МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп‘ютерної та програмної інженерії
Кафедра комп’ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ Аліна САВЧЕНКО
«____» _____ 2021 р

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИЦІ ОСВІТНЬОГО СТУПЕНЯ

«МАГІСТРА»

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ «ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ»

Тема: «Web-застосунок для управління послугами в сфері інформаційних технологій»

Виконавиця: Ткаченко Анна Сергіївна

Керівник: д. т. н., доцент Савченко А.С.

Нормоконтролер: _____ Ігор РАЙЧЕВ

Київ - 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп‘ютерної та програмної інженерії

Кафедра комп’ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 “Інформаційні технології”, 122 “Комп’ютерні науки”, “Інформаційні управляючі системи та технології”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Аліна САВЧЕНКО
“_____”_____ 2021 р.

ЗАВДАННЯ

на виконання дипломної роботи студентки

Ткаченко Анни Сергіївни
(прізвище, ім’я, по батькові)

1. Тема роботи: «Web-застосунок для управління послугами в сфері інформаційних технологій» затверджена наказом ректора № 2228/ст. від 12.10.2021р.

2. Термін виконання роботи: з 11.10.2021 по 27.12.2021р.

3. Вихідні дані до роботи: процес отримання та надання послуг в сфері інформаційних технологій для організацій, які проводять автоматизацію та оптимізацію робочого процесу.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): вступ, аналітичний огляд і постановка задачі, огляд технологій та інструментів для розробки web-застосунку, розробка функціональної складової та інтерфейсу для користувачів, висновки.

5. Перелік обов’язкового ілюстративного матеріалу: схема бази даних web-застосунку, схема складових компонентів архітектури web-застосунків.

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Підпис керівника
1	Аналіз літератури та джерел за темою дипломної роботи.	11.10.2021р. – 21.10.2021р.	
2	Розробка та затвердження плану дипломної роботи.	22.10.2021р.	
3	Проведення консультації з науковим керівником щодо створення першого розділу.	25.10.2021р.	
4	Аналітичний огляд і постановка задачі.	26.10.2021р. – 01.11.2021р.	
5	Аналіз існуючих рішень для управління послугами в сфері інформаційних технологій.	02.11.2021р. – 10.11.2021р.	
6	Огляд технологій для розробки web-застосунку.	11.11.2021р. – 16.11.2021р.	
7	Розробка компонентів web-застосунку.	17.11.2021р. – 07.12.2021р.	
8	Висновки та оформлення пояснювальної записки дипломної роботи.	08.12.2021р. – 14.12.2021р.	
9	Підписання необхідних документів у встановленому порядку.	15.12.2021р. – 17.12.2021р.	
10	Підготовка до захисту та попередній захист дипломної роботи на випусковій кафедрі.	20.12.2021р. – 27.12.2021р.	

7. Дата видачі завдання: 11.10.2021 р.

Керівник дипломної роботи _____ Аліна САВЧЕНКО
(підпис керівника) (П.І.Б.)

Завдання прийняла до виконання _____ Анна ТКАЧЕНКО
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Web-застосунок для управління послугами в сфері інформаційних технологій» складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел і містить 83 сторінки тексту та 41 рисунок. Список використаних джерел містить 32 найменування.

Метою розробка web-застосунку для управління послугами в сфері інформаційних технологій для оптимізації надання та отримання послуг організаціями та покращення робочих процесів за рахунок отриманих послуг.

Предметом дослідження є автоматизація процесу управління послугами за допомогою web-застосунку.

Об'єкт дослідження є процес отримання та надання послуг в сфері інформаційних технологій організаціям.

Методи дослідження, технічні та програмні засоби: порівняльний аналіз, розробка бази даних, клієнтських та серверних компонентів на основі React в середовищі WebStorm, обробка літературних джерел.

Отримані результати та їх новизна: розроблено web-застосунок для оптимізації отримання та надання послуг в сфері інформаційних технологій з урахуванням актуальних технологій розробки додатків.

Матеріали дипломної роботи можуть бути використані при розробці web-застосунків для управління послугами в межах однієї або багатьох організацій.

Ключові слова: УПРАВЛІННЯ ПОСЛУГАМИ, БАЗА ДАНИХ, СФЕРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ, MYSQL, REACT, JAVASCRIPT.

ЗМІСТ

ВСТУП

На сьогодні інформаційні технології є основою майже всіх підприємств та компаній сучасної епохи. Поступовий прогрес неможливий без допомоги компаній технічної підтримки. Більше того, підприємства та компанії, незалежно від того, великі вони чи малі, завжди шукають компанії з IT-продуктів та послуг, щоб допомогти їм розширити свої підприємства або впровадити нові функції для покращення робочого процесу.

Технічна підтримка та супроводження за допомогою залучення сторонніх фахівців забезпечує стабільну роботу інформаційної системи на підприємстві. Завдяки професійному IT-сервісу керівники фірм можуть розраховувати на ряд переваг:

- Немає необхідності витрачати час співробітників на розробку та підтримку працездатності IT-інфраструктури, що дозволяє звільнити ресурси на більш важливу діяльність компанії;
- Організація отримує якісне обслуговування від кваліфікованих та досвідчених спеціалістів та системних адміністраторів;
- Підвищується ефективність роботи інформаційного відділу підприємства, оскільки впровадження чіткої стратегії, розробленої спеціалістами, оптимізує бізнес-процеси;
- Весь об’єм послуг прописується в договорі, тому клієнт може здійснювати повний контроль над їх якістю.

Із залученням сторонніх кваліфікованих спеціалістів значно підвищується відмовостійкість IT-системи підприємства в цілому. Компанія має можливість скоротити час простоїв, пов’язаних з несправністю обладнання, а бізнес стає більш конкурентноздатним та гнучким.

Тому на сьогодні є досить актуальним процес автоматизації надання та отримання послуг в сфері інформаційних технологій.

Розділ 1. Аналітичний огляд і постановка задачі

Протягом останніх років відбувається активний розвиток сфери інформаційних технологій. Новітні можливості дозволяють підвищити ефективність бізнес-процесів багатьох організацій, оптимізувати роботу співробітників та налагодити робочі процеси в компаніях.

Всі користувачі ІТ-технологій та фірми, які працюють з обробкою цифрових даних повинні мати можливість звернутися до кваліфікованих та досвідчених співробітників для впровадження бажаних технологій в свої робочі процеси.

1.1. Основні види послуг в сфері інформаційних технологій

Послуги інформаційних технологій - це послуги, які дозволяють підприємствам отримувати доступ до технічних засобів та інформації, які вони використовують для своїх операційних процесів та повсякденних завдань. Часто команди працівників з досвідом роботи в галузі інформатики керують цими послугами для організацій у багатьох галузях промисловості.

Залежно від типу бізнесу організації, відділ, який відповідає за надання ІТ-послуг може складатися як із внутрішніх, так і із зовнішніх ІТ спеціалістів. Наприклад, у сфері медицини команда ІТ може надавати послуги, які забезпечують роботу баз даних лікарень та їх легкість у використанні для співробітників. В умовах малого бізнесу ІТ-послуги можуть включати облікові записи для зберігання даних для бізнесу та безпеку мережі для транзакцій через Інтернет.

.

Існує багато IT-послуг, які можуть принести користь бізнесу та допомогти їм працювати безперебійно та ефективно. Більшість з них дозволяє працівникам взаємодіяти з технологіями, які допомагають їм виконувати свої робочі обов'язки або спілкуватися один з одним.

Основні види IT-послуг, що використовуються в роботі багатьох організацій, включають:

1. Хмарні послуги – надають працівникам багато способів взаємодії з необхідними технологіями. Хмара - це підключена до Інтернету платформа, яка може зберігати, отримувати та надавати доступ до інформації і програм. Оскільки хмара зберігає інформацію, працівники можуть отримати до неї доступ та використовувати її вдома чи в офісі. Деякі хмарні служби можуть навіть віддалено запускати операційні системи, дозволяючи віддалено використовувати внутрішні бізнес-програми, не встановлюючи їх на комп'ютерах.

Хмарні сервіси дозволяють співробітникам приєднуватися до нарад і працювати над проектами з будь-якого місця, що особливо корисно, якщо в організації є працівники, які працюють віддалено [1].

За допомогою хмарних послуг працівники організацій можуть отримувати доступ до актуальної інформації з будь-якого пристрою, підключеного до мережі Інтернет і не хвилюватися за втрату важливих даних, оскільки вони зберігатимуться незалежно від пристроїв, що можуть вийти з ладу.

2. Голосовий протокол через Інтернет (VoIP) – це інструмент комунікації для бізнесу. VoIP дозволяє учасникам команди здійснювати дзвінки та надсилати повідомлення через Інтернет-з'єднання, а не через телефонну лінію. Це допомагає у внутрішньому міжміському зв'язку для підприємств, які можуть мати офіси в різних місцях. Більшість послуг VoIP вимагають платних підписок, які підприємства можуть використовувати із традиційними телефонними послугами або як альтернативу [2].

Відео конференції дуже схожі на дзвінки VoIP, за винятком того, що вони включають відео та аудіо. Підприємства, в яких працівники працюють віддалено або знаходяться за межами підприємства, значною мірою покладаються на цей

інструмент зв’язку. IT-компанії можуть встановити обладнання та програмне забезпечення, необхідне для здійснення дзвінків [2].

3. Рішення для резервного копіювання – захищають від втрати інформації, зберігаючи копії даних на зовнішньому обладнанні або онлайн -платформах, таких як хмарний сервіс.

Служби резервного копіювання інформації можуть допомогти захистити інформацію вашого бізнесу у разі виникнення будь-яких проблем, таких як відключення електроенергії або збій системи. Існує багато сервісів резервного копіювання інформації, таких як резервне копіювання файлів, резервне копіювання серверів і навіть резервне копіювання на робочому столі.

Впровадження резервного копіювання даних у хмарі може допомогти покращити захист даних організації без збільшення навантаження на персонал із інформаційних технологій. Вигода від економії праці може бути значною і достатньою для того, щоб компенсувати деякі додаткові витрати, пов’язані з хмарним резервним копіюванням, наприклад плата за передачу даних [3].

Підприємства можуть планувати автоматичне резервне копіювання своїх файлів, щоб зберегти найновіші версії файлів, які вони створюють, щоб забезпечити безпеку даних.

4. Безпека мережі – послуги мережевої безпеки допомагають захистити мережу бізнесу від несанкціонованого доступу. IT-служби можуть створювати та видавати дозвіл на доступ тим, хто цього потребує.

Якщо в процесі діяльності організації виявлено ризик виникнення певної атаки (наприклад, якщо ваша компанія працює з конфіденційною інформацією), варто отримати цю послугу для забезпечення надійного захисту даних [1].

IT-служби також надають багато інших послуг для захисту вашої мережі, включаючи:

- побудову брандмауерів;
- встановлення антивірусного програмного забезпечення;
- встановлення віртуальних приватних мереж (VPN);
- регулярні перевірки мережі.

5. Моніторинг – IT-служби можуть контролювати вашу мережу, інтернет-трафік, комп'ютери та інші пристрої з підтримкою Інтернету. Моніторинг дозволяє IT-командам відстежувати, хто звертається до інформації їхнього бізнесу, чи потрібна будь -яка технологія ремонту та чи потребують комп'ютери оновлення чи модифікації програмного забезпечення. По суті, ця IT-служба гарантує, що члени вашої команди мають необхідні інструменти та ресурси для виконання своїх завдань.

6. Поштові послуги – підприємства та організації покладаються на електронну пошту в багатьох комунікативних цілях. Від індивідуальних повідомлень та оновлень для всієї компанії до відносин з клієнтами, електронна пошта є важливою частиною підтримки ділових відносин.

IT-команди часто можуть наглядати за обліковими записами електронної пошти своїх компаній і давати рекомендації щодо постачальників, які відповідають потребам їх організації в спілкуванні [4].

Послуги електронної пошти IT також можуть бути корисними у процесах продажів та маркетингу, де IT-команди працюють із спеціалістами з продажу та маркетингу, підтримуючи відкриту комунікацію з клієнтами та лідерами продажів.

7. Інформаційна звітність – завдяки збору та моніторингу даних IT-служби можуть надавати звіти про інформацію про вашу організацію. Вони можуть створювати звіти на такі теми, як використання технологій та зберігання інформації, щоб допомогти вам зрозуміти, наскільки ефективні поточні послуги вашої компанії.

Звітність – це важлива складова робочого процесу для всіх членів команди, коли йдеться про оновлення проекту в цілому. Однак це також величезний плюс для керівників проекту, які хочуть бути впевненими, що проект просувається і завдання виконуються вчасно [5].

IT-послуги багатьох організацій пропонують програмне забезпечення, яке допомагає упорядкувати фінансову звітність та звітність про проекти. Служби інформаційної звітності також можуть упорядковувати інформацію таким чином, щоб допомогти вам краще зрозуміти свій бізнес та прийняти важливі рішення.

8. Дистанційна підтримка – багато IT-сервісів можуть надавати технічну підтримку віддалено. Наприклад, якщо на одному з ваших пристроїв виникає проблема, IT -служби можуть отримати доступ до нього через Інтернет для усунення несправностей та їх усунення.

Віддалена підтримка дозволяє їм швидше знаходити та виправляти проблеми та може застосовуватися до кількох пристроїв, включаючи більшість мобільних пристроїв.

Наявність IT-послуг з можливістю віддаленої підтримки може скоротити час на оновлення програм та виправлення технічних помилок, тому ваша робота може тривати з мінімальними перервами [6].

9. Програмне забезпечення як послуга (Saas) – відноситься до програм, які вимагають платної підписки. Це може включати програмне забезпечення для обробки текстів або бази даних.

Доступ до програмного забезпечення як послуги зазвичай здійснюється через web-браузер, користувачі входять у систему за допомогою імені користувача та пароля. Замість того, щоб кожен користувач встановлював програмне забезпечення на своєму комп’ютері, користувач може отримати доступ до програми через Інтернет [7].

IT-служби можуть з’єднати вас із потрібними службами та відстежувати з'єднання, щоб допомогти вам переконатися, що ви можете ним користуватися, коли вам це потрібно. Використання програмного забезпечення як послуги іноді включає доступ до хмари, тому ваша команда може працювати віддалено на власних пристроях у разі потреби. За більшості підписок SaaS підприємства отримують додаткові функції IT-послуг, включаючи технічну підтримку програмного забезпечення.

10. Розробка програмного забезпечення – IT-послуги часто можуть включати процеси розробки програмного забезпечення для створення власних додатків, що відповідають конкретним бізнес-потребам.

Добре розроблена стратегія розробки програмного забезпечення включає покроковий план дій від перевірки ідеї до запуску продукту [8].

Наприклад, починаючий бізнес може покладатися на свою команду IT-сервісів для створення програмного забезпечення для організації та ведення обліку капіталу. У таких випадках IT може надати послуги, необхідні для розробки персоналізованого програмного забезпечення та забезпечення життєздатності програми шляхом постійного тестування, розробки та обслуговування.

11. Усунення несправностей та технічна підтримка – IT-послуги для багатьох організацій також включають технічну підтримку та усунення несправностей для програмних програм, програм та онлайн -інструментів.

Команди, які пропонують ці послуги, часто проводять користувачів у процесі усунення технічних помилок або рекомендують подальші дії, такі як надсилання пристрою на ремонт.

Коли користувач звертається до технічної підтримки щодо помилки в текстовому процесорі, вони завжди можуть допомогти усунути діалогове вікно помилки, що з'являється. IT-спеціалісти також можуть допомогти зрозуміти, який плагін не працює і видалити його [9].

Завдяки спеціалізованій службі технічної підтримки ваша організація може отримати необхідну технічну допомогу, коли у співробітників виникають питання чи проблеми під час використання програмного забезпечення компанії.

12. Технологічна підготовка – оскільки технології та програмне забезпечення можуть часто змінюватися, допомога вашій команді навчитися користуватися оновленими пристроями та програмним забезпеченням може допомогти їм ефективно використовувати ці інструменти з мінімальними питаннями. Багато IT -послуг включають технологічне навчання для бізнес-команд, які використовують технічні інструменти та ресурси для повсякденної роботи. У цих випадках IT-команда може навчити персонал вашої компанії тому, як користуватися певними інструментами, програмами та додатками.

13. Установка та обслуговування обладнання – багато IT-послуг для бізнесу також включають установку обладнання, обслуговування та ремонт.

Послуги з технічного обслуговування та підтримки апаратного забезпечення – це профілактичні та відновлювальні послуги, які ремонтують або оптимізують

обладнання, включаючи технічне обслуговування за контрактом та ремонт за кожним випадком. Підтримка обладнання також включає в себе онлайн та телефонне технічне усунення несправностей і допомогу з налаштування, а також усі платні оновлення гарантії на обладнання [10].

Наприклад, корпоративний IT-відділ часто відповідає за встановлення комп'ютерів, жорстких дисків, принтерів, модемів та маршрутизаторів, необхідних для роботи компанії. IT-команди також оцінюють бізнес-процеси, щоб визначити типи обладнання, які відповідають його потребам. У багатьох підприємствах технічні групи також відповідають за обслуговування обладнання та оновлення комп'ютерів, модемів або маршрутизаторів для підтримки продуктивності та продуктивності.

Список послуг в сфері інформаційних технологій з часом тільки збільшується, і кожна послуга може ефективно вплинути на ту чи іншу сферу діяльності в організації. Тому на сьогодні є актуальною задачею надати можливість всім організаціям отримати будь-яку послугу від професійного спеціаліста для оптимізації власного бізнесу або створення нового продукту.

1.2. Огляд існуючих систем для управління послугами

Багато IT-компаній надають перелік можливих послуг у вигляді web-застосунків або додатків для мобільних телефонів та планшетів, оскільки це найбільш поширений спосіб пошуку та отримання інформації для більшості користувачів.

Одними з поширених прикладів є наступні системи:

1. web-сайт цифрової агенції Great Pro (рис. 1.1). Дана компанія пропонує послуги з розробки інтернет-магазинів, корпоративних сайтів, односторінкових сайтів, допомагає запустити онлайн-продажі товарів, розкрити та грамотно презентувати особливості вашої організації.

