Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Дипломна робота
магістра

з теми: «**ВЕБ-ДОДАТОК CITY INFORMER**»

Виконав: студент групи KN1-M21
спеціальності 122 Комп’ютерні науки
Шлапак Андрій Вікторович

Керівник: **Слободянюк О. В.,**
кандидат технічних наук, доцент
кафедри комп’ютерних наук

Кам’янець-Подільський – 2022

ЗМІСТ

ВСТУП 3
РОЗДІЛ 1. ТЕХНОЛОГІЇ РОЗРОБКИ ВЕБ-ДОДАТКІВ 4
1.1. Поняття веб-додатку 4
1.2. Особливості веб-додатків 7
1.3. Переваги та недоліки веб-додатків 9
1.4. Архітектура веб-додатків 10
1.5. Відмінність веб-додатку від сайту 14
РОЗДІЛ 2. ПРИКЛАДИ ВЕБ-ЗАСТОСУНКІВ ДЛЯ УПРАВЛІННЯ КОМУНАЛЬНИМИ ПОСЛУГАМИ 19
2.1. Веб застосунок 104.ua 19
2.2. Веб-застосунок ienergo.com.ua 21
РОЗДІЛ 3. СТВОРЕННЯ ВЕБ-ЗАСТОСУНКУ CITY INFORMER 24
3.1. Принцип роботи сіті інформера 24
3.2. Архітектура сіті інформера 25
3.3. Розробка дизайну веб-застосунку 26
3.4. Написання коду в Angular 31
ВИСНОВКИ 35
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 36
ДОДАТКИ 37

ВСТУП

Актуальність теми. Глобалізація інформатизації суспільства і активний процес науково-технічного розвитку в області інформаційних систем сприяють формуванню єдиного світового інформаційного простору. Сьогодні інформаційні технології спрощують всі сфери життя людини. Поява Інтернету та веб-додатків посприяла заміні звичайним сервісам.

Метою роботи є огляд поняття веб-додатку і поняття City informer, та з врахуванням особливостей здійснити практичну реалізацію інформера використовуючи фреймворк Angular.

Відповідно до поставленої мети були сформульовані наступні **завдання дослідження:**

1) провести огляд веб-додатку;
2) розглянути поняття City informer;
3) створити веб-додаток City informer використовуючи фреймворк Angular.

Об'єктом дослідження є методи та технології розробки програмних застосунків.

Предметом дослідження є технології розробки програмного забезпечення.

Методи дослідження: спостереження, аналіз, узагальнення, синтез.

Структура роботи. Дана магістерська робота складається зі вступу, теоретичної частини, практичної частини, висновків та додатків.

РОЗДІЛ 1. ТЕХНОЛОГІЇ РОЗРОБКИ ВЕБ-ДОДАТКІВ

1.1 Поняття веб-додатку

Рис. 1.1 Схема веб-додатку

У сучасному світі використання веб-додатків зростає з кожним днем. Професіонали програмного забезпечення, такі як розробник програмного забезпечення та тестеризація програмного забезпечення, повинні ознайомитися з веб-додатками [1].

Веб-додаток — це програма-клієнт-сервер, що зберігається на віддаленому сервері, яка використовує веб-браузери та веб-технології для виконання певних функцій через Інтернет через інтерфейс браузера [1].

Як було сказано вище, це прикладна програма клієнт-сервер, тому в середовищі клієнт-сервер багато комп'ютерів можуть обмінюватися такою інформацією, як збереження інформації в базі даних. "Клієнт" може використовуватися для введення інформації, а "сервер" використовується як сховище для інформації [1].

Простими словами, ви можете визначити його як комп'ютерну програму, яка виконує певні завдання у свого клієнта, використовуючи веб-браузер. Веб-додатки також відомі як веб-додатки [1].

Приклади до поширених. Вона включає веб-пошту, таку як Gmail, Yahoo та AOL, роздрібні продажі в Інтернеті, онлайн-форми, торгові візки, текстовий процесор, електронні таблиці, редагування відео та фотографій, конвертація файлів, сканування файлів, Google Apps, такі як Google Docs, Google Таблиці, Google Слайди, інтернет-сховища тощо [1].

Існує шість різних типів веб-додатків:

- Статичний
- Динамічний
- Інтернет-магазин або електронна комерція
- Портал веб-додатків
- Анімовані
- Система управління вмістом

Веб-додаток може виконувати певні функціональні можливості через Інтернет, використовуючи веб-браузери та веб-технології. Програми обробляють сховище та отримують інформацію за допомогою скриптів на стороні сервера, таких як PHP та ASP; представити інформацію користувачеві за допомогою сценаріїв на стороні клієнта, таких як JavaScript та HTML [1].

Найкращим прикладом для веб-додатків є веб-адреса електронної пошти, де Gmail та Outlook Microsoft надають клієнтам електронної пошти на основі веб-сторінок [1].

Ви можете зробити наступні дії:

- Ви можете відображати звіти у графічному форматі.
- Щоб отримати інформацію про товар в Інтернеті.

