Міністерство освіти і науки України
Український державний університет науки і технологій

Комп'ютерних технологій і систем
(назва факультету)
Комп'ютерні інформаційні технології
(повна назва кафедри)

Пояснювальна записка
до кваліфікаційної роботи
Магістр
(ступінь вищої освіти)

на тему: Дослідження впливу React на фреймворки розробки Web-застосунків
за освітньою програмою 12 Інформаційні технології
зі спеціальності: 121 Інженерія програмного забезпечення

Виконав студент групи:
(підпис студента) Владислав ТРИПУТІН
(Ім'я ПРІЗВИЩЕ)

Керівник: (підпис) доц. Вадим АНДРЮЩЕНКО
(посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: (підпис) доц. Світлана ВОЛКОВА
(посада, Ім'я ПРІЗВИЩЕ)

Консультанти:
Техніко-економічні розрахунки (підпис) доц. Микола ГНЕНИЙ
(назва розділу) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент (підпис)

Дніпро – 2022 рік

Міністерство освіти і науки України
Український державний університет науки і технологій

Комп'ютерних технологій і систем
(назва факультету)
Комп'ютерні інформаційні технології
(повна назва кафедри)

Пояснювальна записка

до кваліфікаційної роботи
Магістр
(ступінь вищої освіти)

на тему: Дослідження впливу React на фреймворки розробки Web-застосунків
за освітньою програмою 12 Інформаційні технології
зі спеціальності: 121 Інженерія програмного забезпечення

Виконав студент групи:

(підпис студента)	Владислав ТРИПУТІН (Ім'я ПРІЗВИЩЕ)
Керівник: (підпис)	доц. Вадим АНДРЮЩЕНКО (посада, Ім'я ПРІЗВИЩЕ)
Нормоконтролер: (підпис)	доц. Світлана ВОЛКОВА (посада, Ім'я ПРІЗВИЩЕ)

Консультанти:

(назва розділу)	(підпис)	(посада, Ім'я ПРІЗВИЩЕ)
(назва розділу)	(підпис)	(посада, Ім'я ПРІЗВИЩЕ)
(назва розділу)	(підпис)	(посада, Ім'я ПРІЗВИЩЕ)
(назва розділу)	(підпис)	(посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Computer technologies and systems
(faculty)
Computer information technology
(department)

Explanatory Note

to Master's Thesis
Master
(higher education degree)

on the topic: Researching the impact of React on Web application development frameworks
according to educational curriculum 12 Information technologies
in the Speciality: 121 Software engineering

Done by the student of the group:

(name, surname)

Scientific Supervisor:

(position, name, surname)

Normative controller :

(position, name, surname)

Supervisors

(Chapter title heading)	(position, name, surname)
(Chapter title heading)	(position, name, surname)
(Chapter title heading)	(position, name, surname) (по
(Chapter title heading)	(position, name, surname) (по

Dnipro – 2022

Український державний університет науки і технологій

Факультет Комп'ютерних технологій і систем кафедра Комп’ютерні інформаційні технології
Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____ доц. Вадим ГОРЯЧКІН
(підпис)
«___»_____2022 р.

ЗАВДАННЯ

до дипломної роботи на здобуття ОС___магістр___
(освітній ступінь)
студента групи (ПЗ2121) 961-М Трипутіна Владислава Сергійовича
(номер групи) (ПІБ)

1 Тема дипломної роботи: _Дослідження впливу React на фреймворки розробки Web-застосунків_____
затверджена наказом по університету від «02» листопада 2022 р. № 1113.

2 Термін подання студентом закінченої роботи_____

3 Вихідні дані до дипломної роботи _____
_____
_____
_____

4 Зміст пояснювальної записки (перелік питань до розробки) складається з вступу, аналізу розвитку технологій екосистеми React, екосистемного підходу React, розробки і тестування аналітичного додатку, дослідження ефективності різних фреймворків, техніко-економічних розрахунків, загальних висновків та бібліографічного списку._____

5 Перелік демонстраційного матеріалу: вступ, історія виникнення React, причини популярності React, переваги та недоліки React, конкуренти React, концептуальна основа, проведення експерименту, результати експерименту.

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Техніко-економічні розрахунки	Доцент Гненний М.В.		

КАЛЕНДАРНИЙ ПЛАН

№ пор.	Назва розділів дипломної роботи	Термін виконання розділів роботи	Примітка
1	Вступ		
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами		від 70 джерел
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження		
4	Постановка задачі, технічне завдання		30%
5	Техніко-економічні показники		
6	Розробка інструментальних засобів дослідження		
7	Виконання досліджень		60%
8	Оформлення тез доповідей		
9	Оформлення статті у фаховий журнал		
10	Оформлення пояснювальної записки		
11	Розробка демонстраційних матеріалів		100%

Дата видачі завдання «02» листопада 2022 р.

Керівник дипломної роботи _____ _____
(підпис) (ПІБ)

Завдання прийняв до виконання _____ _____
(підпис) (ПІБ)

РЕФЕРАТ

Об’єктом цього дослідження є фреймворки для створення інформаційних сайтів.

Предметом дослідження є процес вибору фреймворка при створенні інформаційних сайтів.

Метою роботи є аналіз можливостей фреймворків для створення інформацінйих сайтів.

Методи дослідження: порівняння обраних фреймворків, починаючи від їх популярності використання та закінчуючи порівняння їх швидкодії у різних ситуаціях.

Результати та їх новизна: дослідження робить внесок у визначенні практичної застосовності фреймворків. Результати дослідження дозволяють зробити висновки щодо того в яких ситуаціях той чи інший фреймворк показує кращі результати.

Пояснювальна записка складається зі вступу, 5 розділів, висновків, бібліографічного списку та 5 додатків:

– у вступі описується сутність розробки, її актуальність. Складається із 2 сторінок;

– у першому розділі висвітлюються ключові аспекти React як фреймворку, пояснюються проблеми, які він вирішує, і описується, як можна використовувати функції для того щоб покращити практику веб-розробки та створити складні, але придатні для обслуговування інтерфейси користувача за допомогою React. Обгрунтовується вибір фреймворків для аналізу. Складається з 10 сторінок;

– у другому розділі надано аналіз того чому React такий популярний та проведено попередній аналіз фреймворків. Складається з 16 сторінок;

– у третьому розділі представлене проектування й розробка аналітичного додатку для дослідження. Складається з 11 сторінок;

– у четвертому розділі описано виконані дослідження. Складається з 10 сторінки;

– у п‘ятому розділі розкриті питання техніко-економічних розрахунків. Складається з 9 сторінок;

– додатки містять технічне завдання й робочий проект.

Таблиць – 5, рисунків – 8, бібліографія – 62 джерел.

Ключові слова: фреймворк, аналіз, SPA, React, Angular, Vue.

ЗМІСТ

ВСТУП

Актуальність теми. Людство дедалі більше залежить від інформації. Тому важливо надати її якісно, в актуальному стані, у потрібній кількості та якнайшвидше. У цьому сенс усіх сучасних програм. До описаних вище завдань можна додати інтеграцію продуктів із мережею Інтернет, яка давно стала невід'ємною частиною життя кожної людини. Однією з найпопулярніших та найефективніших мов програмування для таких цілей є JavaScript, який дозволяє створювати інтерактивні, зручні та якісні програми.

В основному, проблеми, що виникають при розробці таких продуктів, пов'язані з великим розміром коду, часом, витраченим для написання і тестування, а також його адаптованістю під різні платформи. У цьому сенсі використання бібліотек JavaScript дозволяє вирішувати ці проблеми. Які перетворюють цю мову на універсальний інструмент, налаштований на виконання певних завдань.

Вибір необхідної бібліотеки та її ефективне використання означає вирішення завдання максимально швидко та ефективно з мінімальними витратами коштів. Що доводить актуальність цього питання. Вебфреймворки допомагають досягти структури в програмах, і вони дають додаткові функції, які можна додати до них без зайвої роботи. Фреймворк дає можливість почати, щоб існувала можливість зосередитися на функціях, а не на деталях конфігурації. На сьогоднішній день існує багато фреймворків, тож іноді буває складно обрати, з якого саме почати. Для певної потреби існує свій фреймворк. Перш ніж обирати потрібний фреймворк спочатку потрібно дізнатися про важливі елементи, які слід враховувати.

Тому обґрунтованою є тема магістерської роботи, у якій вирішується **науково-прикладне завдання** дослідження екосистеми React та порівняння його з іншими фреймворками.

Об'єкт дослідження – фреймворки для створення інформаційних сайтів.

Предмет дослідження – процес вибору фреймворка при створенні інформаційних сайтів.

Мета і завдання дослідження. Метою дослідження є аналіз можливостей фреймворків для створення інформацінйих сайтів.

Для досягнення мети дослідження необхідно вирішити такі **завдання**:

Для виконання основної задачі роботи потрібно виконати такі пункти:

1) провести аналіз предметної області;

2) обґрунтувати вибір фреймворків;

3) визначити критерії порівняння;

4) ознайомитись з обраними інструментами розробки;

5) реалізувати необхідний функціонал для трьох застосунків;

6) провести порівняння за заданими критеріями.

Особистий внесок здобувача полягає у аналізі літератури з теми дослідження, проведенні аналізу існуючих фреймворків, розгляді відомих фреймворків. В якості експериментальної частини атестаційної роботи було розроблено сторінку інформаційного сайту.

Структура та обсяг магістерської роботи. Кваліфікаційна магістерська робота складається із вступу, п’яти розділів, висновків, переліку посилань. Загальний обсяг складає 112 сторінок, з яких основний текст на 66 сторінках, список використаних джерел із 62 найменувань на 4 сторінках. Робота містить 5 таблиць, 8 рисунків.

РОЗДІЛ 1

АНАЛІЗ РОЗВИТКУ ТЕХНОЛОГІЙ ЕКОСИСТЕМИ REACT

У цьому розділі висвітлюються ключові аспекти React як фреймворку, пояснюються проблеми, які він вирішує, і описується, як можна використовувати функції для того щоб покращити практику веб-розробки та створити складні, але придатні для обслуговування інтерфейси користувача за допомогою React.

1.1 Історія виникнення React

Ще в 2011 році розробники Facebook почали стикатися з деякими проблемами з обслуговуванням коду. Оскільки додаток Facebook Ads отримував дедалі більше функцій, команді потрібно було більше людей, щоб він працював бездоганно. Зростаюча кількість членів команди та функцій додатків уповільнили їх як компанію. З часом їхню програму стало важко використовувати, оскільки вони стикалися з великою кількістю каскадних оновлень. Через деякий час інженери Facebook не могли встигати за цими каскадними оновленнями. Їх код вимагав термінового оновлення, щоб стати більш ефективним. У них була правильна модель, але їм потрібно було щось зробити щодо взаємодії з користувачем.

React[5] — це платформа JavaScript. Спочатку React був створений інженерами Facebook для вирішення проблем, пов’язаних із розробкою складних інтерфейсів користувача з наборами даних, які змінюються з часом. Це не є тривіальним заходом і має бути не тільки доступним для обслуговування, але й масштабованим для роботи в масштабі Facebook. React фактично народився в рекламній організації Facebook, де вони використовували традиційний клієнтський підхід Model-View-Controller. Подібні додатки зазвичай складаються з двостороннього зв’язування даних разом із шаблоном візуалізації. React змінив спосіб створення цих програм, зробивши деякі сміливі досягнення у веб-розробці.

React кидає виклик конвенціям, які стали фактичними стандартами для найкращих практик JavaScript фреймворків. React робить це, вводячи багато нових парадигм і змінюючи статус-кво того, що потрібно для створення масштабованих і підтримуваних програм JavaScript та інтерфейсів користувача. Разом зі зміною уявлення про інтерфейсну розробку React має багатий набір функцій, які роблять створення односторінкової програми або інтерфейсу користувача доступним для розробників різного рівня кваліфікації — від тих, хто тільки познайомився з JavaScript, до досвідчених ветерани мережі.

2010 – Перші ознаки React

- Facebook[23] представив xhp у свій стек php і відкрив вихідний код.
- Xhp дозволяв створювати композитні компоненти. Вони представили цей синтаксис пізніше в React.

2011 – ранній прототип React

- Джордан Волке створив FaxJS, ранній прототип React – додав пошуковий елемент у Facebook.

2012 – у Facebook почалося щось нове

- Оголошеннями у Facebook стало важко керувати, тому Facebook потрібно було придумати хороше рішення для цього. Джордан Волке працював над прототипом і створив React.
- 9 квітня: Instagram був придбаний Facebook.
- Instagram хотів прийняти нову технологію Facebook. Через це Facebook чинив тиск, щоб відокремити React від Facebook і зробити його відкритим. Більшу частину цього зробив Піт Хант.
- 8-12 вересня: TechCrunch підриває Сан-Франциско, Марк Цукерберг: «Нашою найбільшою помилкою було занадто багато ставок на HTML5». Він пообіцяв, що Facebook незабаром забезпечить кращий мобільний досвід.

2013 – рік великого старту

- 29-31 травня: JS ConfUS. Джордан Волке представив React. React отримує відкритий код. Цікавий факт: публіка була налаштована скептично. Більшість людей вважали React величезним кроком назад. Це сталося, оскільки на цю конференцію прийшли здебільшого «перші користувачі», однак React націлився на «новаторів». Творці React вчасно усвідомили цю помилку та вирішили розпочати «тур React» пізніше, щоб перетворити ненависників на прихильників.
- червня: React (від Facebook) доступний на JSFiddle
- 30 липня: React і JSX доступні в Ruby on Rails
- 19 серпня: React і JSX доступні в програмах Python
- 14-15 вересня: JSConfEU 2013. Промова Піта Ханта про переосмислення найкращих практик.
- 17 грудня: Девід Нолен представляє OM на основі React. Пояснює, наскільки чудовий React, який досяг перших користувачів. Ця стаття показала, наскільки React кращий за інші альтернативи, що підвищило визнання React.

2014 – рік розширення

- React поступово завоював свою репутацію та почав охоплювати «першу більшість» потенційних користувачів. На цьому етапі їм знадобилося нове повідомлення замість того, щоб покладатися виключно на його технічні переваги, і воно: наскільки React стабільний? Зосередившись на цьому, вони прагнули звернути увагу на такі підприємства, як Netflix.
- Початок 2014 року: стартували конференції #reactjsworldtour, щоб побудувати спільноту та «перетворити ненависників на захисників».
- січня: React Developer Tools стає розширенням Chrome Developer Tools.
- Лютий: представлено Atom – текстовий редактор 21 століття, який можна зламати
- 7-9 квітня: React London 2014

- Червень: з’явився ReactiveX.io.
- 13 липня: Випуск React Hot Loader. React Hot Loader — це плагін, який дозволяє живо перезавантажувати компоненти React без втрати стану.
- 12 грудня: PlanOut: мова для онлайн-експериментів. Випуск PlanOut 0.5, який включає мовний редактор PlanOut на основі React, і забезпечує паритет функцій інтерпретатора з останньою версією PlanOut, яка використовується внутрішньо у Facebook.

2015 – React стабільний

- Початок 2015: Flipboard випускає 28-29 січня: React.js Conf 2015 – Facebook випустив першу версію React Native для React.js Conf 2015 під час технічної розмови.
- Лютий: представлення Relay і GraphQL на React.js Conf.
- 25 березня: Facebook оголосив, що React Native для iOS відкрито та доступно на GitHub.
- червня: Dan Abramov і Andrew Clark випустили Redux.
- вересня: запущено першу стабільну версію нових інструментів розробника React.
- 14 вересня: випущено React Native для Android.
- React Canvas.
- Січень: Netflix подобається React
- Початок 2015: Airbnb використовує React

2016 – React стає мейнстрімом

- Березень: Представлення Mobx
- 22-23 лютого: React.js Conf 2016, Сан-Франциско
- Draft.js був представлений на React.js Conf Ісааком Сальє-Хеллендагом
- Березень: презентація React Storybook
- 2-3 червня: ReactEurope 2016

- 11 липня: представлення системи кодів помилок React.
- Листопад: презентація Blueprint – інструментарію React UI для Інтернету

2017 рік – рік подальших удосконалень

- Початок 2017 року: Airbnb представляє свою нову бібліотеку з відкритим кодом React Sketch.app
- 19 квітня: вихідний код React Fiber відкривається на F8 2017.
- Вересень: переліцензування React, Jest, Flow і Immutable.js
- 26 вересня: React 16: межі помилок, портали, фрагменти та архітектура Fiber
- Жовтень: Netflix видаляє клієнтський React.js
- 28 листопада: React v16.2.0: покращена підтримка фрагментів

2018 – Що зараз відбувається з React?

- 1-2 березня: JSConf Iceland – Beyond React 16 від Дена Абрамова
- 29 березня 2018: випущено React 16.3.0.

2019 – Нові React DevTools

- У 2019 році було запущено нову React DevTools.

2020 – Компоненти сервера React з нульовим розміром комплекту

- У 2020 році вони представили серверні компоненти React з нульовим розміром пакета для початкового відгуку від спільноти розробників React.

1.2 Розвиток архітектури React

Як уже зазначалося, React — це інша концепція, коли мова заходить про веб-розробку загалом. Це відхід від загальноприйнятих робочих процесів і найкращих практик. Чому Facebook ухилився від цих тенденцій на користь створення абсолютно нового бачення процесу веб-розробки? Чи було це просто надзвичайно недбало кинути виклик загальноприйнятим найкращим

практикам, чи для створення React існувало узагальнене ділове обґрунтування?

Якщо подивитися на аргументацію React, то можна побачити, що його було створено, щоб задовольнити певну потребу в певному наборі технологічних проблем, з якими стикається Facebook. Ці виклики були і не є унікальними для Facebook, але те, що Facebook зробив, це вирішував проблеми безпосередньо за допомогою підходу до вирішення проблеми самостійно. Це можна розглядати як аналог філософії Unix, викладеної Еріком Реймондом у його книзі «Мистецтво програмування Unix». У книзі Реймонд пише про правило модульності, яке звучить так: «Єдиний спосіб написати складне програмне забезпечення, яке не впаде нанівець, — це стримувати його глобальну складність, створюючи його з простих частин, з'єднаних чітко визначеними інтерфейси, щоб більшість проблем були локальними, і ви мали надію оновити частину, не порушуючи цілого».

Це саме той підхід, який використовує React у вирішенні проблем складних інтерфейсів користувача. Facebook, розробляючи React, не створив повну архітектуру Model-ViewController, щоб замінити існуючі фреймворки. Не було потреби винаходити це конкретне колесо та ускладнювати проблему створення великомасштабних інтерфейсів користувача.

React був створений для вирішення окремої проблеми. React було створено для роботи з відображенням даних в інтерфейсі користувача. Можна подумати, що відображення даних в інтерфейсі користувача — це проблема, яку вже вирішено, і це правильно. Різниця полягає в тому, що React був створений для обслуговування великомасштабних інтерфейсів користувача — інтерфейсів масштабу Facebook та Instagram — з даними, які змінюються з часом. Цей вид інтерфейсу можна створити та вирішити за допомогою інструментів, які існують поза React. Фактично, Facebook повинен був вирішити ці проблеми до того, як створив React. Але Facebook таки створив React, оскільки він мав вагомі аргументи та виявив, що React можна

використовувати для вирішення конкретних проблем, які виникають під час створення складних інтерфейсів користувача.

React не має на меті вирішити всі проблеми, з якими можна зіштовхнутися в дизайні інтерфейсу користувача та розробці інтерфейсу. React вирішує певний набір проблем і загалом одну проблему. Як заявили Facebook і Instagram, React створює масштабні користувацькі інтерфейси з даними, які змінюються з часом.

Широкомасштабні користувальницькі інтерфейси з даними, які змінюються з часом, ймовірно, можуть бути чимось, до чого багато веб-розробників можуть мати відношення у своїй власній роботі чи хобі програмування. У сучасному світі веб-розробки часто перекладається велика частина відповідальності за інтерфейс користувача на браузер і HTML, CSS і JavaScript. Ці типи програм зазвичай називають односторінковими програмами, де загальний запит/відповідь на сервер обмежений, щоб продемонструвати потужність браузера.

Проблема виникає, коли код проекту «вихідного дня» більше не підтримується. В цьому разі необхідно «закрутити» додаткові фрагменти коду, щоб дані зв’язувалися належним чином. Іноді доводиться реструктуризувати програму, тому що вторинна бізнес-вимога випадково порушила спосіб обробки інтерфейсом кількох взаємодій після того, як користувач розпочав завдання. Усе це призводить до того, що користувальницькі інтерфейси є крихкими, сильно взаємопов’язаними та нелегкими для обслуговування. Це всі проблеми, які React намагається вирішити.

Розглянемо, наприклад, архітектуру Model-View-Controller на стороні клієнта з двостороннім зв’язуванням даних у шаблонах. Ця програма має містити подання, які слухають моделі, а потім подання незалежно оновлюють свою презентацію на основі взаємодії користувача або зміни моделі. У базовій програмі це не є помітним вузьким місцем для продуктивності або, що більш важливо, для продуктивності розробника. Масштаб цієї програми неминуче

зростатиме, оскільки до неї додаватимуться нові моделі та види. Усе це пов’язано через делікатний і заплутаний безлад коду, який може керувати зв’язками кожного з представлень та їхніх моделей. Це швидко стає дедалі складнішим. Елементи, які знаходяться глибоко в ланцюжку візуалізації або у віддаленій моделі, тепер впливають на вихід інших елементів. У багатьох випадках розробник може навіть не знати про оновлення, оскільки підтримувати механізм відстеження стає дедалі складніше. Це ускладнює розробку та тестування вашого коду, а це означає, що стає важче розробити метод або нову функцію та випустити їх. Код тепер менш передбачуваний, а час розробки різко зріс. Це саме та проблема, яку React збирається вирішити.

Спочатку React був уявним експериментом. Facebook думав, що вони вже написали початковий код макета, щоб описати, як може і має виглядати програма, то чому б просто не запустити код запуску знову, коли дані або стан змінюють програму? Ймовірно, ви зараз засмучуєтеся, тому що знаєте, що це означає, що вони жертвуватимуть продуктивністю та користувальницьким досвідом. Коли ви повністю замінюєте код у браузері, ви побачите мерехтіння екрана та спалахи нестилізованого вмісту. Це просто буде здаватися неефективним. Facebook це знав, але також зауважив, що те, що він створив — механізм для заміни стану при зміні даних — насправді певною мірою працює. Тоді Facebook вирішив, що якщо механізм заміни можна оптимізувати, він матиме рішення. Так народився React як рішення певного набору проблем.

Багато хто стверджує, що React — це повномасштабний фреймворк JavaScript на рівні, який можна порівняти з іншими фреймворками, такими як Backbone, Knockout.js, AngularJS, Ember, CanJS, Dojo або будь-яким із численних існуючих фреймворків MVC. На рисунку 1.1 показаний приклад типової структури MVC.

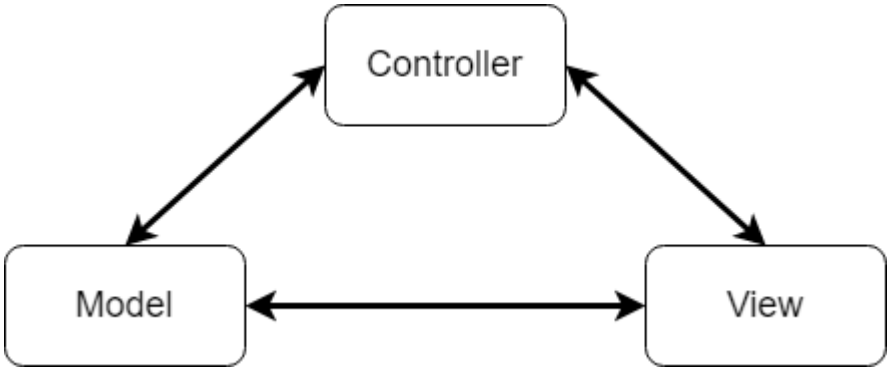

Рисунок 1.1 - Базова архітектура MVC

На рис.1.1 показані основи кожного з компонентів архітектури Model-View-Controller. Модель обробляє стан програми та надсилає події, що змінюють стан, до представлення. Представлення — це зовнішній вигляд користувача та інтерфейс взаємодії з кінцевим користувачем. Представлення може надсилати події до контролера, а в деяких випадках і до моделі. Контролер є головним диспетчером подій, які можна надсилати в модель, щоб оновити стан, і в представлення, щоб оновити презентацію. Ви можете зауважити, що це загальне представлення того, що таке архітектура MVC, і насправді існує так багато варіантів і налаштованих реалізацій, що єдиної архітектури MVC не існує. Суть полягає не в тому, щоб стверджувати, як виглядає структура MVC, а в тому, щоб вказати, чим React не є.

Ця структура MVC насправді не є справедливою оцінкою того, чим React є або має намір бути. Це тому, що React є окремою частиною того, що представляють ці фреймворки. React — це найпростіша форма, лише перегляд цих фреймворків MVC, MVVM або MV*. React — це спосіб описати користувальницький інтерфейс програми та механізм його зміни з часом, коли дані змінюються. React складається з декларативних компонентів, які описують інтерфейс. React не використовує прив’язки спостережуваних даних під час створення програми. React також легко маніпулювати, тому що можна взяти компоненти, які створено, і поєднати їх, щоб створити власні компоненти, які щоразу працюватимуть так, як очікується, оскільки він може

масштабуватися. React може масштабуватися краще, ніж інші фреймворки, завдяки принципам, які керували ним з моменту створення. Створюючи інтерфейси React, вони структуруються таким чином, щоб вони складалися з кількох компонентів.

