НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет електроніки
Акустичних та мультимедійних електронних систем
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Найда С.А.
(підпис) (ініціали, прізвище)
“01” червня 2020 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності (спеціалізації) 171 Електроніка (Електронні та інформаційні системи і технології телебачення, кінематографії та звукотехніки)_____
(код і назва)

на тему: «Користувацький інтерфейс для роботи з аудіо -візуальним контентом» _____

Виконав: студент IV курсу, групи ДВ-61
(шифр групи)

Бабіч Сергій Олександрович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник_____старший викладач Батіна О.А, _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент доцент каф. ЕПС, к.т.н., доц. Катерина КЛЕН _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Студент _____ (Бабіч С.О.)
(підпис)

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут (факультет) Факультет електроніки
(повна назва)

Кафедра Акустичних та мультимедійних електронних систем
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 171 Електроніка Електроніка (Електронні та інформаційні системи і технології телебачення, кінематографії та звукотехніки)
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Найда С.А.
(підпис) (ініціали, прізвище) «25» травня 2020 р.

ЗАВДАННЯ

на дипломну роботу студенту

Бабіча Сергія Олександровича
(прізвище, ім'я, по батькові)

1. Тема роботи: «Користувацький інтерфейс для роботи з аудіо -візуальним контентом» ,
керівник роботи: Батіна Олена Анатоліївна, старший викладач ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «25» травня 2020 р. № 1196-с

2. Строк подання студентом роботи “01” червня 2020 року

3. Вихідні дані до роботи React, Redux, React Native, Angular, UI, Android, iOS, JavaScript.

4. Зміст роботи: Характеристика та застосування засобів розробки користувацьких інтерфейсів, історія еволюції можливостей браузерів та технологій побудови користувацьких інтерфейсів, покращення візуальної складової та спрощення роботи с користувацькими інтерфейсами.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) сучасний користувацький інтерфейс , програмні способи створення користувацького інтерфейсу, порівняння популярності додатків, таблиця порівняння додатків, приклади застосування. Презентація на 10 слайдах.

6. Консультанти розділів роботи

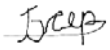
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «20» лютого 2020р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Написання першого розділу: “ 1 АНАЛІТИЧНИЙ ОГЛЯД ТЕХНОЛОГІЙ СТВОРЕННЯ КОРИСТУВАЦЬКИХ ІНТЕРФЕЙСІВ ”	01.04.2020-04.04.2020	
2	Написання другого розділу: “2 ТЕХНІЧНІ АСПЕКТИ СТВОРЕННЯ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ ”	16.04.2020-04.05.2020	

3	Написання третього розділу: “3 РОЗРОБКА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ ”	05.05.2020-11.05.2020	
4	Підготовка матеріалів до друку та оформлення пояснювальної записки	01.06.2020-03.06.2020	
5	Підготовка доповіді до захисту та оформлення плакатів	04.06.2020-07.06.2020	

Студент  Бабіч С.О.

(підпис) (ініціали, прізвище)

Керівник роботи  ст.викладач.Батіна О.

(підпис) (ініціали, прізвище)

УДК 621.397

РЕФЕРАТ

Дипломна робота: 71 с., 4 табл., 27 рис., 22 джерел.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, REACT ДОТАТОК, REDUX, СЕРЕДОВИЩЕ РОЗРОБКИ, ANGULAR, REACT NATIVE, UI.

Об’єктом дипломної роботи є огляд програмного забезпечення та засобів для створення користувацьких інтерфейсів, застосування додатків у різних галузях, а також дослідження еволюції розвитку технологій та браузерів для спрощення роботи з користувацьким інтерфейсом користувачам.

Мета роботи полягає у аналізі програмних засобів, за допомогою яких створюються користувацькі інтерфейси додатків та веб сайтів .

Методом дослідження є теоретичний та порівняльний аналіз сучасних технологій розробки користувацьких інтерфейсів.

В результаті виконання дипломної роботи розглянуто сучасні фреймворки для створення користувацьких інтерфейсів разних платформ, розглянуто їх характеристики, переваги та недоліки, приклади застосування в реальних проектах .

ABSTRACT

The object of the thesis is a review of software and tools for creating user interfaces, its applications in various fields, as well as research on the evolution of technology and browsers to simplify working with user interfaces for users.

The purpose of the work is to analyze the software tools used to create user interfaces for applications and websites.

The research method is a theoretical and comparative analysis of modern technologies for user interface development.

As a result of the thesis, modern frameworks for creating user interfaces of different platforms are considered, their characteristics, advantages and disadvantages, examples of application in real projects are considered.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

WEB	–	Collection of web pages ;
UI	–	User interface ;
URL	–	Uniform Resource Locator ;
MVC	–	Modal Vue Controller;
DOM	–	Document Object Model ;
CSS	–	Cascading Style Sheets ;
HTML	–	Hypertext Markup Language ;
SPA	–	Single page application ;
APP	–	Автоматизована система управління;

ВСТУП

Розробка користувацького інтерфейсу для додатків та веб сайтів є одним з найперспективніших напрямків сучасних IT-розробок. Даний напрямок направлений на легкість та доступність при використанні сервісу користувачем.

Користувацький інтерфейс – це набір засобів та правил, який об'єднує в собі компоненти програми за допомогою яких користувач має змогу спілкуватись з програмним забезпеченням чи системою.

Сучасний світ перейшов на технології спілкування з програмою через інтерфейс користувача, тому світова тенденція йдо до спрощення роботи с інтерфейсом для користувача.

На розробку користувацького інтерфейсу впливає багато чинників, які не пов’язані з програмуванням. Наприклад, при розробці користувацького інтерфейсу враховується та сфера, де буде використовуватись сервіс, час його на розробку, бюджет та команда розробників. Інтерфейс користувача може використовуватися в будь-якому електоронному приладі, починаючи від домофонів, закінчуючи інтерфейсом управління на космічних кораблях

Актуальність роботи полягає в тому, що протягом останнього часу дуже стрімко розвивається попит на веб технології і у споживача з кожним роком зростають вимоги до програмних засобів, які задовольняють його потреби. Тому у цій дипломній роботі розглянуто сучасні технології для реалізації користувацьких інтерфейсів: програмні засоби та їх застосування у різних сферах життя.

Метою роботи є аналіз та застосування фреймворків для створення користувацьких інтерфейсів, які дають можливість швидко поринути в середовище створення користувацьких інтерфейсів, а також їх практичне застосування у житті.

Для досягнення поставленої мети були сформульовані такі завдання:

1. Провести аналітичний огляд технологій для розробки користувацьких інтерфейсів;

2. Дослідити сучасні фреймворки;

3. Провести порівняльний аналіз основних засобів та фреймворків для створення користувацьких інтерфейсів;

1 АНАЛІТИЧНИЙ ОГЛЯД ТЕХНОЛОГІЙ СТВОРЕННЯ КОРИСТУВАЦЬКИХ ІНТЕРФЕЙСІВ

1.1 Поняття користувацького інтерфейсу

Користувацький інтерфейс (UI) – це набір засобів та правил, який об'єднує в собі компоненти програми за допомогою яких користувач має змогу спілкуватись з програмним забезпеченням чи системою.

До елементів користувацького інтерфейсу входять:

- набір задач користувача;
- елементи управління ;
- навігація між сторінками;
- візуальний дизайн екрану;
- відображувана інформація;
- зворотній зв'язок;
- взаємодія, транзакції між користувачами, та сервером.

Проектування UI на разі перетворилось на велику проблему яка по складності може бути важче ніж сама розробка програмного коду, і вимагає, як і процес проектування будь-якої складної системи, відповідних методів, засобів. Саме застосування терміну UI являє собою спробу розробників програмного забезпечення відокремити, принаймні концептуально, функціональне призначення програмних продуктів від проблем, пов'язаних з організацією взаємодії користувача з цими продуктами. Такий поділ є необхідною умовою створення «дружніх» інтерфейсів по відношенню до користувачів програмних продуктів. Поняття дружності - це вектор, який містить відповідно до міжнародної класифікації сім вимог до UI:

- відповідність завдань, що вирішуються користувачем;
- легкість використання;
- керованість;
- відповідність очікуванням користувача;

- недопущення помилок;
- адаптованість / індивідуалізоване;
- легкість вивчення.

Для задоволення потреб різних користувачів було розроблено декілька видів користувацьких інтерфейсів.[2]

Існують такі типи користувацьких інтерфейсів:

- Візуальний ;
- Текстовий ;
- Графічний ;
- Тактильний;
- Жестовий ;
- Голосовий;
- Матеріальний;

1.2 Історія розвитку технології

Графічний інтерфейс користувача розробили у 60-ті роки в науково дослідному інституті Стенфорда Дугласом Енгельбартом.

Згодом концепцію GUI перейняли дослідники з лабораторії Xerox PARC у 1970-х. 1973 року в лабораторії Xerox PARC зібрали команду вчених для дослідження та розробки нового користувацького інтерфейсу. У підсумку, з'явилася концепція графічного інтерфейсу WIMP (Windows, Icons, Menus, Point-n-Click) і в рамках цієї концепції було створено комп'ютер Alto. [21]

1979 року Three Rivers Computer Corporation побудувала робочу станцію PERQ, схожу за принципами роботи на Alto. 1981 року Xerox розробила наступника Alto — Star.

Вперше комерційне втілення концепція GUI отримала 1984 року в продуктах корпорації Apple Computer, що надав GUI. Новий етап в розвитку. [21]

Рисунок 1.1 - Користувацький інтерфейс компанії Apple Computer

Зараз GUI є стандартом для більшості доступних на ринку операційних систем і додатків. Приклади систем, що використовують GUI: Mac OS, GEM, Atari TOS, Microsoft, Windows, Solaris, GNU/Linux. [21]

У 1990 - х роках, Стів Манн розробив ряд стратегій призначених для користувача інтерфейс з впровадженням природної взаємодії з реальним світом як альтернатива інтерфейсу командного рядка (CLI) або графічного інтерфейсу користувача (GUI). Манна EyeTap технологія зазвичай втілює приклад природного призначеного для користувача інтерфейсу.

У 2010 році Даніель Wigdor і Денніс Wixon представив операціоналізацію побудови природних користувацьких інтерфейсів у своїй книзі. В ній, вони ретельно розділяли природні інтерфейси, технологію, використовувану для їх досягнень, і реальності на основі призначеного для користувача інтерфейсу. [3]

1.3 Правило проектування користувацького інтерфейсу

Сьогодні додатки та програми повинні підлаштовуватись під користувача, тому добре продуманий інтерфейс забезпечує плідну взаємодію користувача та програми.

Починаючи проектування потрібно виділити найважливіший принцип який буде забезпечувати вимоги користувача. Дотримання всіх принципів може призвести до того, що це не виправдає очікування користувача та негативно вплине на кінцевий результат.

Принципи розробки користувацького інтерфейсу:

Основне правило проектування – дати користувачу контроль над системою. Розробник інтерфейсу повинен виділити в якості основного такий принцип проектування, при якому продукт буде задовольняти всім вимогам.

Проектувальник дозволяє користувачам вирішувати деякі задачі за власним розсудом.[3]

1.4 Сфери застосування користувацьких інтерфейсів та перспективи технології

Термін «користувацький інтерфейс» часто використовується у сенсі (персональних) комп'ютерних систем та електронних пристроїв. [21]

Види користувацьких інтерфейсів:

1. Інтерфейс «людина-машина» (HMI) зазвичай, є окремим для однієї машини чи частини обладнання та є способом взаємодії між людиною та обладнанням / машиною.
2. Інтерфейс оператора — це спосіб взаємодії, за допомогою якого, можна отримати доступ або керувати багатьма механізмами, які пов'язані із системою контролю Host. [21]

3. Веб-інтерфейси користувача або веб-інтерфейси користувача (WUI), які приймають вхідні дані та забезпечують виведення, створенням веб-сторінок, які передаються Інтернетом і проглядаються користувачем за допомогою програми веб-браузера. У нових втіленнях, використовуються: PHP, Java, JavaScript, AJAX, Apache Flex, NET Framework, або подібні технології для забезпечення керування у дійсному часі в окремій програмі.[21]
4. Голосовий інтерфейс - програма, що дозволяє користувачам керувати голосовим інтерфейсом користувача, які приймають введення і забезпечують виведення, створенням голосових підказок. Введення користувачем, здійснюється натисканням клавіш або кнопок чи виконується словесно. [3]

Система може мати декілька інтерфейсів користувача для обслуговування різних користувачів. Наприклад, комп'ютеризована відео база даних, може містити два інтерфейси користувача, один для керівників відео хостингів (обмежений набір функцій, прилаштований для простоти використання), а інший для працівників (широкий набір функцій, пристосованих для продуктивності)

Перспективи технології

З кожним днем браузери стають швидшими, потужнішими та привабливішими.