- Робітники можуть отримувати інформацію, пов’язану із завданнями, через мобільні пристрої.
- Ця програма дозволяє віддаленим працівникам виконати завдання протягом визначеного часу.
- Клієнти можуть легко відстежувати замовлення та бюджет товарів.
- Нижче перелічені переваги використання веб-додатків:
- Він працює на різних типах платформ.
- Дані захищено та легко взяти резервну копію.
- Ви можете легко оновити програму.
- Ви можете легко використовувати низькі характеристики ПК або смартфонів.
- Це усуває проблеми сумісності, оскільки користувач може отримати доступ до тієї ж версії.
- За допомогою SaaS вони зменшують програмне піратство у веб-додатках на основі підписки.
- За допомогою веб-програми працівники можуть працювати з будь-якого місця, користуючись доступом до Інтернету.

Нижче наведено основні навички, необхідні для розробки веб-додатків:

- Обробляйте сховище та отримуйте інформацію за допомогою скриптів на стороні сервера, таких як PHP, ASP.Net та Ruby.
- Мови скриптування на стороні клієнта (JavaScript, HTML та CSS).
- Photoshop
- WordPress та SEO

- Навички чуйного дизайну
- Інструменти розробки, такі як IDE, редактори (Visual Studio, Eclipse)
- Веб-сервери (Apache, IIS)

1.2 Особливості веб-додатків

Веб-додаток отримує запит від клієнта і виконує обчислення, після цього формує веб-сторінку і відправляє її клієнту через мережу з використанням протокола HTTP [2].

Веб-додаток может бути клієнтом інших служб, наприклад, бази даних або іншого веб-додатку, розташованого на іншому сервері [2].

Останнім часом набуває популярності новий підхід до розробки веб-додатків, який називається Ajax. Сторінки веб-додатка не перезавантажуються повністю, а лише завантажують з сервера ЗМІНИ, що робить їх більш інтерактивними і продуктивними [2].

Особливості веб-додатків:

- **Незалежність від операційної системи клієнта**. Додаток створюється один раз для довільно вибраної платформи і на ній розгортається. Проте різна реалізація HTML, CSS, DOM і інших специфікацій в браузерах може викликати проблеми при розробці веб-додатків і подальшої підтримки [2].
- **Простота використання**. Відвідувач сайта має протягом 10-15 секунд зрозуміти, як почати користуватися сайтом. Не сподівайтесь, що хтось читатиме інструкції або витратить 1-2 години на вивчення системи меню та команд. Успіх веба багато в чому зобов'язаний тому, що велика частина теоретичних побудов, присвячених гіпертексту, була відкинута на користь простих прагматичних рішень, які і послужили основою

ідеальної конструкції. RSS став, можливо, єдиним широко поширеним веб-сервісом саме тому, що він простий. А складні корпоративні набори все ще чекають своєї години [2].

- **Великий час реакції на дії користувача**. Від моменту натискання на посилання до появи результату часто проходить більше 1 секунди (а буває й більше). Тому людина встигає помітити затримку і навіть занудьгувати. Задача програміста — максимально скоротити час реакції додатка. Як мінімум, ПОЧАТОК відповіді має з'явитися менш ніж за 10 секунд [2].
- **Велика кількість користувачів**. В успішних веб-проектах на сайт приходять десятки користувачів на секунду. Тому варто замислитися над оптимізацією швидкості виконання програми.
- **Стійкість до атак**. Через відкритий доступ до керування web-додатком, кількість хакерських атак на такі програми дуже велика [2].
- **Гігантські масиви даних**. Бази даних з мільярдами записів перестали бути екзотикою зовсім недавно, і тому ще немає інструментів для ефективного пошуку та маніпуляції такими великими кількостями інформації [2].
- Ослаблений захист інтелектуальної власності. Системи, подібні вебу, RSS і AJAX, схожі тим, що особливих перешкод для їх повторного використання не існує. Велика частина корисного програмного забезпечення має відкриті тексти, а якщо і немає, то існує мало способів захистити свою інтелектуальну власність. Стандартна функція оглядача інтернет "преглянути вихідний код" дозволяє будь-якій людині скопіювати будь-яку веб-сторінку. RSS був спроектований для того, щоб користувач міг читати контент тоді, коли це зручно йому, а не постачальникові інформації. Найуспішніші веб-сервіси — це, як правило, такі

служби, які можуть бути змінені несподіваним для їх творців чином (some rights reserved) [2].

1.3 Переваги та недоліки веб-додатків

Web-додатки в порівнянні з локальними програмами мають наступні переваги:

- простота доступу до додатка — будь-яка людина, що має комп'ютер, підключений до мережі Інтернет, може використовувати web- додаток [3];
- простота розгортання (установки) — на відміну від локальних додатків web-додатки після завершення розробки не вимагають установки на комп'ютерах користувачів. Досить тільки повідомити їм URL-адресу програми. При зміні додатки все користувачі відразу починають працювати зі зміненою версією [3];
- наявність великої кількості навчених користувачів [3];
- високий рівень розвитку і надійності мережевих з'єднань і web-технологій [3].

