

```
export const Registration = ({ onSubmit }) => {
  const userNameRef = useRef<HTMLInputElement>(null)
  const userNickName = useRef<HTMLInputElement>(null);
  const userPasswordRef = useRef<HTMLInputElement>(null);

  const handleSubmit = (event: React.FormEvent) => {
    event.preventDefault();
    onSubmit(userNameRef.current.value,
userNickName.current.value, userPasswordRef.current.value);
  };

  return (
    <LoginSignUpWrap>
      <form onSubmit={handleSubmit}>
        <div>
          <div>Вхід</div>
          <input
            type="text"
            id="username"
            ref={userNameRef}
            placeholder="E-mail або номер мобільного телефону"
          />
          <div>{NumberExample}</div>
        </div>
        <div>
```

```
          <input
              type="text"
              id="nickName"
              placeholder="Ваше ім'я (нік)"
              ref={userNickName}
            />
            <input
              type="password"
              id="password"
              placeholder="Пароль"
              ref={userPasswordRef}
            />
          </div>
          <button type="submit">Зареєструватися</button>
        </form>
      </LoginSignUpWrap>
    );
};
```

Після того як користувач авторизувався або увійшов на сайт, є можливість побачити зображення користувача в хедері, оновлений хедер зображено на рисунку 2.7.

Рисунок 2.7 – Хедер інтернет-магазину із авторизованим користувачем

Важливим елементом кожного інтернет-магазину є список продуктів, які розміщені на платформі. На рисунку 2.8 зображено список товарів.

Рисунок 2.8 – список товарів розміщених в інтернет-магазині

У лістингу 2.3 зображено код для реалізації списку продуктів.

Лістинг 2.3 – список товарів на сайті

```
export interface Product {
  id: number;
  name: string;
  price: number;
  photo: string;
  description: string;
  brand: string;
}

interface ProductListProps {
  products: Product[];
}

const ProductList: React.FC<ProductListProps> = ({ products }) =>
{
  return (
    <div className="list__wrapper">
      {products.map((product) => (
        <ListItem
          id={product.id}
          name={product.name}
          description={product.description}
          price={product.price}
          photo={product.photo}
          brand={product.brand}
        />
      ))}
    </div>
  );
};

export default ProductList;
```

На лістингу 2.4 зображено дочірній компонент від цілого списку елементів товарів, це одиниця товару, або один елемент із цілого списку

Лістинг 2.4

```
const ListItem: React.FC<Product> = ({
  id,
  name,
  description,
  price,
  brand,
  photo
}) => {
```

```
  return (
    <div key={id} className="list-item__wrap">
      <img src={photo} alt={name} className="img-block" />
      <span className="brand">{brand}</span>
      <span>{name}</span>
      <span>{description}</span>
      <p>{price}₴</p>
    </div>
  );
};

export default ListItem;
```

Кожен товар має власну сторінку на яку ми можемо перейти, і на рисунку 2.9 сторінку продукту.

Рисунок 2.9 – Індивідуальна сторінка продукту

Код для сторінки продукту зображено на лістингу 2.5.

Лістинг 2.5.

```
const handleBuy = () => {
```

```
    onBuy();
  };

  return (
    <div className="product__wrapper">
      <h2 className="product__name">{name}</h2>
      <img src={image} alt={name} />
      <div>
        <div className="price-part">
          <p>{price}₴</p>
          <p>Є в наявності</p>
        </div>
      </div>
      <button onClick={handleBuy} className='button_buy'>
        <CartIcon /> До кошика
      </button>
    </div>
  );
};

export default ProductPage;
```

Було розроблено тестову оплату на сайті за допомогою технології stripe, форму оплати можна побачити на рисунку 2.10.