Тести на потужність демонструють значне підвищення продуктивності більшості найпопулярніших браузерів.

Збільшується швидкість завантаження сторінок. Mozilla повідомляє, що новий компілятор буде в 10-15 разів швидше, ніж попередній.

Всі сучасні браузери підтримують WebGL 2, що дозволяє отримати абсолютно новий рівень 3D-текстури, рендеринга об’єктів та глибини фрагментів.

Рисунок 1.2 – Анімація в браузері

Нові можливості браузера відкрито підтримують складні анімації. Анімація у 2020 році досліджуватиметься ще глибше. Розвиток технологій дає велику кількість інформації користувачу. І цей розвиток потрібно підтримувати та розвивати, для утримання користувачів. Тому інтерфейси повинні розроблятись в міру виходу нових технологій, щоб бути конкурентоспроможними. [2]

1.5 Висновки до розділу 1:

У даному розділі було розкрито такі аспекти користувацьких інтерфейсів:

- Визначення кристувацького інтерфейсу ;
- Історія розвитку кристувацького інтерфейсу;
- Види користувацьких інтерфейсів;
- Елементи користувацького інтерфейсу;
- Правило пректування користувацького інтерфейсу ;

2 ТЕХНІЧНІ АСПЕКТИ СТВОРЕННЯ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

2.1 Основні етапи розробки інтерфейсів

В минулому розробка ПЗ та UI розвивались лише за рахунок еволюції технологій та систем, на базі яких програми будувались. Це називалось системно- керованою або технологічно- керованою розробкою. Інтерфейси користувачів абсолютно не враховувались. Їм пропонувались програмні функції з інтерфейсом, який розробники були в стані запропонувати. З початку 80- х років акцент було перенесено на розробку, орієнтовану на користувача, причому до розробки залучались і самі користувачі. Однак їм відводилась пасивна роль: у них з’ясовували, які вимоги вони висувають до комп'ютера і які задачі вони вирішуватимуть за його допомогою. Зараз більшість розробників дотримуються нових методологій, названих розробкою із залученням користувачів та розробкою, орієнтованою на користувачів, що навчаються. Новий підхід полягає в тому, що на користувачів потрібно дивитись як на активних учасників самого процесу розробки. Подібне залучення користувачів до розробки сприяє демократизації, а також служить гарантією, що одержана комп'ютерна система буде відповідати запитам та вимогам користувачів.[21]

Розробка, орієнтована на користувачів, що навчаються, спрямована на те, щоб в процесі вирішення своїх задач людина навчалась новим навичкам роботи з комп'ютером, тобто на її інтелектуальний розвиток, тренування її уявлення і одержання знань в різних областях. [21]

Розробка, орієнтована на користувача, заснована на наступних керуючих принципах:

- розуміння потреб користувачів є рушійною силою усього проекту;

- все, що користувачі бачать і до чого доторкаються, повинно проектуватись сумісними зусиллями;
- інноваційний проект завжди є результатом інтенсивної роботи команди спеціалістів в різних областях;
- конкурентоздатний проект вимагає постійного акценту на змагання;
- проект, затверджений користувачем, керує розробкою коду;
- рішення, що приймаються, повинні базуватись на зворотному зв'язку з користувачами;
- інформація від зворотного зв'язку з користувачами повинна збиратись часто, з науковою точністю та швидкістю;
- зворотній зв'язок здійснюється як з потенційними, так і з вже існуючими клієнтами;
- послідовно повинна стандартизуватись і впроваджуватись розробка, орієнтована на користувача;
- розробка, орієнтована на користувача, повинна постійно вдосконалюватись. [21]

Проектування UI може здійснюватись як окремо, так і сумісно з останнім процесом розробки продукту. Зараз більшу увагу приділяють елементам інтерфейсу і об'єктам, які сприймаються і використовуються користувачами, а не функційності програм. В багатьох проектах власне розробка UI та програмування продукту ведуться паралельно, особливо на ранніх стадіях. На більш пізніх етапах враховуються вимоги UI та зворотнього зв'язку, які виявляються в результаті тестування на зручність застосування.

Процес складається з 4 основних етапів:

- збирання і аналіз інформації від користувачів;

- розробка UI;

- побудова UI;

- підтвердження якості створеного UI.

Перший етап: збір та аналіз інформації від користувачів Починати потрібно саме з цього — з користувачів. Перш ніж приступити до розробки та побудови будь-якої системи, потрібно виявнити, які проблеми споживачі або користувачі хочуть вирішити, і як вони звикли працювати. Потрібно спостерігати за користувачами, розпитувати їх. Слід звернути увагу на те, які обмеження накладаються їх комп'ютерними системами на технічне та програмне забезпечення. Потрібно весь час пам'ятати, що запропоноване рішення повинно відповідати не лише теперішнім, але й майбутнім потребам користувачів. Існує ряд ключових питань, які слід поставити на етапі аналізу інформації від користувачів. Проектування та постановка питань, а також проведення аналізу є справжнім мистецтвом. Потрібно бути дуже уважними при опитуванні користувачів та аналізі їх відповідей. Перший етап — дії по збиранню та аналізу інформації — може бути розділений на 5 кроків: - визначення профіля користувача — профіль користувача відповідає на питання “Що представляє собою ваш користувач?”, він дозволяє скласти уявлення про вік, освіту, переваги користувача, одержати іншу необхідну інформацію; - аналіз задач, які стоять перед користувачем — це визначення того, чого хочуть користувачі і яким чином вони збираються вирішувати свої задачі; - збирання вимог, наданих користувачами — відповідає на питання “Яку, з точки зору користувача, користь принесе їм пропонований продукт чи інтерфейс?”; практично в усіх проектах ПЗ враховуються вимоги користувачів, що допомагає визначити особливості проекту та структуру UI.

Щодо збирання вимог, то існують деякі спільні для всіх користувачів вимоги: скорочення роботи з папером, зменшення помилок користувача, автоматизація існуючих ручних процесів, підвищення швидкості здійснення

транзакцій; - аналіз робочого середовища користувачів — відповідає на питання “Де користувачі вирішують задачі, які стоять перед ними?”, тобто потрібно визначити характеристики середовища, які можуть впливати на виконання користувачами своєї роботи. Потрібно зібрати інформацію щодо фізичного боку робочого середовища (освітлення, шум, робочий простір, температура, кількість персоналу), місця роботи користувача та ступінь його мобільності (офіс, квартира, стаціонарно, з пересуваннями), питань ергономіки та умов праці (чи задіює зір, слух, робота ведеться стоячи/сидячи), особливих запитів (рівень підготовки, фізичний стан, інтерес до пізнавального процесу, особливості мови та можливі недоліки), інтернаціоналізації та інших культурологічних умов (переклад, кольори, іконки, текст, повідомлення); - відповідність вимог користувачів задачам, які стоять перед ними — перевірка на реалістичність вимог; якщо вимоги користувача на співрозмірні виконуваним задачам, то потрібно запропонувати їм оптимальний варіант; слід перевірити, чи не перевищують можливості продукту дійсні потреби клієнта. Ітераційний метод передбачає повернення до етапу аналізу вимог користувача, щоб перевірити, чи не змінились в процесі проектування і розробки профіль користувачів, задачі, характеристики середовища або самі вимоги. Для побудови якісного продукту потрібно періодично повертатись до першого етапу, щоб поновлювати відомості про користувачів[21]

Другий етап: розроблення та реалізація UI Розроблення UI для програмного продукту звичайно вимагає значних витарт часу і ресурсів. Етап розробки складається з певних кроків, виконуваних в заданій послідовності. Існує велика спокуса почати програмування фінальної версії продукту вже зараз, не займаючись розробкою інтерфейсу. Однак потрібно пройти всі етапи процесу розробки, перш ніж перейти до програмування. Розроблення включає в себе наступні кроки: - визначення цілі з точки зору зручності застосування продукту — на ранніх стадіях розробки продукту потрібно

точно визначити, що він зможе зробити для користувачів; цілі розробки найкраще сформулювати в термінах, які характеризують дії користувача.

Чотири області, найбільш підходящі для встановлення цілей і задач: придатність, ефективність, легкість в засвоєнні, оцінка користувачами якості продукту; - розробка задач і сценарію дії користувачів — сценарій є описом дій, виконуваних користувачем, це послідовність задач, які стоять перед користувачами, або подій, спрямованих на досягнення єдиної цілі; слід розробити декілька сценаріїв користувача, причому чим більше їх буде, тим менше ймовірність, що буде пропущено ключові об'єкти чи операції, необхідні в інтерфейсі; - визначення цілей та операцій інтерфейсу — найскладнійший і найважливіший крок; на даному кроці потрібно: виділити об'єкти, дані і дії з сценаріїв та задач, які стоять перед користувачами; переглянути і уточнити список об'єктів і дій сумісно з користувачами; накреслити діаграму взаємодії між об'єктами; заповнити матрицю прямого маніпулювання об'єктами; - визначення іконок об'єктів та візуального представлення — визначення, як найкраще представити визначені об'єкти на екрані, в якому вигляді їх побачать користувачі, яку інформацію вони міститимуть. [21]

При визначеннях представлень об'єктів слід враховувати спосіб спілкування користувачів з кожним з об'єктів і з інформацією, яка міститься в них; - розробка меню об'єктів і вікон — вияснення, як користувачі будуть спілкуватись з визначеними і розробленими об'єктами і вікнами. При першому зверненні до ітераційного процесу розробки потрібно швидше створювати прототип, ніж будувати UI. [21]

Прототипування є виключно цінним способом створення перших проектів і демонстрації продукту, особливо на ранніх етапах тестування на зручність застосування. Мета прототипування — швидко і легко візуалізувати різні альтернативні варіанти розробки, а не створювати код, який повинен стати частиною продукту. Необхідно слідувати 3 “золотим”

правилам при використанні прототипів як частини процесу розробки інтерфейсів: - прототипуйте на раніх стадіях і не треба забувати про ітераційний принцип розробки; - створення різних альтернативних варіантів;

Третій етап: підтвердження якості UI тестування на зручність застосування є ключовим елементом ітеративного процесу розробки. Воно полягає в тому, щоб видати продукт на руки великій кількості користувачів і подивитись, чи зможуть вони з ним працювати. Мета тестування на зручність застосування повинна полягати в оцінці поведінки, дій і ступеня задоволеності користувачів. Більшість розробників звертаються до такого виду тестування ближче до кінця проектування. Однак це надто пізно, щоб на основі його результатів вносити зміни. Навіть якщо вони й вносяться, не можна бути впевненими в тому, що виправлений продукт можна використовувати без проведення повторного тестування. Розробники повинні обов'язково бути присутніми при проведенні тестування. Тоді вони зможуть побачити, як користувачі працюють з їх продуктами. Однак вони не повинні мати змогу здійснювати технічну підтримку користувачів при тестуванні. [21]

Щоб досягти успіху в створенні продукту, план повинен задовольняти наступні вимоги.

Рисунок 2.1 – план розробки користувацького інтерфейсу.

2.2 Програмне забезпечення для створення користувацького інтерфейсу

Будь-який користувацький інтерфейс потребує програмне забезпечення для її розробки та взаємодії з іншими технологіями. Потрібно розглянути найпоширеніші фреймворки, які є головними програмними компонентами при побудові користувацьких інтерфейсів.[2]

2.2.1 React

React - це бібліотека JavaScript для створення користувацьких інтерфейсів.

В основі всіх додатків React лежать компоненти. Компонент - це автономна фунція, який виводить деякі дані. Можно написати елементи інтерфейсу, такі як кнопка або поле введення, в якості компонента React. Компоненти є

складовими. Компонент може включати в себе один або кілька інших компонентів.

Для створення програмного забезпечення React, пишемо компоненти React, які відповідають різним елементам інтерфейсу. Потім організуємо ці компоненти всередині компонентів більш високого рівня, які визначають структуру нашого додатку.

Наприклад, форму. Форма може складатися з безлічі елементів інтерфейсу, таких як поля введення, мітки або кнопки. Кожен елемент всередині форми може бути записаний як компонент React. Потім можна написати компонент більш високого рівня та сам компонент форми. Компонент форми визначатиме структуру форми і включати в себе кожен з цих елементів інтерфейсу.

