Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«**Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень**»

Виконав: студент 2-го курсу, групи 1КІ-20м
спеціальності 123 — Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
Коваленко М. М.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. ОТ
Войцеховська О. В.
(прізвище та ініціали)
«____» _____ 2021 р.

Опонент: к.т.н., професор каф. ЗІ
Кондратенко Н.Р.
(прізвище та ініціали)
«____» _____ 2021 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
(прізвище та ініціали)
«____» _____ 2021 р

Вінниця ВНТУ 2021

АНОТАЦІЯ

УДК 004.4

Коваленко М. М. Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень. Магістерська кваліфікаційна робота зі спеціальності 123 — комп'ютерна інженерія, освітня програма — комп'ютерна інженерія. Вінниця: ВНТУ, 2021. 106 с.

На укр. мові. Бібліогр.: 33 назв; рис.: 47; 13 табл.

Магістерська кваліфікаційна робота присвячена розробці веб-застосунку для ретрансляції навчальних матеріалів в умовах карантинних обмежень. Він дозволяє пришвидшити з’єднання між веб-застосунком для показу трансляції та програмним забезпеченням для створення трансляцій. У теоретичній частині роботи розглянуті аналоги, та проведений їх порівняльний аналіз. Проведена розробка структури додатку та виконаний аналіз функціональної частини веб-застосунку. У верифікаційній частині магістерської роботи проведено тестування та розроблено інструкцію користувача.

Веб- розроблена для використання у веб-браузерах з використанням середовища розробки Visual Studio Code та мови програмування JavaScript.

Ключові слова: веб-застосунок, ретрансляція, стрімінг, Visual Studio Code, тестування, програмне забезпечення.

ABSTRACT

Kovalenko M.M. Cross-platform web application for retransmission of educational materials under quarantine restrictions. Magister's thesis in specialty 123 - computer engineering. Vinnitsa: VNTU, 2021. — 106 p.

In Ukrainian language. Bibliographer: 45 titles; fig.: 47; tabl. 13.

The master's qualification work is devoted to the development of a web application for retransmission of educational materials in the conditions of quarantine restrictions. It speeds up the connection between a webcast to show broadcasts and webcast software. In the theoretical part of the work analogues are considered, and their comparative analysis is carried out. In the part of the work where the structure of the application is directly developed and the analysis of the functional part of the web application is performed. In the testing part of the master's thesis, testing was conducted and user instructions were developed.

The program is designed for use in web browsers using the Visual Studio Code development environment and JavaScript programming language.

Keywords: web application, structure, Visual Studio Code, testing, softwa

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – _12 Інформаційні технології_____
Спеціальність – Комп'ютерна інженерія_____
Освітньо-професійна програма – _ Комп'ютерна інженерія _____

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____д.т.н., проф. Азаров О.Д.
_____ 2021 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Коваленко Максим Максимович.
(прізвище, ім'я, по батькові)

1. Тема роботи. «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень»

керівник роботи Войцеховська О.В., к.т.н., доцент

затверджені наказом вищого навчального закладу від "24" 09. 2021 року № 277

2. Строк подання студентом роботи ___ _____ 2021 року

3.Вихідні дані до роботи: створені методи розробки веб-застосунків, навчальні матеріали на тему створення веб-застосунків, технічні документації мови JavaScript та бібліотеки React. Веб-застосунки-аналоги.

4. Зміст текстової частини: аналіз сучасних методів та технологій створення веб-застосунків; аналіз та вибір інструментів для розробки веб-застосунку для ретрансляції навчальних матеріалів; розробка веб-застосунку для ретрансляції навчальних матеріалів, тестування веб-застосунку та рекомендації користувачу; економічний розділ.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):

Структура веб-застосунку для ретрансляції навчальних матеріалів,. Сторінка генерації ретрансляційного ключа. Структурна схема серверного обміну. Схема роботи потоків в Gulp. Зовнішній вигляд вікна відео плеєру трансляції.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Войцеховська О.В., к.т.н., доцент		
5	Кавецький В.В к.т.н., доцент		

