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

Дипломна робота
бакалавра
з теми: «РОЗРОБКА ВЕБ ЗАСТОСУНКУ «РОЗРАХУНОК БУДІВЕЛЬНИХ МАТЕРІАЛІВ»»

Виконав студент 4 курсу,
групи KN1-B18
спеціальності 122 Комп’ютерні науки
Ткачук Валерій Віталійович

Керівник:
Слободянюк О.В., кандидат технічних наук, доцент

Кам’янець-Подільський – 2022

ЗМІСТ

ВСТУП

Кожна людина, яка хоч раз у житті стикалася з ремонтом чи будівництвом, знає, наскільки важливо точно та правильно прорахувати матеріали. Іноді ціна на ремонт у квартирі навіть незначної помилки при будівництві чи ремонті може виявитися надто високою. Однак, якщо людина не знаєте всіх нюансів побудови, не пам'ятаєте або не вмієте користуватися алгебраїчними формулами тоді і постає питання: «Як це прорахувати?». Веб застосунок «Розрахунок будівельних матеріалів» безкоштовно допоможе користувачу у попередньому розрахунку вартості на ремонт квартири, дозволить приблизно зорієнтуватися з розцінкою робіт, виходячи із зазначених параметрів.

Метою даної роботи є розробка веб застосунку для розрахунку будівельних матеріалів за допомогою бібліотеки React.

Для дослідження поставленої цілі в дипломній роботі необхідно вирішити наступні задачі:

1. Розглянути поняття веб застосунків, види, структура, історія та їхня роль в сьогоденні.
2. Висвітлити принципи роботи веб застосунків та нові бібліотеки для розробки веб додатків.
3. Показати причину популярності бібліотеки React сьогодні.
4. Розглянути механізми роботи та принципи створення React додатків.
5. Вилучити актуальність розробки веб застосунку «Розрахунок будівельних матеріалів»
6. Висвітлити принципи роботи веб застосунку, рекомендації до використання користувачем.

Об’єктом даної роботи є проектування та веб застосунку «Розрахунок будівельних матеріалів».

Предметом даної роботи є технології створення веб додатків.

РОЗДІЛ 1. ПОНЯТТЯ ВЕБ ЗАСТОСУНКУ

1.1 Види, структура і роль веб технологій

Веб-сервіс - це програмна система, призначена для підтримки взаємодії комп'ютера з комп'ютер через Інтернет. Веб-сервіси не є новими і зазвичай набувають форми інтерфейсів прикладного програмування. В даний час екстремальної конкуренції в бізнесі, в освіті, промисловості обмін інформацією та ефективне спілкування – це потреба кожного дня. Всесвітня павутина є важливим ресурсом для всіх сфер життя: освіта, зайнятість, уряд, торгівля, охорона здоров'я, відпочинок та інші. Web - система взаємопов'язаних гіпертекстових документів, доступних через інтернет. За допомогою веб-браузера користувач переглядає веб-сторінки, які можуть містити текст, зображення, відео, інші мультимедійні матеріали та перехід між ними за допомогою гіперпосилань. [1].

Всесвітня павутина була створена 1989 року сером Тімом Бернерсом-Лі, який працював у ЦЕРН (Європейська організація з ядерних досліджень) у Женеві,

Швейцарії. З того часу Бернерс-Лі грав важливу роль у посібнику розробкою веб-стандартів і в останні роки відстоює своє бачення семантичної мережі [2].

Web 1.0 була епохою, коли Netscape був єдиним браузером довгий час. Web 2.0 це час, коли люди стали розуміти, що це програмне забезпечення не дозволяє використовувати Інтернет, який має велике значення. Нові технології дозволяють здійснювати інтелектуальний пошук та ведуть до Web 3.0 [3]. У часи Web 2.0 бачення мережі, якою інформація розбивається на блоки «мікроконтенту», які можуть бути розподілені за десятками доменів. Таким чином, мережа документів перетворилася у мережу даних.

У Web 1.0 невелика кількість людей створювала веб-сторінки для великої кількості читачів. В результаті користувачі могли отримати інформацію, перейшовши безпосередньо до джерела. WWW або Web 1.0 є системою взаємопов'язаних гіпертекстових документів, доступ яких здійснюється через мережу Інтернет. Перша реалізація Web 1.0, згідно з Бернерс-Лі, могла розглядатися як "сайт, доступний тільки для читання". Іншими словами, ранній веб дозволяв користувачам шукати інформацію та читати її. Це був найпростіший спосіб взаємодії між користувачем та контентом. Однак це саме те, що більшість власників сайтів хотіли: веб-сайти повинні були надавати необхідну інформацію користувачам у будь-яке час.

Web 2.0 - термін, введений ДіНуччі в 1999 році пізніше в 2004 році він був змінений Тімом О'Рейлі та Дейлом Дугерті, які повідомили, що друге покоління www (World Wide Web) орієнтоване на можливість користувачів вносити інформацію в режимі онлайн за допомогою веб-спільнот, соціальних мереж тощо. Web 2.0 є динамічним та більш інтерактивним веб-знанням та спрямованим на створення статичних HTML-сторінок. Це також означало більш затребуване видання Web, де нові інструменти дозволили практично будь-якій людині зробити свій внесок, незважаючи на їх практичні знання. В даний час Web 2.0 знаходиться на початковому етапі, так звана мережа «читання-запис», на думку Бернерса Лі. Можливість змінювати контент і взаємодіяти з іншими веб-користувачами за короткий час різко змінила орієнтацію мережі Інтернет. Метою створення Web 2.0 була освіта покращеної форми WWW. Такі технології, як блоги, вікі, підкасти, соціальні мережі, веб-API та вебслужби (eBay та Gmail) показують значні переваги перед звичайними сайтами Web 1.0, які використовувалися лише читання [4].

Тім О'Рейлі стверджував, що Web 2.0 - це, безумовно, перспективний напрямок у WWW, в якому використовуються новітні технології та концепції для того, щоб зробити користувача більш інтерактивним, корисним та залученим. Це має на увазі ще один спосіб з'єднання світу шляхом збору інформації та забезпечення ефективного обміну даними.

Таблиця 1.1 – Інструменти Web 2.0

Інструменти Web 2.0	Характеристика
Блог	Блог - дивовижний апарат, заснований на двосторонньому листуванні. Блог - сайт, де користувачі можуть писати свої думки, рекомендації, вносити пропозиції. Записи перераховані у певних категоріях, які можна шукати.
Вікі	Слово "вікі" означає "швидко". Wiki - це сторінки веб-сайту або набір сторінок, які можуть бути змінено будь-якою людиною, якій дозволено доступ до Інтернету.
Соціальна мережа	Соціальна мережа - тип веб-сервісів, у яких користувач може взаємодіяти та спілкуватися друг з другом по всьому світу. Є багато сайтів соціальних мереж (SNS). Facebook - сайт соціальної мережі, де люди взаємодіють із друзями, родичами, колегами, учителями і так далі.

	Твіттер - дуже популярний представлений мікроблогами веб-сайти, головним чином знаменитостей, політиків, бізнес-лідерів та інших. Твіттер дозволяє користувачам писати короткі повідомлення, які помітні для інших користувачів.
Подкаст	Термін «подкаст» було визначено Дейвом Вінером та Адамом Каррі у 2004 році та їх підписчиків було більше 1 мільйона за перші два дні. Подкаст є поєднанням двох слів трансляція і iPod, що означає аудіофайл, доступний у мережі Інтернет. У подкастінг інструменти, аудіо синдикуються через RSS (Really Simple Syndication).
Мешап	Web Mashups нагадує веб-сайт або веб-сторінку, яка об'єднує дані та адміністрацію з різних джерел у мережі Інтернет. Це сукупність інформації з різних веб-сайтів у Мережі. Web

	Mashup розділили на сім категорій: месенджер, пошук, мобільний, шопінг, картограф, кіно та спорт.

Web 3.0 - це термін, який використовується для опису різних змін у використанні Web і взаємодії кількома шляхами. До них відноситься перетворення Інтернету на базу даних, перехід до створення контенту, доступного для кількох додатків, що не використовують браузер, використання технологій штучного інтелекту, семантичної мережі, геопросторової мережі або 3D-мережі. Тім Бернерс-Лі придумав Giant Global Graph (GGG) як ще один аспект Web 3.0

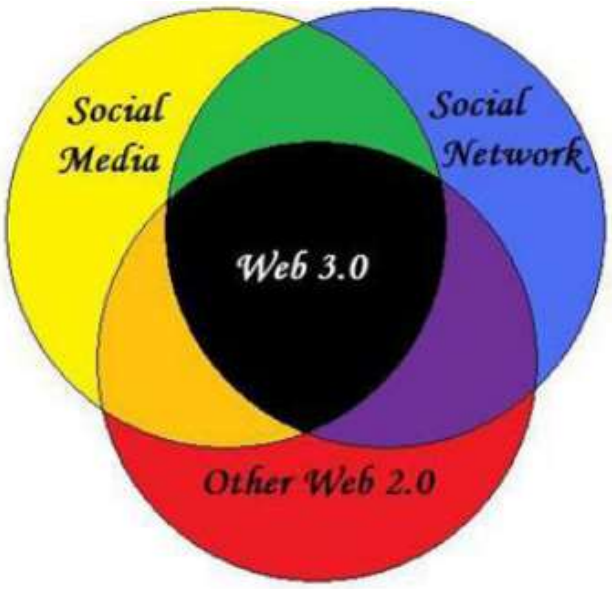


Рис. 1.1 Структура Web 3.0

Основні особливості Web 3.0. Семантичне павутиння дозволить машинам розуміти інформацію. Таким чином, семантична мережа значною мірою полегшить процес прийняття рішень споживачами шляхом надання більш актуальної інформації.

Основна застосовність може бути в рамках пошукових систем. Така семантична пошукова система здійснюватиме пошук не пропозиції, рядка, а розуміти весь синтаксис пошукового запиту, його концепцію. Великою інновацією в наступні роки буде також прогресивне підключення комп'ютерів, багатьох фізичних об'єктів, що породжують створення Інтернету речей, концепція якого передбачає повсюдне використання інтелектуальних пристроїв та об'єктів. Вони здатні взаємодіяти та співпрацювати з іншими пристроями за допомогою дротових та бездротових з'єднань та мати унікальні схеми адресації для створення сервісів та програм, адаптованих до потреб користувачів. Таким чином, ми вступаємо в нову еру, в якій Інтернет речей замінить традиційну мережу Інтернет, яку ми знаємо сьогодні, що призводить до зниження витрат і підвищення ефективності використання наявних ресурсів. Спільно з Інтернетом речей штучний інтелект та 3D-графіка сприяють тому, що комп'ютери будуть поводитися як люди і давати швидкі прогресивні вирішення багатьох проблем.