- Важливо відзначити, що кожен компонент в додатку React дотримується суворих принципів управління даними. Складні, інтерактивні інтерфейси часто пов'язані зі складними даними і станом додатки. React обмежий і націлений на те, щоб дати інструменти, що дозволяють передбачити, як додаток буде виглядати в даному наборі обставин. [5]

Основні переваги використання React:

- Можливість розуміти, як компонент буде відмалюватись, дивлячись на вихідний код. Це може бути важливою перевагою, хоча це нічим не відрізняється від шаблонів Angular. [5]
- Зв'язування JavaScript і HTML в JSX робить компоненти простими для розуміння. Зазвичай поділяється уявлення (HTML) і функціональність (JavsScript). Це призводить до монолітного JavaScript файлу, який містить всю функціональність для однієї сторінки, і треба стежити за складним потоком JS-> HTML-> JS. [5]

Порівняння коду написання компоненти на HTML та React

```
<div>
    <div>
        <span>Line</span>
        <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.
            Velit quo doloribus culpa, voluptates officia accusamus.
            Blanditiis?
        </p>
    </div>
</div>
```

Рисунок 2.2 Розмітка , написана на HTML

```
<div>
<span>Line</span>
<p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.
    Velit quo doloribus culpa, voluptates officia accusamus.
    Blanditiis?
</p>
/div>
```

Рисунок 2.3 Розмітка , написана на React

Як видно з даного прикладу, різниці для програміста, з точки зору написання скелету компоненти немає. Але, завдяки тому, що ми працюємо з JSX, це суттєво зменшує час завантаження сторінок сайту, та підвищує його роботу.

Зв'язування функціональності безпосередньо з розміткою і упаковка цього в портативний, автономний «компонент», робить код краще в цілому.

Можливість відрисовувати React на сервері. Якщо розробляти публічний сайт або додаток, то краще відрисовку компонента та інтерфейсу в цілому покасти на сервер. Клієнтський рендеринг - це причина, чому SoundCloud працює повільно, і чому Stack Overflow (використовуючи тільки серверний рендеринг) працює так швидко. Можно відрисовувати React на сервері, і слід вибрати саме цей шлях.

Завдяки перевикористанню коду стало набагато простіше створювати мобільні додатки. Код, який був написаний під час створення сайту, може бути знову використаний для створення мобільного застосування. Якщо

планувати використовувати не тільки сайт, але і мобільний додаток, немає необхідності наймати дві великі команди розробників.

Як підсумок – компонентно-орієнтований підхід, можливість з легкістю змінювати наявні компоненти і перевикористати код перетворюють React розробку в безперервний процес поліпшення. Компоненти, які були створені під час роботи над тим чи іншим проектом, не мають додаткових залежностей. Таким чином, ніщо не заважає використовувати їх знову і знову в проектах різного типу. Весь попередній досвід може бути з легкістю застосований при роботі над новим сайтом або навіть при створенні мобільного додатка. Використовуючи передові можливості, такі як Virtual DOM або ізоморфний JavaScript, React розробники можуть з високою швидкістю створювати високопродуктивні додатки, незважаючи на рівень їх складності. Можливість з легкістю заново використовувати вже наявний код підвищує швидкість розробки, спрощує процес тестування, і, як результат, знижує витрати. Той факт, що ця бібліотека розробляється і підтримується висококваліфікованими розробниками і набирає все більшої популярності з кожним роком, дає підстави сподіватися, що тенденція до подальших поліпшень продовжиться. [5

2.3 Angular

Angular 8 - це остання версія програми Angular, яка підтримує Google, яка базується на клієнтах та на основі браузера. Замість того, щоб говорити з сервером для перезавантаження HTML-сторінок, Angular дозволяє веб-сайтам швидко та ефективно відрисовуватись. Це фреймворк "все в одному" із вбудованими інструментами для маршрутизації, рішень для управління компонент та перевірки форми. Angular побудований на TypeScript, який є сумісним з JavaScript.

Зокрема, було введено цікаве поняття прив'язки даних, яке означало автоматичне оновлення подання кожного разу, коли модель (дані)

змінювалися, і навпаки. На додаток до цього була представлена ідея директив, яка дозволила винайти власні HTML-теги, реалізовані через JavaScript.

Спеціальні теги, які будуть оброблені AngularJS і перетворяться на повноцінний елемент інтерфейсу, як вказується базовим кодом. (Звичайною роботою було б кодування відповідної директиви).

Ще однією досить важливою річчю була Dependency Injection, яка дозволила компонентам програми з'єднуватися разом таким чином, щоб полегшити багаторазовість і перевірку коду.

AngularJS став популярним дуже швидко і отримав багато шаблонів. Тим не менш, його технічні працівники вирішили зробити ще один крок і приступили до розробки нової версії, яка спочатку називалася Angular 2 (пізніше просто Angular без частини "JS"). Не випадково Angular отримав нову назву: насправді він був повністю перероблений, в той час як багато понять були переглянуті.

Перший стабільний випуск Angular 2 був опублікований у 2016 році, і з тих пір AngularJS почав втрачати свою популярність на користь нової версії. Однією з головних особливостей Angular 2 була можливість розвиватися для декількох платформ: веб, мобільних та рідного робочого столу (тоді як AngularJS не має мобільної підтримки поза коробкою).

Потім, щоб зробити речі ще складнішими, до кінця 2016 року вийшов Angular 4. В офіційному дописі блогу, технічні працівники вирішили дотримуватися семантичної версії з моменту Angular 2, тому не стали називати Angular 4 як Angular 3.

Дотримуючись цього принципу, зміна основної версії (наприклад, "2.x.x" стає "3.x.x") означає, що деякі зміни були внесені. Проблема полягає в тому, що компонент Angular Router вже був у версії 3. Тому для виправлення цієї нерівності було вирішено взагалі пропустити Angular 3. На щастя, перехід від Angular 2 до 4 виявився менш болісним, ніж від AngularJS до Angular 2, хоча

багато розробників все ще були досить розгублені щодо всього цього безладу.

Angular 5 був випущений у листопаді 2017 року. Він також сумісний з попередніми кутовими версіями. Кутовий 6 повинен вийти досить скоро, сподіваємось, що це ще більше цікавих функцій та вдосконалень.

Переваги:

- Angular надає не тільки інструменти, але й шаблони дизайну, щоб побудувати проект шляхом вмонтування готових компонент. Якщо програма Angular створена належним чином, в програмному коді не буде заплутаних класів і методів, які важко модифікувати та ще важче перевірити. Код структурований зручно, і не потрібно буде витрачати багато часу, щоб зрозуміти, що відбувається.
- Це JavaScript, але краще. Angular будується на TypeScript, який, в свою чергу, спирається на JS ES6. Не потрібно вивчати абсолютно нову мову, але все одно потрібно вивчати синтаксис та функції, такі, як статичне введення тексту, інтерфейси, класи, простори імен, декоратори тощо.
- У Angular уже є багато інструментів, щоб почати розробляти додаток відразу. Існують директиви, щоб надати HTML - елементам динамічну поведінку. Можливість контролювати форми за допомогою FormControl та вводити різні правила перевірки. Можливість легко надсилати асинхронні запити HTTP різних типів. Можливість налаштувати маршрутизацію.

Особливості Angular:

- Компоненти відокремлюються. Angular прагнув усунути щільне з'єднання між різними компонентами додатку. Ін'єкція відбувається у стилі NodeJS, та можна легко замінити різні компоненти.

- Усі маніпуляції з DOM відбуваються там, де це має відбутися. Завдяки Angular не потрібно з'єднувати між собою презентацію компоненти та логіку програми, що робить розмітку набагато чистішою та простішою.
- Тестування лежить в основі. Angular призначений для ретельного випробування, і він підтримує як одиничне, так і ціле тестування з такими інструментами, як Жасмін і Трантрактор.
- Angular готовий для мобільних платформ та настільних ПК, тобто є одних код для декількох платформ.
- Angular активно підтримується і має велику спільноту та екосистему. Ви можете знайти безліч матеріалів на цій основі, а також багато корисних сторонніх інструментів. [6]

2.2.3 VUE JS

Vue вважає себе "прогресивним фреймворком JavaScript", а також використовує термін "поступово прийнятна екосистема". Як випливає з назви, основна бібліотека Vue лише надає шар перегляду стандартного шаблону архітектури. Однак Vue також може бути розширений сотнями різних компонентів і бібліотек, які дозволяють розробникам створювати надійні односторінкові програми.

На відміну від своїх найбільш прямих конкурентів - Angular та React - Vue бракує підтримки такої великої компанії, як Google або Facebook. Тим не менш, ним користується все більша кількість відомих технологічних фірм. Сюди входять Nintendo, Expedia і китайські компанії Alibaba і Baidu.

Нижче наведено переваги використання технології Vue.js у веб-розробці. [9] Дуже невеликий розмір. Успіх JavaScript- фреймворку залежить від її розміру. Чим менший розмір, тим більше він буде використовуватися. Однією з найбільших переваг Vue.js є його невеликий розмір. Розмір цього фреймворку становить 18–21 КБ, і користувач не потребує часу, щоб

завантажувати та використовувати її. Це не означає, що він має низьку швидкість через невеликі розміри. Тому він обігрує всі об’ємні фреймворки, такі як React.js, Angular.js та Ember.js. [9]

Легко зрозуміти і розробляти програми. Однією з причин популярності цього фреймворку є те, що його зрозуміти досить просто. Користувач може легко додати Vue.js до свого веб-проекту через його просту структуру. Як малі, так і великі шаблони масштабів можуть бути розроблені за допомогою фреймворку, що економить багато часу. У разі будь-якої проблеми, користувач може легко простежити за блоками помилки. Все це через свою просту структуру. [9]

Vue.js також популярний серед веб-розробників, оскільки полегшує їх інтеграцію з існуючими програмами. Це тому, що він заснований на фреймворках JavaScript і може бути інтегрований в інші програми, побудовані на JavaScript. Це означає, що це корисно для розробки нових веб-додатків, а також зміни попередніх програм. Ця інтеграція можлива, оскільки Vue.js має компоненти для всього. [9]

Розробники завжди люблять використовувати фреймворк з детальною документацією, оскільки їм завжди легко написати перший код. Документація на Vue.js настільки не вичерпна, що будь-який користувач, який трохи знає JavaScript та HTML, може розробити власне додаток чи веб-сторінку.

Велика гнучкість - ще одна перевага Vue.js. Це дозволяє користувачеві писати свій шаблон у HTML-файл, файл JavaScript та чистий файл JavaScript за допомогою віртуальних вузлів. Ця гнучкість також дозволяє зрозуміти розробникам React.js, Angular.js та будь-який інший новий фреймворк JavaScript. Vue.js виявився дуже корисним у розробці тих простих додатків, які працюють безпосередньо від браузерів. [9]

Vue.js також сприяє двосторонній комунікації завдяки своїй архітектурі MVVM (Model-View-ViewModel), що дозволяє досить легко обробляти

HTML-блоки. У цьому відношенні це здається дуже близьким до Angular.js, що також прискорює блоки HTML.

Можна сказати, що Vue.js має явні переваги перед усіма попередніми структурами, такими як Angular.js та React.js. Коротше кажучи, він поєднує в собі особливості всіх старих фреймворків. Завдяки великому вибору аудіо та відео шаблонів, можна швидко інтегрувати любий інтерфейс перегляду в додаток. [9]

2.2.4 React Native

React Native допомагає створювати реальні та захоплюючі мобільні додатки лише за допомогою JavaScript, що підходить як для платформ Android, так і iOS. Програма React Native доступні як для iOS, так і для Android платформ, що допомагає заощадити час на розробку. React Native набув великої популярності, а також підтримується Facebook. Він сьогодні має величезну підтримку громади.

React Native побудований на концепції ReactJS, що дало величезну конкуренцію давно улюбленому AngularJS.

Хоча між ReactJS та React Native є деякі подібності та різниці, про які йдеться нижче:

React Native - це структура, яка створює ієрархію компонентів інтерфейсу для створення коду JavaScript. У ньому є набір компонентів як для iOS, так і для Android платформ для створення мобільного додатку з власним виглядом та роботою. ReactJS - це бібліотека JavaScript з відкритим кодом для створення інтерфейсів користувача. Однак і React Native, і ReactJS розробляються Facebook, використовуючи однакові принципи дизайну, за винятком проектування інтерфейсів.

Оскільки він використовує той самий код для створення програм React Native iOS або додатків та веб-додатків React Native Android, нам тільки

потрібно знати HTML, CSS та JavaScript, замість Swift для розробки на iOS та Java для Android.

Завдяки React Native є можливість надавати інтерфейс користувача як для iOS, так і для Android платформ.

Це фреймворк з відкритим кодом, який може бути сумісний з іншими платформами, такими як Windows або tvOS найближчим часом.

Оскільки компоненти React Native мають відповідні права, можемо використовувати ці компоненти для створення додатків для Android та iOS.

Також є можливість включити React Native компоненти до коду існуючої програми, або використати код на основі Кордови за допомогою плагіна.

React Native Development є порівняно простим, швидким та ефективним. React Native - це вибір для тих розробників, які мають досвід роботи в JavaScript, оскільки немає необхідності вивчати специфічні для Java для Android або Swift для Ios як писалось раніше.

React Native - це інтерфейс, орієнтований на інтерфейс користувача, завдяки чому додатки швидко завантажуються та надають більш плавне відчуття.

