Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

Порівняльна характеристика JavaScript фреймворків (React.js, Angular.js, Vue.js)

Текстова частина до курсової роботи
за спеціальністю „Інженерія програмного забезпечення” 121

Керівник курсової роботи
доц, к.н. Жежерун О.П.

_____
(підпис)
“____” _____ 2021 р.

Виконала студентка
Рибак Н. Я.

_____
(підпис)
“____” _____ 2021 р.

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав. Кафедри мультимедійних систем,

доц.,к.ф.-м.н.

_____ О.П. Жежерун

(підпис)

“____” _____ 2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

Рибак Наталі Ярославівні, студентці 1 курсу магістерської програми за спеціальністю «Інженерія програмного забезпечення» факультету Інформатики.

Тема: Порівняльна характеристика JavaScript фреймворків (React.js, Angular.js, Vue.js)

Вихідні дані:

Звіт з порівняльною характеристикою обраних сучасних JavaScript фреймворків та бібліотек за метриками.

Зміст ТЧ до курсової роботи:

Вступ
Частина 1: Аналіз предметної області
Частина 2: Теоретичні відомості
Частина 3: Опис реалізації програмного продукту
Висновки
Список використаної літератури
Додатки

Дата видачі „___” _____ 2021 р. Керівник _____
(підпис)

Завдання отримала _____
(підпис)

Зміст

Глосарій

Поняття	Визначення
DOM(Document Object Model)	Об'єктна модель документа — специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами.
JavaScript (JS)	Динамічна, об’єктно-орієнтована прототипна мова програмування.
TypeScript (TS)	Мова програмування, представлена Microsoft; позиціонується як засіб розробки веб-застосунків, що розширює можливості JavaScript.
UX (User Experience)	Досвід користування або те, що людина відчуває при користуванні продуктом, системою чи сервісом, включаючи емоції, враження, практичність та корисність.
URL	Стандартизована адреса певного ресурсу.
Webpack	Пакувальник статичних модулів для сучасних додатків JavaScript з відкритим кодом.
Плагін	Додаток до основної програми, призначений для розширення або використання її можливостей.
Деплой	Deployment(процес загрузки версії сайту на сервер)
Неймінг	Згода по іменуванню файлів, змінних, класів чи методів.
Хостинг	Послуга надавання дискового простору, підключення до мережі та інших ресурсів для розміщення фізичної інформації на сервері, що постійно перебуває в мережі (наприклад інтернеті).
Прототипне програмування	Стиль об'єктно-орієнтованого програмування, при якому відсутнє поняття класу, а повторне використання (успадкування) проводиться шляхом клонування наявного примірника об'єкта — прототипу.
Фреймворк	Каркас програмного застосунку; інфраструктура програмних рішень, що полегшує розробку складних систем.

Анотація

У роботі представлено загальну та порівняльну характеристику трьох найпоширеніших JavaScript фреймворків: Angular, React.js та Vue.js. Проаналізовано їх переваги та недоліки, доцільність використання для багатосторінкових та односторінкових застосунків. Розглянуто такі сфери порівняння як популярність на ринку праці, легкість вивчення, управління станом додатку та продуктивність виконання.

Для прикладу було створено невеликі застосунки для демонстрації функціоналу введення-виведення даних, роботи з API та компонентами односторінкового застосунку.

У першому розділі наведено аналіз предметної області та обґрунтування обраних фреймворків. У другому описані загальні відомості про роль мови програмування JavaScript у розробці інтерфейсів користувача, а також загальний огляд інструментів. Третій розділ містить розгорнутий опис ключових функціональних можливостей застосунків та заключне порівняння.

Вступ

Неспадна популярність мови JavaScript породжує велику кількість нових інструментів для швидшої та зручнішої розробки, що ускладнює процес вибору. На сьогодні не існує єдиного, універсального фреймворку, лише фаворити серед розробників, які мають декілька відмінних один від одного принципів роботи. Проведення порівняльної характеристики дозволить визначити основні переваги та недоліки, а також доцільність використання для різних застосунків.

Метою курсової роботи є формування порівняльної характеристики фреймворків Angular, React.js та Vue.js, для досягнення якої потрібно виконати наступні завдання:

1) здійснити аналіз предметної області та обґрунтувати вибір фреймворків;
2) визначити критерії порівняння;
3) обрати засоби розробки;
4) розробити три застосунки використовуючи кожен інструмент;
5) виконати заключне порівняння;

Об’єктом дослідження є фреймворки Angular, React.js та Vue.js.

Предметом дослідження є вивчення особливостей роботи кожного з них для розробки односторінкового веб-застосунку.

Використане програмне забезпечення: Angular, React.js, Vue.js, Javascript, Bootstrap, Angular Materials, News API, Webstorm IDE.

Робота складається з вступу, трьох розділів, висновку та додатків.

В першому розділі роботи досліджено предметну область та обґрунтовано вибір фреймворків для дослідження. Розглянуто принципи роботи, їх основні переваги та недоліки. Виділено критерії порівняння, які мають бути проаналізовані для кожного інструменту.

Другий розділ містить теоретичні відомості про світ JavaScript та огляд допоміжних інструментів розробника та загальну характеристику фреймворків.

У третьому розділі проаналізоване технічне завдання та надано порівняльну характеристику для Angular, React та Vue. Наведено опис реалізації

ключових функцій сайту за допомогою обраних інструментів: введення та виведення даних, відображення контенту та робота з компонентами застосунку.

Розділ 1. Аналіз предметної області

1.1 Аналіз сучасного стану питання та обґрунтування теми

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.

1.2 Обґрунтування обраних фреймворків

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

a. Опубліковані рейтинги

Веб-ресурс State of JS 2020 опублікували рейтинг дев’яти найбільш використовуваних фреймворків [1], який ранжується за задоволеністю користувачів, інтересом до інструменту, відсотком використання та обізнаністю. За останніми двома критеріями впродовж останніх чотирьох років трійка Angular, React та Vue.js тримається на вершині списку (рис. 1.2.1), що свідчить про надійність, наявність неспрадного інтересу серед компаній, а також гарантує велику кількість документації та готових рішень.

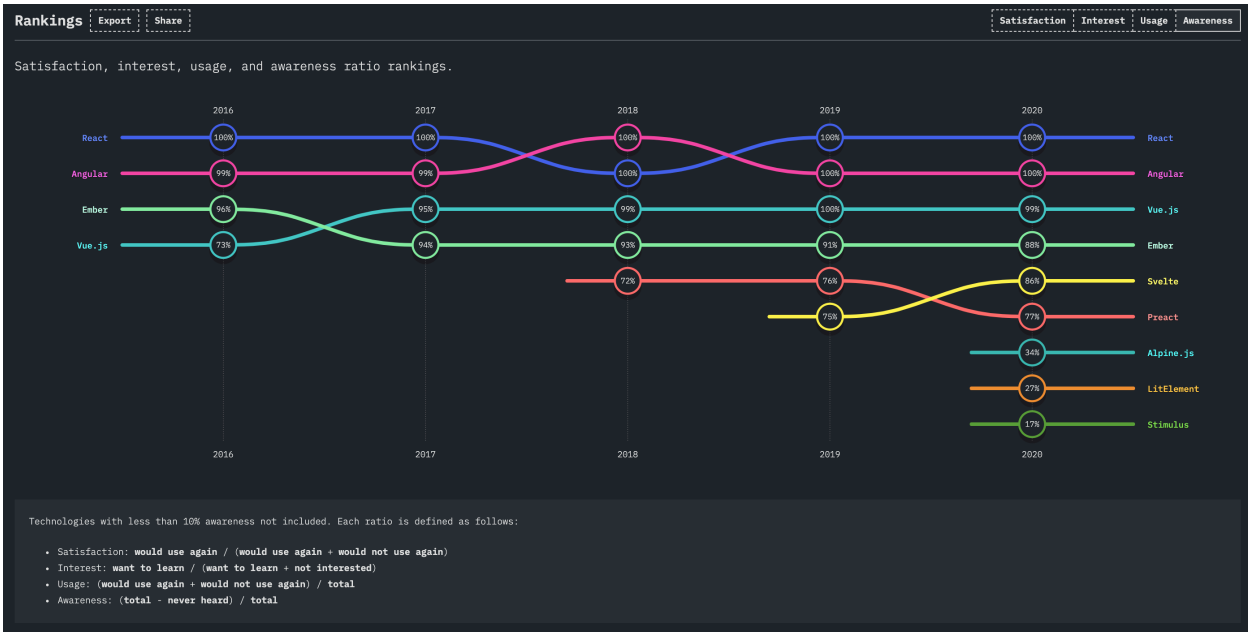


Рисунок 1.2.1 Рейтинг State of JS 2020

Наступний показовий рейтинг – 2020 Developer Survey [2] від компанії Stackoverflow(рисунок 1.2.2), який свідчить про зростаючу зацікавленість до обраних фреймворків. Джерело дає коментар про те, що бібліотека jQuery є лідером серед запитів, проте повільно втрачає позиції завдяки Angular та React. Можна зробити заключення про те, що понад 35% респондентів використовує одну з цих технологій.

