МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:
Завідувач кафедри
_____ д. е. н., проф. П. М. Грицюк
«_____» _____ 20___ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня «бакалавр»
за освітньо-професійною програмою «Інформаційні системи та технології»
спеціальності 122 «Інформаційні системи та технології»
на тему: «Розробка та впровадження інформаційно-аналітичних систем в сферу готельного бізнесу»

Виконав:
здобувач вищої освіти 4 курсу, групи ІСТ-41
Матвійчук Роман Сергійович
Керівник:
к. пед. наук, ст. викл. Парфенюк О.В.
Рецензент: к.тех. наук, доцент
Барановський С.В.

Рівне – 2021

АНОТАЦІЯ

Матвійчук Р. С. Розробка та впровадження інформаційно-аналітичних систем в сферу готельного бізнесу. Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»: 67 с., 14 рис., 2 табл., 1 додаток на 12 стор., 19 літературних джерел.

Об’єкт дослідження – розробка та оптимізація сайту для бронювання квитків у готелі.

Предмет досліджень – методи оптимізації сайту.

Методи дослідження – аналіз, порівняння, систематизація та узагальнення наукової літератури вітчизняних і зарубіжних авторів, електронних ресурсів для з’ясування стану розробленості проблеми технології створення вебресурсу з використанням фреймворків; методи системного аналізу для визначення видо-типологічної класифікації фреймворків; порівняльний аналіз для вибору програмного забезпечення; синтез для розробки інформаційно-концептуальної моделі веб-ресурсу; створення вебресурсу з використанням оптимального програмного забезпечення; тестування для аналізу зручності користування веб-ресурсом.

Дипломна робота присвячена розгляду фреймворка React. Результатом роботи функціонуючий сайт за допомогою фреймворку React, а саме, сайт бронювання номерів готелю. Були досліджені такі фреймворки, як Vue, Angular, і, відповідно, React, що дозволяють будувати додатки під різні операційні системи, використовуючи спільну логіку додатку. Було виділено основні недоліки та переваги фреймворка React порівнянні з іншими фреймворками та його підходів в розробці сайту. Обґрунтовано вибір даного фреймворка для розробки та впровадження інформаційно-аналітичних систем в сферу готельного бізнесу .

Ключові слова: Vue, Angular, React, Windows, Back-end, Front-end, сайт, бронювання, фреймворк, бібліотека.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ООП - Об'єктно Орієнтоване Програмування

БД - База Даних

ФП – Функціональне Програмування

HTML – HyperText Markup Language

ALS – Angular Language Service

CSS – Cascading Style Sheets

JS – JavaScript

JSX – JavaScript XML

DSL – Domain Specific Language

DOM – Document Object Model

PWA - Progressive Web App

API - Application Programming Interface

CLI - Command Line Interface

XML - eXtensible Markup Language

ВСТУП

Загально відомими термінами при розробці сайтів є HTML, CSS, JavaScript. Це інструменти, за допомогою яких відбувається створення сайтів. Проте розробники сайтів, зокрема працівники Facebook в першу чергу підбирають фреймворк для подальшої роботи з розробки, зокрема це – React, ASP.NET, Django, Laravel і так далі.

Люди та компанії створюють сайти для різних цілей: щоб продавати товари та послуги, розміщувати й знаходити інформацію, здобувати знання, спілкуватися з іншими користувачами, розважатися тощо. Кожен сам знаходить причину створити сайт.

Створення сайту може дуже сильно відрізнятися, взалежності від поставлених задач.

В будь-якому випадку, створення сайту складається з наступних етапів (для кожного конкретного сайту вони можуть бути насиченими або ж дуже короткими, але обов'язково присутні всі):

- Аналіз, проектування і розробка технічного завдання
- Дизайн сайту (розробка або вибір з готових варіантів)
- Верстання сторінок сайту
- Реалізація функціональної частини (це може бути збірка з готових модулів, або програмування, або ж і те, й інше, в тому числі з використанням конструктора сайтів)
- Тестування і, за необхідності, виправлення знайдених помилок
- Первинне наповнення сайту контентом
- Встановлення сайту (частіше всього закачка на хостинг, прописування шляхів до бази даних, тощо)
- Технічна підтримка сайту

Ми живемо в час, який супроводжується стресами та втомою. Людина не може проводити весь час в режимі постійної роботи і для того, щоб мінімізувати кількість нервових розладів та бути у тонусі, потрібно час від часу виділяти кілька тижнів на відпочинок. Без сумніву, краще всього

відпочивається десь на березі моря. Але для комфортного перебування у відпустці треба обрати хороший, красивий, комфортний готель. Забронювати його потрібно завчасно і саме для цього покликанні сайти. Такий сайт повинен бути відкритим, інформаційним, а також мати привабливий, але в той самий час простий дизайн. Створення такого сайту можливе за допомогою фреймворка React.

Актуальність роботи. Розробка та впровадження сучасних та зручних інформаційно-аналітичних систем в сферу готельного бізнесу, зокрема розробка сайту для бронювання квитків у готелях.

Мета роботи – залучити Front-end для створення сайту бронювання номерів готелю

Об'єкт дослідження – розробка та оптимізація сайту для бронювання квитків у готелі.

Предмет досліджень – методи оптимізації сайту.

Методи дослідження – аналіз, порівняння, систематизація та узагальнення наукової літератури вітчизняних і зарубіжних авторів, електронних ресурсів для з'ясування стану розробленості проблеми технології створення вебресурсу з використанням фреймворків; методи системного аналізу для визначення видо-типологічної класифікації фреймворків; порівняльний аналіз для вибору програмного забезпечення; синтез для розробки інформаційно-концептуальної моделі веб-ресурсу; створення вебресурсу з використанням оптимального програмного забезпечення; тестування для аналізу зручності користування веб-ресурсом.

Структура роботи: обсяг пояснювальної записки – 64 с., 29 рис.,
1 додаток на 12 стор., 19 літературних джерел.

.

РОЗДІЛ 1. АНАЛІЗ СТАНУ ВИВЧЕННЯ ПРОБЛЕМИ

1.1. Поняття бібліотеки та фреймворку

Під час створення перших програм, кожен програміст повинен був написати свій код “з чистого листа”, ось у програміста є мова програмування і вже на цій мові писали весь код який потрібен програмі. На сьогоднішній день це виглядає дуже важко, адже сьогодні більшість коду пишеться методом “копіювати – вставити”. Програмісти розуміли, що кожен раз вони пишуть один і той самий код, тому було би добре цей код кудись винести, адже він допоможе полегшити життя інших програмістів також. Ось так з’явились бібліотеки програмування, коли програміст збирав весь код який йому знадобиться, потрібні функції, наприклад, робота з рядками, робота з файлами, пам’ятю, тощо. Тому з часом, замість того щоб писати свої методи роботи, наприклад, з файлами, брав готову бібліотеку, підключав до свого коду і починав використовувати. Але знову ж таки, пройшов час, і та ж сама проблема настигла і бібліотеки. Програмісти кожен раз використовували фактично одні і ті ж самі бібліотеки, прийшло розуміння того, що всі програми які пишуться, складаються приблизно з однакових частин. Вони мають приблизно одну і ту ж саму архітектуру, а відрізняються в якихось частинах, наприклад в бізнес-логіці, зображеннях, запитах до бази даних, а інше все обв’язування одне і те ж саме. Тому якщо раніше писався код який викликав бібліотеки, то на сьогоднішній день створені фреймворки - основний код, якийфактично сам викликає бібліотеки, а програміст лише дописує його. Тобто фреймворк робить основне виконання коду, а код програміста підключається до нього(як маленькі функції, які фреймворк буде викликати). Виходить так, що програміст не пише основний код, основний код вже написаний в середині фреймоврку.

Але може виникнути питання, яка ж різниця між фреймворком та бібліотекою, адже фактично, вони двоє покликані для того, щоб полегшити роботу програмісту, фактично є кодом, який хтось написав за Вас. Різниця є. Можна програму, яку Ви пишете, порівняти з будинком, який облаштовуєте. Ви можете будувати будинок самі, а коли треба буде купляти меблі, поїдете в

магазин і купите те, що Ви вважаєте за потрібне. Ви тримаєте ситуацію на особистому контролі. Фреймворк в свою чергу можна порівняти з будівельною компанією. Вам пропонується декілька проєктів на вибір. Так, ви трішки обмежені в архітектурі і дизайні свого будинку, але професіонали багато чого беруть на себе. Вони повністю контролюють ситуацію і дадуть Вам знати, коли Ви зможете внести свій внесок. Тобто, використовуючи бібліотеки, програміст самостійно відповідає за потік програми. Тільки він вирішує, коли задіяти до роботи сторонню функціональність. Фреймворк, в свою чергу, сам відповідає за потік. Він виділяє декілька місць для розміщення коду програміста, але викликати його, чи ні, вирішує він сам.

Якщо дуже коротко і грубо, то фреймворк – це набір бібліотек, які автоматично викликаються. Якщо підсумувати все вище написане, то:

- фреймворки і бібліотеки - це код, написаний кимось іншим, який вирішує деякі спільні завдання, не обтяжуючи вас реалізацією цього рішення.

- фреймворк інвертує управління програмою і каже програмісту, що йому потрібно.

- бібліотека не втручається в потік програми. Її методи можна викликати тільки тоді, коли вони потрібні.[8]

1.2. Основні етапи створення веб-сайту

На сьогоднішній день дуже важко знайти людину, яка б не знала, що таке веб-сайт. Перший сайт був створений майже 30 років тому, 6 серпня 1991 року. Звичайно, пройшло дуже багато часу і наразі неможливо уявити сучасний інтернет без сайтів, а кількість інформації та курсів про створення цих ресурсів просто зашкалює. Створення функціонального веб-сайту непросте. Потрібно завжди дотримуватися балансу в плані дизайну та інформаційного вмісту. Існує багато сайтів з хорошим дизайном, але поганим інформаційним вмістом, та навпаки – поганим дизайном, але з прекрасною кількістю корисної інформації. Проте все ж таки, на мою думку, дизайн займає приблизно 60% успішності сайту, адже є велика вірогідність того, що при поганому, незрозумілому

дизайні, користувач просто вийде з сайту, навіть якщо в ньому є та інформація, яка була потрібна даному користувачу. Веб-розробка – це не лише впровадження кодів на веб-сайті. Саме тому, дизайн повинен бути гарним, стильним, але в той самий час він не повинен бути складним, щоб не заплутати користувача. Проте не потрібно забувати і про інформаційну складову. Люди, які відвідують Ваш веб-сайт, повинні отримати те, за чим вони прийшли, знайти ту інформацію, яку шукають. З розвитком верстки сайтів, з'явилась така думка, що метод створення сайту шляхом написання html – коду вже застарілий. Пояснюється це тим, що в ремеслі створення сайтів з'явились системи управління контентом (CMS), такий як WordPress, або конструктори, такий як Wix. WordPress – це найбільш відомаCMS з відкритим кодом. Майже 30 відсотків веб-сайтів у світі розміщено на цій платформі. Це приголомшливе число, враховуючи той факт, що кількість сайтів в інтернеті перевищує позначку у півтора мільярда. Вікс – найпопулярніший, і, можливо, той конструктор, який найбільш динамічно розвивається в світі. Якщо проаналізувати відгуки про цей ресурс, то він є доволі перспективним та надійним. Користувачі зазначають, що доволі часто саме тут з'являється більшість цікавих нововведень та фішок, а з часом їх переймають інші сервіси, інтерпретуючи по-своєму. Без сумніву, подібні сервіси спрощують життя, але при цьому Ви не можете контролювати створення власного сайту повністю. Веб-розробник використовує мови програмування, такі як HTML, CSS та Javascript, щобвеб-дизайн працював і функціонував так, як він хоче. Виділено 8 основних кроків для створення хорошого сайту.

1. Аналіз та збір інформації

Цей крок зазвичай дуже часто ігнорується програмістами у процесі розробки. Щоб Ваш веб-дизайн в кінцевому результаті подобався і вам, і користувачам Вашого сайту, необхідно детально розібрати тему, на яку Ви будете його робити. Ви можете надавати послуги чи продавати товари, покращити брендинг чи попрацювати над видимістю в Інтернеті. Також потрібно взяти до уваги, на яку цільову аудиторію буде розрахований Ваш

ресурс. Це можуть бути діти, підлітки, чи дорослі люди Всі ця інформація необхідна для успішної веб-розробки.

2. Планування

Після того, як ви закінчили збір інформації на тему, яка буде основою Вашого сайту, потрібно створити малюнок Вашого сайту, її ще називають мапою сайту. Її можна створити, намалювавши на листку паперу, або застосувавши спеціальні інтернет-сервіси. Карта сайту складається з інформації, зібраною на першому етапі. Основним мотивом мапи сайту є створення зручного для користувача веб-сайту та створення структури сайту. Після цього потрібно створити каркас. Каркас сайту припускає візуалізацію сторінки, без використання графіки або тексту. Каркас покаже усю структуру сайту. Крім цього, вирішіть, які функції ви хочете застосувати на сайті. Цей пункт планування включає в себе вхід на сайт, підписку на електронну пошту, зв'язок з адміністратором, чат у режимі реального часу та багато іншого.

3. Дизайн

Ключова частина успіху будь-якого сайту – це веб-дизайн. Веб-дизайн створюється відповідно до цільової аудиторії. Ваш додаток, який ви розробляєте для університету чи школи повинен радикально відрізнятись від того, що ви розробляєте для перегляду фільмів чи продажу товарів. Інші, не менш важливі речі, про які слід пам'ятати - це тема, кольорова гамма, де розмістити текст, зображення, відео тощо. Дизайн-макет систематично структурує вашу сторінку, щоб її вигляд був сучасним та привабливим.

4. Розробка

Цей етап включає в себе розробку коду для запуску на ньому сайту. Веб-розробник використовуватиме його, щоб він працював злагоджено та безперебійно. Це найважливіший крок у побудові, оскільки графічний дизайн на попередньому етапі оживає. В першу чергу, відповідно до мапи сайту, розробляється головна(перша) сторінка.

Тут ви нарешті можете розпочати створення веб-сайту. Всі попередні етапи розробки використовуються для створення, власне, веб-дизайну. Як вже

було сказано, спочатку створюється головна сторінка, а пізніше додаються всі інші. Всі вони створюються відповідно до ієрархії сайтів, створеної, як зазначено у мапі сайту. Знання та розуміння кожної технології розробки на цьому етапі має вирішальне значення.

5. Написання вмісту

Цей етап іде одразу після розробки. Інформаційний та привабливий вміст необхідний для того, щоб привернути увагу користувачів. Вміст для веб-сайту слід модифікувати для того, щоб люди могли знаходити те, що вони шукають. Це робиться за допомогою заголовків, підзаголовків, тегів, тощо.

Колонки для написання контенту - це те місце, де вам потрібно писати та публікувати вміст. Це може бути що завгодно, інформація про бренд, інформування людей про новий товар або послугу, створення потенційних клієнтів або просто Ваші думки. Вміст також залежить від тематики сайту.

6. Тестування

Це ще одна рутинна частина процесу веб-розробки. Перед запуском сайту потрібно перевірити всі сторінки та посилання для того, щоб переконатися в цілісності сайту, а також засвідчити той факт, що нічого не порушено. Потрібно перевірити кожну форму, сценарій, а також запустити програмне забезпечення для перевірки правопису, щоб знайти можливі помилки. Для того, щоб переконатись в тому, що Ваш код відповідає чинним стандартам веб-розробки, потрібно перевірити його, використавши валідатор коду. На цьому етапі ваш сайт тестується на деякі характеристики, зокрема:

- Швидкість веб-сайту
- Сумісність між браузерами
- Кілька тестів на екрані

Усі необхідні тести виконуються на веб-сайті перед тим, як запустити його у Всесвітню павутину.

7. Технічне обслуговування

По завершенню всіх попередніх етапів та запуску веб-сайту в інтернет, варто не забувати про той факт, що Ви також повинні його обслуговувати. Щоб

уникнути подальших незручностей, більшість розробників надають послуги з технічного обслуговування. Розробники повинні виконувати такі функції, як надання клієнтам вихідного коду та проектних документів, робота над зворотними зв’язками та підтримка після розробки. Цей крок дуже важливий, оскільки основна мета веб-сайту заключається в тому, що він повинен працювати для людей без перебоїв. Для такої роботи йому потрібна постійна технічна підтримка.

8. Вибір фронтенд фреймворку

Ну і останній крок в нашому списку по створенню сайту, проте не останній по значущості це вибір фронтенд фреймворку. Стрімкий розвиток JavaScript привів до бурхливого розвитку фронтенду. І за декілька років з'явилися безперечні лідери в особі: React, Vue.js і Angular. Усе це привело до величезної кількості статей, в яких йде порівняння фреймворков і бібліотек по наступному ряду критеріїв :

- кількість зірок на GitHub
- кількість відкритих питань на форумах
- розмір компільованого додатку
- кількість згадок в пошуковій системі Google
- кількість вакансій на той чи інший фреймворк

Який саме фреймворк вибрати, вирішувати саме Вам, проте від цього вибору буде залежати зручність та швидкість написання коду для Вашого сайту. Не потрібно забувати що React на момент написання цієї роботи є самим популярним фреймворком, що підтверджує рисунок 1.1.