Однак web-додатки мають і свої недоліки, перелічені нижче:

- "Слабопов’язана" архітектура web-мережи — відсутність підтримки стану сеансу роботи і затримка при перезавантаженні кожної сторінки. Кожна перевантаження (або оновлення сторінки) викликає помітну затримку, викликану необхідністю встановити HTTP- з'єднання, обробити запит на сервері, передати по мережі у відповідь HTTP-повідомлення і перезавантажити сторінку браузером. Це створює скачки та періодичною очисткою роботи користувача [3].
- Обмежений набір елементів управління для проектування форм додатки. Поточна версія мови HTML підтримує тільки

обмежений набір елементів управління (текстові елементи, радиокнопки, прапорці, списки, що випадають і командні кнопки) [3];

- Неузгоджені підходи до оформлення та способу взаємодії з користувачами. Хоча web-мережа надає розробникам і дизайнерам значну гнучкість і можливість проявити свої творчі здібності, що виходять в результаті цього розмаїття і неузгодженість призначених для користувача інтерфейсів і способів взаємодії з web-додатками часто створюють проблеми для користувачів. Це викликано тим, що користувачі стикаються з великою різноманітністю стилів візуальних інтерфейсів і способів взаємодії, кожен з яких пропонує свій словник об'єктів, дій і візуальних уявлень, змішаних в одному і тому ж додатку [3].

1.4 Архітектура веб-додатків

Архітектура веб-додатку описує розташування всіх компонентів веб-програми, а також акцентує взаємодію між різними компонентами програми, системами проміжного програмного забезпечення сторонніх виробників, веб-службами та базами даних. Так забезпечується моментальний знімок взаємодії між кількома програмами, що працюють разом для надання послуг кінцевим користувачам [4].

Архітектура програмного забезпечення включає всі високорівневі компоненти системи та взаємодію всередині них [4].

Дизайн програмного забезпечення — це проектування на рівні коду, спрямоване на розподіл бізнес-логіки додатків за різними модулями зі своїми специфічними цілями. Він допомагає в побудові бізнес-логіки додатку та управлінні нею [4].

Рис. 1.2 Схема архітектури веб-додатку

Основні поняття зі схеми описані нижче.

Користувач-агенти. Агенти користувача (користувач-агенти) — це в першу чергу інструменти, які допомагають користувачам взаємодіяти з сервером. Користувач переважно взаємодіє з сервером через веб-браузери — Google Chrome, Firefox, Microsoft Edge, Safari — або мобільні програми на платформах Android та IOS. Агент відправляє запит на сервер і отримує відповідь, яка потім показується користувачеві [4].

DNS. Сервери доменних імен – це щось на зразок адресної книги для веб-сайтів. Щоразу, коли користувач надсилає запит на веб-адресу за допомогою браузера, браузер використовує DNS, щоб знайти реальну адресу веб-сервера (IP-адреса). Браузеру необхідно з'ясувати, на якому сервері знаходиться веб-сайт, щоб він міг відправляти запити HTTP туди, куди потрібно [4].

Балансувальник навантаження. Балансувальник навантаження насамперед займається горизонтальним масштабуванням. Він надсилає

вхідні запити на один із декількох серверів, а потім надсилає користувачеві відповідь, яку отримує від цих серверів. Зазвичай сервери веб-додатків існують у вигляді кількох копій, що дзеркально відображають один одного, щоб забезпечити узгодженість та доступність. Балансувальник навантаження розподіляє завдання між ними. Він добре працює у тандемі з автоматичним масштабуванням серверів [4].

Віртуальна машина. В обчислювальній техніці віртуальна машина (VM) – це віртуалізація/емуляція комп'ютерної системи. Віртуальні машини засновані на комп'ютерних архітектурах та забезпечують функціональність, аналогічну фізичному комп'ютеру. Їх реалізація може містити спеціалізоване апаратне забезпечення, програмне забезпечення або їх комбінацію [4].

Веб-сервер. Веб-сервер обробляє запит користувача і надсилає відповідь, що містить дані (HTML, JSON, XML і т.д.) у браузер. Веб-сервери, розміщені на віртуальній машині, використовують обчислювальні потужності, що надаються віртуальними машинами для виконання своїх завдань. Веб-сервер зазвичай відноситься до внутрішніх інфраструктур, таких як база даних, сервер кешу, черга завдань та інші [4].

Бази даних. База даних надає інструменти для організації, додавання, пошуку, оновлення, видалення та виконання обчислень з даними. Як правило, існують реляційні бази даних — MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database, Microsoft Access і т.д., а також нереляційні бази даних — MongoDB, Dynamo DB та інші [4].

Служба кешування. Служба кешування забезпечує збереження даних, що дозволяє зберігати та шукати дані. Кешування дозволяє посилатися на попередній результат, щоб значно прискорити обчислення. Кеші застосовуються на різних технологічних рівнях, включаючи операційні системи, та в мережах, включаючи мережі доставки контенту (Content Delivery Network, CDN) та DNS, веб-додатки та бази даних [4].