1.3 Обгрунтування обраних фреймворків

Для даного дослідження фреймворки були обрані за критеріями популярності, такими як:

а. Опубліковані рейтинги

Веб-ресурс State of JS 2021 опублікували рейтинг дев'яти найбільш використовуваних фреймворків, який ранжується за задоволеністю користувачів, інтересом до інструменту, відсотком використання та обізнаністю. За останніми двома критеріями впродовж останніх чотирьох років трійка Angular[1], React та Vue.js тримається на вершині списку, що свідчить про надійність, наявність неспрадного інтересу серед компаній, а також гарантує велику кількість документації та готових рішень.

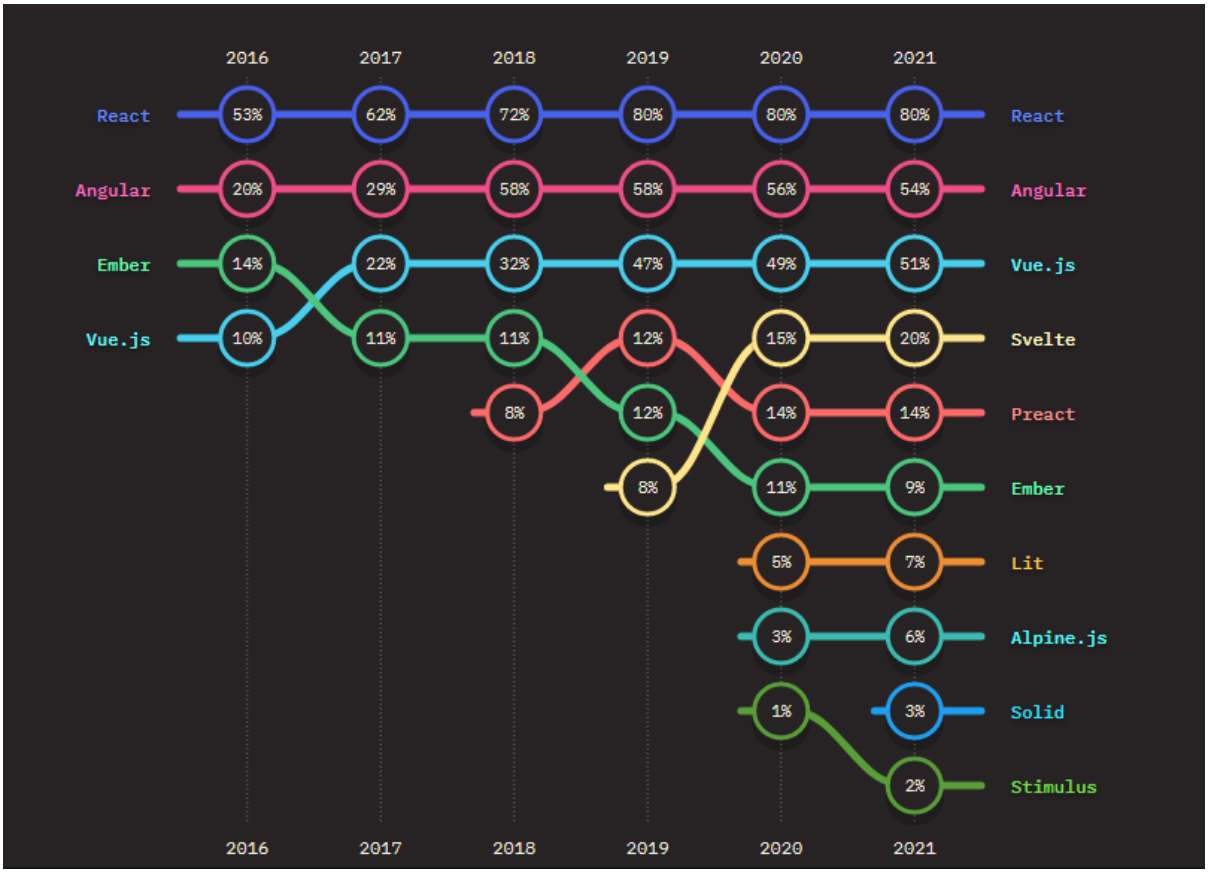

Рисунок 1.2 - Рейтинг State of JS 2021

Наступний показовий рейтинг – 2022 Developer Survey від компанії Stackoverflow(рисунок 1.3), який свідчить про зростаючу зацікавленість до обраних фреймворків. Джерело дає коментар про те, Node.js і React.js — це дві найпоширеніші веб-технології, які використовуються професійними розробниками та тими, хто вчиться програмувати. Angular частіше використовують професійні розробники, ніж ті, хто вчиться програмувати (23% проти 10%), Vue.js використовується професійними розробниками та тими, хто вчиться програмувати у співвідношенні (19.9% проти 13.48 %). Можна зробити заключення про те, що понад 42% респондентів використовує одну з цих технологій.

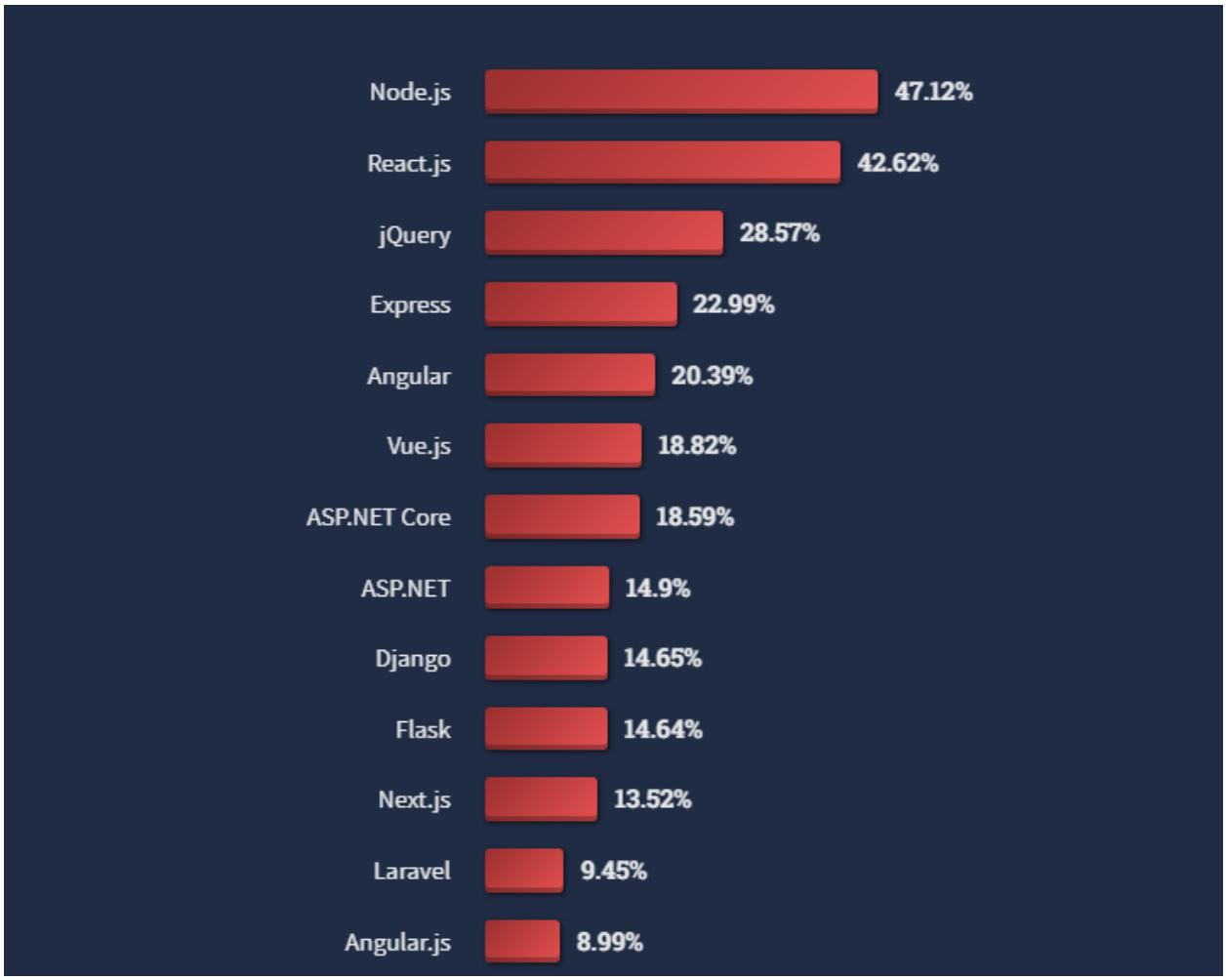

Рисунок 1.3 - Рейтинг 2022 Developer Survey

Компанія JetBrains також проводить дослідження серед користувачів їхніх програмних продуктів. Вони збирають дані по десятці мов програмування, серед яких є JavaScript [3] та опитування щодо веб-фреймворків (рис. 1.4). У ньому з великим відривом лідує React, а Vue.js та Angular йдуть наступними.

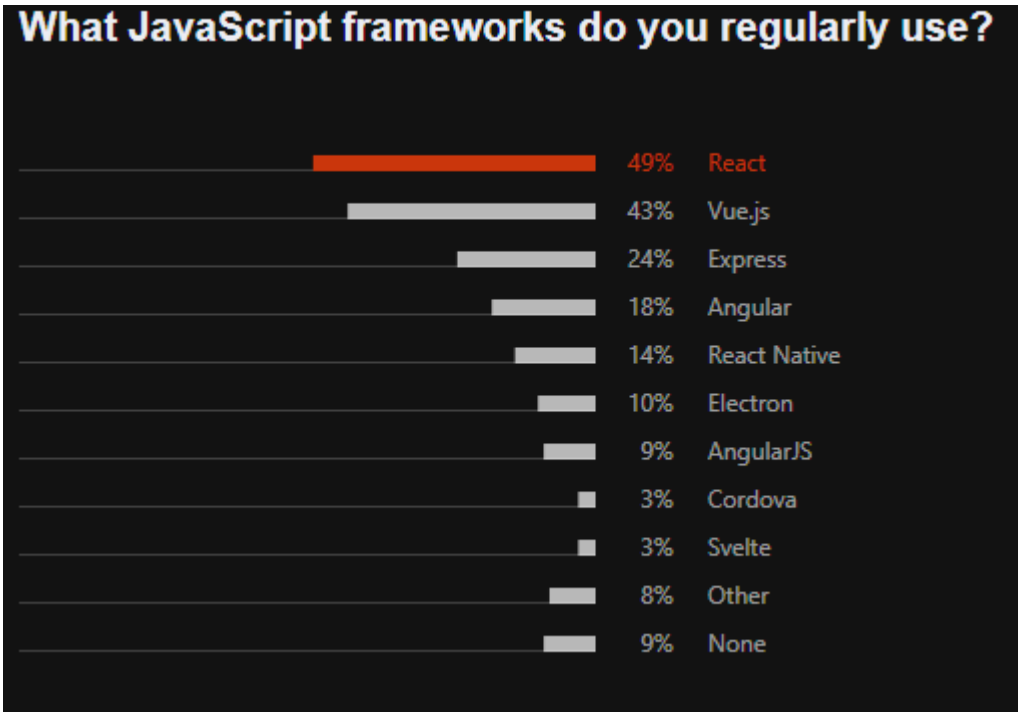

Рисунок 1.4 - Опитування Jetbrains

b. Кількість вакансій на ринку

Було проаналізовано чотири найбільших веб-ресурсів з вакансіями у сфері IT в Україні. У таблиці 1.1 наведені цифри – приблизна кількість об’яв, які містять у назві чи описі назви одного з трьох фреймворків. На першому місці знаходиться React, далі – Angular. А останнє місце Vue пояснюється тим, що даний інструмент відносно молодий у порівнянні з іншими.

c. Відсоток пошуку

Для аналізу кількості запитів було використано сервіс Google Trends. React домінує у обсягах пошуку та має 57,5%, при цьому Angular збирає значну частку в 31,5%, а Vue.js набирає 11%.

1.4 Постановка задачі

Для виконання основної задачі роботи потрібно виконати такі пункти:

1) провести аналіз предметної області;

2) обґрунтувати вибір фреймворків;

3) визначити критерії порівняння;

4) ознайомитись з обраними інструментами розробки;

5) реалізувати необхідний функціонал для трьох застосунків;

6) провести порівняння за заданими критеріями.

Порівняльна характеристика має проводитись за такими критеріями:

1) історія та головні принципи роботи фреймворків;

2) складність освоєння та подальшого глибшого вивчення;

3) розробка в контексті малих та багатосторінкових застосунків;

4) розширення та односторінкові застосунки;

5) оптимізаційні можливості;

6) легкість управління станом;

7) продуктивність;

8) популярність та вакансії.

Розроблені системи мають володіти такими функціональними можливостями:

1) введення та виведення даних;

2) робота з Web-API;

3) відображення даних;

4) додавання та видалення елементів на сторінці.

1.5 Аналіз предметної області

1.5.1Опис проблеми. Актуальність дослідження

Світ JavaScript – це середовище, що включає широкий вибір інструментів для розробки, бібліотек та фреймворків. Проте з великою кількістю варіантів виникає багато плутанини, особливо для новачків.

На сьогодні існує близько тридцяти JavaScript-фреймворків, що орієнтовані на розробку веб-застосунків. Вони є важливою частиною сучасної

фронтенд-розробки та забезпечують програмістів надійними інструментами для створення масштабованих інтерактивних додатків. Також все більше компаній використовують їх для своїх проектів, а отже знання найпопулярніших фреймворків стає вимогою для багатьох вакансій.

Популярність розробників інтерфейсів користувача не спадає, а мова JavaScript вже 8 років знаходиться на першому місці серед запитів на платформі Stackoverflow, що приваблює початківців у сфері IT, а значить питання вибору найзручнішого інструменту виникає постійно та робить дану тему дослідження актуальною.

Висновки до розділу 1

У цьому розділі представлено концепції, які дозволили Facebook створити React. Ви дізналися, як концепції React зазвичай розглядаються як такі, що відходять від загальноприйнятих найкращих практик розробки інтерфейсу користувача. Виклик статус-кво та теорії тестування дозволили React стати високопродуктивним і масштабованим фреймворком JavaScript для створення інтерфейсів користувача.

Рішення «за» та «проти» певної технології значною мірою залежать від варіанту використання та інших різноманітних обставин: розміри проектів, знання працівників, попередній досвід і подібні терміни впливають на цей процес. Angular, React і Vue не стоять далеко один від одного. У той час як перші два відрізняються переважно мовою розробки та філософією поділу файлів, вони схожі на велику технологічну компанію, яка спонсорує їхню розробку, що забезпечує певний рівень довіри для бази користувачів або тих, хто планує їх використовувати. Vue у цьому відношенні є повною протилежністю, оскільки він розроблений переважно з урахуванням інтересів спільноти. Крім того, він узяв деякі з найкращих функцій із уже існуючих фреймворків і зумів створити щось нове, що мало величезне зростання популярності у 2017 році та продовжується. Щоб узагальнити кілька

відповідних випадків використання та їхні відповідні рекомендації щодо вибору фреймворку, перегляньте наступну колекцію:

- Висока оцінка TypeScript ⇒ Angular
- Підкреслення вказівок і структури в проектах ⇒ Angular
- Виходець із фону об’єктно-орієнтованого програмування ⇒ Angular
- Висока важливість гнучкості ⇒ React або Vue
- Великий масштаб застосування ⇒ Всі три
- Неглибокий початковий процес навчання ⇒ Vue
- Акцент на використання найновіших, найпопулярніших технологій ⇒ Vue
- Велика екосистема ⇒ React
- Розділення проблем в одному файлі ⇒ Vue
- Дизайнери, необхідні для роботи з HTML-кодом ⇒ Angular або Vue
- Сильна увага до використання JavaScript ⇒ React

У світі JavaScript, де регулярно публікуються нові фреймворки, може бути нерозумним довго чекати з впровадженням нової технології, оскільки на той час вона може бути застарілою. Хоча це, звісно, значною мірою залежить від розміру компанії: тоді як менші команди розробників можуть швидше протестувати та прийняти нову структуру, більшим компаніям потрібно більше часу для оцінки, оскільки неправильне рішення може призвести до фінансових проблем заднім числом.

РОЗДІЛ 2

ЕКОСИСТЕМНИЙ ПІДХІД REACT

2.1 Ефект популярності React

Хоча інші технології, такі як Angular, були доступні, коли Facebook розробляв ReactJS, більшість розробників були змушені багато програмувати. Розробники, які використовують інші фреймворки, стикаються з проблемою переробки більшості кодів, навіть коли створюють компоненти, які часто змінюються. Їм потрібна була структура, яка могла б дозволити їм розбивати складні компоненти та повторно використовувати коди для швидшого завершення своїх проектів.

ReactJS надав рішення, яке шукали розробники. Він використовує JSX (унікальний синтаксис, який дозволяє використовувати лапки HTML, а також синтаксис тегів HTML для візуалізації певних підкомпонентів). Це дуже корисно для сприяння побудові машинозчитуваних кодів і водночас поєднання компонентів у файл, який можна перевірити одноразово.

Сьогодні ReactJS став дуже популярним завдяки своїй надзвичайної простоті та гнучкості. Багато людей навіть називають це майбутнім веб-розробки.

Частково ця величезна популярність пояснюється тим фактом, що провідні корпорації, такі як Facebook, PayPal, Uber, Instagram і Airbnb, використовують його для вирішення проблем, пов’язаних з інтерфейсом користувача. Ця довіра привернула багато людей до фреймворку.

Існує кілька причин, чому React настільки швидко став популярним:

- Робота з DOM API важка. React дає розробникам можливість працювати з віртуальним браузером, який дружелюбніший за реальний. Віртуальний браузер React - свого роду посередник між розробником та реальним браузером.

•React дозволяє розробникам декларативно описувати їх інтерфейси і модель стану (далі — state) цих інтерфейсів. Це означає, що замість опису кожної операції з інтерфейсами розробникам необхідно лише описати ці інтерфейси з позиції кінцевого state (як функція). Коли в state відбуваються зміни, React, спираючись на ці зміни, оновлює відповідні інтерфейси користувача.

•React — це лише JavaScript, в ньому дуже мало API для вивчення, лише деякі функції та способи їх використання. Тому ваші навички в JavaScript-те, що робить вас краще як React розробника. Жодних бар'єрів для входження. JavaScript розробник може стати хорошим розробником React за кілька годин.

Але це далеко не весь перелік. Спробуємо зрозуміти всі причини такого стрімкого зростання популярності React. Одна з них – Virtual DOM (алгоритм порівняльного аналізу). Ми будемо працювати з прикладом, щоб продемонструвати практичну користь такого алгоритму в нашому розпорядженні.

React офіційно подається як JavaScript бібліотека для побудови інтерфейсів користувача. Важливо підкреслити дві частини цього опису:

React це JavaScript бібліотека. Це не фреймворк. Це не комплексне рішення, нам часто потрібні інші бібліотеки для виконання поставлених завдань. Так ось React не знає нічого про способи вирішення цих завдань. Він сфокусований лише на одній, але виконуючи її чудово.

Те, з чим React справляється чудово, і є друга частина опису: побудова інтерфейсів користувача. Інтерфейс користувача - це те, що ми надаємо користувачам для їх взаємодії з машинною системою. Інтерфейси користувача знаходяться скрізь: від звичайної кнопки на мікрохвильовій печі до приладової дошки космічного корабля. Якщо пристрій може зрозуміти JavaScript, ми можемо використовувати React для опису на ньому інтерфейсу користувача.

З того моменту, як веб-браузери почали розуміти JavaScript, ми можемо використовувати React, щоб описувати веб-інтерфейси (Web User Interfaces). Мені подобається використовувати слово описувати цьому контексті, т.к. це саме те, що ми дійсно робимо за допомогою React: ми просто вказуємо те, що хочемо, а React побудує потрібний інтерфейс користувача в браузері. Без React або схожої бібліотеки ми були б змушені вручну будувати інтерфейси користувача, використовуючи нативні Web APIs і JavaScript.

Словосполучення "декларативний React" саме це і означає: ми описуємо інтерфейси користувача в React, передаючи йому наші побажання (а не спосіб їх виконання). React сам потурбується про те, як це зробити і перекласти наші декларативні описи (які ми пишемо на React) в актуальні інтерфейси користувача в браузері. React поділяє цю просту декларативну силу з HTML, але саме з React ми отримуємо декларативність HTML-інтерфейсів, яка надає як статичні, а й динамічні дані.

React має три основні концепції проектування, які зробили його популярним:

1 — Використання багаторазових, складових і компонентів, що мають state

У React ми описуємо інтерфейси користувача за допомогою компонентів. Ви можете думати про компоненти як про прості функції (у будь-якій мові програмування). Ми викликаємо функції з певними аргументами, після чого вони повертають нам результат. Ми можемо перевикористовувати функції на наш розсуд і утворювати більші функції з менших функцій.

Те саме з компонентами; ми називаємо їх аргументи "properties" і "state", а результат компонента - опис інтерфейсу користувача (який схожий на HTML для браузера). Ми багато разів можемо використовувати компонент у багатьох інтерфейсах користувача, а компоненти можуть містити інші компоненти.

Однак, на відміну від чистих функцій, компонент React може мати власний state для зберігання даних, які можуть змінюватися протягом часу.

2 - Суть реактивних оновлень

Сама назва бібліотеки React пояснює цю концепцію. Коли state компонента (вхідні дані) змінюється, інтерфейс користувача змінюється разом з ним (результат). Зміна в описі інтерфейсу користувача повинна бути відображена на пристрої, на якому ми працюємо.

У браузері нам потрібно створювати HTML інтерфейси у Document Object Model (DOM). З React нам не потрібно турбуватися про те, як відобразити ці зміни і навіть коли внести ці зміни до браузера; React сам відреагує на зміни в state та автоматично оновить DOM у потрібний момент.

3 — Віртуальне відображення інтерфейсів у пам'яті

За допомогою React ми пишемо HTML за допомогою JavaScript. Ми розраховуємо на можливості JavaScript для створення вдосконаленого HTML, що покладається на якісь дані та працюючи з ними. Удосконалений HTML - це те, що зазвичай роблять інші JavaScript фреймворки. Наприклад, Angular розширює HTML такими можливостями як циклами, умовами та іншими.

Коли ми отримуємо дані із сервера (AJAX), нам потрібно щось більше, ніж HTML для обробки цих даних. Тут потрібно використовувати вдосконалений HTML або JavaScript для створення HTML. Обидва підходи мають переваги та недоліки. React використовує останній підхід, наголошуючи на тому, що переваги сильніші за недоліки.

Насправді є одна важлива перевага – використання JavaScript для рендерингу HTML спрощує для React збереження віртуального HTML інтерфейсу в пам'яті (широко відоме як Virtual DOM). React використовує Virtual DOM, щоб спочатку відобразити віртуальне дерево HTML, а потім кожного разу, коли змінюється state і ми отримуємо нове дерево HTML, яке потрібно для DOM браузера, замість написання цілого нового дерева React запише тільки різницю між новим і попереднім деревами (у React є обидва

дерева в пам'яті). Цей процес відомий як Tree Reconciliation, і це найкраще, що відбулося у веб-розробці з часів AJAX.

2.2 Аналіз впливу React на інші фреймворки

React є однією з найпопулярніших веб-бібліотек для професійних розробників, незважаючи на те, що вона відносно нова в цій галузі. Це легше для сприйняття, і його можна використовувати для створення та тестування надійних та інтуїтивно зрозумілих інтерфейсів користувача швидко та легко.

Разом зі своєю дочірньою платформою React Native для мобільної розробки, React є однією з найкращих інтерфейсних веб-бібліотек, доступних розробникам прямо зараз.

Хоча немає очевидних ознак того, що React відстає з точки зору популярності, поява його альтернатив, а саме Vue і Backbone, призвела до того, що проблеми з React потрапили в центр уваги. Документація погана, і темпи розробки можуть бути приголомшливими для певних груп.

Переваги React

Ось деякі з переваг цього дивовижного фреймворку з відкритим кодом.

Швидкість

ReactJS використовує спеціальний синтаксис під назвою JSX (розширення JavaScript), який дозволяє HTML-лапти та синтаксис тегів відтворювати певні підкомпоненти.

Хоча розробники можуть писати лише на JavaScript, JSX набагато простіший у використанні та дозволяє розробникам швидко та легко створювати веб-додатки з рядками HTML.

Крім того, React дозволяє розробникам повторно використовувати окремі частини своєї програми як на стороні клієнта, так і на стороні сервера, що додатково прискорює швидкість розробки.

Гнучкість

Як обговорювалося раніше, програма React складається з кількох компонентів, які мають власну логіку та елементи керування, а отже,

відповідають за рендеринг певного фрагмента коду для багаторазового використання.

Модульна природа багаторазового коду полегшує розробку та підтримку програм, оскільки такі елементи можна зберігати разом з іншими як будівельні блоки для складних програм.

Продуктивність

React займається лише керуванням станом і рендерингом цього стану у віртуальний DOM, який є кросплатформним API програмування, що працює з компонентами HTML, XML або XHTML.

Традиційно оновлення DOM суттєво знижують продуктивність програми. Впроваджуючи віртуальні DOM, які повністю існують у пам’яті, React змушує додатки швидко реагувати та працювати швидше.

Недоліки React

Високі темпи розвитку

Хоча висока швидкість розробки має свої переваги, вона також за своєю суттю приголомшлива, особливо для нових розробників.

Середовище розробки додатків постійно змінюється, іноді настільки швидко, що розробникам може бути неприємно постійно застосовувати нові методології.

Як розробник React ви завжди будете оновлювати свої навички та вивчати нові методи.

Документація

Природа React, що постійно розвивається, також робить документацію одним із найгірших аспектів веб-фреймворку.

У React є кілька інструментів і бібліотек, які підвищують продуктивність, але вони, як і сам React, регулярно оновлюються.

Іноді розробка відбувається настільки швидко, що не вистачає часу на написання повної документації, тобто розробники часто залишаються з мізерною документацією, яку важко розгорнути.