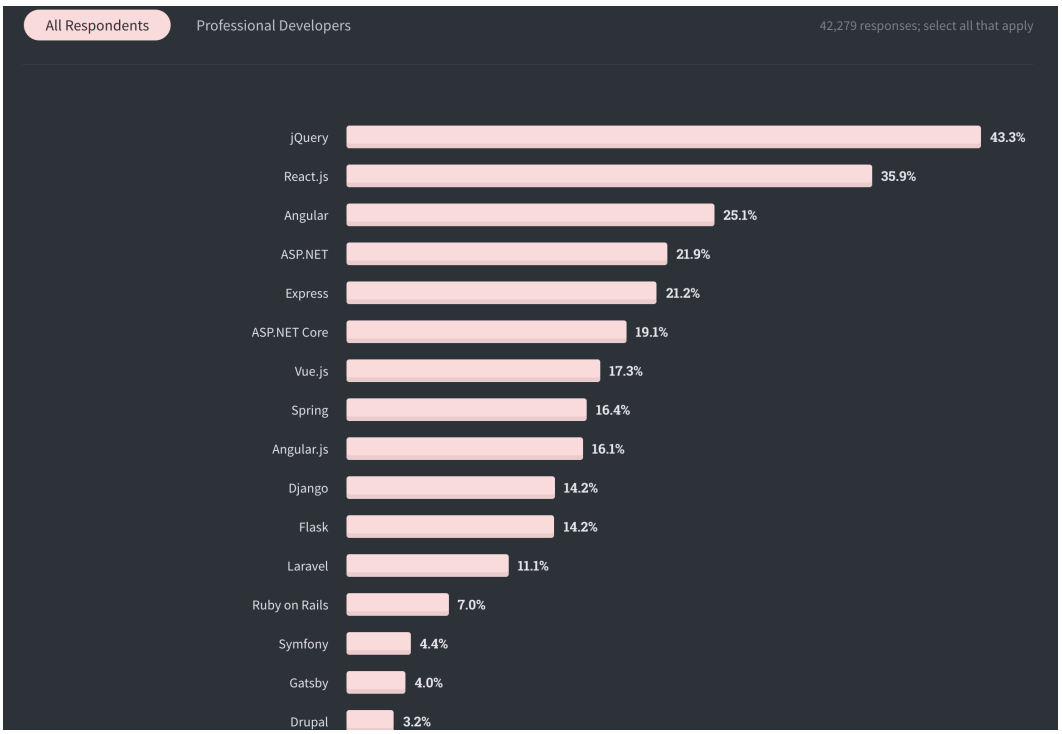


Рисунок 1.2.2 Рейтинг 2020 Developer Survey

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. 1.2.3). У ньому з великим відривом лідує React, а Angular та Vue.js йдуть наступними.

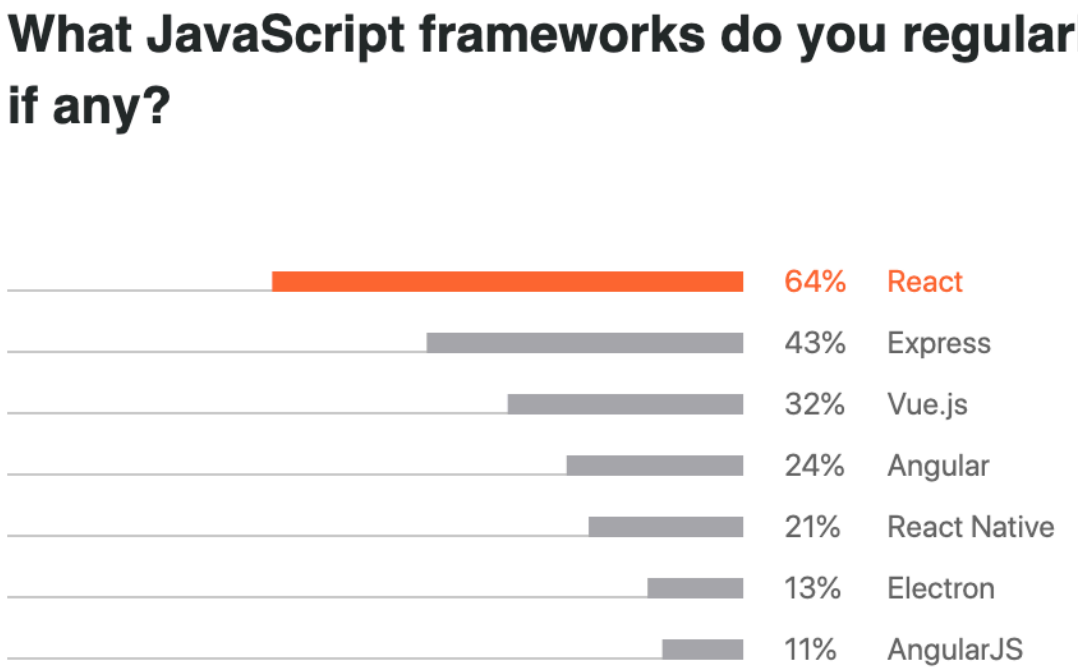


Рисунок 1.2.2 Опитування Jetbrains

b. Кількість вакансій на ринку

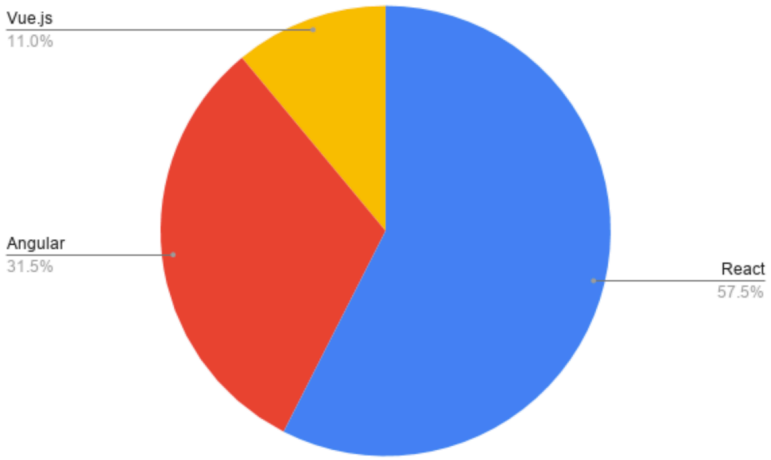
Було проаналізовано чотири найбільших веб-ресурсів з вакансіями у сфері IT в Україні. У таблиці 1.2.1 наведені цифри – приблизна кількість об’яв, які містять у назві чи описі назви одного з трьох фреймворків. На першому місці знаходиться React, далі – Angular. А останнє місце Vue пояснюється тим, що даний інструмент відносно молодий у порівнянні з іншими.

	Angular	React	Vue
ua.indeed.com	≈ 13 000	≈ 21 200	≈ 4 750
jobs.dou.ua	≈ 790	≈ 1 380	≈ 470
djinni.co	≈ 2 160	≈ 3 710	≈ 1 240
grc.ua	≈ 100	≈ 140	≈ 60

Таблиця 1.2.1 – Числа вакансій

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%.



Source: Google Trends

1.3 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.

Розділ 2. Теоретичні відомості

2.1 Світ JavaScript

Вводячи адресу сайту у стрічці браузера, клієнт тим самим надсилає запит на сервер та отримує відповідь, яка типово містить в собі html-документ (html-код, який браузер може розпарсити для створення об'єктної моделі документа). Цей код також може включати стилі сторінки та посилання на js-файл або сам JavaScript код, який працюватиме на стороні клієнта. І це є поясненням популярності мови JavaScript у сучасності: цей код може виконуватись на сторінці, змінюючи контент не надсилаючи при цьому нових запитів до сервера та не перезавантажуючи сторінку, що створює так званий гарний User Experience – позитивне враження від користування застосунком. Щоразу спостерігаючи відкриття моделі або нового вікна на сайті, чи коли сторінка містить динамічні елементи та зміни у реальному часі, існує великий шанс того, що за це відповідає JavaScript код. Тобто клієнт не отримує нову сторінку з сервера, а частини сторінки змінюються завдяки коду. Це можуть бути стилі або ж навіть додавання чи прибирання певних елементів об'єктної моделі документу. Також є ще один паттерн розробки, за якого весь фронтенд керується JavaScript кодом, створюючи Single Page Application(односторінковий застосунок).

У центрі світу JavaScript знаходиться так званий "vanilla" JavaScript-чиста мова програмування, без додаткових інструментів. JavaScript версії ES5 - це мова, підтримувана більшістю браузерів, проте написання великої кількості логіки для сторінки без використання додатків буде довшим, заплутаним та складнішим. Для спрощення роботи розробника були створені такі інструменти як бібліотеки – сторонні пакети, що містять функції, класи та об'єкти, що можна використати для полегшення написання програми. Автори бібліотек вже написали оптимізовані методи, які можна використати для того щоб досягти тієї ж мети меншою кількістю коду, дозволяючи сфокусуватись на бізнес логіці застосунку. Прикладами бібліотек є jQuery та LoDash.