Мережа доставки контенту. Мережа доставки контенту, або мережа поширення контенту (CDN), є географічно розподіленою мережею проксі-серверів і пов'язаних з ними центрів обробки даних. Мета полягає в тому, щоб забезпечити високу доступність та продуктивність за рахунок просторового розподілу сервісу щодо кінцевих користувачів. CDN обслуговують більшу частину інтернет-контенту, включаючи веб-об'єкти (текст, графіку та скрипти), завантажувані об'єкти (медіафайли, програмне забезпечення, документи), додатки (електронна комерція, портали), потокові медіа у прямому ефірі, потокові медіа за запитом та сайти соціальних мереж [4].

Зовнішнє сховище. Зовнішні сховища використовуються для зберігання ресурсів, включаючи зображення, відео, документи та інші великі файли. У багатьох випадках із цими файлами також виконуються операції. Приклад — Youtube автоматично конвертує відео, завантажені користувачами, у кількох роздільних здатнах. Facebook перетворює зображення та відео користувачів у формат, зручний для мобільних пристроїв. Оскільки зберігання цих даних на локальному диску віртуальної машини є дорогим і споживає великі обчислювальні ресурси, вони зазвичай зберігаються в зовнішньому сховищі. Хмарне сховище – один із найпопулярніших варіантів. Хмарне сховище — це модель хмарних обчислень, яка зберігає дані в Інтернеті через постачальника хмарних обчислень, який керує сховищем даних як послугою. Сховище поставляється на вимогу з максимальною пропускною спроможністю та витратами, а також унеможливлює купівлю власної інфраструктури зберігання даних та управління нею [4].

Веб-служби. Веб-служби надають загальну платформу, яка дозволяє кільком додаткам, побудованим різними мовами програмування, мати можливість взаємодіяти один з одним [4].

Веб-служби використовують SOAP (Simple Object Access Protocol, Протокол простого доступу до об'єктів) для надсилання даних XML між додатками. Дані надсилаються за запитом HTTP. Дані, які надсилаються з веб-служби до програми, називаються SOAP-повідомленням. Повідомлення SOAP — це не що інше, як XML-документ. Оскільки документ написаний мовою XML, клієнтська програма, що викликає веб-службу, може бути написана будь-якою мовою програмування [4].

Існує два основних типи веб-сервісів:

- Простий протокол доступу до об'єктів або веб-служби SOAP.
- Надсилання репрезентативного стану або веб-служби REST.

Сховище даних. Сховище даних — це система, що використовується для складання звітів та аналізу даних, і вважається основним компонентом бізнес-аналітики. Кожен сучасний додаток збирає, зберігає та аналізує дані. Сховища даних — це центральні сховища інтегрованих даних із одного або кількох розрізнених джерел. Вони зберігають поточні та історичні дані в одному місці, яке використовується для виконання бізнес-аналітики та створення звітів для підприємства [4].

Два основні підходи до побудови системи сховища даних – це вилучення, перетворення, завантаження (ETL) та вилучення, завантаження, перетворення (ELT) [4].

Згадані три процеси:

- Вилучення. Вилучення даних із кількох джерел;
- Перетворення. Перетворення даних, включаючи очищення, поділ та категоризацію;
- Завантаження Завантаження даних у сховище.

1.5 Відмінність веб-додатку від сайту

Веб-додаток (рис. 1.3) — це комп’ютерна програма або програмне забезпечення, доступ до якого можна отримати, завантаживши та

встановивши додаток або також через веб-браузер. Це як веб-сайт, але з інтерактивними елементами для залучення користувачів. Прикладами веб-додатків для початківців є Pinterest, Gmail, Udemy тощо.

Рис. 1.3

Типи веб-додатків

- Односторінкова програма: SPA — це статична веб-сторінка, яка об’єднує інтерактивні, цікаві та оновлювані елементи. Найкращим прикладом є ваша електронна поштова скринька, де бічна панель залишається статичною під час навігації сторінкою. Ви можете читати електронні листи, надсилати нові електронні листи та отримувати сповіщення на одній сторінці.
- Прогресивні веб-програми: PWA — це платформа, яка працює в Інтернеті, але зазвичай використовується для мобільних програм через особливості. PWA можуть працювати в автономному режимі та

надсилати push-сповіщення. Ви можете додати PWA на головний екран, що не є загальною функцією веб-сайтів.