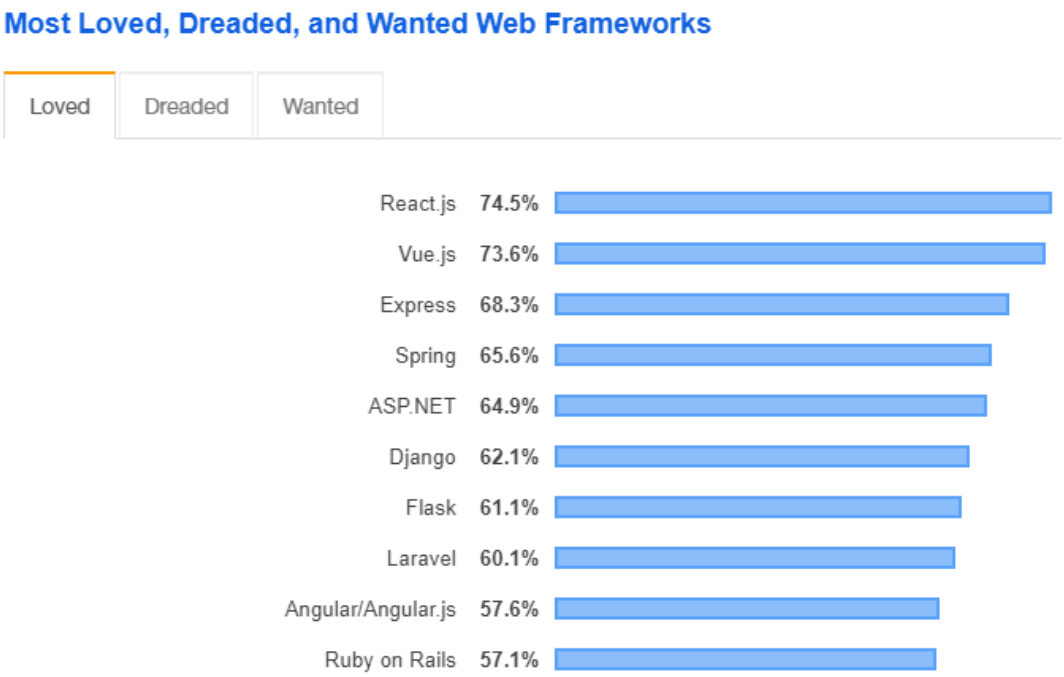

Рис. 1.1 – Аналіз популярності фреймворків

Так, відрив від фреймворку Vue складає менше одного відсотку, але факт залишається фактом, React – самий популярний фреймворк.

1.3. Основні відмінності Front-end і Back-end

Було б нелогічно розказати про фреймворки, але не розказати про їх призначення. Фреймворки поділяються на два види: Front-end фреймворки та Back-end фреймворки. Для розробки нашого сайту ми використовуємо лише Front-end, проте для повної картинки уявлення потрібно сказати і про Back-end. Front-end розробники працюють над тим, що може бачити користувач, тоді як Back-end розробники будують інфраструктуру, яка його підтримує, а також залучають до роботи БД. Структура додатку за участю Front-end та Back-end показана на рисунку 1.2.

Для високоефективного додатку або веб-сайту, обидві ці структури є необхідними компонентами. Нерідкі випадки, коли компанії схиляються до розбіжності між тим чи іншим застосунком, намагаючись орієнтуватися на розробку нового програмного забезпечення.

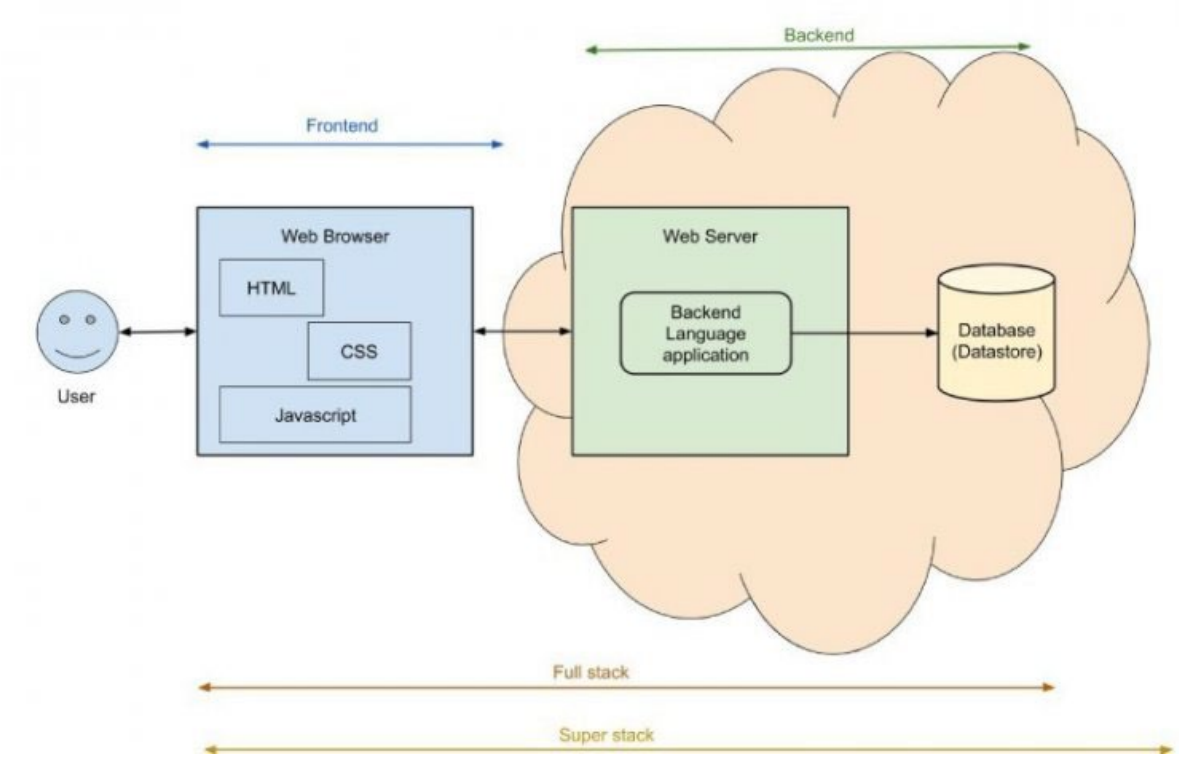

Рис. 1.2 – Загальна структура сайту

Зрештою, на ринку зростає кількість інструментів, спрямованих на те, щоб допомогти розробникам стати більш орієнтованими на «повний стек», тому нетехнічним працівникам легко припустити, що між спеціалістами з Front-end та Back-end немає великої різниці. Проте знову ж таки, опитування на Stack overflow показує, що потреба у Full-stack та у Back-end розробниках вища ніж у Front-end спеціалістів (рис1.3).

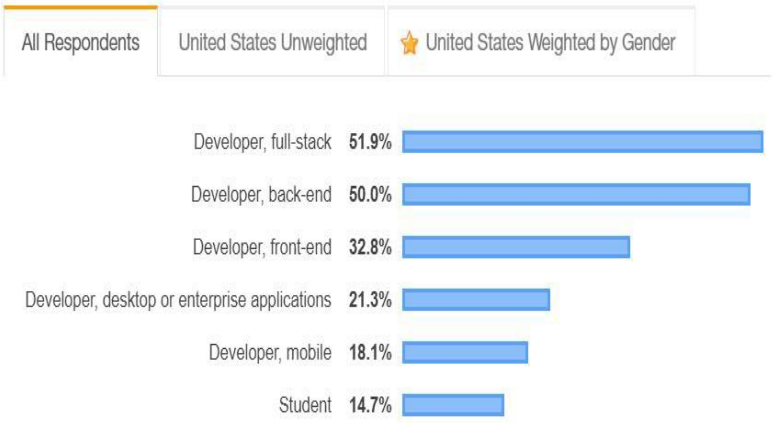

Рис. 1.3 – Аналіз потреби тих чи інших розробників

Але варто зазначити, що Front-end і Back-end розробники працюють в тандемі, це команда, яка працюєнад створенням систем і робить все для того,

щоб програма або веб-сайт працювали належним чином. Проте, якщо розібратись глибше, то цілком зрозуміло, що вони мають різні обов'язки. Термін "Front-end" стосується користувальницького інтерфейсу, картинка, яку бачить користувач, тоді як "Back-end" означає всі ті речі, які не доступні для звичайного користувача, вони працюють за кадром, а саме сервер, додаток та базу даних.

Back-end

Back-end, складається з сервера, який надає дані на запит користувача, програми, яка їх каналізує, та бази даних, яка отримує, зберігає та організовує інформацію. Для прикладу можна взяти ситуацію, коли клієнт переглядає товар на веб-сайті. В цей момент він взаємодіє саме з інтерфейсом. Після того, як він визначився з потрібним йому товаром, користувач натискає кнопку для покупки та заповнює всю потрібну інформацію для покупки. Ця інформація зберігається всередині бази даних, яка знаходиться на сервері. Якщо ж клієнт вирішив перевірити стан своєї покупки, сервер дістає відповідну інформацію, оновлює її даними відстеження та подає через інтерфейс. Основою Back-end розробників є створення додатків, які можуть знаходити та доставляти дані в інтерфейс. Багато з них використовують надійні бази даних на рівні підприємств, такі як Oracle, Teradata, Microsoft SQL Server, IBM DB2, EnterpriseDB та SAP Sybase ASE. Існує також ряд інших популярних баз даних, включаючи MySQL, NoSQL та PostgreSQL. Найкращі Back-end фреймворки 2020 року:

- Laravel
- Flask
- Express.js
- Django
- Ruby on Rails
- Meteor
- Spring
- Koa

- Nest.js
- Node.js
- Beego

Рис. 1.4 – Back-end фреймворки

Full-stack

Full-stack розробник, це людина, яка однаково добре володіє навичками Front-end та Back-end розробника. Серед спеціалістів існує така думка, що розробка повного стеку одним програмістом не є практичним рішенням. Однак все частіше компанії шукають на роботу людей, які спеціалізуються і на Front-end, так і на Back-end, відомі як Full-stack розробники. Вони є потужними командними гравцями, оскільки вони мають широкий рівень знань, щоб побачити загальну картину, дозволяючи їм пропонувати способи оптимізації процесу або усунення перешкод, які можуть уповільнювати роботу системи.

Front-end

Ми підібрались до основи дипломного проекту. Фронтальний інтерфейс побудований з використанням комбінації таких технологій, як мова розмітки гіпертексту (HTML), JavaScript та каскадні таблиці стилів (CSS). Саме ця

складова відповідає за картинку на сайті. Front-end розробники тісно співпрацюють з дизайнерами та аналітиками. Все, що Ви бачите на веб-сайті, стало можливим завдяки інтерфейсному розробнику. Всі фотографії, переходи між сторінками, логотип, текст, все що можна побачити. Але для такої роботи, розробники даного амплуа повинні бути знайомі з фреймворками, такими як Angular, Ember, Backbone та React,. Ці застосунки дозволяють розробникам не відставати від зростаючого попиту на корпоративне програмне забезпечення, не жертвуючи якістю. З цього випливає висновок, що вони цілком заслужено займають своє місце, як стандартні засоби розробки. Один із них є React, про який буде йти мова в наступному пункті

1.4. Загальні відомості про React

Фактично 90% вакансій Front-end web development містить у вимогах досвід роботи з React. Як вже було сказано, React – це фреймворк(JS - бібліотека) для створення інтерфейсу, який змінює відображення без перезагрузки сторінки. Завдяки цьому, додаток швидко реагує на дії користувача. React був створений компанією Facebook. Бібліотеку створено Джорданом Волком (Jordan Walke), програмістом з Facebook. Автор працював над проєктом під впливом XHP, фреймворку HTML для PHP. 2011-го року даний реліз з'явився у новинах Facebook, а вже через рік – у блозі Instagram. Також фреймворк був представлений як проєкт з відкритим початковим кодом на конференції розробників JSConf US, що проходила у Сполучених Штатах у травні 2013 року. На конференції React.js Conf (рис. 1.5), влаштовану Фейсбуком у березні 2015-го, проект було представлено як відкрите програмне забезпечення [6].

React дозволяє спрощувати створення великих динамічних додатків. Його мета заключається в тому, щоб швидко працювати, бути простим та добре масштабуватись.

React – це інструмент для обробки користувацького інтерфейсу. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з

іншими бібліотеками, такими як Redux.

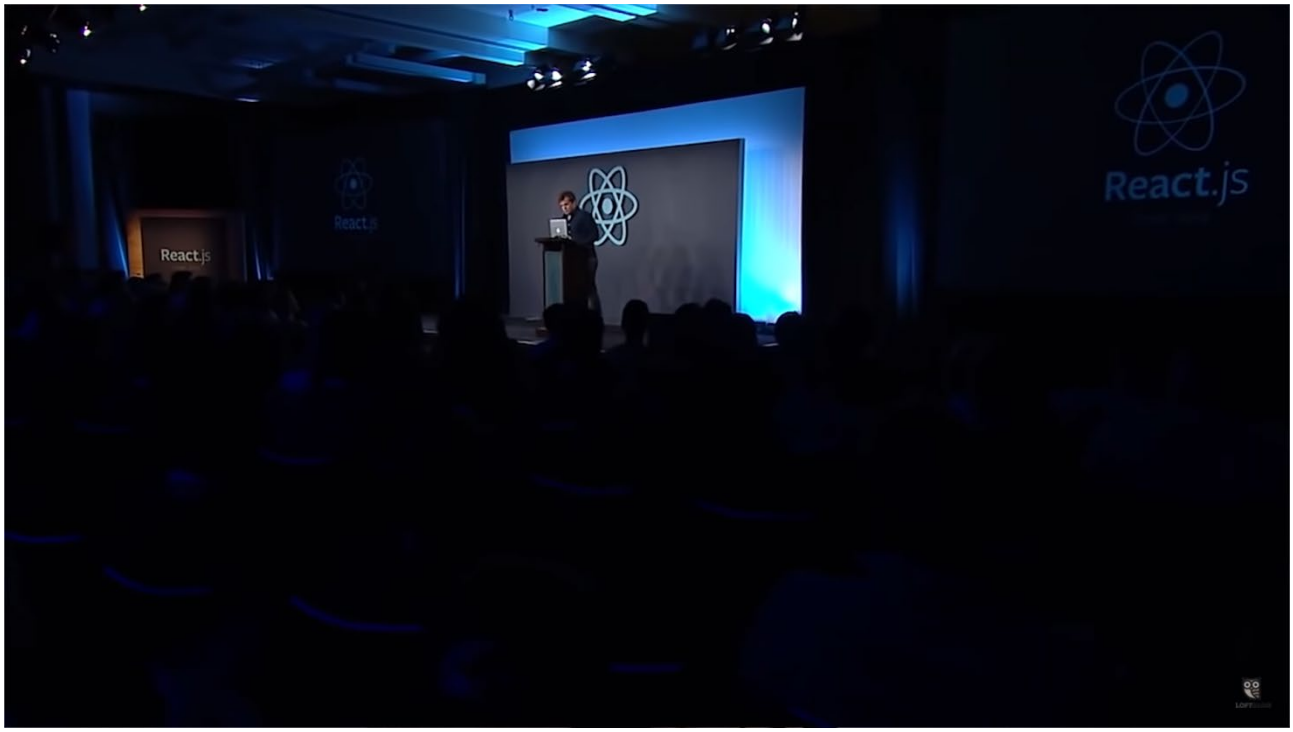

Рис. 1.5 – Офіційна презентація React

В даний час React використовують Khan Academy, Netflix, Yahoo, Airbnb, Sony, Atlassian та інші. Тобто можна зробити висновок, що React користується дуже великою популярністю не лише серед рядових програмістів, але й серед великих міжнародних компаній [18].

1.5. Особливості React. Поняття DOM

Як вже було сказано, React був створений не окремою людиною,а великою міжнародною компанією. Однією з основних причин створення React було бажання цієї компанії поліпшити механізми формування інтерфейсів. Цю бібліотеку розробили тоді, коли в Facebook зіткнулися з деякими проблемами, що стосуються розробки та підтримки проекту. Зміни в інтерфейсі додатків виконуються шляхом модифікації об'єктної моделі документа (DOM). Існують такі поняття, як віртуальна DOM і реальна DOM. Від того, наскільки саме організований процес роботи з DOM, серйозно залежить продуктивність JavaScript-інструментів для розробки інтерфейсів. Перед тим, як ми почнемо вникати в те, що з себе представляє DOM віртуальний, давайте трохи

поговоримо про те, чим є DOM реальний. DOM (абревіатура від Document Object Model) - спосіб представлення структурного документу за допомогою об'єктів [9]. Це кроссплатформенна і незалежна угода для представлення і взаємодії з даними в HTML, XML і так далі. Веб-браузери обробляють складові DOM, і ми можемо взаємодіяти з ними, використовуючи JavaScript і CSS. Ми можемо працювати з вузлами документу, змінювати їх дані, видаляти і вставляти нові вузли. Тобто, браузер, коли дає запит на сторінку і отримує у відповідь від серверу її html – код повинен спочатку його розібрати. Він продивлюється весь наш html – код і створює дерево елементів. Все починається з елементу document, потім в нього вкладається елемент html, а вже в свою чергу в нього вкладається елемент body. Далі, всі елементи вкладаються один в одного в правильній послідовності (рис.1.6).