7. Дата видачі завдання___24.09.2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
	Постановка задач роботи	24.09.2021	
	Пошук матеріалів про перспективні напрямки розвитку технологій розробки веб-застосунків		
	Пошук та аналіз методів ретрансляції		
	Аналіз сучасних інструментів для створення веб-застосунків		
	Підготовка матеріалів та опис веб-застосунку		
	Підготовка економічної частини		
	Оформлення пояснювальної записки		
	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	17.12.2021	

Студент _____ _____
(підпис)

Керівник роботи_____ _____
(підпис)

ЗМІСТ

ВСТУП

В умовах пандемії COVID-19 все більше уваги приділяється засобам навчання школярів та студентів шляхом використання мережі Інтернет.

Веб-застосунки, для надання навчальних матеріалів бувають різні, як для підвищення кваліфікації, так і в цілях розважального контенту. Наприклад веб-застосунки з різними курсами, або такі популярні сервіси як YouTube, Twitch та інші.

Існує велике різноманіття веб-застосунків, які мають певні курси для навчання вузького кола людей, проте застосунків ціль яких полягає у навчанні студентів та школярів не так багато, такі сервіси більше орієнтовані на розважальний, ніж на навчальний контент. При цьому вони мають обмежений функціонал або потребують використання проміжних сторонніх сервісів для ретрансляції.

Отже, **актуальною** є розробка веб-застосунку для ретрансляції навчальних матеріалів в умовах карантинних обмежень, що будуть напряму підключатись до стороннього програмного забезпечення для створення трансляцій, зокрема без використання сторонніх сервісів. Це дасть можливість педагогічним працівникам записувати матеріал та використовувати його для показу студентам у прямому ефірі.

Метою даної роботи є створення веб-застосунку для ретрансляції навчальних матеріалів, задля покращення комунікації в умовах карантинних обмежень.

Для досягнення вище вказаної мети поставлено та вирішено такі завдання:

— аналіз сучасного стану розвитку веб застосунків з метою навчання;

— дослідження сучасних аналогів веб-застосунків для ретрансляцій;

— розробка вимог до бібліотек та фреймворків для розробки додатку;

— аналіз середовищ розробки веб-застосунків для домашнього використання;

— вибір інструментів для розробки веб-застосунку;

— розробка архітектури веб-застосунку;

— розробка клієнтської та серверної частини веб-застосунку;

— розробка методу трансляції матеріалів для відображення на медіа плеєрі;

— реалізація відеоплеєру, для перегляду навчальних матеріалів.

Об’єкт дослідження — процес взаємодії веб-застосунків зі стороннім програмним забезпеченням для створення трансляцій.

Предмет дослідження — програмні засоби для перегляду навчальних матеріалів з використанням технологій розробки веб-застосунків для використання у браузері.

Наукова новизна отриманих результатів: набув подальшого розвитку метод ретрансляції навчальних відеоматеріалів, який дає можливість покращити процес взаємодії веб-застосунків зі стороннім програмним забезпеченням для створення трансляцій, зокрема дозволяє напряму підключатись до такого програмного забезпечення для створення трансляцій без використання проміжних сервісів чи серверів.

Практична цінність отриманих результатів полягає в тому, що на основі проведених теоретичних досліджень і отриманих наукових результатів розроблено архітектуру веб-застосунку, що буде корисний для викладачів або студентів в умовах карантинних обмежень, призначений для покращення процесу навчання шляхом ретрансляції.

Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримано автором особисто. Достовірність теоретичних положень підтверджена результатами тестування розробленого мобільного додатку.

Апробація результатів кваліфікаційної роботи: доповідь на Всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2022)» [1].

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ СТВОРЕННЯ ВЕБ-ЗАСТОСУНКІВ

1.1 Методи розробки веб-застосунків

Розробка веб-застосунків базується на написанні коду, сформованому трьома елементами, такими як HTML, CSS, JavaScript та певними додатками (бібліотеками, фреймворками), які спрощують написання та підвищують функціонал додатку.