Проте навіть при використанні бібліотек може стати вкрай важко маніпулювати станом застосунку і це одна з найважливіших проблем яку вирішують фреймворки. Яка ж різниця між цими двома інструментами? Бібліотеки зазвичай вирішують якусь одну задачу, наприклад jQuery має багато функціоналу для роботи з DOM, а LoDash містить множину утиліт таких як робота з масивами чи створенням унікальних ідентифікаторів об'єктів, але те як буде реалізований застосунок вирішує розробник. Натомість фреймворки дають більше ніж просто допомогу у реалізації методів, вони диктують структуру написання програми та керують багатьма низькорівневими деталями. Наприклад додавання елементів до об'єктної моделі документа є типовою задачею, що виконується фреймворком автоматично.

Світ JavaScript містить також інструменти побудови проекту, такі як Babel та webpack – пакети, що використовуються виключно під час розробки застосунку та які дозволяють зменшити, оптимізувати код, збирати докупи окремі файли або ж вмикати функції, використання яких інакше було б недоступним. Варто також зауважити, що існують інші версії мови JavaScript, серед яких TypeScript та ES6. Єдиною версією JS яка використовується більшістю браузерів є ES5 та щоб перетворити код нової версії у зворотну сумісну версію і потрібен транскомпілятор Babel. Таким чином під час написання коду можна використовувати ECMAScript 2015+ (ES6) з усіма його перевагами, після чого Babel інтегрує його у webpack, який керує всім процесом побудови, та транспілює ES6 у ES5. Typescript – це не нова версія JS, а нова мова, яку розробила компанія Microsoft. Вона є нарощенням над JavaScript та додає нові можливості, які не виконуються браузером і тому мають бути скомпільовані іншим компілятором та вбудовані у робочий процес. Серед таких можливостей варто виділити типи, оскільки JS характеризується динамічною типізацією, тобто оголошена змінна може змінювати тип даних які містить у собі в процесі розробки. Це може бути корисним, проте також може зробити код непередбачуваним. Typescript, у свою чергу, додає статичну типізацію, тож стає очевидним які дані буде містити в собі та чи інша змінна.

2.2 Огляд фреймворків

Перед початком розробки застосунків потрібно ознайомитись з підходами до використання фреймворків для їх створення та визначити передумови з боку JavaScript.

Існує два шляхи використання фреймворків. Перший з них – Multi Page Application: за такого підходу, на кожен шлях (кожну URL) повертається окрема html сторінка, зрендерена сервером. Додавання такого фреймворку як Vue.js може посилити окремі сторінки, для чого потрібно буде додати імпорт до кожної з них. Узагальнюючи, використання багатосторінкового застосунку з додаванням фреймворку передбачає створення окремих екземплярів з кодом для кожної з сторінок, де він має бути застосований. Проте не кожен з фреймворків дозволяє використати такий підхід. Альтернативою є створення Single Page Application, коли сервер повертає лише одну сторінку для всіх URL які можуть бути до нього надіслані. Справа в тому, що дана сторінка не міститиме контенту, в ній буде заключений JavaScript застосунок, набагато більший, ніж кожен з тих що був би доданий до окремих сторінок. Відтак увесь фронтенд буде керуватись за допомогою JavaScript, а не лише окремі його частини. Для користувача це виглядатиме так, ніби він отримуватиме різні сторінки, проте зміни відбуватимуться за рахунок коду, а сама сторінка не буде оновлюватись. За рахунок цього усі дії відбуватимуться швидше, оскільки не потрібно надсилати новий запит для кожного шляху і чекати на отримання нової сторінки. І навіть у випадках, коли необхідно отримати або записати дані у базу, наприклад, JavaScript виконуватиме це “за кулісами”, що знову ж таки не змушує користувача залишати односторінковий застосунок. Прикладом MPA є Facebook, а SPA – сервіс Gmail.

Перевагами використання багатосторінкового застосунку є SEO-оптимізація. На прикладі Facebook зрозуміло, чому було обрано такий підхід. Тому що розробники хотіли щоб кожну з сторінок можна було знайти за допомогою пошукових двигунів, а для цього кожна з них має бути проіндексована. Сканувальникам Google набагато важче обробляти сторінки з

великою кількістю JavaScript коду, і звісно ж вони не можуть передбачити як програмно зміниться кожен шматок односторінкового застосунку, а значить MPA є кращим рішенням у випадку, коли не лише головна сторінка має бути знайдена користувачем. Ще одним плюсом є те, що цей підхід старіший, а отже існує багато шаблонів, посібників, готових розробок, а також рішень з питань безпеки веб-ресурсу. Також є великий шанс, що навіть з вимкненим Javascript користувач все одно отримає велику кількість даних сторінки, чого не скажеш про застосунок який повністю керується кодом. Серед недоліків MPA варто виділити затримку, оскільки перезавантаження сторінки по кожному кліку навряд чи робить користування сайтом зручнішим, особливо у мобільних версіях.

Головною перевагою односторінкового застосунку є висока швидкість реагування на дії користувача. Очевидно, раз не потрібно очікувати на відповідь після кожного кліку, вся взаємодія відбувається майже миттєво, що викликає гарні враження у клієнтів. Також можна відмітити краще відокремлення бізнес логіки та логіки представлення, використання SPA буквально змушує розробника явно розмежувати дії фронтенду та бекенду. Великим плюсом є можливість підтримки веб-ресурсу за тимчасової відсутності підключення до інтернету, адже вся сторінка завантажується одразу і неробочими будуть лише частини які відповідають за запити до сервера. Найбільшим недоліком є перевага альтернативного підходу – оптимізація пошукових двигунів. Якщо це є критичним для застосунку, використання однієї сторінки може бути важче, проте не неможливо. Нові інструменти які допомагають вирішити цю задачу з’являються з кожним днем, вони використовують попередньо зрендерені частини застосунку на сервері, щоб Google сканував те ж, що побачить користувач. Такі технології лише починають з’являтись, тож оптимізація для цього підходу є важчою. Додатково можна перерахувати недоліки у вигляді можливих проблем з безпекою, оскільки про цей патерн інформації доступно набагато менше, а

також те, що застосунок не працюватиме без увімкненого JavaScript, хоч і відсоток таких користувачів складає менше 2%.

Перед початком розробки потрібно визначити які передумови має кожен з фреймворків з точки зору JavaScript, а саме яка мова необхідна для роботи з ними. Vue.js цілком добре працює з використанням і ES5, і ES6, а налаштувавши робоче середовище за допомогою додаткових інструментів можна легко використовувати всі допоміжні функції з поділу коду у різні файли, мінімізації чи покращених методів. Використання TypeScript також можливе, проте офіційної підтримки воно не має. Наступним фреймворком є React.js і він також добре працює з ES5, але це не кращий підхід до створення застосунку. Він не має такої великої кількості документації, як ES6, та принцип цього фреймворку «Everything is JavaScript»(«Усе є JavaScript») передбачає використання найновіших та найзручніших версій цієї мови. TS у свою чергу теж можна застосувати, проте як і з попереднім фреймворком, неофіційно. З Angular все абсолютно інакше, теоретично тут можна використати ES5 чи ES6, але він не буде працювати з ними так добре, як з TypeScript. Але для виконання в браузері, TypeScript в результаті все одно компілюється у JavaScript, тож назріває питання, чому не писати одразу на JS? Тому що Angular використовує дуже багато особливостей, які він пропонує – класи, типи, інтерфейси та декоратори. Пропонується інструмент Angular CLI за допомогою якого дуже легко налаштувати увесь робочий процес, тож не складає ніякого додаткового навантаження почати роботу з даним фреймворком.

В якості передумов також слід зазначити хто стоїть за створенням обраних інструментів, оскільки це частково свідчить про перспективи їх розвитку. Розробником Angular стала внутрішня команда Google, React завдячує своїй появі Facebook, а для Vue.js це не компанія, а команда колаборatorів які виявили бажання доєднатись до ініціативи Евана Ю.

Кожен фреймворк має так звану філософію, загальну характеристику принципів роботи. Angular – це масштабний фреймворк з великою кількістю вбудованих можливостей, таких як роутинг, валідація форм, обробка http-

запитів, завдяки чому багато користувачів називають його скоріше платформою, ніж фреймворком. Для нього існує ціла екосистема, з власним інтерфейсом командного рядка, прогресивною підтримкою застосунку та багатьма іншими інструментами, яку розробляє офіційна команда. Таке рішення може виглядати перевантаженим, проте воно досить часто виключає необхідність шукати сторонні пакети для реалізації функціональності, оскільки майже все потрібне можна знайти у вбудованих можливостях.

React у свою чергу має зовсім протилежний підхід. Він є мінімалістичним та сфокусованим на побудові інтерфейсу користувача, а також позиціонується скоріше як бібліотека, ніж фреймворк та має набагато менше вбудованих особливостей. Якщо розробка вимагає додавання роутингу чи обробки http-запитів, то потрібно буде шукати додаткові пакети.