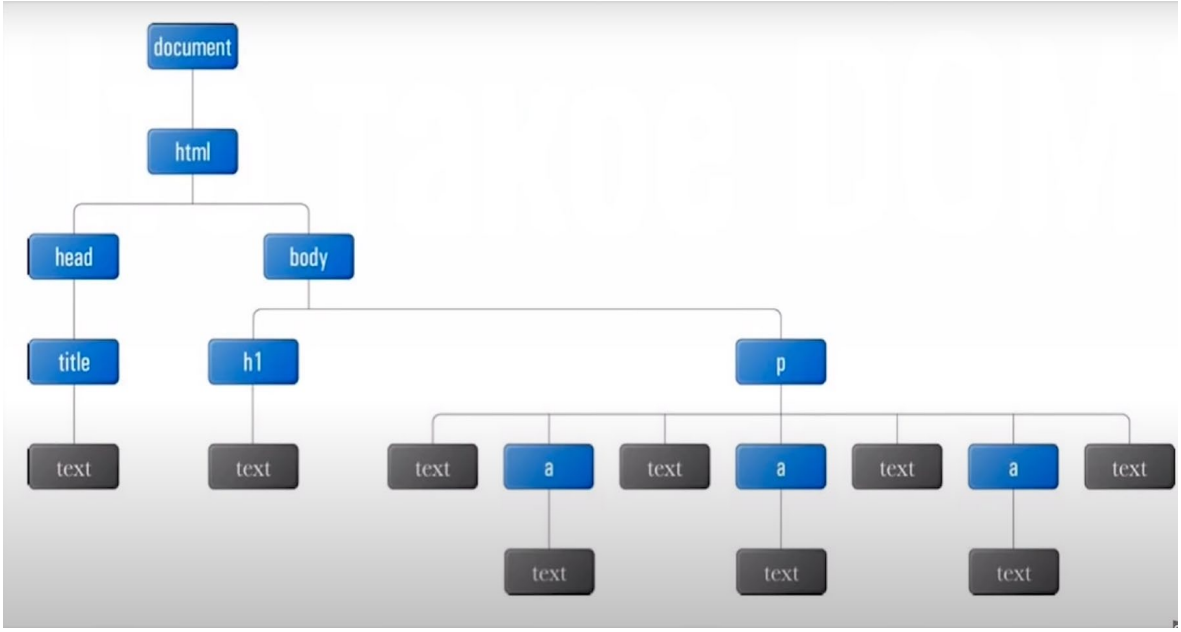

Рис. 1.6 – Послідовність структури сайту

Подивитись, що таке DOM можна за допомогою клавіші F12 у браузері (рис. 1.7).

В наші дні DOM API являється практично кроссплатформеним і кроссбраузерним. Так в чому ж проблема? Головна проблема DOM - він ніколи не був розрахований для створення динамічного призначеного для користувача інтерфейсу. Ми можемо працювати з ним, використовуючи JavaScript і бібліотеки на кшталт jQuery, але їх використання не вирішує проблем з

продуктивністю. Подивіться на сучасні соціальні мережі, такі як Twitter, Facebook або Pinterest. Після невеликого скролінгу, ми матимемо десятки тисяч DOM- вузлів, ефективно взаємодіяти з якими - завдання не з легенів.

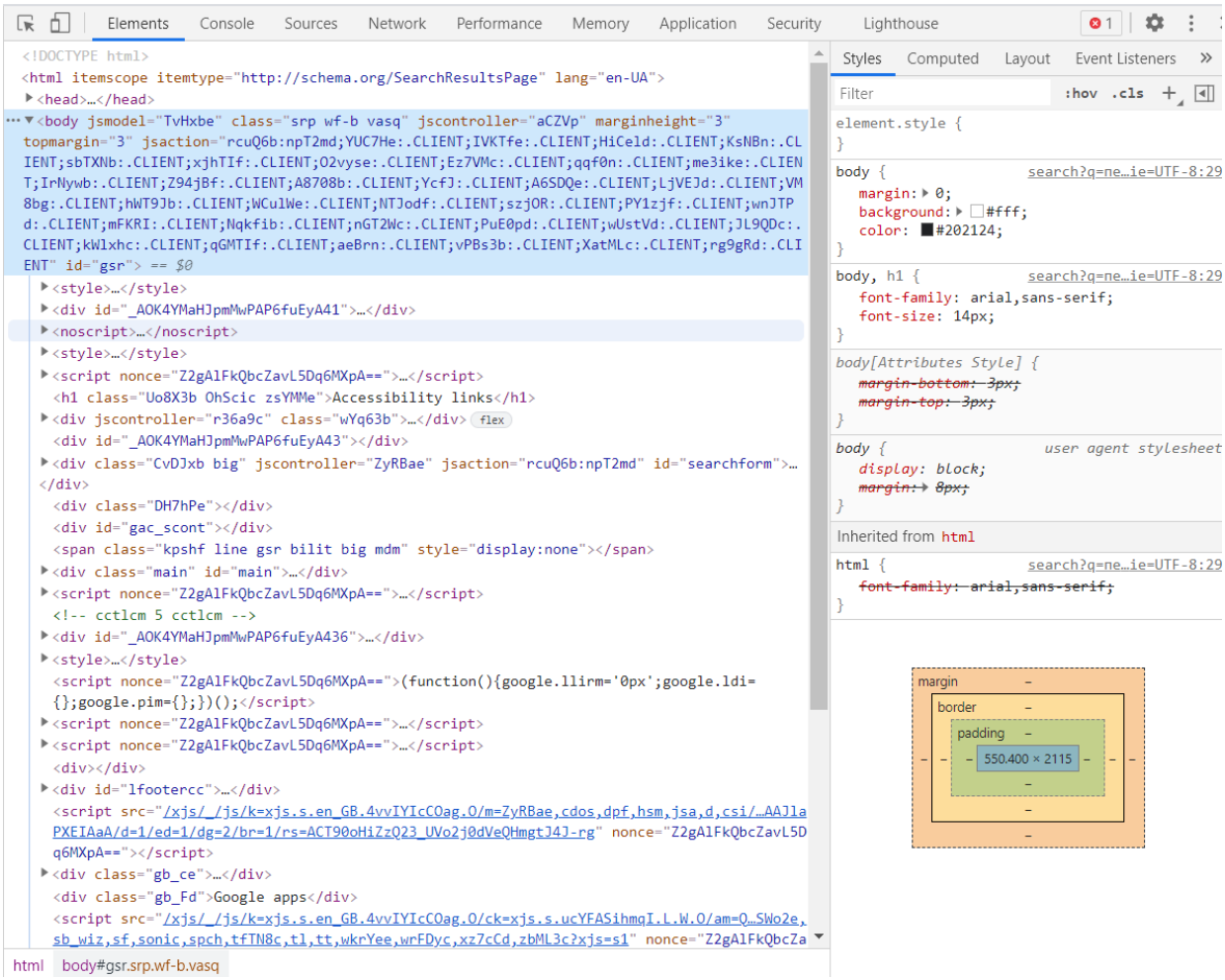

Рис. 1.7 –Структура DOM

Для прикладу, спробуйте перемістити 1000 div – блоків на 5 пікселів вліво. Це може зайняти більше секунди – це надто багато для сучасного інтернету. Ви можете оптимізувати скрипт і використати деякі прийоми, але у результаті це викличе лише головний біль при роботі з величезними сторінками і динамічним UI.

Нині W3C працює над новим стандартом Shadow DOM. Shadow DOM – це робоча чернетка стандарту W3C. Специфікація, що описує метод об'єднання декількох DOM – дерев в одну ієрархію і як ці дерева взаємодіють один з одним в межах документу, що дозволяє краще скомпонувати DOM. Інший варіант

полягає у використанні підходу з Virtual DOM. Virtual DOM не є стандартом і зрештою ми як і раніше взаємодіємо з DOM, але робимо це як можна рідше і більш ефективно. Замість того, щоб взаємодіяти з DOM безпосередньо, ми працюємо з його легкою копією. Ми можемо вносити зміни в копію, виходячи з наших потреб, а після цього застосовувати зміни до реальному DOM. При цьому відбувається порівняння DOM - дерева з його віртуальною копією, визначається різниця і запускається перемальовування того, що було змінено. Структура DOM вказана на рисунку 1.8.

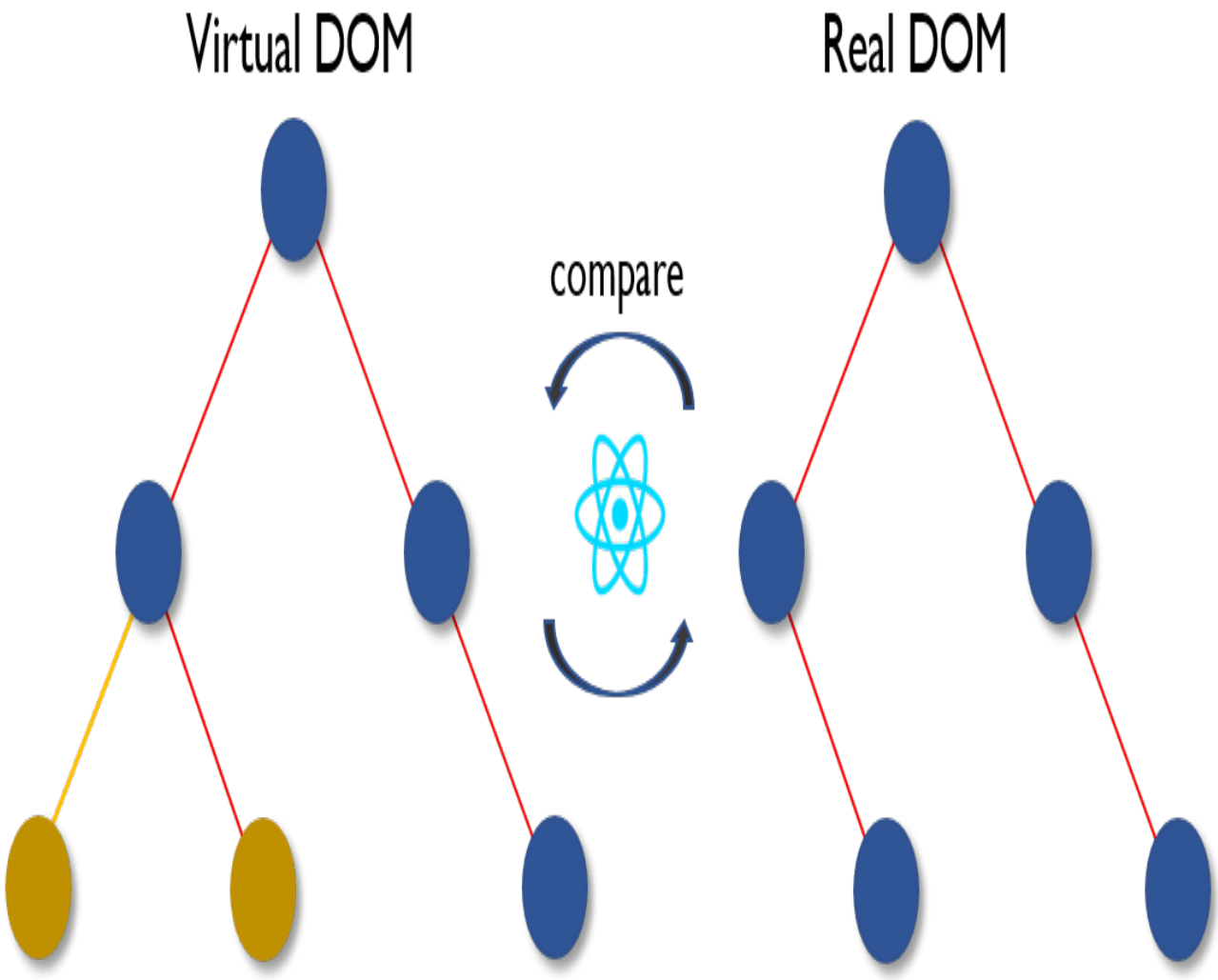

Рис. 1.8 – Структури Virtual DOM та Real DOM

Такий підхід працює швидше, тому як не включає усі ваговиті частини реального DOM. Але тільки якщо ми робимо це правильно. Є дві проблеми: коли саме робити повторне перемальовування DOM і як це зробити ефективно.

Якщо дані змінюються і потребує оновлення є два варіанти дізнатися, що

дані змінилися: Перший з них – "dirty checking"(брудна перевірка) полягає в тому, щоб опитувати дані через регулярні проміжки часу і рекурсивно перевіряти усі значення в структурі даних. Другий варіант – "observable"(спостережуваний) полягає в спостереженні за зміною стану. Якщо нічого не змінилося, ми нічого не робимо. Якщо змінилося, ми точно знаємо, що треба оновити.

Ефективні алгоритми порівняння робить цей підхід дійсно швидким. Угрупування операцій читання/запису при роботі з DOM, ефективне оновлення тільки під-дерев. Як ви розумієте, це не так просто і реалізація може виявитися досить складною, але є деякі бібліотеки, які допомагають реалізувати цей підхід в наших проектах. Однією з таких найвідоміших бібліотек власне і є React.

1.6. Класифікація React

З самого початку, React позиціонувався користувачами як фреймворк, але з часом на сайті самого React уточнили, що React – це все ж таки бібліотека (рис.1.9)[7].

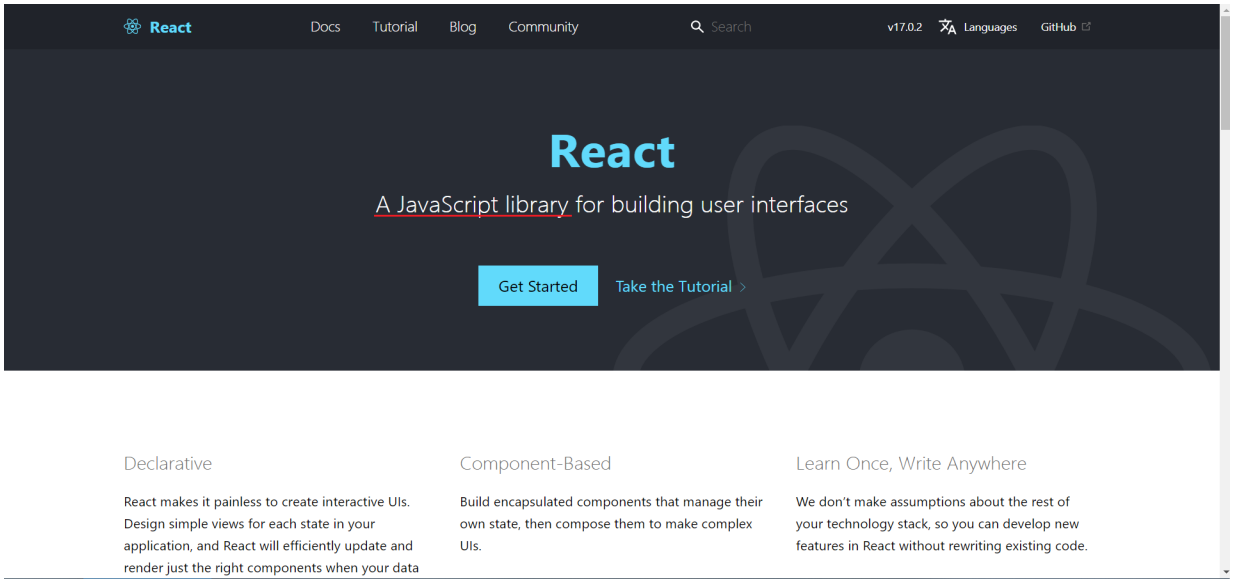

Рис. 1.9 – Головна сторінка офіційного сайту React

На цьому можна було б закрити дане питання і більше до нього не повертатись, проте на найбільшому інформаційному сайті – Wikipedia,

написано,що React це фреймворк (рис.1.10).

React

Тип	фреймворк
Розробники	Facebook, Instagram та співтовариство
Перший випуск	2013
Стабільний випуск	16.13.1 (19 березня 2020; 14 місяців тому[1])
Версії	17.0.2 (22 березня 2021)[2]
Репозиторій	github.com/facebook/react
Платформа	багатоплатформова
Операційна система	крос-платформова програма
Мова програмування	JavaScript
Розмір	128 Кб мінімізована версія 559 Кб версія для розробки
Стан розробки	Активний
Ліцензія	MIT License
Вебсайт	reactjs.com, uk.reactjs.org (українська версія)

Рис.1.10 – Фрагмент із Wikipedia

Було б правильно думати, що розробники React краще знають, що вони зробили, фреймворк чи бібліотеку, але ж чому тоді на такому великому ресурсі як Wikipedia вказано, що React все ж таки фреймворк. Як вже було сказано, різниця між бібліотекою і фреймворком в тому, що бібліотека зазвичай фокусується на вузькій області. Він надає набір допоміжних функцій, методів і т. д., які ви можете викликати у своєму проекті для досягнення певної функціональності. В основному це колекція визначень класів, написана в для сприяння повторному використанню коду. Немає необхідності кожного разу розпочинати з нуля, ви можете легко використати написане іншими, щоб заощадити час і енергію. Оскільки бібліотека має вузьку сферу застосування, API- інтерфейси також менше, і, кінець кінцем, проекту потрібно менше залежностей для досягнення мети. Припустимо, що у бібліотеці Util Java є метод reverse () для звернення до будь-якого рядка. Тепер вам не треба оголошувати змінні і запускати цикл для звернення рядка, ви можете викликати sb.reverse () і все. В той самий час, фреймворки - це фундамент, на якому

розробники створюють програми для конкретних платформ. Вони призначені для зменшення числа проблем, з якими програміст стикається під час розробки. Фреймворки можуть мати певні або невизначені функції і об'єкти, які програміст може використати або перезаписати для створення додатки. Основне їх завдання - надання стандартизованого коду, який може бути застосований до різних проектів додатків.