Створення та розробка саме кросплатформених веб-застосуноків, тобто таких, які можуть використовуватись на більшості пристроїв, підключених до мережі Інтернет (таких як мобільні телефони, комп'ютери, телевізори та інші), має свої особливості.

Створення кросплатформених веб-застосунків включає в себе такі кроки:

— визначення концепції додатку та аналіз схожих існуючих додатків на ринку;

— визначення проблеми, яку повинен вирішити додаток;

— планування робочого процесу веб-додатку, визначивши шлях до вирішення проблеми,

— створення візуальної карти, за якою буде видно детальні кроки розробки проекту;

— створення прототипу додатку та відкриття до нього доступу певній кількості користувачів, щоб протестувати додаток та отримати певний зворотний зв'язок (фідбек);

— визначення технологій, що будуть використовуватись: бібліотеки, фреймворки, різноманітні програмні інтерфейси (API) такі, які будуть підходити під вирішення задачі, яка стоїть перед розробником проекту;

— тестування веб-додатка — це постійний процес, який зазвичай відбувається під час і після фази створення. Причому можна автоматизувати тестування або зробити це вручну. На етапі тестування потрібно спробувати

охопити тестування функціональності, зручності використання, сумісності, безпеки та продуктивності.

Ще одним важливим пунктом створення веб додатку, є забезпечення кросплатформеності додатку, щоб була можливість в будь який час, скористатись ним на різних присторях.

У обчислювальній техніці кросплатформений або мультиплатформений атрибут — це атрибут комп'ютерного програмного забезпечення або обчислювальних методів і концепцій, які реалізовані та взаємодіють на кількох комп'ютерних платформах. Кросплатформене програмне забезпечення можна розділити на два типи: один потребує окремої збірки або компіляції для кожної платформи, яку він підтримує, а інший може бути запущений безпосередньо на будь-якій платформі без спеціальної підготовки. Наприклад, програмне забезпечення, написане мовою інтерпретації, або попередньо скомпільований портативний байт-код, для якого інтерпретатори або пакети є загальними або стандартними компонентами всіх платформ.

Для створення динамічних HTML-сторінок зазвичай використовуються спеціальні серверні розширення, які називаються скриптами. Типовим завданням для сценарію є отримання інформації з якогось зовнішнього джерела, яка потім представляється у вигляді HTML-документа і передається на сервер, який, у свою чергу, надсилає її клієнту. Крім того, скрипти дозволяють взаємодіяти з клієнтом в інтерактивному режимі, обробляючи дані, надіслані від клієнта до сервера.

Перше завдання — розробити та створити статичну веб-сторінку (статична частина веб-документа). Для його вирішення використовуються спеціальні засоби для розробки HTML-документів. Друге завдання передбачає розробку інструментів, що розширюють можливості веб-сервера, і полягає у реалізації можливості динамічної зміни вмісту сторінок у поєднанні з можливістю інтерактивної роботи. Третя проблема полягає в створенні розширень на стороні сервера з використанням різноманітних інструментів розробки.

У процесі створення веб-сторінки прийнято розрізняти дві складові - веб-дизайн і веб-програмування. Між ними немає чіткої межі. Найчастіше під веб-дизайном розуміють розробку статичної частини веб-сторінки в HTML. Включення фрагментів Java в HTML-документ зазвичай є доменом веб-програмування. Розробка розширень для веб-сервера належить виключно до області веб-програмування.

Веб-програми використовують протокол HTTP для спілкування. Для передачі бінарних файлів по протоколу HTTP використовується специфікація MIME (Multipurpose Internet Mail Extension), згідно з якою формат даних описується так: <тип> / <підтип>.

1.2 Аналіз існуючих веб застосунків для навчання, їх переваги та недоліки

Веб застосунків для навчання онлайн існує досить багато. Кожен із них має як свої переваги, так і недоліки. Найпопулярнішими є: coursera.org, Edx.org, habr.com.