На сайті представлені приклади їх робіт та форма для зворотного зв’язку з клієнтами. Така система дозволяє організаціям звернутися за допомогою для розробки власного сайту або запуску власного бізнесу для продажу товарів.

Рис. 1.1. Головна сторінка web-сайту Great Pro

Посилання: https://pro.greatpro.com.ua/

2. web-сайт компанії IT Specialist Home (рис. 1.2) містить послуги, які включають забезпечення кібербезпеки інфраструктури організації та бізнес-додатків, моніторинг продуктивності бізнес-операцій і призначеного для користувача досвіду, інтеграцію бізнес-додатків у функціональні процеси компанії та багато інших.

На сайті представлені дипломи сертифікації компанії, контакти для зв’язку зі спеціалістами та форми заповнення заявок для клієнтів.

Рис. 1.2. Головна сторінка web-сайту IT Specialist Home

Посилання: https://www.my-itspecialist.com/uk/

3. мобільний додаток організації Poly Service (рис. 1.3) дозволяє отримати послуги з обслуговування та ремонту інформаційного обладнання, програмування систем управління документів для організацій, модернізацію та оптимізацію існуючих інформаційних процесів в компанії. При завантаженні додатку клієнт заповнює форму реєстрації та отримує доступ до замовлення основних послуг, які можуть надати IT-спеціалісти.

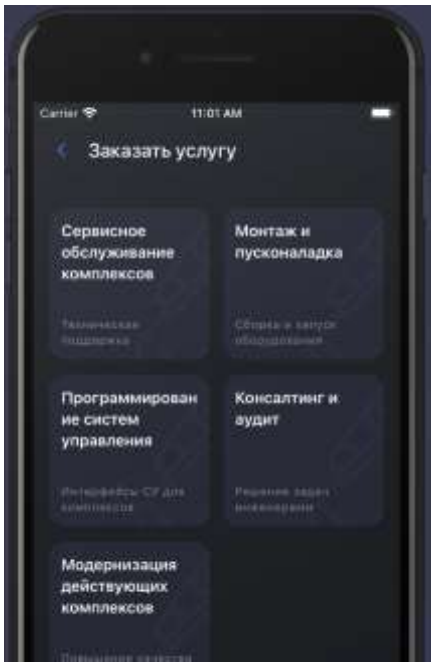

Рис. 1.3. Головна сторінка додатку Poly Service

Оформлення управління інформаційними послугами у вигляді web-сайтів або мобільних додатків є досить ефективним, оскільки це найбільш популярний спосіб пошуку необхідної інформації користувачами та забезпечення зручного обміну повідомленнями між замовником та клієнтом, тому в даній роботі розглянуто розробку системи для управління IT-послугами у вигляді web-застосунку.

1.3. Особливості використання web-застосунків для управління послугами

Web-застосунок – це програма з певним набором функціоналу, яка використовує браузер в якості клієнта. Іншими словами, якщо додаток для здійснення бізнес-логіки потребує мережевого з'єднання та наявності на стороні користувача браузера, то його відносять до web-застосунку.

У web-додатку клієнт (браузер) і сервер — це різні програми, які часто виконуються на різних комп'ютерах (і навіть у різних операційних системах). Отже, дві половини програми можуть спілкуватися, але зазвичай обмінюються лише невеликими об'ємами інформації [11].

Web-застосунки асоціюються з динамічно генерованим контентом в залежності від запиту користувача. Для роботи web-додатків необхідна клієнтська частина, тобто інтерфейс, представлення на екрані користувача, і серверна частина для обробки запитів та запису в базу даних отриманих відомостей. Це дозволяє користувачам взаємодіяти з компанією, використовуючи онлайн-форми, системи управління контентом та різні інтерактивні елементи.

Крім того, web-застосунки дозволяють обмінюватися інформацією, співпрацювати над проектами та працювати над загальними документами незалежно від місця розташування та пристрою.

Web-застосунки включають безліч онлайн-послуг та функцій. Вони надають можливість працювати з деякими найбільш часто використовуваними програмами, включаючи текстові редактори та електронні таблиці. Навіть такий простий процес,

як заповнення контактної форми на web-сайті, передбачає використання web-застосунку.

Використання web-застосунків в якості системи управління інформаційними послугами надає багато переваг як клієнтам, так і IT-спеціалістам:

- забезпечення можливості замовників швидко та легко знаходити інформацію про необхідні послуги на web-сайтах з великим об'ємом даних;
- забезпечення можливості IT-компаній збирати, зберігати та аналізувати дані, отримані від відвідувачів сайту. Довгий час використовувався метод, при якому дані, введені в форму, надсилалися для обробки спеціальним додаткам або спеціально призначеним співробітникам у вигляді повідомлень електронною поштою. Web-застосунок дозволяє зберігати дані безпосередньо в базі даних, а також отримувати дані та формувати звіти на основі отриманих даних для аналізу;
- забезпечення динамічного контенту на сторінці, що дозволяє компаніям керувати наповненням web-сторінки відповідно до оновлень інформаційних послуг. Web-застосунок звільняє дизайнера від рутинної роботи постійного оновлення сторінок сайту.

1.4. Види web-застосунків, принцип роботи

Web-застосунки зазвичай розробляються за допомогою мов програмування, які підтримуються браузерами, такими як JavaScript та HTML. Деякі сторінки web-застосунків є динамічними і потребують обробки на стороні сервера. Інші повністю статичні і не вимагають обробки на сервері.

Web-програма вимагає, щоб web-сервер керував запитами клієнта, сервер додатків виконував функціональну логіку відповідно до запиту користувача, включаючи обробку та збереження інформації в базі даних [12].

Типовий процес роботи web-застосунків виглядає наступним чином:

1. Користувач генерує запит до web-сервера через web-браузер, або через інтерфейс користувача програми;

2. Web сервер пересилає цей запит на відповідний сервер web-додатків;

3. Сервер web-додатків виконує необхідну логіку відповідно до отриманого запиту – наприклад, виконує запит до бази даних або обробку даних, потім генерує результат роботи;

4. Сервер web-додатків надсилає результати на web-сервер із необхідною інформацією або обробленими даними;

5. Web-сервер надає відповідь клієнту із потрібною інформацією, яка потім з'являється в інтерфейсі користувача.

Відповідно до функціональності та способу представлення інформації web-застосунки розділені на декілька категорій, кожна з яких має свої особливості роботи. Основними видами є статичні, динамічні, комерційні, портальні web-додатки та системи управління наповненням сайту [13].

Статичні web-застосунки – це найперший вид, який з’явився в мережі Інтернет. Як правило, він не має ніякої взаємодії між користувачем і сервером, розробляється за допомогою HTML і CSS для відображення лише відповідного контенту та даних.

Прості web-сайти зазвичай включають статичні сторінки, створені для інформаційних потреб [14].

В деяких випадках статичні web-застосунки містять GIF-файли або відео для залучення відвідувачів. Прикладом статичного web-застосунка може бути web-сайт портфоліо або web-сайт компанії, представлений у вигляді візитної картки.

Схема функціонування статичного web-застосунку представлена на рис. 1.4. Статичний web-сайт містить набір відповідних HTML-сторінок та файлів, розміщених на комп’ютері, на якому встановлений web-сервер.

Web-сервер – це програмне забезпечення, яке надає web-сторінки у відповідь на запити web-браузерів. Зазвичай запит на сторінку створюється під час натискання посилання на web-сторінці, вибору закладки в браузера або вводу URL-адреси в адресному рядку браузера.

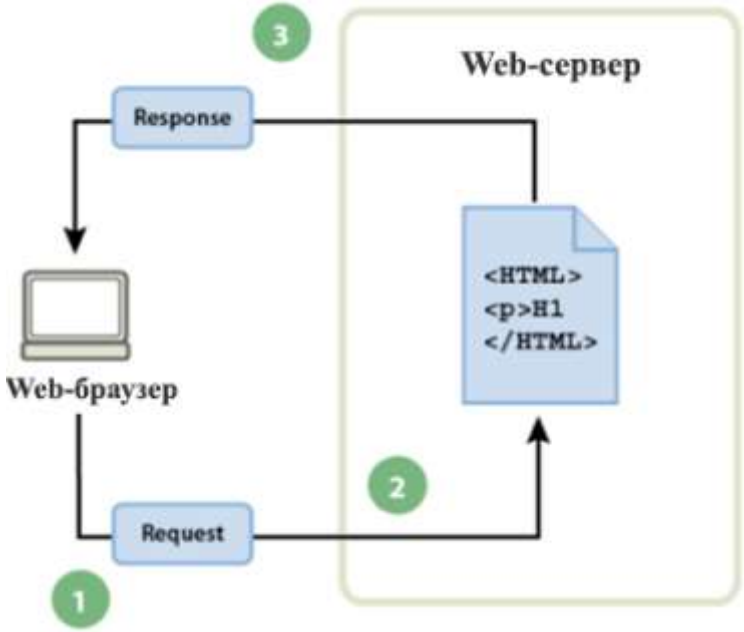

Рис. 1.4. Схема функціонування статичного web-застосунку

Кінцевий вміст статичної web-сторінки визначається розробником і залишається незмінним.

Весь HTML-код створюється розробником до того моменту, коли сторінка буде розміщена на сервері. Коли web-сервер отримує запит на видачу статичної сторінки, то після аналізу запиту сервер знаходить необхідну сторінку і надсилає її браузеру.

Динамічні web-застосунки – це оновлені статичні, оскільки їх складніше розробити з технічної точки зору. Їх основна мета – взаємодія з клієнтом. Вони мають різні інтерактивні елементи та методи залучення клієнтів до послуг або продуктів, які пропонує web-застосунок.

З появою Інтернету, особливо після популяризації блогів і соціальних мереж, потужність медіа-інтерактивності різко зросла і на сьогодні більшість організацій намагається залучати до участі користувачів на web-сторінках [15].

Такі web-застосунки використовують бази даних для збереження приватних та загальнодоступних даних, які відображаються на web-сторінці. Для управління частинами сервера та інтерфейсу такі програми зазвичай мають панель адміністратора, де адміністратори можуть змінювати вміст та додавати нові інтерактивні елементи.

Схема функціонування динамічного web-застосунку представлена на рис. 1.5. Коли web-сервер отримує запит на видачу статичної сторінки, він надсилає сторінку безпосередньо браузеру.

Проте, коли виконується запит на динамічну сторінку, дії web-серверу неоднозначні. Сервер передає сторінку спеціальній програмі, яка формує кінцеве наповнення. Така програма називається сервером застосунків [16].

Сервер застосунків виконує читання коду, який знаходиться на сторінці, формує кінцевий результат відповідно до прочитаного коду, а потім видаляє його зі сторінки. В результаті цих операцій отримується статична сторінка, яка передається серверу, який в свою чергу надсилає її клієнтському браузеру.

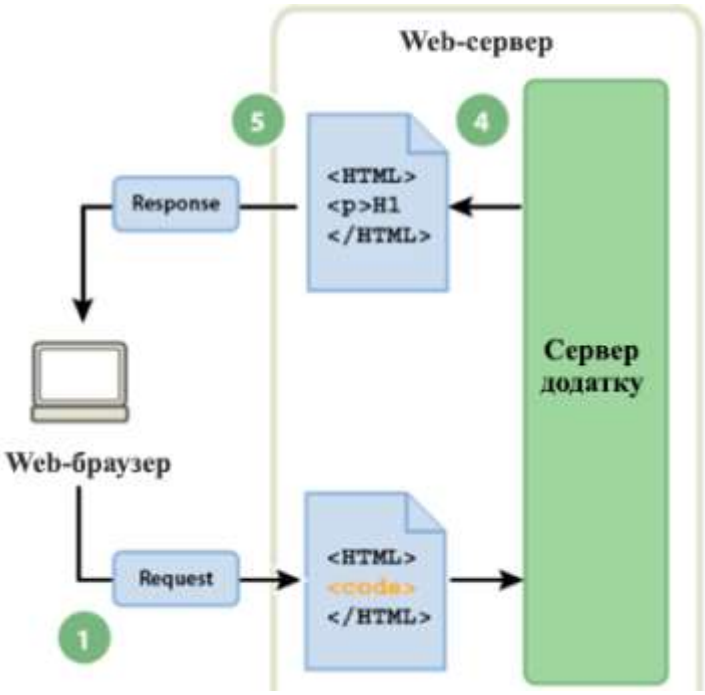

Рис. 1.5. Схема функціонування динамічного web-застосунку

Для розробки надійного динамічного web-застосунка використовується багато мов програмування. Найпоширеніші – PHP та ASP.NET. Прикладом такого додатка може бути практично будь-який web-застосунок, подібний до типового web-сайту цифрового агентства, де користувачу потрібно залишити запит [17].

Сервер динамічного web-застосунку також дозволяє працювати з серверними ресурсами, таким як бази даних. Наприклад, динамічна сторінка може виконати

запит на сервер для отримання інформації з бази даних та виведення її на HTML сторінку.

Використання сховища бази даних дозволяє відокремити дизайн сайту від його наповнення. Замість того, щоб створювати окремі HTML-сторінки для кожного окремного контенту, розробник створює один шаблон, який може обробляти різні дані, що надходять зі сховища. При кожному запиті клієнта відбувається оновлення контенту в шаблоні та відображення актуальної інформації.

Схема функціонування динамічного web-сайту з використанням бази даних представлена на рис. 1.6.

Web-браузер виконує запит на динамічну сторінку, яку знаходить web-server та надсилає до серверу додатку. Сервер додатку сканує сторінку та виявляє можливі запити, які надсилає до драйверу бази даних.

При виконанні певного запиту база даних повертає відповідну інформацію, яка додається до наповнення динамічної сторінки та відображається в браузері користувача.

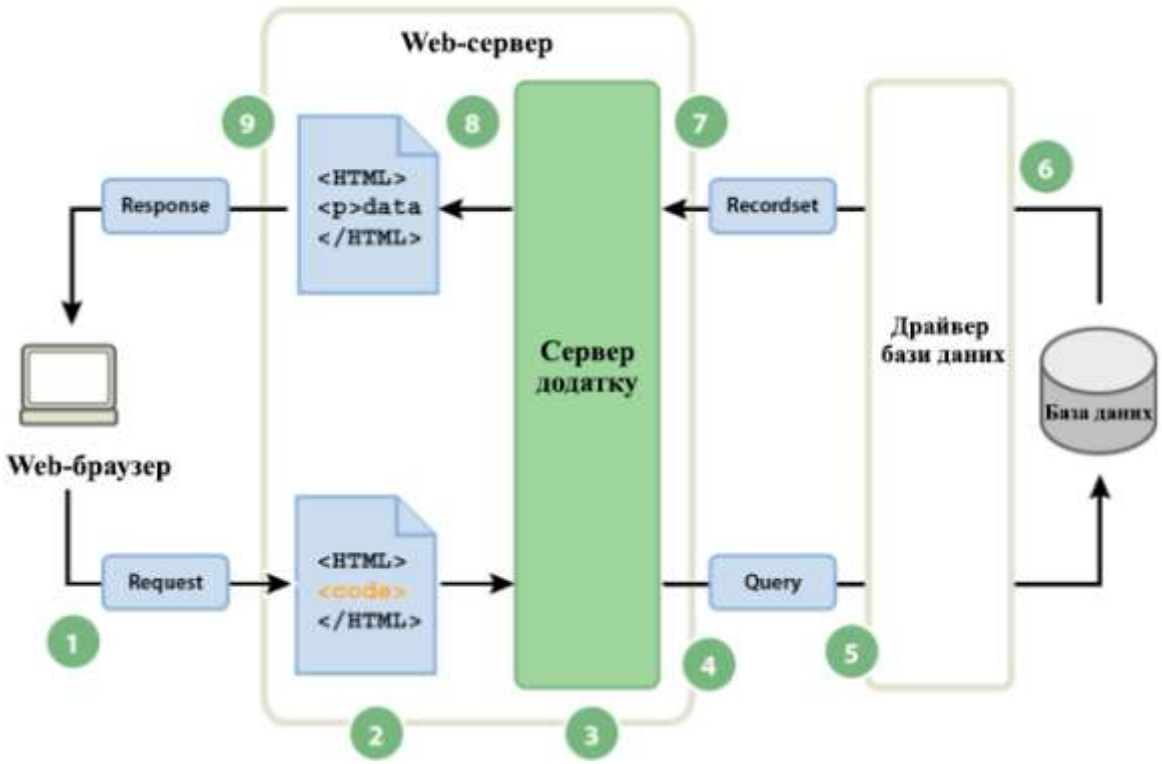

Рис. 1.6. Схема функціонування динамічного web-застосунку з використанням бази даних

Якщо web-застосунок безпосередньо рекламує продукти або послуги потенційним клієнтам, то він називається інтернет-магазин або web-застосунок електронної комерції [18].

Під електронною комерцією мається на увазі процес купівлі або продажу товарів чи послуг через Інтернет. Інтернет-магазини стають все більш популярними через швидкість і простоту використання для клієнтів.

Продажі в Інтернеті можуть допомогти бізнесу вийти на нові ринки та збільшити продажі і прибуток. Якщо організація зацікавлена в продажі товарів чи послуг іншим підприємствам, вона може використовувати Інтернет, щоб знайти потенційних клієнтів, оголосити тендери та пропонувати продукти для продажу (чи через власний web-сайт, або через сайт електронного ринку) [19].

Порівняно з динамічним web-додатком, даний тип вимагає більше функцій, оскільки клієнти повинні виконати певні дії, щоб придбати та отримати товари чи послуги.

Функції даного виду web-застосунків включають системи електронних платежів, панель управління для адміністратора (переважно для заповнення переліку продуктів, його оновлення або видалення товарів, а також для управління замовленнями клієнтів та обробки платежів) та особистий кабінет для користувача.

Web-додаток порталу відноситься до застосунків, які містять багато різних розділів або категорій, доступ до яких реалізований на домашній сторінці. Після входу користувача в систему функціонал порталу також дозволяє постачальнику послуг відстежувати активність користувачів на web-сайті. Прикладами такого типу додатків можуть бути форуми або чати.