Якщо ви знаєте або працювали з C ++ або Java, ви можете зв'язати цю концепцію з реалізацією абстрактних функцій/методів. Фреймворки мають ширше охоплення і включають практично все, що вимагається для створення призначеного для користувача застосування. І бібліотеки, і фреймворки грають життєво важливу роль в розробці програмного забезпечення.

Бібліотека виконує визначену або чітко визначену операцію, тоді як фреймворк надає каркас, де програмісти визначають зміст додатка в операції. Але коли справа доходить до ключової різниці, я б сказав - інверсія контролю. Коли ми викликаємо функцію або метод з бібліотеки, ми контролюємо ситуацію. Але в іншому випадку фреймворк викликає наш код і, таким чином, управління інвертується. У більшості випадків структура тільки забезпечує концепцію. Робота додатка полягає в подальшому визначенні функціональності для кінцевих користувачів. Тисячі бібліотек доступні практично для кожної мови і допомагають програмістам повторно використати код, написаний іншими.

З іншого боку, фреймворки допомагають їм зосередитися на розвитку, а не на проблемах. Дійсно, сам React це тільки рівень представлення. Але багато хто використовує React в зв'язці з Redux, а ще є Relay, який позиціонує себе як фреймворк. Фреймворк - це рішення для реалізації архітектури додатку, коли всі рішення будуть ґрунтуватися на підходах, продиктованими фреймворком. Бібліотека - набір інструментів для вирішення приватного завдання і підхід поширюється на реалізацію одного з компонентів, але не на весь додаток в цілому. React це саме не фуллстек, але все таки фреймворк, оскільки використовуючи його - 99% застосувань повністю будується з використанням

його підходів, не відходячи від його ідеологій[17].

РОЗДІЛ 2. ПОРІВНЯЛЬНИЙ АНАЛІЗ АКТУАЛЬНИХ ФРЕЙМВОРКІВ

2.1 Порівняння React з іншими фреймворками

В будь-якій сфері, вибір покупця чи користувача на рахунок того, що вибрати або, що купити не може обійтись без порівняння. Кожна людина при виборі будь чого, завжди хоче обрати найкраще і сфера IT не є виключенням. Вибір фреймворку чи бібліотеки не лише для написання сайту, але і для простого вивчення, є дуже важливою складовою початку роботи в сфері розробки web-додатків. На сьогоднішній день багато хто починає гніватися, якщо бачать, що React знаходиться в одному списку з фреймворками(Angular, Ember, Vue і так далі). Проте, все ж таки, в тандемі з тим же Redux, React сміливо можна назвати фреймворком. Порівнювати React будемо з такими фреймворками, як Angular та Vue. Для цього потрібно розібратись, що таке Angular та Vue, їх переваги та недоліки в порівнянні з React.

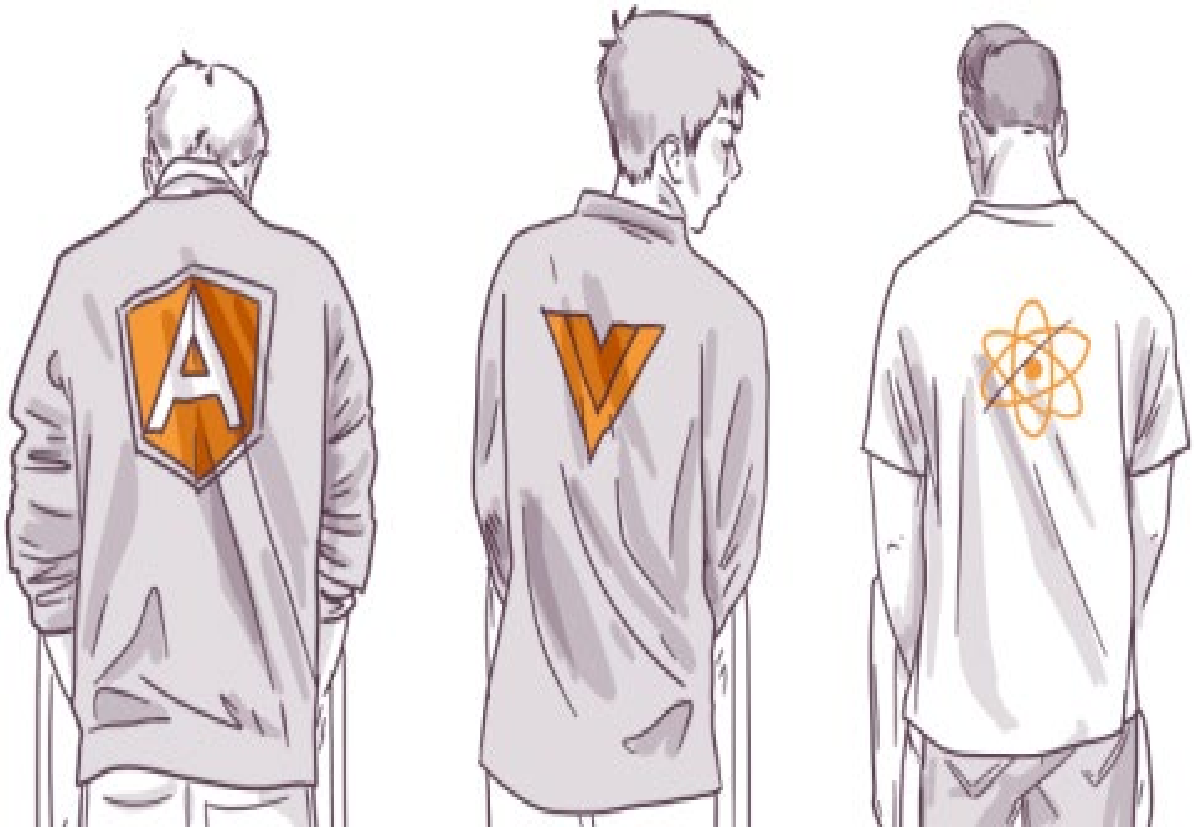

Рис. 2.1 – Angular, Vue та React [2]

Vue. Vue - це молодий фреймворк із зростаючою аудиторією. Найпростіший для вивчення в трійці, можна почати роботу "з коробки", при

цьому досить потужний для професійних розробників. Vue не має такої кількості вбудованих функцій, як Angular, але їх більше, ніж у React. Популярний вибір серед новачків, є детальна документація різними мовами, в тому числі і українською.

Переваги Vue.js:

- Vue.js має характеристики, які дуже схожі з характеристиками Angular, а це, використовуючи різні компоненти, може допомагти в оптимізації блоків HTML.
- Велика кількість детальної літератури, яка дозволяє користувачам, які вирішили вивчити цей фреймворк, заощадити багато часу.
- Схожість з іншими фреймворками з позиції архітектури та дизайну дозволяє швидко переходити від інших застосунків до Vue.
- Дуже добре підходить для того, щоб інтегрувати невеликі інтерактивні елементи в основний додаток без ризику неуспішної компіляції, а також для створення додатків різного рівня складності.
- Прекрасний в плані розробки багаторазових шаблонів великого розміру, які можуть бути зроблені за достатньо короткий срок.
- Vue.js зберігає всі свої переваги в плані гнучкості та швидкості при розмірі близько двадцяти кілобіт. Це дозволяє цій платформі досягти набагато кращої продуктивності в порівнянні з іншими.

Недоліки:

- Незважаючи на достатню кількість літератури, число обговорень чи проектів в інтернеті дуже невелике через те, що дана платформа була створена відносно недавно.
- Якщо Ваш проект на Vue дуже великий, все ж таки можуть виникнути проблеми з інтеграцією елементів, а інформацію для того, аби вирішити дану проблему знайти дуже важко.

Компанії, які використовують Vue.js: Xiaomi, Alibaba, WizzAir, EuroNews, Grammarly, Gitlab і Laracasts, Adobe, Behance, Codeship, Reuters.

Angular. Angular забезпечує більше всяких штук з коробки, в тому числі

він більш вдалий CLI, ніж React або Vue, внаслідок чого побудова серйозних рішень є більш простим. Angular популярний в ентерпрайз-середовищі, де має широку публіку. Код є більш перенасиченим і складним у порівнянні з іншими фреймворками, але це справа звички. Angular з кожним роком зростає у популярності. Станом на травень 2021 року, кожного дня завантажується близько 1,5 мільйонів Angular/core, а кількість користувачів подвоюється кожного року.

Дана платформа ідеально підходить для розробки односторінкових веб-ресурсів, які мають у своєму складі однуHTML сторінку, та CSS в поєднанні з JS. Його мета полягає в тому, щоб на основі шаблону MVCрозширяти браузерні додатки, а також спрощувати їх розробку та тестування..

Переваги Angular:

- Забезпечення підтримки роботи Typescript.
- Можливість автоматичного заповнення шаблону компоненти html через наявність ALS.
- Наявність нових функцій, які використовують бібліотеки npm, наприклад -generation Angular.
- Великий обсяг літератури, яка дозволяє розробнику відшукати всю необхідну інформацію.
- Передбачувану поведінку додатку забезпечує одностороння прив'язка даних, яка зводить до мінімуму ризик можливих помилок.
- MVVM (Model-View-ViewModel), надає розробникам можливість працювати над одним розділом програми окремо, при цьому застосовувати один і той же набір даних.
- Структура і архітектура, спеціально створені для великої масштабованості проекту.

Недоліки Angular:

- Якщо порівнювати з більшістю різноманітних фреймворків, то вивчення ангуляру ускладнюється через його різноманітність, а саме структур, таких якInjectables, Pipes, Modules та інших. Для порівняння, у React та Vue.js, є лише

«Component».

- Якщо брати до уваги різні показники продуктивності, то у даної платформи вона відносно повільна.

Компанії, які використовують Angular: Microsoft, Autodesk, MacDonald's, UPS, Cisco Solution Partner Program, AT & T, Apple, Adobe, GoPro, ProtonMail, Clarity Design System, Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase.[1]

React. React пропонує просте і функціональне створення компонентів, а також пропагує їх використання для підтримки елегантного коду API. Фреймворк дуже популярний, особливо в різних стартапи. Маючи великий вибір легкодоступних опенсорсний плагінів і розширень, можна розробити практично будь-який тип веб-сайту. Шаблони компонентів створюються з використанням JSX, це може трохи відрізнятися від того, до чого ви звикли.

Наявність великого вибору розширень - це здорово, але для початківців розробників може виявитися досить заплутаним [14].

Переваги React:

- JSX та простий дизайн дозволяють новим користувачам легко користуватись даною платформою. Звичайно, не можна забувати і той факт, що кількість літератури та інформації в інтернеті про вивчення реакту дуже велика.
- Розробники надають перевагу написанню JavaScript відповідно з сучасними стандартами і витрачають менше часу на код, який специфічний виключно для фреймворку.
- Швидкіть це мабуть найбільш серйозна перевага, а досяглась вона через впровадження React Virtual DOM і різного рівня оптимізацій рендерингу.
- Сервер без проблем підтримує рендеринг, що робить його мабуть однією з найкращих платформ для контент-орієнтованих застосунків.
- PWA шикарно підтримується завдяки генератору додатків “create-react-app”.
- Прив'язка даних є односторонньою, що означає менше небажаних

побічних ефектів.

- Redux, найпопулярніша платформа для управління станом додатків в React, її легко вчити і використовувати.
- Реалізація концепції ФП, що в свою чергу дозволяє створювати простий код.
- Вбудована підтримка JSX.
- Якщо є потреба змінити версію, то з цим немає жодних проблем
- Навички, отримані в процесі роботи на React стануть Вам у нагоді при розробці на React Native.

Недоліки:

- Неоднозначніть користувачів в плані написання CSS в React, яка призводить до складнощів в написанні стилів.
- Для тих хто працює з ООП є погані новини, адже React відходить від застосування компонентів на основі класів, а це в свою чергу є проблемою для розробників, які працюють з цим методом.
- При початкових етапах вивчення React, початковим розробникам може заморочити голову змішування JSX та шаблонів.

Компанії, які використовують React: Facebook, Instagram, Netflix, New York Times, Yahoo, Khan Academy, Whatsapp, Codecademy, Dropbox, Airbnb, Asana, Atlassian, Intercom, Microsoft, Slack, Storybook і багато інших[13].

2.2 Порівняння React та Vue

Якщо проаналізувати всі переваги та недоліки даних фреймворків, то можна зробити висновок, що основне порівняння React потрібно проводити саме з Vue.

React і Vue багато в чому схожі. Вони обидва:

- використовують Virtual DOM
- надають реактивність і компонентну структуру
- фокусуються на кореневої бібліотеці, виносячи інші питання, такі як роутинг або управління глобальним станом додатку, в додаткові бібліотеки

Внаслідок таких схожих ніш, потрібно приділили порівнянню цих фреймворків найбільше часу. Моєю метою було не лише упевнитися в технічній точності, але також і зберегти баланс. Можна вказати, де React перевершує Vue, наприклад - у багатстві екосистеми і достатку доступних призначених для користувача засобів промальовування проте, не варто забувати і про пункти, де Vue виявляється кращим за React. Зрозуміло, що і в технічних питаннях існують елементи смаку, і в основному мета цього порівняння вказати причини, по яким Vue чи React може виявитися корисним.

Почнемо з швидкодії виконання. Як React, так і Vue - виключно швидкі, тому швидкість навряд чи буде вирішальним фактором при виборі між ними. Якщо вас цікавлять цифри, то можете вивчити стороннє порівняння продуктивності , яке фокусується на базі сирої продуктивності відтворення / поновлення дерева дуже простих компонентів. Щодо оптимізації, то у React, коли стан компоненти змінюється, воно запускає повторне промальовування всього піддерева компоненти, починаючи з себе. Щоб уникнути непотрібного повторного відтворення дочірніх компонентів, вам потрібно або використовувати Pure Component, або реалізовувати should Component Update всюди де це можливо. Вам також може знадобитися використовувати незмінні (immutable) структури даних, щоб зробити зміни вашого стану більш зручними до оптимізації. Однак, в деяких випадках ви не можете розраховувати на таку оптимізацію, тому що PureComponent/shouldComponentUpdate припускають, що відображення всього піддерева визначається даними поточного компонента. Якщо це не так, то така оптимізація може призвести до неузгодженим станом DOM.

У Vue залежності компонента автоматично відслідковуються під час відтворення, тому система точно знає, які компоненти дійсно необхідно повторно малювати при зміні стану. Кожен компонент можна розглядати як той, який має shouldComponentUpdate, автоматично реалізований для вас, без обмежень для вкладених компонентів.

В цілому, це усуває необхідність цілого класу оптимізацій продуктивності

для розробника, що дозволяє більше зосередитися на побудові самого додатка в міру його масштабування.

Не менш важливою складовою для будь-якого web-додатку є масштабування. Для великих додатків, як Vue так і React надають надійні рішення для роутінга. Спільнота React також породила досить інноваційні рішення в галузі управління станом додатку (див. Flux / Redux). Ці підходи, і навіть сам Redux легко інтегруються в додатки на Vue. Насправді, Vue зробив наступний крок, створивши Vuex - натхненну Elm реалізацію паттерну управління станом додатку. Vuex глибоко інтегрований з Vue, що, неабияк полегшує життя розробникам.

В якості ще однієї важливої відмінності між React і Vue можна згадати той факт, що всі додаткові бібліотеки Vue, включаючи бібліотеки для керування станом додатки і для роутінга (серед інших завдань), офіційно підтримуються в актуальному відповідно до ядром бібліотеки. React, навпаки, вважає за краще віддати ці питання на відкуп спільноті, тим самим створюючи більш фрагментовану екосистему. Втім, як уже зауважувалося раніше, в силу популярності React, його екосистема значно ширша, ніж у Vue.

Також варто зазначити що у Vue є CLI генератор проектів , який дозволяє легко почати новий проект за допомогою інтерактивного майстра. Його також можна використовувати для миттєвого прототипування компоненту. React також робить успіхи в цій галузі за допомогою create-react-app , але в даний час у нього є кілька обмежень:

- Він не допускає ніякої конфігурації під час створення проекту, в той час як Vue CLI працює поверх оновлюваної runtime-залежності, яка легко розширюється за допомогою плагінів .
- Існує тільки один шаблон для односторінкового додатку, в той час як Vue пропонує широкий вибір опцій за замовчуванням для різних цілей і систем збірки.
- Немає можливості створювати проекти з користувацьких пресетів налаштувань , що може бути особливо корисно для enterprise-оточень з

усталеними раніше угодами.

Важливо зауважити, що багато з цих обмежень - слідства свідомо прийнятих рішень команди create-react-app, і в них є і свої плюси. Наприклад, якщо ваш проект не вимагає для користувача настройки процесу складання, його можна буде оновлювати як залежність.

React відомий своєю досить крутою кривою вивчення. До того моменту, коли новачок зможе щось написати, йому доведеться дізнатися про JSX, а ймовірно - і про ES2015 +, оскільки багато прикладів використовують синтаксис ES2015-класів. Крім того доведеться розібратися з системами збірки, оскільки, хоча технічно і існує можливість використовувати Babel самостійно для live-компіляції коду, для production цей підхід в будь-якому випадку не годиться.

Vue масштабується вгору нітрохи не гірше, ніж React, і в той же час його можна масштабувати і вниз - аж до варіанту використання разом з jQuery. Саме так - для старту в найпростішому випадку досить просто додати тег скрипта на HTML-сторінку.

```
<script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
```

Починаючи з цього моменту можна писати код на Vue, і навіть використовувати production-версію, не страждаючи докорами сумління і хвилюваннями щодо продуктивності.