Coursera.org — особливістю даної платформи є курси з ігровими елементами, що є дуже ефективним способом навчання. Також цей додаток має досить простий інтерфейс та зовнішній вигляд (рисунок 1.1).

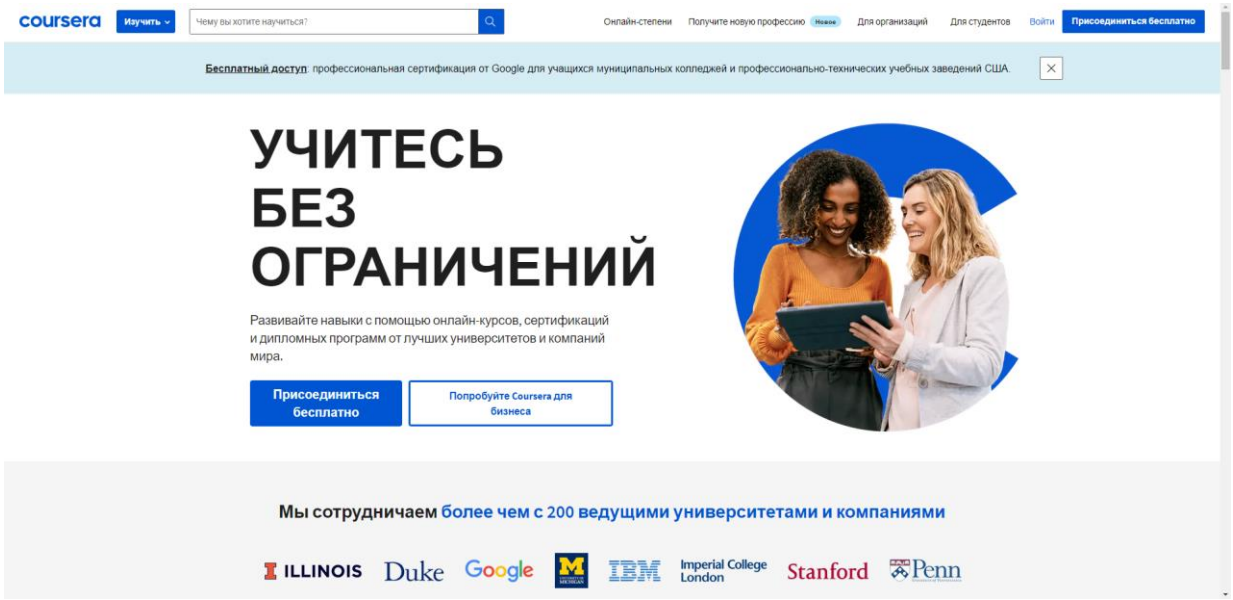


Рисунок 1.1 — Зовнішній вигляд веб застосунку Coursera

Основними перевагами, що відрізняють даний веб застосунок від інших є:

— платформа пропонує велику кількість різноманітних курсів для широкого спектру спеціальностей;

— постійний зворотній зв’язок, що дозволяє в будь-який момент часу звернутись за допомогою з приводу різних курсів або функціоналу сайту;

— велика кількість курсів на вибір;

— геймифіковані курси.

Також слід зазначити недоліки даного додатку:

— орієнтованість на бізнес та отримання роботи, що займає більшу частину функціоналу;

— для використання 90% контенту, необхідна місячна підписка;

— орієнтованість здебільше на англомовних користувачів;

— відсутність відеоматеріалів.

Наступним розглянемо додаток Edx.org. Це популярна платформа з навчальними матеріалами, що зарекомендувала себе, як сервіс з великою кількістю лекцій та відеоматеріалів з відомих університетів. Зовнішній вигляд веб застосунку подано на рисунку 1.2.

Слід визначити ряд переваг, що властиві даній платформі:

— простота навігації по матеріалам;

— велика кількість категорій навчальних матеріалів;

— потужна оптимізаця;

— великий показник підтримки.