Vue є чимось середнім між двома попередніми фреймворками, з одного боку він пропонує вбудовані можливості як Angular, проте не всі з них, фокусуючись на тих речах, які необхідні для написання коду – роутинг та робота з станом застосунку, а з іншого залишає простір для вибору доступних рішень у інших речах, таких як валідація форм, наприклад. Можна сказати що Vue це більше ніж React, але менше за Angular. Сфокусований на коді, даний інструмент не перевантажений великою масою вбудованих можливостей, проте пропонує потужний інтерфейс командного рядка та деякі вбудовані особливості.

Розділ 3. Опис реалізації програмного продукту

3.1 Аналіз технічного завдання

Головним завданням є надання порівняльної характеристики таким JavaScript фреймворкам як Angular, React та Vue.

Для проведення порівняння необхідно проаналізувати такі критерії:

- складність вивчення на початку та подальший розвиток;
- використання у багатосторінкових застосунках та можливість додання до існуючого додатку;
- розробка односторінкового застосунку;
- продуктивність;
- популярність на ринку вакансій.

Аби оцінити роботу фреймворків та зробити висновки про легкість чи складність їх освоєння, а також порівняти синтаксис та об’єм результуючих проектів було розроблено три односторінкових застосунки що демонструють роботу компонентів сторінки, роутингу, додавання елементів за допомогою поля введення, видалення елементів, роботу з API.

3.2 Установка засобів розробки

a. 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
```

b. React.js

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.development.js"></script>
Та друге, для компоненти ReactDOM:
<script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
Або ж за допомогою пакету Create React App командою: npx create-react-app <Назва проекту>.

c. Vue.js

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/dist/vue.js"></script>
Або версія посилання для подального використання, з оптимізацією розміру та швидкості: <script src="https://cdn.jsdelivr.net/npm/vue@2"></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.

3.3 Опис розробки програми

a. Angular застосунок

Після встановлення Angular CLI стає доступною команда ng, за допомогою якої можна створювати як сам проект, так і окремі сервіси чи компоненти. Отже, команда ng new angular-course-work створить проект з назвою «angular-course-work», а команда ng serve у папці проекту запустить сервер розробки, який захостить його та дасть адресу, за якою можна його відвідати (http://localhost:4200/ Ця команда також виконає всі необхідні компіляції та оптимізації, щоб код міг виконуватись у браузері. Створений командою проект містить папку для end-to-end тестування, папку з Node.js залежностями, основну папку з файлами для початку роботи та файли конфігурацій, такі як package.json, tsconfig.json, та інші. Важливо знати, що index.html файл – це та

сама одна сторінка, яка буде містити весь застосунок, проте в процесі розробки вона не буде зачіпатись, оскільки все відбувається у інших файлах, а main.ts файл є тим, що виконується в першу чергу, запускаючи весь проект за допомогою коду : platformBrowserDynamic().bootstrapModule(AppModule). AppModule – це клас, без ніякого контенту окрім імпортів та секції NgModule (рисунок 3.3.1) з декораторами - підсилювачами до інших TypeScript конструкцій, таких як класи, методи та властивості. Синтаксис декораторів дозволяє приєднати додаткову інформацію до класу, що перетворює його у модуль.

Рисунок 3.3.1 AppModule.ts

На прикладі AppModule (рис. 3.3.1) можна продемонструвати, що секція declarations зберігає інформацію про те, які компоненти використовує цей модуль, а bootstrap список виділяє кореневий компонент, який містить усі інші та відображатиметься на старті. У декораторі AppComponent є поле selector: 'app-root', який схожий з класом стилів, за допомогою якого цей кореневий компонент викликається у index.html файлі. Завдяки цьому всі майбутні компоненти будуть додаватись до app.component.html файлу, замість index.html.

Для імплементації зміни компонентів за шляхами використовується RouterModule, імпорт якого потрібно додати до списку imports декораторів класу AppModule, а параметром є список шляхів routes (рисунок 3.3.2). Використання директиви <router-outlet></router-outlet> вказує де потрібно

відобразити поточний шлях. Для вказання посилань на визначені шляхи використовується властивість routerLink(рисунок 3.3.3).

```
const routes: Routes = [
  { path : '', component: AboutComponent},
  { path : 'blog', component: AboutComponent},
  { path : 'todo-list', component: AboutComponent},
];
@NgModule({
  declarations: [
    AppComponent,
    TodolistComponent,
    BlogComponent,
    AboutComponent
  ],
  imports: [
    BrowserModule,
    RouterModule.forRoot(routes)
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Рисунок 3.3.2 AppModule.ts роутинг

app.component.html | main.ts | package.json

```
<div>
  <a routerLink="/">About</a>
  <a routerLink="/blog">Blog</a>
  <a routerLink="/todo-list">TodoList</a>
</div>
<router-outlet></router-outlet>
```

Рисунок 3.3.3 app.component.html роутинг

Angular має директиви – спеціальний синтаксис інструкцій, які використовуються для відображення контенту з умовами, наприклад, або спрощення виведення списку. Серед них найчастіше використовуються вбудовані директиви NgIf, NgFor та NgModel. Наприклад, статті відображені на сторінці блогу (рисунок 3.3.4) відображаються за допомогою *ngFor, для чого шаблон для однієї статті загорнутий у div з визначенням функції проходження по списку.

Для поля вводу потрібно використати декоратор @Input(), що позначатиме, що змінна може змінюватись ззовні. А щоб визначити спеціальні події, наприклад відмітка зробленого завдання, використовується EventEmitter

та метод emit(). Відмінним від інших фреймворків викликом обробки кліку з шаблону є (click)=””.

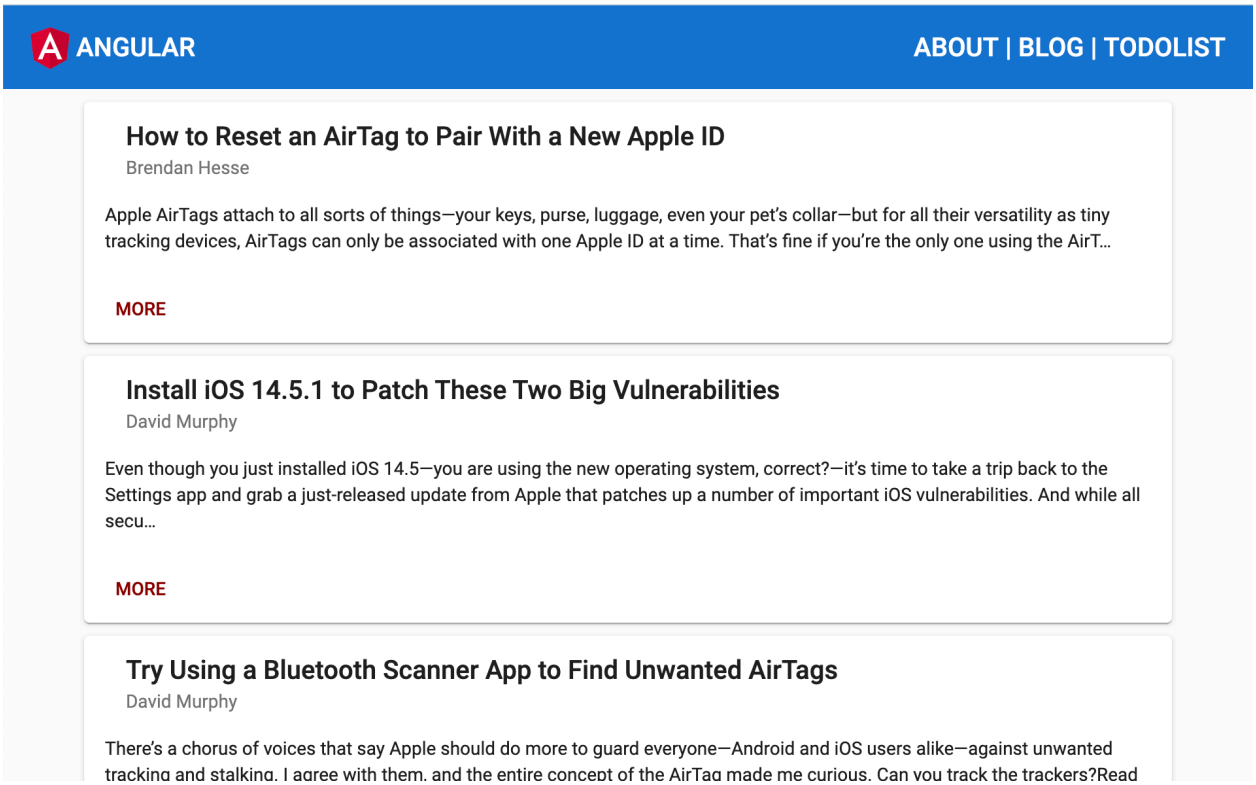


Рисунок 3.3.4 Сторінка новин News API

b. React застосунок

За допомогою пакету Create React App було створено стартовий проект, а за допомогою команди npm start за адресою http://localhost:3000/ можна побачити логотип React та посилання на офіційний сайт. Всередині початкового проекту міститься package.json файл з необхідними налаштуваннями, папка public містить файли що обслуговується сервером, такі як index.html файл, а також src папка з main.js файлом, який імпортує App.js – кореневий компонент та файли стилів.