Оскільки знання JSX, ES2015 і систем збірки не потрібно для початку роботи з Vue, в середньому у нових розробників йде не більше дня на читання керівництва , що дозволяє дізнатися досить інформації для побудови нетривіальних додатків.

Ну і звичайно, не можна обійти стороною використання HTML & CSS.

У React абсолютно все - це JavaScript. Не тільки структури HTML, виражені через JSX, останні тенденції також включають управління CSS всередині JavaScript. Цей підхід має свої переваги, але також змушує йти на компроміси, які можуть здатися неефективними для кожного розробника.

Vue охоплює класичні веб-технології та ґрунтується на них. Щоб показати вам що це значить, розглянемо кілька прикладів.

JSX vs Шаблони

У React всі компоненти реалізують свій UI в render-функціях з використанням JSX, декларативним XML-подібним синтаксисом, який працює в JavaScript.

Render-функції з JSX мають кілька переваг:

- Ви можете використовувати всі можливості мови програмування (JavaScript) для побудови свого уявлення. Це включає в себе тимчасові змінні, управління розгалуженням і прямі посилання на значення JavaScript в області видимості.
- Підтримка інструментів (наприклад, лінтінг, перевірка типів, автодоповнення в редакторі) для JSX в деяких відношеннях більш просунута, ніж те, що є в даний час для шаблонів Vue.

У Vue також є `render` - функції і навіть підтримка JSX , тому що іноді ці можливості справді потрібні. Проте, за замовчуванням Vue пропонує шаблони як більш просту альтернативу. Будь-який валідний HTML також буде валідним шаблоном Vue, і це призводить до виникнення кількох переваг:

- Для багатьох розробників, які працюють з HTML, шаблони просто більш природні для читання і написання. Якщо це робить розробника більш продуктивним, то перевага очевидна.
- HTML-шаблони полегшують поступову міграцію існуючих додатків для використання можливостей реактивності Vue.
- Це також полегшує дизайнерам і менш досвідченим розробникам розбиратися і вносити доопрацювання в поточну кодову базу.
- Ви можете навіть використовувати препроцесори, такі як Pug (раніше відомий як Jade), щоб створювати ваші шаблони у Vue.

Є люди, які стверджують, що важливо вивчити додаткову мову DSL для написання шаблонів – проте я вважаю, що ця різниця в кращому випадку поверхнева. По-перше, JSX не означає, що користувачеві не потрібно нічого

вчити - це додатковий синтаксис на основі простого JavaScript, тому він простий в освоєнні для будь-якого, хто знайомий з JavaScript, але говорити, що це просто для всіх, було б помилкою. Точно так само шаблон є просто додатковим синтаксисом поверх простого HTML і, отже, має дуже низький поріг входження для тих, хто вже знайомий з HTML. За допомогою DSL Vue можедопомогти користувачеві зробити більше з меншою кількістю коду (наприклад, з v-on модифікаторами). Схожа завдання може включати в себе набагато більше коду при використанні простих функцій JSX або render-функцій.

На більш високому рівні можна розділити компоненти на дві категорії: презентаційні та логічні. Взагалі рекомендовано використовувати шаблони для презентаційних компонентів і render-Функції / JSX для логічних. Процентне співвідношення цих компонентів залежить від типу програми, але зазвичай презентаційні компоненти більш поширені[19].

Модульний (компонентний) CSS

За винятком випадків поділу компонентів на кілька файлів (наприклад, за допомогою CSS-модулів), для обмеження області видимості CSS в React зазвичай використовується підхід CSS-in-JS (наприклад, styled-components і emotion). Це являє собою новий компонентно-орієнтований підхід до стилізації, який відрізняється від звичайного процесу розробки CSS. Крім того, незважаючи на підтримку вилучення CSS в окремий файл стилів на етапі складання, як і раніше може бути необхідно, щоб під час виконання були підключені в збірку для коректної роботи стилізації. У той час, як ви отримуєте доступ до динамічності JavaScript при створенні ваших стилів, цей компроміс часто збільшує розмір збірки і час виконання.

Якщо ви шанувальник підходу CSS-in-JS - багато популярних бібліотек підтримують Vue (наприклад, styled-components-vue і vue-emotion). Головною відмінністю між React і Vue тут буде те, що за замовчуванням стилізація Vue виконується через знайомі теги “style” в однофайлових компонентах, що є перевагою для людей, які в програмуванні вже довгий час. Однофайлові

компоненти надають вам повний доступ до CSS в тому ж файлі, що і решта коду вашого компонента.

```
<style scoped>
  @media (min-width: 250px) {
    .list-container:hover {
      background: orange;
    }
  }
</style>
```

Опціональний атрибут `scoped` автоматично обмежує область видимості CSS поточним компонентом, додаючи елементам унікальні атрибути (такі як `data-v-1-21e5b78`), і компілюючи `.list-container:hover` в будь-що-небудь на зразок `.list-container[data-v-1-21e5b78]:hover` [4].

Рис. 2.2 – Відмінність Vue та React

Нарешті, стилізація в однофайлових компонентах Vue дуже гнучка. За допомогою vue-loader , ви можете використовувати будь-який препроцесор, постпроцесор і навіть глибоку інтеграцію з CSS-модулями - все в елементі <style>[15].

2.3 Порівняння React та Angular

Насправді, як React, так і Angular чудово підходять для інтерфейсної розробки. І вони однаково добре справляються зі створенням великомасштабних додатків.

Але є одна велика різниця між Angular та React: React.js використовує віртуальний DOM (об'єктна модель документа - дозволяє отримувати доступ та змінювати вміст документа, макет та навіть структуру). Тоді як Angular працює на реальному DOM. Уявіть, ви хочете оновити дані профілю користувача, скажімо, їх прізвище, або змінити опис якогось товару у Вашому інтернет-магазині. Real DOM, замість того, щоб змінювати лише цю частину інформації, оновлює всю деревоподібну структуру таблиць HTML, поки не дійде до необхідних даних. У нашому випадку це прізвище.

Хоча віртуальний DOM дозволяє нам оновлювати зміни, не перезаписуючи весь HTML-документ віртуально. Це робить оновлення набагато швидше та забезпечує швидку роботу - незалежно від розміру програм. Так, Angular це повноцінний фреймворк, проте реальна функція DOM робить додатки повільнішими при обробці запитів даних. Якщо провести алегорії цих платформ з будь-якою компанією, то Angular це та фірма, яка зробить все по одному шаблону. Вона вже давно на ринку і не дасть Вам великий вибір, але будьте впевнені, що про кінцевий результат Ви не будете шкодувати. За плечами цієї компанії достатньо досвіду, аби не розчарувати.В той самий час, React більш лояльний в плані вибору. Це відносно новий гравець, і він впроваджує щось нове. Ви зможете налаштувати товар під себе, змінювати його колір, характеристики, тощо. При цьому, це буде зроблено швидко, та надійно, без якогось конкретного шаблону. Так, Angular теж зробить надійно, проте це

буде не так швидко. Але при цьому, не можна віддати однозначну перевагу комусь одному, адже все залежить від типу завдання. Комусь для вирішення потреби достатньо зробити просто, та по шаблону, а комусь потрібно зробити швидко та більш креативно.

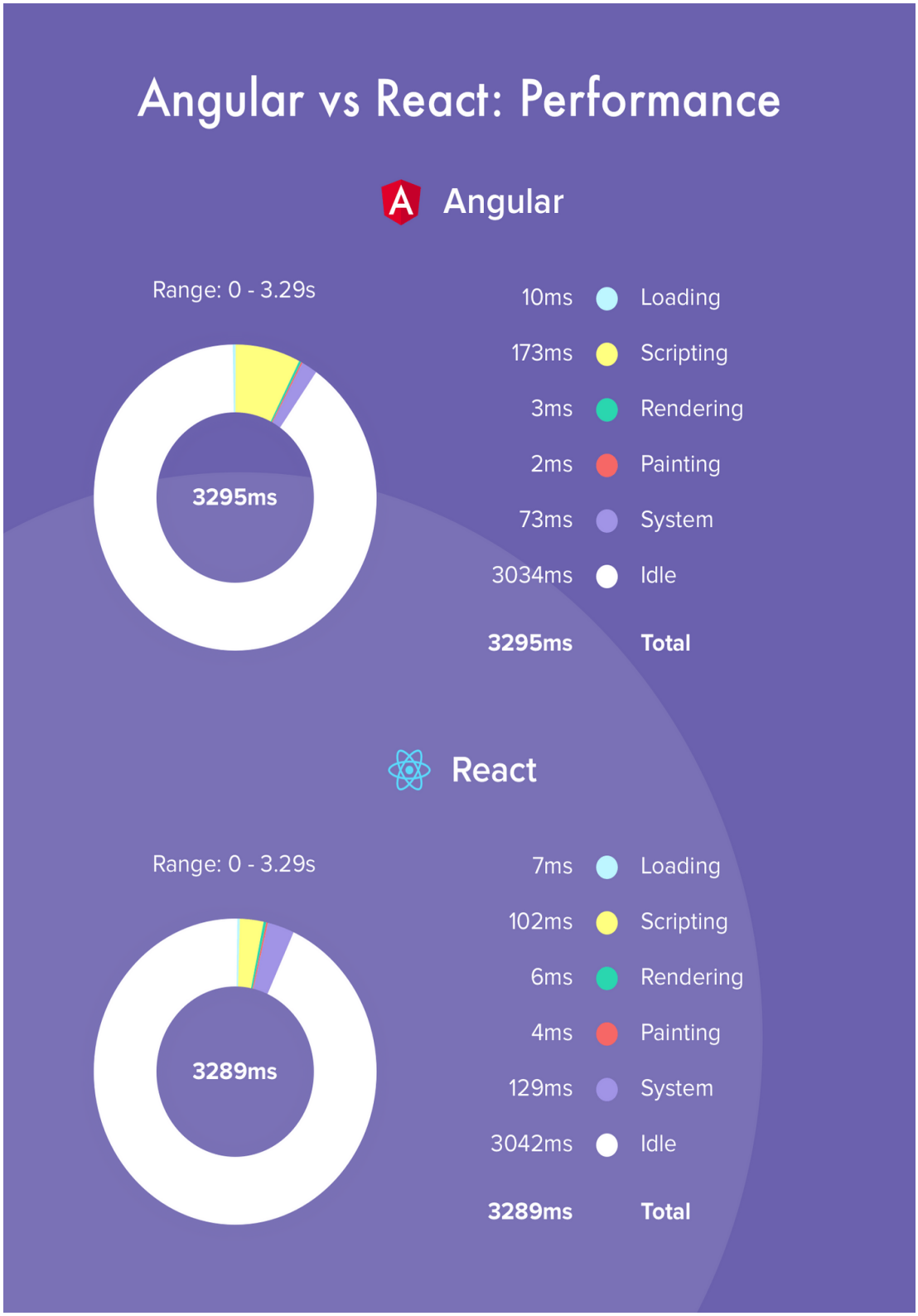

Рис 2.3 – Порівняння швидкості роботи Angular та React

По рисунку 2.3, який знаходиться на попередній сторінці можна зрозуміти, що React буде швидшим ніж Angular, хоча здається, що різниця мінімальна, проте вона буде більш відчутною при тестуванні цих фреймворків на дуже великих додатках.

Розглянемо різницю між прив’язкою даних

Прив’язка даних - це синхронізація даних між бізнес-логікою та інтерфейсом користувача.

Різниця між Angular та React.js полягає в тому, що Angular використовує одно- і двосторонню прив'язку даних : зміна даних впливає на вигляд, а зміна подання викликає зміни в даних(рис.13).

React використовує одностороннє прив’язку : при розробці програми, розробники часто вкладають дочірні компоненти в батьківські компоненти вищого порядку.

Одностороння прив'язка робить код стабільнішим, а також значно полегшує налагодження збірки додатків за допомогою React та Angular. Тим не менше, одностороннє / двостороннє прив'язування Angular простіше в роботі і робить фреймворк більш гнучким.

Розглянемо детальніше. Коли Angular встановлює прив'язку даних, існують два "спостерігачі"(це спрощення)

```
//js
$scope.name = 'test';
$timeout(function()  { $scope.name = 'another' }, 1000);
$timeout(function()  { console.log($scope.name); }, 5000);

<!-- html -->
<input ng-model="name" />
```

Введення почнеться з test, потім another обновится до 1000 мсек. Будь-які зміни $scope.name з коду контроллера або зміни введення будуть відбиті в журналі консолі через 4000 мсек. Зміни <input />у $scope.nameвідбиваються у

властивості автоматично, оскільки ng - modelsetup відстежує введення і повідомляє $scope про зміни. Зміни з коду і зміни з HTML є двосторонніми [5].

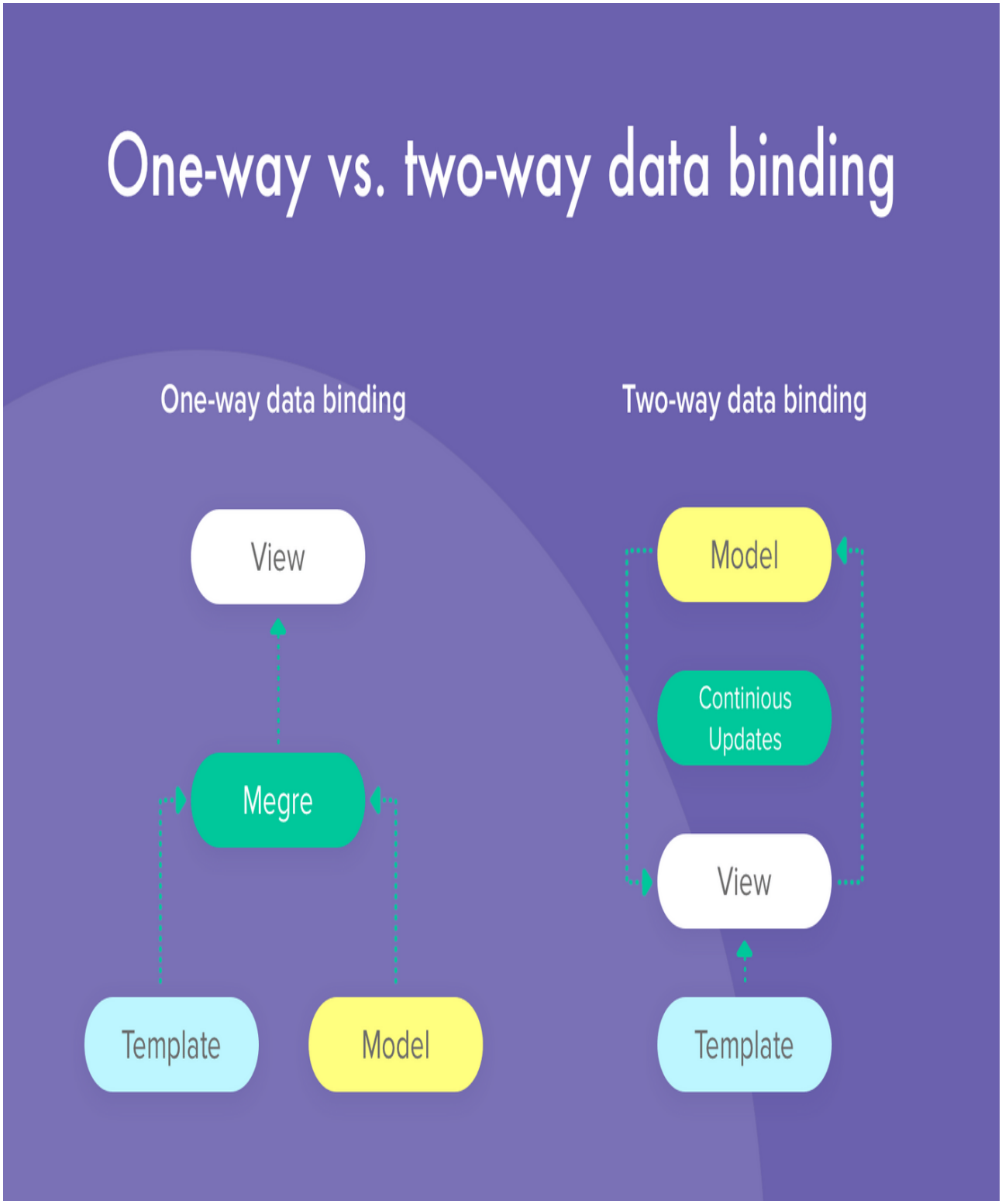

Рис. 2.4 – Схема одно- та двосторонньої прив'язки даних

В плані архітектури компонентів React та Angular суттєво відрізняються, адже все ж таки Angular – це повноцінний фреймворк. Ось як це працює: ви

створюєте компоненти, які управляють власним станом, і структурують їх разом у більш складні інтерфейси. Проте, якщо архітектура проекту базується на React, вам знадобляться багаторазові інтеграції та допоміжні засоби. Деякі з них:

- Redux (контейнер, який прискорює роботу React у великих програмах)
- Babel (перетворює JSX на JavaScript, щоб програма була зрозумілою для браузерів)
- Webpack (стандартний пакет модулів)

На відміну від React, Angular - це суто повноцінний фреймворк, який має безліч готових функцій, таких як:

- RxJS вводить поняття реактивного програмування в JS
- Angular CLI - це потужний інтерфейс командного рядка
- Angular Universal використовується для візуалізації на стороні сервера

Як правило, і Angular, і React.js мають надійні екосистеми. React легше зрозуміти, але для повного використання його потенціалу потрібні численні інтеграції, такі як Redux.

До речі, оскільки React - це не фуллстек фреймворк(бібліотека), ви можете інтегрувати її в будь-який проект, навіть якщо сам проект написаний на Angular[16].