Правильно сказати, що мобільні телефони - це напівдуша кожної людини, і коли мова йде про розробку Android або розробку iOS, бізнес часто заплутується, чи варто пропонувати своїм споживачам мобільний додаток із чудовим інтерфейсом користувача та видатним інтерфейсом. Тому їх вибір часто припадає саме на React Native.[8]

2.2.5 Electron

Electron JS - це фреймфорк, яка дозволяє користувачеві створювати набори настільних програм із HTML5, CSS та JavaScript. Це проект з відкритим кодом, розпочатий Чен Чжао, інженером GitHub.

Це в основному поєднання двох неймовірно популярних технологій: Node.js і Chromium. Таким чином, будь-яка написана веб-програма може

працювати на Electron JS. Аналогічно, будь-яка програма Node.js, яку пишемо, може використовувати цю технологію.

Початок Electron JS розпочалася в січні 2013 року з пошуку інструменту для створення міжплатформенного текстового редактора, на якому користувач може працювати з такими технологіями, як JavaScript, HTML та CSS. Він був заснований 15 липня 2013 року, призначений полегшити розвиток платформи для створення "Atom". Atom - безкоштовний текстовий редактор з відкритим вихідним кодом для Linux, macOS, Windows з підтримкою плагінів , написаних на Node.js , і вбудованих під керуванням Git . Більшість плагінів мають статус вільного програмного забезпечення, розробляються і підтримуються спільнотою. Версія 1.0 була випущена 25 червня 2015 р.[7]

2.2.6 Ember.js

Ember.js - це JavaScript-фреймворк для розробки клієнтської частини веб-додатків, амбітний проект, який останнім часом привертає до себе багато уваги. [7]

Фреймворк Ember.js включає в себе безліч сучасних концепцій і технологій зі світу JavaScript. Серед його можливостей хотілося б відзначити наступні:

- Використання транспілятора Babel для забезпечення підтримки ES2016.
- Підтримка засобів тестування Testem і QTest , що відкриває можливості за модульним, інтеграційному і приймальним тестування.
- Підтримка збирача Broccoli.js .
- Інтерактивне перезавантаження веб-сторінок, що прискорює процес розробки.
- Підтримка шаблонізатора Handlebar .

- Використання моделі розробки, при реалізації якої в першу чергу створюються URL-маршрути. Це забезпечує повну підтримку глибоких посилань.
- Наявність шару для роботи з даними, заснованого на JSON API, але підтримує підключення до будь-якого API, з яким потрібно налагодити роботу. [7]

Крім того, варто сказати, що Ember.js - це фреймворк, орієнтований виключно на фронтенд. Він підтримує безліч способів взаємодії з різною бекенд, серверною частиною додатку, але все, що відноситься до серверного коду, не входить в сферу відповідальності Ember. [7]

Інтерфейс командного рядка Ember.js, `ember-cli`, відкриває доступ до безлічі можливостей цього фреймворка. `Ember-cli` підтримує програміста на всіх етапах роботи. Він спрощує створення додатка, розширення його функціональності, тестування і запуск проекту в режимі розробки. [7]

2.2.7 JQuary

jQuery - це швидка, невелика і багатофункціональна бібліотека JavaScript, що входить у один файл .js.

jQuery полегшує життя веб-розробника. Він надає безліч вбудованих функцій, за допомогою яких можна легко та швидко виконувати різні завдання.

Важливі функції jQuery:

- Вибір DOM: jQuery надає селектори для отримання елемента DOM на основі різних критеріїв, таких як назва тегу, id, ім'я класу css, ім’я атрибута, значення, n-та дитина в ієрархії тощо.
- Маніпуляція з DOM: можливість керувати елементами DOM за допомогою різних вбудованих функцій jQuery. Наприклад, додавання або видалення елементів, зміна вмісту html, клас css і т.д.

- Спеціальні ефекти: можливість застосовувати спеціальні ефекти до елементів DOM, таких як елементи показу або приховування, зменшення видимості або зменшення видимості, ефект ковзання, анімація тощо.
- Події: Бібліотека jQuery включає функції, еквівалентні подіям DOM, як клік, dblclick, mouseenter, mouseleave, blur, keyup, keydown тощо. Ці функції автоматично вирішують проблеми між подіями.
- Ajax: jQuery також включає прості у використанні функції AJAX для завантаження даних із серверів, не завантажуючи всю сторінку.

Підтримка крос-браузера: бібліотека jQuery автоматично обробляє проблеми між веб-браузером, тому користувачеві не потрібно турбуватися про це. jQuery підтримує IE 6.0+, FF 2.0+, Safari 3.0+, Chrome і Opera 9.0+. [10]

2.3 Порівняння Веб-фреймворків

WEB фреймворк — це каркас, призначений для створення динамічних веб-сайтів, мережевих додатків, сервісів або ресурсів. Він спрощує розробку і позбавляє від необхідності написання рутинного коду. Багато фреймворків спрощують доступ до баз даних, полегшують розробку інтерфейсу, а також зменшують дублювання коду.

Великий вибір засобів для проектування користувацьких інтерфейсів дає можливість вирішити, який фреймворк використовували доцільніше, для задоволення потреб програми або додатку.

Аналізуючи всі характеристики фреймворків з пункту 2.2 складено Табл.2.3.1, що наочно показує рейтинг програм відносно одна одної. При створенні таблиці беремо до уваги найсуттєвіші параметри: підтримка веб сокетів, робота с DOM, обмін данними через XMLHttpRequest, анімація, валідація, підтримка Canvas, доступність, масштабованість, підтримка браузерів.[11]

Таблиця 2.1 - Порівняння функціональності фреймворків

	AngularJS	Ember.js	jQuery	React	Vue	Electron	React Native
Знаходження функції	Так	Так	Так	Так	Так	Так	Так
«Обгортання» DOM	Так	Ні	Так	Так	Так	Так	Так
Обмін даними через XMLHttpRequest	Так	Так	Так	Так	Так	Так	Так
WebSocket	Так	Так	Так	Так	Так	Так	Так
Other data retrieval	Так	Так	Так: XML, HTML	Так	Так	Так	Так
Візуальні ефекти	Так	Так	Так	Так	Так	Так	Так
Анімація / складні віз. ефекти	Так	Так	Так	Так	Так	Так	Так
Чарти і панель керування							

Продовження таблиці 2.1

	AngularJS	Ember.js	jQuery	React	Vue	Electron	React Native

Підтримка кнопки «назад»	з додатковою бібліотекою	Так	з плагінами	з додатковою бібліотекою	Так	Так	Так
Керування історією							
Отримання даних з віджетів і валідація	Так	Так	з плагінами	з додатковою бібліотекою	з додатковою бібліотекою	Так	з додатковою бібліотекою
Сітка (grid)	Так	Так	з плагінами	Так	Так	Ні	Так
Деревовидна структура							
Візуальний редактор	Ні	Так	з плагінами	Так	Так	Так	Так
Автозавершення	Ні		Так				
Генерація HTML	Так	Так	Так	Так	Так	Ні	Так
Підтримка Canvas	Так	Так	з плагіном	Так	Так	Так	Так
Підтримка смартфонів/планшетів	Так	Так	з плагіном	Так	Так	Так	Так

Продовження таблиці 2.1

	AngularJS	Ember.js	jQuery	React	Vue	Electron	React Native
Доступніс	Так	Ні	Так	Так	Так	Так	Так

ть							
Інструменти розробника	Так	Так	Так (з плагіном)	Так	Так	Так	Так
Робота офлайн							
Крос-браузерна 2d векторна графіка	Так	Так	з плагіном	Так	Так	Так	Так

Таблиця 2.2 - Порівняння ваги фреймворків

Фреймворк	**Розмір**	**Ліцензія**
AngularJS	144 kB (мініфіковано + стиснено)	MIT
Ember.js	95 kB (мініфіковано + gzip-стиснення),	MIT
	340 kB (мініфіковано),	
	1.5 MB (без компресії)	
jQuery (library)	32 KiB (мініфіковано + gzip-стиснення),	MIT
	93 KiB (мініфіковано),	

Продовження таблиці 2.2

Фреймворк	**Розмір**	**Ліцензія**
React	133K	MIT

Vue	58.8K	MIT
The Electron	33.2мБ	MIT
React Native	5 мБ	MIT

Таблиця 2.3 - Порівняння підтримки браузерами

Фреймворк	Internet Explorer	Mozilla Firefox	Safari	Opera	Chrome
AngularJS (1.3)	8+ (9+)	4+	5+	11+	30+
Ember.js	6+	3+	4+	10.6+	14+
jQuery	6+	2+	3+	9+	1+
React	9+	4+	5+	11+	30+
Vue	11+	4+	6+	11+	30+
The Electron	-	-	-	-	70
React Native	-	-	-	-	-

2.4 Особливості розробки користувацького інтерфейсу на React

Бібліотека React була вперше випущена компанією Facebook в 2013 році. Для того, щоб зрозуміти, наскільки популярною ця технологія стала за минулий час, звернемося до опитування розробників, проведеним сайтом StackOverflow в цьому році. Більш 50 000 розробників поділилися інформацією про свою роботу і професійних перевагах. Крім більш-менш передбачуваних результатів, які описують стан IT-індустрії на сьогоднішній день, є також і дещо цікаве щодо безпосередньо React. Ця бібліотека стала однією з найулюбленіших і затребуваних технологій, а також самої трендової технологією на StackOverflow.

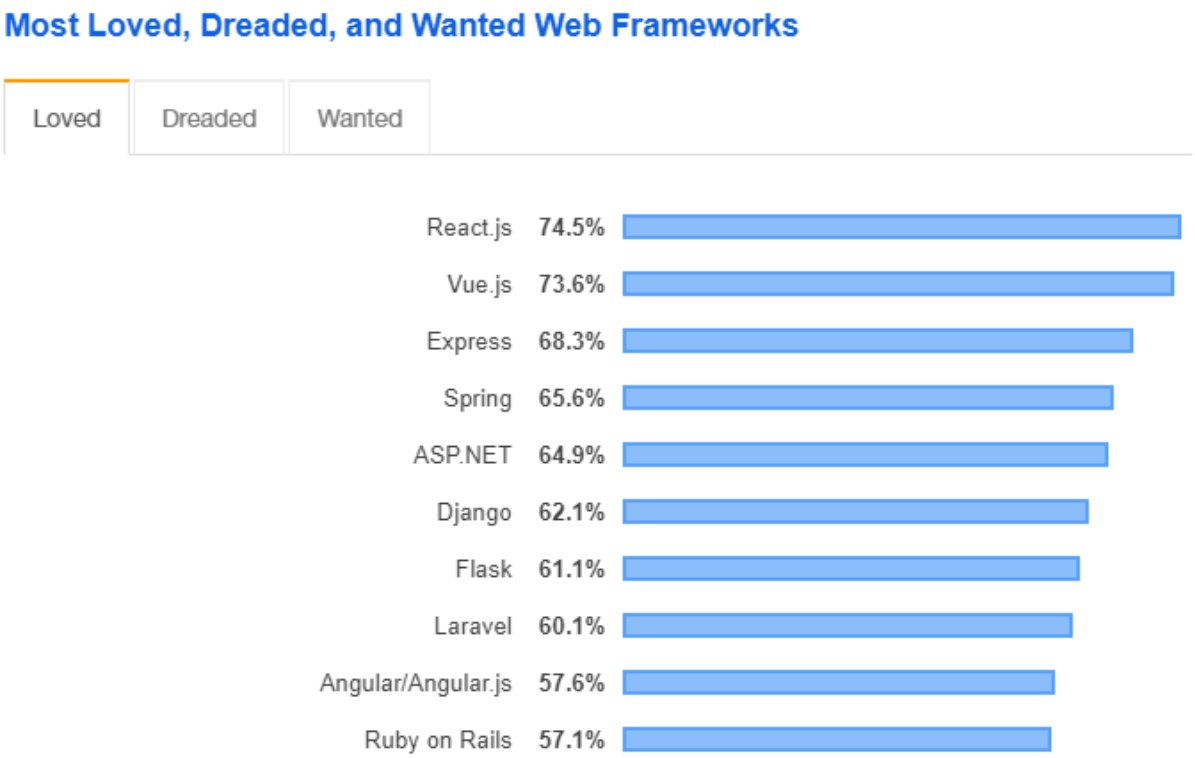


Рисунок 2.4 – Професійні переваги технології з точки зору розробників, по даним з сайту StackOverflow.

React це бібліотека для створення користувацьких інтерфейсів. Однією з її характерних особливостей є можливість використовувати JSX, мова програмування з близьким до HTML синтаксисом, який компілюється в JavaScript. Розробники можуть вимагати більшої продуктивності додатків за допомогою Virtual DOM. З React можемо створювати ізоморфні додатки, які допоможуть позбавити нас від неприємних ситуацій, коли користувач з нетерпінням чекає, коли ж нарешті завершиться завантаження даних і на екрані його комп'ютера нарешті з'явиться щось крім анімації завантаження.