1.2. Принципи роботи веб застосунків

Коли справа доходить до розробки додатків для Інтернету, існує дві основні методики, що потрібно розуміти: на стороні сервера та на стороні клієнта. У серверних методологіях програма, яка обробляє дані, працює на сервері з використанням таких мов, як PHP, ASP, і останнім часом JavaScript. У клієнтських методологіях додаток працює в межах браузер за допомогою JavaScript. Багато веб-додатків використовують комбінацію обох серверів і методології на стороні клієнта

Коли користувач відкриває свій веб-браузер ("клієнт") і запитує веб-сторінку через гіпертекст Протокол передачі (HTTP), ввівши Uniform Resource Locator (URL), сервер надає сторінку як комбінацію ресурсів, включаючи файли HTML, файли JavaScript, CSS (каскадна таблиця стилів) файли, файли зображень та аудіофайли.

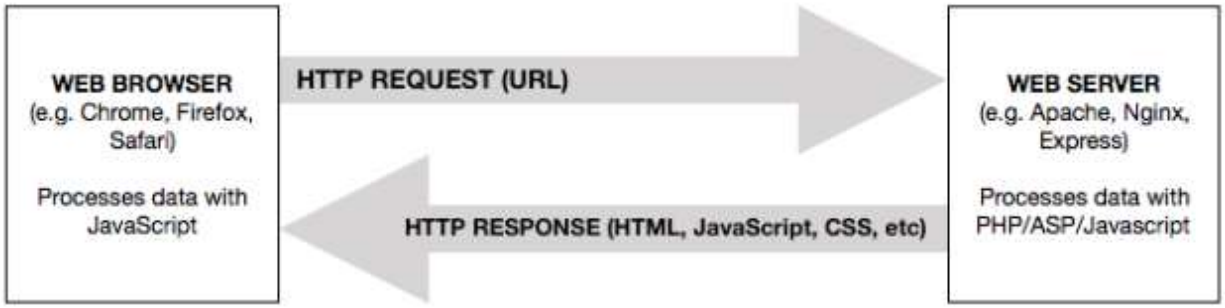


Рис. 1.2 HTTP-запит і процес відповіді між клієнтом і сервером

Загалом, файли HTML визначають структуру веб-сторінки, файли CSS визначають зовнішній вигляд, і файли JavaScript визначають логіку. Більшість веб-сторінок також використовують зображення та звук багатий інтерфейс користувача. Веб-додаток - це специфічний тип веб-сторінки, що забезпечує подібне функціональність та досвід роботи з традиційними настільними додатками.

Безперервний розвиток технологій та технічних характеристик мережі Інтернет протягом двох десятиліть її існування забезпечило плавний перехід у способі представлення інформації - від тривіальної презентації простих статичних гіпертекстових документів до найскладніших інтерактивних мультимедійних інформаційних систем Методологія програмування для Інтернету постійно вдосконалюється, наростаючими темпами з'являються нові засоби розробки.

Різноманітність сучасних веб-додатків та необхідність розробки ефективних інтерфейсів, функціональні можливості яких зростають з кожним днем, а також необхідність підтримки їхньої стабільної роботи не тільки в останніх версіях популярних веб-браузерів, але і в застарілих (але при цьому ще використовується) версіях - все це робить розробку на JavaScript без використання допоміжних засобів надзвичайно неефективною та стомлюючою. Досі існує проблема різної поведінки різних типів браузерів (так званих двигунів - Trident, Gecko, WebKit, Presto) в області інтерпретації JavaScript, підтримки об'єктної моделі документа (DOM) та каскадних таблиць стилів (CSS).

Починаючи з середини 2000-х. серед розробників клієнтської частини веб-додатків все популярніше стають так звані JavaScript-фреймворки. JavaScript-фреймворк (або бібліотека JavaScript) це набір функцій та засобів, що полегшують реалізацію сценаріїв JavaScript, сумісних із більшістю сучасних браузерів. Кожна бібліотека в процесі свого розвитку проходить активне тестування в різних веб-браузерах, і це певною мірою гарантує однакову роботу веб-додатків на основі цих бібліотек у різних типи браузерів (кросбраузерність).

Основні функціональні можливості типового сучасного JavaScript-фреймворку

Операції із елементами об'єктної моделі документа (DOM). Як правило, в основі бібліотек міститься реалізація функцій пошуку та вибірки елементів DOM з використанням селекторів CSS (Cascading Style Sheets, каскадні

таблиці стилів) або XPath (XML Path Language). Даний функціонал дозволяє отримувати посилання на елементи моделі документа згідно з ідентифікаторами цих елементів, їх класів, значень атрибутів, а також з використанням різноманітних комбінацій цих властивостей.

Крім вибірки елементів з використанням селекторів, JavaScript-бібліотеки часто надають функціонал пошуку та вибірки елементів DOM відповідно до їх відносної позиції у дереві документа, тобто з використанням наступних відносин: батьківський елемент, дочірній елемент, елементи предки, елементи-нащадки та сусідні елементи.

Природним продовженням розвитку функціоналу фреймворків JavaScript, пов'язаного з пошуком та вибіркою елементів, є реалізація функцій, призначені для маніпуляцій з моделлю DOM. Це функції зміни значення елементів, атрибутів, класів CSS, вмісту; їх переміщення, копіювання, видалення тощо. Використання функцій, пов'язаних з об'єктною моделлю документа, є однією з основних незаперечних переваг у порівнянні з реалізацією коду на «чистому» JavaScript, оскільки реалізація DOM досі не є повною в різних версіях та типах браузерів.

Підтримка асинхронних запитів HTTP (AJAX). Використання концепції AJAX (Asynchronous JavaScript і XML) вже неодноразово довело свою ефективність при реалізації веб-додатків. Подібне об'єднання технологій динамічного обігу до сервера (найчастіше за допомогою об'єкта XMLHttpRequest) та DHTML (Dynamic HTML) для організації інтерактивних веб-інтерфейсів має наступні переваги: воно зменшує навантаження на сервер і дає можливість заощаджувати трафік за рахунок істотного зменшення обсягів даних, що передаються, і перенесення частини навантаження на сторону веб-клієнта, а також підвищує інтерактивність інтерфейсів і їх прискорення

Функціонал для роботи з AJAX у JavaScript фреймворках є набором зручних у використанні методів, що дозволяють вибрати тип відповіді (XML, JSON, HTML або текст), призначити функції, що запускаються при успішному та невдалому запит, вибрати метод запиту (GET або POST), налаштувати

синхронну передачу даних або асинхронну і т. д. Можливості конфігурування відрізняються у різних бібліотек, але вони полегшують роботу з AJAX, істотно скорочуючи обсяги коду, збільшуючи його читання та прозорість, а також позбавляють необхідних перевірок на сумісність із поточним браузером, оскільки, наприклад, браузер Internet Explorer версії 6 використовує ActiveX замість об'єкта XMLHttpRequest. Крім того, існує різниця в реалізації XMLHttpRequest в інших типах браузерів.

Опрацювання подій. Традиційний спосіб призначення подій браузера шляхом впровадження в HTML код елемента спеціальних атрибутів є дуже неефективним, особливо з огляду на функціональну складність інтерфейсів сучасних веб-додатків. Крім того, частково існує проблема кросбраузерності. Практично у всіх сучасних JavaScript-фреймворках реалізований власний функціонал обробки подій, який є простою та зручною альтернативою стандартному способу.

Крім безпосереднього призначення обробників подій клієнта (таких, як клік мишею) по елементу, подвійний клік, наведення курсору, переміщення миші тощо), елементам, вже створеним у моделі документа, сучасні фреймворки дозволяють здійснювати делегування подій. Делегування подій у JavaScript – це спосіб призначення елементу DOM обробки події шляхом передачі керування від обробника батьківського елемента. Іншими словами, функції-обробники призначаються не самим елементам, а їхнім батькам, що наводить до сприятливих наслідків:

– у коді стає менше функцій, якими потрібно оперувати;
– займається менше пам'яті;
– зменшується кількість зв'язків між JavaScript-кодом та DOM;
– зникає необхідність видалення непотрібних подій під час зміни DOM;
– стає можливим наскрізне додавання подій до елементів DOM, що створюються на льоту.

Майже кожен JavaScript-фреймворк має набір допоміжних (службових) функцій, які покликані полегшити розробку. Часто серед таких функцій можна побачити:

- Визначення «движка» та версії браузера;
- Обхід масиву з виконанням функції на кожній ітерації;
- Пошук значення в масиві чи об'єкті;
- Об'єднання масивів або об'єктів;
- обробку рядків (синтаксичний аналіз тощо) та ін.

Крім описаних вище можливостей, бібліотеки JavaScript часто мають можливість розширення за допомогою модулів або плагінів. Поряд із розширеннями від розробників бібліотек, існує безліч сторонніх проектів. Оскільки більшість із JavaScript-фреймворків поставляється з відкритим вихідним кодом за ліцензією вільного/відкритого ПЗ (BSD, GPL або MIT), з плином часу навколо цих продуктів зазвичай виникають спільноти активних веб-розробників, багато з яких створюють власні модулі та викладають їх у вільний доступ.

Загалом використання бібліотек JavaScript дає Великі переваги розробки веб-додатків. Серед очевидних плюсів – зменшення обсягу коду, підвищення зручності читання, підтримка кросбраузерності. На додаток до перерахованого, розробник отримує можливість правильно структурувати програмний код, оскільки необхідність фізично змішувати HTML та JavaScript практично зникає.

Найпопулярнішими на сьогоднішній день JavaScript-фреймворку є Dojo, Mootools, YUI, ExtJS, JQuery

Кожен із фреймворків є потужним інструментом для веб-розробника, зі своїми позитивними та негативними сторонами. Всі вони мають підтримку розглянутого набору функціональних можливостей – асинхронної передачі даних AJAX, обробки подій та ін. Крім базової функціональності, яка загалом збігається у всіх перерахованих бібліотек, що розглядаються JavaScript-фреймворки відрізняються наявністю і можливостями модулів і елементів, що

підключаються. веб-інтерфейс (віджетів). Наприклад, ExtJS володіє потужним набором функцій для створення графічного інтерфейсу користувача веб-додатку, так званого віконного інтерфейсу на веб-сторінці у стилі настільної операційної системи.

1.3. Технології розробки веб застосунків з використанням сучасних бібліотек

Мова JavaScript впевнено набирає популярності серед frontend-розробників. За допомогою нього можна реалізовувати цікаві рішення інтерфейсів, створювати інтерактивні та ергономічні сайти, а також розробляти веб-програми більш просто і зрозуміло.