Обов’язковою частиною односторінкового застосунку є роутинг, що забезпечує імітацію зміни сторінок. Оскільки React не має вбудованого для цього інструменту, потрібно його додатково встановити командою : npm install react-router-dom. Щоб використати ReactRouter, потрібно замінити JSX div всередині App() функції на виклик <Router/> з описаними шляхами. Також важливим нюансом є необхідність імпорту React у кожному js файлів, інакше

при використанні роутингу виникне помилка рендеру. На відміну від інших фреймворків, при визначенні шляху «/» та «/about» на сторінці будуть відображатись обидва компоненти, оскільки «/» є частиною іншого шляху, що можна виправити додавши exact до визначення(рис. 3.3.5).

```
import './App.css';
import {BrowserRouter as Router, Route} from 'react-router-dom';
import React from "react";
import About from "./components/About";
import Todo from "./components/Todo";
import Blog from "./components/Blog";

function App() {
    return (
        <Router>
            <div>
                <Route exact path="/" component={About}/>
                <Route path="/todo" component={Todo}/>
                <Route path="/blog" component={Blog}/>
            </div>
        </Router>
    );
}

export default App;
```

Рисунок 3.3.5 App.js шляхи React Router

Варто також у випадку деплою програми конфігурувати повернення index.html сторінки у випадку 404 помилки, коли сервер не розпізнає введений шлях.

У React немає такої легкої реалізації оновлення елементу, як у Angular чи Vue, та навіть заключивши змінну у фігурні дужки можна лише визначити її за допомогою змінної, далі ж вона буде незмінною. Для оновлення елементу потрібно повторно викликати функцію render. В управлінні станом програми допомагають компоненти, їх властивості(props) та функція setState. Наприклад, для зміни додавання нового елементу у список за натиском кнопки, потрібно визначити конструктор компоненти та зберегти у змінній state початкове значення масиву, а у функції що буде викликана кнопкою змінити її вміст за допомогою setState (рис.3.3.6). Зі зміною стану, React порівнюватиме стару та нову об’єктну модель та заново відобразить сторінку.

```
class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      todos: ['Cooking', 'Sports'],
      newTodoInput: '',
      todoWasDeleted: false
    };
  }

  addTodo() {
    const oldTodos = this.state.todos;
    this.setState({
      todos: oldTodos.concat(this.state.newTodoInput)
    });
  }
```

Рисунок 3.3.6 Робота зі станом програми React

Відображення списку також має свою особливість, проте на відміну від інших фреймворків вона не у спеціальних скороченнях, а у ключах елементів списку. Для того щоб React зміг оновити об’єктну модель належним чином, не оновлюючи весь список за кожної зміни, потрібно призначити ключі – унікальні ідентифікатори. Для самого ж проходження по списку використовується звичайна JavaScript функція map, а для моніторингу подій поля введення чи натискання кнопки використовуються функції onChange та onClick.

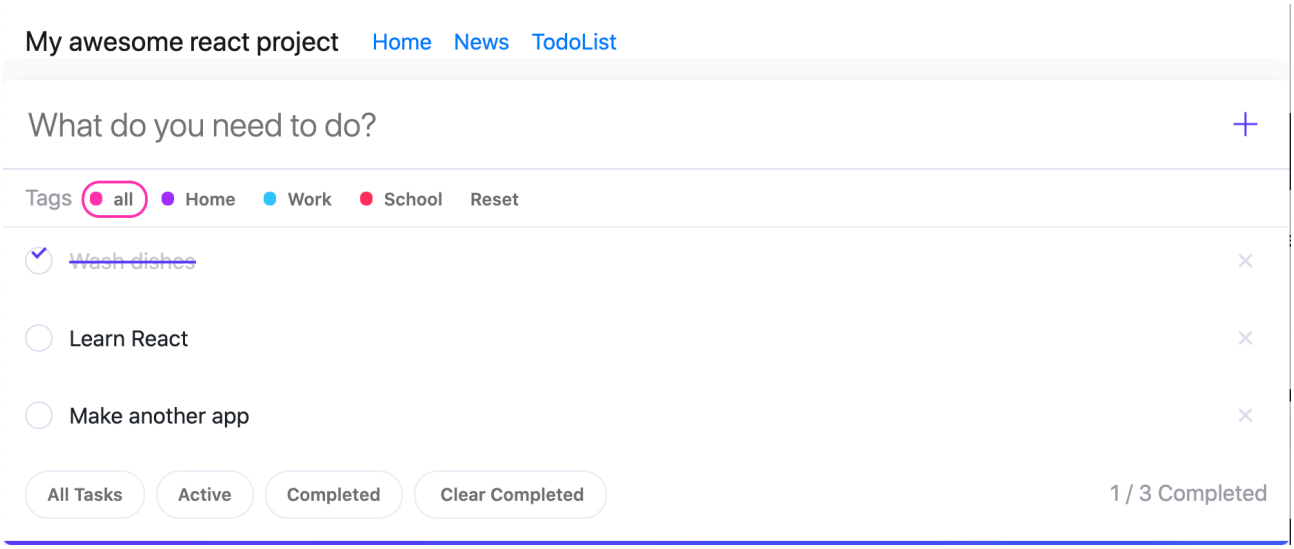


Рисунок 3.3.7 Поле введення та відображення списку Todo

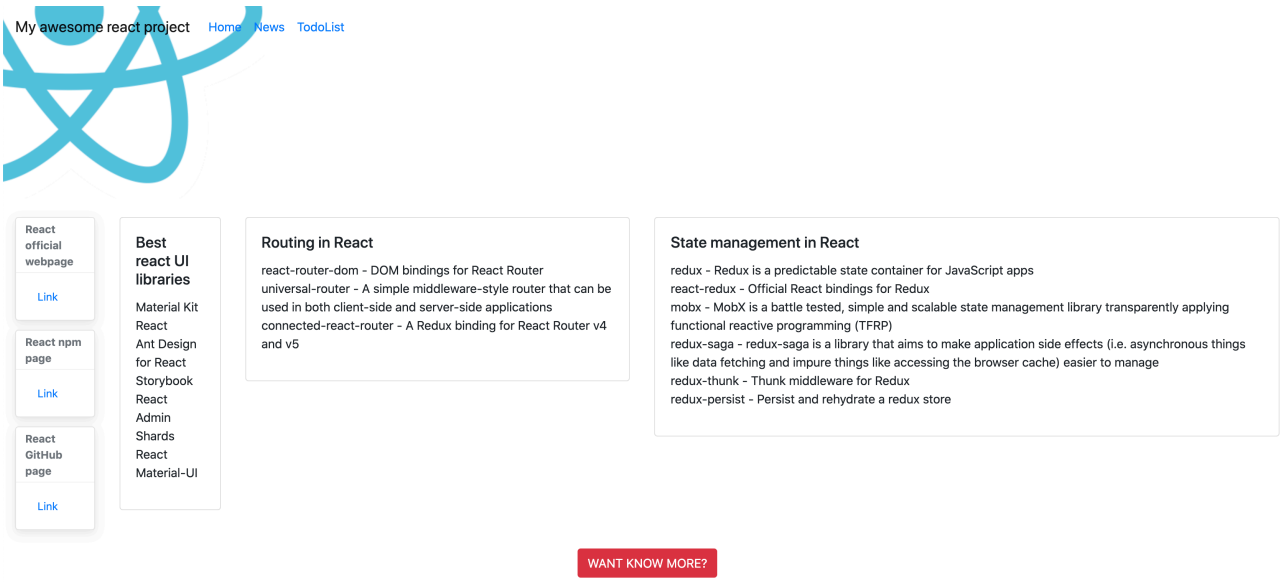


Рисунок 3.3.8 Home сторінка React

c. Vue застосунок

Для створення початкового проекту було використано Vue CLI та команду vue create <Назва проекту>. Всередині знаходитиметься main.js файл, який відтворює все що є у кореневому компоненті App.vue, а також папку з додатковими компонентами та файли конфігурацій package.json, babel.config.js, та інші потрібні для запуску та розробки налаштування. Файли з розширенням .vue містять чітко розмежовані секції: <template> з html-шаблоном яка є єдиною обов'язковою секцією, <script> з експортом об'єкту та кодом, щоб до нього мали доступ інші компоненти, та <style> зі стилями. В результаті, такі шаблони та скрипти перетворяться на JavaScript, а стилі будуть включені до фінального index.html файлу. Для використання додаткових компонент всередині App.vue або інших, потрібно просто додати імпорт у <script> секцію. Також можливе використання локальних компонент, які не можуть бути використані глобально, а лише всередині якогось іншого, для чого потрібно включити цей локальний компонент у перелік властивості components глобального, визначивши пару ключ-значення з селектора та назви компоненту.