View частина

Технічно React — це не фреймворк, а бібліотека, яка обробляє частину V архітектури MVC і нічого більше.

Незважаючи на те, що React є одним із найкращих, розробка повноцінних додатків за допомогою React вимагає використання інструментів і бібліотек сторонніх розробників.

Наприклад, щоб розгорнути React у серверних програмах, вам доведеться використовувати інші фреймворки, такі як Next.js.

Нижче наведені деякі з найкращих конкурентів React JS:

InfernoJS

InfernoJS — це швидка, легка, керована продуктивністю альтернатива React, яка надає бібліотеку JavaScript для розробки ефективних інтерфейсів користувача.

Він надзвичайно досвідчений у ізоморфному рендерингу переглядів даних у реальному часі та дотримується архітектури одностороннього потоку даних. Inferno має часткову синтетичну систему подій, що значно пришвидшує рендеринг, оновлення та видалення елементів із DOM шляхом делегування певних подій.

Ember JS

Ember[3], мабуть, є однією з найкращих альтернатив React і забезпечує структуру шаблонів, підтримку URL-адрес і рендеринг на стороні клієнта.

Він зручний для API та містить усі основні ідіоми, що значно покращує швидкість розробки. Сучасна система керування дозволяє розробникам переходити до останніх оновлень, тоді як інтегровані шаблони керма виконують кілька подій одночасно з меншим кодом.

Backbone JS

Backbone JS — ще одна популярна бібліотека JavaScript, яка надає структуру важким моделям додатків JS за допомогою користувацьких подій і зв’язування ключ-значення.

Він ідеально підходить для створення односторінкових веб-додатків, які зберігають координацію кількох сторінок, оскільки залежить лише від однієї бібліотеки JS, тобто Underscore.js і певних елементів із jQuery.

Backbone має інтерфейс RESTful JSON і використовує механізм шаблонів для легкого створення складних масштабованих інтерфейсів користувача.

Vue JS

VueJS є однією з найшвидше зростаючих альтернатив React і в основному націлений на розробку односторінкових програм і інтерфейсів користувача.

Фреймворк JavaScript з відкритим кодом є однією з найбільш гнучких, зручних для розробників і найшвидших альтернатив, доступних розробникам прямо зараз.

Він має гнучку процедуру інтеграції та менший розмір бібліотеки, що дозволяє Vue пропонувати підвищену продуктивність, але його легше вивчати та застосовувати, ніж React.

Angular JS

Angular JS був одним із найпопулярніших фреймворків, доступних на ринку, який дозволяв розробникам створювати динамічні односторінкові веб-додатки, додатки для комп’ютерів і мобільних пристроїв (SPA).

Він відіграв важливу роль у розробці, структуруванні та спрощенні програм JavaScript з архітектурою MVC і MVVM.

Angular JS в основному використовувався для розробки програм і SPA на основі веб-сайтів, які мали взаємодію з користувачем, оскільки він міг завантажувати всі нові сторінки з сервера та динамічно переписувати існуючий код.

Preact

Preact — одна з найшвидших віртуальних бібліотек DOM і альтернатив React JS, доступних розробникам. Він пропонує одну з найтонших абстракцій DOM і може бути використаний безпосередньо через браузер.

Preact пропонує численні розширені функції, такі як зв'язаний стан, рендеринг стану тощо, що робить його ідеальною альтернативою для розробки веб- і мобільних додатків і PWA.

Mithril

Mithril — це клієнтська платформа JavaScript, яка в основному використовується для створення односторінкових програм. Його легко освоїти, він завантажується надзвичайно швидко та має менший API порівняно з його альтернативами.

Він має гідну підтримку для сучасних браузерів і пропонує кілька утиліт, таких як маршрутизація, XHR, сумісність з Flux, надійність тощо.

Розробники можуть легко почати вивчати маршрутизацію, компоненти, а XHR займає менше часу для створення програм.

Cycle JS

Cycle JS — це реактивний фреймворк JS і альтернатива React, яка може працювати з реактивними та функціональними потоками.

Його можна використовувати для написання чистої асинхронної логіки з API в основі; Використовуючи Cycle JS, програма описана як одна функція, яка приймає потік подій як вхід і вихід з найменшою кількістю помилок.

Svelte

Svelte — досить популярний безкоштовний компілятор з відкритим вихідним кодом, який використовується для створення інтерфейсів користувача.

Він не містить сценарію фреймворку, а замість цього перекладає свою роботу на етап компіляції для створення кібернетично покращених веб-програм.

На відміну від своїх аналогів, Svelte не виконує свою роботу в браузері, а натомість миттєво оновлює DOM, щоб забезпечити кращу реакцію, масштабованість і швидкість.

jQuery

jQuery — одна з найпопулярніших існуючих бібліотек JavaScript, яка використовується понад 70% найпопулярніших веб-сайтів в Інтернеті.

Ця безкоштовна бібліотека з відкритим вихідним кодом розроблена для спрощення обходу дерева HTML DOM і маніпулювання ним, а також для створення плагінів на основі бібліотеки JS, що дозволяє розробникам створювати динамічні веб-сайти та веб-програми з розширеними ефектами та високорівневими тематичними віджетами.

Не можна заперечувати, що React є і залишатиметься одним із найпопулярніших інтерфейсних веб-фреймворків.

Однак тепер, коли доступно багато альтернатив React, розробники матимуть більше можливостей.

Вам потрібно буде ретельно зважити вимоги вашого проекту та вибрати веб-платформу для вашого проекту, яка найкраще їм відповідає.

Якщо ви не можете самостійно дійти висновку, подумайте про те, щоб зв’язатися з однією з провідних компаній, що займаються розміщенням додатків, і поговорити з її експертами, щоб дізнатися більше.

Вони також можуть допомогти вам вибрати найкращі програмні рішення для будь-яких майбутніх проектів, уважно оцінивши ваші вимоги.

2.3 Порівняльний аналіз фреймворків

Angular2

В даний час загальна продуктивність пристроїв користувача збільшувалася, тому стало можливим виконання логіки програми в браузері. Це призвело до підходу до створення односторінкових додатків або SPA, для розробки яких призначався AngularJS. Він також надавав можливість замінити низькопродуктивний jQuery, пропонуючи розробникам такі функції, як

двостороння прив'язка даних та можливість організації модулів для імпорту зовнішніх скриптів. Потім AngularJS був повністю переписаний під Angular2, відповідно змінилося безліч основних концепцій фреймворку. У Angular2 відсутні контролери або моделі подання, натомість є компоненти, які складаються із шаблону, класів та метаданих. В той час, як AnjularJS був зосереджений на шаблоні MVC, Angular2 повністю покладається на ієрархію компонентів [3]. Важливим нововведенням фреймворку є TypeScript, який додає у проект строгу типізацію, а також різний новий функціонал, такий як декоратори, ієнуми тощо. Згодом TypeSrypt компілюється JavaScript для роботи в браузері [3]. Angular2, також як React та Vue, заснований на компонентах, які відображають інформацію та шаблони, виконують дії з даними та складаються з трьох окремих файлів: HTML-файл для шаблону, CSS-файл для стилізації та TS-файл для контролю. Наслідуючи цей підхід, реалізується принцип поділу проблем, а також код стає більш організованим і структурованим. Всередині компонентів обмін даними між поданням (шаблоном HTML) та моделлю (файлом TypeScript) відбувається за рахунок прив'язки даних та подій. В Angular2 реалізовано власну систему маршрутизації для зручності використання SPA. Визначивши маршрути в окремому файлі або модулі, Angular2 обробляє логіку того, який компонент повинен відображатися залежно від поточного активного URL.

Переваги Angular:

• Нові функції, такі як покращений RXJS, прискорена компіляція (менше 3 секунд) та новий лаунчер HttpClient.

• Детальна документація, яка дозволяє кожному розробнику отримати всю необхідну інформацію без звернення по допомогу до колег. Тим не менше, навчання потребує багато часу.

• Двостороння прив'язка даних, яка забезпечує чудову поведінку програми, що мінімізує ризики можливих помилок.

• MVVM (Model-View-ViewModel), яка дозволяє розробникам окремо працювати в одному розділі програми з використанням одного і того ж набору даних.

• Впровадження залежностей функцій пов'язаних із компонентами з модулями та модульності в цілому.

Недоліки Angular:

• Складний синтаксис, що походить від першої версії Angular. Тим не менш, Angular використовує TypeScript 2.4, який вивчити не так вже й складно.

• Проблеми з міграцією, які можуть виникнути під час переходу від старої версії до нової.

Компанії, які використовують Angular: Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase.

Vue.js:

Vue описують як прогресивний фреймворк, принципи проектування якого частково взяті з патерну Model-View-ViewModel. Згідно з офіційним сайтом, фреймворк можна використовувати для розробки SPA та невеликих проектів у поєднанні з іншими бібліотеками, аналогічно React. Всі компоненти Vue дуже схожі на Custom Elements, які є частиною специфікації веб-компонентів і дозволяють створювати нові теги HTML або змінювати існуючі, доповнюючи їх функціями, використовуючи JavaScript, HTML і CSS.

Масштабованість – одна з головних переваг цього фреймворку серед інших технологій. Vue надійно працює у всіх підтримуваних браузерах. Крім того, компоненти фреймворку додають функціональність, яку не можуть забезпечити нативні елементи, наприклад, передачу подій, що настроюються, і міжкомпонентний потік даних.

Структурування Vue передбачає, кожен компонент повинен мати єдиний кореневий елемент. Функції Vue відображають Virtual DOM.

Фреймворк встановлює мінімальний набір компонентів необхідної кількості маніпуляцій з DOM-деревом.

Фреймворк надає офіційний інтерфейс командного рядка для покращення розробки, особливістю якого є той факт, що команда розробників пропонує набір попередньо налаштованих шаблонів для різних варіантів використання.

Переваги Vue.js:

• Посилений HTML. Це означає, що Vue.js має багато подібних до Angular характеристик. Це допоможе оптимізувати обробку HTML-блоків з використанням різних компонентів.

• Детальна документація. Vue.js має дуже хорошу документацію, яка може збільшити швидкість навчання розробників та заощадити багато часу на розробку програми з використанням базових знань HTML та JavaScript.

• Адаптивність. Vue.js забезпечує швидкий період переходу від інших фреймворків до Vue.js через його подібність з Angular і React з точки зору дизайну та архітектури.

• Чудова інтеграція. Vue.js можна використовувати як для створення односторінкових програм, так і для більш складних веб-інтерфейсів. Найважливіше, що невеликі інтерактивні частини можна легко інтегрувати в існуючу інфраструктуру, не надаючи негативного впливу на всю систему.

• Велике масштабування. Vue.js допомагає розробляти досить великі шаблони для багаторазового використання, які можна розробити без витрачання величезної кількості часу у вигляді простої структури.

• Крихітний розмір. Vue.js може важити близько 20 КБ і зберігати свою швидкість і гнучкість, що дозволяє досягти набагато більш високої продуктивності, порівняно з іншими фреймворками.

Недоліки Vue.js:

• Нестача ресурсів. Vue.js, як і раніше, має досить невелику частку ринку в порівнянні з React або Angular. Це означає, що обмін знаннями у рамках фреймворку все ще формується.

• Ризик надмірної гнучкості. Іноді у Vue.js можуть виникати проблеми при інтеграції у величезні проекти, а досвіду про можливі рішення досі немає. Але вони обов'язково з'являться найближчим часом.

• Відсутність повної англомовної документації. Це призводить до деяких складнощів на різних етапах розробки. Тим не менш, дедалі більше матеріалів перекладаються англійською мовою.

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

Angular vs React

Обидві бібліотеки можна використовувати для мобільних і веб-додатків, тоді як Angular, як правило, краще підходить для складніших додатків, готових до корпоративного використання.

React часто потребує додаткових модулів і компонентів, що зберігає невелику основну бібліотеку, але означає додаткову роботу під час включення зовнішніх інструментів. З іншого боку, Angular — це скоріше повноцінне рішення, яке не потребує додаткових функцій, як це часто робить React, хоча має крутішу криву навчання для свого ядра порівняно з React.

React більше підходить для розробників JavaScript середнього та просунутого рівня, які знайомі з концепціями від ES6 і вище, тоді як Angular надає перевагу тим самим розробникам, які також знайомі з TypeScript.

React vs Vue

Вибір між React і Vue часто обговорюється, і це непросто. Vue має активну та постійно зростаючу спільноту, яка в багатьох відношеннях переважає над React. Розробники React все ще створюють багато нових компонентів і додаткових можливостей, тому немає жодних ознак того, що React також занепадає.

Vue, як правило, більше підходить для невеликих, менш складних програм, і його легше вивчати з нуля порівняно з React. Vue може бути легше інтегрувати в нові або існуючі проекти, і багато хто вважає, що використання шаблонів HTML разом із JSX є перевагою.

Загалом, Vue може бути найкращим вибором, якщо ви новий розробник і не настільки знайомі з розширеними концепціями JavaScript, тоді як React досить добре підходить для досвідчених програмістів і розробників, які працювали з об'єктно-орієнтованим JavaScript, функціональним JavaScript і подібними концепціями. .

Angular vs Vue

У більшості випадків ви, мабуть, не вибираєте між Angular і Vue. Це дуже різні бібліотеки з дуже різними наборами функцій і кривими навчання. Vue є очевидним вибором для менш досвідчених розробників, а Angular буде кращим для тих, хто працює над великими програмами.

Велика бібліотека на кшталт Angular потребувала б більшої старанності, щоб слідкувати за новинками, тоді як Vue була б менш вимогливою в цьому відношенні, і той факт, що два найновіші основні випуски Vue знаходяться в окремих репозиторіях, допомагає.

Слід також зазначити, що Vue створив розробник, який раніше працював над Angular для Google, тому це ще одна річ, про яку слід пам'ятати, хоча це не матиме великого впливу на ваше рішення.

Таблиця 2.1 - Історія Angular vs React vs Vue

	Angular	**React**	**Vue**
Випуск	2010	2013	2014
Сайт	**angular.io**	**reactjs.org**	**vuejs.org**
Поточна версія	13.x	17.x	3.x
Використовується	Google, Wix	Facebook, Uber	Alibaba, GitLab

Популярність

Оскільки «angular» і «react» є загальними словами, важко зрозуміти їхню популярність із Google Trends. Хоча хорошим показником їх популярності є кількість зірок, які отримують їхні репозиторії на GitHub. Раптова зміна кількості зірок Vue відбулася в середині 2016 року, і нещодавно Vue виявився серед найпопулярніших фреймворків разом із React.

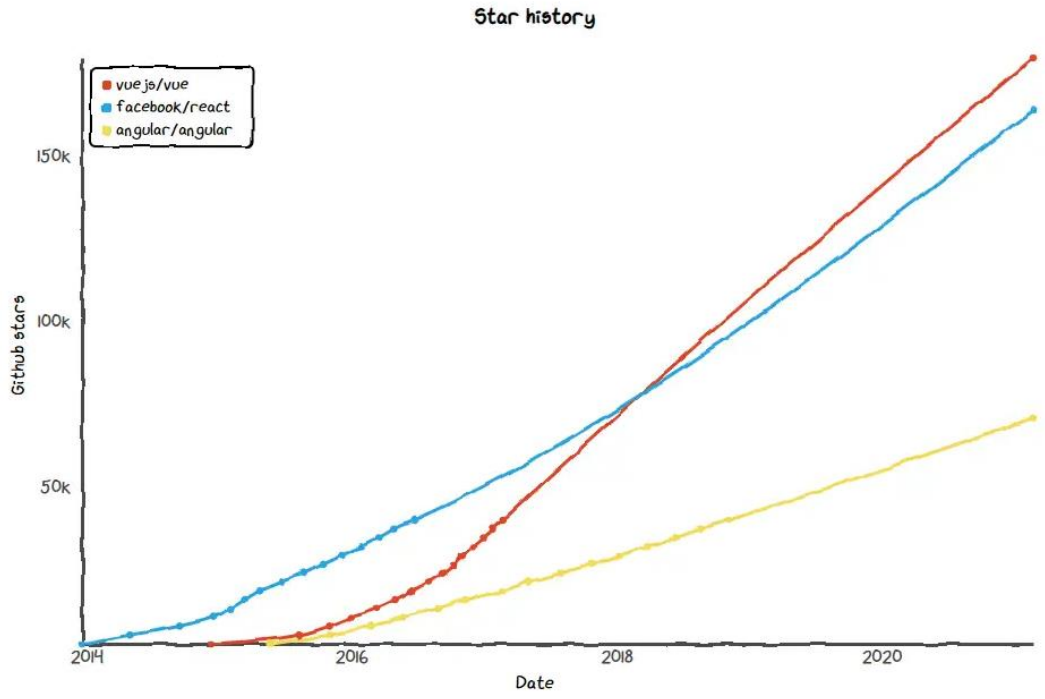

Рисунок 2.1 - Кількість зірок у проектах GitHub для Angular, React і Vue

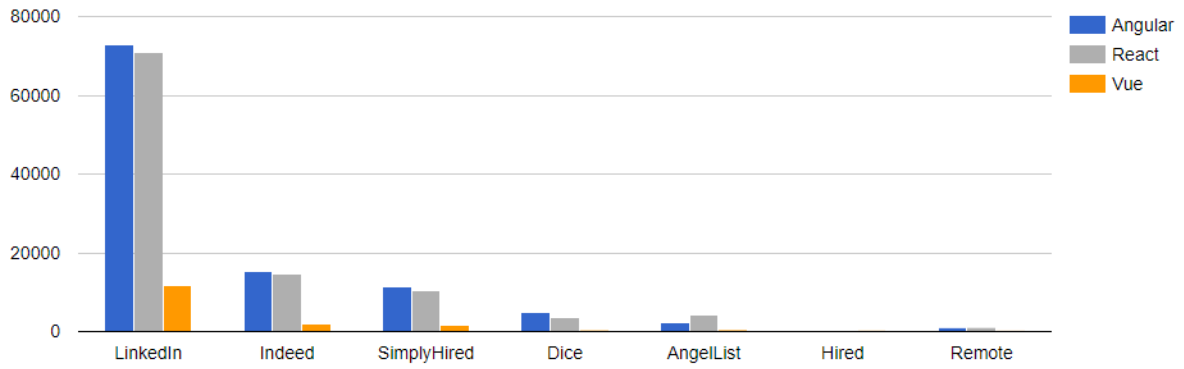

Рисунок 2.2 - Ринок праці для Angular vs React vs Vue

Як видно з тенденцій кінця 2018 року, кількість вакансій, для яких потрібен набір навичок Angular або React, приблизно однакова, тоді як у Vue все ще була лише частина цієї кількості (близько 20%).

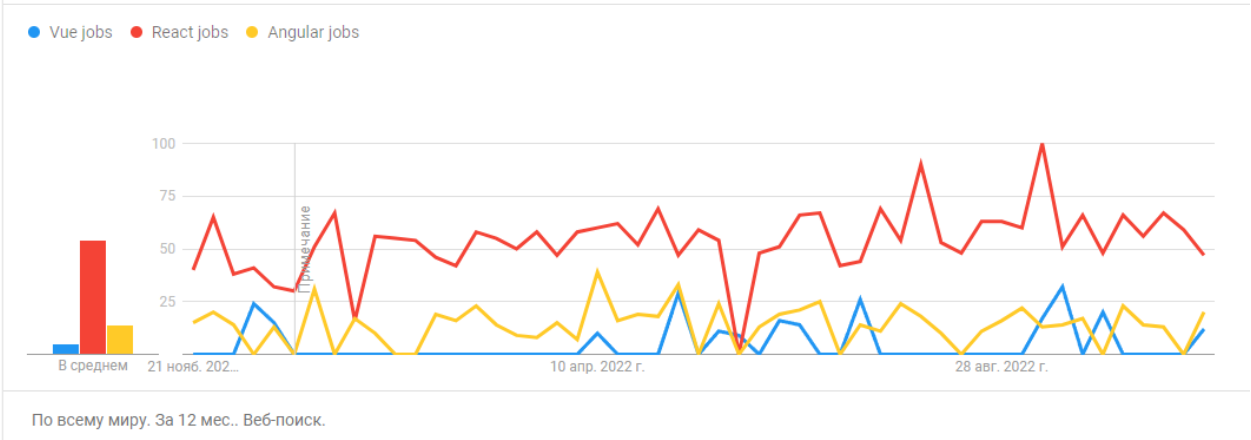

Рисунок 2.3- Ринок праці для Angular vs React vs Vue за 2022 рік

Громада та розвиток

Тепер, коли ви знайомі з історією та останніми тенденціями для кожного з цих фреймворків, ми подивимося на спільноту, щоб оцінити розвиток цих фреймворків. Ми вже бачили, що для всіх фреймворків протягом останнього року регулярно надходили додаткові випуски, що свідчить про те, що розробка йде повним ходом.

Давайте подивимося на Angular проти React проти Vue щодо статистики їхніх репозиторіїв GitHub (і зауважте, що цифри Vue також включають окреме сховище Vue 3.0):

Таблиця 2.2 – Статистика GitHub

	Angular	**React**	**Vue**
# Watchers	3.1k	6.7k	6.3k
# Stars	78.4k	180k	218k
# Forks	20.6k	36.5k	35.7k
# Contributors	1,500+	1,500+	400+

Порівнюючи Vue з React, Vue має величезну кількість спостерігачів, зірочок і розгалужень. Це показує популярність Vue серед користувачів і його цінність порівняно з React. Однак кількість учасників для Vue нижча, ніж для Angular і React.

Одним із можливих пояснень є те, що Vue повністю керується спільнотою з відкритим кодом, тоді як Angular і React мають значну частку співробітників Google і Facebook, які вносять свій внесок у репозиторії.

Згідно зі статистичними даними, усі три проекти показують значну активність у розробці, і це, безумовно, продовжуватиметься в майбутньому — просто ця статистика не може бути основою для відмови від використання жодного з них.

Додатковим показником, який ви захочете розглянути, є значок GitHub «Використовується», який має бути активований автором сховища. Це показує, скільки інших репозиторіїв на GitHub залежать від цього сховища. Репо GitHub від Angular показує 1,7 мільйона, React наразі показує майже 5,7 мільйона користувачів, тоді як Vue показує понад 167 000 для обох своїх репозиторіїв разом узятих. Досить різниця між трьома фреймворками, але це значною мірою пов’язано з тим, що Vue є новішим фреймворком і не дає повної картини загального попиту.

Міграції

Коли ви працюєте з обраним фреймворком, ви не хочете хвилюватися про оновлення фреймворку, яке зіпсує ваш код. Хоча в більшості випадків ви

не зіткнетеся з багатьма проблемами від однієї версії до іншої, важливо тримати руку на пульсі, оскільки деякі оновлення можуть бути більш значними та вимагати налаштувань, щоб підтримувати сумісність.

Angular планує основні оновлення кожні шість місяців. Існує також період у шість місяців, перш ніж будь-які основні API будуть припинені, що дає вам час у два цикли випуску (один рік), щоб внести необхідні зміни, якщо такі є.

Коли справа доходить до Angular проти React, Facebook заявив, що для них стабільність є надзвичайно важливою, оскільки величезні компанії, такі як Twitter і Airbnb, використовують React. Оновлення через версії, як правило, є найпростішим у React, зі скриптами, такими як react-codemod, які допомагають вам мігрувати.

У розділі «Міграція» документів Vue 3 Vue згадує, що Vue 2 і Vue 3 багато в чому однакові, тоді як 90% API однакові, якщо ви переходите з 1.x на 2. Існує Vue 2 на Vue 1 допоміжний інструмент міграції, який працює на консолі для оцінки статусу вашої програми.

Робота з Vue vs Angular vs React

Тут є кілька важливих характеристик, головними з яких є загальний розмір і час завантаження, доступні компоненти та крива навчання.

Розмір і час завантаження

Розміри бібліотек не будуть такими великими факторами, оскільки кешування та мініфікація сьогодні досить стандартні. Хоча між розмірами фреймворків може бути значна різниця (наприклад, Angular є найбільшим), вони все одно малі порівняно із середнім розміром веб-сторінки (приблизно 2 МБ за останніми даними). Крім того, якщо ви використовуєте популярний CDN для завантаження цих бібліотек, велика ймовірність того, що бібліотека користувача вже завантажена в його локальній системі.

Компоненти

Компоненти є невід'ємними частинами всіх трьох фреймворків, незалежно від того, чи ми говоримо про Vue, React чи Angular. Компонент зазвичай отримує вхідні дані та змінює поведінку на основі них. Ця зміна поведінки зазвичай проявляється як зміна інтерфейсу користувача певної частини сторінки. Використання компонентів полегшує повторне використання коду. Компонентом може бути кошик на сайті електронної комерції або вікно входу в соціальну мережу.

Angular:

В Angular компоненти називаються директивами. Директиви — це лише маркери на елементах DOM, які Angular також може відстежувати та додавати певну поведінку. Таким чином, Angular відокремлює частину інтерфейсу користувача компонентів як атрибути тегів HTML і їх поведінку у вигляді коду JavaScript. Це те, що відрізняє його від Angular проти React.

React:

Цікаво, що React поєднує UI та поведінку компонентів. Наприклад, ось код для створення компонента hello world у React. У React та сама частина коду відповідає за створення елемента інтерфейсу користувача та диктує його поведінку.

Vue:

Розглядаючи Vue проти React, у Vue UI та поведінка також є частиною компонентів, що робить речі більш інтуїтивно зрозумілими. Крім того, Vue має широкі можливості налаштування, що дозволяє комбінувати інтерфейс користувача та поведінку компонентів із сценарію. Крім того, ви також можете використовувати препроцесори у Vue, а не в CSS, що є чудовою функціональністю. Vue чудовий, коли справа доходить до інтеграції з іншими бібліотеками, такими як Bootstrap.