Системи для управління контентом – це тип web-застосунка, де користувач не потребує допомоги технічної команди, оскільки він може самостійно змінювати наповнення web-сайту, не вивчаючи жодних мов програмування. Загальні особливості систем управління наповненням сайтом включають:

- створення контенту дозволяє користувачам легко додавати та форматувати вміст;

- зберігання інформації в одному місці, дотримуючись послідовності додавання даних користувачем;
- наявність багатьох робочих процесів та призначення ролей для керування вмістом, таких як автори, редактори та адміністратори;
- забезпечення публікації, упорядкування та розміщення контенту в реальному часі.

Прикладами систем управління контентом є WordPress, Joomla та Drupal.

На основі аналізу принципу роботи web-застосунків можна виділити наступні переваги їх використання:

- Забезпечення доступу з будь-якого пристрою: з web-застосунком можна працювати з будь-якої точки світу з комп'ютера, планшета або смартфона, підключеного до мережі Інтернет;
- Економія: web-застосунки працюють на всіх платформах і виключають необхідність розробки додатку окремо для різних операційних систем чи платформ;
- Адаптивність: якщо для більшості додатків необхідні визначені операційні системи, то для роботи з web-застосунком підходить будь-яка операційна система та будь-який браузер;
- Відсутність клієнтського програмного забезпечення: інсталяція, обслуговування та модернізація значно дешевші та швидкі, оновлення до останньої версії додатку відбувається при черговому завантаженні сторінки;
- Мережева безпека: web-система має єдину точку входу, яку можна централізовано налаштувати та забезпечити надійну безпеку;
- Масштабованість: зі збільшенням навантаження на систему не треба збільшувати потужність клієнтських місць, web-застосунок дозволяє обробляти більшу кількість даних, як правило, лише силами апаратних ресурсів, без необхідності зміни коду та архітектури;
- Захист від втрати даних: дані користувачів зберігаються в хмарному сховищі або в базі даних, за цілісність яких відповідають провайдери. Дані захищені

від втрати при будь-яких пошкодженнях апаратних складових пристрою користувача.

1.5. Постановка задачі

На сьогодні більшість бізнес-процесів в організаціях пов’язані з інформаційними технологіями, а саме з використанням різноманітних технічних пристроїв та обробки даних за допомогою новітніх технологій. Впровадження та підтримка зазначених пристроїв і процесів потребує певного рівня кваліфікації співробітників і в кожній організації може виникнути необхідність звернення до кваліфікованих спеціалістів.

Враховуючи наведене, актуальною є задача розробки системи управління послугами в сфері інформаційних технологій. Така система дозволить IT-компаніям розміщувати послуги, які вони можуть надати, а іншим організаціям забезпечить можливість комунікації з досвідченими працівниками у зазначеній сфері.

Основні можливості такої системи мають включати:

- реєстрацію IT-організацій (постачальників послуг);
- реєстрацію інших організацій (замовників послуг);
- розміщення замовлень на виконання послуг в сфері інформаційних технологій;
- перегляд існуючих замовлень відповідно до обраної IT-послуги з урахуванням технологій, які необхідно використати для їх виконання;
- перегляд зареєстрованих IT-спеціалістів та організацій з урахуванням основного стеку технологій, виконаних робіт та рейтингу на сайті;
- можливість надсилання запиту для виконання замовлення на визначену роботу.

Система дозволить багатьом організаціям отримувати бажані послуги від IT-експертів та ефективно витрачати час і ресурси на основні бізнес-процеси компанії.

Враховуючи наведене, метою виконання даної дипломної роботи є розробка web-застосунку для управління послугами в сфері інформаційних технологій для

оптимізації надання та отримання послуг організаціями та покращення робочих процесів за рахунок отриманих послуг.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- провести аналітичний огляд предметної області, визначити основні види послуг в сфері IT;
- провести аналіз існуючих рішень для управління послугами, визначити їх слабкі сторони;
- визначити технології та засоби для розробки web-застосунку для управління IT послугами;
- визначити архітектуру web-застосунку для управління послугами та необхідний функціонал;
- спроектувати базу даних додатку;
- розробити інтерфейс web-застосунку;
- розробити функціональну складову web-застосунку.

Висновки до розділу 1

За останні роки сфера інформаційних технологій невпинно розвивається і поширюється список можливих IT-послуг, які включають хмарні, мережеві, поштові послуги, ведення цифрової звітності, розробку програмного забезпечення та інші. Не всі організації мають відділ з кваліфікованими IT-спеціалістами для впровадження та підтримки функцій, які дозволяють оптимізувати та підвищити ефективність бізнес-процесів компанії.

В даній роботі розглянуто проектування та розробку системи управління послугами в сфері інформаційних технологій, яка дозволить всім організаціям отримувати бажані консультації та допомогу з впровадженням новітніх процесів за допомогою кваліфікованих спеціалістів.

Оскільки більшість користувачів шукає необхідну інформацію в мережі Інтернет, було вирішено розробити зазначену систему у вигляді web-застосунку, що забезпечить ефективну взаємодію між замовниками та постачальниками послуг.

На сьогодні деякі IT-організації розробили власні мобільні додатки або web-сайти для поширення інформації про послуги, які вони можуть надати іншим користувачам, проте ця інформація не є централізованою. В проектованій системі передбачається можливість реєстрації для багатьох компаній, що дозволить користувачам шукати та обирати бажані послуги в одному місці.

Розділ 2. Проектування web-застосунку, огляд технологій та інструментів для його розробки

На сьогодні розробка web-застосунків вимагає заходів, які включають розробку зрозумілого та естетичного макету додатку, опис взаємодії елементів системи з користувачем, визначення архітектурної структури та розробку функціональної складової і контенту.

Коли користувач вводить URL адресу у вікні браузера, вона надсилається до DNS-сервера, з якого повертається унікальна IP-адреса сервера, на якому зберігається необхідний web-застосунок. Раніше для web-додатків не розроблялася архітектура на стороні клієнта, браузери отримувати статичні файли з серверу і відображали їх користувачу. Проте з розвитком технологій web-браузери стали потужнішими, що призвело до того, що велика частина функціональних можливостей була перенесена на сторону клієнта.

Раніше для забезпечення інтерактивності на стороні клієнта широко використовувалися такі технології, як Adobe Flash 8 та Microsoft Silverlight 9. За останні роки вони застаріли і більшість web-браузерів їх не підтримує. Натомість стрімкого розвитку набули рішення з використанням мови програмування JavaScript, а фреймворки та бібліотеки на основі даної мови стали невід'ємними частинами web-інженерії.

Традиційно web-додатки складалися зі статичних HTML-сторінок, пов'язаних між собою гіперпосиланнями. При переході між посиланнями браузер отримував нову HTML-сторінку з web-сервера та відображав її, що робило програму передбачуваною.

Недоліком такого підходу є те, що відтворення нових HTML-сторінок негативно впливає на продуктивність роботи web-застосунку, оскільки вимагає повного повторного створення нового інтерфейсу для кожної внесеної зміни.

Для односторінкових додатків (SPA) існує лише один файл HTML, а частини сторінки динамічно відтворюються за допомогою JavaScript. Для оновлення вмісту дані витягуються динамічно з web-сервера або стороннього API [20].

Під час початкового завантаження SPA завантажується мінімізований код для фреймворку, тому початковий час завантаження трохи довший, ніж для традиційних web-сторінок. Використовуючи кешування в браузері та завантажуючи частини програми при необхідності, час завантаження SPA можна зменшити. Такий підхід до розробки був використаний для створення web-застосунку для управління послугами, описаного в даній роботі [20].

2.1. Архітектура web-застосунків

Архітектура web-застосунків включає компоненти (рис. 2.1), які відповідають за клієнтську та серверну сторони:

- компоненти клієнтської сторони створюються за допомогою JavaScript, HTML, CSS та відповідають за користувацький інтерфейс: інформаційні панелі, сповіщення, журнали активності та інші елементи. Web-браузер запускає код та перетворює його в інтерфейс. Налаштування операційної системи або інших складових пристрою користувача не потрібні;
- серверні компоненти створюються за допомогою Java, .Net, NodeJs, Python та інших мов програмування. Сервер складається з двох частин: логіки застосунку та бази даних. Логіка - це операційний центр web-застосунку. База даних містить всю необхідну для додатку інформацію (наприклад, облікові дані для входу користувачів в систему).

Рис. 2.1. Складові компоненти архітектури web-застосунків

При інтеграції всіх компонентів та формуванні web-застосунку відбувається створення наступних 4 рівнів:

- рівня представлення даних;
- рівня обслуговування даних;
- рівня бізнес логіки;
- рівня доступу до даних.

Рівень представлення даних відображає інтерфейс для користувачів та робить взаємодію з компонентами більш простою. Цей рівень архітектури web-застосунків містить компоненти інтерфейсу, які обробляють та відображають дані додатку, а також компоненти, які встановлюють та обробляють дії користувача з елементами сторінки.

Основна мета рівня представлення - отримати вхідні дані, обробити запити користувачів, надіслати їх до рівня обслуговування даних та відобразити відповідний результат на сторінці [21].

Даний рівень доступний через браузер та компоненти інтерфейсу, які взаємодіють з іншими рівнями. Основні технології, які використовуються на даному рівні: JavaScript, HTML, CSS. HTML скрипти відповідають за наповнення та тему додатку, CSS відповідає за його зовнішній вигляд. Відповідь на запити користувачів при взаємодії з елементами інтерфейсу та компоненти наступних рівнів генеруються

за допомогою інструментів мови програмування JavaScript та пов’язаних з нею фреймворків.

Рівень обслуговування отримує дані від попереднього рівня, передає на рівень бізнес-логіки для обробки та отриманий результат повертає до рівня презентації. Передаючи дані, якими оперує рівень бізнес-логіки, рівень обслуговування захищає інформацію web-застосунку, оскільки він ізолює бізнес-логіку від клієнтської сторони.

Рівень бізнес-логіки відповідає за правильний обмін даними. Цей рівень визначає логіку бізнес-операцій і правил та відповідає за завершення обробки запитів клієнта з браузера, визначення шляхів доступу до даних додатку [21].

Прикладом роботи рівня бізнес-логіки є процес обробки входу в систему web-застосунку. Напрямки, за допомогою яких серверна частина отримує дані та клієнтські запити формуються саме на рівні бізнес-логіки.

У web-застосунку для управління послугами в сфері інформаційних технологій даний рівень визначає послідовність дій, які будуть виконувати компанії або IT-фахівці для реєстрації та авторизації в системі а також для розміщення або надсилання запиту на виконання певного виду послуг.

Рівень доступу до даних пропонує спрощений варіант отримання інформації, що зберігається в таких сховищах, як двійкові або XML файли. Даний рівень також відповідає за управління операціями CRUD: створення, читання, оновлення, видалення даних і має назву рівня збереження або сховища, який об’єднується з рівнем бізнес-логіки. Таким чином, рівень бізнес-логіки отримує інформацію про те, до якої бази даних необхідно застосовувати зміни та як оптимізувати отриману інформацію.

Правильний розподіл основного функціоналу web-застосунку між відповідними рівнями дозволяє призначити кожному компоненту відповідальність за виконання окремого сервісу, що покращує інтегрованість та полегшує пошук помилок при розробці та використанні додатку [21].

2.2. Технології для розробки клієнтської складової

Для розробки клієнтської складової web-застосунку для управління послугами в сфері інформаційних технологій були використані такі технології як HTML, CSS та бібліотека React, побудована на основі мови програмування JavaScript.

2.2.1. HTML

HTML (HyperText Markup Language) – мова розмітки гіпертексту, яка використовується при створенні основної структури сторінок web-застосунку. Елементи сторінки описуються за допомогою тегів: назва тегу відповідає за призначення та відображення елементу, текст всередині тегу відповідає за наповнення, контент елементу.

Обов'язковими тегами в HTML документі є <html></html>, <head></head> і <body></body> [23]. Атрибут html є обгорткою для структури сторінки, весь контент має бути описаний всередині нього. Атрибут head містить службову інформацію про сторінку, наприклад, назву вкладки в web-браузері, посилання на зовнішні файли, які необхідно підключити до сторінки, дані про розробника, початкові конфігурації для зовнішнього вигляду сторінки. В атрибуті body розміщується контент сторінки, всі елементи, які відображаються користувачу.

Найбільш поширеними HTML тегами є:

- - для відображення параграфів тексту;
- - елемент структури, що відповідає за контейнер;
- - використовуються для обертання частин тексту;
- <table></table> - для відображення таблиць;
- , - для відображення списків та їх елементів;
- <a> - для додавання посилань на сторінку;
- <form></form> - для додавання форми;
- <input /> - для відображення полів форми.

При розробці HTML файлів важливо використовувати семантику – правила для оптимізації елементів сторінки для браузерів. HTML семантика дозволяє створювати інтуїтивно зрозумілі елементи та допомагає сервісам для розпізнавання елементів на сторінках досконало відтворювати структуру сайту для людей з обмеженими можливостями.

Основними семантичними HTML тегами є:

- <header></header> - містить шапку сторінки, в якій зазвичай розміщується логотип та навігаційне меню;
- <nav></nav> - використовується для навігаційного меню сайту;
- <section></section> - використовується для основних секцій на сторінці;
- <article></article> - використовується для відокремлення основних статей в секціях;
- <footer></footer> - містить футер сторінки, в якому зазвичай містяться посилання на контакти розробника або компанії, інформація про права на сторінку.

Для визначення додаткової інформації для кожного тегу можна додавати атрибути, наприклад:

- class – для визначення класу елементу для подальшої стилізації;
- id – унікальний ідентифікатор елементу;
- href – атрибут для посилання, містить ресурс, на який необхідно перейти;
- value – для визначення початкового значення полів форми.

Всі зазначені теги відповідають за опис загальної структури та розміщення елементів на сторінці. Для оформлення зовнішнього вигляду кожного блоку використовується CSS.

2.2.2. CSS

CSS (Cascading Style Sheets) – таблиці стилів, які використовуються для стилізації елементів на web-сторінці. Для застосування стилів до елементу використовуються різні типи селекторів, які включають:

- універсальний селектор - для призначення стилів для всіх елементів HTML сторінки;
- селектор тегів – для призначення стилів для елементів за визначеним тегом;
- селектор класів – для призначення стилів для елементів з певним значенням атрибуту class;
- селектор ідентифікаторів – для призначення стилів для елементів з певним значенням атрибуту id;
- селектор атрибутів – для призначення стилів для елементів з визначеними атрибутами;
- селектор дочірніх елементів – для призначення стилів для дочірніх елементів певних блоків.

Кожен селектор має свою специфічність, яка допомагає вирішувати проблеми при створенні конфліктів. Проблеми можуть виникати, коли до одного елемента застосовується декілька селекторів і необхідно визначити, які саме стилі необхідно призначити [23].

Також для стилізації елементів використовуються псевдо класи та псевдо елементи. Стилізація за допомогою псевдо класів призначає стилі елементам при певному стані: при наведенні, переході за посиланням, при кліку та інших подіях. Стилізація псевдо елементів відповідає за зовнішній вигляд частин інтерфейсу: першої літери або лінії параграфу, додаванні та стилізації елементів перед або після визначених блоків.

Таблиці стилів допомагають розробити досконалий інтерфейс для користувачів, що допомагає збільшити кількість користувачів на сайті та покращити розташування елементів на сторінці.

Після створення структури HTML сторінки та написання необхідних стилів необхідно забезпечити інтерактивність: можливість користувачів взаємодіяти з елементами, наприклад, відкриття нового вікна при натисканні на певний блок. Для цього використовується мова програмування JavaScript.

2.2.3. JavaScript

JavaScript - це високорівнева мова програмування, яка спочатку використовувалася для написання невеликих скриптів у web-браузері для додавання логіки на web-сторінках. Сьогодні JavaScript використовується для розробки складних web-застосунків та мобільних додатків.

Скрипти, написані мовою Javascript, розроблені для безпосередньої взаємодії з існуючим об’єктом або системою і не потребують явного кроку компіляції. При запуску JavaScript в браузері, він взаємодіє з об’єктною моделлю документа (DOM), яка представлена у вигляді дерева, вузлами якого є елементи HTML сторінок.

JavaScript — це мультипарадигмальна мова програмування, яка підтримує керування різними подіями, функціональний та імперативний стилі програмування [24]. JavaScript має вбудовані функції, необхідні для функціонального програмування, оскільки він підтримує передачу функцій як аргументів до інших функцій, призначаючи їх змінним і зберігаючи їх у структурах даних. Функції вищого порядку є важливою частиною програмування, які дотримуються двох правил: вони приймають одну або кілька функцій як аргументи і повертають функцію як результат. JavaScript має кілька вбудованих функцій вищого порядку.

JavaScript відповідає стандарту ECMAScript, створеному міжнародною організацією Ecma. Перша специфікація ECMAScript була випущена в 1997 році, і з того часу було опубліковано кілька оновлених версій. JavaScript залишається найпопулярнішою реалізацією ECMAScript з моменту його заснування. Окрім сумісності з ECMAScript, JavaScript надає додаткову функціональність. У червні 2015 року була опублікована велика нова специфікація з оновленим та доповненим функціоналом під назвою ECMAScript2015 (також називається ES6) [24].

Специфікація включає два основних оновлення для створення більш складних програм, включаючи синтаксичне доповнення для декларацій класів і спільного використання модулів. JavaScript є об'єктно-орієнтованою мовою на основі прототипів, але з ES6 було додано альтернативний і більш звичний спосіб створення об'єктів і вирішення проблем успадкування.

Node.js, також відомий як Node, є простим і ефективним середовищем виконання для запуску JavaScript поза браузером за допомогою движка V8.

Node часто використовується як середовище виконання на стороні сервера, але також використовується на стороні клієнта для створення web-додатків JavaScript. Node містить менеджер пакетів (NPM), який використовується для налаштування програми Node та управління залежностями в програмі [25].

Менеджер пакетів налаштовується в каталозі проекту за допомогою файлу конфігурації package.json. Використовуючи NPM, розробники мають доступ до великої бібліотеки сторонніх пакетів з відкритим кодом. Ім’я пакета додається до конфігураційного файлу, а вихідний код пакета завантажується з бібліотеки NPM і встановлюється локально в папку під назвою node_modules. Поле «скрипти» в package.json використовується для вказівки певних операцій, зазвичай воно містить сценарій запуску, сценарій збірки та тестовий сценарій.