Мною вже було сказано, що React користується найбільшою популярністю серед користувачів, проте ми розглянули результати опитування на Stack Overflow. Але є ще такий популярний серед розробників сервіс, як GitHub.

Серед користувачів цього сервісу також була виведена статистика, з приводу того, що має більшу популярність, React чи Angular (рис.2.5).

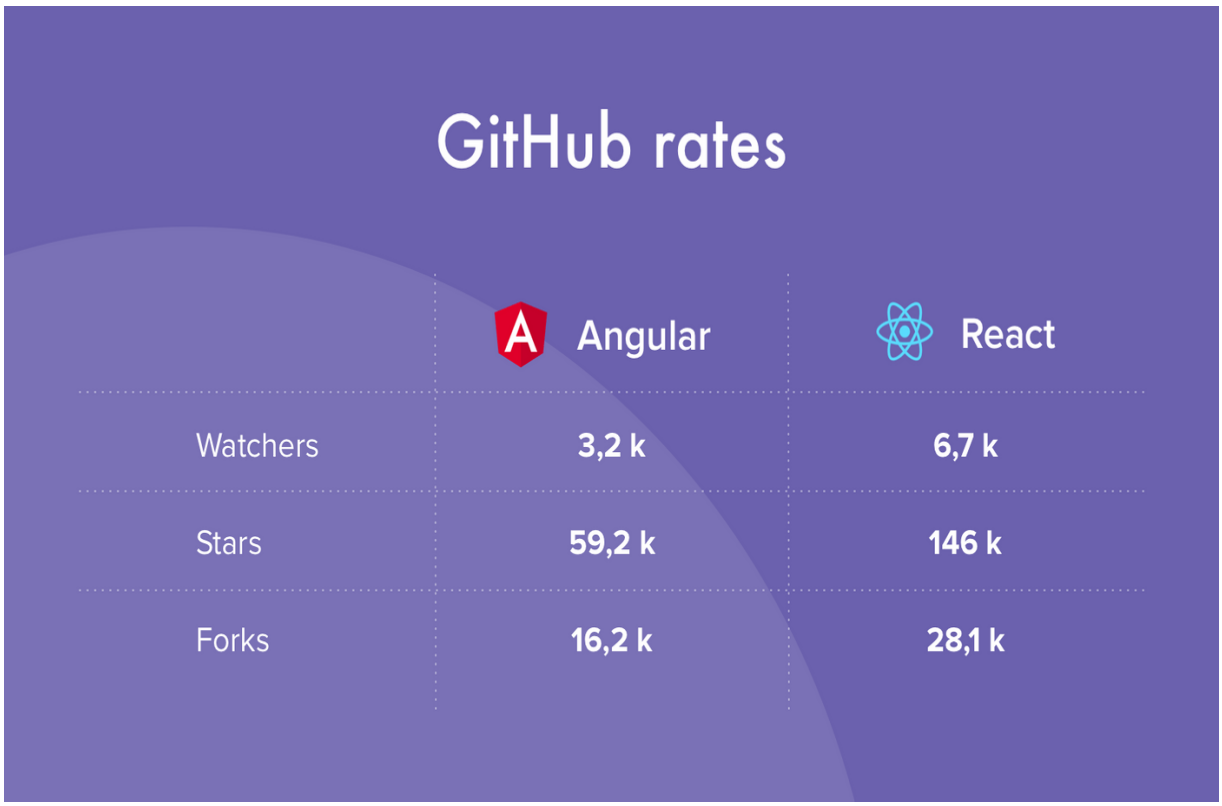

Рис. 2.5 – Дані про популярність Angular та React

І знову ж таки, React отримує чисту перемогу над Angular. Статистика з NPM показує наступні результати: NPM нараховує8 140 830 завантажень щотижня, що виглядає вражаюче. Щодо Angular то все не так добре: 1 609 139 завантажень щотижня. Але як щодо статистики серед розробників, в якому обсязі вони використовують той чи інший застосунок? Згідно з опитуванням на Stack Overflow, про яке я щойно згадав, більше розробників заявили, що обирають React замість Angular: **31,3%** React.js проти **30,7%** Angular. Крім того, ось що показує опитування на **stateofjs.com**(21,717 респондентів):

- **React** - 71,7% використовували React раніше і будуть використовувати його знову (**задоволені користувачі** : 89,33% серед 16 099 користувачів)
- **Angular** - 21,9% використовували Angular раніше і будуть використовувати його знову (**задоволені користувачі** : 37,95% серед 11 582 користувачів)

На JetBrains ситуація також незмінна. Серед 7000 розробників статистика показує наступе:

- 54% регулярно використовують React

- 23% регулярно використовують Angular

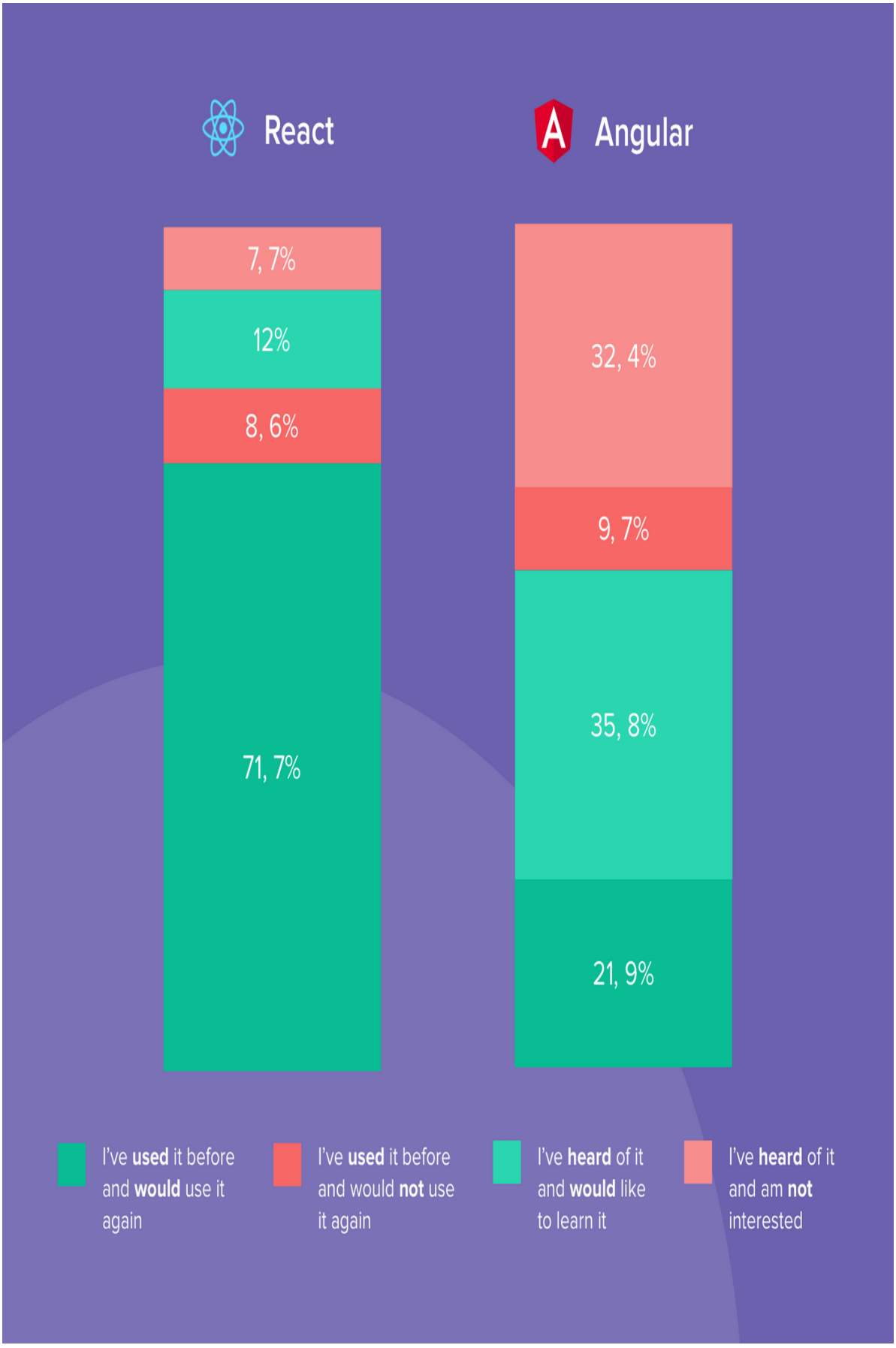

Рис. 2.6 – Статистика використання React та Angular

Знову ж таки, підведемо підсумок порівняння в таблиці, яка подана на рисунку 2.7

Рис. 2.7 – Порівняння React та Angular

РОЗДІЛ 3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Вибір середовища розробки для написання проекту

Для мене, як і для багатьох розробників, Visual Studio Code є найбільш зручним середовищем, для написання проектів. Він поєднує у собі простоту редактора вихідного коду, та потужні інструменти для розробника.

Дана програма має в собі чудовий цикл редагування, збірки, компілювання, що призводить до мінімальних затрат Вашого часу.

Велика перевага VSC в тому, що його підтримують такі операційні системи як macOS, Linux та Windows (рис.3.1).

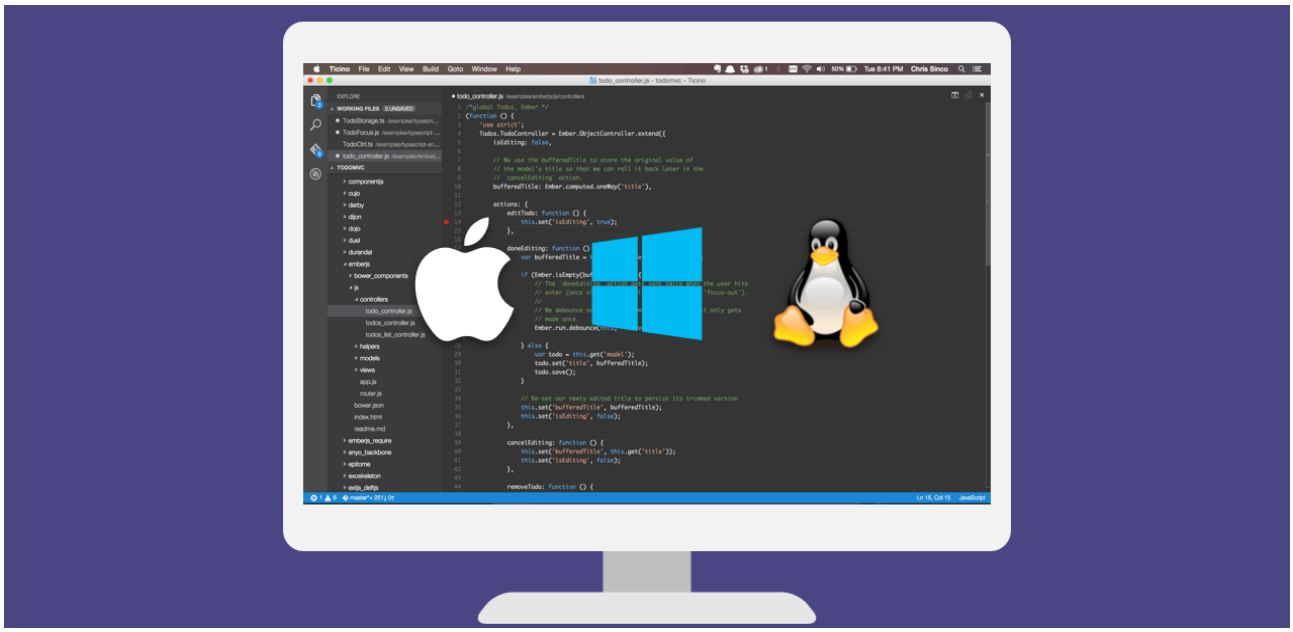

Рис. 3.1 – Підтримка VS Code трьома операційними системами

3.2 Реалізація проекту

3.2.1 Головна сторінка сайту. Перша сторінка сайту – це те, що має імпонувати користувачу. Дуже “складний” дизайн є помилкою, адже весь світ рухається в сторону мінімалізму.

Саме тому головна сторінка зроблена максимально простою та легкою для сприйняття (рис.3.2, рис.3.3).

Рис 3.2 – Головна сторінка

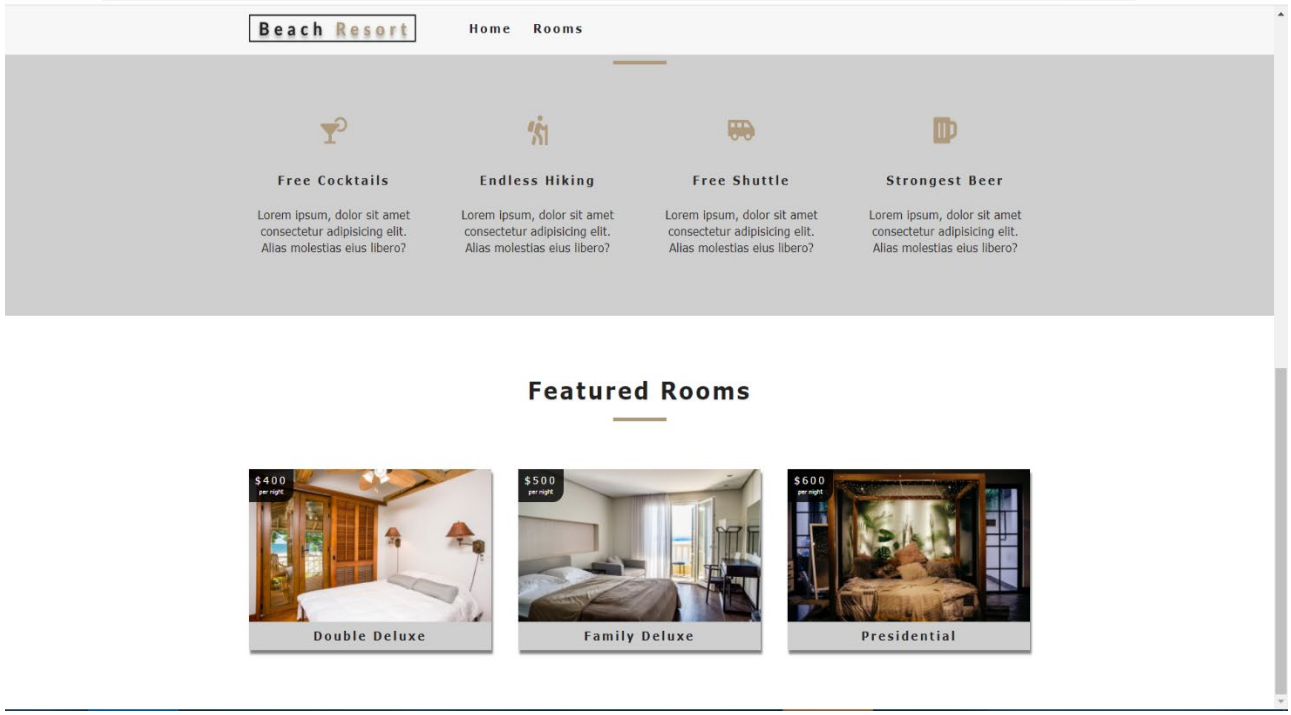

Рис. 3.3 – Головна сторінка

3.2.2 Сторінка сайту Rooms. При натисканні на “Rooms” відбувається перехід до тої частини сайту, де можна буде переходити на той номер, який Вам до вподоби (рис.3.4, рис.3.5).