Крива навчання

Отже, наскільки складно вивчити кожну з цих структур?

Angular:

Angular має круту криву навчання, вважаючи, що це повне рішення, і оволодіння Angular вимагає від вас вивчення пов'язаних концепцій, таких як TypeScript і MVC. Незважаючи на те, що для вивчення Angular потрібен час, інвестиція приносить дивіденди з точки зору розуміння того, як працює інтерфейс.

React:

React пропонує посібник із початку роботи, який допоможе налаштувати React приблизно за годину. Документація є ретельною та повною, а рішення типових проблем уже є на Stack Overflow. React не є повним фреймворком, і розширені функції вимагають використання бібліотек сторонніх розробників. Це робить криву навчання основної структури не такою крутою, але залежить від шляху, який ви підете з додатковими функціями. Однак навчання використовувати React не обов'язково означає, що ви використовуєте найкращі практики.

Vue:

Vue забезпечує кращі можливості налаштування, і тому його легше вивчати, ніж Angular або React. Крім того, Vue перетинається з Angular і React щодо їх функціональності, як-от використання компонентів. Отже, перехід до Vue з будь-якого з двох є простим варіантом. Однак простота та гнучкість Vue — це палиця з двома кінцями — вона допускає поганий код, що ускладнює налагодження та тестування.

Незважаючи на те, що Angular, React і Vue мають значну криву навчання, їх використання безмежне. Наприклад, ви можете інтегрувати Angular і React з WordPress і WooCommerce для створення прогресивних веб-додатків.

Висновки до розділу 2

ReactJS — це продуктивна, ієрархічна та функціональна технологія, яка має низку переваг. Фреймворк пропонує широкий контекст із такими можливостями, як тривимірні зображення, загальні елементи, одностороннє інформаційне з'єднання, тригери та JSX.

Angular є найбільш зрілим фреймворком, має хорошу підтримку з точки зору учасників і є повним пакетом.

Однак крива навчання є стрімкою, і концепції розробки в Angular можуть відштовхнути нових розробників.

Angular — хороший вибір для компаній із великими командами та розробників, які вже використовують TypeScript.

React достатньо дорослий, щоб бути зрілим, і має величезну кількість внесків від спільноти. Він отримав широке визнання. Ринок праці для React справді хороший, і майбутнє для цього фреймворку виглядає яскравим.

React виглядає як хороший вибір для тих, хто починає працювати з інтерфейсними фреймворками JavaScript, стартапів і розробників, яким подобається гнучкість. Можливість легкої інтеграції з іншими фреймворками дає велику перевагу для тих, хто хоче певної гнучкості у своєму коді.

Vue є найновішим на арені, без підтримки великої компанії. Однак за останні кілька років він справді успішно виступив як сильний конкурент для Angular і React, особливо з випуском Vue 3.0. Можливо, це відіграє свою роль у тому, що багато китайських гігантів, таких як Alibaba та Baidu, обирають Vue як свою основну інтерфейсну структуру JavaScript. Vue має бути вашим вибором, якщо ви віддаєте перевагу простоті, але також любите гнучкість.

РОЗДІЛ 3

РОЗРОБКА І ТЕСТУВАННЯ АНАЛІТИЧНОГО ДОДАТКУ

У цьому розділі використовується поєднання кількісних і якісних методів для розробки способу порівняння структур SPA. Описано, як проводилося дослідження. Першим кроком, перш ніж запропонувати метод порівняння структур SPA, є проведення дослідження літератури. Це дослідження проводиться для того, щоб з'ясувати, чи проводилися аналогічні дослідження, і знайти порівняльні методи, які використовувалися раніше. Нарешті, аналіз методів, використаних у літературі, призвів до першого проекту методу порівняння. Цей метод складається з теоретичного та практичного порівняння, які розширюються далі в дослідженні. По-друге, проводиться більш повне дослідження літератури, щоб створити основу для абстракції структур SPA. Це дослідження стало основним джерелом даних, які були зібрані та відсортовані за різними критеріями. Як пропонує [14, с. 88-90], виділено ключові поняття та терміни. Однак замість термінів використовується слово критерій. Нарешті, вибирається методологія тестування та виконуються випробування продуктивності прототипів.

3.1 Формалізація задачі

Використовується загальний метод тестування продуктивності, запропонований [34]. Цей метод використовується для отримання структури того, як виконати тест продуктивності веб-додатку. Метод складається з семи кроків:

1. Визначення тестового середовища
2. Визначення критеріїв прийнятності продуктивності
3. Планування та проектування тестів
4. Налаштування тестового середовища
5. Реалізація дизайну тестування
6. Виконання тесту

7. Аналіз результатів, звітування та повторне тестування

Зазвичай, це серверна служба, яка тестується у веб-додатку. Натомість цей документ охоплює клієнтську сторону SPA. Вирішальною частиною продуктивності SPA є прив’язки даних, час завантаження та не менш важливий розподіл ресурсів.

Прив'язки даних

Прив’язки даних є важливою частиною веб-додатків. Як правило, двостороннє зв’язування даних використовується в рамках SPA. Ці прив’язки можна перевірити, завантаживши дані, і побачити, як швидко вони відображаються в View. Крім того, тести складаються з оновлення прив’язок даних новими даними. Цей тест показує, як швидко прив’язки даних реагують на зміни в моделі даних.

Час завантаження

Для початкового завантаження сервер повинен надіслати файл JavaScript користувачеві, а SPA має ініціалізувати. Для звичайного користувача час завантаження програми не повинен перевищувати 0,1 - 1 секунди. Однак до 10 секунд є верхньою межею для більшості користувачів, тобто початкові думки користувачів можуть бути перервані, і вони вирішать зробити щось інше [45].

Розподілення ресурсів

Смартфон може не мати такого ж обсягу доступної пам’яті, як настільний комп’ютер. Таким чином, необхідно обмежити навантаження на пам'ять під час роботи SPA. Існують різні способи контролювати це, наприклад, обмежити кількість зв’язків даних під час першого початкового завантаження.

3.2 Функції додатку

Наступні операції тестуються для кожного фреймворку:

- створити рядки: тривалість створення 1000 рядків після завантаження сторінки (без розминки).

```
export let benchRun = new class extends CPUBenchmarkPuppeteer {
  constructor() {
      super(benchmarksCommon.cpuBenchmarkInfos[benchmarksCommon.BENCHMARK_01]);
  }
  async init(page: Page) {
    await checkElementExists(page, "pierce/#run");
    for (let i = 0; i < config.WARMUP_COUNT; i++) {
      await clickElement(page, "pierce/#run");
      await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(1)>td:nth-of-type(1)", (i*1000+1).toFixed());
      await clickElement(page, "pierce/#clear");
      await checkElementNotExists(page, "pierce/tbody>tr:nth-of-
type(1000)>td:nth-of-type(1)");
    }
  }
  async run(page: Page) {
      await clickElement(page, "pierce/#run");
      await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(1000)>td:nth-of-type(1)", ((config.WARMUP_COUNT+1)*1000).toFixed());
  }

}
```

- замінити всі рядки: Тривалість для заміни всіх 1000 рядків таблиці (з 5 ітераціями розминки).

```
export const benchReplaceAll = new class extends CPUBenchmarkPuppeteer {
  constructor() {
      super(benchmarksCommon.cpuBenchmarkInfos[benchmarksCommon.BENCHMARK_02]);
  }
  async init(page: Page) {
      await checkElementExists(page, "pierce/#run");
      for (let i = 0; i < config.WARMUP_COUNT; i++) {
        await clickElement(page, "pierce/#run");
        await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(1)>td:nth-of-type(1)", (i*1000+1).toFixed());
      }
  }
  async run(page: Page) {
    await clickElement(page, "pierce/#run");
    await checkElementContainsText(page, "pierce/tbody>tr:nth-of-type(1)>td:nth-
of-type(1)", "5001");
  }
}
```

- часткове оновлення: час для оновлення тексту кожного 10-го рядка для таблиці з 10 000 рядків (з 5 ітераціями розминки).

```
export const benchUpdate = new class extends CPUBenchmarkPuppeteer {
  constructor() {
```

```
      super(benchmarksCommon.cpuBenchmarkInfos[benchmarksCommon.BENCHMARK_03]);
  }
  async init(page: Page) {
    await checkElementExists(page, "pierce/#run");
    await clickElement(page, "pierce/#run");
    await checkElementExists(page, "pierce/tbody>tr:nth-of-type(1000)>td:nth-of-
type(1)");
      for (let i = 0; i < 3; i++) {
        await clickElement(page, "pierce/#update");
        await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(991)>td:nth-of-type(2)>a", ' !!!'.repeat(i + 1));
      }
  }
  async run(page: Page) {
    await clickElement(page, "pierce/#update");
    await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(991)>td:nth-of-type(2)>a", ' !!!'.repeat(3 + 1));
  }
}
```

- вибрати рядок: тривалість виділення рядка у відповідь на клацання рядка. (з 5 ітераціями розминки).

```
export const benchSelect = new class extends CPUBenchmarkPuppeteer {
  constructor() {
      super(benchmarksCommon.cpuBenchmarkInfos[benchmarksCommon.BENCHMARK_04]);
  }
  async init(page: Page) {
    await checkElementExists(page, "pierce/#run");
    await clickElement(page, "pierce/#run");
    await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(1000)>td:nth-of-type(1)", "1000");
    for (let i = 0; i <= config.WARMUP_COUNT; i++) {
      await clickElement(page, `pierce/tbody>tr:nth-of-type(${i+5})>td:nth-of-
type(2)>a`);
      await checkElementHasClass(page, `pierce/tbody>tr:nth-of-type(${i+5})`,
"danger");
    }
  }
  async run(page: Page) {
      await clickElement(page, "pierce/tbody>tr:nth-of-type(2)>td:nth-of-
type(2)>a");
      await checkElementHasClass(page, "pierce/tbody>tr:nth-of-type(2)",
"danger");
  }
}
```

- поміняти місцями рядки: час поміняти місцями 2 рядки в таблиці з 1000 рядків. (з 5 ітераціями розминки).

```
export const benchSwapRows = new class extends CPUBenchmarkPuppeteer {
  constructor() {
      super(benchmarksCommon.cpuBenchmarkInfos[benchmarksCommon.BENCHMARK_05]);
  }
  async init(page: Page) {
      await checkElementExists(page, "pierce/#run");
      await clickElement(page, "pierce/#run");
      await checkElementExists(page, "pierce/tbody>tr:nth-of-type(1000)>td:nth-
of-type(1)");
      for (let i = 0; i <= config.WARMUP_COUNT; i++) {
          let text = ((i%2) == 0) ? "2" : "999";
          await clickElement(page, "pierce/#swaprows");
          await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(999)>td:nth-of-type(1)", text);
      }
  }
  async run(page: Page) {
      await clickElement(page, "pierce/#swaprows");
      await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(999)>td:nth-of-type(1)", "2");
      await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(2)>td:nth-of-type(1)", "999");
  }
}
```

- видалити рядок: тривалість видалення рядка для таблиці з 1000 рядків. (з 5 ітераціями розминки).

```
export const benchRemove = new class extends CPUBenchmarkPuppeteer {
  constructor() {
      super(benchmarksCommon.cpuBenchmarkInfos[benchmarksCommon.BENCHMARK_06]);
  }
  async init(page: Page) {
    await checkElementExists(page, "pierce/#run");
    await clickElement(page, "pierce/#run");
    await checkElementExists(page, "pierce/tbody>tr:nth-of-type(1000)>td:nth-of-
type(1)");
      for (let i = 0; i < config.WARMUP_COUNT; i++) {
          await checkElementContainsText(page, `pierce/tbody>tr:nth-of-
type(${config.WARMUP_COUNT - i + 4})>td:nth-of-type(1)`, (config.WARMUP_COUNT - i
+ 4).toString());
          await clickElement(page, `pierce/tbody>tr:nth-of-
type(${config.WARMUP_COUNT - i + 4})>td:nth-of-type(3)>a>span:nth-of-type(1)`);
          await checkElementContainsText(page, `pierce/tbody>tr:nth-of-
type(${config.WARMUP_COUNT - i + 4})>td:nth-of-type(1)`, "10");
      }
      await checkElementContainsText(page, `pierce/tbody>tr:nth-of-
type(5)>td:nth-of-type(1)`, "10");
      await checkElementContainsText(page, `pierce/tbody>tr:nth-of-
type(4)>td:nth-of-type(1)`, "4");
```

```
      await checkElementContainsText(page, `pierce/tbody>tr:nth-of-
type(6)>td:nth-of-type(1)`, "11");
      await clickElement(page, `pierce/tbody>tr:nth-of-type(6)>td:nth-of-
type(3)>a>span:nth-of-type(1)`);
      await checkElementContainsText(page, `pierce/tbody>tr:nth-of-
type(6)>td:nth-of-type(1)`, "12");

  }
  async run(page: Page) {
    await clickElement(page, `pierce/tbody>tr:nth-of-type(4)>td:nth-of-
type(3)>a>span:nth-of-type(1)`);
    await checkElementContainsText(page, `pierce/tbody>tr:nth-of-type(4)>td:nth-
of-type(1)`, "10");
  }
}
```

- створити багато рядків: тривалість створення 10 000 рядків (без розминки)

```
export const benchRunBig = new class extends CPUBenchmarkPuppeteer {
  constructor() {
    super(benchmarksCommon.cpuBenchmarkInfos[benchmarksCommon.BENCHMARK_07]);
    }
    async init(page: Page) {
      await checkElementExists(page, "pierce/#run");
      for (let i = 0; i < config.WARMUP_COUNT; i++) {
        await clickElement(page, "pierce/#run");
        await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(1)>td:nth-of-type(1)", (i*1000+1).toFixed());
        await clickElement(page, "pierce/#clear");
        await checkElementNotExists(page, "pierce/tbody>tr:nth-of-
type(1000)>td:nth-of-type(1)");
      }
    }
    async run(page: Page) {
      await clickElement(page, "pierce/#runlots");
      await checkElementExists(page, "pierce/tbody>tr:nth-of-type(10000)>td:nth-
of-type(2)>a");
    }
  }
```

- додавання рядків до великої таблиці: тривалість додавання 1000 рядків до таблиці з 10 000 рядків (без розминки).

```
  export const benchAppendToManyRows = new class extends CPUBenchmarkPuppeteer {
  constructor() {
      super(benchmarksCommon.cpuBenchmarkInfos[benchmarksCommon.BENCHMARK_08]);
  }
  async init(page: Page) {
    await checkElementExists(page, "pierce/#run");
```

```
    await clickElement(page, "pierce/#run");
    await checkElementExists(page, "pierce/tbody>tr:nth-of-type(1000)>td:nth-of-
type(1)");
  }
  async run(page: Page) {
    await clickElement(page, "pierce/#add");
    await checkElementExists(page, "pierce/tbody>tr:nth-of-type(2000)>td:nth-of-
type(1)");
  }
}
```

- очистити рядки: тривалість очищення таблиці, заповненої 10 000 рядками. (без розминки)

```
export const benchClear = new class extends CPUBenchmarkPuppeteer {
  constructor() {
      super(benchmarksCommon.cpuBenchmarkInfos[benchmarksCommon.BENCHMARK_09]);
  }
  async init(page: Page) {
    await checkElementExists(page, "pierce/#run");
    for (let i = 0; i < config.WARMUP_COUNT; i++) {
      await clickElement(page, "pierce/#run");
      await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(1)>td:nth-of-type(1)", (i*1000+1).toFixed());
      await clickElement(page, "pierce/#clear");
      await checkElementNotExists(page, "pierce/tbody>tr:nth-of-
type(1000)>td:nth-of-type(1)");
    }
    await clickElement(page, "pierce/#run");
    await checkElementContainsText(page, "pierce/tbody>tr:nth-of-type(1)>td:nth-
of-type(1)", (config.WARMUP_COUNT*1000+1).toFixed());
  }
  async run(page: Page) {
      await clickElement(page, "pierce/#clear");
      await checkElementNotExists(page, "pierce/tbody>tr:nth-of-
type(1000)>td:nth-of-type(1)");
  }
}
```

- готова пам'ять: використання пам'яті після завантаження сторінки.

```
export const benchReadyMemory = new (class extends MemBenchmarkPuppeteer {
  constructor() {
    super(benchmarksCommon.memBenchmarkInfos[benchmarksCommon.BENCHMARK_21]);
  }
  async init(page: Page) {
    await checkElementExists(page, "pierce/#run");
  }
  async run(page: Page) {return await Promise.resolve(null);}
```

```
})();
```

- пам'ять для виконання: використання пам'яті після додавання 1000 рядків.

```
export const benchRunMemory = new (class extends MemBenchmarkPuppeteer {
  constructor() {
    super(benchmarksCommon.memBenchmarkInfos[benchmarksCommon.BENCHMARK_22]);
  }
  async init(page: Page) {
    await checkElementExists(page, "pierce/#run");
  }
  async run(page: Page) {
    await clickElement(page, "pierce/#run");
    await checkElementExists(page, "pierce/tbody>tr:nth-of-type(1)>td:nth-of-
type(2)>a");
  }
})();
```

- оновити пам'ять: використання пам'яті після 5 разів клацання оновити для таблиці з 1000 рядків.

```
export const benchUpdate5Memory = new (class extends MemBenchmarkPuppeteer {
  constructor() {
    super(benchmarksCommon.memBenchmarkInfos[benchmarksCommon.BENCHMARK_23]);
  }
  async init(page: Page) {
    await checkElementExists(page, "pierce/#run");
  }
  async run(page: Page) {
    await clickElement(page, "pierce/#run");
    await checkElementExists(page, "pierce/tbody>tr:nth-of-type(1000)>td:nth-of-
type(2)>a");
    for (let i = 0; i < 5; i++) {
      await clickElement(page, "pierce/#update");
      await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(1)>td:nth-of-type(2)>a", " !!!".repeat(i));
    }
  }
})();
```

- замінити пам'ять: використання пам'яті після 5 разів клацання створює 1000 рядків.

```
export const benchReplace5Memory = new (class extends MemBenchmarkPuppeteer {
  constructor() {
    super(benchmarksCommon.memBenchmarkInfos[benchmarksCommon.BENCHMARK_24]);
  }
  async init(page: Page) {
    await checkElementExists(page, "pierce/#run");
  }
```

```
  async run(page: Page) {
    for (let i = 0; i < 5; i++) {
      await clickElement(page, "pierce/#run");
      await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(1000)>td:nth-of-type(1)", (1000 * (i + 1)).toFixed());
    }
  }
})();
```

- повторне очищення пам’яті: використання пам’яті після створення та очищення 1000 рядків протягом 5 разів.

```
export const benchCreateClear5Memory = new (class extends MemBenchmarkPuppeteer {
  constructor() {
    super(benchmarksCommon.memBenchmarkInfos[benchmarksCommon.BENCHMARK_25]);
  }
  async init(page: Page) {
    await checkElementExists(page, "pierce/#run");
  }
  async run(page: Page) {
    for (let i = 0; i < 5; i++) {
      await clickElement(page, "pierce/#run");
      await checkElementContainsText(page, "pierce/tbody>tr:nth-of-
type(1000)>td:nth-of-type(1)", (1000 * (i + 1)).toFixed());
      await clickElement(page, "pierce/#clear");
      await checkElementNotExists(page, "pierce/tbody>tr:nth-of-
type(1000)>td:nth-of-type(1)");
    }
  }
})();
```

- оновити пам'ять: використання пам'яті після 5 разів клацання оновити для таблиці з 1000 рядків.
- час запуску: тривалість завантаження та аналізу коду JavaScript і відтворення сторінки.
- Постійно інтерактивний: метрика TimeToConsistentlyInteractive: песимістичний TTI – коли ЦП і мережа безперечно простоюють. (більше жодних завдань ЦП понад 50 мс)
- час завантаження сценарію: метрика ScriptBootUpTtime: загальна кількість мс, необхідна для аналізу/компіляції/оцінки всіх сценаріїв сторінки

- вартість роботи основного потоку: метрика маяка MainThreadWorkCost: Загальна кількість часу, витраченого на роботу над основним потоком, включає стиль/макет тощо.
- загальна вага в байтах: метрика маяка TotalByteWeight: вартість передачі по мережі (після стиснення) усіх ресурсів, завантажених на сторінку.

3.3 Тестування та налагодження

Немає сумніву, що для порівняння чогось такого складного, як каркас, необхідна абстракція. Як запропоновано в [20], абстракція може бути зроблена шляхом створення «концептуальної основи», «архітектури плану» або «структури проекту». Абстракцію можна побудувати за допомогою сценарного або критеріального підходу. Усі методи, використовують деякі з цих концепцій, які підсумовано нижче:

Таблиця 3.1

Концепція	Посилання
Критеріальне порівняння	[10, 15, 21, 40, 41, 57]
Порівняння за сценарієм	[13, 16, 33, 38, 39]
Збір бізнес-цілей	[10, 13, 57]
Оцінювання за допомогою так\ні	[15, 21, 40]
Оцінювання з відсотками	[10, 21, 57]
Оцінювання з обговоренням	[9, 13, 20, 33, 38, 39, 41, 57]
Зважене оцінювання	[10, 21, 38, 57]
Оцінювання одного кандидата	[13, 16, 33, 38, 39]
Порівняння різних кандидатів	[10, 15, 21, 40, 41, 57]
Містять «соціальні аспекти»	[13, 16, 39]
Прототипування	[13, 15, 38, 40, 41]
Тестування продуктивності	[41]
Один/багато метрик для кожного критерію	[15, 20, 21, 41]

Як видно з таблиці 3.1, усі методи на основі сценаріїв можуть одночасно оцінювати лише одну архітектуру-кандидата, а методи на основі критеріїв використовуються для порівняння різних кандидатів. У [10, 13, 57] автори пропонують використовувати бізнес-цілі як основу для збору сценаріїв. Ці сценарії є специфічними та можуть відрізнятися від проекту до проекту. Іншим способом порівняння архітектури програмного забезпечення є використання багатьох різних груп співробітників і зовнішніх експертів. Деякі з цих аспектів запропоновані [13, 16, 39]: • Семінари з різними командами розробників, щоб отримати ширшу перспективу. • Залучення всієї команди на початку фази розробки. • Залучення сторонніх експертів до процесу проектування. • Презентація результатів для інших команд, залучених до процесу проектування. Ці соціальні аспекти настійно рекомендуються під час роботи над проектом. Сценарії можуть добре вписуватися в початковий процес розробки. Однак у цьому документі пропонується метод, який можна використовувати для порівняння інфраструктур SPA, тому сценарії не є відповідним вибором. Причина цього полягає в тому, що метод, запропонований у цій дисертації, не має жодних конкретних частин, які вимагають інших учасників. Також рекомендується, щоб одна й та сама особа або особи використовували метод, оскільки відповіді можуть відрізнятися від людини до людини, що робить фактичне порівняння упередженим. Автори SACAM, C. Stoermer, F. Bachmann і C. Verhoef стверджують: «Порівняння архітектур програмного забезпечення передбачає набір критеріїв. Порівняння без жодних критеріїв не дає жодних обґрунтувань щодо вибору». [57]. Як зазначено в [57], використання критеріїв дає обґрунтоване обґрунтування того, наскільки добре працює архітектура чи фреймворк у порівнянні з іншими. Природно, що важливо виділити ряд критеріїв, які є загальними та істотними. Ця робота зосереджена на визначенні набору критеріїв як фундаментальної частини запропонованого методу. Порівняння здійснюється шляхом

визначення кожного критерію та формулювання запитань, запропонованих [15,21,40,41]. Ці запитання можна розглядати як показники того, наскільки добре запитання відповідають критеріям, запропонованим у [15, 20, 21, 41]. Ваги можна використовувати, однак ця теза не пропонує жодного конкретного способу їх використання. Ігнасіо Фернандес-Вілламор та інші у [21] використовують ваги для кожного запитання та дають оцінку кожному критерію на основі запитань та їх ваги.

Висновки до розділу 3

Щоб обмежити кількість критеріїв, вибираються лише критерії з двома або більше посиланнями. Стабільність і зрілість пов’язані, оскільки зрілість програмного забезпечення впливає на стабільність, тому вони об’єднані за одним критерієм. Щоб мати можливість використовувати фреймворк SPA в якомога більшій кількості браузерів, сумісність і портативність є важливими, тому ці два критерії згруповані разом. У інформатиці постійність часто стосується збереження поточного стану програми в пам’яті. Щоб зберегти стан SPA в пам’яті, потрібна хороша продуктивність кешу. Це призводить до групування стійкості та продуктивності кешу разом. Зручність використання може бути визначена стандартом ISO 9241-11, який охоплює сферу, де користувач повинен бути задоволений використанням програмного забезпечення [18]. Це добре поєднується з простотою, тому що якщо щось є простим у використанні, воно повинно мати високу зручність використання, навпаки, те, що непросто у використанні, має мати нижчий рівень зручності використання. Тому простота і зручність у використанні згруповані. Документація, розмір фреймворку та внутрішній досвід вибираються як запитання для використання відповідно до відповідних критеріїв.

РОЗДІЛ 4

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РІЗНИХ ФРЕЙМВОРКІВ

4.1 Підготовка до експерименту

Angular

Розробка за допомогою Angular фреймворку потребує використання Angular CLI – інтерфейсу командного рядка, який необхідний для ініціалізації компонент проекту та проведення наступних маніпуляцій з файлами. Основні версії Angular CLI відповідають підтримуваній основній версії Angular, але незначні змінені версії можуть випускатися окремо. За встановлення CLI за допомогою менеджера пакетів npm відповідає команда: npm install -g @angular/cli

React

Для початку роботи з React.js потрібно використати два CDN посилання, перше для включення React компоненти:

<script crossorigin src="https://unpkg.com/react@17/umd/react.production.min.js"></script>

Та друге, для компоненти ReactDOM:

<script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.production.min.js"></script>

Або ж за допомогою пакету Create React App командою:

npx create-react-app.