Існує безліч фреймворків, які регулярно оновлюються та вдосконалюються. І вибрати найбільш підходящий фреймворк виявляється нетривіальним завданням.

На початковому етапі створення проекту доцільно провести аналіз, який допоможе вибрати потрібну технологію. Фреймворк — це платформа, яка визначає структуру веб-програми. Його функціонал дозволяє здійснити взаємодію веб-ресурсу із сервером.

На веб-сайтах, написаних без використання фреймворку, початковий контент зберігається на сервері. Тому при завантаженні нового матеріалу на сайт потрібне перезавантаження сторінки. Перевагою ж фреймворку є постійні блоки, які зберігаються від однієї Зміни до іншої. Це дозволяє добиватися миттєвого зворотного зв'язку з користувачем при додавання нового контенту.

Даний принцип можна спостерігати при створенні односторінкових веб-застосунків, електронної комерції, хмарних сервісів та багатьох соціальних мереж. Так, перехід користувача з одного розділу в іншій здійснюється миттєво. Перевантаження сторінки не відбувається, оскільки незмінно залишається каркас - стала частина проекту.

Переваги використання фреймворку:

- Фреймворки є повністю безкоштовними та мають відкритий вихідний код;
- Використання вбудованих шаблонів допомагає створювати проекти вищої якості, при цьому задіюється менше рядків коду;

- Висока швидкість розробки, яка досягається за рахунок відкритого доступу до документації та безлічі форумів.

За реалізацією проектів на класичному JS та HTML ховається багато труднощів, що призводить до появі великої кількості фреймворків на ринку. Але найпопулярнішими серед веб-розробників залишаються Angular, React та Vue

Angular - це кроссплатформенний фреймворк, розроблений компанією Google. Він дотримується суворої ієрархії і є великою інфраструктурою для комплексних рішень. Частина серверної служби фреймворку переноситься на клієнтську сторону, що зменшує навантаження на сервер.

Завдяки суворій типізації мови TypeScript, яка використовується на Angular, розробка стає зручніше та простіше для розуміння. При компіляції код перекладається JavaScript, тобто TS необхідний лише на етапі розробки.

До додаткових переваг Angular слід зарахувати:

- доступну документацію;
- потужні інструменти для розробки;
- підтримку спільноти;
- актуальність;
- стабільність.

Слабкою стороною фреймворку є високий поріг входження, оскільки потрібно бути знайомим з підмножиною мови JS – TypeScript.

Іншою проблемою стає регулярний вихід нових версій – у 2022 році було представлено Angular 13.Тобто необхідно систематично вдосконалювати знання для роботи з цим фреймворком. Також за великі можливості, передбачені Angular, доводиться платити навантаженням на продуктивність.

React - це бібліотека функцій з відкритим кодом, яка використовується для розробки інтерфейсів. Варто зазначити, що цей інструмент вважатиметься повноцінним фреймворком лише із підключенням сторонніх JavaScript бібліотек.

Сильні сторони React:

- підтримка Meta;
- висока швидкість роботи;
- велике community;
- кросплатформність;
- розробка UI на основі окремих компонентів;
- технологія Virtual DOM.

Однак необхідність у сторонніх бібліотеках для роботи робить процес розробки заплутанішим. Відсутні стандарти написання коду на HTML і CSS, що є у його конкурентів - Angular і Vue.JS. Доводиться вдаватися до додаткових плагінів, оскільки не всі стандартні браузери підтримують цей фреймворк.

Є доступ до управління низькорівневим функціоналом. Тому React підходить для досвідчених розробників.

Vue.js - це прогресивний JavaScript фреймворк з відкритим кодом, який може використовуватись і як бібліотека. Він простий у освоєнні, на відміну від вищезгаданих фреймворків, при цьому за продуктивністю не поступається React.

Примітно те, що за створенням даного інструменту стоїть лише один талановитий розробник - Еван Ю. Через що великими корпораціями цей інструмент сприймається з часткою скептицизму. Тим не менш, Vue.js користується попитом серед китайських компаній, зокрема всім відома Xiaomi.

Цей фреймворк переважно відповідає за подання, тим самим дозволяючи спростити взаємодія з іншими проектами та бібліотеками. Крім того, Vue.js більше підходить для невеликих проектів.

Сильні сторони:

- швидкість;
- невелика вага;
- лаконічність;
- відданість стандартам HTML, CSS;

- підтримка TS, JSX;
- низький поріг входу

Слабкими місцями є недостатньо велика спільнота і відсутність структури. Проте репутація Vue.js зростає.

Все частіше для створення сучасних веб-проектів розробники вдаються до використання JavaScript фреймворків. Перш ніж зробити вибір на користь одного з фреймворків, необхідно уточнити вимоги, проаналізувати переваги кожного інструменту.

РОЗДІЛ 2. ПОНЯТТЯ ПРО БІБЛІОТЕКУ REACT

2.1. Аналіз інноваційних принципів та нововведень бібліотеки React

React - популярна бібліотека, яка використовується для створення користувачів інтерфейсів. Вона була створена у Meta з метою вирішити низку проблем, пов'язаних з великомасштабними сайтами, керованими даними. У момент випуску бібліотеки в 2013 році цей проект сприймався з деякою часткою скептицизму, оскільки угоди про React досить незвичайні.

У спробі не злякати нових користувачів основна команда розробників React написала статтю під назвою Why React? (Чому саме React?), у якій рекомендувалося Give It (React) Five Minutes (Приділити їй (React) лише п'ять хвилин). Їм хотілося спочатку надихнути людей на роботу з бібліотекою, перш ніж ті подумають, що підхід розробників занадто шалений Так, React - це невелика бібліотека, яка не постачається з усім, що вам може знадобитися для створення вашої програми.

Так, завдяки React прямо у JavaScript створюється код, схожий на HTML. І, очевидно, всі ці теги вимагають попередньої обробки перед запуском у браузері. А для цього, швидше за все, знадобиться вбудований інструмент, такий як Webpack. [13]

А потім почали з'являтися питання: як можна буде перетворити цей JSX? Як завантажити дані? Куди подіти CSS? Що таке декларативне програмування? У кожному напрямі виникало ще більше запитань, як впровадити цю бібліотеку у повсякденну роботу. З кожною розмовою з'являлася нова термінологія, виникали нові технічні прийоми та дедалі більше запитань.

Бібліотека невелика за розміром і використовується лише для однієї частини роботи. Вона не містить інструментарію, що очікується від традиційного фреймворку JavaScript. Основні рішення про те, якими засобами

екосистеми користуватися, приймають розробники. Крім того, постійно з'являються нові набори інструментів, а старі відходять другою план. За такої кількості бібліотек зовсім не дивує відчуття, що за ними всіма неможливо наздогнати.

React вступила в епоху зрілості у дуже важливий, але хаотичний період історії JavaScript. Раніше Європейська асоціація виробників комп'ютерів (ECMA) досить рідко випускала нові специфікації. Іноді на випуск однієї з них витрачалося до десяти років. Це означало, що розробникам не потрібно було дуже часто освоювати новий синтаксис.

А з 2015 року нові функціональні властивості мови та синтаксичні доповнення випускатимуться щороку. Номерні випуски системи (ECMAScript3, ECMAScript 5) будуть замінені щорічними (ECMAScript 2016, ECMAScript 2017). У міру розвитку мови першопрохідники спільноти React прагнуть використати новий синтаксис. Найчастіше це свідчить, що документація передбачає знання синтаксичних новинок ECMAScript.

Розмови про стомлюваність JavaScript були все більш популярними, але така думка склалася тільки через складність складання програмного продукту. У минулому файли JavaScript просто додавалися до сторінки. Тепер вони повинні пройти процес складання, зазвичай у вигляді автоматизованого безперервного процесу постачання програмного продукту. Є формований синтаксис, який потрібно транспілювати для роботи у всіх браузерах. Є JSX, що підлягає перетворення на JavaScript. Є SCSS, можливо, що вимагає попередньої обробки. Ці компоненти потребують тестування, яке вони мають успішно пройти. React рішав ці проблеми, але тепер ще треба засвоїти Webpack, впоратися з розбиттям коду, його стисненням, тестуванням.

2.2. Механізми роботи бібліотеки React

Бібліотека React спромоглася серйозно знизити гостроту проблеми неконтрольованих мутацій завдяки використанню архітектури Flux. Замість приєднувати до довільної кількості довільних об'єктів (моделей) обробники подій, що викликають оновлення DOM, бібліотека React дала розробникам єдиний спосіб управління станом компонента. Це диспетчеризація дій, що впливають на сховище даних. Коли змінюється стан сховища, система пропонує компоненту перерендеруватися.

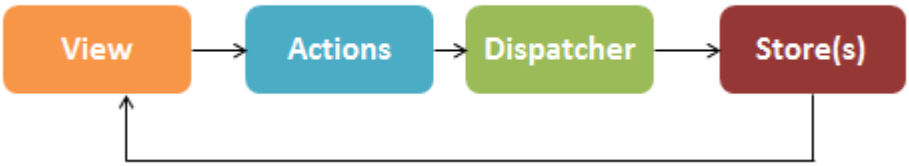


Рис. 2.1 Архітектура Flux

Читання даних з DOM для реалізації якоїсь логіки - це анти-паттерн. Той, хто так чинить, йде врозріз з метою використання React. Натомість дані потрібно читати зі сховища, а рішення, засновані на цих даних, потрібно приймати до того, як буде відрендеровано відповідні компоненти.

Якби детермінований рендеринг компонентів був би єдиною фішкою React, то це вже було б чудовою інновацією. Але команда розробників React не зупинилася. Ця команда представила світові бібліотеку, що має й інші цікаві, унікальні можливості. А з розвитком проекту в React з'явилося ще найбільш корисного.[14]

Одне з найважливіших нововведень був JSX. JSX - це розширення JavaScript, що дозволяє декларативно створювати компоненти інтерфейсу користувача. JSX має такі помітні можливості:

- Застосування простої декларативної розмітки.

- Код розмітки розташований там, де код компонента.
- Реалізація принципу поділу відповідальності (наприклад - відокремлення опису інтерфейсу від логіки стану та від побічних ефектів). Причому реалізація, заснована не на використанні різних технологій (наприклад, HTML, CSS, JavaScript).
- Абстрагування керування змінами DOM.
- Абстрагування від особливостей різних платформ, для яких створюють програми React. Завдяки використанню React можна створювати додатки, призначені для безлічі платформ (мова йде, наприклад, про розробку для мобільних пристроїв з використанням React Native, про програми для систем віртуальної реальності, про розробку для Netflix Gibbon, про створення Canvas/WebGL -інтерфейсів, про проект (react-html-email).