- Веб-сайт — це набір взаємопов'язаних веб-сторінок, доступних в Інтернеті під доменним іменем. Веб-сайт має стандартний інтерфейс і дизайн. Користувач може отримати доступ до веб-сайту за допомогою URL-адреси, яку можна шукати у веб-браузері. Прикладами веб-сайтів для початківців є Walmart, eBay, WordPress тощо

Типи сайтів

- Динамічні веб-сайти: це складні веб-сайти. Вони забезпечують внутрішні взаємодії з користувачем. Для створення цих веб-сайтів використовуються передові мови та бази даних. Важливо пам'ятати, що різні фактори можуть впливати на вміст, який вони відображають.
- Статичні веб-сайти: кодуються за допомогою простих мов, таких як HTML, CSS і JavaScript. Однак відсутність бази даних унеможливлює взаємодію користувачів. Вони відображають однаковий вміст для всіх користувачів, які до них звертаються

Характеристики веб-додатку

- Вибираючи веб-додаток для свого бізнесу, ви повинні знати, які характеристики змушують вас вибрати веб-додаток:
- Веб-додаток має бути зручним для користувача, щоб користувачі могли легко осягати сайт і виходити з нього.
- Веб-додатки мають бути легкодоступними та безперебійно працювати на всіх пристроях.
- Воно повинно бути привабливим і привабливим. Вибір хороших і креативних шаблонів дизайну та кольорів, які представляють ваш бренд або те, для чого ви створюєте веб-програму, є важливим.
- Щоб залучити користувачів на сайт або веб-додаток, вони повинні завантажуватися швидко. Плавне та швидке функціонування вашого сайту допомагає користувачам легко взаємодіяти.

- Веб-додатки повинні забезпечувати безпеку даних. Якщо в пристрої користувача виникла якась помилка, дані слід зберегти та зберігати на зовнішньому сервері, щоб уникнути втрати даних.
- На відміну від програм на телефоні, веб-програми не потребують оновлення. Автоматичне оновлення веб-додатків.

Характеристики веб-сайтів

- Якщо ви вважаєте, що ваш бренд/бізнес потребує веб-сайту для присутності в Інтернеті, ви повинні мати на увазі наступні характеристики:
- Зручна навігація дійсно може покращити взаємодію з користувачем. Якщо веб-додаток забезпечує легку навігацію для переходу з однієї сторінки на іншу, це може дійсно підвищити ефективність сайту.
- Веб-сайт має бути адаптивним. Веб-сайт повинен належним чином працювати на всіх пристроях.
- Користувачам потрібна лише URL-адреса, щоб відкрити веб-сайт. Вони забезпечують легкий доступ для користувачів.
- Веб-сайти легко та швидко створювати, оскільки вони написані простими мовами, тому не потрібно багато зусиль і програмістів
- Редагувати та оновлювати веб-сайти легко, оскільки код простий і його можна змінювати.

Коли вибрати веб-сайт?

Різниця між веб-сайтом і веб-додатком – це те, до чого більшість користувачів сприймають легковажно, оскільки вони синоніми. Але для тих, хто не обізнаний, веб-сайт – це те, що робить ваш бренд сяючим в Інтернеті. Це допоможе вам просувати свій бренд, залучати потенційних клієнтів, створювати автентичність і стати єдиним у своєму роді. Веб-сайт

економічно ефективний і простий у використанні, доступ і впровадження. Часто вони навіть допомагають вам досягти довіри.

Коли вибрати веб-програму замість веб-сайту?

Веб-програми стали досить популярними завдяки цифровізації, оскільки користувачі хочуть, щоб усе було комфортно; веб-програми допомагають їм зберегти цей комфорт. Якщо ваш бізнес працює над тим, щоб полегшити життя користувача, тоді веб-додаток для вас. Веб-додатки легко отримати, підтримувати та оновлювати. Вони працюють майже у всіх браузерах і тому більш зручні для користувача. Користувачам не потрібно турбуватися про втрату даних, а збережені дані зберігаються та можуть бути відновлені в будь-який момент. Веб-програми є досить інтерактивними за своєю природою, тому вони можуть забезпечити вам або вашому бренду швидкий розвиток, якого ви прагнете досягти.

РОЗДІЛ 2. ПРИКЛАДИ ВЕБ-ЗАСТОСУНКІВ ДЛЯ УПРАВЛІННЯ КОМУНАЛЬНИМИ ПОСЛУГАМИ

2.1 Веб застосунок 104.ua

Сайт дозволяє виконати користувачу наступні дії:

- купити газове обладнання;
- передати показання;
- сплатити газ і його доставку;
- зареєструватись;
- контролювати розрахунки.

При авторизації користувач має мати при собі особовий рахунок, згідно якого буде ідентифікуватися оплата. В особистому кабінеті можна спостерігати за балансом рисунок 2.1, за споживанням рисунок 2.2, за газом рисунок 2.3 та доставною рисунок 2.4

Рис. 2.1 Баланс

Рис. 2.2 Споживання

Рис. 2.3 Газ

Рис. 2.4 Доставка

На сайті присутня підтримка користувачів. З одного акаунта 104.ua можна керувати як особистим, так і бізнес-кабінетом. Одночасно в акаунті може бути до 25 особових рахунків [6].

2.2 Веб-застосунок ienergo.com.ua

«Особистий кабінет споживача електроенергії» є точкою доступу до персоніфікованої інформації і призначений для надання сервісу з віддаленої роботи із оперативними даними про Особовий рахунок Споживача, в точу числі про платежі, що надійшли; показники лічильників; виставлення рахунків та оплати спожитої електричної енергії [7].

На цьому сайті також при реєстрації потрібно вказати особистий рахунок. Зайшовши в особистий кабінет можна спостерігати за сумою до сплати рисунок 2.5, статус споживача рисунок 2.6 та діючий тариф [7].