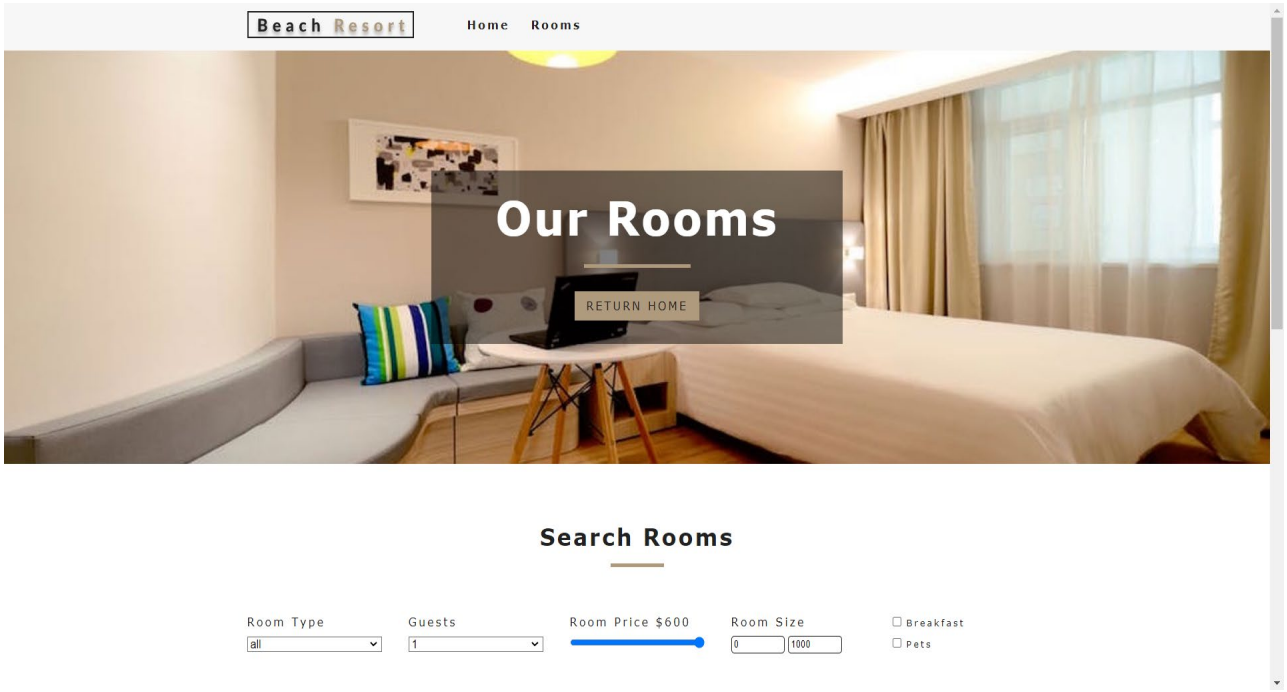

Рис. 3.4 – Сторінка Rooms

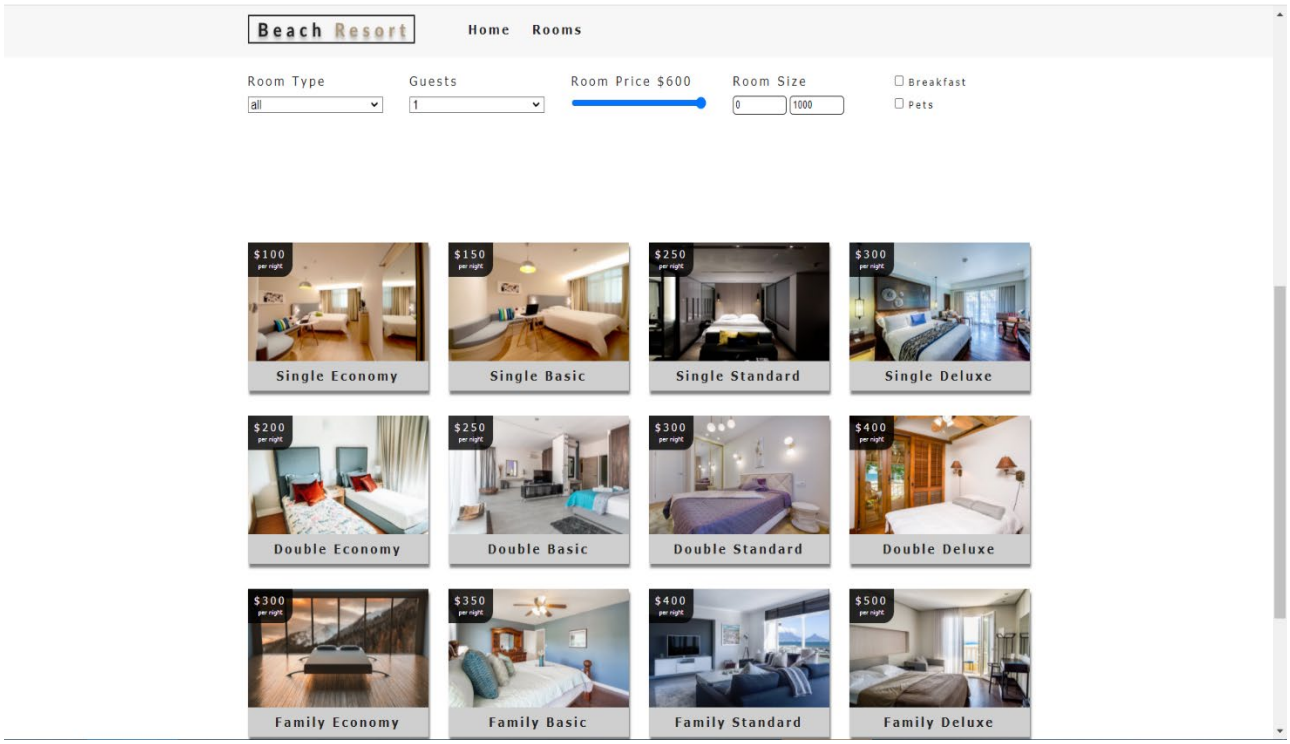

Рис. 3.5 – Сторінка Rooms

Якщо Ви помітили, то на даній сторінці реалізований фільтр, завдяки якому можна вибрати номер за його типом, вмістом людей, ціною, розміром, а також наявністю сніданку та можливість проживання з тваринкою. Якщо задати

в фільтр деякі параметри, а саме, тип кімнати – однокімнатна, кількість гостей – один, ціна – до 166 доларів, то на сторінці залишаться лише ті кімнати, які відповідають опису (рис.3.6).

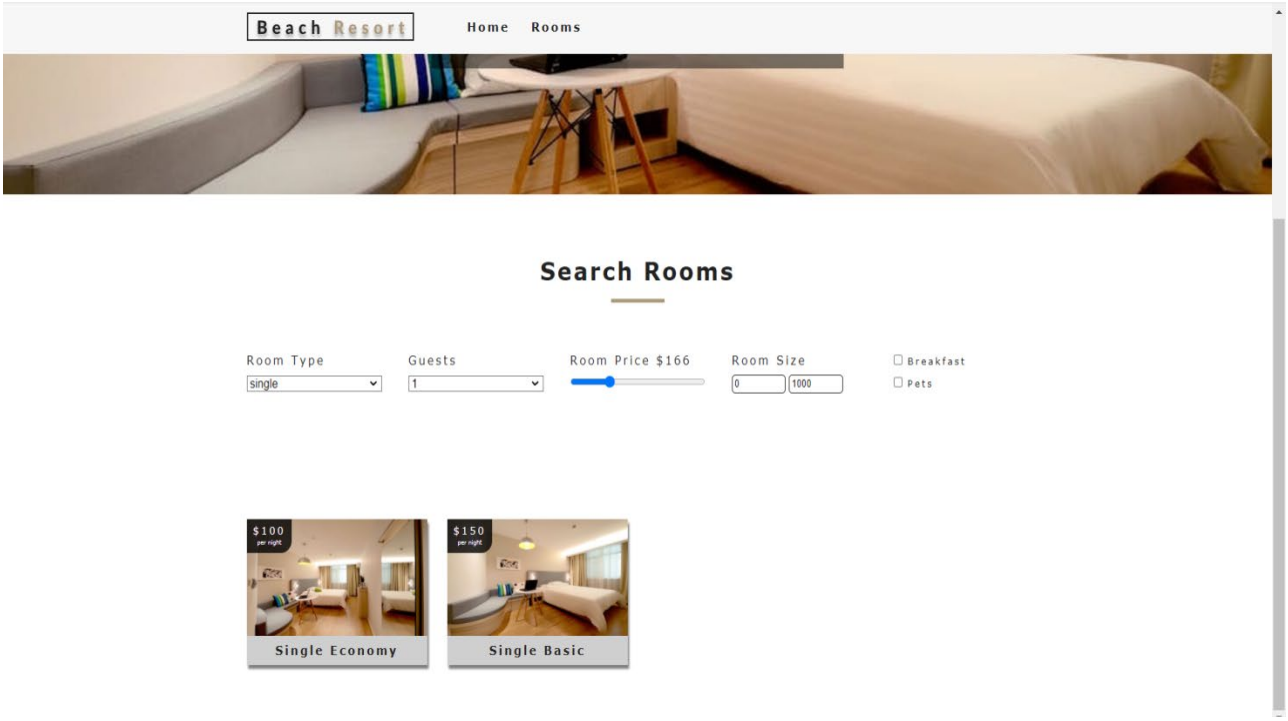

Рис. 3.6 – Реалізація фільтру

Проте, якщо при заданій інформації в фільтрі кімнати немає, то сторінка буде виглядати наступним чином(рис.3.7).

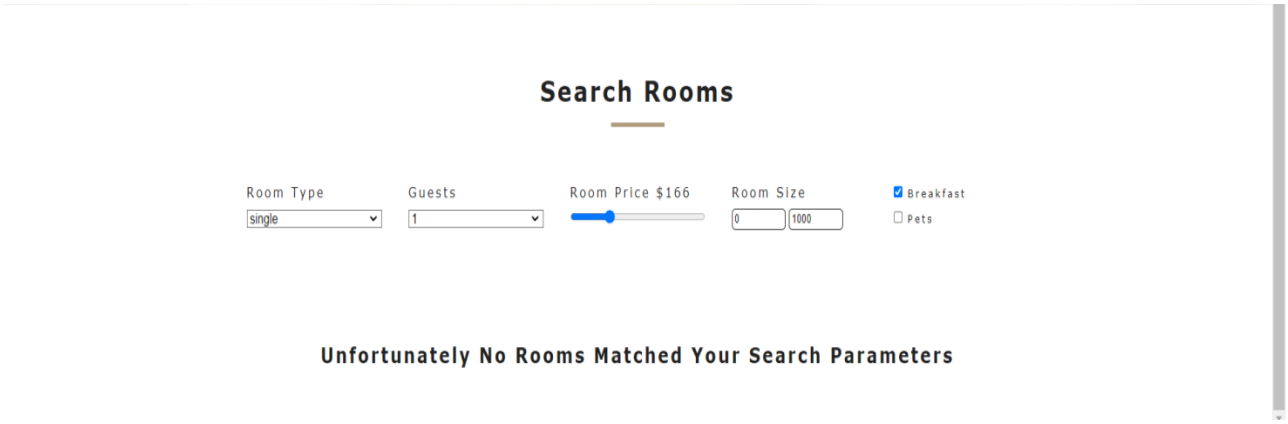

Рис. 3.7 – Реалізація фільтру

Натиснувши на будь-яку з цих кімнат, відбудеться перехід на сторінку з детальною інформацією про кімнату, яку було вибрано. Наприклад, виберемо президентський номер(рис.3.8).

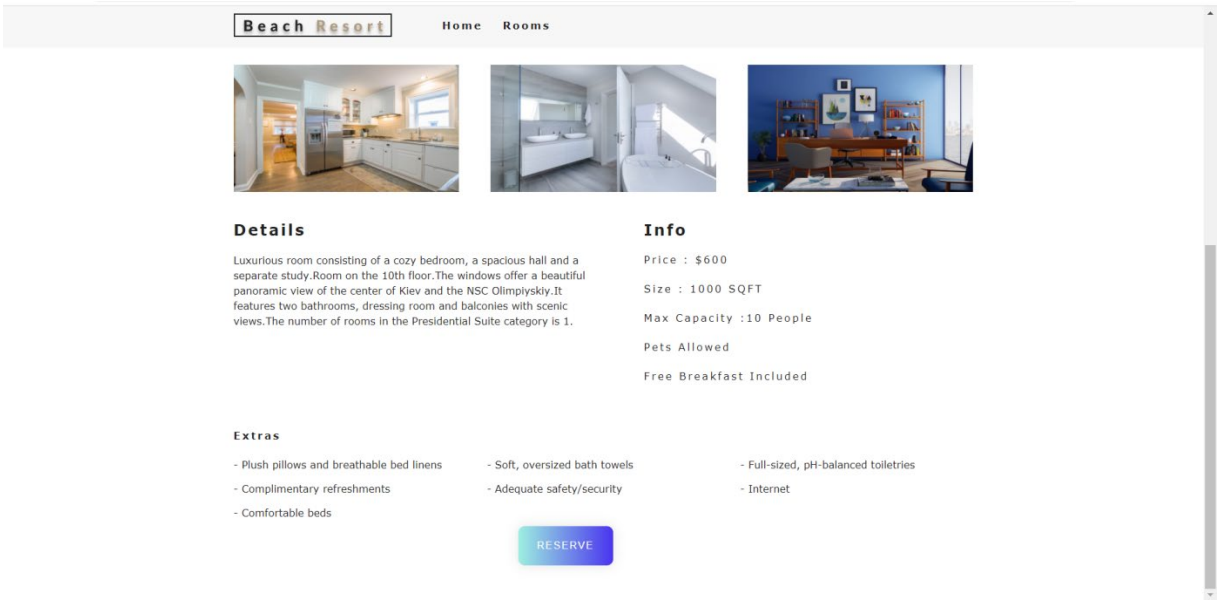

Рис. 3.8 – Детальна інформація про номер

Тут вказана вся інформація про номер і якщо потенційному клієнту дана кімната сподобалась, то при натисканні на кнопку “Reserve” його перекидає на форму для заповнення інформації.

3.2.3 Сторінка заповнення інформації. Ось так виглядає перша сторінка даної форми (рис.3.9):

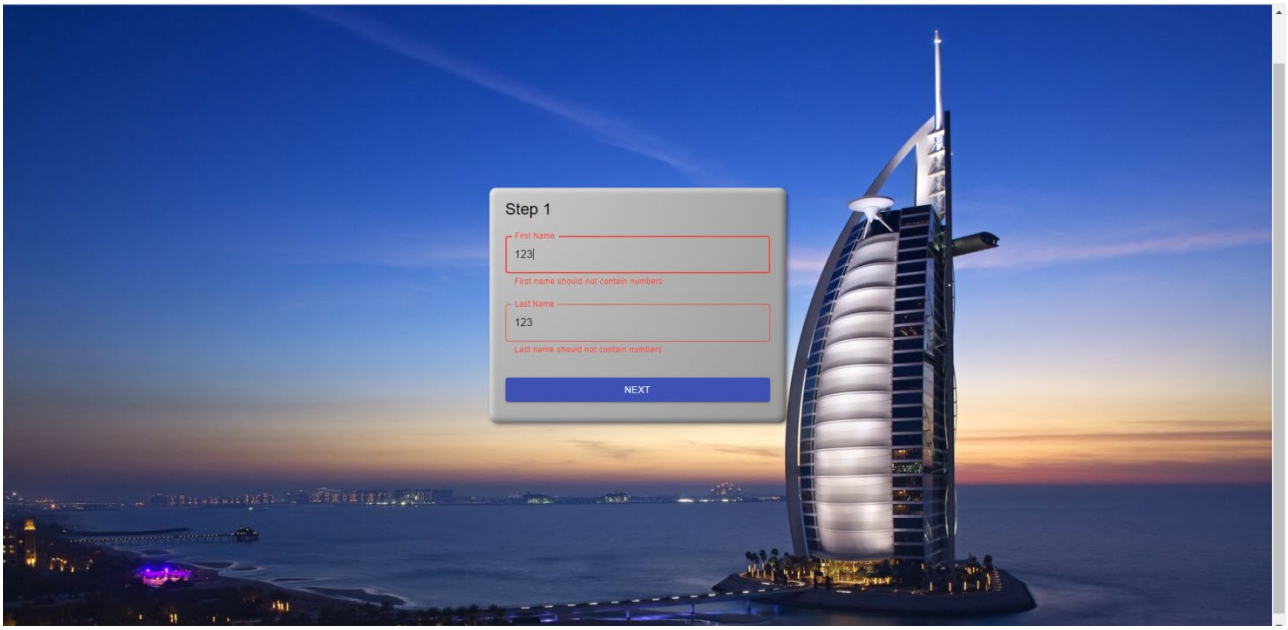

Рис. 3.9 – Перша сторінка форми заповнення інформації

При некоректному введенні імені чи прізвища, валідація не пропустить Вас далі. Те ж саме буде відбуватись, якщо дана форма буде пуста.

Аналогічно і з другим етапом(рис. 3.10)

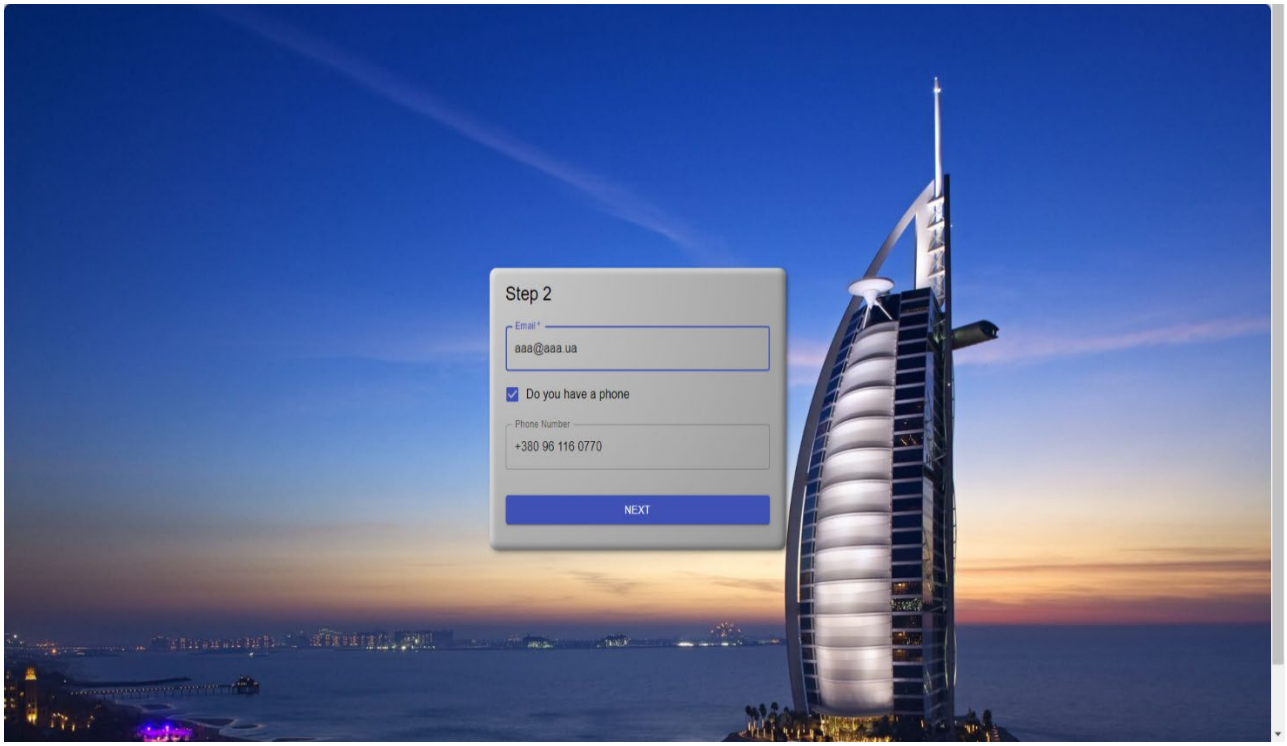

Рис. 3.10 – Друга сторінка форми заповнення інформації

В третьому етапі реалізована функція завантаження файлів, в даному випадку, фото паспорту (рис. 3.11).