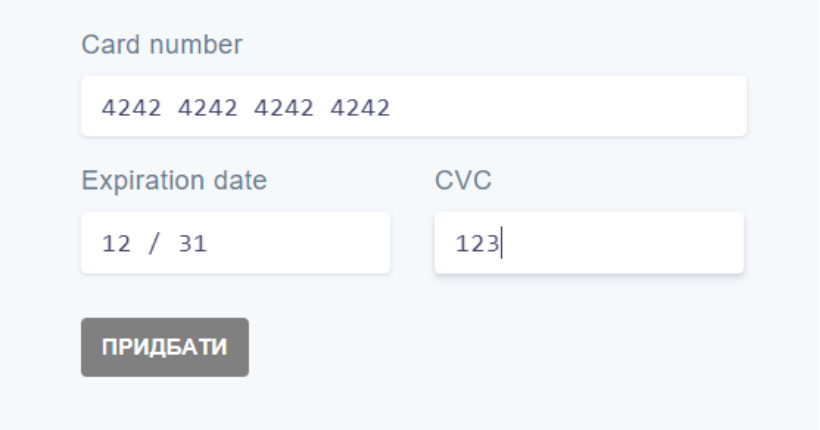

Рисунок 2.10 – Форма оплати на сайті

Код тестової оплати прикріплено в лістингу 2.6.

Лістинг 2.6 – Тестова оплата

```
const PaymentForm = () => {
  const stripe = useStripe();
  const elements = useElements();
  const options = useOptions();
  const handleSubmit = async (event) => {
    event.preventDefault();

    if (!stripe || !elements) {
      // Stripe.js has not loaded yet. Make sure to disable
```

```
      // form submission until Stripe.js has loaded.
      return;
    }

    const payload = await stripe.createPaymentMethod({
      type: "card",
      card: elements.getElement(CardNumberElement)
    });
    console.log("[PaymentMethod]", payload);
  };
  return (
    <form onSubmit={handleSubmit}>
      <label>
        Card number
        <CardNumberElement options={options} />
      </label>
      <div style={{ display: "flex" }}>
        <label style={{ minWidth: "210px", marginRight: 30 }}>
          Expiration date
          <CardExpiryElement options={options} />
        </label>
        <label style={{ minWidth: "210px" }}>
          CVC
          <CardCvcElement options={options} />
        </label>
      </div>
      <button type="submit" disabled={!stripe}>
        Придбати
      </button>
    </form>
  );
};

export default PaymentForm;
```

Хоч даний онлайн-магазин ще не готовий до повноцінного використання проте більшість функцій які необхідні для його існування було додано.

2.3 Розробка та підключення програмного інтерфейсу до веб-сайту

Для того щоб додаток мав можливість використовуватись без доступу до мережі необхідно підключити IndexedDB та SW.

На сам перед нам необхідно створити або відкрити нашу базу даних. Для цього використовується об’єкт IndexedDB та завдяки ньому відкриваємо базу даних, одразу передавши у виклик ім’я бази даних та її версію.

Після відкриття бази даних необхідно визначити структуру об’єктного сховища. Об’єктне сховище є контейнером для зберігання об’єктів. Для того щоб

визначити структуру об’єктного сховища потрібно викликати відповідний методо .

Відкриття та визначення структури об’єктного сховища продемонстровано в лістингу 2.7.

Лістинг 2.7 – відкриття бази даних та визначення структури об’єктного сховища

```
const request = indexedDB.open('myDatabase', 1);
request.onupgradeneeded = function(event) {
   const db = event.target.result;
   const objectStore = db.createObjectStore('myObjectStore', {
keyPath: 'id' });
};
```

Отримавши доступ до об’єктного сховища та визначення структури даного сховища ми можемо отримати доступ до нього. Даний процес можна побачити на лістингу 2.8.

Лістинг 2.8 – отримання доступу до бази даних та об’єктного сховища.

```
request.onsuccess = function(event) {
   var db = event.target.result;
   var transaction = db.transaction(['myObjectStore'],
'readwrite');
   var objectStore = transaction.objectStore('myObjectStore');
   // Виконуйте операції з об'єктним сховищем тут
};
```

Після того, як пройшли всі етапи підключення баз даних та визначення структур об’єктних сховищ, ми можемо виконувати операції із базою даних. Після того як ми отримали доступ до об’єктного сховища ми можемо виконувати різні операції, такі як:

- додавання;
- читання;
- оновлення;
- видалення даних.