Для переходу від однієї згенерованої стартової сторінки проекту до односторінкового застосунку потрібно встановити пакет Vue Router командою npm install –save vue-router. Це офіційний пакет команди Vue, який дозволяє

додати функціональність роутингу. Додатково потрібно явно вказати Vue використовувати Router, визначити список шляхів routes, ініціалізувати об'єкт VueRouter та додати його у властивості об'єкту Vue(рис. 3.3.9). Але це лише додає функціонал, а для того щоб додати роутер на сторінку у шаблоні App.vue (рис. 3.3.10) необхідно вказати <router-view></router-view>, за допомогою чого буде відображатись компонент відповідно до введеного шляху. Для функціоналу навігації, тобто переключення шляхів не вручну, а за посиланнями, використовується інша компонента, пропонована пакетом Vue Router - <router-link/> (рис. 3.3.11) .

main.js

```
import Vue from 'vue'
import Router from 'vue-router'

import App from './App'
import Home from './views/Home'
import News from './views/News'
import Todos from './views/Todos'

Vue.use(Router);

const routes = [
  { path: '/', component: Home },
  { path: '/news', component: News },
  { path: '/todos', component: Todos }
];

const router = new Router( options: {
  mode: 'history',  base: '/',  routes
});

new Vue({
  router,
  render: h => h(App)
}).$mount( elementOrSelector: '#app');
```

Рисунок 3.3.9 Vue Router у main.js

App.vue

```
<template>
  <div id="app">
    <navigation />
    <router-view></router-view>
  </div>
</template>
```

Рисунок 3.3.10 Шаблон App.vue

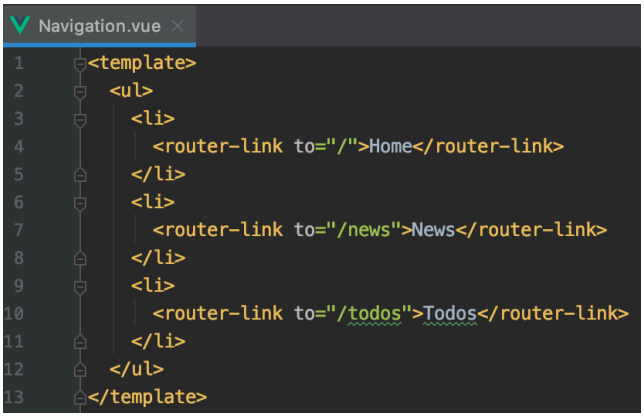


Рисунок 3.3.11 Шаблон навігаційної панелі Navigation.vue

Важливим пунктом синтаксису Vue.js є скорочення, такі як v-on, v-bind, v-for, v-if, v-model. Наприклад, для відображення списку новин з NewsAPI варто використати v-for, що застосує шаблон описаний у News.vue до кожного елементу списку новин. V-if використовується для умовного відображення, наприклад фільтрації виконаних завдань на сторінці TodoList, v-on служить командою для прослуховування подій, таких як натиск на кнопку додавання введеного завдання до списку. V-model дозволяє налаштувати двосторонню прив’язку, завдяки якій змінна буде динамічно змінювати своє значення.

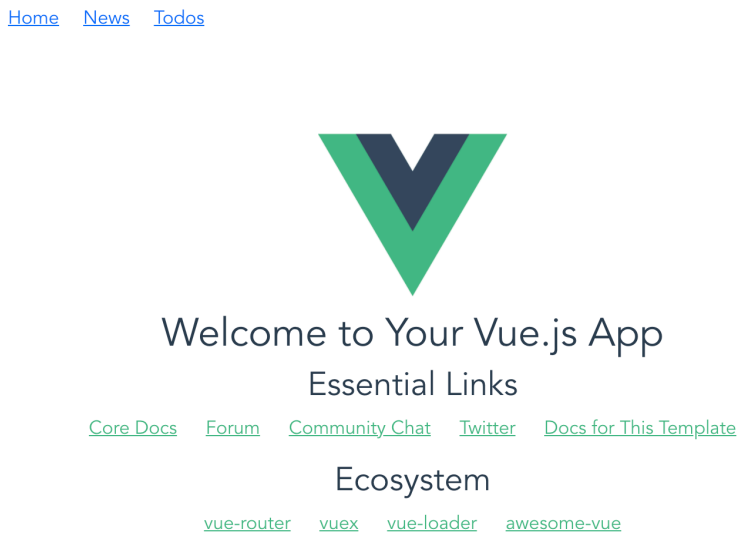


Рисунок 3.3.12 Головна сторінка Vue

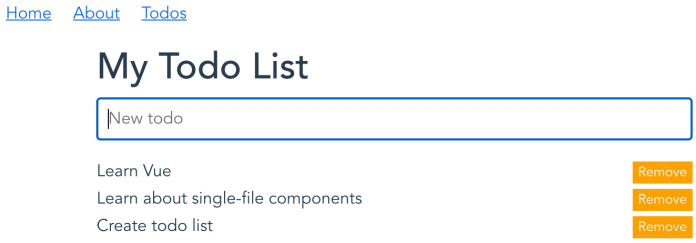


Рисунок 3.3.13 Список задач Vue

3.4 Порівняльна характеристика

Для досягнення мети даного дослідження необхідно провести порівняння за наступними критеріями: загальний огляд принципів роботи та синтаксис, складність початку освоєння фреймворків та подальшого вивчення, розробка багатосторінкових та односторінкових застосунків, оптимізація та продуктивність, популярність та вакансії.

У другому розділі було розглянуто загальні філософії кожного з фреймворків. Angular є масштабною платформою з багатьма вбудованими функціями та допоміжними інструментами, React – мінімалістичною бібліотекою з великою громадською підтримкою, а Vue чимось середнім, сфокусованим на написанні коду та лише необхідних для цього речах. Усі розглянуті фреймворки несуть у собі ідею побудови інтерфейсу користувача за допомогою комбінування багаторазових компонентів. Це основна ідея, яку має кожен з них, хоч синтаксис і відрізняється. Для Angular та Vue має бути чітке розмежування html шаблонів з інструкціями які фреймворки можуть зрозуміти, наприклад відображення списку чи умовний вираз, та js/ts файлів, що маніпулюють даними, коли це потрібно. У React не має такого розподілу між html та JavaScript, замість цього компоненти є частинами js-функцій. Він має особливий синтаксис що називається JSX і використовується для того щоб описати який html контент React має відобразити у DOM відповідно до заданих

умов. Отже, для React використовується суміш JavaScript та HTML, а Vue та Angular чітко розмежовують відображення та скрипт.

Розглянувши загальну характеристику фреймворків, можна приблизно оцінити як буде відбуватись освоєння кожного з них початківцем. Звісно, даний критерій залежить від початкових знань Javascript та досвіду роботи з бібліотеками у минулому, тому вважатимемо початківця людину з базовим знанням мови програмування та малим досвідом додаткових інструментів. Відкинувши особисті вподобання, можна припустити, що крива навчання Angular буде досить крутою. Перш за все тому що налаштування проекту є досить складними, не можна просто почати писати код на TypeScript у будь-якому веб-проекті, потрібно створити новий проект з необхідними налаштуваннями та за потреби оптимізувати під нього існуючий код. Інтерфейс командного рядка Angular робить даний процес простішим, але може бути незвичним для початківця, який звик додавати імпорт нових пакетів на веб сторінку. Додатковою складністю може бути необхідність вивчення TypeScript, так само як розуміння екосистеми Angular, його синтаксису та принципу роботи. Але розділення частин логіки та представлення позбавляє необхідності вивчати їх суміш, лише інструкції які потрібно додати у шаблони компонент для роботи з ними. React матиме схожу криву, проте менш різку на початку, оскільки окрім знання JavaScript більше нічого не потрібно і здається що в даному випадку відсутні складні налаштування, але це не зовсім так. Для можливості використання JSX синтаксису потрібна більш складна конфігурація проекту, з чим також можуть допомогти додаткові інструменти. За рахунок того що потрібно призвичаїтись до комбінації JavaScript та HTML та того, як працює сам React, освоєння може видатись так само не надто легким, як і з Angular. Vue завдячує своїй популярності насамперед думці більшості користувачів про те, що він легкий у вивченні. Складні налаштування проекту потрібні лише у випадку коли застосунок великий та розробка потребує просунутих можливостей, в іншому випадку можна додати імпорт на сторінку та почати використовувати Vue. Він не потребує окремої мови чи особливого

синтаксису, лише JavaScript, а розділення шаблону та скрипта є цілком зрозумілим.

Далі слід розглянути наскільки легко використовувати кожен з фреймворків у малих застосунках або на сторінках MPA. Для цього варто відповісти на наступні запитання:

a. Чи можна почати роботу з фреймворком зі стрічки імпорту?