При розробці web-застосунків частіше використовують бібліотеки та фреймворки, розроблені на основі мови JavaScript, які надають рішення для прискорення створення додатків.

Фреймворк часто є складнішим, ніж бібліотека, а архітектура програми заздалегідь визначена. Розробник повинен займатися лише впровадженням функціональних можливостей домену. Недоліками використання фреймворку є суворі правила та шаблони, яких необхідно дотримуватися, що негативно впливає на розмір програми. Бібліотеки, з іншого боку, пропонують функціональність, яку розробники можуть використовувати за необхідності, і дозволяє самостійно контролювати архітектуру. Це призводить до створення програми, яка містить лише необхідну кількість функцій.

Існує багато бібліотек і фреймворків, які допоможуть розробнику під час розробки складних web-додатків. Кількість завантажень фреймворків на сьогодні представлена на рис. 2.2.

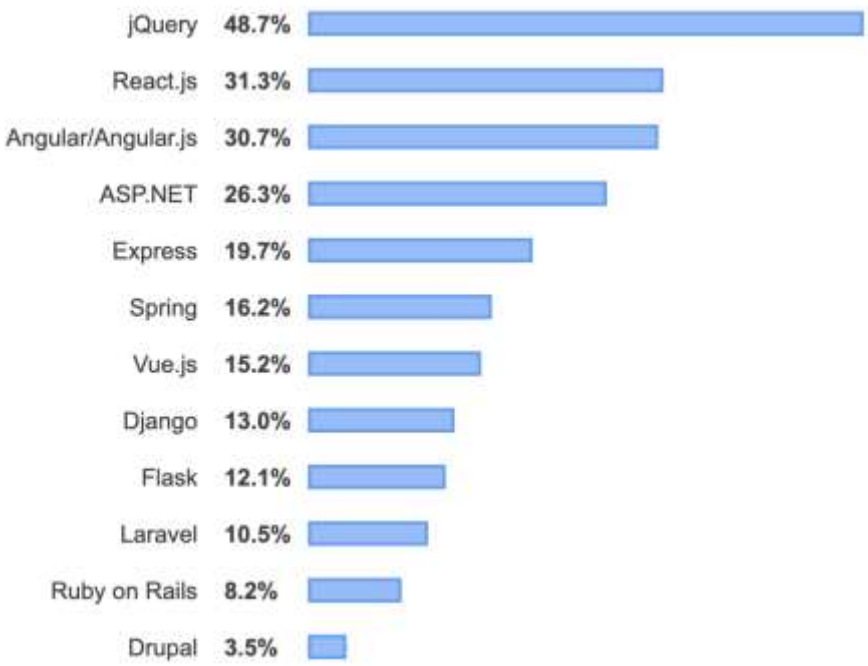

Рис. 2.2. Кількість завантажень фреймворків в процентному співвідношенні

На даний момент найпопулярнішими фреймворками та бібліотеками є React, Angular і Vue. За кількістю завантажень за допомогою NPM найпопулярнішою бібліотекою є React, саме вона використовується для розробки web-застосунку, описаного в даній роботі.

2.2.4. React

React — це бібліотека JavaScript для побудови користувацьких інтерфейсів, її можна використовувати для розробки окремих частин сайту або для створення користувацьких інтерфейсів для цілих web-додатків. React не встановлює жодного конкретного архітектурного шаблону для управління даними, він зосереджений виключно на створенні складних інтерфейсів користувача з невеликих будівельних блоків, які називаються компонентами [26].

React компоненти

Кожен компонент складається з методу візуалізації, який повертає опис того, що потрібно відтворити, і може бути представлений у вигляді HTML або інших компонентів React. Опис того, що потрібно відтворити, використовує синтаксис під назвою JSX, який є комбінацією HTML і JavaScript. Компонент можна визначити як клас або функцію. Важливими складовими компонентів є стан (state) і властивості (props).

Стан компонента - це об'єкт, який дозволяє зберігати значення змінних та може змінюватися на основі взаємодії користувача з елементами на сторінці або інших дій у програмі. Об'єкт стану не є обов'язковим, компоненти, що мають стан, називаються компонентами з визначенням стану, тоді як компоненти, які не мають стану, називаються презентаційними компонентами [26].

Властивості компонента — це незмінні дані, які передаються компоненту під час створення сторінки від батьківського елементу. За допомогою об'єкту властивостей компоненти React є більш гнучкими та придатними для повторного використання, оскільки один компонент може вести себе та виглядати по-різному в залежності від змінних, які отримує компонент.

В React дані передаються від батьківських компонентів до дочірніх за допомогою змінної props. У великих програмах React дерево компонентів стає глибоким, що призводить до необхідності передавати функції зворотного виклику та інші дані на кілька рівнів.

В Reacti є два види основних види компонентів: класові та функціональні [26].

Функціональний компонент — це чиста функція, яка приймає властивості (props) як вхідні дані і повертає елемент у вигляді JSX синтаксису. Приклад функціонального компоненту представлений на рис. 2.3.

```
1 const Greeting = (props) => {
2   return <h1>Hello, {props.name}</h1>
3 }
```

Рис. 2.3. Приклад функціонального компоненту

Компонент на основі класу створюється шляхом розширення класу React.Component. Стан компонента на основі класу оновлюється за допомогою методу setState() і зчитується за допомогою this.state всередині класу. Використання методу setState() запускає оновлення відображення компонента, що дозволяє відображати на сторінці актуальну інформацію для користувача. Приклад класового компоненту представлений на рис. 2.4.

```
class Greeting extends React.Component {
  render() {
      return <h1>Hello, {this.props.name}</h1>
  }
}
```

Рис. 2.4. Приклад класового компоненту

Компоненти на основі класів включають методи життєвого циклу, які можна використовувати для створення більш складної поведінки [26].

Методи життєвого циклу компонентів

Методи життєвого циклу — це вбудовані методи, які викликаються, коли стан або властивість компонента оновлюються, до або після візуалізації компонента або коли компонент зникає зі сторінки. Огляд методів життєвого циклу компонента на основі класів показано на рис. 2.5.

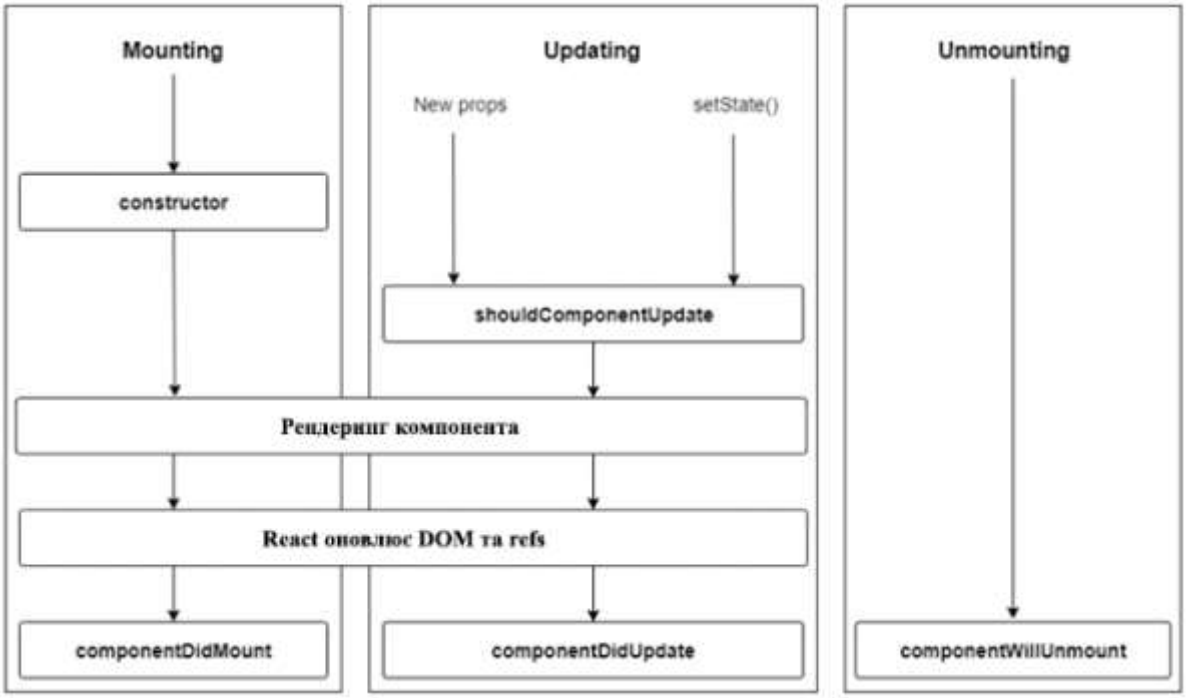

Рис. 2.5. Схема методів життєвого циклу компонентів

Метод componentDidMount() виконується після першого відображення компонента на сторінці. Зазвичай в цьому методі прописується логіка створення таймерів, виконання підписок на інші функції, завантаження початкових даних із серверу [27].

Метод componentDidUpdate() виконується при кожному оновленні компонента. В ньому прописується логіка для оновлення даних, які відображаються в компоненті.

Метод componentWillUnmount() виконується після видалення компонента зі сторінки. В ньому відбуваються дії, протилежні до тих, що були виконані в componentDidMount(): скасування підписок, видалення таймерів та інше.

Метод shouldComponentUpdate() приймає значення наступних властивостей та стану, і в залежності від отриманих значень може повернути true або false. В залежності від повернутого значення компонент перемальовується або залишається незмінним [27].

Методи життєвого циклу допомагають управляти відображенням компонента на різних стадіях та полегшують обробку даних, необхідних для визначення контенту, що буде відображатися на web-сторінці.

React Hooks

Щоб надати функціональному компоненту можливість використання стану та методів життєвого циклу, як для класових компонентів, використовуються React Hooks. Використання функціональних компонентів з React Hooks може зменшити розмір і складність програми React у порівнянні з компонентами на основі класів.

React Hooks використовуються для більш складних функціональних компонентів, вони «підключаються» до функцій React. React Hooks відповідають умовам найменування, які починаються зі слова use [28].

Стан не існує у функціональному компоненті за замовчуванням, замість цього можна використовувати хук React, який має назву useState. Він зберігає стан протягом усього існування компонентів. Хук useState і всі інші хуки можна використовувати більше одного разу в одному компоненті. Функціональний компонент, що використовує хук useState, представлений на рис. 2.6.

```
const Counter = () => {
  const [count,setCount] = useState(0);
  return (
    <div>
      <p>{count}</p>
      <button onClick={()=>setCount(count+1)}>
        Increment
      </button>
    </div>
  )
}
```

Рис. 2.6. Приклад функціонального компоненту з використанням useState

Вбудовані методи життєвого циклу не існують для функціональних компонентів, але їх можна створити за допомогою хука useEffect. За замовчуванням хук useEffect буде відтворювати роботу методу для кожного повторного відтворення компонента, але його можна налаштувати так, щоб він виконувався лише для певних змін. Приклад хука useEffect представлений на рис. 2.7.

```
const Counter = () => {
  const [count,setCount] = useState(0);
  useEffect(() => {
    docoument.title = 'Count: ${count}'
  })
  ...
}
```

Рис. 2.7. Приклад використання хука useEffect

Для обміну даними між компонентами в React використовується вбудований функціонал Context API. Контекст можна спільно використовувати між кількома компонентами і ініціалізувати за допомогою методу createContext, доступного в бібліотеці React. Щоб додати дані до контексту та зробити їх доступними для дочірніх компонентів, використовується провайдер, доступний в екземплярі Context [28].

Провайдер — це компонент, який приймає одне значення властивості, де вказуються дані для контексту, які можуть бути змінними, функціями або об’єктами. Всередині компонента Provider можна отримати доступ до даних контексту з будь-якого дочірнього компонента, використовуючи також Consumer, доступний у екземплярі Context. Натомість додаток не обмежується одним екземпляром контексту, для різних цілей використовується кілька різних контекстів.

Передача функцій зворотного виклику дочірнім компонентам за допомогою Context API дозволяє дочірньому компоненту змінювати стан батьківського компонента. Згідно з офіційною документацією, Context використовується для обміну даними, які вважаються глобальними для дерева компонентів React. Використання контексту може зменшити можливість повторного використання компонентів, оскільки вони звертаються до даних, отриманих через контекст від іншого компонента.

Для функціонального компонента, загорнутого в постачальник контексту, useContext діє як провайдер, і дані можуть бути доступні. Приклад використання хука useContext представлений на рис. 2.8.

```
const PersonsContext = React.createContext();
<PersonsContext.Provider value={{persons: persons}}>
    <Greetings />
</PersonsContext.Provider>

const Greetings = () => {
  const { persons } = useContext(PersonsContext);
  return persons.map(person =>
    <h1>Hello, {person.name}</h1>
  )
}
```

Рис. 2.8. Приклад використання хука useContext

Окрім вбудованих хуків, можна створювати користувацькі, які дозволяють повторно використовувати функціональні можливості між компонентами.

Ще однією перевагою використання React є використання Virtual DOM.

Віртуальне DOM-дерево

Віртуальний DOM - це полегшена версія реального DOM-дерева web-застосунку. При зміні певного компонента в додатку, React порівнює реальний DOM з віртуальним, та перемальовує на сторінці лише ті компоненти, в яких відбулись зміни. React обчислює мінімальний набір мутацій DOM, необхідних для того, щоб справжня DOM виглядала як віртуальна. Схема порівняння представлена на рис. 2.9.

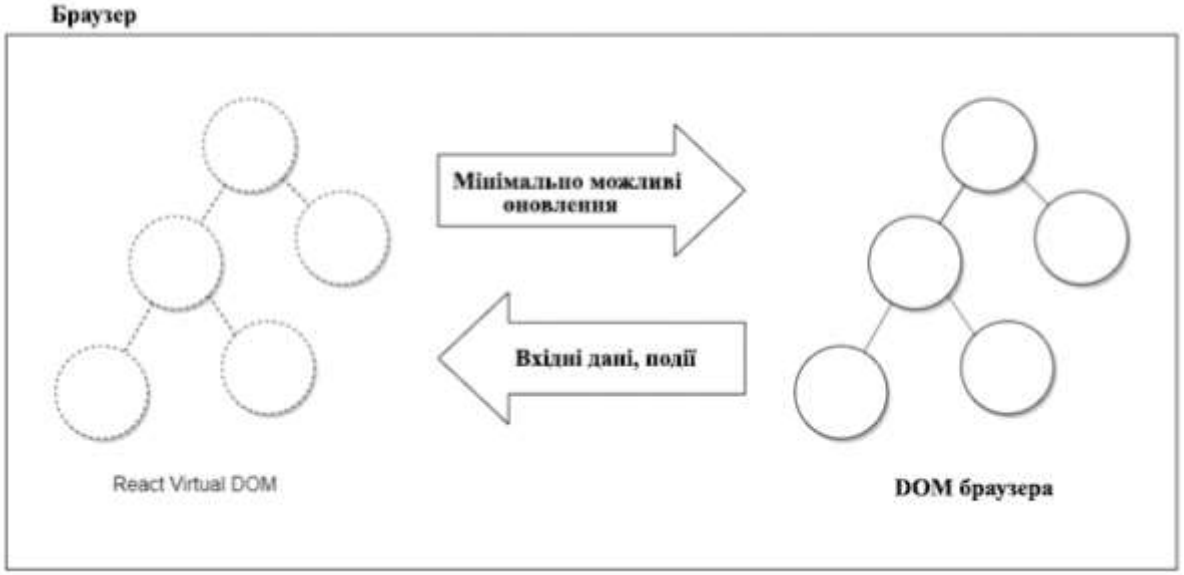

Рис. 2.9. Схема порівняння реального DOM-дерева з віртуальним

Маніпуляції з DOM виконуються за допомогою JavaScript, розміщуються в чергу і виконуються пакетно, щоб заощадити час. Розробники використовують JSX, щоб вирішувати, як компоненти мають відображатися в різних станах, а React дбає про маніпулювання DOM, щоб дістатися до різних станів.

Бібліотека React включає компілятор, який обробляє декларативний синтаксис JSX в Javascript, який може інтерпретувати браузер. На рис. 2.10 представлений код JSX перед компіляцією, а на рис. 2.11 код, який виконує браузер. Віртуальний DOM не покладається на браузер і тому може використовуватися також і для відтворення на стороні сервера.

```
<div>
  <Greeting name="John" />
</div>
```

Рис. 2.10. Код JSX перед компіляцією

```
React.createElement("div",null,
  React.createElement(Greeting, {name: "John"})
)
```

Рис. 2.11. Код, який виконує web-браузер

Інструмент create-react-app

React надає інструмент під назвою create-react-app, який налаштовує програму React з нуля за допомогою Node. Він включає в себе корисні інструменти, які прискорюють розробку, наприклад, перезавантаження та налагодження, менеджер пакетів, комплектувальник і компілятор [29].

Необхідними умовами для використання create-react-app є встановлений та налаштований Node і менеджер пакетів. Середовище налаштовується за допомогою команди npm init react-app <application-name> з використанням версії NPM версії 6 або вище.

Механізм JavaScript V8 не може обробляти синтаксис, такий як JSX, тому потрібен компілятор, який перетворює JSX на звичайний JavaScript код, який можуть зрозуміти Node і всі web-браузери. Набір інструментів create-react-app включає компілятор під назвою Babel. Babel-core містить необхідну функціональність для його роботи і функції presets, які використовуються для компіляції різних видів JavaScript, preset-env використовується для компіляції сучасного ECMAscript, а preset-react використовується для компіляції коду JSX.

Інструмент create-react-app включає пакет, який об’єднує та мінімізує всі активи програми, як модулі вузлів і необхідні файли JavaScript під назвою Webpack. Webpack налаштовується за допомогою файлу webpack.config.js, де вказано точку входу файлів, які мають бути з’єднані, і куди мають бути виведені з’єднані файли [29].

Крім того, сервер розробки, який розміщує програму локально, також налаштований у файлі конфігурації, який використовує бібліотеку web-packdevserver. Для додаткової функціональності можна використовувати плагіни Webpack. Один плагін, який використовується в додатку create-react, називається Hot Module Replacement, який запускає перебудову модулів, коли вносяться зміни у вихідний код.