Якщо до появи JSX, потрібно було декларативно описувати інтерфейси, то не можна було обійтися без використання HTML-шаблонів. У ті часи не було загальновизнаного стандарту створення таких шаблонів. Кожен фреймворк використав власний синтаксис. Цей синтаксис доводилося вивчати тому, кому, наприклад, потрібно було пройтися в циклі за деякими даними, вбудувати в текстовий шаблон значення змінних або прийняти рішення про те, який компонент інтерфейсу виводити, а який - ні.

У наші дні, якщо подивитися на різні фронтенд-інструменти, виявиться, що без спеціального синтаксису, як директива *ngFor з Angular, теж не обійтися. Але, оскільки JSX можна назвати надмножиною JavaScript, створюючи JSX-розмітку можна користуватися існуючими можливостями JS.

Наприклад, перебрати набір елементів можна, скориставшись методом Array.prototype.map. Можна використовувати логічні оператори, організовувати умовний рендеринг за допомогою тернарного оператора. Можна користуватись чистими функціями, можна конструювати рядки з використанням шаблонних літералів. Загалом тому, хто описує інтерфейси засобами JSX, доступні всі можливості JavaScript. В цьому полягає величезна перевага React перед іншими фреймворками та бібліотеками.

При роботі з JSX необхідно враховувати деякі особливості, які, спочатку, можуть здатися незвичними.

- Тут використовується підхід до іменування атрибутів елементів, який відрізняється від того, який прийнятий у HTML. Наприклад, class перетворюється на className. Йдеться застосування стилю іменування camelCase.
- Кожен елемент списку, який потрібно вивести, повинен мати постійний унікальний ідентифікатор, призначений для використання в JSX-атрибуті key. Значення ідентифікатора має бути незмінним під час різних маніпуляцій з елементами списку. Насправді більшість елементів списків у моделях даних мають унікальні id, ці ідентифікатори зазвичай чудово показують себе ролі значень для key

React не нав'язує розробнику єдиний правильний спосіб роботи з CSS. Наприклад, компоненту можна передати JavaScript-об'єкт із стилями, записавши його у властивості style. За такого підходу більшість звичних імен стилів буде замінено їх еквіваленти, записані за правилами camelCase. Але цим можливості роботи зі стилями не обмежуються. На практиці розробники одночасно користуються різними підходами до стилізації React-додатків. Вибір конкретного підходу залежить від того, що потрібно стилізувати. Наприклад, глобальні стилі застосовують для оформлення тем додатків та макетів сторінок, а локальні стилі для налаштування зовнішнього вигляду конкретного компонента.

React дає в наше розпорядження кросбраузерну обгортку SyntheticEvents, що представляє синтетичні події та призначену для уніфікації роботи з подіями DOM. Синтетичні події дуже корисні з кількох причин:

- Вони дають змогу уніфікувати особливості різних платформ, пов'язані з обробкою подій. Це спрощує розробку кросбраузерних програм.
- Вони автоматично вирішують завдання управління пам'яттю. Якщо наприклад, збирається створити якийсь список з нескінченним прокручуванням, користуючись лише чистими JavaScript і HTML, то

доведеться делегувати події або підключати та відключати обробники подій у міру появи та приховування елементів списку. Все це потрібно буде робити, щоб уникнути витоків пам'яті. Синтетичні події автоматично делегуються кореневому вузлу, що призводить до того, що React-розробникам не доводиться вирішувати завдання управління пам'яттю.

- У роботі використовуються пули об'єктів. Механізми підтримки синтетичних подій здатні генерувати тисячі об'єктів за секунду та організовувати високопродуктивну роботу з такими об'єктами. Якщо вирішувати подібні завдання, щоразу створюючи нові об'єкти, це призведе до частої потреби виклику збирача сміття. А це, у свою чергу, може призвести до уповільнення програми, до видимих затримок у роботі інтерфейсу користувача і анімацій. Об'єкти синтетичних подій створюються заздалегідь і вміщуються в пул об'єктів. Коли потреби у події немає, вона повертається у пул. В результаті розробник може не турбуватися про те, що збирач сміття заблокує головний потік JavaScript, очищаючи пам'ять від об'єктів, що стали непотрібними.

Через використання пулу подій до властивостей синтетичної події не можна звернутися з асинхронної функції. Для реалізації такої схеми роботи потрібно взяти дані з об'єкта події та записати їх у змінну, доступну асинхронній функції.

Концепція життєвого циклу React-компонентів спрямовано захист стану компонента. Стан компонента не повинен змінюватися під час його виведення на екран. Це досягається завдяки наступній схемі роботи: компонент виявляється в якомусь стані та рендер. Потім, завдяки подіям життєвого циклу, виявляється можливим застосування щодо нього ефектів, можна впливати з його стан, працювати з подіями.

Розуміння особливостей життєвого циклу компонентів React дуже важливо для того, щоб розробляти інтерфейси і при цьому не боротися з React, а користуватися цією бібліотекою так, як задумано її розробниками. «Бій» з

React, на зразок неправильної зміни стану компонентів або читання даних з DOM, зводить нанівець сильні сторони цієї бібліотеки.

React, починаючи з версії 0.14, з'явився синтаксис описів компонентів, заснованих на класах, що дозволяє обробляти події життєвого циклу компонентів. У життєвому циклі React-компонентів можна виділити три найважливіші етапи: Mount (монтування), Update (оновлення) та Unmount (розмонтування). Етап Update можна розділити на три частини: Render (рендеринг), Precommit (підготовка до внесення змін до дерева DOM), Commit (внесення змін до дерева DOM).

- Render - на цьому етапі життєвого циклу компонента проводиться його рендерінг. Метод компонента render() повинен бути детермінованою функцією, що не має побічних ефектів. Цю функцію слід розглядати як чисту функцію, яка отримує дані з вхідних параметрів компонента та повертає JSX.
- Precommit - на цьому етапі можна прочитати дані з DOM, використовуючи метод життєвого циклу компонента getSnapShotBeforeUpdate. Це може бути дуже доречним, наприклад, якщо перед повторним рендерингом компонента потрібно дізнатися щось на кшталт позиції скролінгу або розмірів візуалізованого елемента.
- Commit - на цій фазі життєвого циклу компонента React оновлює DOM та рефи. Тут можна скористатися методом componentDidUpdate або хуком useEffect. Саме тут можна виконувати ефекти, планувати оновлення, використовувати DOM та вирішувати інші подібні завдання.

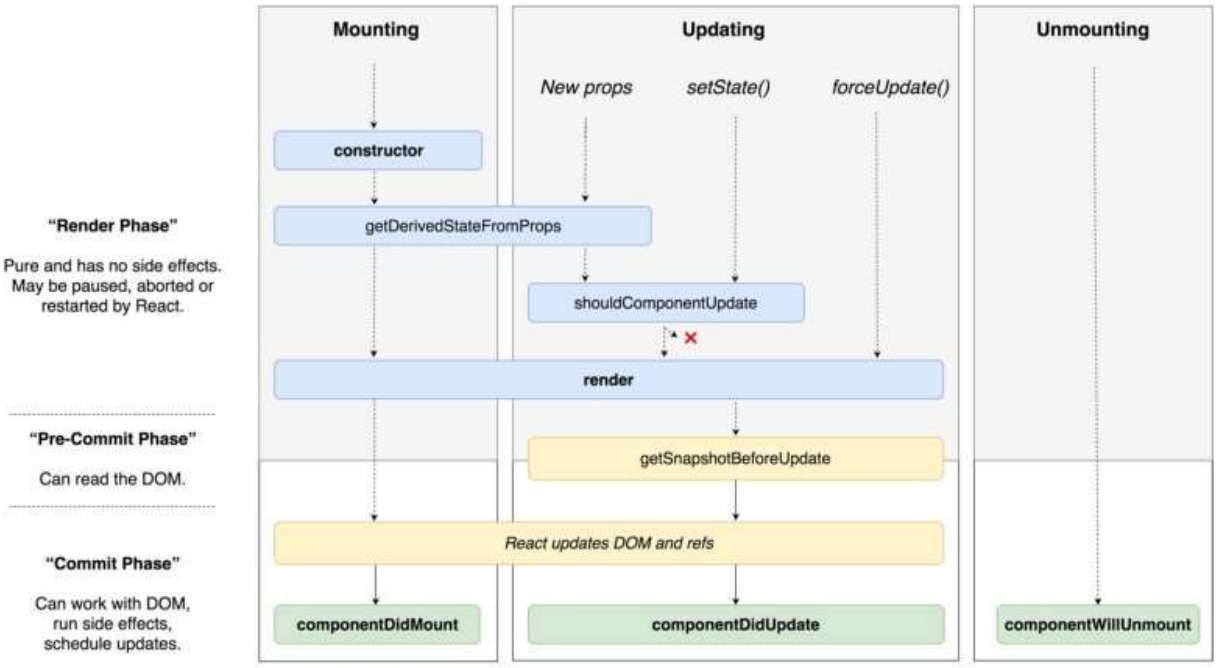


Рис. 2.2 Життєвий цикл React-компонентів

Представлення компонентів у вигляді довготривалих класів - це не найкраща ментальна модель React. Стан React-компонентів не повинен мутувати. Застарілий стан має замінюватись на новий. Кожна така заміна викликає повторний рендеринг компонента. Це дає React його, мабуть, найголовнішу та найціннішу можливість: підтримку детермінованого підходу до створення візуальних уявлень компонентів.

У React 16.8 з'явилася нова концепція - хуки React. Це функції, які дозволяють підключатися до подій життєвого циклу компонентів, не користуючись синтаксисом класів і не звертаючись до методів життєвого циклу компонентів. Компоненти, в результаті, стало можливим створювати над вигляді класів, а вигляді функцій.[15]

Виклик хука, в цілому, означає появу побічного ефекту - такого, що дозволяє компоненту працювати зі своїм станом і підсистемою введення-виведення. Побічний ефект - це будь-яка зміна стану, видима за межами функції, за винятком зміни значення, що повертається функцією.

Хук useEffect дозволяє ставити побічні ефекти в чергу для їх подальшого виконання. Вони будуть викликатися в потрібний час життєвого циклу компонента. Цей час може настати відразу після монтування компонента (наприклад, при виклику методу життєвого циклу componentDidMount), під час фази Commit (метод componentDidUpdate) безпосередньо перед розмонтуванням компонента (componentWillUnmount).

Багатьом компонентам потрібно виконувати якісь дії під час їх монтування, потрібно щось оновлювати при кожному перемальовуванні компонента, потрібно звільняти ресурси перед розмонтуванням компонента для запобігання витоку пам'яті. Завдяки використанню useEffect всі ці завдання можна вирішити в одній функції, не розділяючи їх вирішення на 3 різних методи, не змішуючи їх код з кодом інших завдань, не пов'язаних з ними, але також потребують цих методів.