Для Angular відповідь буде ні, потрібні додаткові налаштування, компіляція TypeScript, яка не може відбуватись прямо у браузері. Він не створений для багатосторінкових застосунків та не буде ефективним для малих додатків. А Vue та React мають позитивну відповідь, можна використати CDN посилання чи імпорт з офіційної сторінки та почати використання фреймворків.

b. Чи підходить інструмент для розробки багатосторінкового застосунку?

Повторно, Angular зовсім не підходить для даних цілей, він містить надто багато коду для простих задач. Натомість Vue є чудовим рішенням для такого завдання, легко контролює та змінює малі шматки об’єктної моделі. З React відповідь також позитивна, проте це не дуже просто виконати, оскільки зі збільшенням кількості сторінок доведеться налаштовувати складніше середовище, збільшувати кількість одних і тих самих включень на сторінках та дублювати код.

c. Чи вимагає робочий процес складних налаштувань?

Так, для Angular незалежно від об’єму проекту необхідно налаштовувати повноцінне середовище, потрібно працювати з TypeScript, збирати всі розділені файли та оптимізувати результуючий пакет. Для Vue – ні, це може дати свої плюси, додаткові можливості, проте не є вимогою. Для React формально це не потрібно, проте є хорошою практикою та працює набагато краще з використанням ES6, Babel, webpack, тобто з налаштованим робочим процесом.

d. чи містить фреймворк багато накладного навантаження?

І знову так, навіть з усіма оптимізаціями та викиданням невикористовуваних функцій, концепція Angular з усіма модулями, пакетами є надмірною для контролю маленької частини однієї сторінки. З іншого боку Vue залишається

досить невеликим навіть зі своїм вбудованим функціоналом. А React є найбільш мінімалістичним серед них, настільки, що розробники самі називають його скоріше бібліотекою ніж фреймворком та є створеним саме для управління малими частинами DOM.

Отже, можна з впевненістю сказати що для розробки простого або багатосторінкового застосунку Vue буде відповідати всім вимогам та буде найкращим рішенням. Angular зовсім не створений для такого підходу, а React може бути впроваджений, проте з деякими складнощами у налаштуванні.

Протилежним підходом та критерієм є використання фреймворків у застосунках які повністю керуються мовою JavaScript. Тут потрібно зважати на легкість управління великою кількістю коду, необхідність використання сторонніх бібліотек, можливості оптимізації та управління станом. Для Angular існує велика кількість рекомендацій з неймінгу компонентів, файлів, їх структури, а також наявний Angular CLI який дуже допомагає у організації проекту, тож орієнтуватись у великій множині скриптів досить не складно. Vue хоч і має інтерфейс командного рядка, але він не є офіційним інструментом та не містить рекомендацій, тож можуть виникнути складнощі з підтримкою коду у великих застосунках. Така ж ситуація з React, доступні рекомендації від спільноти, але потрібно самостійно пробувати та визначати найзручніший підхід для організації застосунку. Щодо сторонніх бібліотек, команда Angular максимально забезпечила цей фреймворк інструментами, що можуть знадобитись для розробки, а також є багато офіційних додаткових пакетів, як от готові стилізовані компоненти Angular Materials, валідація форм для ng-form чи вбудований роутинг. Спільнота теж розробляє багато цікавих та корисних пакетів, проте звертатись до них користувачам Angular потрібно набагато рідше ніж іншим фреймворкам. Vue зі свого боку потребує більшої уваги сторонніх бібліотек, він має свій пакет з роутингом та VueX для управління станом, проте такі речі як валідації форми йому не доступні. React містить найменшу кількість функціоналу, тож для великих застосунків потрібно буде обирати допоміжні бібліотеки для стану, Redux чи MobX, включати ReactRouter для

роутингу. Потрібно пам’ятати що сторонні інструменти не є офіційними пакетами та не гарантують підтримку функціоналу робочим та оптимізованим. Також їх теж потрібно обирати, оскільки існує багато альтернатив які вирішують одну і ту саму задачу, що може бути складно, особливо для новачків які тільки починають знайомитись з фреймворком. Також перевага Angular у великій кількості оптимізаційних інструментів і за бажання можна самостійно спробувати покращити продуктивність. Vue та React самостійно оптимізують існуючий код при створенні пакету і майже не мають шляхів стороннього впливу на цей процес. Останнім компонентом цього критерії є легкість управління станом програми, оскільки у великих застосунках це може бути проблематично. Angular використовує сервіси для цієї потреби, але зі збільшенням застосунку з’являється потреба використовувати сторонню бібліотеку NgRx – вона потрібна для управління станом програми за допомогою спостерігачів та потребує додаткового вивчення. Для Vue існує офіційний пакет VueX, який чудово інтегрується у програму та спрощує керування. React має декілька неофіційних пакетів, які цілком справляються з цією задачею, але вимагають вибору та додаткового вивчення. Отже, Angular створений для розробки односторінкових, складних застосунків, тож буде гарним рішенням. Vue та React не мають такого чіткого фокусу на даний підхід до використання, проте мають всі засоби для його реалізації.

Наступним критерієм є продуктивність: стартова, що стосується запуску застосунку та продуктивність виконання. Продуктивність запуску вимірюється тим, як швидко сторінка підвантажується, коли користувач її відвідує та як швидко стає інтерактивною. Це в першу чергу залежить від розміру пакету проекту, кількості коду яка має бути завантажена. Помилкою є оцінювання стартової продуктивності на малих застосунках, на кшталт «Hello world». Варто брати до уваги більші розробки та порівнювати їх. Тенденцією є те, що Angular проекти є трохи більшими, оскільки це все ж великий фреймворк і навіть при тому що функціональність яка не використовується не буде включена до фінального коду, все одно результуючий пакет зазвичай більший за інші. Але в

середньому стартова продуктивність усіх трьох фреймворків для великих застосунків є приблизно однаковою. Вони всі швидкі та користуючись рекомендаціями з оптимізації та розробки, проблем зі стартовою продуктивністю не виникає. Продуктивність виконання стосується відповідей на дії користувача і тут все залежить від коду розробника та частково коду фреймворку, як він обробляє зміни інтерфейсу, управляє реактивністю застосунку. У цих вимірах теж не буде конкретних переможців, усі фреймворки дуже добре справляються з даними задачами та різні тести також виділяють різних фаворитів. Так само як не варто орієнтуватись в розмірах елементарних проектів, не варто довіряти тестам продуктивності які не відповідають реальним умовам, наприклад відображення десяти тисяч рядків. Для окремих сценаріїв використання результати можуть відрізнятись, але правда в тому що усі три фреймворки мають чудову продуктивність як на старті, так і у роботі, тож не варто брати до уваги цей критерій при виборі між ними.

Наостанок варто додати інформацію про розвиток фреймворків Angular, React та Vue. Усі три є стабільними та готовими до використання, в той самий час знаходячись у фазі активної розробки. Це означає що вони все ще крок за кроком еволюціонують, до них додаються нові можливості, а застарілі поступово прибираються. Цей розвиток є повільним аби не залишити ніякий функціонал без підтримки, але в результаті більш потужні та швидкі фреймворки є вигідними як для користувачів, так і для розробників.

Висновки

У результаті даного дослідження можна навести наступні висновки, беручи до уваги постановку завдання курсової роботи.

Був проведений аналіз предметної області, стану розвитку екосистеми мови JavaScript. Незалежно від того що є безліч інструментів розробки, найбільшу популярність мають Angular, React та Vue. Щороку виходять десятки статей з рейтингом та порівняннями, що доводить актуальність теми та постійний розвиток фреймворків.

Були розглянуті різні критерії оцінки та порівняння інструментів розробки та описані підходи до їх використання. Функціонал був протестований за допомогою невеликих застосунків, що демонструють роботу роутингу, управління станом та використання повторюваних компонентів.

Було обґрунтовано вибір фреймворків та виконано порівняльну характеристику за обраними критеріями.