У файлі конфігурації package.json присутній сценарій, який запускає сервер розробки webpack і прослідковує зміни. За допомогою сценарію запуску програма розміщується локально за допомогою порту, визначеного в конфігурації webpack, і його можна переглянути у браузері за адресою http://localhost:<порт>. package.json також містить сценарій збірки, який об'єднує та мінімізує вихідні файли програми в папку збірки, готову до розміщення на web-сервері для виробництва. Третій доступний сценарій – це тестовий сценарій, який запускає тестовий запуск, що виконує пошук файлів з розширенням .test.js.

Згенерований додаток на React містить наступні директорії:

- node_modules: містить всі необхідні залежності для розробки додатків React;

- public: директорія, що обробляє статичні файли в програмі, включаючи index.html, що містить тег , де відображається програма React. Крім того, він містить посилання на піктограму програми, корисні метадані, назву програми, а також посилання на файл manifest.json, розташований у загальнодоступній папці. Файл manifest.json – це файл JSON, який містить інформацію про те, як він повинен поводитися під час встановлення на мобільний телефон як прогресивної web-програми;
- src: директорія, що містить вихідний код, включаючи файли CSS і Javascript. Точка входу, яка відображає програму в DOM, називається index.js.

В розробленому web-застосунку всі файли розташовані відповідно до описаної структури, що дозволяє оперативно знаходити необхідні файли та налаштувати оптимальну взаємодію між компонентами додатку.

Клієнтський інтерфейс розробленого проекту створений на основі фреймворку React, який зазвичай запускається в середовищі Node.js.

2.3. Технології для розробки серверної складової

Для розробки серверної складової web-застосунку використано Node.js - середовище виконання на стороні сервера, побудоване на основі движку JavaScript V8, який адаптує функціональні можливості та синтаксис JavaScript для розробки серверної частини (backend).

Поєднання React.js з Node.js дозволяє розробникам створювати ізоморфні програми; зв'язок між клієнтською та серверною сторонами web-застосунку є швидким та ефективним.

Node.js вважається ідеальним бекенд-партнером React.js через його чудові сумісні функції. І інтерфейсні, і серверні бібліотеки використовують одну мову JavaScript, що дозволяє розробляти сучасні додатки.

Особливості використання Node.js включають:

1. Масштабованість — Node.js має ряд вбудованих функцій, які роблять його потужним інструментом для роботи з будь-якою кількістю користувачів і запитів;

2. Швидкість — Node.js повністю заснований на JavaScript, що робить його легким і швидким при обробці даних та масштабованості додатку;

3. Гнучкість — Node.js має ряд вбудованих API, які надають величезну кількість функціональних можливостей;

4. Крос-платформність — Node.js підтримує iOS, Android і web. Це робить його потужним інструментом для створення однієї програми з підтримкою декількох платформ.

Комбінація React та Node.js надає наступні переваги при розробці web-застосунків:

- можливість створення універсальних програм з JS-кодом на клієнті та сервері;
- створення оптимізованих для SEO застосунків, які використовують відтворення переглядів додатків не стороні сервера;
- використання одного механізму JavaScript V8 для візуалізації на стороні клієнта та сервера;
- використання компонентів React DOM, спеціально розроблених для роботи з Node.js;
- використання пакетів коду npm для прискорення циклу розробки web-додатків;
- використання модулів Node.js для об’єднання програми React в один файл.

Для постійного зберігання інформації дані часто зберігаються в базі даних на сервері. Щоб програма на стороні клієнта отримувала та оновлювала дані з цієї бази, використовується API (інтерфейс програмного забезпечення). Зв'язок між клієнтом і API здійснюється за допомогою HTTP-запитів.

API може бути розроблений кількома способами, два найбільш часто використовуваних є API передачі репрезентаційного стану (RESTful) і новий тип API під назвою GraphQL API.

2.3.1. GraphQL API

Відсутність детальних специфікацій для REST API призвело до різних підходів до його розробки. Основна структура для API REST така ж: при використанні даних із REST API метод в HTTP-запиті описує, яку дію слід виконати з даними, а URL-адреса визначає, до яких даних слід застосувати операцію. URL-адреса відповідає кінцевій точці, визначеній для API, що містить обробники маршрутів, де API виконує обчислення, отримує дані з бази даних або викликає інші API. Обробник маршруту повертає дані, часто у форматі JSON.

GraphQL був випущений у 2015 році Facebook і став альтернативою REST API. GraphQL — це мова запитів для створення та використання API GraphQL. Клієнт може запитувати точні дані, які він хоче отримати, на відміну від REST API, де кожна кінцева точка повертає фіксований фрагмент даних [30]. API GraphQL містить лише одну кінцеву точку, з якої можна використовувати всі дані. Завдяки такому підходу нові запити можна робити від клієнта без змін або створення нових кінцевих точок на стороні сервера. Зв'язок між клієнтом і API здійснюється через HTTP-запити, як і з REST API.

При використанні даних із стороннього API розробник не може контролювати структуру кінцевих точок на стороні сервера. У REST API додатку доводиться робити багато викликів туди й назад, щоб зібрати всі необхідні дані, що призводить до поганої відповідності між API та програмою. Програма також повинна знати відповіді API для різних кінцевих точок, щоб правильно їх обробляти. GraphQL вирішує ці проблеми, надаючи гнучкий API, де клієнт отримує більше контролю над даними.

Опис операцій, доступних в API, називається схемою – це план API типів та їх зв'язків. Схема строго типізована за допомогою мови визначення схем GraphQL (SDL). Перед тим як запит буде виконуватись GraphQL API, він перевіряється відповідно до схеми. Основна частина API GraphQL — це тип, який описує об’єкт, який може бути отриманий API. Існують вбудовані скалярні типи, це примітивні типи: ID, String, Int, Float і Boolean [30].

Існує спеціальний тип під назвою Query, який описує, які запити доступні в API і що буде повертати кожен запит. Приклад запиту представлений на рис. 2.12.

```
type person {
  name: String
  age: Int
}
type Query {
  persons: [person]
}
```

Рис. 2.12. Приклад опису запиту Query

Квадратні дужки вказують, що він повертає масив певного типу.

Іншим типом є тип Mutation, який визначає, які мутації, тобто оновлення даних, доступні в API. Кожен запит і мутація має власний resolver, де вказана логіка виконання обчислень, зв’язку з базою даних або виклику інших API. Resolver повертає дані відповідно до специфікації схеми.

Найпростіший запит — це запит API на отримання поля всередині об’єкта, представлений на рис. 2.13. Запит визначається за допомогою ключового слова "query", а перша пара фігурних дужок називається кореневим об'єктом. В представленому прикладі на кореневому об'єкті вибирається поле "person", для кожного об'єкта person вибирається поля "ім'я" та "вік" .

```
query GetPersonsQuery{
  persons {
    name,
    age
  }
}
```

Рис. 2.13. Приклад написання запиту Query

Результат запиту представлений на рис. 2.14.

Рис. 2.14. Результат запиту

Для оновлення даних використовується мутація. Змінні в мутації використовуються, щоб сказати API, які об’єкти та поля слід змінити. Проста мутація представлена на рис. 2.15.

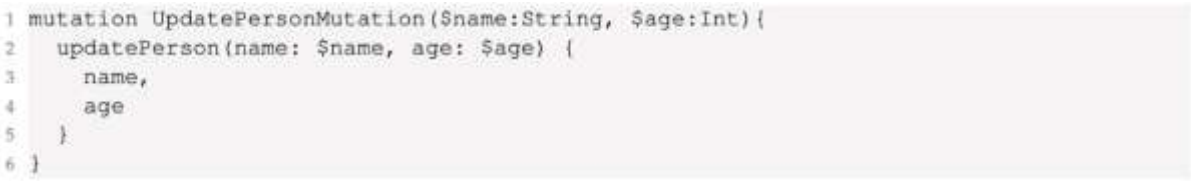

Рис. 2.15. Приклад мутації

Як і у випадку із запитами, мутація повертає дані, які корисні для отримання компонентом клієнтського інтерфейсу при новому стані після оновлення на сторінці. Результат виконання мутації представлений на рис. 2.16.

Рис. 2.16. Результат виконання мутації

Окрім запиту та мутації, існує третій тип операції, який називається підпискою. Ця операція схожа на запити, але натомість вона прослуховує зміни в даних, наданих API і отримує сповіщення про зміни.

Для зручного та швидкого управління інформацією в розробленому додатку використовується Apollo client.

2.3.2. Apollo client

Apollo Client — це повноцінне рішення для управління даними, що дозволяє оновлювати та зберігати інформацію у web-застосунках, а також отримувати даних із API GraphQL.

Використання Apollo Client для керування даними призводить до створення архітектури з односпрямованим потоком даних, подібної до архітектурного шаблону Flux. В даному випадку використовується єдине джерело істини, що містить стан програми, це називається кешом клієнта Apollo. Кеш зберігає всі результати отриманих запитів [31].

Зберігаючи в кеші результати запиту, клієнт Apollo запобігає відправці кількох мережевих запитів до API. Можна обійти кеш, якщо дані потрібно повторно отримати з сервера, змінивши політику отримання на «тільки для мережі». Кеш клієнта Apollo нормалізує отримані дані, тому, якщо список користувачів витягується в запиті, кожен користувач кешується окремо, тому наступні запити для одного користувача отримуються з нормалізованого кешу.

Окрім виконання запиту на віддалені дані з API, можна виконувати запит та зберігати таким способом і локальні дані. Локальні дані зберігаються поряд із віддаленими даними в кеші та отримуються за допомогою додаткового декоратора на ім’я @client, розміщеного після запиту. Доступні локальні типи даних і запити вказуються за допомогою схеми так само, як і для API GraphQL на сервері, описаного раніше в цьому розділі. Оновлення локальних даних у кеші виконується за допомогою клієнтських розпізнавачів, як і для віддалених даних.

Односпрямований потік з використанням клієнта Apollo представлений на рис. 2.17.

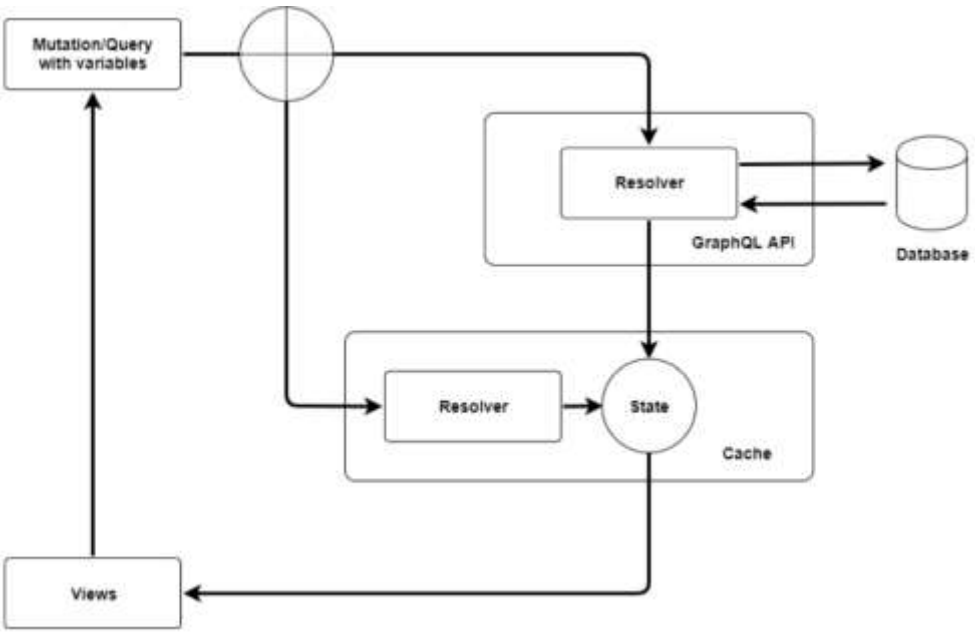

Рис. 2.17. Схема використання Apollo client

2.3.3. Клієнт Apollo в React

Бібліотека apollo-react використовується для надання функціональних можливостей клієнта Apollo для React Views. Клієнт Apollo ініціалізується в кореневому компоненті програми, і для доступу до нього з будь-якого дочірнього компонента є чотири варіанти. Кожна опція надає можливість виконувати запити та мутації в компонентах React.

1. Обгортання кореневого компонента компонентом ApolloProvider з бібліотеки apollo-react. Сутність клієнта Apollo встановлюється як властивість для ApolloProvider, і компоненти нижче по дереву можуть отримати доступ до клієнта за допомогою методу withApollo, доступного з пакету react-apollo. Такий підхід представлений на рис. 2.18.

```
<ApolloProvider client={client}>
  <Greetings />
</ApolloProvider>

const Greetings = ({ client }) => {
  const { loading, error, data } = client.query(GET_PERSONS)
  if (loading) return "Loading..."
  if (error) return error.message
  return data.persons.map(person =>
    <h1>Hello, {person.name}</h1>
  )
}
export default withApollo(Greetings);
```

Рис. 2.18. Приклад обгортання компонента в ApolloProvider

2. Використання хуків React для доступу до Apollo Client у функціональних компонентах має схожий підхід. ApolloProvider підключено з додаткової бібліотеки з іменем @apollo/react-hooks, а клієнт Apollo доступний з дочірніх компонентів за допомогою хука useApolloClient. Використовуючи хуки useQuery та useMutations, можна виконувати запити та мутації, що представлено на рис. 2.19.

```
<ApolloProvider client={client}>
  <Greetings />
</ApolloProvider>

const Greetings = () => {
  const { loading, error, data } = useQuery(GET_PERSONS)
  if (loading) return "Loading..."
  if (error) return error.message
  return data.persons.map(person =>
    <h1>Hello, {person.name}</h1>
  )
}
export default Greetings;
```

Рис. 2.19. Використання useQuery всередині ApolloProvider

3. Використання компонента ApolloConsumer з будь-якого компонента, доступного в бібліотеці apollo-react, шляхом створення функції render, яка приймає єдиний аргумент - клієнт Apollo. Приклад компонента з використанням ApolloConsumer представлений на рис. 2.20.

```
const Greetings = () => (
  <ApolloConsumer>
    {client => {
    const { loading, error, data } = client.query(GET_PERSONS)
     if (loading) return "Loading..."
     if (error) return error.message
     return data.persons.map(person =>
        <h1>Hello, {person.name}</h1>
     )
    }}
  </ApolloConsumer>
);
```

Рис. 2.20. Приклад компонента з використанням ApolloConsumer

Використання описаних технологій Apollo Client дозволяє оперативно та швидко маніпулювати даними в додатку, розробленому за допомогою React, а використання Node.js для розробки серверної складової надає багато можливостей для створення оптимізованого та універсального web-застосунку.

Ще однією важливою частиною проекту є база даних, оскільки розроблений додаток має зберігати велику кількість інформації про клієнтів, IT-спеціалістів, замовлення та інше. В даній роботі використовується реляційна база даних та мова запитів SQL для управління необхідною інформацією.

2.4. Технології для розробки бази даних

Електронні таблиці обробляють числа, а бази даних – інформацію, зокрема структуровану інформацію. Бази даних можуть бути розроблені для будь-яких потреб, які виникають в процесі роботи з даними, наприклад:

- відстеження, впорядкування та редагування даних;
- збирання даних та створення необхідних звітів;
- створення сховища для складних та динамічних web-застосунків.

Найпоширенішою технологією баз даних сьогодні є реляційна база даних. Реляційні бази даних зберігають дані нормалізованим способом — це означає, що дані розбиваються на різні таблиці, щоб уникнути надмірності. Хоча реляційні бази даних існують вже досить давно, вони пропонують універсальний інструмент як для зберігання даних, так і для управління ними.

Основа сучасної технології баз даних почалася в 1970-х роках з першої «реляційної моделі даних». Її акцент був на ретельній організації [32]. Сьогодні реляційні бази даних залишаються важливими та актуальними для створення web-сайтів; будь-який додаток, який відображає дані з бази, повинен мати:

- сценарії отримання даних на стороні сервера;
- зовнішній вигляд, розроблений за допомогою HTML і CSS для відображення даних;
- підтримку SQL, мови бази даних, для можливості виконання запитів;
- систему управління базами даних (СУБД).

Наприклад, в розробленому web-застосунку мають відображатися замовлення клієнта за допомогою запитів в базу даних. Сторінка буде відображатися користувачеві за допомогою HTML, CSS і React, тоді як серверна сторона обробляє з'єднання з базою даних, виконує запит і повертає дані для відображення. Система управління базою даних — це механізм, який запускає базу даних і дозволяє коду на стороні сервера взаємодіяти з базою даних.

Реляційні бази даних складаються з двох або більше таблиць із пов'язаною інформацією, кожна зі стовпцями та рядками. Ці зв'язані таблиці називаються об'єктами бази даних; для створення та керування ними потрібна система управління реляційною базою даних (RDBMS) [32]. Системи управління базами даних дозволяють розробникам створювати та підтримувати програму бази даних, включаючи інструменти для:

- створення даних для запитів;
- редагування існуючих даних;
- проектування загальної структури бази даних;
- збір інформації та генерація звітів;
- перевірки даних в таблицях на відповідність.

Вони часто містять вбудовану мову програмування для автоматизації деяких з цих функцій, наприклад SQL. Використовуючи мову запитів, розробники можуть отримувати дані на основі значення певного стовпця, об'єднувати пов'язані дані з

результатом, виконувати розширені обчислення, форматувати дані у потрібний спосіб тощо.

Редагувати дані можна також за допомогою SQL. Виконуючи запит, можна створювати, оновлювати та видаляти рядки таблиці на основі визначених критеріїв [32].

Проектування структури бази даних також здійснюється за допомогою SQL. Структура реляційної бази даних складається з таблиць, стовпців, індексів та обмежень. У таблиці зберігаються дані певного типу. Стовпці — це властивості сутностей у таблиці. Індекси оптимізують спосіб запиту даних. Обмеження — це правила для даних, що зберігаються в стовпцях. Наприклад, деякі стовпці не можна залишати порожніми. Обмеження корисні, щоб переконатися, що в базі даних не дозволяється зберігати недійсні дані.

В розробленому web-застосунку розробка бази даних є важливим етапом, оскільки саме в ній має зберігатися основна інформація про клієнтів, IT-організації та замовлення, що реєструються на сайті.