2.3. Принципи створення застосунків за допомогою бібліотеки React

Розробка додатку на React не є кардинально новим процесом. Всі кроки є зрозумілими.

Крок 1 - Створення нового проекту за допомогою Create React App. На цьому кроці створюється нова програма, використовуючи диспетчер пакетів npm для запуску віддаленого скрипта. Скрипт скопіює необхідні файли в нову директорію та встановить усі залежності.

При установці Node також встановлюється програма для керування пакетами npm. npm встановить пакети JavaScript для проекту та відстежуватиме деталі проекту.

npm також включає інструмент npx, що відповідає за запуск пакетів, що виконуються. Це означає, що ви зможете запускати код програми Create React App без попереднього завантаження проекту.

Використовуваний пакет виконає інсталяцію create-react-app у вказану директорію. Для початку він створить у директорії новий пакет. Якщо цієї директорії не існує, пакет, що виконується, просто створить її. Скрипт також запустить команду npm install у директорії проекту для завантаження всіх додаткових залежностей.

Крок 2 - Використання скриптів react-scripts. Якщо перейти в директорію проекту можна замітити кілька файлів:

- node_modules/ містить усі зовнішні бібліотеки JavaScript, що використовуються програмою.
- Директорія public/ містить базові файли HTML, JSON та зображень. Це кореневі ресурси проекту.
- У директорії src/ міститься код React JavaScript для проекту. Це основна директорія для роботи
- Файл .gitignore містить кілька директорій та файлів за промовчанням, які система контролю вихідного коду git ігноруватиме, у тому числі директорію node_modules. Ігноровані елементи - це великі директорії або

файли журналу, які не потрібні під час контролю вихідного коду. Також тут наведені деякі директорії, які створюються за допомогою деяких скриптів React.

- README.md - це файл розмітки, що містить багато корисної інформації про програму Create React App, у тому числі огляд команд та посилання на розширені опції конфігурації. Зараз краще залишити файл README.md у первозданному вигляді. У міру розвитку проекту будуть замінюватись дані за умовчанням.

Останні два файли використовуються диспетчером пакетів. При запуску початкової команди npx створюється базовий проект та встановили додаткові залежності. Під час встановлення залежностей створився файл package-lock.json. Цей файл використовується npm для перевірки точності версій пакетів. Так можна бути впевнені, якщо хтось інший встановить проект, у нього будуть ідентичні залежності. Оскільки цей файл створюється автоматично.

Останній файл - package.json. Він містить метадані про проект, включаючи заголовок, номер версії та залежності. Також він містить скрипти, які можна використовувати для запуску проекту.

Відкривши файл, можна замітити об'єкт JSON, що містить усі метадані. Якщо подивитися на об'єкт scripts, він містить чотири різні скрипти: start, build, test та eject.

Перший скрипт запускає локальне середовище розробки. Другий скрипт виконує складання проекту.

Коротко про скрипт build. Для запуску будь-якого скрипта npm необхідно просто ввести в терміналі команду npm run script_name. Існує кілька спеціальних скриптів, де можна опустити run у складі команди, але вводити команду повністю ніколи не буде помилкою. Щоб запустити скрипт build.

Якщо відкрити файл .gitignore, можна помітити, що директорія build ігнорується git. Це пов'язано з тим, що директорія build просто є мініфікованою

та оптимізованою версією інших файлів. Контроль версій не потрібний, оскільки можна будь-коли запустити команду build. Скрипт test

Коротко про скрипт test - один із спеціальних скриптів, для яких не потрібне ключове слово run, однак він працюватиме і з цим ключовим словом. Цей сценарій запускає програму тестування Jest. Програма тестування шукає у проекті файли з розширенням .spec.js або .test.js, а потім запускає ці файли. Після запуску цього скрипта на терміналі будуть виведені результати тестування, а командний рядок терміналу буде прихований.

Слід звернути увагу на кілька речей. По-перше, як говорилося вище, тест автоматично виявляє всі файли з тестовими розширеннями, включаючи .test.js та .spec.js. Jest може виявляти тести в ієрархії коду, тобто можна розміщувати тести в директорії і Jest знайде їх там.

По-друге, Jest не просто запускає тест один раз і закривається. Він продовжує працювати у терміналі. Якщо внести зміни до вихідного коду, він знову проведе тести.

Також можливо обмежити склад тестів, використовуючи одну з опцій з клавіатури. Наприклад, якщо набрати «o», тести будуть виконуватися лише з файлами, які були внесені зміни. Це допоможе заощадити багато часу зі збільшенням обсягів тестових наборів.

Коротко про скрипт eject. Останній скрипт називається npm eject. Цей скрипт копіює залежності та файли конфігурації у проект, забезпечуючи повний контроль над кодом, але при цьому витягуючи проект із інтегрованої системи інструментів програми Create React App.

Цінність програми Create React App полягає в тому, що не потрібно турбуватися про значну кількість аспектів конфігурації. Для створення сучасних додатків JavaScript потрібно багато різних інструментів, від Webpack та інших складальних систем до Babel та інших засобів компіляції. Програма Create React App виконує усі завдання з налаштування конфігурації, тому вилучення проекту із програми означає, що доведеться вносити всі складні налаштування самостійно.

Мінус програми Create React App полягає в тому, що не можливо повністю персоналізувати проект. Для більшості проектів це неважливо, проте якщо потрібно повністю контролювати всі аспекти процесу складання, потрібно буде виконати вилучення коду. Однак після вилучення коду не можливо оновлювати його з новими версіями програми Create React App, і доведеться вручну додавати всі покращення.

Крок 3 - Запуск сервера. На цьому етапі ініціалізується локальний сервер і запускається проект у браузері.

Для запуску проекту використається ще один скрипт npm. Як і скрипт npm test, цей скрипт не потребує використання команди run. Цей скрипт запускає локальний сервер, виконує код проекту, запускає програму відстеження змін коду та відкриває проект у браузері.

Якщо запускається скрипт у локальній системі, він відкриє проект у вікні браузера і перемкне фокус із терміналу на браузер.

Якщо цього не станеться, можна відкрити URL-адресу http://localhost:3000/, щоб побачити сайт у роботі. Якщо порт 3000 вже запущено інший сервер, це не становить проблеми. Програма Create React App знайде наступний порт і запустить сервер на ньому. Іншими словами, якщо вже запущено проект порту 3000, новий проект буде запущено порту 3001.

Поки скрипт працює, у буде активний локальний сервер. Для зупинки скрипта слід закрити вікно або вкладку терміналу або ввести CTRL+C або ⌘+c у вікні або вкладці терміналу, де виконується скрипт.

РОЗДІЛ 3. СТВОРЕННЯ ЗАСТОСУНКУ ДЛЯ РОЗРАХУНКУ БУДІВЕЛЬНИХ МАТЕРІАЛІВ

3.1. Актуальність розробка веб застосунку «Розрахунок будівельних матеріалів»

В наш час, коли ціни на будівництво та ремонт досить високі, розумно згадати про те, що економія будівельних матеріалів допоможе зберегти свої гроші, не погіршуючи якість будівельно-ремонтних робіт. Економія у справах будівельної галузі завжди була актуальною темою, а зараз вона набуває все більшої популярності, у тому числі контроль та вибір оптимальних матеріалів на всіх стадіях будівництва.

Правильне управління великим будівництвом потребує досвіду та залучення експертів, які зможуть знайти резерви зниження витрат. Послуги таких компаній коштують дешевше, ніж ефект від їх участі, крім того, знижуються ризики від неправильного проектування, оскільки експертна компанія запропонує вести будівельний контроль великого об'єкта.

Ремонт - дуже відповідальний процес, тому перед його початком обов'язково потрібно грамотно прорахувати витрати всіх матеріалів, їх ціну та ціну на будівельні роботи. Якщо ремонт капітальний то інколи потрібно не просто поклейка шапалер, а навіть калькулятор для зведення стін

Перед відвідуванням магазину багато хто використовує улюблену народну методику - приблизно прикинути кількість будівельних матеріалів, а там уже як вийде. Цей спосіб вкрай неефективний і зазвичай призводить до нераціональної витрати бюджету, оскільки дуже часто виникає необхідність докуповувати матеріали додатково. Особливо болісно це відчувається, коли не вистачає трохи шпалер, ламінату, плитки та доводиться купувати цілий комплект, бо поштучно такої продукції не реалізують. Також потрібно бути

готовим до такої ситуації, коли під час відвідування магазину консультант скаже, що товар потрібної марки закінчився.

Зворотна ситуація полягає в тому, що будматеріалів закуповується набагато більше, ніж необхідно для ремонту і, у разі, неможливості повернення товару в магазин (що буває досить часто) знову витрачаються «гроші на вітер» і, до того ж, повинні підготувати значну площу. для зберігання матеріалів.

Якщо потрібно розрахувати вартість матеріалів для будівництва будинку своїми руками, то спочатку необхідно підготувати проектну документацію (самостійно або за допомогою сторонніх організацій), щоб розуміти, які будуть кінцеві параметри основних елементів конструкції. Обов'язково потрібно враховувати:

- фундамент;
- несучі та внутрішні стіни;
- перекриття;
- дах;
- сходи.

Власне, весь наступний розрахунок вартості будинку передбачає розрахунок матеріалів їх виготовлення. Потрібно пам'ятати, що в цьому випадку отримуємо тільки вартість будівництва будинку під так званий «чорний ключ», тобто коробка без обробки, внутрішніх перегородок, стяжок.

Грамотний та надійний розрахунок вартості будівництва будинку може бути отриманий лише за допомогою онлайн-калькуляторів чи організацій, які займаються цим професійно. Другий випадок, також має на увазі значні витрати і доцільний тільки для великих будівель, де явно зрозуміло, що розбіжності в кошторисі можуть легко перекрити ціну подібних послуг. Для приватного домобудування найбільш прийнятний другий варіант.

За допомогою сервісу будівельних онлайн калькуляторів користувач може отримати максимально ефективний розрахунок кількості матеріалів як для будівництва будинку, так і для ремонту, який враховує особливості

монтажу та дозволяє значно заощадити на бюджеті. Наприклад, при підрахунку шпалер для обробки кімнати, програма вважає, що якщо залишкова довжина рулону буде меншою за висоту стелі, то береться новий рулон, а залишки старого використовуються для дрібніших елементів, наприклад, під вікно.

Онлайн калькулятор розрахунку вартості матеріалів для будівництва будинку дозволяє не тільки автоматизувати типові операції, але й значно заощадити бюджет, використовуючи просунутий алгоритм обчислень.