Vue

Аналогічно з попереднім інструментом, для початку розробки можна використати наступні CND посилання, додавши його у index.html файл. Посилання для розробки, що містить допоміжні консольні попередження:

<script src="https://cdn.jsdelivr.net/npm/vue@2.7.13"></script>

Або версія посилання для подального використання, з оптимізацією розміру та швидкості:

```
<script type="module">
 import Vue from
'https://cdn.jsdelivr.net/npm/vue@2.7.13/dist/vue.esm.browser.js'
</script>
```

Альтернативним способом інсталяції Vue.js та створення проекту є застосування Vue CLI – системи для пришвидшення розробки на Vue, яка дозволяє отримати початкові налаштування для збірки проекту за допомогою webpack та має багато допоміжних плагінів.

4.1.1 Опис використаного програмно-апаратного забезпечення

Експримент проводився на Razer Blade 15 із зарядженою батареєю та i7-8750H (64 ГБ оперативної пам’яті, Windows 11, Chrome 104.0.5112.81) за допомогою драйвера тесту puppeteer зі зменшеним трасуванням.

4.1.2 Опис підходу для порівняння фреймворків

У наведеній нижче таблиці наведено зведення критеріїв, які можна знайти в дослідженні:

Критерій	Посилання
Продуктивність кешу	[41]
Документація	[41, 55]
Інтеграція	[55]
Ремонтопридатність	[41]
Зрілість	[41]
Модифікованість	[20]
Модульність	[20, 21, 41]
Продуктивність	[20, 41, 55]
Наполегливість	[20, 21]
Популярність	[21, 41]
Портативність	[20, 40, 41]
Презентація	[21]

Надійність	[15]
Масштабованість	[20, 55]
Простота	[15, 20, 21]
Безпека	[55]
Перевіряємість	[20, 21, 41]
Прозорість	[20]
Юзабіліті	[15, 21]

Щоб обмежити кількість критеріїв, вибираються лише критерії з двома або більше посиланнями. Стабільність і зрілість пов’язані, оскільки зрілість програмного забезпечення впливає на стабільність, тому вони об’єднані за одним критерієм. Щоб мати можливість використовувати фреймворк SPA в якомога більшій кількості браузерів, сумісність і портативність є важливими, тому ці два критерії згруповані разом. У інформатиці постійність часто стосується збереження поточного стану програми в пам’яті. Щоб зберегти стан SPA в пам’яті, потрібна хороша продуктивність кешу. Це призводить до групування стійкості та продуктивності кешу разом. Зручність використання може бути визначена стандартом ISO 9241-11, який охоплює сферу, де користувач повинен бути задоволений використанням програмного забезпечення [18]. Це добре поєднується з простотою, тому що якщо щось є простим у використанні, воно повинно мати високу зручність використання, навпаки, те, що непросто у використанні, має мати нижчий рівень зручності використання. Тому простота і зручність у використанні згруповані. Документація, розмір фреймворку та внутрішній досвід вибираються як запитання для використання відповідно до відповідних критеріїв.

Безпека

Сьогодні багато веб-додатків зберігають облікові дані користувачів або номери кредитних карток у базі даних, підключеній до Інтернету. Якщо це буде порушення безпеки у веб-додатку, усі дані користувача можуть бути

викрадені, а довіра користувачів може бути втрачена. OWASP (Open Web Application Security Project) [4] — некомерційна організація та онлайн-спільнота, основною метою якої є підвищення безпеки програмного забезпечення. Одним із найвідоміших їхніх внесків є список 10 найпоширеніших проблем безпеки веб-додатків.

SQ1: Коли виявлено помилку, пов’язану з безпекою, чи має структура політику безпеки [3]? Політика безпеки — це те, як розробники, що стоять за фреймворком, вирішують питання безпеки. Ця політика може стосуватися відповіді на проблему електронною поштою, кого сповістити та скільки часу займає оновлення фреймворку.

SQ2: Чи є у фреймворку заохочення для пошуку помилок [29]? Такі акції, як правило, приносять гроші людям, які знаходять помилки безпеки у веб-програмі. Це може змусити більше людей шукати помилки, пов’язані з безпекою.

SQ3: Чи запобігає фреймворк міжсайтовій підробці запитів [1]? Підробка міжсайтового запиту — це проблема безпеки, коли форма у веб-програмі є вразливою до маніпуляцій, про що користувач не знає.

SQ4: Чи запобігає фреймворк міжсайтовому сценарію [5]? Міжсайтовий сценарій є ще одним недоліком безпеки веб-додатків. Використовуючи цей недолік, у браузері можна запустити шкідливий JavaScript.

Модульність

Мета модульності полягає в тому, щоб дати структуру того, як має відбуватися розвиток з конкретним SPA. Прикладом модульності може бути розділення всіх компонентів. Якщо компонент переміщено з програми, він повинен працювати як ізольована одиниця та бути повністю функціональним. Розділення також може означати наявність окремого перегляду та рівня даних у програмі. Такий підхід полегшує зміну ізольованих функцій у певному компоненті та полегшує обслуговування SPA.

MQ1: Чи підтримує фреймворк шаблон проектування, який спрощує розробку окремих шарів або компонентів [41]? Як згадувалося вище, розділення є важливим, і структура може допомогти цьому, запропонувавши шаблон проектування, який підходить для дизайну на основі компонентів або структури шарів.

MQ2: Чи підтримує фреймворк компонентний підхід до розробки? Наявність компонентів, які працюють як ізольовані одиниці, забезпечує кращу модульність, де компоненти можна переміщувати або змінювати, не заважаючи решту програми.

Популярність

Використання популярного фреймворку може бути важливим фактором як для зрілості, так і для безпеки. Багато досвіду накопичено на StackOverflow та подібних форумах, де розробники обговорюють, як вони вирішували типові проблеми. Велика база розробників може створити більше звітів про помилки, і більше людей, ймовірно, буде залучено до вирішення цих проблем [52].

PQ1: Скільки спостерігачів GitHub має репозиторій фреймворку [41]? Якщо фреймворк має багато підписників на GitHub, ширша аудиторія може спробувати версію, яка все ще знаходиться в розробці.

Зрілість і стабільність

Не існує точного методу вимірювання зрілості та стабільності програмного забезпечення. CMM (Capability Maturity Model) можна використовувати для вимірювання зрілості процесу розробки програмного забезпечення в організації [47]. Однак важко знати процеси, оскільки вони не є загальнодоступними. Натомість зрілість і стабільність можна визначити як широко використовується рамка великими корпораціями.

MSQ1: Чи існує система більшої корпорації, яка використовує структуру SPA у виробничому середовищі? Якщо велика компанія

використовує цей фреймворк в одній зі своїх виробничих систем, це може свідчити про зрілість і стабільність фреймворку та її віру в його майбутнє.

Простота та зручність використання

Майже неможливо додати нову функціональність до програмного забезпечення, не вносячи жодних змін у кодову базу. Зберігаючи невеликі зміни, можна уникнути деяких дефектів. Навпаки, якщо потрібні значні зміни, розробнику не потрібно вводити більше складності, ніж необхідно. Враховуючи контекст, простота може відрізнятися між користувачами фреймворку. Для початкового розробника програмного забезпечення може бути простіше змінити код, ніж для нового розробника. Це питання може бути вирішено шляхом певного розподілу компетенції, наприклад, документації чи інструкцій [37, с. 49-50].

SUQ1: Чи є власний досвід? Шанси на успіх вищі, якщо вже є знання фреймворку. Крім того, досвідчений персонал може навчити менш досвідчених колег.

SUQ2: Чи існує документація [41]? Під час роботи з новим фреймворком або розширення існуючої програми новими функціями документація важлива для вирішення проблем.

SUQ3: Чи забезпечує будь-який сторонній інструмент створення коду [21]? Замість того, щоб писати повторюваний код, інструменти сторонніх розробників можуть генерувати код і економити час для розробника.

SUQ4: Чи підтримує будь-яка IDE цю структуру? IDE (інтегроване середовище розробки) служить розробнику для підтримки автоматичного завершення коду або документації API під час написання.

SUQ5: Яку мову JavaScript підтримує фреймворк? Існує багато різних версій JavaScript і існує безліч мов на основі JavaScript, метою яких є спрощення розробки.

Портативність і сумісність

JavaScript виконується в браузері, і важливо, щоб фреймворк підтримувався якомога більшою кількістю браузерів. Не менш важливою є підтримка мобільних пристроїв, оскільки багато людей використовують свої мобільні пристрої для перегляду Інтернету.

PCQ1: Кожен випуск яких браузерів тестується [41]? З міркувань переносимості важливо, щоб усі випуски тестувалися в якомога більшій кількості браузерів.

PCQ2: Яка остання версія браузера підтримується [40, 41]? Більшість користувачів частіше оновлюють свої браузери, за винятком користувачів Internet Explorer. Більшість користувачів Internet Explorer використовують версії 8, 9, 10, 11, а також новий веб-браузер Microsoft Edge 12 і Edge 13 [56]. Тому важливо підтримувати останню доступну версію.

PCQ3: Яка найраніша версія браузера підтримується? Підтримка застарілих версій є ще одним важливим фактором, оскільки багато користувачів досі не використовують останні версії свого веб-браузера. Як, видно, з PCQ2, база користувачів Internet Explorer більш розповсюджена в багатьох версіях веб-браузера.

PCQ4: За допомогою якої версії JavaScript (ECMAScript) створено фреймворк? Наразі ECMAScript 6 повністю підтримується не всіма браузерами. З цієї причини весь JavaScript, який використовує нову функціональність ECMAScript 6, потрібно скомпілювати до ECMAScript 5. Якщо фреймворк створено з ECMAScript 6 або будь-якою майбутньою версією ECMAScript, яка наразі не повністю підтримується всіма браузерами, потрібно скомпілювати в останн. версію, яка наразі підтримується.

PCQ5: Чи підтримує платформа мобільні пристрої? Сьогодні багато користувачів переглядають веб-сторінки за допомогою мобільних пристроїв. Тому важливо, щоб фреймворк підтримував мобільні пристрої.

PCQ6: Чи сумісна структура з іншими бібліотеками, які потрібні програмі? При розробці веб-програми зазвичай використовують більше ніж

одну бібліотеку чи фреймворк. Тому важливо, щоб фреймворк був сумісний із бібліотеками або фреймворками, які використовуються.

Продуктивність і стійкість кешу

Продуктивність і стійкість кешу важливі в SPA для покращення взаємодії з користувачем. Якщо початковий час завантаження занадто великий, користувач може вийти з SPA до того, як його буде завантажено належним чином. Ще одна причина мати деякі утиліти для підвищення продуктивності кешу — зменшити навантаження даних із сервера на SPA. Таким чином, це може заощадити ресурси на сервері та зменшити час очікування користувача.

CPQ1: Який розмір файлу JavaScript (розділ 4.1)? Щоб користувач міг використовувати SPA, потрібно завантажити файли JavaScript. Більший розмір файлу збільшує час завантаження та збільшує навантаження на сервер.

CPQ2: Чи існує якась функція для підтримки меншої кількості передачі даних між SPA та сервером [41]? Фреймворк може економити передачу даних шляхом кешування ресурсів.

Можливість тестування

Незалежно від того, чи використовують розробники розробку, керовану тестуванням, чи ні, важливо мати структуру, яку можна належним чином протестувати. Щоб переконатися, що зміни не заважають старим тестам або що нові функції можна протестувати перед новим випуском.

TQ1: чи підтримує фреймворк модульне тестування [21,41]? Модульне тестування дозволяє перевірити модуль, щоб переконатися в його функціональності.

TQ2: Чи підтримує фреймворк інтеграційне тестування [21]? Тестування інтеграції або наскрізне тестування дозволяє перевірити повну програму.

TQ3: Чи підтримує фреймворк функціональне тестування [21]? Функціональне тестування дозволяє перевірити певну логіку в додатку.

TQ4: Чи підтримує фреймворк тестування продуктивності [21, 41]? Тестування продуктивності дозволяє тестувати частини або всю програму за допомогою порівняльного аналізу або профілювання.

TQ5: Чи підтримує фреймворк знущання над об'єктами [21, 41]? Знущання над об'єктами полегшує виконання деяких тестів, коли даних недостатньо або їх важко визначити.

Ремонтопридатність

У розробці програмного забезпечення зазвичай одна команда починає розробку, а інша бере на себе. Одним із прикладів високого рівня ремонтопридатності є те, що новий розробник може почати швидко виправляти проблеми та помилки [51].

MaQ1: Чи підтримує фреймворк будь-які вказівки для розробників? При роботі у великій команді важливо, щоб кожен залучений розробник знав, як працювати з фреймворком.

MaQ2: Чи є досвід використання цього фреймворку серед програмістів [21]? Якщо існує багато розробників, які знають цю конкретну структуру, це зменшує ризик неможливості залучити досвідчених розробників.

MaQ3: Хто стоїть за цим розробником? Розробником є одна особа, команда розробників чи компанія? Компанія, яка розробляє фреймворк, має професіоналів, які працюють з ним щодня, одна людина може не мати змоги працювати з фреймворком щодня та не оновлювати фреймворк так часто, як це необхідно.

MaQ4: Чи використовує команда, що стоїть за фреймворком, його у власному виробничому середовищі? Якщо команда розробників, що стоїть за фреймворком, використовує свою власну структуру у виробничому середовищі, це може бути ознакою того, що вони вірять у свій продукт і що він підходить для виробничої системи.

4.2 Проведення експерименту

Фреймворк	vanillajs, с	vue-v3.2.37, с	angular-v13.0.0, с	react-v17.0.2, с
Створення 1000 рядків (5 розминкових прогонів).	96.7±1.1	119.3±1.2	118.5±1.4	126.2±1.4
Оновлення всіх 1000 рядків (5 розминкових запусків).	97.6±1.5	113.2±1.5	130.7±1.0	126.7±0.9
Оновлення кожного 10-го рядка для 1000 рядків (3 розминки). Уповільнення процесора в 16 разів.	297.1±8.8	359.0±11.2	343.0±6.7	399.5±5.4
Виділення виділеного рядка. (5 розминкових запусків). Уповільнення процесора в 16 разів.	33.1±1.2	53.7±1.2	43.6±0.8	105.2±3.7
Поміняти 2 рядки на таблицю з 1000 рядків. (5 розминкових запусків). Уповільнення процесора в 4 рази.	65.4±3.9	73.6±4.5	512.8±3.4	494.1±7.7
Видалення одного ряду. (5 розминкових запусків).	24.1±0.3	28.0±0.6	25.7±0.4	27.8±0.5
Створення 10 000 рядків. (5 розминкових прогонів з 1k рядків).	968.41±3.5	1,149.2±5.1	1,215.1±11.5	1,488.1±10.2

Додавання 1000 до таблиці з 10 000 рядків. 2х уповільнення процесора.	231.6±4.1	268.8±7.3	303.2±2.8	322.5±4.0
Очищення таблиці з 1000 рядків. Уповільнення процесора в 8 разів. (5 розминкових прогонів).	71.3±4.2	90.0±2.9	231.0±6.8	101.0±2.9
Постійно інтерактивний Песимістичний tti - коли процесор і мережа безумовно простоюють. (більше жодних завдань цп понад 50 мс)	1,879.7±0.7	2,104.84±9.6	2,783.1±0.4	2,554.8±0.2
Загальна вага в кілобайтах мережевої передачі (після стиснення) усіх ресурсів, завантажених на сторінку.	150.6±0.0	196.4±0.0	291.2±0.0	274.5±0.0

Фреймворк	vanillajs, МБайт	vue-v3.2.37, МБайт	angular-v13.0.0, МБайт	react-v17.0.2, МБайт
Використання пам'яті після завантаження сторінки.	1.0	1.3	2.0	1.4
Використання пам’яті після додавання 1000 рядків.	2.5	4.4	5.3	5.6
Використання пам’яті після 5	2.5	4.4	5.4	6.1

разів клацання оновлення кожного 10-го рядка				
Використання пам’яті після створення та очищення 1000 рядків 5 разів	1.0	1.6	2.7	2.2
Використання пам'яті після додавання 10 000 рядків.	16.4	30.5	32.6	39.5

4.3 Результати експерименту

Таблиця 4.1 – Результати експеременту

Питання	Angular	React	Vue
SQ1	Так, email	Hi	Так, email
SQ2	Hi	Hi	Hi
SQ3	Так	Hi	Так
SQ4	Так	Так	Так
MQ1	Так	Так	Так
MQ2	Hi	Так	Так
PQ1	3.1k	6.7k	6.3k
MSQ1	Microsoft, Autodesk, MacDonald’s, UPS, Cisco Solution Partner Program, AT&T, Apple, Adobe, GoPro,	Facebook, Instagram, Netflix, New York Times, Yahoo, Khan Academy, Whatsapp, Codecademy, Dropbox, Airbnb,	Xiaomi, Alibaba, WizzAir, EuroNews, Grammarly, Gitlab и Laracasts, Adobe, Behance, Codeship, Reuters

	ProtonMail, Clarity Design System, Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase.	Asana, Atlassian, Intercom, Microsoft, Slack, Storybook	
SUQ1	Немає даних	Немає даних	Немає даних
SUQ2	Так	Так	Так
SUQ3	Так	Так	Так
SUQ4	Так	Так	Так
SUQ5	ECMAScript	JSX	JSX
PCQ1	IE, Firefox, Chrome	IE, Firefox, Chrome	IE, Firefox, Chrome
PCQ2	Firefox 45, Safari 8, IE 11, Chrome 50	Firefox 45, Safari 8, IE 11, Chrome 50	Firefox 45, Safari 8, IE 11, Chrome 50
PCQ3	Firefox 45, Safari 8, IE 11, Chrome 50, Internet Explorer 9,10	Firefox 45, Safari 8, IE 11, Chrome 50, Internet Explorer 9,10	Firefox 45, Safari 8, IE 11, Chrome 50, Internet Explorer 9,10
PCQ4	ECMAScript 5	ECMAScript 5	ECMAScript 5
PCQ5	Так	Так	Так
PCQ6	Немає даних	Немає даних	Немає даних

CPQ1	154 kB	142 kB	
CPQ2	$cachefactory	Hi	Hi
TQ1	Так	Так	Так
TQ2	Так	Hi	Hi
TQ3	Так	Так	Так
TQ4	Hi	Hi	Hi
TQ5	Так	Так	Так
MaQ1	Так	Так	Так
MaQ2	Так	Так	Так
MaQ3	Google Inc.	Facebook Inc.	Hi
MaQ4	Так	Так	Так

Висновки до розділу 4

Порівняти фреймворки абсолютно об'єктивно досить складно. Але аналіз дозволить розробнику-початківцю підібрати платформу. Підведемо підсумки:

Якщо потрібно швидко вивчити середовище, варто вибирати між Vue.js і React.

На Vue.js легко перейти як користувачу Angular, так і React. Адже тут виходить чистий HTML-код, знайомий усім розробникам. Прийоми та техніки використовуються приблизно ті ж, що й у Angular.

Якщо передбачається розробка великого проекту, варто розглядати Angular як основу. Він забезпечує максимальну гнучкість та швидкість рендерингу. Величезний досвід інших розробників дозволить вирішити питання, які обов'язково виникнуть під час роботи над додатком. React виявиться занадто об'ємним, а для Vue.js ще немає великої кількості гайд-лайнів.

Якщо до розробки в майбутньому будуть залучатись інші програмісти, то Vue.js стане найкращим вибором. Адже цей фреймворк не тільки простий для вивчення, а й дозволяє міняти програму без руйнування його архітектури.

Якщо для проекту передбачається багатоступінчасте оновлення та розширення функціональності в майбутньому, то варто використовувати Vue.js або React через чудову зворотну сумісність.

Кожен із фреймворків гарний по-своєму, є свої сильні та слабкі сторони. Тому варто дати ще одну важливу пораду: що простіше вивчитиме команді програмістів, то й слід використовувати для розробки.

РОЗДІЛ 5

ТЕХНІКО-ЕКОНОМІЧНІ РОЗРАХУНКИ

Основна мета розробки техніко-економічного обґрунтування є надання фінансової оцінки передбачуваних витрат і отриманих корисних результатів, а також оцінка рентабельності проекту і в кінцевому підсумку економічної доцільності його розробки та реалізації.

Початковим етапом розрахунку витрат на оплату праці розробника є оцінка розміру програмного забезпечення. Основною відмінністю методів оцінки трудових витрат є тип використовуваних критеріїв оцінки якості.

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях.

$$E = a \cdot S^b \cdot EAF, \tag{5.1}$$

де E – витрати праці на проект (в людино-місяцях);

Sb – розмір коду (в KLOC);

EAF – фактор уточнення витрат (effort adjustment factor).

Для простих систем, a = 2,4; b = 1,05

За допомогою програми cloc [66] було виконано підрахунок розміру програмного коду. Цей додаток пропускає порожні рядки коду.

http://cloc.sourceforge.net v 1.53 T=8.0 s (202.4 files/s, 99198.6 lines/s)

Language	files	blank	comment	code
JavaScript	4	77	7	491
HTML	1	13	0	31
TS	2	34	12	89
Vue	1	58	8	135
SUM:	8	182	27	749

Рисунок 5.1 – Підрахунок розміру програмного коду

Звідси виходить, що розмір програмного коду становить 749 рядків:

$$E = 2{,}4 \cdot 0{,}749^{1{,}05} \cdot 1 = 1{,}77$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 1,77 людино-місяців.

Нижче подано розрахунок вартості розробки «Дослідження впливу React на фреймворки розробки Web- застосунків». Основними статтями витрат є:

– базова заробітна плата;
– відрахування на соціальні потреби;
– витрати на управління;
– вартість персонального комп'ютера та ліцензійних основних програмних засобів.

Базова заробітна плата (Basic salary, BSP) — це оцінка роботи, виконаної інженерами-програмістами при створенні програмних продуктів, і визначається відповідно до кількості розробників, часу розробки (годин) і погодинної оплати праці. Розрахунок заробітної плати здійснюється у вигляді таблиці. 5.1.

Таблиця 5.1 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати, грн
			чол	місяців	
1	інженер-програміст	16 621,58 [64]	1	1,75	29087,76

Програмний продукт, описаний у проекті, був розроблений програмістом у період з 26.09.21 по 20.11.21, що становить 45 днів або приблизно 9 робочих тижнів. Вартість робочого часу розрахована з розрахунку 40 годин на тиждень. Погодинна оплата праці кваліфікованого

інженера-програміста становить 103,88 грн./год. Отже, робочий час витрачається:

$$t_{\text{розробки}} = N_{\text{чол}} \times N_{\text{тиж}} \times N_{\text{год}}, \quad (5.2)$$

де $N_{\text{чол}}$ – кількість виконавців, чол;

$N_{\text{тиж}}$ – тривалість розробки;

$N_{\text{год}}$ – витрати робочого часу, год;

$$t_{\text{розробки}} = 1 \cdot 9 \cdot 40 = 360 \text{ чол/год.}$$

ОЗП визначається за формулою:

$$\text{ОЗП} = t_{\text{розробки}} \cdot N \cdot K_{KB}, \quad (5.3)$$

де tрозробки – витрати праці у чол/год;

N – погодинна ставка;

KKB – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. Коефіцієнт кваліфікації розробника (k) - ступінь підготовленості виконавця до дорученої йому роботі (він визначається залежність від стажу праці та становить:

- для працюючих до 2 років- 0,75;
- від 2 до 3 років 1,0;
- від 3 до 5 років - 1,1-1,2;
- від 5 до 7 років - 1,3-1,4;
- понад 7 років - 1,5-1,6.

В даному випадку KKB приймається 0,75. ОЗП складає:

$$\text{ОЗП} = 360 \cdot 103{,}88 \cdot 0{,}75 = 28047{,}6 \text{ грн.}$$

Відрахування на соціальні потреби встановлюються у відсотках від суми заробітної плати (22% [67]):

$$C_{\text{соц}} = \frac{\text{ОЗП} \cdot 22\%}{100\%} \tag{5.4}$$

$$C_{\text{соц}} = \frac{28047{,}6 \cdot 22\%}{100\%} = 6170{,}47\text{грн.}$$

Отримані результати за (5.3) та (5.4) підсумовуються. Вони складають 34218,07грн. та визначають основні прямі витрати.

Накладні витрати враховують загальногосподарські витрати на забезпечення виконання роботи: витрати на опалення, електроенергію, амортизацію будівель, заробітну плату адміністративного персоналу тощо. Вони визначаються у відсотках (30-40%) від суми прямих витрат:

$$\text{С}_{\text{накл}} = \frac{(\text{ОЗП} + \text{С}_{\text{соц}}) \cdot 35\%}{100\%}; \tag{5.5}$$

$$\text{С}_{\text{накл}} = \frac{(28047{,}6 + 6170{,}47) \cdot 35\%}{100\%} = 11976{,}32\text{грн.}$$

Протягом усього терміну експлуатації нового обладнання компанія щороку витрачає певну суму грошей, пов’язану з її діяльністю.