Створені компоненти можуть бути з легкістю змінені і використані заново в нових проектах. Високий відсоток перевикористання коду зменшує час на розробку, що, в свою чергу, призводить до більш високого рівня контролю якості. Використовуючи React Native мобільні додатки для Android і iOS, використовуючи досвід JavaScript і React розробки. [12]

Яку користь React дає користувачам та бізнесу?

- Virtual DOM може підвищити продуктивність високонавантажених додатків, що може знизити ймовірність виникнення можливих незручностей і покращує користувальницький досвід;
- Використання ізоморфного підходу допомагає робити рендеринг сторінок швидше, тим самим дозволяючи користувачам відчувати себе більш комфортно під час роботи з додатком. Пошукові системи індексують такі сторінки краще. Оскільки один і той же код може бути використаний як в клієнтської, так і в серверній частині програми, немає необхідності в дублюванні одного і того ж функціоналу. В результаті час розробки і витрати знижуються;
- Завдяки перевикористанню коду стало набагато простіше створювати мобільні додатки. Код, який був написаний під час створення сайту, може бути знову використаний для створення мобільного додатку. Якщо ви плануєте використовувати не тільки сайт, але і мобільний додаток, немає необхідності наймати дві великі команди розробників. [12]

Ізоморфні додатки.

Коли говориться про ізоморфний додаток або про ізоморфних JavaScript, мається на увазі, що є можливість використовувати один і той же код як в серверній, так і в клієнтській частині програми. Коли користувач відкриває сайт в своєму браузері, вміст сторінки має бути завантажено з сервера. У випадку з SPA-додатками (Single Page Application), це може зайняти деякий час. Під час завантаження користувачі бачать або порожню сторінку, або анімацію завантаження. З огляду на, що за сучасними стандартами очікування протягом більш ніж двох секунд може бути досить помітним незручністю для користувача, скорочення часу завантаження може виявитися

вкрай важливим. А ось ще одна вагома проблема: пошукові машини не індексують такі сторінки так добре, як нам хотілося б.

Виконання JavaScript коду на стороні сервера допомагає виправити подібні проблеми. Якщо створювати ізоморфні додатки, є можливість отримати помітну вигоду, виробляючи рендеринг на стороні сервера. Після завантаження сторінки можемо все ще продовжувати рендеринг компонентів. Така можливість рендеринга сторінок як на сервері, так і на клієнті приводить до помітних переваг, таким як можливість кращого індексування сторінок пошуковими машинами і поліпшення користувацького досвіду.

Більш того, такий підхід дозволяє знизити час, що витрачається на розробку. При використанні деяких сучасних фреймворків, ми повинні створювати компоненти, які повинні рендери на стороні сервера, а також шаблони для клієнтської сторони додатки. React розробники можуть створювати компоненти, які працюють на обох сторонах.

Virtual DOM.

Document Object Model, або DOM, - це спосіб представлення та взаємодії з об'єктами в HTML, XHTML і XML документах. Відповідно до цієї моделі, кожен такий документ являє собою ієрархічне дерево елементів, зване DOM-деревом. Використовуючи спеціальні методи, є можливість отримати доступ до певних елементів документа і змінювати їх так, як ми хочемо. Коли створюємо динамічну інтерактивну веб-сторінку, хочемо, щоб DOM оновлювався так швидко, як це можливо після зміни стану певного елемента.

Для даної задачі деякі фреймворки використовують прийом, який називається «dirty checking» і полягає в регулярному опитуванні стану документа і перевірці змін в структурі даних. Подібне завдання може стати справжнім головним болем в разі високонавантажених додатків. Virtual DOM, в свою чергу, зберігається в пам'яті. Саме тому в момент, коли «справжній» DOM змінюється, React може змінювати Virtual DOM в одну

мить. React «збирає» такі зміни порівнює їх зі станом DOM, а потім перемальовує змінилися компоненти.

При цьому підході не потрібно робити регулярні оновлення DOM. Саме тому може бути досягнута більш висока продуктивність React додатків. Другий наслідок випливає з ізоморфної природи React: можливість виробляти рендеринг на стороні сервера зовсім як на стороні клієнта.

Оскільки для реакту розроблено безліч різних бібліотек, працювати та відтворювати відео та аудіо файли стало набагато проще.

ReactPlayer - бібліотека для відтворювання відео, має в собі безліч налаштуваннь. Не тільки для перегляду але й швидкого редагування відео, прямо під час перегляду. [12]

Встановлення та налаштування, дуже легке, потрібно тільки скачати та підключити бібліотеку до сайту.

```
npm install react-player –save
```

Імпортувати та додати в компоненту для відтворення

```
import React, { Component } from 'react'

import ReactPlayer from 'react-player'

class App extends Component {

  render () {

    return <ReactPlayer url='https://www.youtube.com/watch?v=ysz
5S6PUM-U' playing />

  }

}
```

Фрагмент коду, який відрисовує вікно відео плеера

Рисунок 2.5-Демонстрація ReactPlayer .

Бібліотека Player

Player бібліотека для відтворювання відео, є більш простішою, та має вигляд констуктора, з якого можна зібрати плеер на різні потреби. Вона має бізліч налаштувань, які легко інтегруються та не залежать один від одного.

Імпортуємо плеер:

```
import { Player, ControlBar } from 'video-react';
```

Рисунок 2.6 –Імпорт Player в компоненту.

Налаштовуємо всі вихідні файли та вмонтовуємо плеер в компоненту:

```
const sources = {
  sintelTrailer: 'http://media.w3.org/2010/05/sintel/trailer.mp4',
  bunnyTrailer: 'http://media.w3.org/2010/05/bunny/trailer.mp4',
  bunnyMovie: 'http://media.w3.org/2010/05/bunny/movie.mp4',
  test: 'http://media.w3.org/2010/05/video/movie_300.webm'
};
```

Рисунок 2.7 – Налаштування файлів для відображення.

```
<Player
```

```
          ref={player => {
            this.player = player;
          }}
          autoPlay
        >
          <source src={this.state.source} />
          <ControlBar autoHide={false} />
                                      </Player>
```

Рисунок 2.8 –Додавання Player в компоненту.

Як результат, бачимо в вікні браузеру преер з можливостями перемотки, збільшення, зменшення гучності, паузою, та розкриттям на весь екран.

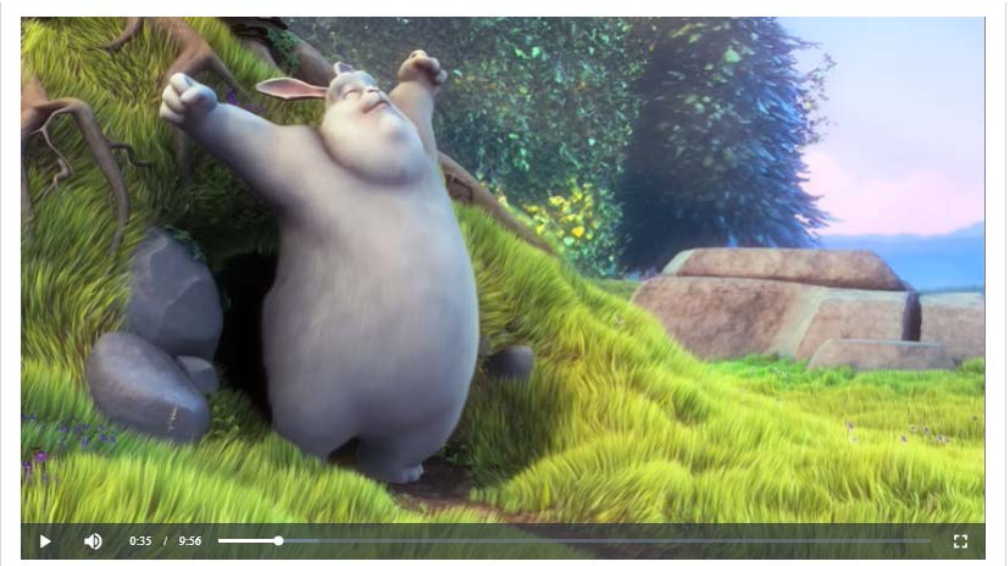


Рисунок 2.9 –Результат роботи компоненти.

2.5 Висновки до розділу 2:

- Визначено поняття користувацького інтерфейсу;
- Розглянуто аспекти розробки інтерфейсів;
- Проведений аналіз технологій для створення користувацького інтерфейсу;
- Проведено порівняння технологій для створення користувацького інтерфейсу;

3 РОЗРОБКА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

Метою розділу є створення програми, що використовує технологію React , яка:

Підтримується мобільними та десктопними браузерами ;

Дозволить вести пошук пісні чи кліпу, за допомогою фрагмента з цієї пісні чи завантаження фрагмента пісні. Дозволить прослуховувати, переглядати інформацію, та переходити на сайт дистриб'ютора.

Для досягнення поставленої мети вирішено використовували технологію React і систему керування даними Redux. В купі - ці технології повністю задовільняють потреби та задачі, поставлені при розробці інтерфейсу.

3.1 Створення схеми побудови інтерфейсу

Перед початком написання програми, потрібно розібратись, що нам потрібно використовувати.

Оскільки нам потрібно спілкуватись з сервером, нам потрібно розробити користувацький зрозумілий інтерфейс. В нашому випадку будемо використовувати React для написання інтерфейсу програми. На сервер, ми повинні віддавати деяку інформацію для пошуку, та отримувати відповідь від нього. Тому, нам потрібно місце для зберігання даних. Тому для цього використовуємо Redux для отримання, зберігання та роботу з даними.

Також, для дотримання правил написання додатку, для його подальшого розширення та зрозумілості для інших розробників використовуємо архітектурний паттерн FLUX та MVC.

Redux дозволяє керувати станом веб-додатків, створених на будь-якому JavaScript фреймворку, наприклад, React, Meteor або Angular.

Redux виник з ідеї Flux - архітектурного патерну.

Особливість Redux в тому, що він дає один об'єкт який зберігає стан всього додатку в одному місці і який може включати дані з серверного API

або зовнішнього API, стану навігації, інформацію про користувача, перемикання стан кнопки і т.д.[13]

Це значною мірою знижує можливість допускання помилом, зменшує масштабованість, та дає можливість швидко діагнозтувати місця помилок

Основні поняття Redux - Store, Actions, Reducers і Subscriptions.

Initial state - це почтаковий стан додатку.

Flux - це архітектура, яку команда Facebook використовує при роботі з React. Це новий архітектурний підхід, який доповнює React і принцип однонаправленого потоку даних.

Компоненти Flux:

- Actions / Дії - хелпери, що спрощують передачу даних Диспетчера
- Dispatcher / Диспетчер - приймає Дії і розсилає навантаження зареєстрованим обробникам
- Stores / Сховища - контейнери для стану програми та бізнес-логіки в обробниках, зареєстрованих в диспетчері
- Controller Views / Уявлення - React-компоненти, які збирають стан сховищ і передають його дочірнім компонентів через властивості.

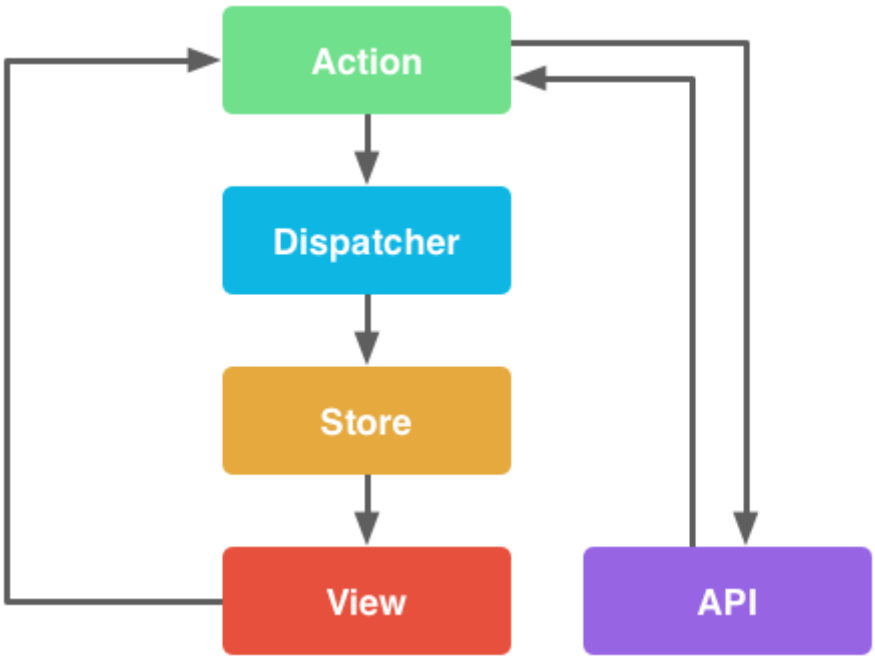


Рисунок 3.1- Архітенкрура Flux

В такий спосіб наша програма та код буде доступний для розуміння іншим розробникам, легко підтримуватись та розширюватись.