3.2. Принципи роботи веб застосунку

Як і більшість веб сайти, застосунок для розрахунків будівельних матеріалів починає свою роботу з файлу «index.js». На відмінну від звичайних статичних сторінок на HTML, CSS та JS «index.js» є не великим файлом, займає 16 стрічок але грає ключову роль в роботі сайту. Головною функцією є «ReactDOM.render» React DOM порівнює елемент і його дочернє дерево з попередньої версії і вносить в DOM тільки мінімальні необхідні зміни. В параметри до «ReactDOM.render» передається «BrowserRouter» в якому міститься ключовий елемент - App компонента

App компонента є значно більшою ніж Index файл. Імпортує безліч файлів. Головна задача цієї компоненти дати браузеру зрозуміти, що потрібно перемалювати, перезагрузити відносно даних які знаходяться в посиланні на даний момент, через тег «Route». Також вона відмальовує і блоки які не залежать від посилання, в таких випадках тег «Route» не використовується.

```
const App = (props) => {
    return (
            <div className='app-wrapper'>
                <HeaderContainer />
                <Navbar/>
                <div className='app-wrapper-content'>
                    <Route path="/dialogs" render={() => <DialogsContainer />}/>
                    <Route path="/profile/:userId?" render={() => <ProfileContainer />}/>
                    <Route path="/users" render={() => <UsersContainer />}/>
                    <Route path="/news" render={() => <News />}/>
                    <Route path="/music" render={() => <Music />}/>
                    <Route path="/home" render={() => <Home />}/>
                    <Route path="/converters" render={() => <Converters />}/>
                    <Route path="/cubic" render={() => <CubicContainer />}/>
                    <Route path="/carpet" render={() => <CarpetContainer/>}/>
                    <Route path="/flooring" render={() => <FlooringContainer/>}/>
                    <Route path="/wallpaper" render={() => <WallpaperContainer/>}/>
                    <Route path="/brick" render={() => <BrickContainer/>}/>
                    <Route path="/roofing" render={() => <RoofingContainer/>}/>
                </div>
                <FooterContainer />
            </div>
    );
}
```

Рис. 3.1 Приклад App компоненти в проекті.

На рисунку. 3.1 видно, що App компонента повертає об’єкт, який визиває різні функції, до прикладу «HeaderContainer» які в свою чергу відображаються в вікні браузера. Також у блоці «app-wrapper-content» є кілька тегів «Route» які зчитують інформацію з посилання і перемальовують блоки відносно параметру «path». Важно пам’ятати що сам сайт не обновляється при цьому в браузері, адже це порушує саму ідею React, оновлюється тільки окремо взятий блок.

Якщо залізти в конкретну функцію, то можна побачити, що вона являє собою стрілочну функцію яка вертає об’єкт для відображення і експортує цю функцію. Сам файл з розширенням «.jsx», що хоч схожий по синтаксису на HTML код але має явну різницю. До прикладу якщо в HTML для створення класу потрібно писати «class» то у jsx файлах потрібно писати «className».

Майже всі функції, які відмальовують чось на сторінці, мають свої файли стилю. Використання їх теж не є схожим на HTML верстку. Для того щоб клас міг працювати як слід, потрібно імпортувати стилі і в них передавати клас через крапку.

Також для того щоб використовувати перенаправлення, замість тегу «a» використовується тег «NavLink». З допомогою «NavLink» браузер не перенапрявляє користувача за посиланням вказаним в «to» а просто змінює URL сторінки і дає знати що зміни були внесені. Далі дані зчитуються і вносяться відповідні зміни за участі App компоненти. «NavLink» слід використовувати адже він допомагає оптимізувати проект, зменшити витрати трафіку зі сторони користувача, адже не потрібно кожного разу оновляти сторінку а лише частину залежно віл посилання.

```jsx
import s from './Navbar.module.css';
import {NavLink} from "react-router-dom";

const Navbar = () => {
    return <nav className={s.nav}>

        <div className={s.item}>
            <NavLink to="/home" activeClassName={s.activeLink}>Home</NavLink>
        </div>
        <div className={s.item}>
            <NavLink to="/converters" activeClassName={s.activeLink}>Construction converters</NavLink>
        </div>
        <div className={s.item}>
            <NavLink to="/home-calculators" activeClassName={s.activeLink}>Home calculators</NavLink>
        </div>
        <div className={s.item}>
            <NavLink to="/other" activeClassName={s.activeLink}>Other calculators</NavLink>
        </div>
    </nav>
}

export default Navbar;
```

Рис. 3.2 Приклад стрілочної функції «Navbar» в проекті.

Основні блоки контенту завжди огорнені в контейнер, який забезпечує швидкий доступ до любої потрібної частини коду. Набагато простіше повторно використовувати компоненти, якщо розділити їх на дві категорії. Наприклад Контейнер та Презентаційні компоненти.

Презентаційні компоненти:

- Можуть містити як презентаційний, так і контейнери-компоненти всередині і зазвичай містять деяку власну розмітку DOM і стилі.
- Часто дозволяє звернутися через this.props.children.
- Не залежить від решти програми, таких як дії Flux або Сторони.
- Не використовується для завантаження та зміни даних.
- Отримує дані та функції зворотного дзвінка лише через props.
- Рідко зберігає свій стан (зазвичай вони відносяться до інтерфейсу, ніж до інших даних).

- Написані як функціональні компоненти до тих пір, поки не потребують стану, використання в життєвому циклі або оптимізації продуктивності.
- Наприклад, Page, Sidebar, Story, UserInfo, List.

Компоненти-контейнери:

- Можуть містити як презентаційний, так і контейнери-компоненти всередині, але зазвичай не мають розмітки DOM, за винятком деяких блоків div як обгортка, і не мають жодних стилів.
- Містять дані або поведінку презентаційних чи інших компонентів-контейнерів.
- Викликають дії Flux і передають їх як функції зворотного виклику в презентаційні компоненти.
- Часто мають стан і є джерелом даних.
- Найчастіше створюються з використанням компонентів високого порядку, таких як connect() у React Redux, createContainer() у Relay або Container.create() у Flux Utils, ніж пишуться вручну.
- Наприклад, UserPage, FollowersSidebar, StoryContainer, FollowedUserList.

Переваги такого підходу:

Найкращий поділ у вирішенні проблем. Програміст розуміє його програму краще при написанні компонентів слідуючи підходу.

Найкраще повторне використання. Людина може використовувати один і той же компонент з різними джерелами даних і перетворити їх на окремий компонент-контейнер, який можна використовувати повторно.

Презентаційні компоненти є «палітрою» програми. Спеціаліст може вставити їх на окрему сторінку і дати дизайнеру можливість настроїти всі їх варіації, минаючи логіку.

Це змушує витягти «компоненти макета» як Sidebar, Page, ContextMenu та використовувати this.props.children замість дублювання однакової розмітки та шарів у кількох компонентах-контейнерах.

```
import ...

let mapStateToProps = (state) => {
    return {
        calculatePage: state.calculatePage,
        calculatePageMoney: state.calculatePageMoney,
        calculatePageWallpaper: state.calculatePageWallpaper,
    }
}

let mapDispatchToProps = (dispatch) => {
    return {
        calculateData: (calculated) => {
            dispatch(CalculateCreator(calculated))
        },
        calculateDataMoney: (money) => {
            dispatch(CalculateCreatorMoney(money))
        },
        calculateDataWallpaper: (wallpaper) => {
            dispatch(CalculateCreatorWallpaper(wallpaper))
        }
    }
}

export default compose(
    connect(mapStateToProps, mapDispatchToProps)
)(Wallpaper);
```

Рис. 3.3 Контейнерна компонента «WallpaperContainer.jsx».

На Рис. 3.3. видно, що компонента містить дві функції «mapStateToProps» та «mapDispatchToProps». Ці функції в кінцевому результаті будуть експортовані і результат буде переданий в props далі по дереву проекту в презентаційну компоненту. Якщо «mapStateToProps» встановлює значення в state, то «mapDispatchToProps» приймає дані передані йому з презентаційної компоненти виконує dispatch, щоб оновити store

необхідно викликати метод dispatch(). Він викликається у об'єкта store, який створюється в store.js.

Далі данні попадають в відповідний редюсер. В файлі є стрілочна функція «calculateReducer» яка приймає state та action, якщо state передається від компонент вище то action це деякий набір інформації, що походить від додатка до сховища і який вказує, що потрібно зробити.

Якщо тип в action буде рівний «CALC» то буде створена нова змінна «body» яка потім попаде в state і попаде числом в масив на місце «calc». Так з кожним прорахунком буде добавлятись нова позиція в масиві. Потім редюсер експортує функцію на зовні.

```
const CALC = 'CALC';

let initialState = {...};

const calculateReducer = (state {cubic: any[]} = initialState, action) => {
    switch (action.type) {
        case CALC:
            let body = action.calculateData;
            return {
                ...state,
                cubic: [...state.cubic, {id: 6, calc: body}]
            };
        default:
            return state;
    }
}


export const CalculateCreator = (calculateData) => {
    return {
        type: CALC,
        calculateData
    }
}

export default calculateReducer;
```

Рис. 3.4 Редюсер «calculate-redicer.js»

Що до масивів, вони зберігаються в великому файлі під назвою «store.js», який є одним з основних в ієрархії файлів проекту. Файл містить масив «store» який в свою чергу містить підмасив «_state» з пустими даними, але при роботі додатку ці данні заповняються потрібною інформацією для прорахунку, тощо. Але store це не лише файл для зберігання даних це також один з ключових файлів який забезпечує передачу даних зі редюсерів в state.

```
let store = {
    _state: {...},
    _callSubscriber() {...},
    getState() {...},
    subscribe(observer) {...},
    dispatch(action) {

        this._state.profilePage = profileRedicer(this._state.profilePage, action);
        this._state.dialogsPage = dialogsRedicer(this._state.dialogsPage, action);
        this._state.calculatePage = calculateReducer(this._state.calculatePage, action);
        this._state.calculatePageMoney = calculateReducerMoney(this._state.calculatePageMoney, action);
        this._state.calculatePageWallpaper = calculateReducerWallpaper(this._state.calculatePageWallpaper, action);
        this._state.sidebar = sidebarRedicer(this._state.sidebar, action);

        this._callSubscriber(this._state);
    }

}
```

Рис. 3.5 Файл «store.js»

Коли всі данні дійшли до state їх потрібно відмалювати на сторінці. Данні через props попадають презентаційну компоненту, де дробляться на відповідні змінні і використовуються в відображенні на сторінці. З props, а в подальшому і з поділених змінних вибираються потрібні данні і передаються як змінні в заготовлений компонент «CalcItem» який вже на пряму відмальовує на сторінці данні.

```
const Wallpaper = (props) => {

    let state = props.calculatePage;
    let stateMoney = props.calculatePageMoney
    let stateWallpaper = props.calculatePageWallpaper

    let calculateElements = state.cubic.map(d => <CalcItem width={d.width} key={d.id} id={d.id}/>);
    let calculateMoney = stateMoney.calcMoney.map(m => <CalcItem money={m.money} key={m.id} id={m.id}/>);
    let calculateWallpaper = stateWallpaper.calcWallpaper.map(w => <CalcItem wallpaper={w.wallpaper} key={w.id}
                                                                             id={w.id}/>);
```

Рис. 3.6 Поділ даних з props

Також всі прорахунки відбуваються нижче по коду і саме там передаються в контейнерну компоненту.