Експлуатаційна вартість персонального комп'ютера визначається при розробці програмного забезпечення в залежності від вартості комп'ютера. Операційні витрати включають:

- вартість витратних матеріалів;
- плата за ремонт;
- заробітна плата ремонтників;
- оренда приміщення;
- додаткові витрати - прибирання приміщення, охорона, оренда, комунальні послуги;

– амортизація персональних комп'ютерів і програмного забезпечення;
– плата за електроенергію ($С_{ел}$), визначається за такою формулою:

$$С_{ел} = P \cdot B \cdot T_{розр}, \quad (5.6)$$

де Р – потужність комп’ютера та допоміжних споживачів електричної енергії, приймається 0,5 кВт/год;

B – вартість 1 кВт/годин для непобутових споживачів,

складає 1,68 грн [3];

Tрозр – час роботи з ЕВМ, приймається рівним робочому часу

Витрати на електроенергію визначаються так:

$$C_{ел} = 0{,}5 \cdot 1{,}68 \cdot 360 = 302{,}4 \text{ грн.}$$

Вартість витратних матеріалів за весь термін експлуатації ($С_{вм}$) становить близько 10% вартості комп’ютера. Вартість АРМ передбачається 27 000 грн. [6], термін експлуатації 5 років. Отже, ці витрати можна визначити під час створення програмного засобу:

$$Свм = Вком \cdot NДNексп \cdot 365 \cdot 10\% 100\%,$$

де $B_{КОМ}$ – вартість персонального комп’ютеру;

$N_Д$ – кількість днів розробки програмного продукту;

$N_{експ}$ – термін експлуатації персонального комп’ютеру.

Витрати на витратні матеріали визначаються так:

$$C_{вм} = 27000 \cdot \frac{45}{5 \cdot 365} \cdot \frac{10}{100} = 66{,}57 \text{ грн.}$$

Зарплата ремонтника ($C_{\text{рем}}$) визначається таким чином: системний інженер потрібен для ремонту 50 комп'ютерів. Припустимо, що його середньомісячна заробітна плата становить 10 000 грн. Тоді, виходячи з комп'ютера, його зарплата під час розробки програмного продукту становить:

$$Срем=СремNKOM \cdot Tміс,$$

де $C_{\text{рем}}^{\prime}$ – середньомісячна заробітна плата;

N_{KOM} – кількість комп'ютерів на одного ремонтника.

$T_{міс}$ – час розробки програмного продукта, міс.

Заробітна плата ремонтника ($C_{\text{рем}}$) буде складати:

$$C_{\text{рем}} = \frac{10000}{50} \cdot 1{,}75 = \ 350 \text{ грн.}$$

За статистикою витрати на комплектуючи вироби (С$_{KOM}$) для ремонту персонального комп'ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$C_{KOM} = C_{BM} = 66{,}57 \text{ грн.} \qquad (5.9)$$

Амортизаційні відрахування для персональних комп'ютерів (АПК) базуються на припущенні, що період амортизації на даний момент дорівнює етичному періоду виходу з експлуатації комп'ютерного обладнання та становить 3 роки. Отже, за 3 роки амортизаційні відрахування ПК дорівнюють вартості комп'ютера. Під час проектування амортизаційні відрахування складатимуть:

=

$$АКП = 27000 \cdot \frac{45}{3 \cdot 365} = 1109{,}6$$

Амортизаційні відрахування на програмне забезпечення (АПЗ) залежать від циклу його заміни. Якщо прийняти термін морального старіння 5 років для Windows і 2 роки для Visual Studio, то амортизаційні відрахування для програмного забезпечення дорівнюють його вартості.

На персональному комп'ютері використовується операційна система Windows 10, а програмне забезпечення написано за допомогою програмного середовища Visual Studio Code.

$$АКП_w = 7\,899 \cdot \frac{45}{5 \cdot 365} = 194{,}7$$

$$АКП_w = 0 \cdot \frac{45}{2 \cdot 365} = 0$$

Розрахунок амортизаційних відрахувань ПЗ зведено в таблицю. 5.2 Додаткові витрати ($C_{доД}$): прибирання приміщень, охорона, комунальні послуги важко оцінити точно, що еквівалентно 50% зарплати інженерів-програмістів, що становить 8310,8 гривень на місяць.

Оренда житла на одну людину становитиме 2700 гривень на місяць [5]. Тобто за весь період розробки – 4725 грн. Загальна вартість експлуатації ПК становить:

$$C_{експ} = C_{ел} + C_{ВМ} + C_{рем} + АПК + АПО + C_{ор} + C_{доД}; \qquad (5.11)$$

$C_{\text{експ}} = 302{,}4 + 66{,}57 + 350 + 1109{,}6 + 194{,}7 + 4725 + 8310{,}8 = 15059{,}07$ грн

Результати розрахунків зведено у табл. 5.3.

Таблиця 5.2 – Використовуване програмне забезпечення

Найменування програмного забезпечення	Вартість програмного забезпечення, грн	Джерело придбання	Амортизаційні відрахування, грн
Windows 10	7 899	https://www.microsoft.com/uk-ua/d/windows-10-pro/df77x4d43rkt?activetab=pivot:%D0%BE%D0%B3%D0%BB%D1%8F%D0%B4tab	194,7
Visual Studio Code	0	https://visualstudio.microsoft.com/	0
Всього:	7 899		377,2

Таблиця 5.3 – Експлуатаційні витрати на ПК і ПЗ.

Найменування витрат	Витрати, грн
Витрати на електроенергію	302
Вартість витратних матеріалів	66
Витрати на ремонт	350
Амортизація персонального комп’ютера	1109
Амортизація програмного забезпечення	194

Найменування витрат	Витрати, грн
Основна заробітна плата	
Відрахування на соціальні потреби	
Накладні витрати	
Експлуатаційні витрати	15059
Всього	

Оренда приміщення	4725
Додаткові витрати	8310
Всього	15059

Таким чином, витрати на створення програмного продукту складають:

$$C_{\text{розробки}} = \text{ОЗП} + C_{\text{соц}} + C_{\text{накл}} + C_{\text{експ}}; \qquad (5.12)$$

$$C_{\text{розробки}} = 28047 + 6170 + 11976 + 15059 = 61252 \text{ грн.}$$

Розрахунок витрат зведено у табл. 5.4.

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, грн
Основна заробітна плата	28047
Відрахування на соціальні потреби	6170
Накладні витрати	11976
Експлуатаційні витрати	15059
Всього	61252

Висновки до розділу 5

Найменування витрат	Витрати, грн
Основна заробітна плата	28047
Відрахування на соціальні потреби	6170
Накладні витрати	11976
Експлуатаційні витрати	15059
Всього	61252

На основі отриманих значень техніко-економічних показників проекту оцінюється вартість розробки Web-застосунків. За розрахунками, вартість розробки становить близько 61252 грн.

ВИСНОВКИ

Представлено концепції, які дозволили Facebook створити React. Ви дізналися, як концепції React зазвичай розглядаються як такі, що відходять від загальноприйнятих найкращих практик розробки інтерфейсу користувача. Виклик статус-кво та теорії тестування дозволили React стати високопродуктивним і масштабованим фреймворком JavaScript для створення інтерфейсів користувача.

ReactJS — це продуктивна, ієрархічна та функціональна технологія, яка має низку переваг. Фреймворк пропонує широкий контекст із такими можливостями, як тривимірні зображення, загальні елементи, одностороннє інформаційне з'єднання, тригери та JSX.

Angular є найбільш зрілим фреймворком, має хорошу підтримку з точки зору учасників і є повним пакетом.

Однак крива навчання є стрімкою, і концепції розробки в Angular можуть відштовхнути нових розробників.

Angular — хороший вибір для компаній із великими командами та розробників, які вже використовують TypeScript.

React достатньо дорослий, щоб бути зрілим, і має величезну кількість внесків від спільноти. Він отримав широке визнання. Ринок праці для React справді хороший, і майбутнє для цього фреймворку виглядає яскравим.

React виглядає як хороший вибір для тих, хто починає працювати з інтерфейсними фреймворками JavaScript, стартапів і розробників, яким подобається гнучкість. Можливість легкої інтеграції з іншими фреймворками дає велику перевагу для тих, хто хоче певної гнучкості у своєму коді.

Vue є найновішим на арені, без підтримки великої компанії.

Однак за останні кілька років він справді успішно виступив як сильний конкурент для Angular і React, особливо з випуском Vue 3.0. Можливо, це відіграє свою роль у тому, що багато китайських гігантів, таких як Alibaba та

Baidu, обирають Vue як свою основну інтерфейсну структуру JavaScript. Vue має бути вашим вибором, якщо ви віддаєте перевагу простоті, але також любите гнучкість.

Щоб обмежити кількість критеріїв, вибираються лише критерії з двома або більше посиланнями. Стабільність і зрілість пов’язані, оскільки зрілість програмного забезпечення впливає на стабільність, тому вони об’єднані за одним критерієм. Щоб мати можливість використовувати фреймворк SPA в якомога більшій кількості браузерів, сумісність і портативність є важливими, тому ці два критерії згруповані разом. У інформатиці постійність часто стосується збереження поточного стану програми в пам’яті. Щоб зберегти стан SPA в пам’яті, потрібна хороша продуктивність кешу. Це призводить до групування стійкості та продуктивності кешу разом. Зручність використання може бути визначена стандартом ISO 9241-11, який охоплює сферу, де користувач повинен бути задоволений використанням програмного забезпечення [18]. Це добре поєднується з простотою, тому що якщо щось є простим у використанні, воно повинно мати високу зручність використання, навпаки, те, що непросто у використанні, має мати нижчий рівень зручності використання. Тому простота і зручність у використанні згруповані. Документація, розмір фреймворку та внутрішній досвід вибираються як запитання для використання відповідно до відповідних критеріїв.

Порівняти фреймворки абсолютно об'єктивно досить складно. Але аналіз дозволить розробнику-початківцю підібрати платформу. Підведемо підсумки:

Якщо потрібно швидко вивчити середовище, варто вибирати між Vue.js і React.

На Vue.js легко перейти як користувачу Angular, так і React. Адже тут виходить чистий HTML-код, знайомий усім розробникам. Прийоми та техніки використовуються приблизно ті ж, що й у Angular.

Якщо передбачається розробка великого проекту, варто розглядати Angular як основу. Він забезпечує максимальну гнучкість та швидкість рендерингу. Величезний досвід інших розробників дозволить вирішити питання, які обов'язково виникнуть під час роботи над додатком. React виявиться занадто об'ємним, а для Vue.js ще немає великої кількості гайд-лайнів.

Якщо до розробки в майбутньому будуть залучатись інші програмісти, то Vue.js стане найкращим вибором. Адже цей фреймворк не тільки простий для вивчення, а й дозволяє міняти програму без руйнування його архітектури.

Якщо для проекту передбачається багатоступінчасте оновлення та розширення функціональності в майбутньому, то варто використовувати Vue.js або React через чудову зворотну сумісність.

Кожен із фреймворків гарний по-своєму, є свої сильні та слабкі сторони. Тому варто дати ще одну важливу пораду: що простіше вивчитиме команді програмістів, то й слід використовувати для розробки. Також вивчайте доступні матеріали, розвивайтеся, не стійте на місці, адже Vue.js та React постійно змінюються.

Адже час, витрачений вивчення специфіки фреймворку, відсуває випуск докладання у комерційне використання на невизначений термін. При цьому техпідтримка реалізованого проекту може виявитися непосильним тягарем, який «розжене» штатних програмістів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. AngularJS: Security. https://docs.angularjs.org/guide/security, 2016. Accessed: 2016-04-05.
2. BenchmarkJS. https://benchmarkjs.com/, 2016. Accessed: 2016-04-05.
3. EmberJS: Security. http://emberjs.com/security/, 2016. Accessed: 2016-04-05.
4. OWASP. https://www.owasp.org/index.php/Main_Page, 2016. Accessed: 2016-04-05.
5. React: JSX Gotchas. https://facebook.github.io/react/docs/jsx-gotchas.html, 2016. Accessed: 2016-04-05.
6. TodoMVC. https://www.todomvc.com/, 2016. Accessed: 2016-04-22.
7. AngularJS and community. Data Binding. https://docs.angularjs.org/guide/databinding, 2016. Accessed: 2016-04- 05.
8. Len Bass, Paul Clements, and Rick Kazman. Software Architecture in Practice. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edition, 2003.
9. PerOlof Bengtsson, Nico Lassing, Jan Bosch, and Hans van Vliet. Architecture level modifiability analysis (alma). Journal of Systems and Software, 69(1–2):129 – 147, 2004.
10. Klaus Bergner, Andreas Rausch, Marc Sihling, and Thomas Ternité. Dosam – domain-specific software architecture comparison model. In Proceedings of the First International Conference on Quality of Software Architectures and Software Quality, and Proceedings of the Second International Conference on Software Quality, QoSA'05, pages 4–20, Berlin, Heidelberg, 2005. SpringerVerlag.
11. Mathias Bynens and John-David Dalton. Bulletproof JavaScript benchmarks. http://calendar.perfplanet.com/2010/bulletproof-javascript-benchmarks/, 2010. Accessed: 2016-04-05.
12. Jing Chen. Hacker Way: Rethinking Web App Development at Facebook. https://www.youtube.com/watch?v=nYkdrAPrdcw, 2014.

13. Paul C. Clements. The architecture tradeoff analysis method. Technical Report CMU/SEI-2000-TN-009, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2000.

14. J. Collis and R. Hussey. Business Research: A Practical Guide for Undergraduate and Postgraduate Students. Palgrave Macmillan, 2009.

15. María del Pilar Salas-Zárate, Giner Alor-Hernández, Rafael Valencia-García, Lisbeth Rodríguez-Mazahua, Alejandro Rodríguez-González, and José Luis López Cuadrado. Analyzing best practices on web development frameworks: The lift approach. Science of Computer Programming, 102:1 – 19, 2015.

16. T.J. Dolan. Architecture Assessment of Information-system Families: A Practical Perspective. Technische Universiteit Eindhoven, 2001.

17. Rolf Ejvegård. Vetenskaplig metod. Studentlitteratur, 2009.

18. International Organization for Standardization. Iso 9241-11:1998, 1998.

19. M. Fowler. Patterns of Enterprise Application Architecture. A Martin Fowler signature book. Addison-Wesley, 2003.

20. Anton Gerdessen. Framework comparison method: Comparing two frameworks based on technical domains, focussing on customisability and modifiability. Master's thesis, UvA, University of Amsterdam, 2007.

21. José Ignacio Fernández-Villamor, Laura Díaz-Casillas, and Carlos Á. Iglesias. A comparison model for agile web frameworks. In Proceedings of the 2008 Euro American Conference on Telematics and Information Systems, EATIS '08, pages 14:1–14:8, New York, NY, USA, 2008. ACM.

22. Apple Inc. JavaScriptCore. http://trac.webkit.org/wiki/JavaScriptCore, 2016. Accessed: 2016-04-22.

23. Facebook Inc. Overview. https://facebook.github.io/flux/docs/overview.html #content, 2016. Accessed: 2016-04-05.

24. Facebook Inc. React (Virtual) DOM Terminology. https://facebook.github.io/react/docs/glossary.html, 2016. Accessed: 2016- 04-22.

25. Facebook Inc. Reconciliation. https://facebook.github.io/react/docs/reconciliation.html, 2016. Accessed: 2016-05-22.

26. Google Inc. AngularJS. https://angularjs.org/, 2016. Accessed: 2016-04-22. 40

27. Google Inc. AngularJS: Unit Testing. https://docs.angularjs.org/guide/unit-testing, 2016. Accessed: 2016-04-22.

28. Google Inc. ChangeDetectorRef. https://angular.io/docs/ts/latest/api/core/ ChangeDetectorRef-class.html, 2016. Accessed: 2016-05-22.

29. Google Inc. Chrome Reward Program Rules. https://www.google.com/about/appsecurity/chrome-rewards/, 2016. Accessed: 2016-04-22.

30. Google Inc. Chrome V8. https://developers.google.com/v8/, 2016. Accessed: 2016-04-22.

31. Google Inc. Features & Benefits. https://angular.io/features.html, 2016. Accessed: 2016-04-22.

32. Google Inc. What are Scopes? https://docs.angularjs.org/guide/scope, 2016. Accessed: 2016-05-22.

33. Software Engineering Institute. Cost Benefit Analysis Method. http://www.sei.cmu.edu/architecture/tools/evaluate/cbam.cfm. Accessed: 2016-04-05.

34. Prashant Bansode Scott Barber J.D. Meier, Carlos Farre and Dennis Rea. Performance Testing Guidance for Web Applications: Chapter 1 – Fundamentals of Web Application Performance Testing. Microsoft Press, 2007.

35. Prashant Bansode Scott Barber J.D. Meier, Carlos Farre and Dennis Rea. Reusing Open Source Code. Microsoft Press, 2011.

36. Ralph E. Johnson. Components, frameworks, patterns. In Proceedings of the 1997 Symposium on Software Reusability, SSR ’97, pages 10–17, New York, NY, USA, 1997. ACM.

37. Max Kanat-Alexander. Code Simplicity. O’Reilly Media, 2012.

38. Rick Kazman, Len Bass, Mike Webb, and Gregory Abowd. Saam: A method for analyzing the properties of software architectures. In Proceedings of the 16th International Conference on Software Engineering, ICSE ’94, pages 81–90, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

39. Rick Kazman, Mark Klein, Mario Barbacci, Thomas Longstaff, Howard Lipson, and S. Carriere. Active reviews for intermediate designs. Technical Report CMU/SEI-98-TR-008, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1998.

40. Joe Lennon. Compare JavaScript frameworks. http://www.ibm.com/developerworks/library/wa-jsframeworks/, 2010. Accessed: 2016-04-05.

41. Tim Johan Malmström. Structuring modern web applications: A study of how to structure web clients to achieve modular, maintainable and longlived applications. Master’s thesis, KTH, School of Computer Science and Communication (CSC), 2014.

42. Mesbah and A. van Deursen. Migrating multi-page web applications to single-page ajax interfaces. In Software Maintenance and Reengineering, 2007. CSMR ’07. 11th European Conference on, pages 181–190, March 2007.

43. Mozilla. SpiderMonkey. https://developer.mozilla.org/en-US/docs/Mozilla/ Projects/SpiderMonkey, 2016. Accessed: 2016-04-22.

44. S. Murugesan. Understanding web 2.0. IT Professional, 9(4):34–41, July 2007.

45. Jakob Nielsen. Usability Engineering. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

46. M.Q. Patton. Qualitative Research & Evaluation Methods. SAGE Publications, 2002.

47. Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber. Capability maturity model, version 1.1. IEEE Softw., 10(4):18–27, July 1993.

48. Dr. Axel Rauschmayer. Chapter 4. How JavaScript Was Created. http://speakingjs.com/es5/ch04.html, 2016. Accessed: 2016-04-05.

49. Dr. Axel Rauschmayer. Chapter 5. Standardization: ECMAScript. http://speakingjs.com/es5/ch05.html, 2016. Accessed: 2016-04-05.

50. Dirk Riehle. Framework design a role modeling approach. Diss. ETH No. 13509, 2000.

51. F. Rosene, J. E. Connolly, and K. M. Bracy. Software maintainability - what it means and how to achieve it. IEEE Transactions on Reliability, R30(3):240–245, Aug 1981.

52. Ioannis Samoladas, Ioannis Stamelos, Lefteris Angelis, and Apostolos Oikonomou. Open source software development should strive for even greater code maintainability. Commun. ACM, 47(10):83–87, October 2004.

53. Douglas C. Schmidt and Frank Buschmann. Patterns, frameworks, and middleware: Their synergistic relationships. In Proceedings of the 25th International Conference on Software Engineering, ICSE ’03, pages 694–704, Washington, DC, USA, 2003. IEEE Computer Society.

54. Meg Sewell. The use of qualitative interviews in evaluation. http://ag.arizona.edu/sfcs/cyfernet/cyfar/Intervu5.htm, 2016. Accessed: 2016-04-05. 42

55. T. C. Shan and W. W. Hua. Taxonomy of java web application frameworks. In e-Business Engineering, 2006. ICEBE ’06. IEEE International Conference on, pages 378–385, Oct 2006.

56. StatCounter Global Stats. Top 12 Desktop Browser Versions Combining Chrome and Firefox (5+) from Apr 2015 to Apr 2016.

http://gs.statcounter.com/#desktop-browser_version_partially _combined-ww-monthly-201504-201604, 2016. Accessed: 2016-05-02.

57. Christoph Stoermer, Felix Bachmann, and Chris Verhoef. Sacam: The software architecture comparison analysis method. Technical Report CMU/SEI-2003- TR-006, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2003.

58. Mikito Takada. Single page apps in depth. http://singlepageappbook.com/goal.html, 2016. Accessed: 2016-04-05.

59. Microsoft Patterns & Practices Team. Microsoft Application Architecture Guide, 2nd Edition. Chapter 1: What is Software Architecture? https://msdn.microsoft.com/en-us/library/ee658098.aspx, 2009. Accessed: 2016-04-05.

60. Microsoft Patterns & Practices Team. Microsoft Application Architecture Guide, 2nd Edition. Chapter 4: A Technique for Architecture and Design. https://msdn.microsoft.com/en-us/library/ee658084.aspx, 2009. Accessed: 2016-04-05.

61. TodoMVC. TodoMVC GitHub Repository. https://github.com/tastejs/todomvc, 2016. Accessed: 2016-05-22.

62. Arthur H. Watson and Thomas J. McCabe. Structured testing: A testing methodology using the cyclomatic complexity metric. Technical report, Computer Systems Laboratory, National Institute of Standards and Technology. http://www.mccabe.com/pdf/mccabe-nist235r.pdf.

63. Борисенков, Методика оценки трудозатрат по разработке программного обеспечения информационных систем [Ел. ресурс] /Борисенков Евгений //URL: https://dspace.enu.kz/bitstream/handle/data/12881/METODIKATRUDOZATRAT.pdf?sequence=1&isAllowed=y.

64. «Зарплати розробників за червень-липень 2021» [Ел. ресурс] URL: https://jobs.dou.ua/salaries/#period=jun2021&city=Dnipro&title=Junior%2

0Software%20Engineer&language=C%23%2F.NET&spec=&exp1=0&exp2=1?period=2022-06&position=Intern/Trainee%20SE

65. «Мінестрество енергетики України» [Ел. ресурс] URL: https://yasno.com.ua/b2c-tariffs
66. «Розміри ЄСВ-2021» [Ел. ресурс] URL: https://services.dtkt.ua/catalogues/indexes/
67. «Аренда офисов в Днепре» [Ел. ресурс] URL: https://besplatka.ua/ru/dnepropetrovsk/nedvizhimost/arenda-komercheskoynedvizhimosti/arendaofisov?prop[136][to]=2500&prop[112][to]=10&currency=
68. «Комп'ютери та ноутбуки» [Ел. ресурс] URL: https://rozetka.com.ua/ua/362083440/p362083440/

Додатки

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
_____ Анатолій
РАДКЕВИЧ

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОРІНЯННЯ ФРЕЙМОРКІВ JAVASCRIPT

Технічне завдання
ЛИСТ ЗАТВЕДЖЕННЯ
44165850.00001-01

Завідувач кафедри КІТ
_____ доц. Вадим ГОРЯЧКІН

Керівник розробки
____ доц. Вадим АНДРЮЩЕНКО

Виконавець
____ Владислав ТРИПУТИН

Нормоконтролер
____ доц. Світлана ВОЛКОВА

2022

ЗАТВЕРДЖЕНО

44165850.00001-01-ЛЗ

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОРІНЯННЯ ФРЕЙМОРКІВ JAVASCRIPT

ТЕХНІЧНЕ ЗАВДАННЯ

44165850.00001-01

Листів 22

2022

АНОТАЦІЯ

Документ 01116130.01211-01 «Програмне забезпечення порівняння фреймворків JavaScript. Технічне завдання» входить до складу програмної документації до програми, яка реалізує програмне забезпечення.

У даному документі представлені призначення та область застосування програмної системи, вимоги, стадії, строки виконання проекту та техніко-економічні відомості.

ЗМІСТ

ВСТУП

Назва розробки: Програмне забезпечення порівняння фреймворків JavaScript

Галузь застосування: робота користувача з single page applicaton.

Наведене технічне завдання поширюється на розробку програмного забезпечення порівняння різних фреймворків, котра використовується для порівняння швидкодії фреймворків при виконанні різних операцій.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмного забезпечення аналізу фреймворків є наказ від «02» листопада 2022 р. № 1113 в.о. ректора Українського державного університету науки і технологій Олександра Пшінька . Згідно наказу призначено тему «Дослідження впливу React на фреймворки розробки Web-застосунків.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Функціональне призначення

Функціональним призначенням розробки є показати на прикладі час та навантаження системи під час роботи різних фреймворків.

2.2 Експлуатаційне призначення

Експлуатаційне призначення полягає у можливості надати користувачу зробити вибір який фреймворк використовувати

3 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції для користувача:

– порівнювати вибрані фреймворки;

– виводити метрики швидкодії виконання операцій.

Розробку виконати на платформі Windows 10.

3.2 Вимоги до надійності

3.2.1 Передбачити контроль безпеки.

3.2.2Передбачити захист від некоректних дій користувача.

3.3 Умови експлуатації

Не висуваються

3.4 Вимоги до складу і параметрів технічних засобів

3.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

3.5 Вимоги до інформаційної та програмної сумісності

3.5.1 Програмне забезпечення повинно працювати під управлінням найбільш поширених браузерів.

3.5.2 Результати повинні бути представлені в наступному форматі: число у вигляді результату з можливим відхиленням.

3.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

3.8 Спеціальні вимоги

Спеціальні вимоги не пред'являються.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

1.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

1.2 Програмне забезпечення повинно мати довідникову систему

1.3 У склад супроводжувальної документації повинні входити наступні документи:

3.3.1 Пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків).

3.4 Технічне завдання.

3.5 Програма.

5 ВИЗНАЧЕННЯ ВИТРАТ НА ПРОЕКТУВАННЯ ПРОГРАМИ

Основна мета розробки техніко-економічного обґрунтування є надання фінансової оцінки передбачуваних витрат і отриманих корисних результатів, а також оцінка рентабельності проекту і в кінцевому підсумку економічної доцільності його розробки та реалізації.

Початковим етапом розрахунку витрат на оплату праці розробника є оцінка розміру програмного забезпечення. Основною відмінністю методів оцінки трудових витрат є тип використовуваних критеріїв оцінки якості.