Після побудови схеми роботи, та визначення переліку технологій можна починати розробку інтерфейсу. [14]

3 .2 Налаштування бібліотек та середовища

Розробка інтерфейсу буде в редакторі Visual Studio Code- це досить легкий кроссплатформений редактор для розробки веб- додатків та менеджер пакетів npm. Для початку в консолі редактору вводимо: npx і create-react-app. Даною командою ми встановимо набір інструментів, які допоможуть створити React-додаток. По завершенні установки зможемо запустити додаток командами:

```
npx create-react-app my-app
cd my-app
npm start
```

Рисунок 3.2 - Команди для початку розробки на React

Після автоматичного налаштування всіх залежностей, переходимо в папку проекту командою cd my-app та командою npm start запускаємо додаток.

Після цього, налаштовуємо всі додаткові бібліотеки, такі як: react-router-dom, axios, redux, react-redux, redux-thunk. Для цього в консолі редактору прописуємо npm install і назву кожної бібліотеки.

```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS D:\helper> npm install react-router-dom
```

Рисунок 3.3 - Додавання в проект необхідних бібліотек

Нашим способом структурування проектів є угруповання по типу файлів:

Рисунок 3.4 - Папки проекта

Сенс в тому, щоб розміщувати компоненти в різні папки в залежності від їх ролі в додатку.

1 Папка api – це об'єкт, що інкапсулює зв'язок із API сервера в якому відбуваються запити на сервер, та отримання відповіді від нього.

2 Components - всі компоненти, які потрібні для відтворення інтерфейсу

3 Img – папка з картинками.

4 REDUX – Business Logic Layer або бізнес-рівень інкапсулює всю бізнес-логіку, всі необхідні обчислення, отримує об'єкти з рівня доступу до даних і передає їх на рівень представлення, або, навпаки, отримує дані з рівня уявлення і передає їх на рівень даних.

5 Utils – всі додаткові залежності програми.

Після налаштування середовища та всіх залежностей можна починати писати програмний код.[15]

3.3 Розробка та завантаження на хостинг

Для подальшого опису процесу розробки програми введемо поняття чистої компоненти та компоненти обгортки .

Чиста компонента – це функція яка повертає ті дані які надходять в неї, без зміни цих даних. Такі компоненти нам потрібні для відображення дизайну сайту, та даних.

Компонента обгортка – функція, яка може змінювати дані, що надходять в неї, виконувати різні операції з сервером, передавати дані в чисту компоненту. Вона потрібна, щоб фільтрувати дані, змінювати їх, та передавати в чисту компоненту.

Всі компоненти, що використовуються в проекті, мають своє місце в деревовидної ієрархічній структурі. У кожної компоненти може бути тільки один батько і безліч нащадків. На рис. 3.1.7 наведено скріншот вікна редактора з панеллю ієрархією компонентів.

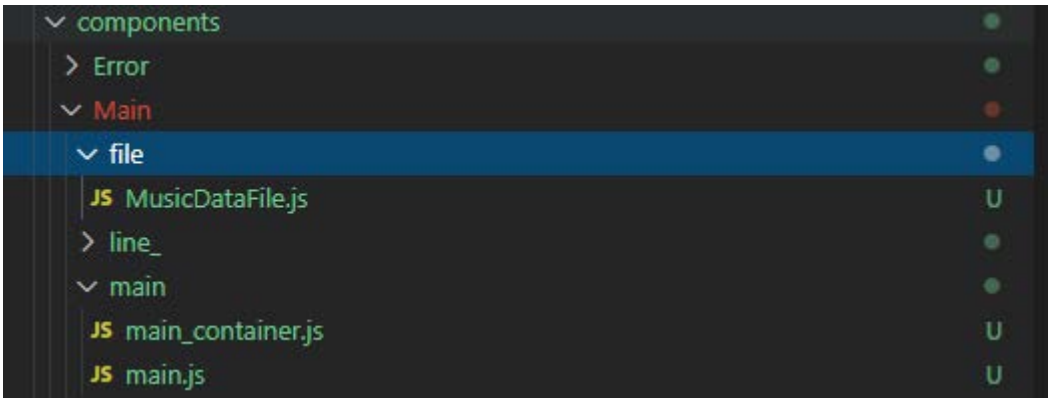


Рисунок 3.5 - Ієрархія компонентів

Папка file та line містить в собі чисті компоненти для відображення користувацького інтерфейсу та інформації.

Папка main - містить компоненту для вибору способу пошуку музики чи кліпу, та функцій для отримання даних від сервера. Після того як кнопка пошуку була натиснута, викликається функція, що передає данні на DALL рівень для відправлення на сервер.

main_container – компонента вищого порядку, яка взаємодіє з через reducer з глобальним об’єктом Store. Вона містить в собі дві функції:

mapStateToProps – функція яка повертає дані з глобального об’єкта Store в компоненту.

mapDispatchToPorops – функція яка, в моєму випадку, передає в компоненту виклик асинхронної функції, для запиту на сервер та запису результату з серверу до об’єкту Store.

music_Reducer - це чиста функція, яка приймає попередній стан і дію (state і action) і повертає наступний стан (нову версію попереднього). Функція називається ред’юсером (reducer) тому, що її можна передати в массив.

Для пошуку інформації потрібно написати форму з полем для вводу та кнопкою відправки, також повинні зробити форму для загрузки файлу.

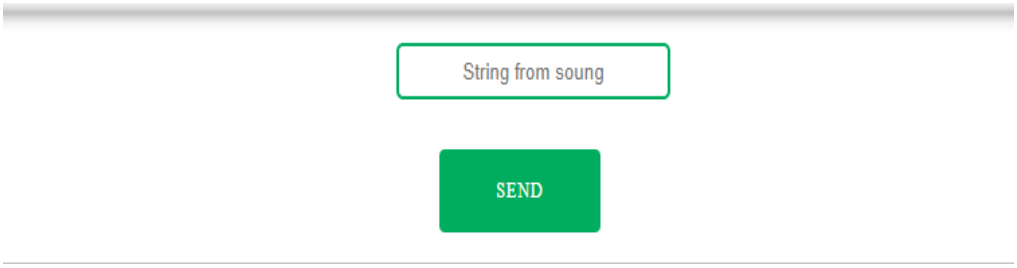


Рисунок 3.6 - Форма вводу фрагменту пісні

Для побудови блоку форми показаної на рисунку, за допомогою розширення JSX створюємо компоненту яка буде відповідальна за форму

```
const MainForm = (props) => {
  return  (
    <div className={main.string_container}>
     <form  onSubmit = {props.handleSubmit}>
     <div>
      <Field component={Input}  className={main.string_container__input}
         validate={[required, maxLength20]}
          name={'someStr'}
          placeholder='String from soung' />
     </div>
     <label></label>
   <div>
 </div>
  <div >
   <button  className={main.container__button}>Send</button>
  </div>
   </form>
 </div>
  )
}
```

Рисунок 3.7 – Побудова компонетни за допомогою JSX

Компонент використовує sendUserLine, обробник для зчитування вхідного значення при будь-якій зміні. MainForm надає нову властивість: onSubmit. Він може використовуватися батьківським компонентом для обробки для натискання кнопки пошуку.

```
sendUserLine = (formData) => {
  let receivedString = formData.someStr.replace(/ /g, '%20');
  this.props.getMusicFromLyrics(receivedString);
}
```

Рисунок 3.8 – Функція отримання обробки та передавання данних

В функції введений рядок, замінює всі пробіли на “%20”- це необхідно для приведення рядка в доступний до зчитування формат для сервера, та передає її в функцію getMusicFromLyrics.

```
}
export const getMusicFromLyrics = (lyrics) => {
  return async (dispatch) => {
    let data = await soundAPI.getMusicFromLyrics(lyrics);
        dispatch(setLyricsData(lyrics));
        dispatch(getMusicData(data.result));
  }
}
```

Рисунок 3.9 – Функція отримання обробки та передавання данних

getMusicFromLyrics повертає функцію, замість звичайного об'єкта. Ця функція приймає аргумент dispatch з Store.

Усередині тіла функції ми спершу відправляємо звичайний синхронний екшен, який повідомляє, що ми починаємо додавання нового об’єкту за допомогою зовнішньої API.

Запит був відправлений на сервер. Потім, робимо GET запит на сервер використовуючи бібліотеку Axios. У випадку позитивної відповіді від сервера, ми передаємо синхронний екшен, використовуючи дані, отримані з сервера.

Для пошуку музики в інтернеті будемо використовувати безкоштовне API розпізнавання музики від сервісу AudD.io. API дозволяє розпізнавати музкику по фрагменту із тексту та фрагменту аудіо файлу. Розпізнавання аудіо потоків за допомогою API коштує 45 доларів на потік на місяць разом із нашим музичним БД або 25 доларів з вашими піснями.

ЦІНА:

0+ запитів на місяць - 5 доларів за 1000 запитів (2 долари за перші 1000);

100 000 запитів на місяць - 420 доларів;

200 000 запитів на місяць - 760 доларів;

500 000 запитів на місяць - 1670 доларів;

1 000 000 запитів на місяць - 2800 доларів;

10 000 000 запитів на місяць - 21000 доларів

Але є безкоштовна пробна версія на 300 пошуків. Будемо використовувати саме її.

```
4   const api_token ='27a15445523a07975211188c7b894ff7';
5   /*let  bodyFormData = new FormData();
6       bodyFormData.append("file", file);
7       bodyFormData.set("api_token", 'e18f6001fd4236175f7dc468d0470702');
8       bodyFormData.set("return", "timecode,apple_music,deezer,spotify");*/
9   const instance = axios.create({
10    baseURL: 'https://api.audd.io/',
11  });
12
13  export const soundAPI = {
14    getMusicFromLyrics(lyrics){
15      let formData = new FormData();
16      formData.append('lyrics', lyrics);
17      formData.set('return','deezer');
18      return instance.get(`findLyrics/?q=${lyrics}&api_token=${api_token}` ).then(response => {
19            console.log(response);
20              console.log(response.data);
21              return response.data;
22            })
23    },
```

Рисунок 3.10 – Функція запиту на сервер .

Для спілкування з сервером використовуємо бібліотеку Axios. У коді на рисунку 3.1.12 використовується метод axios.get (url) який повертає promise, який в результаті в поверне об'єкт response. Необхідні нам дані зберігаються в полі data на рисунку 3.1.12. Якщо необхідно, можна отримати деякі метадані про запит, наприклад статус код (res.status) або більш детально в об'єкті res.request. Axios дозволяє задати параметри які будуть підставляти в усі запити автоматично. Наприклад можна задати базової URL API або ключ авторизації, як показано на рисунку 3.1.12.

.api_token - ключ , по якому сервер ідентифікує нас як тимчасового користувача, та надає доступ до своїх функцій.

URL на який робимо запит проілюстрований на рядку 10 та рядку 18. Також передаємо на сервер введений користувачем рядок та ключ ідентифікації, та на рядку 21 отримуємо від сервера відповідь, яка автоматично записується в змінну data. Далі дані з сервера ми передаємо в грабальний об’єкт Store.

Тепер для відображення отриманої інформації треба підключити чисті компоненти відображення до Redux, створивши деякі компоненти-

контейнери. Технічно, контейнер - це просто React-компонент, який використовує метод store.subscribe() для читання частини Redux-дерева станів і поставляє props, яке він рендерить. Генерація контейнера проходить за допомогою бібліотечної функції React Redux connect(), яка надає багато корисних оптимізацій для запобігання непотрібних ре-рендерів.

Щоб використовувати connect(), потрібно визначити спеціальну функцію mapStateToProps, яка говорить, як трансформувати поточний Redux-стан стору в props, які треба передати та обертається (контейнером) уявлення.

```
export const mapStateToProps = (state) => {
return {
  sound: state.sound.lyrics,
  music:state.sound.music,
  file: state.sound.file,
  musicFile: state.sound.musicFile,
  value: state.sound.value,
  }
}
```

Рисунок 3.11 – Функція надає дані з Store до компоненти.

Після цього, створюємо створюємо компоненту MusicData для візуалізації викликаючи connect() і передав дані з mapStateToProps на зображені на рисунку 3.1.13. Для перегляду відео будемо використовувати ReactPlayer. Завантажуємо компоненту через термінал командою npm і audio-player, та імпотруємо його в проект, та можемо використовувати

Компонент аналізує URL-адресу та завантажує відповідну розмітку та зовнішні SDK для відтворення медіа-файлів з різних джерел . Реквізити можна передавати для управління відтворенням та реагуванням на такі події, як буферизація або закінчення медіа.

```
<ReactPlayer   url={this.props.music[this.state.value].media.length === 2 ?
'nothing':JSON.parse(this.props.music[this.state.value].media)[0].url} playing />
```

Рисунок 3.12 – Компонента ReactPlayer.