Недоліками даного ресурсу є:

— дуже великі ціни на навчальні матеріали та курси;

— не інтуітивний інтерфейс;

— орієнтування на англомовних користувачів;

— відсутність форми відправлення повідомлень користувачу;

— відсутність локалізації.

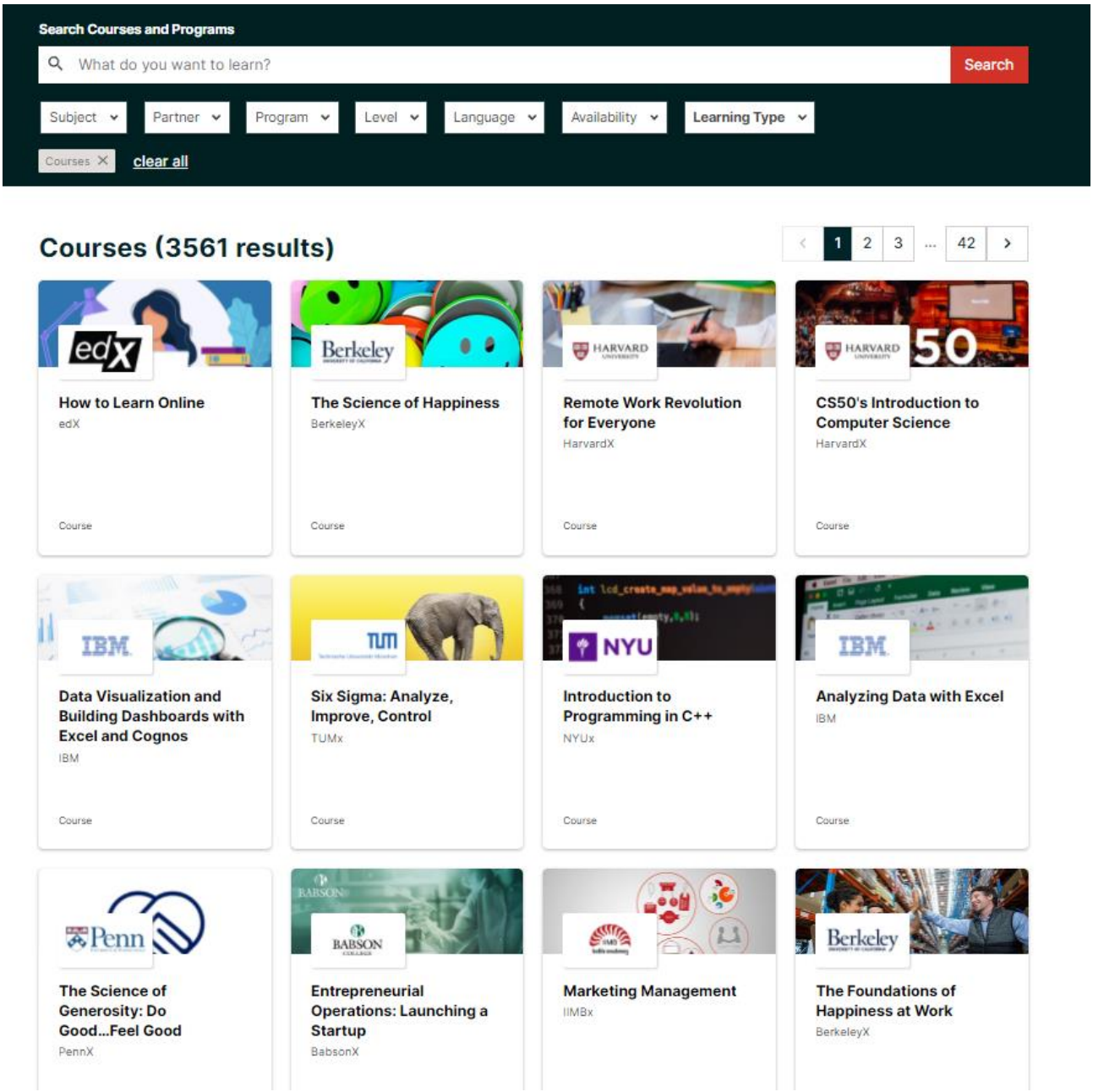


Рисунок 1.2 — Зовнішній вигляд веб застосунку Edx

Ще одним з найвикористовуваніших додатків є Habr.com — російськомовний сайт у форматі системи тематичних колективних блогів з елементами новинного сайту, створений для публікації новин, аналітичних статей, думок, пов’язаних з інформаційними технологіями, бізнесом та Інтернетом (рисунок 1.3).