Висновки до розділу 2

На сьогодні архітектура більшості web-застосунків складається з двох основних частин: клієнтської та серверної. Клієнтська частина відповідає за інтерфейс, з яким взаємодіють користувачі системи на сторінках web-браузера. Серверна частина містить логіку отримання та обробки даних, необхідних для правильної роботи додатку.

В розробленому web-застосунку для управління послугами в сфері інформаційних технологій основними технологіями для розробки компонентів обрано HTML, CSS та React, побудований на основі мови програмування JavaScript. HTML та CSS дозволяють побудувати загальну структуру користувацького інтерфейсу та задати стилі зовнішнього вигляду та розміщення елементів на web-сторінках. React надає можливість використання синтаксису JSX, до дозволяє описувати логіку обробки даних мовою JavaScript та структуру компоненту за

допомогою HTML в одному файлі, розробки компонентів, що можуть бути використані в проекті декілька разів, технологію віртуального DOM-дерева, що оптимізує завантаження сторінки для користувача при оновленні інформації додатку.

На сервері використовується Node.js, що дозволяє побудувати швидкий та гнучкий додаток.

Для управління даними додатку використано Apollo client в React, який надає багато можливостей для виконання запитів різного виду на отримання актуальної інформації та оновлення даних в компонентах і базі даних.

Розділ 3. Розробка web-застосунку для управління послугами в сфері інформаційних технологій

З розвитком інформаційних технологій більшість підприємств намагається автоматизувати робочі процеси для забезпечення підтримки конкурентоспроможності організації. Також це призводить до оптимізації основної діяльності у сферах покращення обробки інформації в межах компанії та налагодження необхідних комунікаційних каналів між співробітниками.

Процеси, які відносяться до впровадження певних пристроїв або функціональності в сфері інформаційних технологій потребують певного рівня кваліфікації співробітників. Оскільки не всі організації мають сформований IT-відділ, може виникнути необхідність залучення сторонніх фахівців для вирішення питань пов’язаних зі сферою IT.

Розроблений web-застосунок дозволяє будь-яким організаціям, зареєструвавшись на сайті, оформити замовлення на певний вид послуги в сфері інформаційних технологій та знайти спеціаліста для вирішення існуючого питання. Це дозволить співробітникам організації приділяти більше часу основним бізнес-процесам компанії і не витрачати його на вивчення нових технологій для виконання завдань, що не відповідають основній сфері діяльності компанії.

3.1. Проектування та розробка бази даних

Основна інформація, необхідна для коректної роботи web-застосунку зберігається в базі даних.

Для розробки бази даних необхідно визначити основні сутності та їх атрибути, про які необхідно зберігати дані. В розробленому web-застосунку основними сутностями є:

- IT-послуга (service) - містить інформацію про послугу в сфері інформаційних технологій, яку хочу отримати замовник;
- IT-спеціаліст (expert) - містить інформацію про IT-спеціаліста або організацію, яка може виконати поставлену задачу;
- Замовник (client) - певна організація або співробітник, які створюють запит на виконання певної послуги;
- Замовлення (order) - оформлений запит від клієнта на певну IT-послугу.

Кожна сутність представлена в базі даних окремою таблицею. Стовпцями таблиці є атрибути кожної сутності. Дані, що вносяться до таблиці, записуються в новий рядок – кожна клітинка відповідає за встановлене значення певного атрибуту для кожного окремого об’єкту.

Схема описаних сутностей та їх атрибутів представлена на рис.3.1.

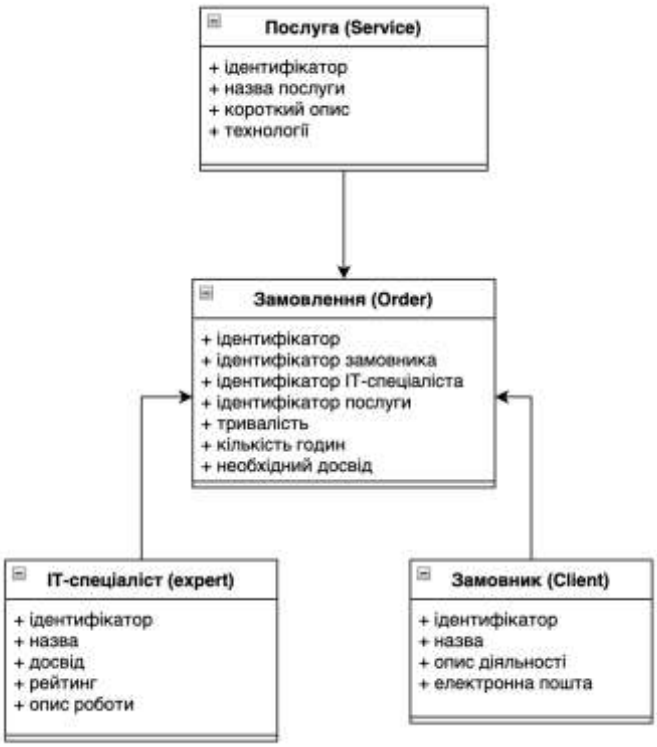

Рис. 3.1. Схема сутностей бази даних web-застосунку

Атрибути IT-послуги складаються з унікального ідентифікатора кожної послуги, її назви, короткого опису робочого процесу та технологій, які можуть використовуватися для виконання певної послуги.

Атрибутами IT-спеціаліста є унікальний ідентифікатор, назва, що може містити назву організації або ПІБ окремого спеціаліста, досвід, що містить інформацію про кількість місяців або років робочого стажу, рейтинг, який розраховується з відгуків на розробленому web-застосунку та опис основної діяльності спеціаліста або організації.

Атрибути замовника містять унікальний ідентифікатор, назву, подібну до назви сутності IT-спеціаліста, опис основної діяльності замовника та електронну пошту для можливості встановлення додаткових контактів.

Сутність замовлення пов’язана з усіма описаними об’єктами за допомогою унікальних ідентифікаторів, містить тривалість часу, необхідного для виконання замовлення, кількість годин, які розробник може витратити протягом тижня та необхідний досвід спеціаліста, який буде відповідати за виконання визначеного замовлення.

Кожний атрибут має свій тип даних та додаткові властивості, які вказують чи може даний атрибут мати нульове (null) значення або бути зв’язком з іншою таблицею.

Нові таблиці в базі даних створені за допомогою мутацій. На рис. 3.2 представлена мутація для створення таблиці замовників.

```
const tableName = 'client';

module.exports = {
  up: (queryInterface, Sequelize) => queryInterface.createTable(
    tableName,
    {
      id: {
        type: Sequelize.INTEGER,
        allowNull: false,
        primaryKey: true,
        autoIncrement: true,
      },
      name: {
        type: Sequelize.STRING,
        allowNull: false,
      },
      work_description: {
        type: Sequelize.STRING,
        allowNull: false,
      },
      email: {
        type: Sequelize.STRING,
        allowNull: false,
      },
    },
  ),
  down: (queryInterface) => queryInterface.dropTable(tableName),
};
```

Рис. 3.2. Код мутації для створення таблиці «client»

При створенні кожної таблиці описується її назва та можливі поля з урахуванням їх особливостей. В мутації описано 2 основних методи: метод up додає описані зміни в базу даних, метод down скасовує внесені зміни. Такий підхід важливий при розробці, оскільки він дозволяє швидко скасувати останні дії при виявленні помилок в створенні таблиці.

Файли для роботи з базою даних розміщені в окремих модулях та складаються з файлів resolvers, які містять файли useCases, що виконуються при відповідній дії. В файлах useCases описана логіка обробки даних для виконання запиту та виклик основних методів роботи з даними з файлу repository. Приклад файлу repository представлений в додатку А.

При запуску команди *npm run graphql:generate* створюються згенеровані файли, які дозволяють використовувати описані запити на роботу з базою даних в компонентах користувацького інтерфейсу. Це дозволяє розділити логіку роботи з базою та інтерфейсом на окремі директорії, що полегшує подальшу розробку та підтримку проекту.

3.2. Розробка інтерфейсу для користувачів

Інтерфейс для користувачів в розробленому web-застосунку складається з наступних основних сторінок:

- головної сторінки web-сайту;
- сторінки з формою для реєстрації;
- сторінки з формою для авторизації;
- сторінки з усіма існуючими замовленнями;
- сторінки з інформацією про зареєстрованих IT-спеціалістів.

Елементи сторінок розроблені на основі React компонентів, які повертають JSX-синтаксис, що складається з HTML-тегів та логіки, написаної мовою JavaScript.

Головна сторінка сайту представлена на рис. 3.3.

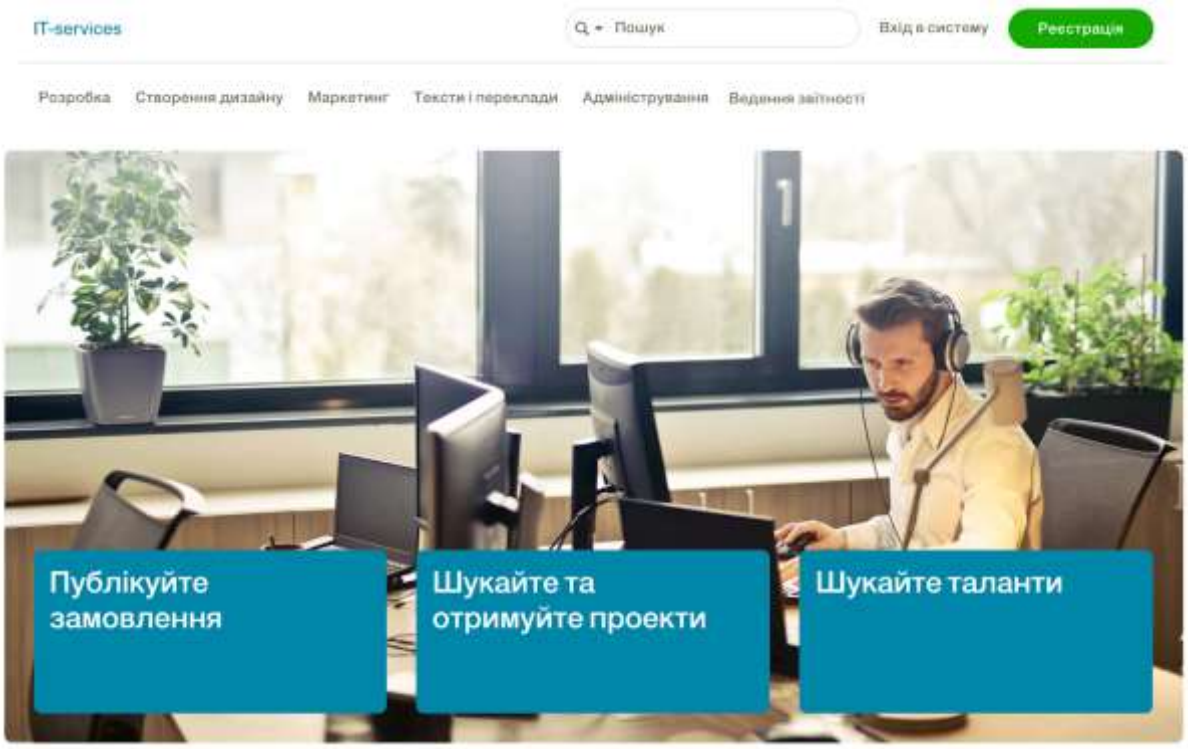

Рис. 3.3. Головна сторінка web-застосунку

На головній сторінці представлене меню з основними видами послуг в сфері інформаційний технологій та кнопки для реєстрації та авторизації на сайті.

Процес реєстрації в розробленому web-застосунку складається з двох етапів: перевірки пошти та налаштування інших даних.

Перший етап реєстрації представлений на рис.3.4.

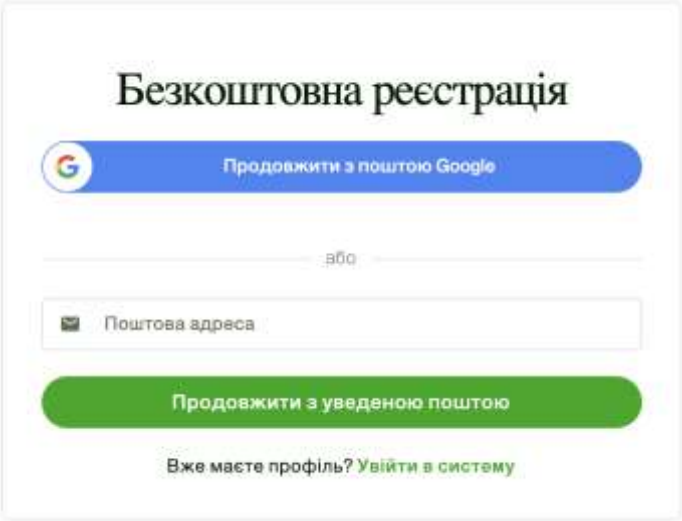

Рис. 3.4. Форма першого етапу реєстрації

На першому етапі користувач може зареєструватися за допомогою пошти google або ввести іншу пошту. Якщо користувач вже зареєстрований на сайті, він може перейти на сторінку авторизації. Другий етап реєстрації представлений на рис. 3.5.

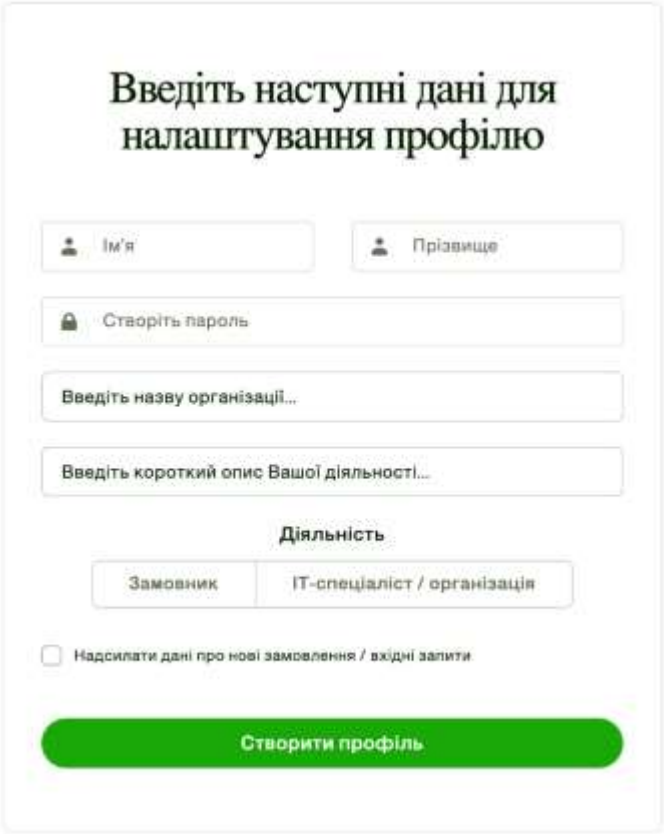

Рис. 3.5. Форма другого етапу реєстрації

На другому етапі користувач має заповнити наступні дані: ім’я, прізвище, назву організації (не обов’язкове поле), короткий опис діяльності та обрати одну з двох опцій - свою роль на сайті, замовника або IT-спеціаліста. Також користувач має створити пароль для можливості наступного входу на сайт.

Для авторизації на сайті необхідно ввести username (ім’я та прізвище), введений при реєстрації або адресу електронної пошти. Якщо реєстрація була виконана за допомогою пошти google, то є варіант такої ж авторизації. Сторінка з формою для авторизації представлена на рис.3.6.

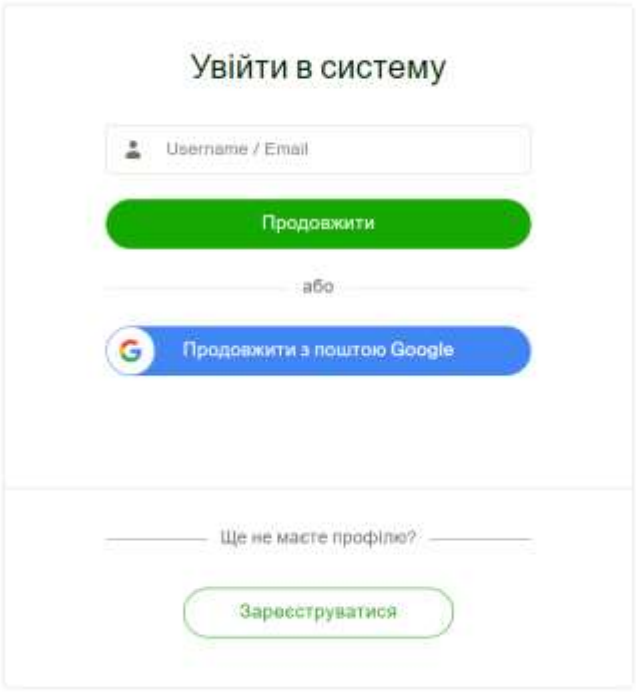

Рис. 3.6. Форма для авторизації користувачів

Після авторизації відкривається доступ до перегляду всіх оформлених замовлень у web-застосунку. Замовлення можна переглядати відповідно до визначеної категорії. На рис. 3.7 представлено приклад перегляду наявних замовлень з розробки.