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях.

$$E = a \cdot S^b \cdot EAF, \tag{5.1}$$

де E – витрати праці на проект (в людино-місяцях);

Sb – розмір коду (в KLOC);

EAF – фактор уточнення витрат (effort adjustment factor).

Для простих систем, a = 2,4; b = 1,05

За допомогою програми cloc [3] було виконано підрахунок розміру програмного коду. Цей додаток пропускає порожні рядки коду.

http://cloc.sourceforge.net v 1.53 T=8.0 s (202.4 files/s, 99198.6 lines/s)

Language	files	blank	comment	code
JavaScript	4	77	7	491
HTML	1	13	0	31
TS	2	34	12	89
Vue	1	58	8	135
SUM:	8	182	27	749

Рисунок 5.1 – Підрахунок розміру програмного коду

Звідси виходить, що розмір програмного коду становить 749 рядків:

$$E = 2{,}4 \cdot 0{,}749^{1{,}05} \cdot 1 = 1{,}77$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 1,77 людино-місяців.

Нижче подано розрахунок вартості розробки «Дослідження впливу React на фреймворки розробки Web- застосунків». Основними статтями витрат є:

– базова заробітна плата;
– відрахування на соціальні потреби;
– витрати на управління;
– вартість персонального комп'ютера та ліцензійних основних програмних засобів.

Базова заробітна плата (Basic salary, BSP) — це оцінка роботи, виконаної інженерами-програмістами при створенні програмних продуктів, і визначається відповідно до кількості розробників, часу розробки (годин) і погодинної оплати праці. Розрахунок заробітної плати здійснюється у вигляді таблиці. 5.1.

Таблиця 5.1 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати, грн
			чол	місяців	
1	інженер-програміст	16 621,58 [2]	1	1,75	29087,76

Програмний продукт, описаний у проекті, був розроблений програмістом у період з 26.09.21 по 20.11.21, що становить 45 днів або приблизно 9 робочих тижнів. Вартість робочого часу розрахована з розрахунку 40 годин на тиждень. Погодинна оплата праці кваліфікованого інженера-програміста становить 103,88 грн./год. Отже, робочий час витрачається:

$$t_{\text{розробки}} = N_{\text{чол}} \times N_{\text{тиж}} \times N_{\text{год}}, \quad (5.2)$$

де $N_{\text{чол}}$ – кількість виконавців, чол;

$N_{\text{тиж}}$ – тривалість розробки;

$N_{\text{год}}$ – витрати робочого часу, год;

$$t_{\text{розробки}} = 1 \cdot 9 \cdot 40 = 360 \text{ чол/год}.$$

ОЗП визначається за формулою:

$$\text{ОЗП} = t_{\text{розробки}} \cdot N \cdot K_{KB}, \quad (5.3)$$

де tрозробки – витрати праці у чол/год;

N – погодинна ставка;

KKB – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. Коефіцієнт кваліфікації розробника (k) - ступінь підготовленості виконавця до дорученої йому роботі (він визначається залежність від стажу праці та становить:

- для працюючих до 2 років- 0,75;
- від 2 до 3 років 1,0;
- від 3 до 5 років - 1,1-1,2;
- від 5 до 7 років - 1,3-1,4;
- понад 7 років - 1,5-1,6.

В даному випадку KKB приймається 0,75. ОЗП складає:

$$\text{ОЗП} = 360 \cdot 103{,}88 \cdot 0{,}75 = 28047{,}6 \text{ грн}.$$

Відрахування на соціальні потреби встановлюються у відсотках від суми заробітної плати (22% [4]):

$$C_{\text{соц}} = \frac{\text{ОЗП} \cdot 22\%}{100\%} \tag{5.4}$$

$$C_{\text{соц}} = \frac{28047{,}6 \cdot 22\%}{100\%} = 6170{,}47\text{грн.}$$

Отримані результати за (5.3) та (5.4) підсумовуються. Вони складають 34218,07грн. та визначають основні прямі витрати.

Накладні витрати враховують загальногосподарські витрати на забезпечення виконання роботи: витрати на опалення, електроенергію, амортизацію будівель, заробітну плату адміністративного персоналу тощо. Вони визначаються у відсотках (30-40%) від суми прямих витрат:

$$\text{С}_{\text{накл}} = \frac{(\text{ОЗП} + \text{С}_{\text{соц}}) \cdot 35\%}{100\%}; \tag{5.5}$$

$$\text{С}_{\text{накл}} = \frac{(28047{,}6 + 6170{,}47) \cdot 35\%}{100\%} = 11976{,}32\text{грн.}$$

Протягом усього терміну експлуатації нового обладнання компанія щороку витрачає певну суму грошей, пов’язану з її діяльністю.

Експлуатаційна вартість персонального комп'ютера визначається при розробці програмного забезпечення в залежності від вартості комп'ютера. Операційні витрати включають:

- вартість витратних матеріалів;
- плата за ремонт;
- заробітна плата ремонтників;
- оренда приміщення;
- додаткові витрати - прибирання приміщення, охорона, оренда, комунальні послуги;
- амортизація персональних комп'ютерів і програмного забезпечення;

– плата за електроенергію ($С_{ел}$), визначається за такою формулою:

$$С_{ел} = P \cdot B \cdot T_{розр}, \quad (5.6)$$

де Р – потужність комп'ютера та допоміжних споживачів електричної енергії, приймається 0,5 кВт/год;

B – вартість 1 кВт/годин для непобутових споживачів,

складає 1,68 грн [3];

Tрозр – час роботи з ЕВМ, приймається рівним робочому часу

Витрати на електроенергію визначаються так:

$$C_{ел} = 0{,}5 \cdot 1{,}68 \cdot 360 = 302{,}4 \text{ грн.}$$

Вартість витратних матеріалів за весь термін експлуатації (C_vm) становить близько 10% вартості комп'ютера. Вартість АРМ передбачається 27 000 грн. [6], термін експлуатації 5 років. Отже, ці витрати можна визначити під час створення програмного засобу:

$$Свм = Вком \cdot NДNексп \cdot 365 \cdot 10\% 100\%,$$

де $B_{КОМ}$ – вартість персонального комп'ютеру;

$N_Д$ – кількість днів розробки програмного продукту;

$N_{експ}$ – термін експлуатації персонального комп'ютеру.

Витрати на витратні матеріали визначаються так:

$$C_{вм} = 27000 \cdot \frac{45}{5 \cdot 365} \cdot \frac{10}{100} = 66{,}57 \text{ грн.}$$

Зарплата ремонтника ($C_{рем}$) визначається таким чином: системний інженер потрібен для ремонту 50 комп'ютерів. Припустимо, що його середньомісячна заробітна плата становить 10 000 грн. Тоді, виходячи з комп'ютера, його зарплата під час розробки програмного продукту становить:

$$Срем=СремNKOM \cdot Тміс,$$

де $C_{\text{рем}}^{\acute{}}$ – середньомісячна заробітна плата;

N_{KOM} – кількість комп'ютерів на одного ремонтника.

$T_{міс}$ – час розробки програмного продукта, міс.

Заробітна плата ремонтника ($C_{\text{рем}}$) буде складати:

$$C_{\text{рем}} = \frac{10\acute{0}00}{50} \cdot 1{,}75 = \ 350 \text{ грн.}$$

За статистикою витрати на комплектуючи вироби (С$_{KOM}$) для ремонту персонального комп'ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$C_{KOM} = C_{BM} = 66{,}57 \text{ грн.} \qquad (5.9)$$

Амортизаційні відрахування для персональних комп'ютерів (АПК) базуються на припущенні, що період амортизації на даний момент дорівнює етичному періоду виходу з експлуатації комп'ютерного обладнання та становить 3 роки. Отже, за 3 роки амортизаційні відрахування ПК дорівнюють вартості комп'ютера. Під час проектування амортизаційні відрахування складатимуть:

=

$$\text{АКП} = 27000 \cdot \frac{45}{3 \cdot 365} = 1109{,}6$$

Амортизаційні відрахування на програмне забезпечення (АПЗ) залежать від циклу його заміни. Якщо прийняти термін морального старіння 5 років для Windows і 2 роки для Visual Studio, то амортизаційні відрахування для програмного забезпечення дорівнюють його вартості.

На персональному комп'ютері використовується операційна система Windows 10, а програмне забезпечення написано за допомогою програмного середовища Visual Studio Code.

$$\text{АКП}_w = 7\,899 \cdot \frac{45}{5 \cdot 365} = 194{,}7$$

$$\text{АКП}_w = 0 \cdot \frac{45}{2 \cdot 365} = 0$$

Розрахунок амортизаційних відрахувань ПЗ зведено в таблицю. 5.2 Додаткові витрати ($C_{\text{дод}}$): прибирання приміщень, охорона, комунальні послуги важко оцінити точно, що еквівалентно 50% зарплати інженерів-програмістів, що становить 8310,8 гривень на місяць.

Оренда житла на одну людину становитиме 2700 гривень на місяць [5]. Тобто за весь період розробки – 4725 грн. Загальна вартість експлуатації ПК становить:

$$C_{\text{експ}} = C_{\text{ел}} + C_{\text{ВМ}} + C_{\text{рем}} + \text{АПК} + \text{АПО} + C_{\text{ор}} + C_{\text{дод}}; \qquad (5.11)$$

$$C_{\text{експ}} = 302{,}4 + 66{,}57 + 350 + 1109{,}6 + 194{,}7 + 4725 + 8310{,}8 = 15059{,}07 \text{ грн}$$

Результати розрахунків зведено у табл. 5.3.

Таблиця 5.2 – Використовуване програмне забезпечення

Найменування програмного забезпечення	Вартість програмного забезпечення, грн	Джерело придбання	Амортизаційні відрахування, грн
Windows 10	7 899	https://www.microsoft.com/uk-ua/d/windows-10-pro/df77x4d43rkt?activetab=pivot:%D0%BE%D0%B3%D0%BB%D1%8F%D0%B4tab	194,7
Visual Studio Code	0	https://visualstudio.microsoft.com/	0
Всього:	7 899		377,2

Таблиця 5.3 – Експлуатаційні витрати на ПК і ПЗ.

Найменування витрат	Витрати, грн
Витрати на електроенергію	302
Вартість витратних матеріалів	66
Витрати на ремонт	350
Амортизація персонального комп’ютера	1109
Амортизація програмного забезпечення	194
Оренда приміщення	4725
Додаткові витрати	8310
Всього	15059

Таким чином, витрати на створення програмного продукту складають:

Найменування витрат	Витрати, грн
Основна заробітна плата	28047
Відрахування на соціальні потреби	6170
Накладні витрати	11976
Експлуатаційні витрати	15059
Всього	61252

$$C_{\text{розробки}} = \text{ОЗП} + C_{\text{соц}} + C_{\text{накл}} + C_{\text{експ}}; \qquad (5.12)$$

$$C_{\text{розробки}} = 28047 + 6170 + 11976 + 15059 = 61252 \text{ грн.}$$

Розрахунок витрат зведено у табл. 5.4.

Найменування витрат	Витрати, грн
Основна заробітна плата	28047
Відрахування на соціальні потреби	6170
Накладні витрати	11976
Експлуатаційні витрати	15059
Всього	61252

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

На основі отриманих значень техніко-економічних показників проекту оцінюється вартість розробки Web-застосунків. За розрахунками, вартість розробки становить близько 61252 грн.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу
1	Вивчення рекомендованої літератури
2	Аналіз існуючих методів розв’язання задачі
3	Постановка та формалізація задачі
4	Аналіз вимог до програмного забезпечення
5	Алгоритмізація задачі
6	Моделювання програмного забезпечення
7	Обґрунтування використовуваних технічних засобів
8	Розробка архітектури програмного забезпечення
9	Розробка програмного забезпечення
10	Налагодження програми
11	Оформлення пояснювальної записки

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Контроль якості програмного продукту здійснюється за допомогою виконання процесу тестування з метою знаходження помилок та їх ліквідації.

Контроль за виконанням роботи здійснюється науковим керівником дипломної роботи.

Прийом програмного продукту та роботи здійснюється уповноваженою комісією.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Борисенков, Методика оценки трудозатрат по разработке программного обеспечения информационных систем [Ел. ресурс] /Борисенков Евгений //URL: https://dspace.enu.kz/bitstream/handle/data/12881/METODIKATRUDOZATRAT.pdf?sequence=1&isAllowed=y.
2. «Зарплати розробників за червень-липень 2021» [Ел. ресурс] URL: https://jobs.dou.ua/salaries/#period=jun2021&city=Dnipro&title=Junior%20Software%20Engineer&language=C%23%2F.NET&spec=&exp1=0&exp2=1?period=2022-06&position=Intern/Trainee%20SE
3. «Мінестрество енергетики України» [Ел. ресурс] URL: https://yasno.com.ua/b2c-tariffs
4. «Розміри ЄСВ-2021» [Ел. ресурс] URL: https://services.dtkt.ua/catalogues/indexes/
5. «Аренда офисов в Днепре» [Ел. ресурс] URL: https://besplatka.ua/ru/dnepropetrovsk/nedvizhimost/arenda-komercheskoynedvizhimosti/arendaofisov?prop[136][to]=2500&prop[112][to]=10&currency=
6. «Комп'ютери та ноутбуки» [Ел. ресурс] URL: https://rozetka.com.ua/ua/362083440/p362083440/

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
_____ Анатолій
РАДКЕВИЧ

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОРІНЯННЯ ФРЕЙМОРКІВ JAVASCRIPT

Робочий проект
ЛИСТ ЗАТВЕДЖЕННЯ
44165850.00001-01

Завідувач кафедри КІТ
_____ доц. Вадим ГОРЯЧКІН

Керівник розробки
____ доц. Вадим АНДРЮЩЕНКО

Виконавець
____ Владислав ТРИПУТИН

Нормоконтролер
____ доц. Світлана ВОЛКОВА

2022

ЗАТВЕРДЖЕНО

44165850.00001-01-ЛЗ

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОРІНЯННЯ ФРЕЙМОРКІВ JAVASCRIPT

Специфікація

44165850.00001-01

Аркушів 2

2022

Позначення	Найменування Документація	Примітки
	Лист затвердження	
	Лист затвердження	
	Текст програми	

ЗАТВЕРДЖЕНО

44165850.00001-01-ЛЗ

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОРІНЯННЯ ФРЕЙМОРКІВ JAVASCRIPT

Текст програми

44165850.00001-01

Аркушів 2

2022

АНОТАЦІЯ

Документ 1116130.01195-01 «Програмне забезпечення порівняння фреймворків JavaScript. Текст програми» відноситься до програмної документації дипломного проекту.

Даний документ містить опис структури проекту програми.

ЗМІСТ

1 ТЕКС ПРОГРАМИ

2.1 Текст програми файлу Ract,js

```
import { Component } from 'react';
import { render } from 'react-dom';

const random = (max) => Math.round(Math.random() * 1000) % max;

const A = ["pretty", "large", "big", "small", "tall", "short", "long",
"handsome", "plain", "quaint", "clean",
  "elegant", "easy", "angry", "crazy", "helpful", "mushy", "odd", "unsightly",
"adorable", "important", "inexpensive",
  "cheap", "expensive", "fancy"];
const C = ["red", "yellow", "blue", "green", "pink", "brown", "purple", "brown",
"white", "black", "orange"];
const N = ["table", "chair", "house", "bbq", "desk", "car", "pony", "cookie",
"sandwich", "burger", "pizza", "mouse",
  "keyboard"];

let nextId = 1;

const buildData = (count) => {
  const data = new Array(count);

  for (let i = 0; i < count; i++) {
    data[i] = {
      id: nextId++,
      label: `${A[random(A.length)]} ${C[random(C.length)]}
${N[random(N.length)]}`,
    };
  }

  return data;
};

const initialState = { data: [], selected: 0 };

class Row extends Component {
```

```
shouldComponentUpdate(nextProps) {
  const { item, selected } = this.props;

  return nextProps.item !== item || nextProps.selected !== selected;
}

constructor(props) {
  super(props);

  this.onSelect = () => {
    const { item, dispatch } = this.props;

    dispatch({ type: 'SELECT', id: item.id });
  };

  this.onRemove = () => {
    const { item, dispatch } = this.props;

    dispatch({ type: 'REMOVE', id: item.id });
  };
}

render() {
  const { selected, item } = this.props;

  return (
      <tr className={selected ? "danger" : ""}>
        <td className="col-md-1">{item.id}</td>
        <td className="col-md-4">
          <a onClick={this.onSelect}>{item.label}</a>
        </td>
        <td className="col-md-1">
          <a onClick={this.onRemove}>
            <span className="glyphicon glyphicon-remove" aria-hidden="true" />
          </a>
        </td>
        <td className="col-md-6" />
      </tr>
```

```
    );
  }
}

class Button extends Component {
  render() {
    const { id, cb, title } = this.props;

    return (
        <div className="col-sm-6 smallpad">
          <button type="button" className="btn btn-primary btn-block" id={id}
onClick={cb}>{title}</button>
        </div>
    );
  }
}

class Jumbotron extends Component {
  shouldComponentUpdate() {
    return false;
  }

  render() {
    const { dispatch } = this.props;

    return (
        <div className="jumbotron">
          <div className="row">
            <div className="col-md-6">
              <div className="row">
                <Button id="run" title="Create 1,000 rows" cb={() => dispatch({
type: 'RUN' })} />
                <Button id="runlots" title="Create 10,000 rows" cb={() =>
dispatch({ type: 'RUN_LOTS' })} />
                <Button id="add" title="Append 1,000 rows" cb={() => dispatch({
type: 'ADD' })} />
                <Button id="update" title="Update every 10th row" cb={() =>
dispatch({ type: 'UPDATE' })} />
```

```
                <Button id="clear" title="Clear" cb={() => dispatch({ type:
'CLEAR' })} />
                <Button id="swaprows" title="Swap Rows" cb={() => dispatch({
type: 'SWAP_ROWS' })} />
              </div>
            </div>
          </div>
        </div>
    )
  }
}

class Main extends Component {
  constructor(props) {
    super(props);

    this.state = initialState;

    this.dispatch = (action) => {
      const { data } = this.state;

      switch (action.type) {
        case 'RUN':
          return this.setState({ data: buildData(1000), selected: 0 });
        case 'RUN_LOTS':
          return this.setState({ data: buildData(10000), selected: 0 });
        case 'ADD':
          return this.setState({ data: data.concat(buildData(1000))});
        case 'UPDATE': {
          const newData = data.slice(0);

          for (let i = 0; i < newData.length; i += 10) {
            const r = newData[i];

            newData[i] = { id: r.id, label: r.label + " !!!" };
          }

          return this.setState({ data: newData });
```

```
        }
        case 'CLEAR':
          return this.setState({ data: [], selected: 0 });
        case 'SWAP_ROWS': {
          if (data.length > 998) {
            return this.setState({ data: [data[0], data[998], ...data.slice(2,
998), data[1], data[999]] });
          }

          return;
        }
        case 'REMOVE': {
          const idx = data.findIndex((d) => d.id === action.id);

          return this.setState({ data: [...data.slice(0, idx), ...data.slice(idx
+ 1)] });
        }
        case 'SELECT':
          return this.setState({ selected: action.id });
      }
    };
  }

  render() {
    const { data, selected } = this.state;

    return (
        <div className="container">
          <Jumbotron dispatch={this.dispatch} />
          <table className="table table-hover table-striped test-data">
            <tbody>
              {data.map((item) => (
                <Row key={item.id} item={item} selected={selected === item.id}
dispatch={this.dispatch} />
              ))}
            </tbody>
          </table>
```

```
          <span className="preloadicon glyphicon glyphicon-remove" aria-
hidden="true" />
        </div>
    );
  }
}

render(
  <Main />,
  document.getElementById('main'),
);
```

2.2 Текст програми файлу app.component.html

```
<div class="container">
    <div class="jumbotron">
        <div class="row">
            <div class="col-md-6">
                <div class="col-sm-6 smallpad">
                    <button type="button" class="btn btn-primary btn-block"
id="run" (click)="run()" ref="text">Create 1,000 rows</button>
                </div>
                <div class="col-sm-6 smallpad">
                    <button type="button" class="btn btn-primary btn-block"
id="runlots" (click)="runLots()">Create 10,000 rows</button>
                </div>
                <div class="col-sm-6 smallpad">
                    <button type="button" class="btn btn-primary btn-block"
id="add" (click)="add()" ref="text">Append 1,000 rows</button>
                </div>
                <div class="col-sm-6 smallpad">
                    <button type="button" class="btn btn-primary btn-block"
id="update" (click)="update()">Update every 10th row</button>
                </div>
                <div class="col-sm-6 smallpad">
                    <button type="button" class="btn btn-primary btn-block"
id="clear" (click)="clear()">Clear</button>
                </div>
```

```
                <div class="col-sm-6 smallpad">
                    <button type="button" class="btn btn-primary btn-block"
id="swaprows" (click)="swapRows()">Swap Rows</button>
                </div>
            </div>
        </div>
    </div>
    <table class="table table-hover table-striped test-data">
        <tbody>
            <tr [class.danger]="item.id === selected" *ngFor="let item of data;
trackBy: itemById">
                <td class="col-md-1">{{item.id}}</td>
                <td class="col-md-4">
                    <a href="#" (click)="select(item, $event)">{{item.label}}</a>
                </td>
                <td class="col-md-1"><a href="#" (click)="delete(item,
$event)"><span class="glyphicon glyphicon-remove" aria-
hidden="true"></span></a></td>
                <td class="col-md-6"></td>
            </tr>
        </tbody>
    </table>
    <span class="preloadicon glyphicon glyphicon-remove" aria-
hidden="true"></span>
</div>
```

2.3 Текст програми файлу app.component.ts

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

interface Data {
  id: number;
  label: string;
}

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
```

```
  styles: []
})
export class AppComponent {
    data: Array<Data> = [];
    selected: number = undefined;
    id: number = 1;
    backup: Array<Data> = undefined;
    version: string;

    constructor() {
        this.version = VERSION.full;
    }

    buildData(count: number = 1000): Array<Data> {
        var adjectives = ["pretty", "large", "big", "small", "tall", "short",
"long", "handsome", "plain", "quaint", "clean", "elegant", "easy", "angry",
"crazy", "helpful", "mushy", "odd", "unsightly", "adorable", "important",
"inexpensive", "cheap", "expensive", "fancy"];
        var colours = ["red", "yellow", "blue", "green", "pink", "brown",
"purple", "brown", "white", "black", "orange"];
        var nouns = ["table", "chair", "house", "bbq", "desk", "car", "pony",
"cookie", "sandwich", "burger", "pizza", "mouse", "keyboard"];
        var data: Array<Data> = [];
        for (var i = 0; i < count; i++) {
            data.push({ id: this.id, label:
adjectives[this._random(adjectives.length)] + " " +
colours[this._random(colours.length)] + " " + nouns[this._random(nouns.length)]
});
            this.id++;
        }
        return data;
    }

    _random(max: number) {
        return Math.round(Math.random() * 1000) % max;
    }

    itemById(index: number, item: Data) {
```

```
        return item.id;
    }

    select(item: Data, event: Event) {
        event.preventDefault();
        this.selected = item.id;
    }

    delete(item: Data, event: Event) {
        event.preventDefault();
        for (let i = 0, l = this.data.length; i < l; i++) {
            if (this.data[i].id === item.id) {
                this.data.splice(i, 1);
                break;
            }
        }
    }

    run() {
        this.data = this.buildData();
    }

    add() {
        this.data = this.data.concat(this.buildData(1000));
    }

    update() {
        for (let i = 0; i < this.data.length; i += 10) {
            this.data[i].label += ' !!!';
        }
    }
    runLots() {
        this.data = this.buildData(10000);
        this.selected = undefined;
    }
    clear() {
        this.data = [];
        this.selected = undefined;
```

```
    }
    swapRows() {
        if (this.data.length > 998) {
            var a = this.data[1];
            this.data[1] = this.data[998];
            this.data[998] = a;
        }
    }
}
```

2.1 Текст програми файлу app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

2.2 Текст програми файлу App.vue

```
<script setup>
import { ref, shallowRef } from 'vue'
import { buildData } from './data'
const selected = ref()
const rows = shallowRef([])
function setRows(update = rows.value.slice()) {
  rows.value = update
}
function add() {
```

```
  rows.value = rows.value.concat(buildData(1000))
}
function remove(id) {
  rows.value.splice(
    rows.value.findIndex((d) => d.id === id),
    1
  )
  setRows()
}
function select(id) {
  selected.value = id
}
function run() {
  setRows(buildData())
  selected.value = undefined
}
function update() {
  const _rows = rows.value
  for (let i = 0; i < _rows.length; i += 10) {
    _rows[i].label += ' !!!'
  }
  setRows()
}
function runLots() {
  setRows(buildData(10000))
  selected.value = undefined
}
function clear() {
  setRows([])
  selected.value = undefined
}
function swapRows() {
  const _rows = rows.value
  if (_rows.length > 998) {
    const d1 = _rows[1]
    const d998 = _rows[998]
    _rows[1] = d998
    _rows[998] = d1
```

```
    setRows()
  }
}
</script>

<template>
  <div class="jumbotron">
    <div class="row">
      <div class="col-md-6">
        <div class="row">
          <div class="col-sm-6 smallpad">
            <button
              type="button"
              class="btn btn-primary btn-block"
              id="run"
              @click="run"
            >
              Create 1,000 rows
            </button>
          </div>
          <div class="col-sm-6 smallpad">
            <button
              type="button"
              class="btn btn-primary btn-block"
              id="runlots"
              @click="runLots"
            >
              Create 10,000 rows
            </button>
          </div>
          <div class="col-sm-6 smallpad">
            <button
              type="button"
              class="btn btn-primary btn-block"
              id="add"
              @click="add"
            >
              Append 1,000 rows
```