Рис. 2.5 Сума до сплати

Рис. 2.6 Статус споживача

Рис. 2.7 Діючий тариф

Загальна інформація.

Рис. 2.8 Оператор системи розподілу

Оператор системи розподілу (АТ «Хмельницькобленерго») здійснює виключно функції розподілу електричної енергії, в тому числі прийом показів лічильників, облік електричної енергії, формування рахунків за

реактивну енергію та розподіл, якщо юридичний споживач обрав схему самостійної оплати за розподіл в ОСР [7].

Рис. 2.9 Формування рахунків

Відповідно, Постачальник здійснює фактично продаж електричної енергії, визначає суму до сплати та формує рахунки на оплату спожитої електричної енергії, розподілу (для юридичних споживачів, що купують енергію у Постачальника за вільними цінами та обрали оплату за розподіл через постачальника) [7].

Сервіс «Особистий кабінет споживача електроенергії» функціонує із двома компаніями: з ОСР та Постачальником. Тобто він є точкою взаємодії споживача і з ТОВ «Хмельницькенергозбут» і з АТ «Хмельницькобленерго», адже за допомогою сервісу споживач має змогу подати покази лічильника/здати рапорт, проаналізувати стан рахунку, сформувати відповідні рахунки та оплатити [7].

І це все відбувається без відвідування сервісних центрів та пунктів

РОЗДІЛ 3. СТВОРЕННЯ ВЕБ-ЗАСТОСУНКУ CITY INFORMER

3.1 Принцип роботи сіті інформера

City Informer має складові із двох частин: серверної (Backend) та клієнтської частини (Frontend). Frontend використовує мову програмування TypeScript, яка просто є доповненням мови JavaScript типізуючи її. Також було використано фреймворк Angular з його вбудованими бібліотеками. Backend написаний на мові програмування Node.js, яка по суті те саме, що JavaScript, але призначена виключно для розробки декстопних та серверних додатків. Тут було використано фреймворк Express.js. Backend та Frontend взаємодіють у форматі REST API, тобто передають дані один одному через JSON формат. Наприклад: користувач заходить на будь-яку сторінку веб-додатку, цим самим відправляє запит від фронтенду до бекенда. У свою чергу бекенд обробляє дані та повертає відповідь фронтенду.

Вводячи в адресну строку браузера доменне ім’я веб-додатку https://city-informer.ua, користувача одразу перекидає на головну сторінку. Тут йому пропонується декілька операцій щодо користуванням цим веб-додатком:

- реєстрація незареєстрованих користувачів;
- перегляд усіх поданих заяв;
- подача заяви, скарги щодо виявленої проблеми.

Якщо користувач авторизується як адміністратор, то при переході на сторінку із усіма доступними заявами, йому надається доступ модифікації. Тобто він може вказати чи дана заява прийнята до роботи або вже є виконаною.

При подачі скарги потрібно вказати вулицю, час, тип виявленої проблеми та фотографію. Далі вказані дані передаються серверній частині, яка зберігає скарги в базу даних. Там формується черга, судячи з якої

комунальні служби будуть їх виконувати почергово . В кінцевому рахунку вони зв’зуються з адміністратором, який модифікує певним чином.

3.2 Архітектура сіті інформера

Як згадувалося раніше фронтенд та бекенд взаємодіють між собою по технології REST API. Фронтенд надсилає запит бекенду та отримує відповідь у форматі JSON. Нижче на рисунку 3.1 продемонстровано, як сервер повертає скаргу в цьому форматі.

```
[
  {
    "type": "сніг",
    "description": "Доброго дня. Сьогодні на вулиці...",
    "author": "Шевченко Вадим",
    "imageUrl": "...",
    "status": "виконано"
  }
]
```

Рис. 3.1 JSON-формат скарги

Такі дані фронтенд перетворює в масив об’єктів JavaScript. Далі з такими фундаментальними типами даних можна легко маніпулювати. В Angular все побудовано на класах, екземпляри яких відображають властивості у шаблоні для виведення даних, тому потрібно перетворити масив об’єктів у властивість класу. Тут відображено найпростіший приклад маніпулювання даними.

Для відображення сторінок у City Informer застосовється технологія маршрутизації. Маршрутизація дозволяє зіставляти запити до програми з певними ресурсами всередині програми. Це означає, що відображення сторінок буде залежити від поставленого маршруту в адресній строці. Наприклад: при **https://city-informer.ua/home** буде відобрататися компонент, який безпосередньо відповідає за відображення домашньої

сторінки, а при **https://city-informer.ua/reports** буде виведення інформації щодо сторінки зі скаргами.

3.3 Розробка дизайну веб-застосунку

Перше, що є, це «шапка» сайту, яка виглядає наступним чином.

Рис. 3.2 Шапка сайту

Вона ділиться на три частини:

- заголовок сайту з посиланням на головну сторінку;
- пошукове поле;
- кнопка з діалоговим вікном реєстрації.