Як результат, отримуємо компоненту яка відображує інформацію, що надійшла з серверу: назву, альбом, групу, посилання на ресус дистриб’ютора, та посилання на ресурс перегляду кліпу.

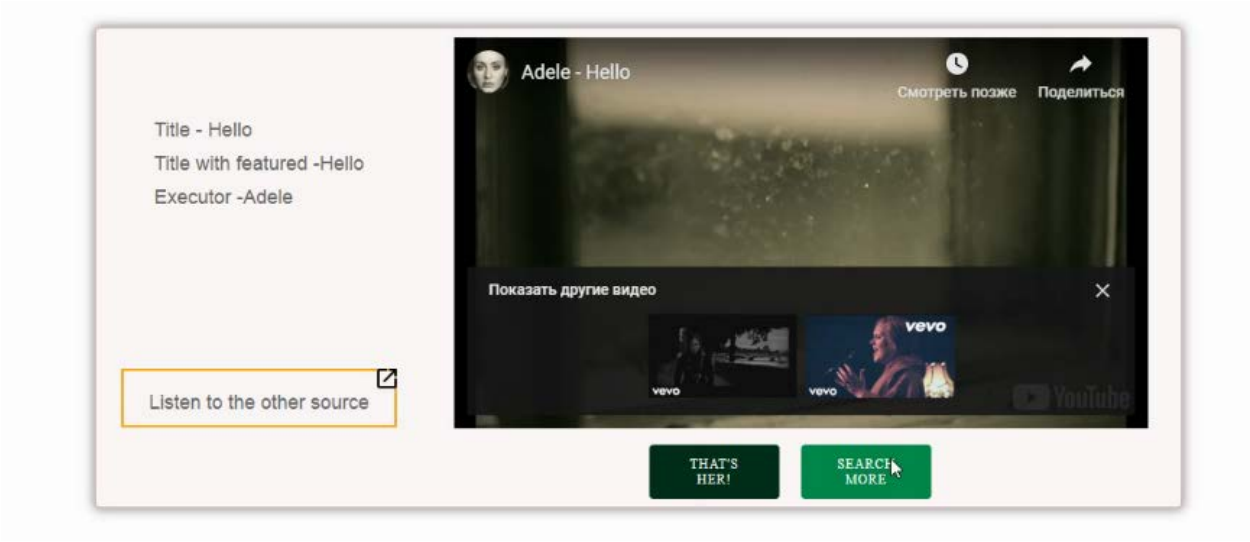


Рисунок 3.13 –Компонента відображення наданої інформації

Аналогічні дії виконуємо для другого способу пошуку пісні через файл з фрагментом цієї пісні. Але з однією відмінністю, а саме на етапі передачі файлу на сервер.

На початку методу відбувається перетворення об'єкта file з деревоподібного в плоску структуру і збереження даних в екземплярі об'єкта FormData . Далі заповнена змінна FormData відправляється AJAX запитом на сервер.

```
saveFile(file) {
  let formData = new FormData();
  formData.append('file', file);
  formData.set('api_token','27a15445523a07975211188c7b894ff7');
  formData.set('return','deezer');
  console.log(file);
  return instance.post('', formData,{
    headers: {
      "Content-Type": "multipart/form-data"
    }
  }).then(response => {
    console.log(response);
    console.log(response.data);
    return response.data;
  })
}
}
```

Рисунок 3.14 – Функція запиту на сервер

Після цього виконуємо дії аналогічно першому способу та відображуємо дані в компоненті Music. Дані, які отримуємо відображаються на екрані: Логотип, назва, альбом, група, плеєр для прослуховування, дата випуску альбому, продюссерський центр та посилання на ресурс дистриб'ютора.

Також, ми маємо можливість прослухати демо -версію пісні. Для прослуховування пісні використовуємо AudoiPlayer.

Завантажуємо компоненту через термінал npm командою

npm i audio-player, імпотруємо його в проект, та можемо використовувати

Особливості плеєра:

1. відтворити віддалений файл;
2. відтворити локальний файл (не для Інтернету);
3. зупинка;
4. пауза;
5. onComplete;
6. onDuration / onCurrentPosition;
7. шукати;
8. відключити звук.

Рисунок 3.15 – Компонента відображення наданої інформації.

Розробка закінчена, тепер потрібно опублікувати сайт в інтернеті. Для цього будемо використовувати GitHub. GitHub є сайтом «соціального кодування». Він дозволяє завантажувати репозиторії коду для зберігання в системі управління версіями Git .

Базове встановлення Github:

- Реєструємось в облікового запису GitHub ;
- Входимо в github.com за ім'ям користувача і паролем.

Підготовка коду для завантаження:

Для початку створюємо папку(репозиторій в який і будемо завантажувати файли) .

Для завантаження файлів в репозиторій використовуємо систему управління версіями Git. Для цього в командному рядку терміналу,знаходячись в середині каталогу веб-сайту, вводимо наступну команду, яка повідомляє інструменту git, щоб він перетворив каталог в репозиторій git:

```
git init
```

Після ініціалізанії папки з файлами, додоаємо всі файли командою:

```
git add .
```

Команда додає всі файли, для подальшого транспортування.

```
git commit –m ‘our comment’
```

Дадамо коментарій до кожного файлу, далі ініціалізуємо репозиторій командою

```
1. git remote add origin https://github.com/SergijBabich/AUDD.git
```

де https://github.com/SergijBabich/AUDD.git - ми повинні скопіювати після створення репозиторія на сайті GitHub.

Нарешті, вводемо в термінал команду,яка в свою чергу додасть файли в репозиторій:

```
git push -u origin master
```

Але для роботи сайту в інтернеті цього не достатньо, оскільки ми викроистовуємо реакт, він надає можливість розташовувати безкоштовно веб сайти на сторінках GitHub за допомогою біліотеки gh-pages

Для ініціплізації бібліотеки, в терміналі вводимо команду npm install gh-pages. Бібліотека автоматично завантажується в проект та дає змогу використовувати ресурси сайту GitHub для відображення сайту.

Додаємо такі залежноті в `package.json`:

```
"scripts": {
    "predeploy": "npm run build",
-   "deploy": "gh-pages -d build",
+   "deploy": "gh-pages -b master -d build",
```

В терміналі вводимо команду:

```
npm run deploy
```

Команда автоматично створює в репозиторії гілку gh-pages для відображення сайту.

Щоб мати доступ до перегляду сайту, потрібно зайти в нашатування резпозиторія на сайті GitHub, розділ Settings, та в блоці Б, в пункті source GitHub pages вибираємо гілку gh-pages-branch.

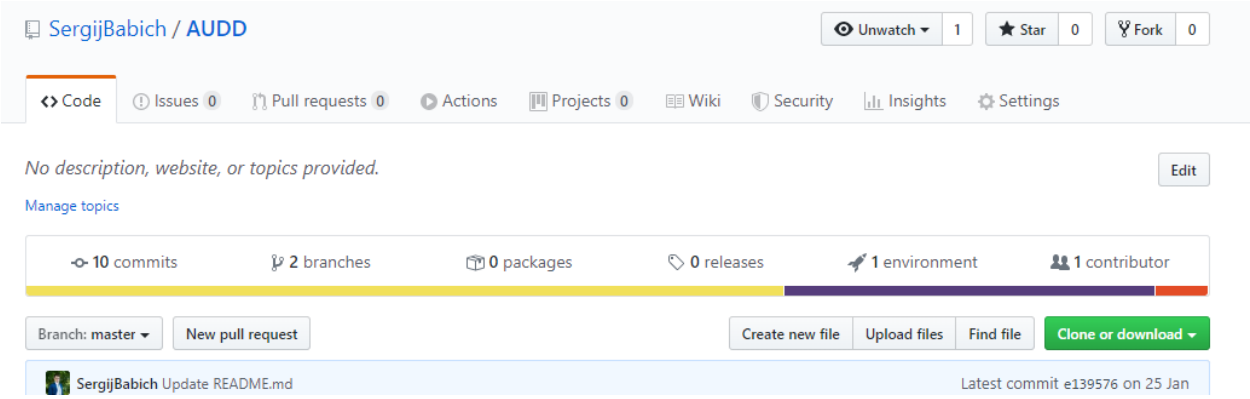


Рисунок 3.16 –Розділ settings.

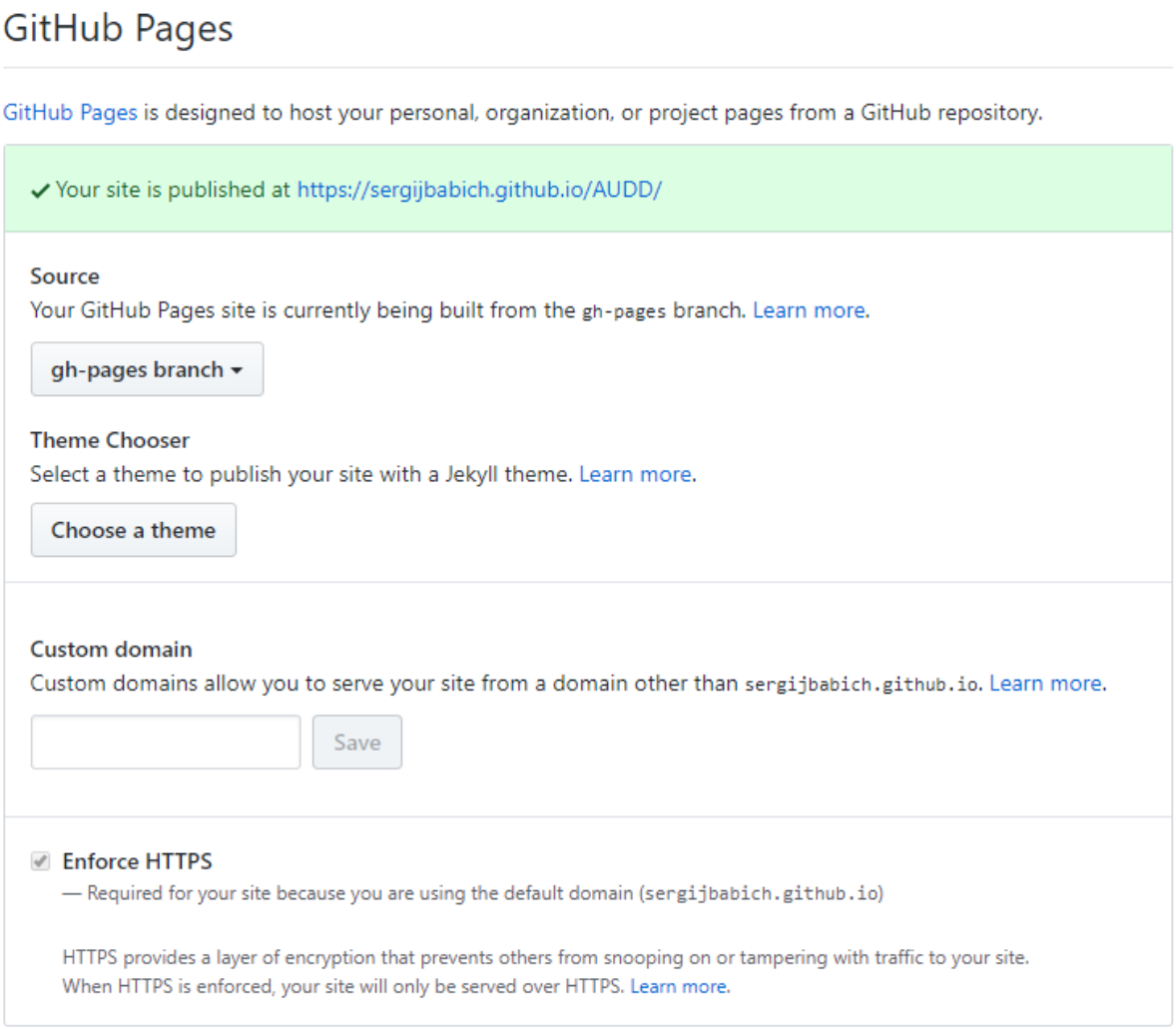


Рисунок 3.17 – Обираємо гілку gh-pages-branch

Ми отримуємо доступ до перегляду сайту з хостингу GitHub.