Для проведення вищезгаданих операцій із базою даних потрібно застосовувати відповідні методи, такі як:

– add;
– get;
– put;
– delete.

На лістингу 2.9 наведено приклад звернення із використанням одного із згаданих раніше методів

Лістинг 2.9 –приклад звернення до IndexedDB

```
var data = { id: 1, name: 'John' };
var addRequest = objectStore.add(data);
addRequest.onsuccess = function(event) {
   console.log('Дані успішно додані до бази даних.');
};
var getRequest = objectStore.get(1);
getRequest.onsuccess = function(event) {
   var result = event.target.result;
   console.log('Отримано дані:', result);
};
```

Це загальний покроковий опис підключення IndexedDB до веб-додатка. В залежності від наших потреб ми можемо додавати більше функціональності, такої як індексація або фільтрація даних.

Підключення SW до веб-застосунка виконується за допомогою JS. Щоб успішно підключити SW до додатка потрібно створити файл Service Worker. Простіше кажучи створити звичайний JS файл та назвати його так як забажаєте, аби було зрозуміло що це SW файл. У створеному файлі потрібно визначити SW за допомогою обробника подій. Найважливішими подіями є:

– install;
– activate;
– fetch.

Подія install викликається, коли SW вперше завантажується в браузері. Ви можете виконати необхідну ініціалізацію, наприклад необхідних ресурсів.

Подія "activate" викликається, коли SW активовано і він готовий до роботи. Це місце, де можна виконати очищення застарілих закешованих ресурсів або виконати інші дії з обслуговуванням.

Подія "fetch" викликається при кожному запиті на ресурс з вашого веб-застосунку. Ви можете відповісти на запит, використовуючи закешовані ресурси або надсиланням запиту на сервер.

В лістингу 2.10 представлено засіб реєстрації SW у нашому веб-застосунку.

Лістинг 2.10 – Реєстрація SW у веб-застосунку

```
if ('serviceWorker' in navigator) {
  window.addEventListener('load', function() {
    navigator.serviceWorker.register('/service-
worker.js').then(function(registration) {
      // Реєстрація Service Worker вдала
      console.log('Service Worker зареєстровано з успіхом:',
registration.scope);
    }, function(err) {
      // Помилка реєстрації Service Worker
      console.log('Помилка при реєстрації Service Worker:', err);
    });
  });
}
```

Після того як підключили SW до нашого веб-застосунку необхідно переконатися чи SW працює правильно, це можна зробити відкривши консоль браузера і перевірити які повідомлення нам приходять.

2.4 Висновки до другого розділу

В другому розділі кваліфікаційної роботи на було здійснено проектування структури та функціоналу інтернет-магазину завдяки підібраними раніше технологіями та методами розробки. Також вибрали відповідну базу даних, яка буде задовольняти вимоги створення інтернет-магазину. Було здійснено розробку інтернет-магазину із використанням бібліотек React і Redux та програмного інтерфейсу IndexedDB.

Проаналізувавши згадані технології та оцінивши всі їхні переваги та недоліки в другому розділі було розроблено веб-застосунок.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Актуальність безпеки життєдіяльності людини

Проблеми безпеки життєдіяльності людини – є одними з найактуальніших в сучасному суспільстві, що тісно пов'язані з бурхливим розвитком науково-технічного прогресу, погіршенням екологічного стану окремих регіонів та планети в цілому. У теперішній час механізми взаємодії людини та природи, людини та техніки, індивіда та суспільства, все частіше порушуються, що призводить до появи багатьох нових небезпек для нормальної життєдіяльності.

Безпека життєдіяльності людини є завжди актуальною і важливою темою. Це охоплює широкий спектр аспектів, які впливають на безпеку та добробут людей в різних сферах їхнього життя. Особливо в контексті зростаючих загроз і викликів, таких як глобальна пандемія, зміна клімату, кібератаки та тероризм безпека стає ще більш актуальною.