Оскільки користувач потрапивши на сайт, потрапляє на домашню сторінку, доцільно було б розпочати з неї. Тут нічого незвичайного

Рис. 3.3 Домашня сторінка

Натиснувши кнопку зі значком користувача, що знаходиться в шапці, маємо діалогове вікно реєстрації. Тут присутні декілька полів вводу

інформації та дві кнопки. Перша відправляє форму на сервер та друга, яка очищає введені раніше дані з полів вводу

Рис. 3.4 Форма реєстрації

Також можна показати сторінку подачи заяви. У ній присутні такі компоненти як поле для вводу:

- вуліці, на якій сталася пригода;
- орієнтовний час виникнення проблеми;
- характер та опис проблеми;
- вібір типу проблеми;
- раніше згадані дві кнопки для відправки та очищення форми.

Для вибору типу проблеми надається перелік самих розповсюджених

Рис. 3.5 Форма подачі заяви

Рис. 3.6 Перелік розповсюджених проблем

Одною з найголовніших сторінок є сторінка із скаргами. У ній вказано:

- номер заяви;
- заголовок;
- час подання заяви;
- статус.

Рис. 3.7 Сторінка з переліком поданих скарг

При авторизації користувачем як адміністратором, відкривається доступ до модифікації скарг. Можна перейти на сторінку окремої скарги для вказання скатусу: прийнято до роботи або виконано. Наглядно можна подивитися на рисунку 3.8.

Рис. 3.8 Карта окремої скарги

На карті видно людину, на якій видно, що вона рареєстрований користувач. Присутній довільний аватар та головне — фото, опис скарги. Для адміністратора додано інтерфейс для управління статусу. Це дві кнопки, які помічають як виконану або як прийняту до роботи скаргу.

3.4 Написання коду в Angular

Прийшов час розповісти чому я обрав саме Angular фреймворк. Потрібно розуміти, що існують ще два найпопулярніших фреймворка, такі як Vue та React. За архітектурою Angular є повноцінним MVC фреймворком, який надає розробникові всі можливості для програмування “з коробки”: шаблони, засновані на HTML; впровадження залежності (Dependency injection); запити Ajax; роутінг; інкапсуляція компонентів CSS; утиліти для тестування компонентів; можливості для створення форм тощо. У Angular розмітка документу (шаблони HTML) і логіка (JavaScript) фізично розділені на різні файли. У Angular присутнє таке поняття як компонетти. Це окрема логічна частина веб-сайту, яка має при собі власну область для стилів, логіки та HTML розмітки. Хоч компоненти є фундаментальною частиною Angular, але є сутності не менш важливі, такі як модулі. Модулі це сутності, які можуть містити в собі компоненти. Можна сказати, що модулі це великі окремі логічні частини. Для того, щоб згенерувати компонент, модуль чи іншу сутність Angular, існує Angular CLI. Це консольний додаток, який дозволяє генерувати вище сказані сутності. Це досить зручно, тому що не потрібно робити це вручну.

```
$ ng g c my-component --skip-tests
```

Рис. 3.9 CLI команда для генерації копонента без тестів до нього

Після введеної команди створилась папка, у якій в середнені файли стілів, логіки та розмітки.

Рис. 3.10 Відображення компоненту у файловій системі

У Angular будь-яка сутність це TypeScript клас. До цього класу застосовуються такі мовні конструкції, як декоратори, задля внесення метаданих.

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-my-component',
  templateUrl: './my-component.component.html',
  styleUrls: ['./my-component.component.scss']
})
export class MyComponentComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }
```

Рис. 3.11 Клас за замовчуванням

```
import {Component, OnInit} from '@angular/core';
import {MatDialog} from "@angular/material/dialog";
import {RegistrationFormDialogComponent} from "../registration-form-dialog/registration-form-dialog.component";

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.scss'],
})
export class HeaderComponent implements OnInit {

  searchValue: string = '';

  constructor(public dialog: MatDialog) {}

  ngOnInit(): void {
  }

  openDialog(enterAnimationDuration: string, exitAnimationDuration: string): void {
    this.dialog.open(RegistrationFormDialogComponent, {config: {
      width: '500px',
      enterAnimationDuration,
      exitAnimationDuration,
    });
  }
}
```

Рис. 3.12 Клас шапки сайту

```
<header class="header">
  <mat-toolbar color="primary" class="header__row">
    <div class="header__column">
      <button mat-icon-button class="example-icon" aria-label="Example icon-button with menu icon">
        <mat-icon>home</mat-icon>
      </button>
      <span class="header__title">City Informer</span>
    </div>

    <div class="header__column">
      <mat-form-field>
        <mat-icon matPrefix>search</mat-icon>
        <input matInput type="search" [(ngModel)]="searchValue" class="header__input">
      </mat-form-field>
    </div>

    <div class="header__column">
      <div class="header__icons">
        <button (click)="openDialog( enterAnimationDuration: '0ms', exitAnimationDuration: '0ms')"
                mat-icon-button class="example-icon"
                aria-label="Example icon-button with share icon">
          <mat-icon>account_circle</mat-icon>
        </button>
      </div>
    </div>
  </mat-toolbar>
</header>
```

Рис. 3.12 Шаблон шапки сайту

Видно, що тут використовуються не тільки звичайні HTML теги, ай готові компоненти бібліотеки Angular Material. Майже завжди вигідно використовувати готові компоненти. Це зменшує затрати часу та містить в собі вже готове рішення.