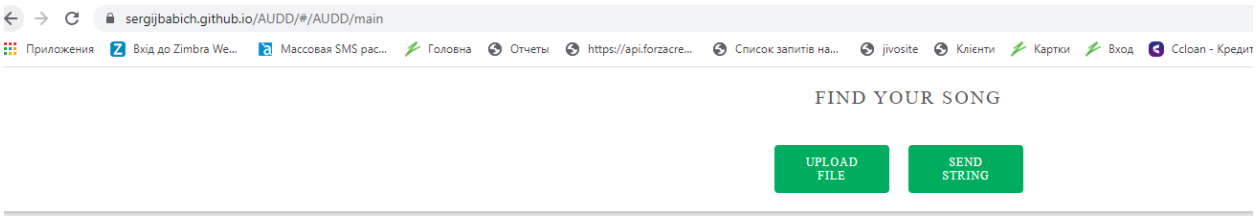


Рисунок 3.18 – Робота сайту на GitHub хостингу

Сайт запущено, тепер кожен користувач моє змогу зайти на сайт та виконати пошук музики по запису з файлу чи по фрагменту з пісні.

Переваги розробки:

- Безкоштовність. Користування сервісов безкоштовне;
- Зрозумілий, простий інтерфейс. Щоб знайти пісню, потрібно натиснути тільки 2 кнопки.
- Не має реестрації. Не потрвбно прходити реєстрація, для користуванням сервісом.

Недоліки:

- Обмежена кількість пошуків. Є можливість только 1000 пошуків;
- Не має історії пошуку.

3.4 Висновки до розділу 3:

- Було розглянуто технологію React та пакет бібліотек;

- На базі бібліотеки React, був розроблений користувацький інтерфейс, який дозволяє вести пошук пісні чи кліпу, за допомогою фрагмента з цієї пісні чи завантаження фрагмента пісні. Дозволяє прослуховувати, переглядати інформацію, та переходити на сайт дистриб'ютора.

Для досягнення поставленої мети вирішено використовували технологію React і систему керування даними Redux. В купі - ці технології повністю задовільняють потреби та задачі, поставлені при розробці інтерфейсу.

- Розглянуто відео-плеери, як елементи бібліотеки React.
- Завантажено готовий проект на хостинг.

ВИСНОВКИ

У ході виконання даної дипломної роботи було досліджено:

1. Технології для розробки користувацьких інтерфейсів.

2. Проведено порівняльний аналіз фреймворків.

3. Розвиток технологій до сучасних показників.

4. Обгрунтовано вибір React фреймворку для побудови користувацьких інтерфейсів.

5. Розроблено веб додаток, що використовує технологію React , яка:

- підтримується мобільними та десктопними браузерами ;

- дозволяє вести пошук пісні чи кліпу, за допомогою фрагмента з цієї пісні чи завантажує фрагмент пісні.

- дозволяє прослуховувати, переглядати інформацію, та переходити на сайт дистриб’ютора

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. В.В. Макаренко, К.О. Трапезон, А.М. Чермянін. Основні вимоги до оформлення атестаційних робіт, дипломних та курсових проектів: методичні рекомендації для студентів усіх форм навчання факультету електроніки. – К.: ФЕЛ НТУУ “КПІ”, 2006. – 112 с..
2. Користувацький інтерфейс операційної системи Windows. URL: https://pidruchniki.com/1402040448804/dokumentoznavstvo/koristuvatskiy_interfeys_operatsiynoyi_sistemi_windows (Дата звернення: 25.05.2020).
3. Історія розвитку користувацького інтерфейсу. URL: https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D1%96%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0 (Дата звернення: 25.05.2020).
4. Етапи розробки користувацького інтерфейсу. URL: https://studopedia.su/12_23303_etapi-rozrobki-koristuvatskogo-interfeysu-Iteratsiyna-priroda-rozrobki.html (Дата звернення: 25.05.2020).
5. Що таке React. URL: https://habr.com/ru/company/ruvds/blog/343022/ (Дата звернення: 26.05.2020).
6. Порівняльна траблиця JavaScript. URL: bit.ly/2WZaww (Дата звернення: 26.05.2020).
7. Релиз Electron 4.4.0. URL: bit.ly/3c3VE3Q (Дата звернення: 26.05.2020).
8. Thinkwik. React Native. URL: https://medium.com/@thinkwik/react-native-what-is-it-and-why-is-it-used-b132c3581df (Дата звернення: 26.05.2020).
9. Основи Vue.js. URL: https://metanit.com/web/vuejs/1.1.php (Дата звернення: 26.05.2020).
10. jQuery Introduction. URL: https://www.w3schools.com/jquery/jquery_intro.asp (Дата звернення: 27.05.2020).

11. Патриция Н.. Что лучше выбрать в 2020 году – React или Vue?. 2019. URL: https://habr.com/ru/company/ruvds/blog/470413/ (Дата звернення: 27.05.2020).
12. Почему стоит использовать React JS при разработке приложений. URL: https://xbsoftware.ru/blog/pochemu-stoit-ispolzovat-react-js-razrabotke-prilozhenij/ (Дата звернення: 27.05.2020).
13. Вілер. К.. Разбираемся с Flux, реактивной архитектурой от Facebook. 2014. URL: https://habr.com/ru/post/246959/ (Дата звернення: 27.05.2020).
14. Єрошкін І.. Как работает Redux?. 2019. URL: https://medium.com/@ivaneroshkin/как-работает-redux-a967d8616398 (Дата звернення: 27.05.2020).
15. Create a New React App. URL: https://ru.react.js.org/docs/create-a-new-react-app.html (Дата звернення: 28.05.2020).
16. Cristian S.. How to create a three-layer application with React. 2018. URL: https://medium.com/programming-essentials/how-to-create-a-three-layer-application-with-react-8621741baca0 (Дата звернення: 28.05.2020).
17. Структура файлов. URL: https://ru.reactjs.org/docs/faq-structure.html (Дата звернення: 28.05.2020).
18. Пишем полноценное приложение на React с нуля за час. URL: https://tproger.ru/translations/react-basic-weather-app/ (Дата звернення: 28.05.2020).
19. Графічний інтерфейс користувача. URL: https://uk.wikipedia.org/wiki/Графічний_інтерфейс_користувача (Дата звернення: 30.05.2020).
20. Інтерфейс користувача. URL: https://uk.wikipedia.org/wiki/Інтерфейс_користувача (Дата звернення: 30.05.2020).
21. Модульне серидовища для навчання. URL: https://msn.khnu.km.ua/pluginfile.php/277742/mod_resource/content/2/%D0%

9B%D0%B5%D0%BA%D1%86%D1%96%D1%8F4.pdf (Дата звернення: 22.05.2020).

ДОДАТОК А
SUMMARY

Summary

The React library was first released by Facebook in 2013. To understand how popular this technology has become in the past, let's turn to a survey of developers conducted by StackOverflow this year. More than 50,000 developers shared information about their work and professional preferences. In addition to the more or less predictable results that describe the state of the IT industry today, there is also something interesting about React itself. This library has become one of the most popular and sought-after technologies, as well as the most trendy technology on StackOverflow.

React is a library for creating user interfaces. One of its distinctive features is the ability to use JSX, a programming language with a close HTML syntax, which is compiled into JavaScript. Developers may require more application performance with Virtual DOM. With React we can create isomorphic applications that will help save us from unpleasant situations when the user is looking forward to when the data download will finally finish and something other than the download animation will finally appear on his computer screen.

Created components can be easily modified and reused in new projects. A high percentage of code reuse reduces development time, which in turn leads to a higher level of quality control. Using React Native mobile applications for Android and iOS, using the experience of JavaScript and React development. [12]

How does React benefit users and businesses?

• Virtual DOM can increase the performance of highly loaded applications, which can reduce the likelihood of possible inconveniences and improve the user experience;

• Using the isomorphic approach helps to render pages faster, thus allowing users to feel more comfortable when working with the application. Search engines index such pages better. Since the same code can be used in both the client and server part of the program, there is no need to duplicate the same functionality. As a result, development time and costs are reduced;

• Reusing the code has made it much easier to create mobile applications. The code that was written during the creation of the site can be reused to create a mobile application. If you plan to use not only the site but also the mobile application, there is no need to hire two large development teams. [12]

Isomorphic applications.

When we talk about an isomorphic application or isomorphic JavaScript, we mean that it is possible to use the same code in both the server and client part of the program. When a user opens a site in their browser, the content of the page must be downloaded from the server. In the case of SPA-applications (Single Page Application), this may take some time. During the download, users see either a blank page or a download animation. Given that by today's standards, waiting more than two seconds can be quite a noticeable inconvenience to the user, reducing the download time can be extremely important. And here's another important problem: search engines do not index such pages as well as we would like.

Running JavaScript code on the server side helps fix these problems. If you create isomorphic applications, it is possible to get significant benefits by rendering on the server side. After loading the page, we can still continue rendering the components. This ability to render pages on both the server and the client leads to significant benefits, such as the ability to better index pages with search engines and improve the user experience.

Moreover, this approach reduces the time spent on development. When using some modern frameworks, we need to create components that need server-side rendering, as well as templates for client-side applications. React developers can create components that work on both sides.

Virtual DOM.

Document Object Model, or DOM, is a way to represent and interact with objects in HTML, XHTML, and XML documents. According to this model, each such document is a hierarchical tree of elements, called a DOM tree. Using special

methods, it is possible to access certain elements of the document and change them as we want. When we create a dynamic interactive web page, we want the DOM to update as quickly as possible after the state of a particular item changes.

For this task, some frameworks use a technique called "dirty checking" and is a regular survey of the document and check for changes in the data structure. Such a task can be a real headache in the case of highly loaded applications. Virtual DOM, in turn, is stored in memory. That is why at the moment when the "real" DOM changes, React can change the Virtual DOM in an instant. React "collects" such changes, compares them with the state of the DOM, and then redraws the changed components.

This approach does not require regular DOM updates. This is why higher performance React applications can be achieved. The second consequence follows from the isomorphic nature of React: the ability to render on the server side just like on the client side. Because many different libraries have been developed for the reactor, it has become much easier to work with and play video and audio files.

ReactPlayer - a library for video playback, has many settings. Not only for viewing but also for quick video editing, right while watching. [12]

JSX

At the heart of any basic website are HTML documents. Web browsers read these documents and display them on your computer, tablet, or phone as web pages. During this process, browsers create something called a Document Object Model (DOM), a representational tree of how the web page is arranged. Developers can then add dynamic content to their projects by modifying the DOM with languages like JavaScript.

JSX (short for JavaScript eXtension) is a React extension that makes it easy for web developers to modify their DOM by using simple, HTML-style code. And—since React JS browser support extends to all modern web browsers—JSX is compatible with any browser platform you might be working with.

This isn’t just a matter of convenience, though—using JSX to update a DOM leads to significant site performance improvements and development efficiency. How? It’s all about the next React feature, the Virtual DOM.

If all of this makes sense but you’re still wondering, “what IS React code?” you can get a visual idea of what React looks like straight from this React examples website. Each of the projects listed here gives an idea of what’s possible with React JS and a look at the source code used to build it.

Meanwhile, if you’re still wondering exactly “what can be done with React JS?” and you’re interested in React JS examples for beginners, these simple projects curated by the official React website provide React JS templates to walk through as you learn React JS.

And finally, if you’re looking for a React JS tutorial to go along with those React JS examples for beginners, you can go straight to the source with ReactJS.org’s Intro to React, a React JS tutorial that requires no previous experience or knowledge. Whether you’ve already been working with the React library, or you’re still at square one figuring out how to install React, this is a great place to start.

It’s common to see React JS described as both a JavaScript library AND a JavaScript framework, so which is it? Or is it both?

At Skillcrush, our curriculum team defines React JS as a JavaScript library and not a framework. This is an important distinction.

The difference between JavaScript libraries (like React) and JavaScript frameworks (like Angular) lies in the fact that—in the case of a library—the developer applies library code in individual instances that call for it. When it comes to frameworks, however, the framework creates a scaffolding that arranges your website or application and provides dedicated areas for framework code to be plugged-in.

To dig into the difference between a library like React JS vs Angular (a framework), you can think of code from a Javascript library in terms of furniture

and decorations for a house you've already built, while a framework is a model home template you use to build the house.

React JS is sometimes mistaken for a full-blown framework since its robust ecosystem and extensibility makes it such a versatile JavaScript library. Remember, when you use React JS to build website and web application UIs, you have access to:

React code snippets and components (building blocks of React code used to create specific parts of a user interface)

The option to use JSX to directly manipulate your DOM

A Virtual DOM to improve your website's performance

But on top of all that, React JS is an open source project, meaning anyone can download and modify its source code for free. This also means that, whatever specific UI function you're hoping to address with React JS, there's a React library to meet your needs. Your React library size can grow exponentially with React's community curated library add-ons, ranging from collections of individual UI features to complete React JS templates for building UI's from the ground up.

(back to top)

Finding success in web development means using the right tools to make your code as effective and efficient as possible. And—when it comes to building user interfaces—React JS is a tool you need to know how to take advantage of.

If you're ready to React (plus other crucial web developer skills like JavaScript, HTML, and CSS), you can sign up today for our Skillcrush Front End Developer + React JavaScript Course. This online course is designed to be completed in four months by spending just an hour a day on the materials.