Забезпечення безпеки життєдіяльності охоплює такі сфери, як:

- особиста безпека;
- безпека на роботі;
- безпека в домашньому середовищі;
- безпека на дорогах;
- безпека під час надзвичайних ситуацій.

До цього відноситься заходи для запобігання нещасним випадкам, профілактика хвороб, захист від насильства та злочинності, а також забезпечення безпеки у виробничих, транспортних та інших громадських місцях

Отже, безпека людини – поняття, що торкається сутності людського життя та сфери її діяльності. Безпека – це збалансованій за експертною оцінкою стан людини, соціуму, держави. Значення основ безпеки дозволяє розширити психологічне поле самозахисту особистості й, зокрема, розвивати в ній здатність піклуватися про себе, задовольняти свої потреби та одержувати задоволення від життя. Активна участь кожного громадянина в піклуванні про довкілля і про самого себе є гарантом безпеки особи, суспільства і людства на шляху до

збалансованого розвитку, є критично важливим компонентом у соціально-культурному житті. Отже свобода і захист є фундаментальними складовими безпеки життя й діяльності людини, поза яким не можливий збалансований розвиток людини.

Технологічний прогрес також вносить нові виклики та загрози для безпеки людини. Розвиток інтернет речей, штучного інтелекту, автономних систем та інших нових технологій вимагає розробки ефективних заходів для забезпечення кібербезпеки та захисту приватності [32].

В сучасних умовах, коли розвиток світової економіки перетворив навколишнє середовище на єдиний інтегрований ресурс, який використовується і змінюється системою суспільного виробництва, ретельне врахування економічного фактора при розробці будь-яких екологічних і соціальних програм має дуже важливе значення щодо безпеки життєдіяльності людини [33].

Найбільш економічно розвинені країни в основному завершили перехід до високопродуктивної ресурсозберігаючої економічної діяльності, що створює достатні умови для вирішення складних екологічних і соціальних завдань. Головними факторами, які сприяли цьому, стали: переміщення у малорозвинені країни галузей, які не потребують висококваліфікованої робочої сили і створюють велику кількість відходів на одиницю продукції; структурна перебудова економіки за рахунок прискорення розвитку високотехнологічних і безвідходних галузей; консервація власних природних ресурсів, зростаючі обмеження щодо їх використання.

Загалом, актуальність безпеки життєдіяльності людини полягає в необхідності розуміння потенційних загроз, прийняття заходів для запобігання небезпекам, підготовки до надзвичайних ситуацій та забезпечення безпеки як на індивідуальному, так і на колективному рівні.

3.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

Сучасний розвиток технічного та технологічного стану виробництва передбачає постійну автоматизацію та оптимізацію виробничих процесів. Сьогодні, напевно, важко уявити компанію, господарська діяльність в якій

здійснювалась би без використання комп’ютерної техніки. Через масовий характер робіт, що виконуються працівниками за допомогою комп’ютера, законодавством України чітко врегульовано норми та вимоги до використання комп’ютерної техніки на підприємстві, безпосередньо й охорона праці при роботі з комп’ютером.

Основними правилами під час роботи за ПК є налаштування свого робочого місця таким чином, щоб забезпечити комфортну позицію тіла під час роботи. Правильно регулюйте висоту стільця, положення монітора, клавіатури та миші.

Під час довготривалої роботи за ПК варто робити регулярні перерви використовуючи метод "20-20-20": кожні 20 хвилин зосередженої роботи, відведіть 20 секунд на відпочинок, дивлячись на відстань приблизно 6 метрів, щоб пом'якшити напругу в очах.

Варто забезпечити достатнє природне або штучне освітлення на робочому місці. Уникайте сильного блиску на екрані монітора, а також недостатнього освітлення, що може призводити до напруження очей.