```
:host ::ng-deep {
  mat-form-field {
    width: 100%;

    .mat-input-element {
      font-size: 16px;
      background-color: white;
      color: black;
      border-radius: 5px;
      padding: 0 10px 0 50px;
      height: 35px;
    }

    .mat-form-field-underline {
      display: none;
    }

    .mat-form-field-wrapper {
      padding-bottom: 15px;
    }

    .mat-form-field-prefix {
      position: relative;
      top: -3px;
      right: -40px;
      z-index: 3;
      color: gray;
    }
  }
}
```

Рис. 3.13 Стилі шапки сайту

Якщо коротко, то компоненти мають методи та властивості. Методи використовують сервіси для отримання даних, а ті в свою чергу записуються у властивості компонента. За допомогою властивостей відображаються дані у шаблоні.

ВИСНОВКИ

В ході виконання магістерської роботи були виконані наступні завдання дослідження:

- ☑ провів огляд веб-додатку;
- ☑ розглянув поняття City informer;
- ☑ створив веб-додаток City informer використовуючи фреймворк Angular та мову TypeScript.

В ході виконання виконання поставлених завдань, я зрозумів, що застосування City Informer є зручним для користувачів. Цей веб-додаток спрощує взаємодію користувачів та сферу комунальних послуг в межах міста. Також цей веб-додаток можна розширювати та додавати новий функціонал.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке веб-додаток [Електронний ресурс] — Режим доступу до ресурсу: https://bit.ly/3WWWJCH;
2. Особливість веб-додатків [Електронний ресурс] — Режим доступу до ресурсу: https://bit.ly/3GgfgUO;
3. Переваги та недоліки веб-додатків[Електронний ресурс] — Режим доступу до ресурсу: https://bit.ly/3WmPhAC;
4. Архітектура веб-додатків [Електронний ресурс] — Режим доступу до ресурсу: https://bit.ly/3WDPvTJ;
5. Відмінність веб-додатку від веб-сайту[Електронний ресурс] — Режим доступу до ресурсу: https://bit.ly/3BXa3Pd;
6. Огляд веб-застосунка 104.ua[Електронний ресурс] — Режим доступу до ресурсу: https://bit.ly/3VpO0Yi;
7. Огляд веб-застосунку ienergo.com.ua [Електронний ресурс] — Режим доступу до ресурсу: https://ienergo.com.ua/cabinet/docs;
8. Документація по Angular [Електронний ресурс] — Режим доступу до ресурсу: https://angular.io/docs

ДОДАТКИ

Додаток А. Код Проекту

```
import {Component, OnInit} from '@angular/core';
import {ProductsService} from "../../services/products.service";
import {FilteredDetailOfProduct, ProductModel} from
"../../interfaces/product.model";
import {FormGroup} from "@angular/forms";

@Component({
  selector: 'app-gallery',
  templateUrl: './gallery.component.html',
  styleUrls: ['./gallery.component.scss']
})
export class GalleryComponent implements OnInit {
  public products: ProductModel[];
  public filteredProducts: ProductModel[];
  public loading: boolean = true;

  private emptyFilters: boolean = true;
  private mapOfFilteredProducts: FilteredDetailOfProduct[];

  constructor(public productsService: ProductsService) { }

  initProducts(): void {
    this.productsService.getProducts().subscribe(value => {
      this.products = value;
      this.filteredProducts = this.products;
      this.loading = false;
```

```
    })
  }

  getMapOfFilteredProducts(form: FormGroup) : FilteredDetailOfProduct[]
{
    const fn = ([detailName, detailValue]) : FilteredDetailOfProduct => {
      const filteredMap = [];

      Object.entries(detailValue).forEach((item) => {
        const [key, value] = item;

        if (value) {
          filteredMap.push(key);
          this.emptyFilters = false;
        }
      });

      return [detailName, filteredMap];
    }

    return Object.entries(form.value).map(fn);
  }

  changeFilteredProducts(mapOfProducts): void {
    if (this.emptyFilters) {
      this.filteredProducts = this.products;
      return;
    }
```

```
    const checkByFieldOrTypeCondition  = (product) => {
      for (let [key, value] of mapOfProducts) {
        const condition = product.details[key] !== undefined &&
(<string[]>value).includes(product.details[key].toString());

        if (condition) return true;
      }
    }

    this.filteredProducts =
this.products.filter(checkByFieldOrTypeCondition);
  }

  onFilterChanges(form: FormGroup) : void {
    this.mapOfFilteredProducts = this.getMapOfFilteredProducts(form);
    this.changeFilteredProducts(this.mapOfFilteredProducts);
    this.emptyFilters = true;
  }

  ngOnInit(): void {
    this.initProducts();
  }
}
```