Після того як данні прораховані та винесені в окремі змінні потрібно вивести на екран блок з наявною інформацією. Це робиться через звичайні js вставки в jsx коді.

```
<div className={s.solutionCont}>
    <div className={s.solution}>
        <div>
            <div className={s.solutionText}>Price</div>
            {calculateMoney}
        </div>
        <div>
            <div className={s.solutionText}>Room surface area</div>
            {calculateElements}
        </div>
        <div>
            <div className={s.solutionText}>Number of rolls</div>
            {calculateWallpaper}
        </div>
    </div>
</div>
```

Рис. 3.7 Принцип відображення блоку результату рахування

Після всіх маніпуляцій на сторінці повинно з’явитись число, яке є вірним в залежності від поставленої задачі.

3.3. Результати, методики та рекомендації до використання застосунку для розрахунку будівельних матеріалів

Після запуску веб застосунку «Розрахунок будівельних матеріалів» користувача одразу перекидає на головну сторінку з посиланням «/home».

На сторінці знаходиться чотири основних блоки контенту, хедер, меню, футер та сам основний контент по центру. В шапці сторінки знаходиться іконка, назва додатку та коротка інформація. В лівій частині екрану розташоване меню для легшої навігації по додатку. Меню структуроване та розбите ні підпункти з якими легко працювати. В центральній частині екрану розміщений основний блок який в залежності від посилання змінюється. І простенький футер знизу

При першому візиті користувачу запропонують найпопулярніші рішення, до прикладу: калькулятор підлоги, калькулятор шпалер, калькулятор цегли.

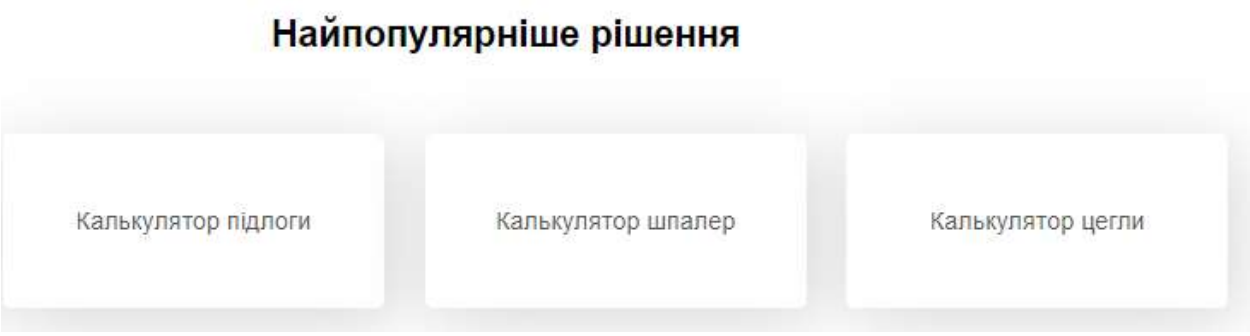


Рис. 3.8 Приклад найпопулярніших рішень

Всі блоки є інтерактивними а також візуально підсвічуються для полегшення орієнтації користувача, Також на них можна натиснути, що заставить додаток оновити центральний блок контенту в залежності від вибору користувача.

Якщо натиснути, наприклад, на калькулятор цегли, блок оновиться і відкриється калькулятор який рахує кількість та ціну цегли потрібної для побудови заданої стіни.

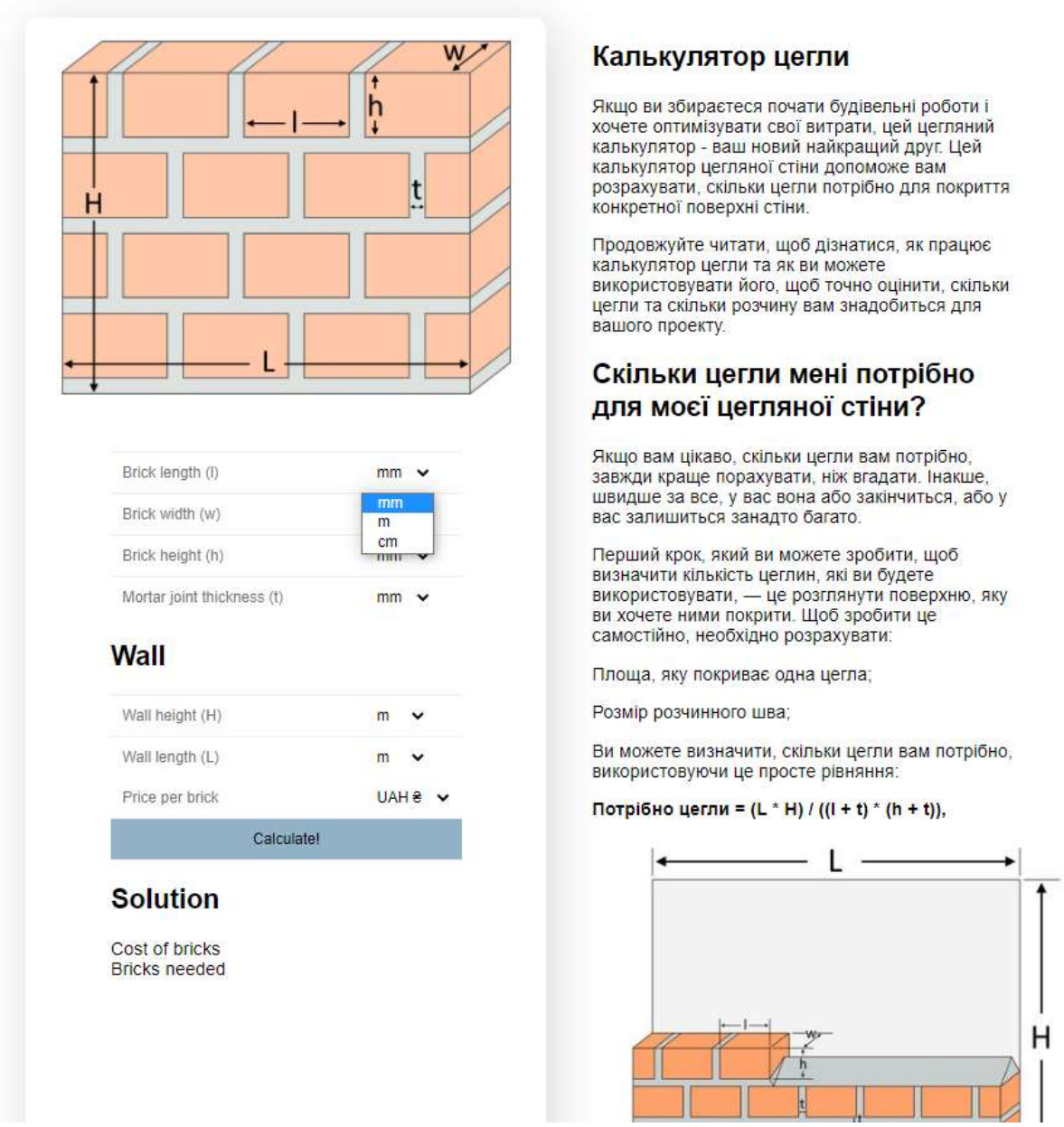
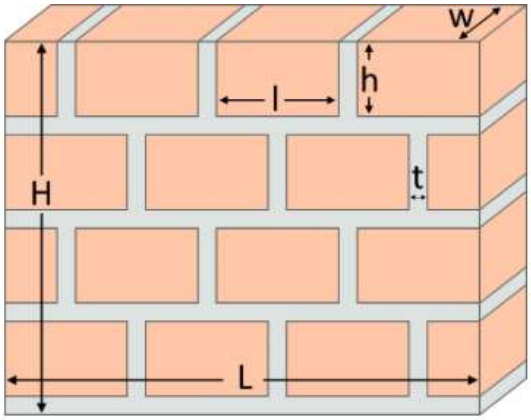


Рис. 3.9 Сторінка «Калькулятор цегли»

Блок контенту містить дві основних частини, це сам калькулятор та частина з потрібною теорією і принципом роботи.

В блоці калькулювання є верхня частина, з малюнком для того щоб користувач міг наглядно бачити як позначається та чи інша величина. Частина

в яку користувач вносить потрібні данні, такі як ширина довжина та висота цегли, довжину та висоту потрібної стіни та ціну за одну цеглу. Також можна вибрати міру виміру, калькулятор автоматично переведе все у метри, якщо це потрібно. Після введення даних користувач отримує відповідь. У даному випадку ціну за всю кількість цегли та саму кількість цегли.



200 mm
100 mm
100 mm
10 mm

Wall

1 m
1 m
15 UAH ₴
Calculate!

Solution

Cost of bricks
660
Bricks needed
44

Рис. 3.10 Результат роботи калькулятору

Також в правій частина знаходиться коротка інформація, в випадку з калькулятором цегли це опис про сам калькулятор, принцип роботи калькулятору та коротка відомість як ним користуватись.

Якщо користувачу більше не потрібен калькулятор цегли, можна перейти на головну сторінку, де він вже може обрати інші калькулятори які йому потрібні, наприклад калькулятор шпалер.

Калькулятор шпалер працює по схожій схемі як калькулятор цегли але кожен засіб відрізняється від іншого, так в калькуляторі шпалер немає блоку для введення довжини та ширини стіни чи цегли. Тут є наприклад поле для дверей, вікон та характеристик шпалер і кімнати, також опис інший.