Не потрібно забувати і про правильне положення тіла сидячи перед ПК, підтримуйте правильну позицію тіла. Зберігайте пряму спину, плечі розслаблені, дотримуйтесь натурального положення рук і зап'ястків. Використовуйте спеціальні підставки для зап'ястків та підкладки для підтримки спини

Також потрібно згадати про керування кабелями за робочим місцем. Організуйте кабелі так, щоб уникнути сплутування і забезпечити безпечний простір для руху. Розташовуйте кабелі так, щоб вони не перешкоджали вашому робочому процесу та не створювали ризику.

Запобігання очному напруженню. Регулюйте яскравість та контрастність монітора для зручного перегляду. Регулярно робіть перерви, щоб пом'якшити напруження в очах, і виконуйте вправи для очей, які розслаблюють їх м'язи.

Основні вимоги до приміщення під час роботи за ПК. Приміщення, в яких планується установка та подальша робота з комп’ютером, повинні відповідати проектній документації будинку, погодженій з уповноваженими державними органами. Крім того, роботодавець повинен враховувати санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна

вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів [34].

3.3 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи розглянули питання з безпеки життєдіяльності, яке полягає у тому, наскільки актуальна безпека людини в сучасному світі. Оскільки на даний момент розвиток технологій відбувається надзвичайно стрімка та людина змушена пристосовуватись до цих змін. Також людині потрібно не тільки створювати нові технології, які можуть напряму впливати на неї саму а й як захищатись від тих технологій.

Щодо охорони праці розглянуто питання, яке полягає у загальному дотримані вимог безпеки та правил під час користування ПК.

ВИСНОВКИ

У даній кваліфікаційній роботі розроблено інтернет-магазину під назвою "Modern Store" з використанням бібліотек React і Redux та програмного інтерфейсу IndexedDB. Основною метою роботи було створення функціонального та зручного інтернет-магазину, який задовольняв би потреби сучасних споживачів та надавав би їм зручну платформу для здійснення покупок.

У першому розділі даної кваліфікаційної роботи було подано та розглянуто велику кількість технологій, завдяки яким можна легко розробити інтернет-магазин. Також було проаналізовано, які технології краще взаємодіють між собою, та які краще обрати для створення інтернет-магазину

У другому розділі даної кваліфікаційної роботи було спроектовано архітектуру інтернет-магазину та відповідно розроблено інтернет-магазин із використанням бібліотек React, Redux та програмного інтерфейсу IndexedDB. Зокрема було розглянуто та підключено Service Worker до застосунку, що надало велику перевагу над іншими додатками. Це дуже корисно для поліпшення користувацького досвіду та забезпечення доступності веб-додатку навіть в умовах обмеженого чи непостійного інтернет-з'єднання.

У третьому розділі даної кваліфікаційної роботи розглянуто психологічні фактори впливу навколишнього середовища на людину, та як сучасні технології впливають на психіку людини, також згадано можливості психологічного захисту від середовища. Також було розглянуто загальні вимоги безпеки із охорони праці для користувачів ПК.

Загалом, розробка інтернет-магазину "Modern Store" з використанням бібліотек React і Redux та програмного інтерфейсу IndexedDB виявилася успішною. Оскільки було отримано нові знання та закріплено пройдений матеріал із розробки онлайн-додатків.

ПЕРЕЛІК ДЖЕРЕЛ

1. Що таке мобільний додаток [Електронний ресурс] – 2020. – Режим доступу до ресурсу https://www.techopedia.com/definition/2953/mobile-application-mobile-app.

2. Що таке PWA [Електронний ресурс] – 2023. – Режим доступу до ресурсу https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps.

3. Що таке PWA [Електронний ресурс] – 2023. – Режим доступу до ресурсу https://learn.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/.

4. Як краще побудувати PWA https://www.freecodecamp.org/news/build-a-pwa-from-scratch-with-html-css-and-javascript/.

5. Дослідження швидкості завантаження сайту [Електронний ресурс] – 2022. – Режим доступу до ресурсу https://wezom.com.ua/ua/blog/kak-uvelichit-skorost-zagruzki-sajta.