Перевагами даного застосунку є:

— відкрита платформа для поширення та розповсюдження знань;

— прямий контакт зі спеціалістами в різних сферах.

Недоліками веб застосунку є:

— застарілий інтерфейс;

— невелика кількість навчальних сфер.

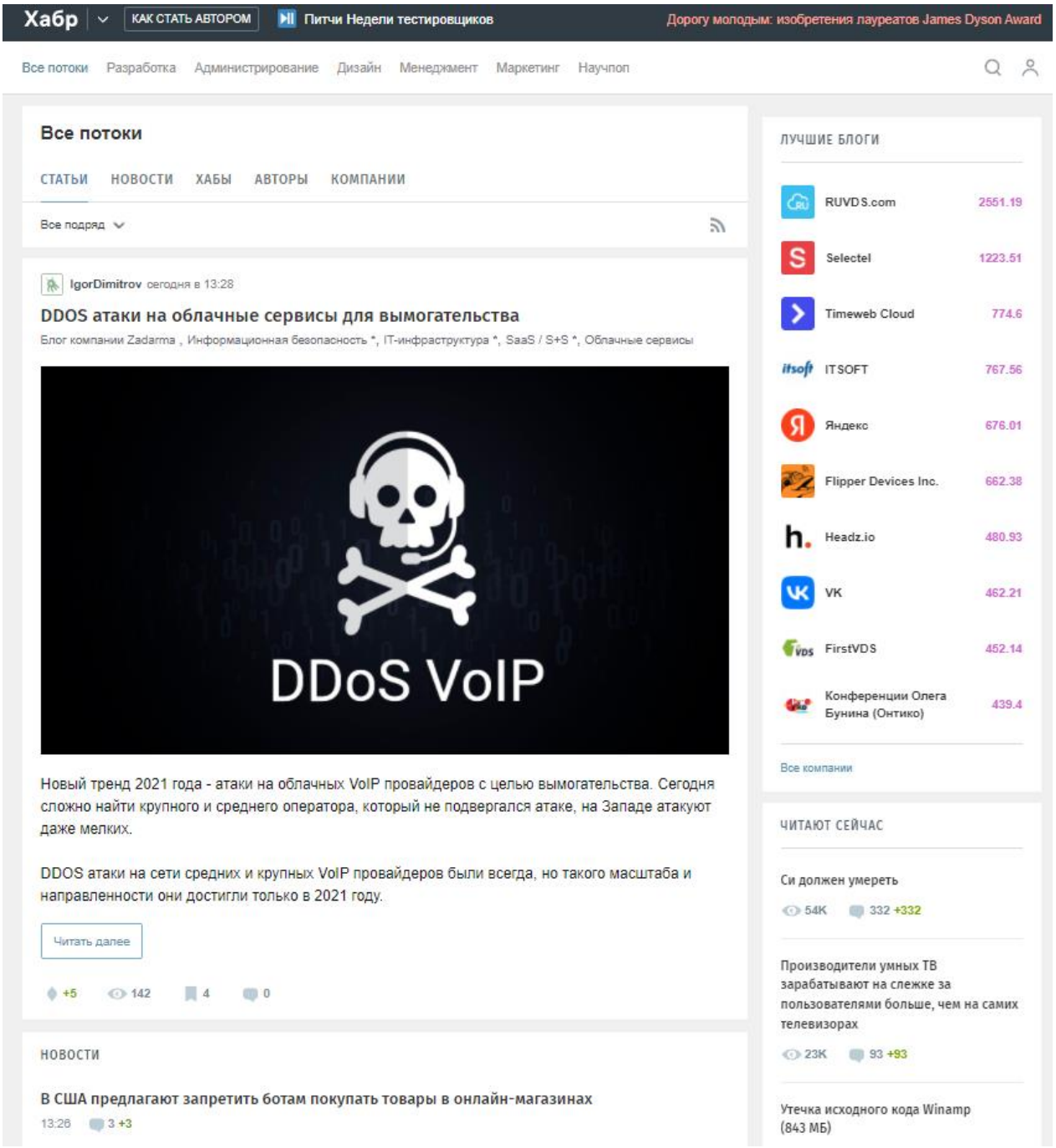


Рисунок 1.3 — Зовнішній вигляд ресурсу Habr

Отже на основі проаналізованих веб застосунків, які створені для розповсюдження знань, підвищення кваліфікації, узагальнимо їх можливості,

а саме переваги та недоліки, в таблиці, що в подальшому дозволить позбутися наявних недоліків у розробці власного веб застосунку.

Результати аналізу наведено у таблиці 1.1.

Таблиця 1.1 — Порівняльний аналіз розглянутих аналогів

Параметр	“coursera.org”	“edx.org”	“habr.com”
Функціональний інтерфейс	+	+	-
Безкоштовний функціонал	-	-	+
Адаптивний дизайн	+	+	-
Відеоперегляд матеріалів	-	+	-

Отже, в розроблюваному веб-застосунку необхідно врахувати такі моменти: дизайн повинен бути адаптивний для можливого використання веб-застосунку з смартфонів, інтерфейс має бути мінімалістичним, проте досить функціональним, для відеоперегляду матеріалів слід застосувати продвинуту систему ретрансляції.

1.3 Аналіз та вибір середовища розробки для створення веб застосунку

Підбір середовища розробки, є дуже важливою частиною створення веб застосунків. Різні середовища мають унікальний функціонал для своїх задач, тому необхідно порівняти існуючі варіанти середовищ розробки та обрати оптимальний для поставленої задачі, а саме розробки веб застосунку для ретрансляції навчальних матеріалів.

Отже, на основі поставленої задачі розглянемо такі середовища розробки:

— VSCode;

— Intellij idea;

— Atom.