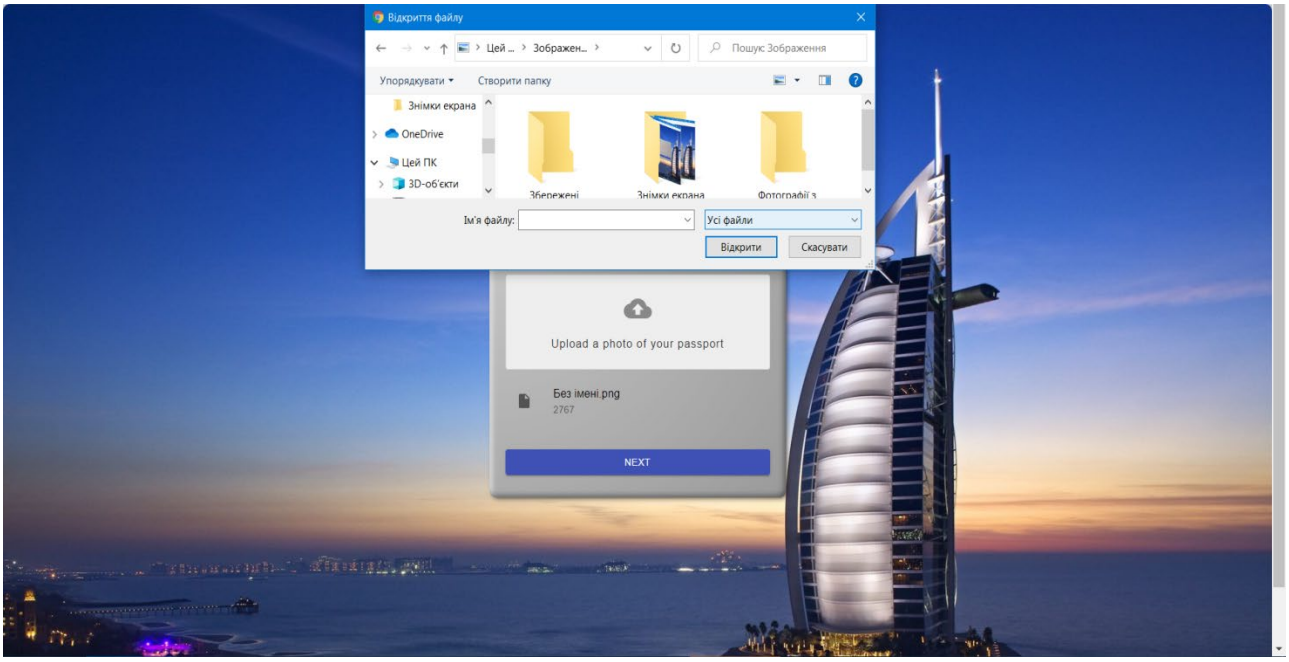

Рис. 3.11 – Третя сторінка форми заповнення інформації

Четвертий етап форми введення показує все те, що було введено на попередніх трьох, для того, щоб перевірити достовірність інформації (рис. 3.12).

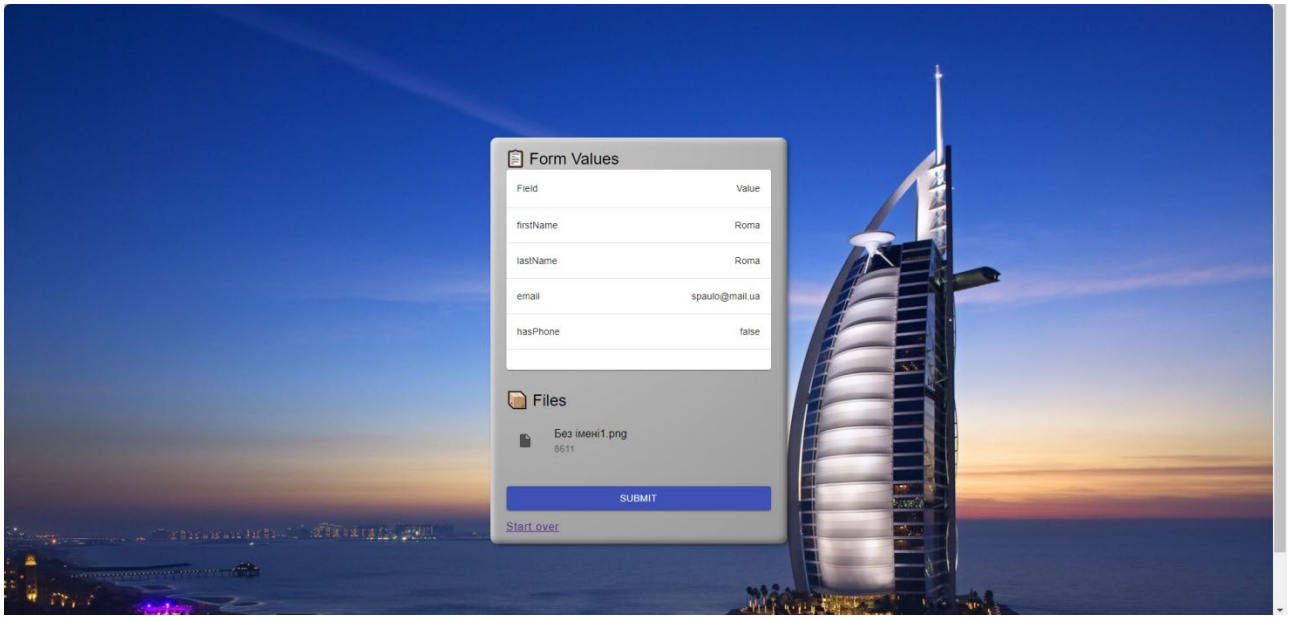

Рис. 3.12 – Четверта сторінка форми заповнення інформації

Звичайно, це лише маленька частина функціоналу. Якщо залучити в даний проект Back-end то функціонал виросте в рази.

Це і онлайн-консультант, і відсилання листа-підтвердження на email, введення БД і багато чого іншого. Моя мета була в тому, щоб зробити максимально великий функціонал, саме на Front-end.

ВИСНОВКИ

1. Можна підбити підсумок, що і Angular, і Vue, і React.js - чудові інструменти, які люблять веб-розробники. Вони мають масу переваг; але поки все ж таки, на мою думку, React є лідером, що підтверджує статистика.

2. React використовується більше, він еволюціонує і залишається модним. Крім того, React.js отримує величезну підтримку від спільноти розробників.

3. React кращий, ніж Angular**,** завдяки реалізації віртуального DOM та оптимізації візуалізації. Міграція між версіями React теж досить проста; не потрібно встановлювати оновлення по одному, як у випадку з Angular.

4. Vue поступається своєму головному конкуренту, якщо говорити про створення великих додатків, адже в своєму арсеналі React має потужний вибір бібліотек та інструментів. Проте, на мою думку, ці дві платформи фактично однакові, а різниця заключається в дрібницях. React був вибраний мною для розробки, тому що на сьогоднішній день він користується найбільшою популярністю серед користувачів та розробників.

5. Нарешті, з React розробники мають безліч існуючих рішень, які вони можуть використовувати. Це прискорює час розробки та зменшує кількість помилок. Також, перевагаReact.js полягає в тому, що він використовує ізольований дебаг, що допомагає розробникам досягти стабільності програми. Більше того, керована компонентами архітектура React дозволяє нам повторно використовувати компоненти і тим самим, економити час на розробку.

6. Хотілося б додати про неявні помилки при розробці сайту. Наприклад, не проводити аналіз сайтів конкурентів. Ми звикли робити все не так як у конкурентів, адже вважаємо, що знаємо краще як повинен працювати наш сайт. Проте, дуже важливим завданням є саме аналіз конкурентів, які вийшли в топ. Подивитись, що і як вони зробили і застосувати цей досвід на собі і при цьому, зробити краще на кожному із етапів. Також, не варто забувати про форми зворотного зв’язку з клієнтом. У кожного є свій улюблений канал комунікації.

Хтось любить телефонувати, дехто надає перевагу виключно листуванню у месенджерах. Насправді, чим більше способів зв’язку з клієнтом, тим краще. Телефони, email, онлайн-консультант, месенджери, соціальні мережі – всі ці канали повинні бути максимально доступні для користувача.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. React или Angular или Vue.js — что выбрать? https://habr.com/ru/post/476312/

2. Сравнение JavaScript фреймворков Vue.js, React и Angular (2019) https://mkdev.me/posts/sravnenie-javascript-freymvorkov-vue-js-react-i-angular

3. Angular vs. React: What to Choose for Your Web App? https://www.cleveroad.com/blog/angular-vs-react

4.Сравнение с другимифреймворками. Vue. https://ru.vuejs.org/v2/guide/comparison.html

5. React vs Angular: сравнительная информация для разработчиков https://webformyself.com/react-vs-angular-sravnitelnaya-informaciya-dlya-razrabotchikov/

6. Матеріал з Вікипедії – React. https://uk.wikipedia.org/wiki/React

7.Офіційний сайт React. https://uk.reactjs.org/

8. Разница между фреймворками и библиотеками в программировании https://senior.ua/articles/raznica-mezhdu-freymvorkami-i-bibliotekami-v-programmirovanii

9. Что такое Virtual DOM? https://habr.com/ru/post/256965/

10. Матеріал з Youtube - Что такое REACT.JS https://www.youtube.com/watch?v=Zs5UJSX9ifg

11. Матеріал з Youtube - What Is React (React js) & Why Is It So Popular? https://www.youtube.com/watch?v=N3AkSS5hXMA

12. Матеріал з Youtube - Что такое framework? Объяснение для новичков. https://www.youtube.com/watch?v=apjXI4Jw7DE

13. React в действии – Томас Марк Тиленс – СПб: Питер, 2019. — 368 с.

14. Путь к изучению React - Robin Wieruch и Alexey Pyltsyn, 2018. – 336 с.

15. Vue.js в действии - Хэнчетт Э., Листуон Б., 2020. – 304 с.

16. Angular для профессионалов - Adam Freeman, 2019 – 800 с.

17. React и Redux. Функциональная веб-разработка - Алекс Бэнкс, Ева Порселло, 2018 – 336 с.

18. React быстро. Веб-приложения на React, JSX, Redux и GraphQL - Мардан А., 2019 – 560 с.

19. Learning Vue.js 2 – Olga Filipova ,2016 – 334 с.

Фрагменти коду проекту

```
147
148   }
149   .login:before {
150     content: '';
151     position: absolute;
152     top: -2px;
153     left: 0;
154     height: 2px;
155     width: 100%;
156   }
157   .password
158   {
159     width: 100%;
160     padding: 1em;
161     position: relative;
162     outline:none;
163     border: 1px solid #b12816;
164     box-shadow: 0px 7px 16px -3px rgba(0, 0, 0, 0.5);
165     border-radius: 25px;
166     margin-top: 5px;
167   }
168   .btn-primary {
169     display: inline-block;
170     text-decoration: none;
171     letter-spacing: var(--mainSpacing);
172     color: var(--mainBlack);
173     background: var(--primaryColor);
174     padding: 0.4rem 0.9rem;
175     border: 3px solid var(--primaryColor);
176     transition: var(--mainTransition);
177     text-transform: uppercase;
178     cursor: pointer;
```

```
import React from "react";
import { useContext } from "react";
import { RoomContext } from "../context";
import Title from "./Title";
// get all unique values
const getUnique = (items, value) => {
  return [...new Set(items.map(item => item[value]))];
};

const RoomsFilter = ({ rooms }) => {
  // react hooks
  const context = useContext(RoomContext);
  const {
    handleChange,
    type,
    capacity,
    price,
    minPrice,
    maxPrice,
    minSize,
    maxSize,
    breakfast,
    pets
  } = context;

  // get unique types
  let types = getUnique(rooms, "type");
  // add all
  types = ["all", ...types];
  // map to jsx
  types = types.map((item, index) => (
    <option key={index} value={item}>
```

```
    </option>
  ));
  // get unique capacity
  let people = getUnique(rooms, "capacity");
  people = people.map((item, index) => (
    <option key={index} value={item}>
      {item}
    </option>
  ));
  return (
    <section className="filter-container">
      <Title title="search rooms" />
      <form className="filter-form">
        {/* select type */}
        <div className="form-group">
          <label htmlFor="type">room type</label>
          <select
            name="type"
            id="type"
            onChange={handleChange}
            className="form-control"
            value={type}
          >
            {types}
          </select>
        </div>
        {/* end of select type */}
        {/* guests  */}
        <div className="form-group">
          <label htmlFor="capacity">Guests</label>
          <select
            name="capacity"
```

```
66                  id="capacity"
67                  onChange={handleChange}
68                  className="form-control"
69                  value={capacity}
70                >
71                  {people}
72                </select>
73              </div>
74              {/* end of guests */}
75              {/* room price */}
76              <div className="form-group">
77                <label htmlFor="price">room price ${price}</label>
78                <input
79                  type="range"
80                  name="price"
81                  min={minPrice}
82                  max={maxPrice}
83                  id="price"
84                  value={price}
85                  onChange={handleChange}
86                  className="form-control"
87                />
88              </div>
89              {/* end of room price*/}
90              {/* size */}
91              <div className="form-group">
92                <label htmlFor="price">room size </label>
93                <div className="size-inputs">
94                  <input
95                    type="number"
96                    name="minSize"
97                    value={minSize}
98                    onChange={handleChange}
```

```
                type="number"
                name="maxSize"
                value={maxSize}
                onChange={handleChange}
                className="size-input"
              />
            </div>
          </div>
          {/* end of select type */}
          {/* extras */}
          <div className="form-group">
            <div className="single-extra">
              <input
                type="checkbox"
                name="breakfast"
                id="breakfast"
                checked={breakfast}
                onChange={handleChange}
              />
              <label htmlFor="breakfast">breakfast</label>
            </div>
            <div className="single-extra">
              <input
                type="checkbox"
                name="pets"
                checked={pets}
                onChange={handleChange}
              />
              <label htmlFor="breakfast">pets</label>
            </div>
          </div>
          {/* end of extras type */}
```

```jsx
import React from "react";
import { Link } from "react-router-dom";
import Hero from "../components/Hero";
import Banner from "../components/Banner";
import Services from "../components/Services";
import FeaturedRooms from "../components/FeaturedRooms";
const home = () => {
  return (
    <>
      <Hero>
        <Banner
          title="luxurious rooms"
          subtitle="deluxe rooms starting at $299"
        >
          <Link to="/rooms" className="btn-primary">
            our rooms
          </Link>
        </Banner>
      </Hero>
      <Services />
      <FeaturedRooms />
    </>
  );
};

export default home;

```

```
import React, { Component } from "react";
import defaultBcg from "../images/room-1.jpeg";
import Hero from "../components/Hero";
import Banner from "../components/Banner";
import { Link } from "react-router-dom";
import { RoomContext } from "../context";

import StyledHero from "../components/StyledHero";
export default class SingleRoom extends Component {
  constructor(props) {
    super(props);
    console.log(this.props);
    this.state = {
      slug: this.props.match.params.slug,
      defaultBcg: defaultBcg
    };
  }
  static contextType = RoomContext;

  // componentDidMount() {
  //   console.log(this.props);
  // }
  render() {
    const { getRoom } = this.context;
    const room = getRoom(this.state.slug);

    if (!room) {
      return (
        <div className="error">
          <h3> no such room could be found...</h3>
```


Link to="/rooms" className="btn-primary"
back to rooms
/Link
/div
);
}
const {
name,
description,
capacity,
size,
price,
extras,
breakfast,
pets,
images
} = room;
const [main, ...defaultImages] = images;
console.log(defaultImages);
return (
<>
StyledHero img={images[0] || this.state.defaultBcg}
Banner title={`${name} room`}
Link to="/rooms" className="btn-primary"
back to rooms
/Link
/Banner
/StyledHero

```
62          <section className="single-room">
63            <div className="single-room-images">
64              {defaultImages.map((item, index) => (
65                <img key={index} src={item} alt={name} />
66              ))}
67            </div>
68            <div className="single-room-info">
69              <article className="desc">
70                <h3>details</h3>
71                <p>{description}</p>
72              </article>
73              <article className="info">
74                <h3>info</h3>
75                <h6>price : ${price}</h6>
76                <h6>size : {size} SQFT</h6>
77                <h6>
78                  max capacity :
79                  {capacity > 1 ? `${capacity} people` : `${capacity} person`}
80                </h6>
81                <h6>{pets ? "pets allowed" : "no pets allowed"}</h6>
82                <h6>{breakfast && "free breakfast included"}</h6>
83              </article>
84            </div>
85          </section>
86          <section className="room-extras">
87            <h6>extras </h6>
88            <ul className="extras">
89              {extras.map((item, index) => (
90                <li key={index}>- {item}</li>
91              ))}
92            </ul>
93            <div>
```