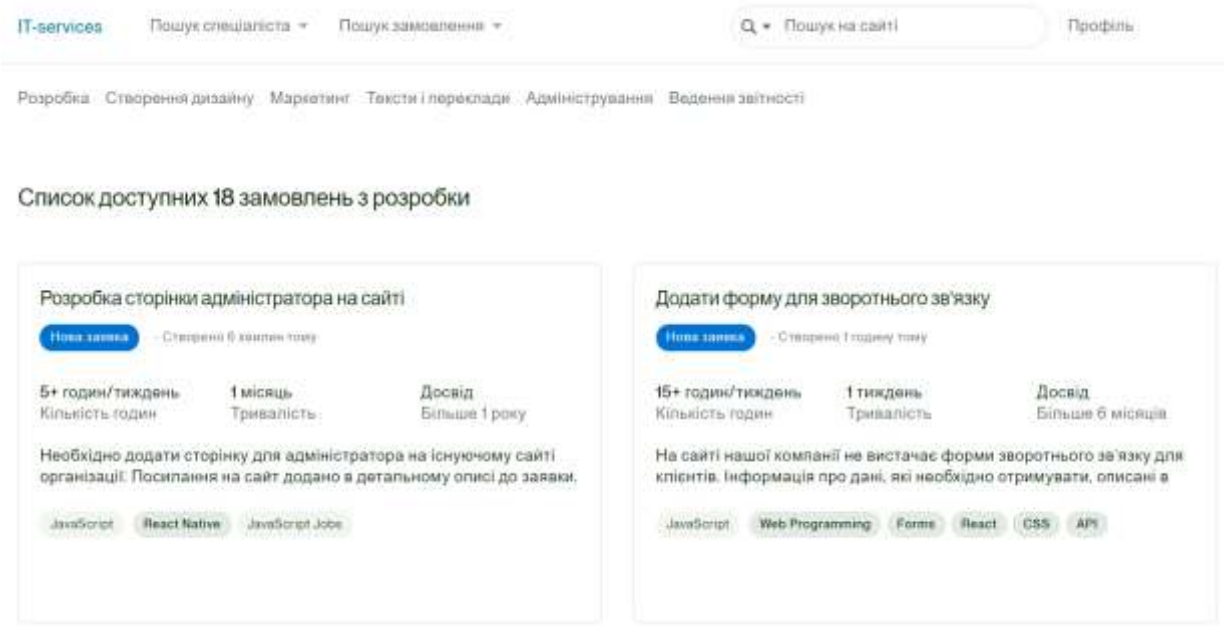

Рис. 3.7. Сторінка зі списком всіх замовлень

В блоці замовлення представлена основна інформація, що допоможе розробнику визначитись, чи може він його виконати. Дані замовлення включать назву, статус та час, що вказують на актуальність виконання замовлення, кількість годин та загальну тривалість виконання роботи (час може бути приблизний, оскільки його визначає замовник, який може помилитися з виставленою оцінкою часу, враховуючи некомпетентність в певних видах інформаційних послуг), короткий опис замовлення, який включає опис необхідного функціоналу, який необхідно розробити та технології, які можуть бути використані при виконанні зазначеної роботи.

При натисканні на певне замовлення IT-спеціаліст або організація може визначитись, надсилати запит на виконання даної роботи чи ні.

При надсиланні запиту на замовлення IT-спеціаліст має дозволити обробити особисті дані, що були введені при реєстрації на сайті, описати коментар до виконання, в якому може зазначити додаткову інформацію про себе або задати питання, що стосуються обраного замовлення. Також, розробник має можливість прикріпити додаткові файли, які можуть вказувати на його досвідченість та професійність у вказаній сфері інформаційних технологій.

Після надсилання запиту визначене замовлення приховується із загального списку та переходить в статус очікування відповіді.

Замовник отримує надісланий запит та вирішує, відкрити особисті контакти для подальшої комунікації з IT-організацією чи ні. В разі відмови, замовник може описати причину і його замовлення знову буде відображатись в списку актуальних робіт.

При підтвердженні запиту на виконання, замовник відкриває свої особисті контакти та налаштовує комунікацію з IT-спеціалістом або організацією. Після виконання поставленої задачі, замовник може залишити відгук про роботу виконавця та вказати певний бал за виконану роботу, що впливатиме на рейтинг виконавця.

Подивитися рейтинг та іншу інформацію про співробітників або зареєстровані IT-організації можна на сторінці, що представлена на рис. 3.8.

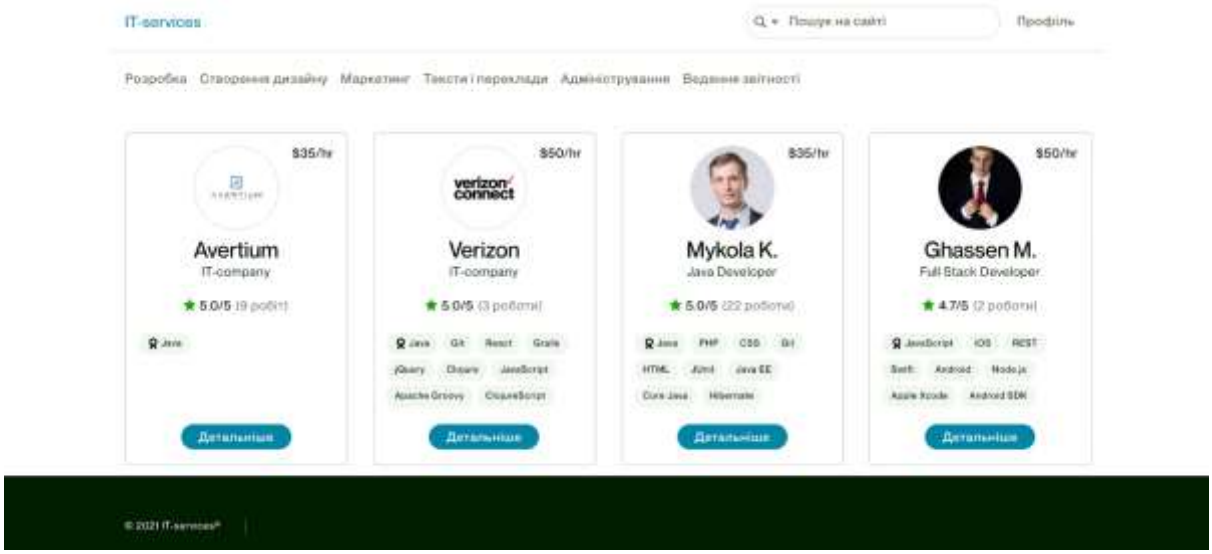

Рис. 3.8. Сторінка зі списком IT-організацій та IT-спеціалістів

Для кожного зі спеціалістів вказана посада та досвід співробітника, кількість виконаних замовлень, отриманих в розробленому web-застосунку, а також основні технології, з якими працює визначений фахівець.

Розроблений інтерфейс, що складається з описаних сторінок, надає можливість користувачам web-застосунку публікувати та виконувати будь-які роботи в сфері інформаційних технологій. Для належної роботи та виконання всіх зазначених функцій в даній роботі використано JavaScript та бібліотеку React, що дозволяють написати логіку виконання основних дій користувачів на сайті.

3.3. Розробка функціональної складової web-застосунку

Оскільки користувацький інтерфейс розроблений за допомогою React компонентів, найкращим рішенням є написання функціональної логіки мовою програмування JavaScript за допомогою вбудованих рішень з бібліотеки React.

Для опису логіки роботи web-застосунку дані отримуються та обробляються безпосередньо в компонентах, та передаються дочірнім елементам за допомогою об’єкта властивостей. Для класових компонентів початкові дані зберігаються в об’єкті стану.

Основні функції розробленого web-застосунку включають:

- реєстрацію нових користувачів;
- авторизацію зареєстрованих користувачів;
- публікацію замовлень (необхідних робіт);
- надсилання запиту на виконання замовлення;
- отримання списку всіх замовлень відповідно до обраної послуги.

За реєстрацію нових користувачів відповідає компонент UserSignUp. Основний код компоненту представлений на рис. 3.9.

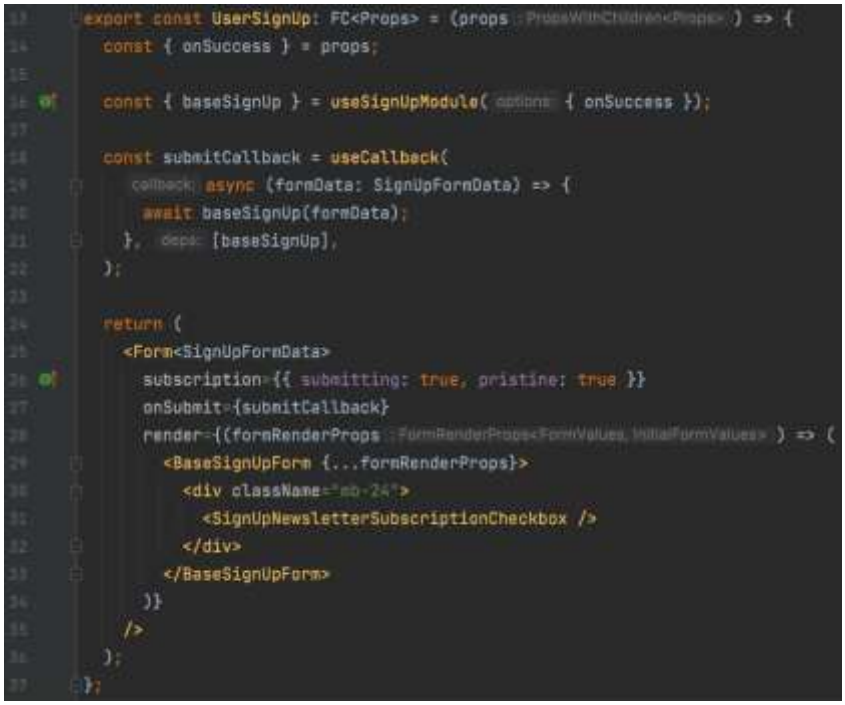

Рис. 3.9. Код компонента UserSignUp

В даному компоненті використано користувацький хук useSignUpModule, який відповідає за обробку функції реєстрації, що працює з сервером. Компонент повертає форму для реєстрації, при надсиланні якої виконується метод submitCallback, який отримує дані, які заповнив користувач на сайті та виконує безпосередню реєстрацію.

Код мутації, яка виконується при реєстрації нового користувача представлена в додатку Б. Для кожного користувача створюється унікальний ключ (токен), який зберігається в local storage у web-браузері користувача. Це дозволяє користувачу не

проходити авторизацію кожного разу при оновленні сторінки або завершенні роботи системи.

Код основного компоненту для авторизації користувачів представлений на рис. 3.10.

```
export const UserSignInPageModule = () => {
  const { redirectUrl } = useRouterQuery<{redirectUrl?: string}>();

  const { t } = useTranslation( ns: [I18N_CODES.loginPage]);

  const signInCallback = useCallback( callback: () => {
    if (redirectUrl) {
      redirect( context: {}, decodeURIComponent(redirectUrl));
    } else {
      redirect( context: {}, ROUTES.user.profile);
    }
  }, deps: [redirectUrl]);

  const { oAuthSignIn } = useSignInModule( options: {
    onSuccess: signInCallback,
  });

  return (
    <...>
  );
};
```

Рис. 3.10. Код компонента UserSignUpPageModule

Для отримання функції авторизації використовується подібний до хука реєстрації розроблений хук useSignInModule. При успішній аторизації користувач переходить на сторінку свого профілю для перегляду та перевірки введеної інформації. Якщо не зареєстрований користувач намагається переглянути замовлення, розміщені на сайті, без авторизації, йому відкриється форма для входу і при успішному введенні даних користувачу відкривається сторінка, на яку він хотів потрапити.

Код компонента, що відповідає за форму авторизації представлений на рис. 3.11.

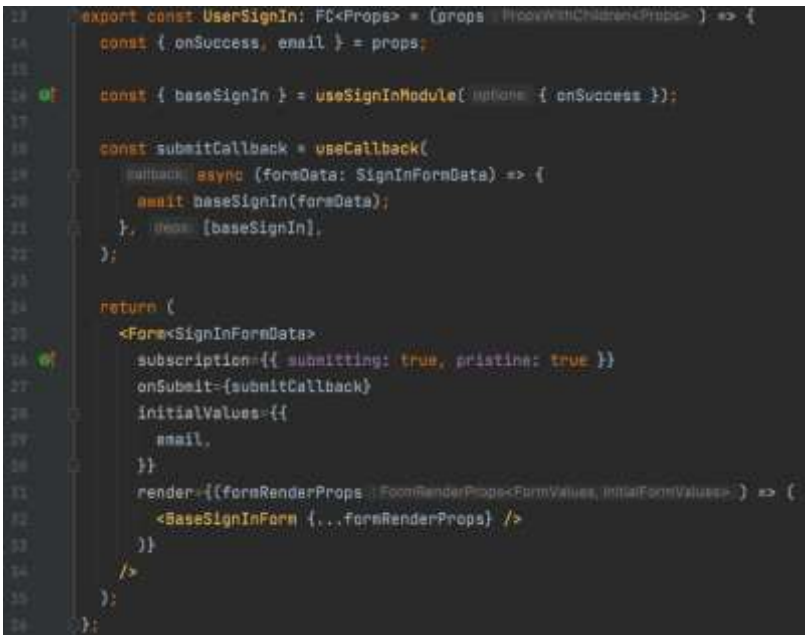

Рис. 3.11. Код компонента UserSignIn

Для обробки створеної форми використовується попередньо створені типи даних полів для авторизації SignInFormData, описана функція submitCallback, яка виконується при відправці даної форми.

При авторизації користувачів відбувається перевірка його ролей (замовник чи виконавець), перевірка введених даних за допомогою виконання запиту в базу даних, а також створюється та зберігається в web-браузері унікальний код користувача для полегшення подальшої роботи з web-застосунком.

Після авторизації в системі замовнику доступний функціонал публікації робіт, тобто, розміщення запитів на отримання певної послуги в сфері інформаційних технологій.

За публікацію нових замовлень відповідає компонент CreateJob (рис. 3.12).

Рис. 3.12. Код компонента CreateJob

При натисканні на кнопку ‘додати замовлення’ відкривається модальне вікно для введення необхідної інформації. Дані про відкриття або закриття модального вікна зберігаються в об’єкті стану функціонального компоненту за допомогою використання хука useState().

При створенні форми для додавання замовлення виконується 2 основні методи: getInitialValues() та handleSubmit(), представлені на рис. 3.13.

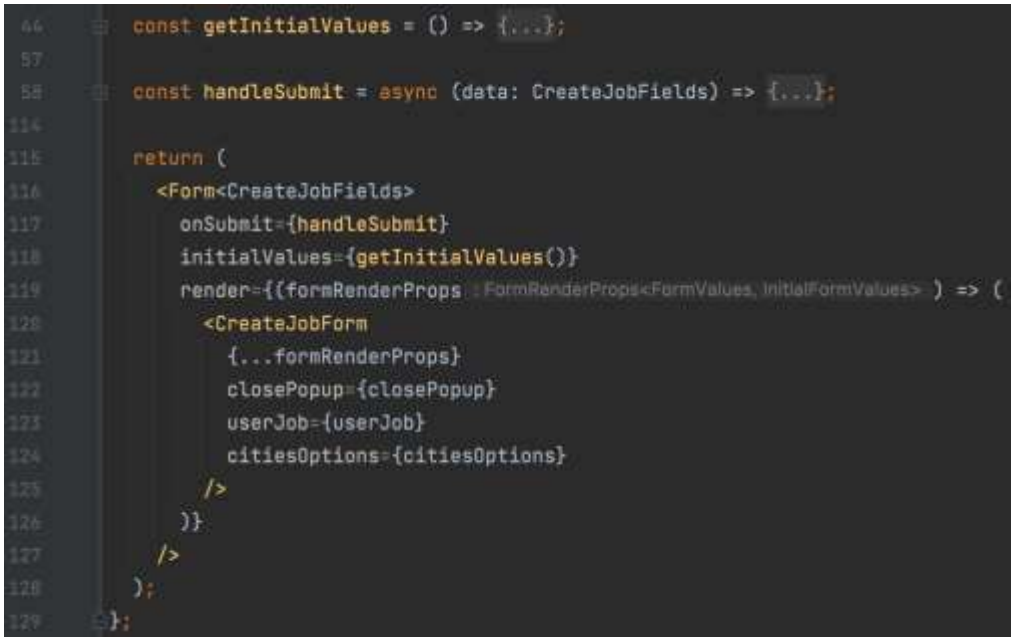

Рис. 3.13. Код для обробки форми створення замовлення

Функція getInitialValues() задає початкові значення для форми. У випадку створення нового замовлення ці значення мають бути нульовими, при редагуванні

певного замовлення ці значення отримуються з бази даних та встановлюються як початкові.

Функція handleSubmit() виконується при надсиланні форми з даними про замовлення. Вся введена інформація обробляється за допомогою відповідної мутації і в базі даних створюється нова сутність в таблиці orders. При створенні нового замовлення в базі даних його статус встановлюється в значення ‘new’.

Авторизовані клієнти можуть переглянути список всіх доступних замовлень на web-сайті. Даний функціонал реалізовано за допомогою компонента OrderList (рис. 3.14).

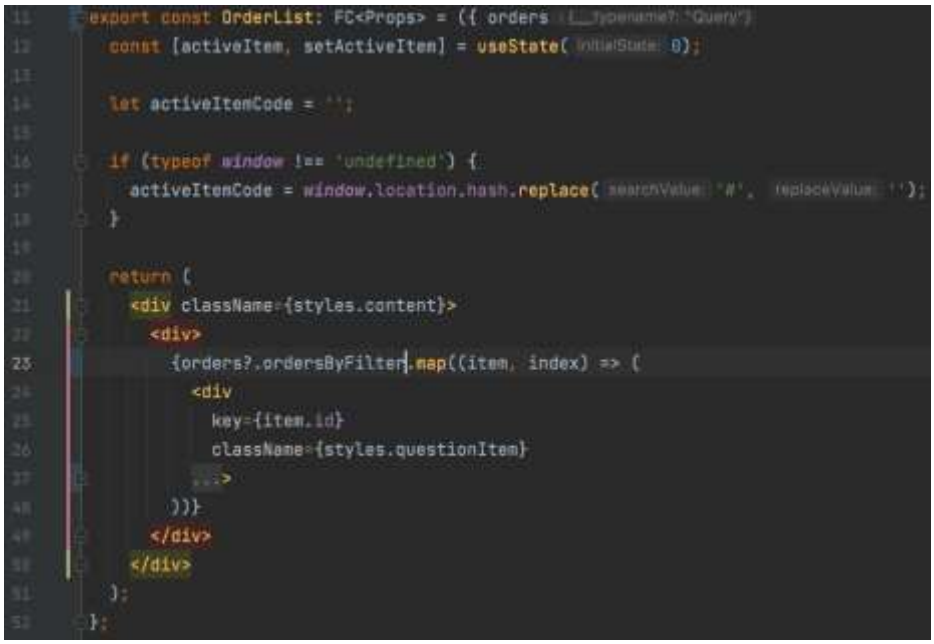

Рис. 3.14. Код компонента OrderList

В даний компонент в об’єкті властивостей приходить массив з усіма замовленнями, отриманий за допомогою виконання запиту в базу даних. При використанні певного фільтру (пошуку замовлень тільки за визначеними послугами в сфері інформаційних технологій) до запиту в базу даних додаються додаткові умови, що забезпечує відбір тільки необхідних робіт.