```
          </button>
        </div>
        <div class="col-sm-6 smallpad">
          <button
            type="button"
            class="btn btn-primary btn-block"
            id="update"
            @click="update"
          >
            Update every 10th row
          </button>
        </div>
        <div class="col-sm-6 smallpad">
          <button
            type="button"
            class="btn btn-primary btn-block"
            id="clear"
            @click="clear"
          >
            Clear
          </button>
        </div>
        <div class="col-sm-6 smallpad">
          <button
            type="button"
            class="btn btn-primary btn-block"
            id="swaprows"
            @click="swapRows"
          >
            Swap Rows
          </button>
        </div>
      </div>
    </div>
  </div>
</div>
<table class="table table-hover table-striped test-data">
  <tbody>
```

```
      <tr
        v-for="{ id, label } of rows"
        :key="id"
        :class="{ danger: id === selected }"
        :data-label="label"
        v-memo="[label, id === selected]"
      >
        <td class="col-md-1">{{ id }}</td>
        <td class="col-md-4">
          <a @click="select(id)">{{ label }}</a>
        </td>
        <td class="col-md-1">
          <a @click="remove(id)">
            <span class="glyphicon glyphicon-remove" aria-hidden="true"></span>
          </a>
        </td>
        <td class="col-md-6"></td>
      </tr>
    </tbody>
  </table>
  <span
    class="preloadicon glyphicon glyphicon-remove"
    aria-hidden="true"
  ></span>
</template>
```

2.3 Текст програми файлу data.js

```
let ID = 1

function _random(max) {
  return Math.round(Math.random() * 1000) % max
}

export function buildData(count = 1000) {
  const adjectives = [
    'pretty',
    'large',
```

```
  'big',
  'small',
  'tall',
  'short',
  'long',
  'handsome',
  'plain',
  'quaint',
  'clean',
  'elegant',
  'easy',
  'angry',
  'crazy',
  'helpful',
  'mushy',
  'odd',
  'unsightly',
  'adorable',
  'important',
  'inexpensive',
  'cheap',
  'expensive',
  'fancy'
]
const colours = [
  'red',
  'yellow',
  'blue',
  'green',
  'pink',
  'brown',
  'purple',
  'brown',
  'white',
  'black',
  'orange'
]
const nouns = [
```

```
    'table',
    'chair',
    'house',
    'bbq',
    'desk',
    'car',
    'pony',
    'cookie',
    'sandwich',
    'burger',
    'pizza',
    'mouse',
    'keyboard'
  ]
  const data = []
  for (let i = 0; i < count; i++)
    data.push({
      id: ID++,
      label:
        adjectives[_random(adjectives.length)] +
        ' ' +
        colours[_random(colours.length)] +
        ' ' +
        nouns[_random(nouns.length)]
    })
  return data
}
```

2. 7 Текст програми файлу main.js

```
import { createApp } from 'vue'
import App from './App.vue'

createApp(App).mount('#main')
```

2. 8 Текст програми файлу vanilla.js

```
'use strict';
```

```
function _random(max) {
    return Math.round(Math.random()*1000)%max;
}

const rowTemplate = document.createElement("tr");
rowTemplate.innerHTML = "<td class='col-md-1'></td><td class='col-md-4'><a class='lbl'></a></td><td class='col-md-1'><a class='remove'><span class='remove glyphicon glyphicon-remove' aria-hidden='true'></span></a></td><td class='col-md-6'></td>";

class Store {
    constructor() {
        this.data = [];
        this.backup = null;
        this.selected = null;
        this.id = 1;
    }
    buildData(count = 1000) {
        var adjectives = ["pretty", "large", "big", "small", "tall", "short", "long", "handsome", "plain", "quaint", "clean", "elegant", "easy", "angry", "crazy", "helpful", "mushy", "odd", "unsightly", "adorable", "important", "inexpensive", "cheap", "expensive", "fancy"];
        var colours = ["red", "yellow", "blue", "green", "pink", "brown", "purple", "brown", "white", "black", "orange"];
        var nouns = ["table", "chair", "house", "bbq", "desk", "car", "pony", "cookie", "sandwich", "burger", "pizza", "mouse", "keyboard"];
        var data = [];
        for (var i = 0; i < count; i++)
            data.push({id: this.id++, label: adjectives[_random(adjectives.length)] + " " + colours[_random(colours.length)] + " " + nouns[_random(nouns.length)] });
        return data;
    }
    updateData(mod = 10) {
        for (let i=0;i<this.data.length;i+=10) {
            this.data[i].label += ' !!!';
            // this.data[i] = Object.assign({}, this.data[i], {label: this.data[i].label +' !!!'});
```

```
        }
    }
    delete(id) {
        const idx = this.data.findIndex(d => d.id==id);
        this.data = this.data.filter((e,i) => i!=idx);
        return this;
    }
    run() {
        this.data = this.buildData();
        this.selected = null;
    }
    add() {
        this.data = this.data.concat(this.buildData(1000));
        this.selected = null;
    }
    update() {
        this.updateData();
        this.selected = null;
    }
    select(id) {
        this.selected = id;
    }
    hideAll() {
        this.backup = this.data;
        this.data = [];
        this.selected = null;
    }
    showAll() {
        this.data = this.backup;
        this.backup = null;
        this.selected = null;
    }
    runLots() {
        this.data = this.buildData(10000);
        this.selected = null;
    }
    clear() {
        this.data = [];
```

```
        this.selected = null;
    }
    swapRows() {
        if(this.data.length > 998) {
            var a = this.data[1];
            this.data[1] = this.data[998];
            this.data[998] = a;
        }
    }
}

var getParentId = function(elem) {
    while (elem) {
        if (elem.tagName==="TR") {
            return elem.data_id;
        }
        elem = elem.parentNode;
    }
    return undefined;
}
class Main {
    constructor(props) {
        this.store = new Store();
        this.select = this.select.bind(this);
        this.delete = this.delete.bind(this);
        this.add = this.add.bind(this);
        this.run = this.run.bind(this);
        this.update = this.update.bind(this);
        this.start = 0;
        this.rows = [];
        this.data = [];
        this.selectedRow = undefined;

        document.getElementById("main").addEventListener('click', e => {
            //console.log("listener",e);
            if (e.target.matches('#add')) {
                e.preventDefault();
                //console.log("add");
```

```
            this.add();
        }
        else if (e.target.matches('#run')) {
            e.preventDefault();
            //console.log("run");
            this.run();
        }
        else if (e.target.matches('#update')) {
            e.preventDefault();
            //console.log("update");
            this.update();
        }
        else if (e.target.matches('#hideall')) {
            e.preventDefault();
            //console.log("hideAll");
            this.hideAll();
        }
        else if (e.target.matches('#showall')) {
            e.preventDefault();
            //console.log("showAll");
            this.showAll();
        }
        else if (e.target.matches('#runlots')) {
            e.preventDefault();
            //console.log("runLots");
            this.runLots();
        }
        else if (e.target.matches('#clear')) {
            e.preventDefault();
            //console.log("clear");
            this.clear();
        }
        else if (e.target.matches('#swaprows')) {
            e.preventDefault();
            //console.log("swapRows");
            this.swapRows();
        }
        else if (e.target.matches('.remove')) {
```

```
                e.preventDefault();
                let id = getParentId(e.target);
                let idx = this.findIdx(id);
                //console.log("delete",idx);
                this.delete(idx);
            }
            else if (e.target.matches('.lbl')) {
                e.preventDefault();
                let id = getParentId(e.target);
                let idx = this.findIdx(id);
                //console.log("select",idx);
                this.select(idx);
            }
        });
        this.tbody = document.getElementById("tbody");
    }
    findIdx(id) {
        for (let i=0;i<this.data.length;i++){
            if (this.data[i].id === id) return i;
        }
        return undefined;
    }
    run() {
        this.removeAllRows();
        this.store.clear();
        this.rows = [];
        this.data = [];
        this.store.run();
        this.appendRows();
        this.unselect();
    }
    add() {
        this.store.add();
        this.appendRows();
    }
    update() {
        this.store.update();
        for (let i=0;i<this.data.length;i+=10) {
```

```
            this.rows[i].childNodes[1].childNodes[0].innerText =
this.store.data[i].label;
        }
    }
    unselect() {
        if (this.selectedRow !== undefined) {
            this.selectedRow.className = "";
            this.selectedRow = undefined;
        }
    }
    select(idx) {
        this.unselect();
        this.store.select(this.data[idx].id);
        this.selectedRow = this.rows[idx];
        this.selectedRow.className = "danger";
    }
    recreateSelection() {
        let old_selection = this.store.selected;
        let sel_idx = this.store.data.findIndex(d => d.id === old_selection);
        if (sel_idx >= 0) {
            this.store.select(this.data[sel_idx].id);
            this.selectedRow = this.rows[sel_idx];
            this.selectedRow.className = "danger";
        }
    }
    delete(idx) {
        this.store.delete(this.data[idx].id);
        this.rows[idx].remove();
        this.rows.splice(idx, 1);
        this.data.splice(idx, 1);
        this.unselect();
        this.recreateSelection();
    }
    removeAllRows() {

    }
    runLots() {
        this.removeAllRows();
```

```
        this.store.clear();
        this.rows = [];
        this.data = [];
        this.store.runLots();
        this.appendRows();
        this.unselect();
    }
    clear() {
        this.store.clear();
        this.rows = [];
        this.data = [];

    }
    swapRows() {
        if (this.data.length>10) {
            this.store.swapRows();
            this.data[1] = this.store.data[1];
            this.data[998] = this.store.data[998];

            this.tbody.insertBefore(this.rows[998], this.rows[2])
            this.tbody.insertBefore(this.rows[1], this.rows[999])

            let tmp = this.rows[998];
            this.rows[998] = this.rows[1];
            this.rows[1] = tmp;
        }



    }
    appendRows() {
           var rows = this.rows, s_data = this.store.data, data = this.data,
tbody = this.tbody;
        for(let i=rows.length;i<s_data.length; i++) {
            let tr = this.createRow(s_data[i]);
            rows[i] = tr;
            data[i] = s_data[i];
            tbody.appendChild(tr);
```

```
        }
    }
    createRow(data) {
        const tr = rowTemplate.cloneNode(true),
            td1 = tr.firstChild,
            a2 = td1.nextSibling.firstChild;
        tr.data_id = data.id;
        td1.textContent = data.id;
        a2.textContent = data.label;
        return tr;
    }
}

new Main();
```

Дослідження впливу React на фреймворки розробки Web-застосунків

Трипутін Владислав Сергійович, Український державний університет науки та технологій (УДУНТ)

Людство дедалі більше залежить від інформації. Тому важливо надати її якісно, в актуальному стані, у потрібній кількості та якнайшвидше. У цьому сенс усіх сучасних програм. До описаних вище завдань можна додати інтеграцію продуктів із мережею Інтернет, яка давно стала невід'ємною частиною життя кожної людини. Однією з найпопулярніших та найефективніших мов програмування для таких цілей є JavaScript, який дозволяє створювати інтерактивні, зручні та якісні програми.

Вибір необхідної бібліотеки та її ефективне використання означає вирішення завдання максимально швидко та ефективно з мінімальними витратами коштів. Що доводить актуальність цього питання. Вебфреймворки допомагають досягти структури в програмах, і вони дають додаткові функції, які можна додати до них без зайвої роботи. Фреймворк дає можливість почати, щоб існувала можливість зосередитися на функціях, а не на деталях конфігурації. На сьогоднішній день існує багато фреймворків, тож іноді буває складно обрати, з якого саме почати. Для певної потреби існує свій фреймворк. Перш ніж обирати потрібний фреймворк спочатку потрібно дізнатися про важливі елементи, які слід враховувати.

Тому обґрунтованою є тема магістерської роботи, у якій вирішується науково-прикладне завдання дослідження екосистеми React та порівняння його з іншими фреймворками.

Спочатку ми з'ясували, про історію розвитку React та встановили, що концепції React зазвичай розглядаються як такі, що відходять від загальноприйнятих найкращих практик розробки інтерфейсу користувача. Виклик статус-кво та теорії тестування дозволили React стати високопродуктивним і масштабованим фреймворком JavaScript для створення інтерфейсів користувача.

Рішення «за» та «проти» певної технології значною мірою залежать від варіанту використання та інших різноманітних обставин: розміри проектів, знання працівників, попередній досвід і подібні терміни впливають на цей процес. Angular, React і Vue не стоять далеко один від одного. У той час як перші два відрізняються переважно мовою розробки та філософією поділу файлів, вони схожі на велику технологічну компанію, яка спонсорує їхню розробку, що забезпечує певний рівень довіри для бази користувачів або тих, хто планує їх використовувати. Vue у цьому відношенні є повною протилежністю, оскільки він розроблений переважно з урахуванням інтересів спільноти. Крім того, він узяв деякі з найкращих функцій із уже існуючих фреймворків і зумів створити щось нове, що мало величезне зростання популярності у 2017 році та продовжується.

У світі JavaScript, де регулярно публікуються нові фреймворки, може бути нерозумним довго чекати з впровадженням нової технології, оскільки на той час вона може бути застарілою. Хоча це, звісно, значною мірою залежить від розміру компанії: тоді як менші команди розробників можуть швидше протестувати та прийняти нову структуру, більшим компаніям потрібно більше часу для оцінки, оскільки неправильне рішення може призвести до фінансових проблем заднім числом

Потім ми також встановили, що ReactJS – це продуктивна, ієрархічна та функціональна технологія, яка має низку переваг. Фреймворк пропонує широкий контекст із такими можливостями, як тривимірні зображення, загальні елементи, одностороннє інформаційне з'єднання, тригери та JSX.

Angular є найбільш зрілим фреймворком, має хорошу підтримку з точки зору учасників і є повним пакетом. Однак крива навчання є стрімкою, і концепції розробки в Angular можуть відштовхнути нових розробників. Angular – хороший вибір для компаній із великими командами та розробників, які вже використовують TypeScript.

React достатньо дорослий, щоб бути зрілим, і має величезну кількість внесків від спільноти. Він отримав широке визнання. Ринок праці для React справді хороший, і майбутнє для цього фреймворку виглядає яскравим. React виглядає як хороший вибір для тих, хто починає працювати з інтерфейсними фреймворками JavaScript, стартапів і розробників, яким подобається гнучкість. Можливість легкої інтеграції з іншими фреймворками дає велику перевагу для тих, хто хоче певної гнучкості у своєму коді.

Vue є найновішим на арені, без підтримки великої компанії. Однак за останні кілька років він справді успішно виступив як сильний конкурент для Angular і React, особливо з випуском Vue 3.0. Можливо, це відіграє свою роль у тому, що багато китайських гігантів, таких як Alibaba та Baidu, обирають Vue як свою основну інтерфейсну структуру JavaScript. Vue має бути вашим вибором, якщо ви віддаєте перевагу простоті, але також любите гнучкість.

Щоб обмежити кількість критеріїв, вибираються лише критерії з двома або більше посиланнями. Стабільність і зрілість пов’язані, оскільки зрілість програмного забезпечення впливає на стабільність, тому вони об’єднані за одним критерієм. Тому, щоб мати можливість використовувати фреймворк SPA в якомога більшій кількості браузерів, сумісність і портативність є важливими, тому ці два критерії згруповані разом. У інформатиці постійність часто стосується збереження поточного стану програми в пам’яті. Щоб зберегти стан SPA в пам’яті, потрібна хороша продуктивність кешу. Це призводить до групування стійкості та продуктивності кешу разом. Зручність використання може бути визначена стандартом ISO 9241-11, який охоплює сферу, де користувач повинен бути задоволений використанням програмного забезпечення. Це добре поєднується з простотою, тому що якщо щось є простим у використанні, воно повинно мати високу зручність використання, навпаки, те, що непросто у використанні, має мати нижчий рівень зручності використання. Тому простота і зручність у використанні згруповані. Документація, розмір фреймворку та внутрішній досвід вибираються як запитання для використання відповідно до відповідних критеріїв.

У висновку ми з’ясували, що порівняти фреймворки абсолютно об’єктивно досить складно. Але аналіз дозволить розробнику-початківцю підібрати платформу. Кожен із фреймворків гарний по-своєму, є свої сильні та слабкі сторони. Тому варто дати ще одну важливу пораду: що простіше вивчитиме команді програмістів, то й слід використовувати для розробки.

УДК

В. С. ТРИПУТІН[1*]

[1*]Студент, Український державний університет науки і технологій, вул. Лазаряна, 2к3, Дніпро, Україна, 49010, тел. +38 (068) 768 10 62, ел. пошта vladtriputin@gmail.com

Дослідження впливу React на фреймворки розробки Web-застосунків

Мета. Метою роботи є аналіз можливостей фреймворків для створення інформаційних сайтів. **Методика.** Для досягнення мети дослідження необхідно вирішити такі завдання: 1) провести аналіз предметної області; 2) обґрунтувати вибір фреймворків; 3) визначити критерії порівняння; 4) ознайомитись з обраними інструментами розробки; 5) реалізувати необхідний функціонал для трьох застосунків; 6) провести порівняння за заданими критеріями. **Наукова новизна.** Дослідження робить внесок у визначенні практичної застосовності фреймворків. Результати дослідження дозволяють зробити висновки щодо того в яких ситуаціях той чи інший фреймворк показує кращі результати. **Практична значимість**. Особистий внесок здобувача полягає у аналізі літератури з теми дослідження, проведенні аналізу існуючих фреймворків, розгляді відомих фреймворків. В якості експериментальної частини атестаційної роботи було розроблено сторінку інформаційного сайту.

Ключові слова: фреймворк, React, застосунок, інформаційний сайт.

Вступ

Людство дедалі більше залежить від інформації. Тому важливо надати її якісно, в актуальному стані, у потрібній кількості та якнайшвидше. У цьому сенс усіх сучасних програм. До описаних вище завдань можна додати інтеграцію продуктів із мережею Інтернет, яка давно стала невід'ємною частиною життя кожної людини. Однією з найпопулярніших та найефективніших мов програмування для таких цілей є JavaScript, який дозволяє створювати інтерактивні, зручні та якісні програми.

В основному, проблеми, що виникають при розробці таких продуктів, пов'язані з великим розміром коду, часом, витраченим для написання і тестування, а також його адаптованістю під різні платформи. У цьому сенсі використання бібліотек JavaScript дозволяє вирішувати ці проблеми. Які перетворюють цю мову на універсальний інструмент, налаштований на виконання певних завдань.

Вибір необхідної бібліотеки та її ефективне використання означає вирішення завдання максимально швидко та ефективно з мінімальними витратами коштів. Що доводить актуальність цього питання. Веб фреймворки допомагають досягти структури в програмах, і вони дають додаткові функції, які можна додати до них без зайвої роботи. Фреймворк дає можливість почати, щоб існувала можливість зосередитися на функціях, а не на деталях конфігурації. На сьогоднішній день існує багато фреймворків, тож іноді буває складно обрати, з якого саме почати. Для певної потреби існує свій фреймворк. Перш ніж обирати потрібний фреймворк спочатку потрібно дізнатися про важливі елементи, які слід враховувати.

Мета

Враховуючи вищезгадане, автори мають за мету аналіз можливостей фреймворків для створення інформаційних сайтів.

Методика

Для досягнення мети дослідження автори вирішити такі завдання: 1) провести аналіз предметної області; 2) обґрунтувати вибір фреймворків; 3) визначити критерії порівняння; 4) ознайомитись з обраними інструментами розробки; 5) реалізувати необхідний функціонал для трьох застосунків; 6) провести порівняння за заданими критеріями.

Результати

Результати дослідження дозволяють зробити висновки щодо того в яких ситуаціях той чи інший фреймворк показує кращі результати. Також на основі проведеного експерименту була створена таблиці (табл. 1) де вказано час завантаження різних фреймворків та кількість зайнятої пам’яті на жорсткому диску (табл. 2).

Таблиця 1

Час завантаження різних фреймворку

Фреймворк	vanillajs, с	vue-v3.2.37, с	angular-v13.0.0, с	react-v17.0.2, с
Створення 1000 рядків (5 розминкових прогонів).	96.7±1.1	119.3±1.2	118.5±1.4	126.2±1.4
Оновлення всіх 1000 рядків (5 розминкових запусків).	97.6±1.5	113.2±1.5	130.7±1.0	126.7±0.9
Оновлення кожного 10-го рядка для 1000 рядків (3 розминки). Уповільнення процесора в 16 разів.	297.1±8.8	359.0±11.2	343.0±6.7	399.5±5.4
Виділення виділеного рядка. (5 розминкових запусків). Уповільнення процесора в 16 разів.	33.1±1.2	53.7±1.2	43.6±0.8	105.2±3.7
Поміняти 2 рядки на таблицю з 1000 рядків. (5 розминкових запусків). Уповільнення процесора в 4 рази.	65.4±3.9	73.6±4.5	512.8±3.4	494.1±7.7
Видалення одного ряду. (5 розминкових запусків).	24.1±0.3	28.0±0.6	25.7±0.4	27.8±0.5
Створення 10 000 рядків. (5 розминкових прогонів з 1k рядків).	968.41±3.5	1,149.2±5.1	1,215.1±11.5	1,488.1±10.2
Додавання 1000 до таблиці з 10 000 рядків. 2х уповільнення процесора.	231.6±4.1	268.8±7.3	303.2±2.8	322.5±4.0
Очищення таблиці з 1000 рядків. Уповільнення процесора в 8 разів. (5 розминкових прогонів).	71.3±4.2	90.0±2.9	231.0±6.8	101.0±2.9
Постійно інтерактивний	1,879.7±0.7	2,104.84±9.6	2,783.1±0.4	2,554.8±0.2

Песимістичний tti - коли процесор і мережа безумовно простоюють. (більше жодних завдань цп понад 50 мс)				
Загальна вага в кілобайтах мережевої передачі (після стиснення) усіх ресурсів, завантажених на сторінку.	150.6±0.0	196.4±0.0	291.2±0.0	274.5±0.0

Таблиця 2

Кількість зайнятої пам’яті на жорсткому диску

Фреймворк	vanillajs, МБайт	vue-v3.2.37, МБайт	angular-v13.0.0, МБайт	react-v17.0.2, МБайт
Використання пам'яті після завантаження сторінки.	1.0	1.3	2.0	1.4
Використання пам’яті після додавання 1000 рядків.	2.5	4.4	5.3	5.6
Використання пам’яті після 5 разів клацання оновлення кожного 10-го рядка	2.5	4.4	5.4	6.1
Використання пам’яті після створення та очищення 1000 рядків 5 разів	1.0	1.6	2.7	2.2
Використання пам'яті після додавання 10 000 рядків.	16.4	30.5	32.6	39.5

Наукова новизна та практична значимість

Авторами цієї роботи на основі порівняння фреймворків та проведення експерименту вияснили що дослідження робить внесок у визначенні практичної застосовності фреймворків.

Висновки

Порівняти фреймворки абсолютно об'єктивно досить складно. Але аналіз дозволить розробнику-початківцю підібрати платформу. Підведемо підсумки:

Якщо потрібно швидко вивчити середовище, варто вибирати між Vue.js і React.

На Vue.js легко перейти як користувачу Angular, так і React. Адже тут виходить

чистий HTML-код, знайомий усім розробникам. Прийоми та техніки використовуються приблизно ті ж, що й у Angular.

Якщо передбачається розробка великого проекту, варто розглядати Angular як основу. Він забезпечує максимальну гнучкість та швидкість рендерингу. Величезний досвід інших розробників дозволить вирішити питання, які обов'язково виникнуть під час роботи над додатком. React виявиться занадто об'ємним, а для Vue.js ще немає великої кількості гайд-лайнів.

Якщо до розробки в майбутньому будуть залучатись інші програмісти, то Vue.js стане найкращим вибором. Адже цей фреймворк не тільки простий для вивчення, а й дозволяє міняти програму без руйнування його архітектури.

Якщо для проекту передбачається багатоступінчасте оновлення та розширення функціональності в майбутньому, то варто використовувати Vue.js або React через чудову зворотну сумісність.

Кожен із фреймворків гарний по-своєму, є свої сильні та слабкі сторони. Тому варто дати ще одну важливу пораду: що простіше вивчитиме команді програмістів, то й слід використовувати для розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. AngularJS: Security. https://docs.angularjs.org/guide/security, 2016. Accessed: 2016-04-05.
2. BenchmarkJS. https://benchmarkjs.com/, 2016. Accessed: 2016-04-05.
3. EmberJS: Security. http://emberjs.com/security/, 2016. Accessed: 2016-04-05.
4. OWASP. https://www.owasp.org/index.php/Main_Page, 2016. Accessed: 2016-04-05.
5. React: JSX Gotchas. https://facebook.github.io/react/docs/jsx-gotchas.html, 2016. Accessed: 2016-04-05.
6. TodoMVC. https://www.todomvc.com/, 2016. Accessed: 2016-04-22.
7. AngularJS and community. Data Binding. https://docs.angularjs.org/guide/databinding, 2016. Accessed: 2016-04-05.
8. Len Bass, Paul Clements, and Rick Kazman. Software Architecture in Practice. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edition, 2003.
9. PerOlof Bengtsson, Nico Lassing, Jan Bosch, and Hans van Vliet. Architecture level modifiability analysis (alma). Journal of Systems and Software, 69(1–2):129 – 147, 2004.
10. Klaus Bergner, Andreas Rausch, Marc Sihling, and Thomas Ternité. Dosam – domain-specific software architecture comparison model. In Proceedings of the First International Conference on Quality of Software Architectures and Software Quality, and Proceedings of the Second International Conference on Software Quality, QoSA'05, pages 4–20, Berlin, Heidelberg, 2005. SpringerVerlag.
11. Mathias Bynens and John-David Dalton. Bulletproof JavaScript benchmarks. http://calendar.perfplanet.com/2010/bulletproof-javascript-benchmarks/, 2010. Accessed: 2016-04-05.