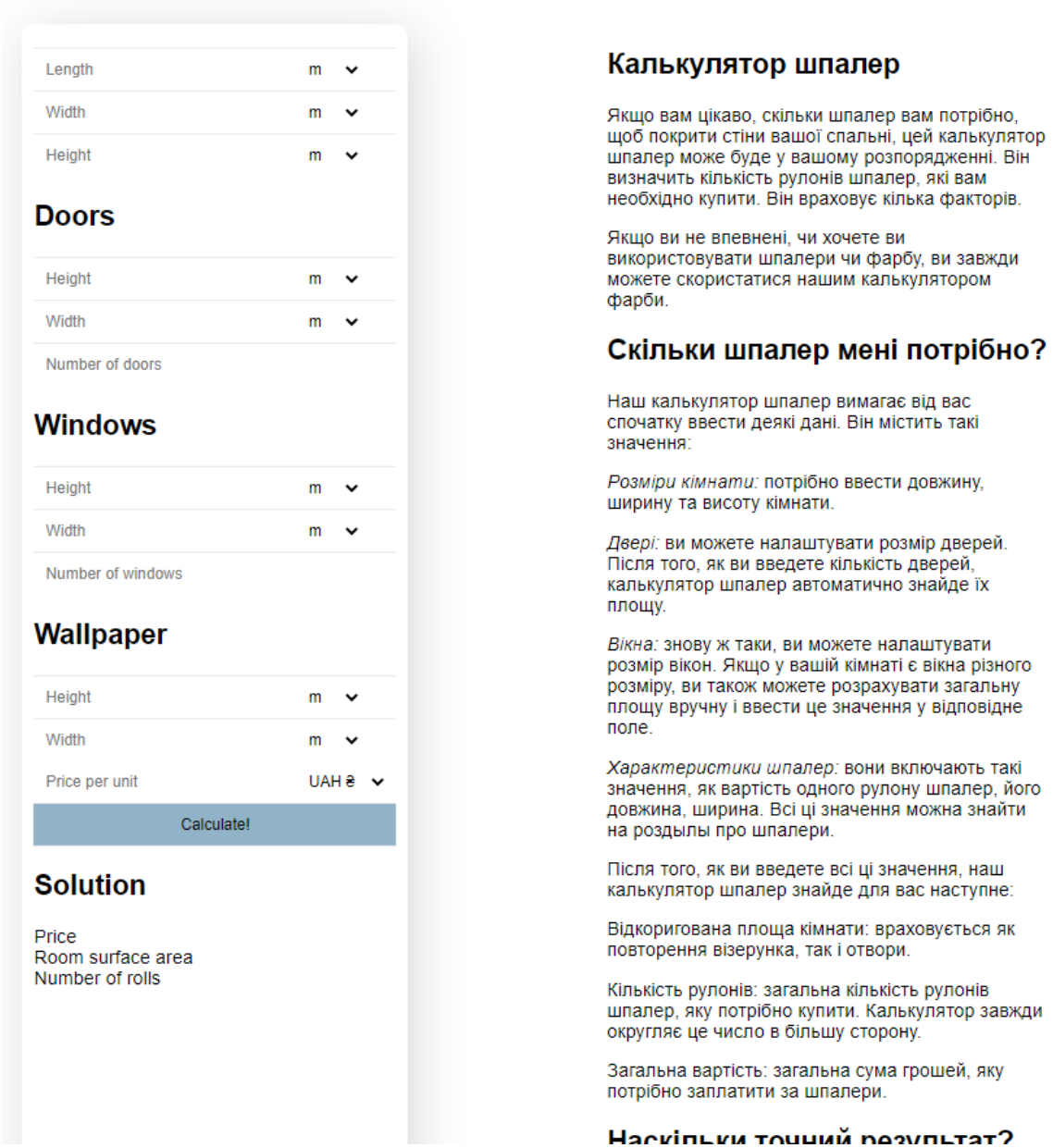


Рис. 3.11 Сторінка «Калькулятор шпалер»

Як і в минулому випадку блок контенту містить два підблоки, частина для розрахунків та інформаційна частина.

В частині для розрахунків знаходяться поля для прорахунку кімнати, також дверей, якщо вони є, якщо немає то в графі «Number of doors» потрібно вписати 0, і частина для внесення інформації про вікна, схожа в минулою. Також присутні форми для введення даних про шпалери, висоту ширини та

ціну за рулон. В кінцевому результаті додаток прорахує загальну ціну рулонів, площу стін для покриття шпалерами та кількість рулонів.

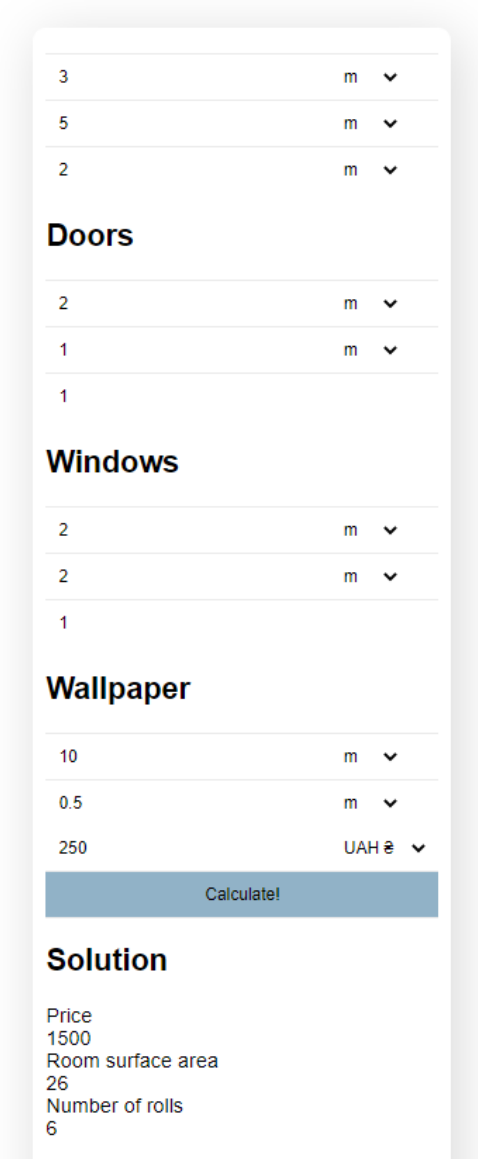


Рис. 3.12 Результат роботи калькулятору

Слід зазначити, загальна кількість рулонів шпалер, яку потрібно купити. Калькулятор завжди округляє це число в більшу сторону, адже при поклейці шпалер можуть ставатись непередбачувані трати матеріалу, тому в ідеалі завжди слід додати 10% додаткових шпалер, щоб врахувати складну

фурнітуру або для обрізків, про що і написано в правій частині з інформацією про калькулятор.

ВИСНОВКИ

Перед початком косметичного та капітального ремонту потрібно правильно розрахувати необхідну кількість матеріалів та скласти бюджет. Щоб чітко уявляти, якою є вартість реалізації проекту, слід використовувати калькулятор ремонту квартири. Він дозволить розрахувати ціну предметів окремо, а потім користувач зможете поєднати інформацію разом.

Для цього був створили безкоштовний калькулятор, що працює онлайн. Його великий плюс у тому, що його не треба завантажувати як програму та забруднювати вкотре свій ПК. Достатньо лише зайти в інтернет і скористатися будівельним калькулятором для розрахунку кількості будівельних матеріалів, які потрібно придбати та дізнатися про їхню підсумкову вартість.

Було розглянуто поняття веб застосунків, види, структура, історія та їхня роль в сьогоденні.

Встановленні і виявленні принципи роботи веб застосунків та нові бібліотеки для розробки веб додатків.

Було показано причину популярності бібліотеки React сьогодні.

Були розглянено механізми роботи та принципи створення React додатків.

Вилучено актуальність розробки веб застосунку «Розрахунок будівельних матеріалів»

Висвітлені і показані наглядно принципи роботи веб застосунку, рекомендації до використання користувачем.

Всі хто займається ремонтом потребують будівельного калькулятора, або калькулятора будівельних матеріалів, який з радістю взяв би на себе проблеми з підрахунком і допоміг користувачу у розрахунку матеріалів для ремонту та їх вартості

Користувачу для підрахунку лише потрібно вибрати необхідний калькулятор, ввести необхідні дані (довжина, ширина висота і т.д.) і

будівельний калькулятор сам розрахує скільки і чого на це потрібно, а також в скільки це обійдеться. Веб застосунку «Розрахунок будівельних матеріалів» дуже простий, зрозумілий та зручний у користуванні.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Umesha Naik omparative study of Web1.0, Web 2.0, Web 3.0 [Електронний ресурс] // Conference: 6th International CALIBER At university of Allahabad, Allahabad. – 2014. – Режим доступу до ресурсу: www.researchgate.net/publication/26445599_ Comparative_Study_of_Web_10_Web_20_and_Web_30C.

2. Rajiv and Manohar Lal Web 3.0 in education and research, 2011. – 335 с. – (BIJT – BVICAM's International Journal of Information Technology).

3. Chlsega A. Negrila Education in Web 3.0 / Ana-Marla Chlsega., 2013. – 58 с.

4. Ітінсон К. С. Вплив Інтернету речей на сучасне суспільство /. - 2019. - № 3 (16). - С. 58-60.

5. Angular vs. React vs. Vue: Порівняння, 2017. Новіков Сергій, 2018. Режим доступу: https://habr.com/post/338068

6. Повний посібник з ReactJS. Тюшкевич Сергій. Режим доступу: https://learn-reactjs.ru/

7. Чому варто використовувати React JS для розробки програм. 2018. Режим доступу: https://xbsoftware.ru/blog/pochemu-stoit-ispolzovat-react-js-razrabotke-prilozhenij/

8. React and React Native / Adam Boduch / 2017 – 31 ст. JavaScript Programming: Pushing the Limits / Jon Raasch / 2013 – 6 ст.

9. IOS 11 Swift Programming Cookbook / Vandad Nahavandipoor / 2017 – 121 ст.

10.У чому сила Redux? – Режим доступу: https://habr.com/post/333848/

11.Методологія БЕМ – Режим доступу: https://ua.bem.info/methodology/

12.React. Введення в JSX - Режим доступу: http://websketches.ru/react-docs/introducing-jsx

13.Banks A. Learning React: Functional Web Development with React and Redux / Alex Banks., 2017. – 350 с. – (1st edition).

14.Thomas M. React in Action / Mark Thomas., 2018. – 360 с.

15.Bugl D. Learn React Hooks: Build and refactor modern React.js applications using Hooks / Daniel Bugl., 2019. – 774 с. – (1st edition).

16.Murray N. Fullstack React: The Complete Guide to ReactJS and Friends / Nate Murray., 2017. – 836 с.

17.Santana Roldan C. React 17 Design Patterns and Best Practices: Design, build, and deploy production-ready web applications using industry-standard practices / Carlos Santana Roldan., 2021. – 394 с. – (3rd ed).

18.Angela Hill M. React Explained: Your Step-by-Step Guide to React / M. Angela Hill, R. Adair., 2019. – 366 с.