При натисканні на певне замовлення воно стає активним, та записується в змінну activeItemCode для полегшення подальшої обробки обраної роботи.

Для кожного замовлення створюється окремий компонент, що виводиться на web-сторінку користувачу з необхідною йому інформацією.

Для виконання певного замовлення IT-спеціалістам необхідно надіслати відповідний запит. Функція для надсилання запиту представлена на рис. 3.15.

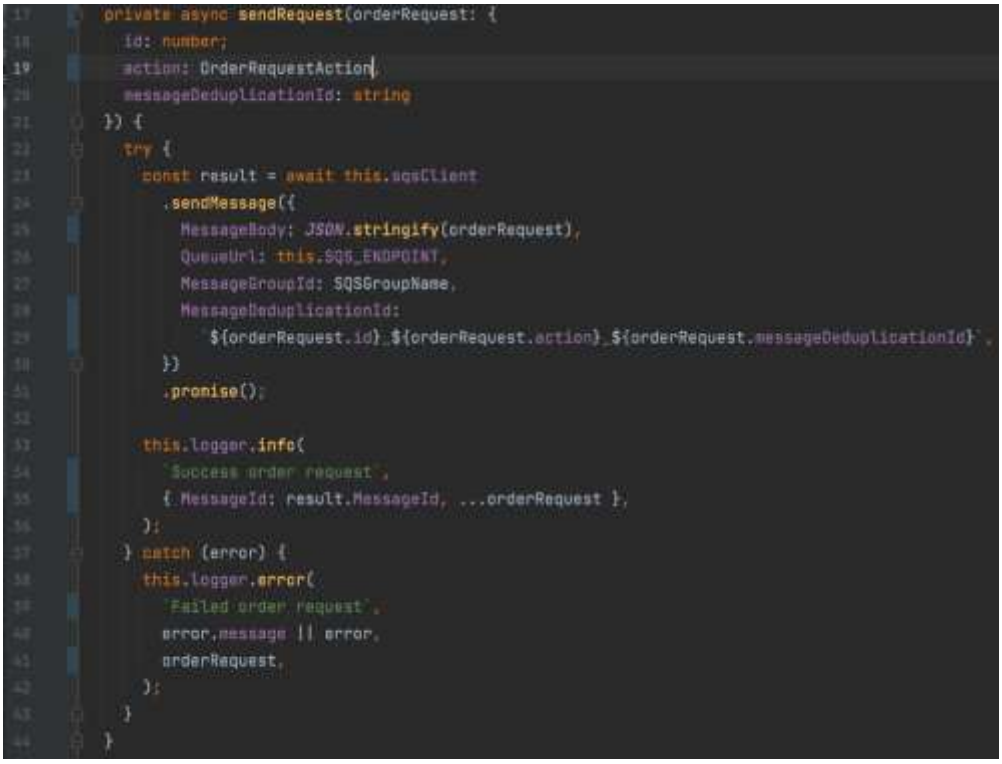

Рис. 3.15. Код методу sendRequest

При виконанні запиту на виконання певного замовлення відбувається надсилання повідомлення замовнику з інформацією про здійснену подію. При виникненні будь-яких помилок в процесі надсилання повідомлення або успішному результаті IT-спеціаліст побачить відповідне повідомлення на сторінці. Помилки при надсиланні запиту можуть виникнути у випадку, якщо в IT-спеціаліста замалий досвід для обраного замовлення або вказана недостовірна інформація при реєстрації на сайті (наприклад, була створена копія певної IT-організації).

При отриманні такого повідомлення замовник може переглянути інформацію про IT-спеціаліста або організацію, яка надіслала запит та прийняти чи відхилити його. При цьому, замовник може вказати (за бажанням) певний коментар та відкрити свої контакти для подальшої комунікації з виконавцем.

Висновки до розділу 3

Розвиток інформаційних технологій призводить до збільшення необхідності впровадження багатьох послуг в сфері інформаційних технологій в робочі процеси більшості організацій. Для автоматизації управління даними послугами розроблено web-застосунок, що дозволяє всім бажаючим знайти IT-спеціаліста для виконання певного виду робіт.

Основними етапами розробки web-застосунку є створення бази даних, проектування користувацького інтерфейсу та логіки роботи додатку.

В розробленій базі даних зберігається основна інформація, необхідна для правильної роботи web-застосунку: інформація про замовників, послуги, IT-спеціалістів (виконавців) та безпосередні замовлення.

Основними сторінками web-сайту є сторінки реєстрації та авторизації користувачів, перегляду списку замовлень та списку IT-організацій (виконавців). Елементи даних сторінок створені за допомогою компонентів React, що повертають елемент у вигляді синтаксису JSX: поєднання HTML з JavaScript.

Для опису логіки функціонування web-застосунку використано мову програмування JavaScript. Основні функції виконуються перед відображенням певних елементів на сайті, що дозволяє спочатку виконати всі необхідні дії, а потім відобразити користувачу актуальну та оновлену інформацію.

Основні можливості системи включають:

- реєстрацію IT-організацій (постачальників послуг);
- реєстрацію інших організацій (замовників послуг);
- розміщення замовлень на виконання послуг в сфері інформаційних технологій;
- перегляд існуючих замовлень відповідно до обраної IT-послуги з урахуванням технологій, які необхідно використати для їх виконання;
- перегляд зареєстрованих IT-спеціалістів та організацій з урахуванням основного стеку технологій, виконаних робіт та рейтингу на сайті;

- можливість надсилання запиту для виконання замовлення на визначену роботу.

Розроблена система надає можливість організаціям отримувати при необхідності бажані послуги від IT-експертів та ефективно витрачати час співробітників на виконання основних робочих процесів компанії.

ВИСНОВКИ

Активний розвиток сфери інформаційних технологій призводить до збільшення кількості процесів, які можна автоматизувати. Кожна організація прагне до впровадження новітніх технологій, які допоможуть покращити продуктивність роботи компанії в цілому.

На сьогодні існує багато послуг в сфері інформаційних технологій, які дозволяють впровадити нові рішення для покращення робочих процесів: хмарні послуги, резервне копіювання, забезпечення безпеки робочої мережі, ведення звітності, розробка програмного забезпечення та інші.

Проте не всі організації мають в штаті достатньо кваліфікованих співробітників для впровадження зазначених послуг. Тому, на сьогодні є актуальною задача автоматизації процесу отримання та надання послуг в сфері інформаційних технологій.

Найбільш поширеним методом пошуку інформації є мережа Інтернет, тому було вирішено виконати поставлену задачу за допомогою розробки web-застосунку.

Основними складовими web-застосунку є клієнтська та серверна частини, що забезпечують розподіл користувацького інтерфейсу від основної функціональності системи.

Клієнтська частина в даній роботі розроблена за допомогою таких технологій, як HTML, CSS та React із застосуванням мови програмування JavaScript. Дана складова відповідає за елементи інтерфейсу, за допомогою яких користувач може отримувати необхідну інформацію та взаємодіяти з додатком у web-браузері.

Основні сторінки, які створені за допомогою зазначених технологій, включать основну сторінку web-застосунку, сторінку для реєстрації та авторизації користувачів, виведення оформлених замовлень на сайті та IT-організацій, які відповідають за їх виконання. Всі елементи на цих сторінках створені за допомогою React компонентів.

Серверна частина створена за допомогою Node.js, який дозволяє побудувати швидкий та гнучкий web-застосунок. Дана складова відповідає за зберігання та обробку даних, які використовуються при роботі користувачів з web-застосунком.

В розробленому web-застосунку створені мутації для реєстрації користувачів, запити на отримання та додавання нових замовлень в базу даних та інші. Основна логіка обробки даних описана в компонентах React за допомогою мови програмування JavaScript. Дані функції включають обробку даних після отримання інформації з бази, фільтрацію замовлень відповідно до обраних послуг на сайті, надсилання повідомлень про виконання запиту на певне замовлення для користувачів.

Для створення та зберігання даних в базі використано Apollo client в React, який дозволяє виконувати усі необхідні види запитів в базу на отримання та оновлення необхідної інформації, що дозволяє показувати користувачу оновлену та актуальну інформацію.

В майбутньому розроблений web-додаток можна покращити за допомогою впровадження наступних функцій:

- додати можливість редагування інформації про створене замовлення;
- налагодити комунікацію між замовником та виконавцем в межах web-застосунку, не потребуючи додаткової інформації про особисті контактні дані;
- розробити адміністративну панель для можливості додавання нових видів послуг.

Розроблений інтерфейс та основний функціонал web-застосунку для управління послугами в сфері інформаційних технологій забезпечує можливість оформлення замовлення для будь-якої організації на отримання бажаної послуги, а IT-спеціалістам знайти роботу та допомогти налагодити та оптимізувати робочі процеси іншим компаніям.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Main Types of IT Services [Electronic resource]. Access mode: https://myva360.com/blog/the-11-main-types-of-it-services (lastaccess: 12.10.21) – Title from the screen.

2. 5 Types of IT Services That Make the Biggest Impact on Your Business [Electronic resource]. Access mode: https://www.technologyhq.org/5-types-services-make-biggest-impact-business/ (lastaccess: 12.10.21) – Title from the screen.

3. What is cloud backup and how does it work? [Electronic resource]. Access mode: https://searchdatabackup.techtarget.com/definition/cloud-backup (lastaccess: 13.10.21) – Title from the screen.

4. 13 Types of IT Services: What They Are and How They Help [Electronic resource]. Access mode: https://www.indeed.com/career-advice/career-development/examples-of-it-services (lastaccess: 15.10.21) – Title from the screen.

5. Top 20 Best Project Management Software in 2021: An Overview [Electronic resource]. Access mode: https://mopinion.com/top-20-best-project-management-software-an-overview/ (lastaccess: 11.10.21) – Title from the screen.

6. How to fix 8 common remote desktop connection problems [Electronic resource]. Access mode: https://searchvirtualdesktop.techtarget.com/tip/ (lastaccess: 16.10.21) – Title from the screen.

7. Software-as-a-Service (SaaS) [Electronic resource]. Access mode: https://www.investopedia.com/terms/s/software-as-a-service-saas.asp (lastaccess: 18.10.21) – Title from the screen.

8. How to Organize Your Software Product Development Process [Electronic resource]. Access mode: https://euristiq.com/how-to-organize-software-product-development/ (lastaccess: 20.10.21) – Title from the screen.

9. Customer Service vs Technical Support: What's The Difference? [Electronic resource]. Access mode: https://www.bmc.com/blogs/customer-service-vs-technical-support/ (lastaccess: 17.10.21) – Title from the screen.

10. Hardware And Software Maintenance Services [Electronic resource]. Access mode: https://www.gartner.com/en/information-technology/ (lastaccess: 25.10.21) – Title from the screen.

11. What is Web Forms [Electronic resource]. Access mode: https://docs.microsoft.com/en-us/aspnet/web-forms/what-is-web-forms (lastaccess: 03.11.21) – Title from the screen.

12. Client-Server Definition [Electronic resource]. Access mode: https://www.omnisci.com/technical-glossary/client-server (lastaccess: 27.10.21) – Title from the screen.

13. Web Application Architecture in 2021: Moving in the Right Direction [Electronic resource]. Access mode: https://mobidev.biz/blog/web-application-architecture-types (lastaccess: 25.10.21) – Title from the screen.

14. Website vs web application: what's the difference and what to choose? [Electronic resource]. Access mode: https://lanars.com/blog/website-vs-web-application-what-s-the-difference-and-what-to-choose (lastaccess: 27.10.21) – Title from the screen.

15. Interactive Content Guide: How to Bring Life to Your Content Marketing Strategy [Electronic resource]. Access mode: https://rockcontent.com/blog/what-is-interactive-content/ (lastaccess: 18.10.21) – Title from the screen.

16. About web applications [Electronic resource]. Access mode: https://helpx.adobe.com/ua/dreamweaver/ (lastaccess: 08.11.21) – Title from the screen.

17. Definition of 'Web Server' [Electronic resource]. Access mode: https://economictimes.indiatimes.com/definition/web-server (lastaccess: 25.10.21) – Title from the screen.

18. E-commerce [Electronic resource]. Access mode: https://searchcio.techtarget.com/definition/e-commerce (lastaccess: 18.10.21) – Title from the screen.

19. E-Commerce: Purchasing and Selling Online [Electronic resource]. Access mode: http://www.gov.pe.ca/photos/original/IPEI_ebiz_ecomm.pdf (lastaccess: 19.10.21) – Title from the screen.

20. What Is a Single Page Application and Why Do People Like Them so Much? [Electronic resource]. Access mode: https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html (lastaccess: 25.10.21) – Title from the screen.

21. Web Applications Architectures: Components, Layers, and Types [Electronic resource]. Access mode: https://www.cleveroad.com/blog/web-application-architecture (lastaccess: 12.11.21) – Title from the screen.

22. HTML <head> Tag [Electronic resource]. Access mode: https://www.w3schools.com/tags/tag_head.asp (lastaccess: 14.11.21) – Title from the screen.

23. CSS selectors [Electronic resource]. Access mode:https://developer.mozilla.org/en-US/docs/Web/CSS/ (lastaccess: 16.11.21) – Title from the screen.

24. Functional Programming with JavaScript [Electronic resource]. Access mode: https://www.telerik.com/blogs/functional-programming-javascript (lastaccess: 13.11.21) – Title from the screen.

25. What exactly is Node.js? [Electronic resource]. Access mode: https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5/ (lastaccess: 15.11.21) – Title from the screen.

26. React. A JavaScript library for building user interfaces [Electronic resource]. Access mode: https://reactjs.org/ (lastaccess: 14.11.21) – Title from the screen.

27. Chapter 5. React component lifecycle events [Electronic resource]. Access mode: https://livebook.manning.com/book/react-quickly/chapter-5/ (lastaccess: 18.11.21) – Title from the screen.

28. How to Use React Hooks in Class Components [Electronic resource]. Access mode: https://infinum.com/the-capsized-eight/how-to-use-react-hooks-in-class-components (lastaccess: 14.11.21) – Title from the screen.

29. How To Set Up a React Project with Create React App [Electronic resource]. Access mode: https://www.digitalocean.com/community/tutorials/ (lastaccess: 16.11.21) – Title from the screen.

30. GraphQL: A data query language [Electronic resource]. Access mode: https://graphql.org/blog/graphql-a-query-language/ (lastaccess: 16.11.21) – Title from the screen.

31. Get real-time updates from your GraphQL server [Electronic resource]. Access mode: https://www.apollographql.com/docs/react/data/subscriptions/ (lastaccess: 14.11.21) – Title from the screen.

32. What Is Database Technology? 2022 Guide to Database Tech [Electronic resource]. Access mode: https://www.upwork.com/resources/data-base-technology (lastaccess: 15.11.21) – Title from the screen.

ДОДАТКИ

Додаток А

Програмна реалізація файлу repository для сутності Order

```
import Sequelize from 'sequelize';
import { Repository } from '@/core/Repository';
import { Order } from '@/models/Order';

export class OrderRepository extends Repository {
  async findById(id: number): Promise<Order | null> {
    return this.models.Order.findByPk(id, { raw: true });
  }

  async getById(id: number): Promise<Order> {
    const order = await this.findById(id);

    if (!order) {
      this.throwNotFoundError(
        orderErrors.OrderNotFound,
        { id },
      );
    }

    return order;
  }

  async findByService(service: string) {
    const where = {
      '$service.name$': service,
    };

    return this.models.Order.findAll({
      include: [
        {
          model: Service,
          attributes: ['name'],
        },
      ],
      where,
      order: [
        ['order', 'ASC'],
      ],
    });
  }

  async findByTechnologies(technologies: string[]) {
    return this.models.Order.findAll({
      where: {
        technology: {
          [Sequelize.Op.in]: technologies,
        },
      },
      order: [
        ['order', 'ASC'],
      ],
    });
  }
}
```

Програмна реалізація мутації для реєстрації нових користувачів на web-сайті

```
import { UseCase } from '@/core';
import { User } from '@/models/User';
import { UserEvents, UserEventSignedUpPayload } from '@/modules/user/user.events';
import { UserEntity } from '@/modules/user/user.entity';
import { UserService } from '@/modules/user/user.service';
import { LanguageRepository } from '@/modules/language/language.repository';
import { LanguageName } from '@/modules/language/language.typedefs';
import { AccessTokenService } from '@/modules/accessToken/accessToken.service';

export interface SignUpUseCaseOptions {
  email: string;
  phone?: string;
  firstName?: string;
  lastName?: string;
  password?: string;
  locale?: LanguageName;
  referralCode?: string;
  newslettersSubscription?: boolean;
}

export interface SignUpUseCaseResult {
  user: User;
  accessToken: string;
}

type Options = SignUpUseCaseOptions;
type Result = SignUpUseCaseResult;

export class SignUpUseCase extends UseCase<Options, Result> {
  private readonly userService = this.makeService(UserService);

  private readonly accessTokenService = this.makeService(AccessTokenService);

  private readonly languageRepository = this.makeRepository(
    LanguageRepository,
  );

  protected async run(options: Options): Promise<Result> {
    const {
      email, password, referralCode, ...rest
    } = options;

    const language = await this.languageRepository.findByName(
      options.locale as LanguageName || '',
    );

    const userOptions = {
      ...rest,
      email: UserEntity.formatEmail(email),
      password,
      languageId: language?.id,
    };

    const user = await this.userService.createWithReferralCode(
      userOptions,
      referralCode,
    );
```

```
    const accessToken = await this.accessTokenService.create({
      userId: user.id,
      email: user.email,
    });

    await this.userService.syncToBpm(user.id);

    this.emitMessage<UserEventSignedUpPayload>(
      UserEvents.UserSignedUp, user,
    );

    return {
      user,
      accessToken: accessToken.token,
    };
  }
}
```