Список використаної літератури

1. Рейтинг State of JS 2020 [Електронний ресурс]
   Режим доступу до статей :
   https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/
2. Рейтинг Stackoverflow 2020 Developer Survey [Електронний ресурс]
   Режим доступу до статей :
   https://insights.stackoverflow.com/survey/2020#technology-web-frameworks
3. JetBrains DevEcosystem рейтинг JavaScript [Електронний ресурс]
   Режим доступу до статей :
   https://www.jetbrains.com/lp/devecosystem-2020/javascript/
4. Top JavaScript Frameworks and Tech Trends for 2021 [Електронний ресурс]
   Режим доступу до статей :
   https://medium.com/javascript-scene/top-javascript-frameworks-and-tech-trends-for-2021-d8cb0f7bda69
5. Офіційний сайт React [Електронний ресурс]
   Режим доступу до статей : https://reactjs.org/
6. Офіційний сайт Angular [Електронний ресурс]
   Режим доступу до статей : https://angular.io/
7. Офіційний сайт Vue [Електронний ресурс]
   Режим доступу до статей : https://vuejs.org/
8. Офіційний сайт Vue CLI[Електронний ресурс]
   Режим доступу до статей : https://cli.vuejs.org/

Додатки

Додаток А. Лістинг програмного коду Angular

Код класу post.model.ts

```
export interface Post {
  author: string;
  title: string;
  description: string;
  url: string;
  urlToImage: string;
  publishedAt: string;
  content: string;
}
```

Код класу blog.service.ts

```
import {Injectable} from '@angular/core';
import {Post} from '../models/post.model';
import {Subject} from 'rxjs';
import {HttpClient} from '@angular/common/http';
import {map} from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class BlogService {
  constructor(private http: HttpClient) {
  }

  private posts: Post[] = [];
  private postsUpdated = new Subject<Post[]>();
  private postsUrl = 'https://newsapi.org/v2/everything?' +
    'q=Apple&' +
    'from=2021-05-03&' +
    'sortBy=popularity&' +
    'apiKey=3ad3737508124403ae66cf29e56e7b63';

  getPosts() {
    return this.http
      .get<{ status: string; totalResults: any; articles: any }>(
        this.postsUrl
      )
      .pipe(map((postData) => {
        return postData.articles.map(post => {
          return {
            author: post.author,
            title: post.title,
            description: post.description,
            url: post.url,
            urlToImage: post.urlToImage,
```

```
          publishedAt: post.publishedAt,
          content: post.content
        };
      });
    }))
    .subscribe(transformedPosts => {
      this.posts = transformedPosts;
      this.postsUpdated.next([...this.posts]);
    });
  }
}
```

Код класу blog.component.html

```
<div class="posts-list">
  <div *ngFor="let post of posts">
    <mat-card class="blog-card">
      <mat-card-header>
        <mat-card-title>{{post.title}}</mat-card-title>
        <mat-card-subtitle>{{post.author}}</mat-card-subtitle>
      </mat-card-header>
      <mat-card-content>
        <p>
          {{post.description}}
        </p>
      </mat-card-content>
      <mat-card-actions>
        <button mat-button class="more-button">MORE</button>
      </mat-card-actions>
    </mat-card>
  </div>
</div>
```

Код класу app.component.html

```
<mat-toolbar class="toolbar">
  <img
    width="40"
    alt="Angular Logo"
    src="data:image />
  <span>Angular</span>
  <span class="spacer"></span>
  <mat-button routerLink="/"> About </mat-button> |
  <mat-button routerLink="/blog" >  Blog </mat-button> |
  <mat-button routerLink="/todo-list"> TodoList</mat-button>
</mat-toolbar>
<router-outlet></router-outlet>
```

Код класу app.component.ts

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

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',
```

```
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}
```

Код класу todo.model.ts
Код класу todo.service.ts

Додаток Б. Лістинг програмного коду Vue

Код класу App.vue

```
<template>
  <div id="app">
    <navigation />
    <router-view></router-view>
  </div>
</template>
<script>
import Navigation from "./components/Navigation";
import Home from './views/Home';
export default {
  name: "App",
  components: {
    navigation: Navigation,
    Home
  }
};
</script>

<style>
#app {
  font-family: "Avenir", Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  color: #2c3e50;
  margin: 2em;
}
</style>
```

Код класу News.vue

```
<template>
  <div id='about-1'>
    <div id='art-1'>
      <b-row class="article" v-for="i in this.articles">
        <b-col sm="2"></b-col>
```

```
      <b-col sm="8">
        <b-container fluid="lg" class="border">
          <h1>{{ i.author }}</h1>
          <h5>{{ i.title }}</h5>
          <b-card-text>
            {{ i.content }}
          </b-card-text>
          <b-button href="#" variant="primary">Click me!</b-button>
        </b-container>
      </b-col>
      <b-col sm="2"></b-col>
    </b-row>
  </div>
 </div>
</template>

<script>
 import axios from 'axios'
 export default {
  name: 'News',
  articles: null,
  el: '#art-1',
  data() {
    return {
      articles: null,
    }
  },
  async created() {
    var url = 'https://newsapi.org/v2/everything?' +
        'q=Apple&' +
        'from=2021-05-03&' +
        'sortBy=popularity&' +
        'apiKey=3ad3737508124403ae66cf29e56e7b63';
    this.articles = await axios.get(url, { responseType: 'json' }).then(response =>
response.data.articles)
    console.log(this.articles)
  }
 }


</script>

<style>
 .article {
  margin: 15px 0px 15px 0px;
 }
 .border {
  padding: 10px 20px 10px 20px;
  border-radius: 5px;
  border: 1px solid gray;
 }
</style>
```

Код класу Home.vue

```
<template>
  <div id="app">
    <img width="25%" src="../assets/logo.png" />
    <div class="hello">
      <h1>{{ msg }}</h1>

      <h2>Essential Links</h2>
      <ul>
        <a href="https://vuejs.org">Core Docs</a>
        <a href="https://forum.vuejs.org">Forum</a>
        <a href="https://chat.vuejs.org">Community Chat</a>
        <a href="https://twitter.com/vuejs">Twitter</a>
        <a href="http://vuejs-templates.github.io/webpack/">Docs for This Template</a>
      </ul>
      <h2>Ecosystem</h2>
      <ul>
        <a href="http://router.vuejs.org/">vue-router</a>
        <a href="http://vuex.vuejs.org/">vuex</a>
        <a href="http://vue-loader.vuejs.org/">vue-loader</a>
        <a href="https://github.com/vuejs/awesome-vue">awesome-vue</a>
      </ul>
    </div>
  </div>
</template>

<script>
export default {
  name: 'Home',
  data() {
    return {
      msg: 'Welcome to Your Vue.js App'
    };
  }
};
</script>

<style scoped>
#app {
  font-family: "Avenir", Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
h1,
h2 {
  font-weight: normal;
}
ul {
  list-style-type: none;
  padding: 0;
}
```

```
a {
  display: inline-block;
  margin: 0 10px;
}
ul > a {
  color: #42b983;
}
</style>
```

Код класу Navigation.vue

```
<template>
  <ul>
    <li>
      <router-link to="/">Home</router-link>
    </li>
    <li>
      <router-link to="/news">News</router-link>
    </li>
    <li>
      <router-link to="/todos">Todos</router-link>
    </li>
  </ul>
</template>

<script>
  export default {
    name: 'Navigation'
  }
</script>

<style scoped>
  ul {
    margin: 0;
    padding: 0;
  }
  li {
    display: inline-block;
    padding: 1em 0.5em;
  }
</style>
```

Додаток В. Лістинг програмного коду React

Код класу Header.js

```
import React, { useState, Component } from 'react';
import { Link } from 'react-router-dom'
import {  Navbar,  NavbarBrand,  Nav,  NavItem,  NavLink,  NavbarText} from 'reactstrap';
import { observer } from 'mobx-react'
@observer
class Header extends Component {
  state = {  }
```

```
 render() {
  return (
   <div>
   <Navbar light expand="md">
    <NavbarBrand href="/">My awesome react project</NavbarBrand>
     <Nav className="mr-auto" navbar>
      <NavItem>
       <NavLink>
        <Link to="/">Home</Link>
       </NavLink>
      </NavItem>
      <NavItem>
       <NavLink>
        <Link to="/tests">News</Link>
       </NavLink>
      </NavItem>
      <NavItem>
       <NavLink>
        <Link to="/todo">TodoList</Link>
       </NavLink>
      </NavItem>
     </Nav>
     <NavbarText></NavbarText>
   </Navbar>
  </div>
   );
 }
}

export default Header;
```

Код класу Posts.js

```
import React, { Component } from 'react'
import { observer, inject } from 'mobx-react'
import {
  Card, CardText, CardBody, Container,   CardTitle, CardSubtitle, Button, Row, Col} from
'reactstrap';
import API from '../../api/index'
const api = new API()

@inject('data')
@observer
class Posts extends Component {
 componentDidMount() {
  api.getData()
 }
 render() {
  let articles = this.props.data.newsData.articles
  return (  <>
     <div>
      {!this.props.data.isLoading && articles.map(i =>
      <Container className="themed-container" fluid={true} style={{ marginBottom: '10px' }}>
      <Row>
```

```
          <Col xs="6" sm="4"></Col>
          <Col xs="6" sm="4">
            <Card>
              <CardBody>
                <CardTitle tag="h5">{i.author}</CardTitle>
                <CardSubtitle tag="h6" className="mb-2 text-muted">{i.title}</CardSubtitle>
                <CardText>{i.content}</CardText>
                <Button>Learn more</Button>
              </CardBody>
            </Card>
          </Col>
          <Col xs="6" sm="4"></Col>
        </Row>
        </Container>
        )}
      </div>
    </>
  )
 }
}
export default Posts
```

Код класу App.js

```
import React from 'react'
import { Provider } from 'mobx-react'
import { BrowserRouter } from 'react-router-dom'
import { history } from 'stores/routing'
import { stores } from 'stores'
import 'styles/main.scss'

import Routes from 'view'

const App = () => (
 <Provider {...stores}>
   <BrowserRouter history={history}>
     <Routes />
   </BrowserRouter>
 </Provider>
)
export default App
```