6. Що таке Trusted Web Activities [Електронний ресурс] – 2019. – Режим доступу до ресурсу https://thanhtungvo.medium.com/what-is-trusted-web-activities-and-how-to-build-a-simple-twa-android-app-dbbdd8590bfc.

7. Опис PWA та найкращі приклади додатку [Електронний ресурс] – 2023. Режим доступу до ресурсу https://www.codica.com/blog/best-examples-of-pwa/.

8. Порівняння PWA та мобільного додатку [Електронний ресурс] – 2022. Режим доступу до ресурсу https://www.magestore.com/blog/pwa-vs-native-app-and-how-to-choose-between-them/.

9. Порівняння PWA та мобільного додатку [Електронний ресурс] – 2023. Режим доступу до ресурсу https://www.sommo.io/blog/pwa-vs-native-app-which-is-better-in-2023.

10. Що таке IndexedDB [Електронний ресурс] – 2023. Режим доступу до ресурсу https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API.

11. Опис роботи SW [Електронний ресурс] – 2019. Режим доступу до ресурсу https://web.dev/service-worker-mindset/.

12. Опис роботи SW [Електронний ресурс] – 2023. Режим доступу до ресурсу https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API.

13. Опис роботи PWA разом із SW [Електронний ресурс] – 2023. Режим доступу до ресурсу https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Tutorials/js13kGames/Offline_Service_workers.

14. Різниця між HTTP та HTTPS [Електронний ресурс] – 2022. Режим доступу до ресурсу https://ssl.com.ua/blog/ukr/http-vs-https/.

15. Опис файлу Web App manifest [Електронний ресурс] – 2023.Режим доступу до ресурсу https://developer.mozilla.org/en-US/docs/Web/Manifest.

16. Різниця між бібліотекою та фреймворком [Електронний ресурс] – 2019. Режим доступу до ресурсу – https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/.

17. Список актуальних CMS систем для розробки веб-сайтів [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://www.entrepreneur.com/guide/small-business-tools/5-best-cms-platforms-in-2023.

18. Список популярних фреймворків для розробки веб-сайтів [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://hackr.io/blog/web-development-frameworks.

19. Список популярних фреймворків для розробки веб-сайтів [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://www.monocubed.com/blog/most-popular-web-frameworks/.

20. Документація до Bootstrap [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://getbootstrap.com/docs/5.3/getting-started/introduction/.

21. Документація до UIKit [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://getuikit.com/docs/introduction#package-contents.

22. Документація до Django [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://www.djangoproject.com/.

23. Документація до Flask [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://flask.palletsprojects.com/en/2.3.x/.

24. Документація до Ruby on Rails [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://rubyonrails.org/.

25. Документація до Spring [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://spring.io/projects/spring-framework.

26. Документація до AngularJS [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://docs.angularjs.org/guide.

27. Документація до React [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://react.dev/blog/2023/03/16/introducing-react-dev.

28. Документація до Vue [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://vuejs.org/guide/introduction.html.

29. Документація до Ember.js [Електронний ресурс] – 2023. Режим доступу до ресурсу – https://guides.emberjs.com/release/.

30. Детальний опис переваг та недоліків бібліотеки Redux [Електронний ресурс] – 2022. Режим доступу до ресурсу – https://thetechprint.com/pros-and-cons-of-redux.

31. Детальний опис переваг та недоліків використання IndexedDB [Електронний ресурс] – 2018. Режим доступу до ресурсу – https://sweetcode.io/indexeddb-client-storage/

32. Актуальність проблеми безпеки людини [Електронний ресурс] – 2016. Режим доступу до ресурсу – https://studfile.net/preview/5152269/.

33. Актуальність проблеми безпеки людини [Електронний ресурс] – 2016. Режим доступу до ресурсу – http://op.iee.kpi.ua/1/%D1%80.54-58.pdf.

34. Охорона праці під час роботи за ПК [Електронний ресурс] – 2018. Режим доступу до ресурсу – https://www.victorija.ua/dovidnik/osnovni-pravyla-dotrymannya-ohorony-pratsi-pry-roboti-na-personalnyh-eom.html.