VSCode — це інтегроване середовище розробки, створений компанією Microsoft для Windows. Позиціонується як легкий у користуванні редактор кода для створення кросплатформенних додатків та розробки веб- та хмарних-додатків. Інтерфейс даного середовища розробки показано на рисунку 1.4.

VSCode надає можливість створити унікальний інтерфейс, шляхом підключення різноманітних інструментів для поставлених задач. Також має вбудований інструмент для роботи з Git, різні функції для редагування, маркування коду та потужні інструменти рефакторингу коду.

Розглянемо особливості даного середовища розробки як сніппети, IntelliSense, клавіатурні скорочення та відладчик.

Використання сніпетів — це серйозний спосіб підвищення продуктивності програміста. Справа в тому, що вони допомагають автоматизувати ручну працю;

IntelliSense — це система автозавершення коду. Вона є інтелектуальною, що вигідно відрізняє її з інших подібних систем.

Велика кількість вбудованих налаштувань для комбінацій клавіш на клавіатурі, що допомагають в рефакторингу та написанні коду.

У VS Code є відладчик, який дозволяє запускати код і встановлювати в ньому точки зупинки, вказуючи місця, в яких виконання програми повинно бути припинено. Це дозволяє, не залишаючи редактора, налагоджувати програми, аналізуючи їх внутрішній стан під час виконання. Відладка — це більше, ніж виведення повідомлень у консоль інструментів розробника Chrome. У ході налагодження можна дізнаватися про те, що відбувається всередині програми і, в результаті, виявляти і усувати джерела різних проблем. Якщо у програміста, наприклад, є підозри з приводу правильності роботи якоїсь функції, він може, в ході налагодження, виконати цю функцію в покроковому режимі, досліджуючи стан процесу в ході цього процесу.

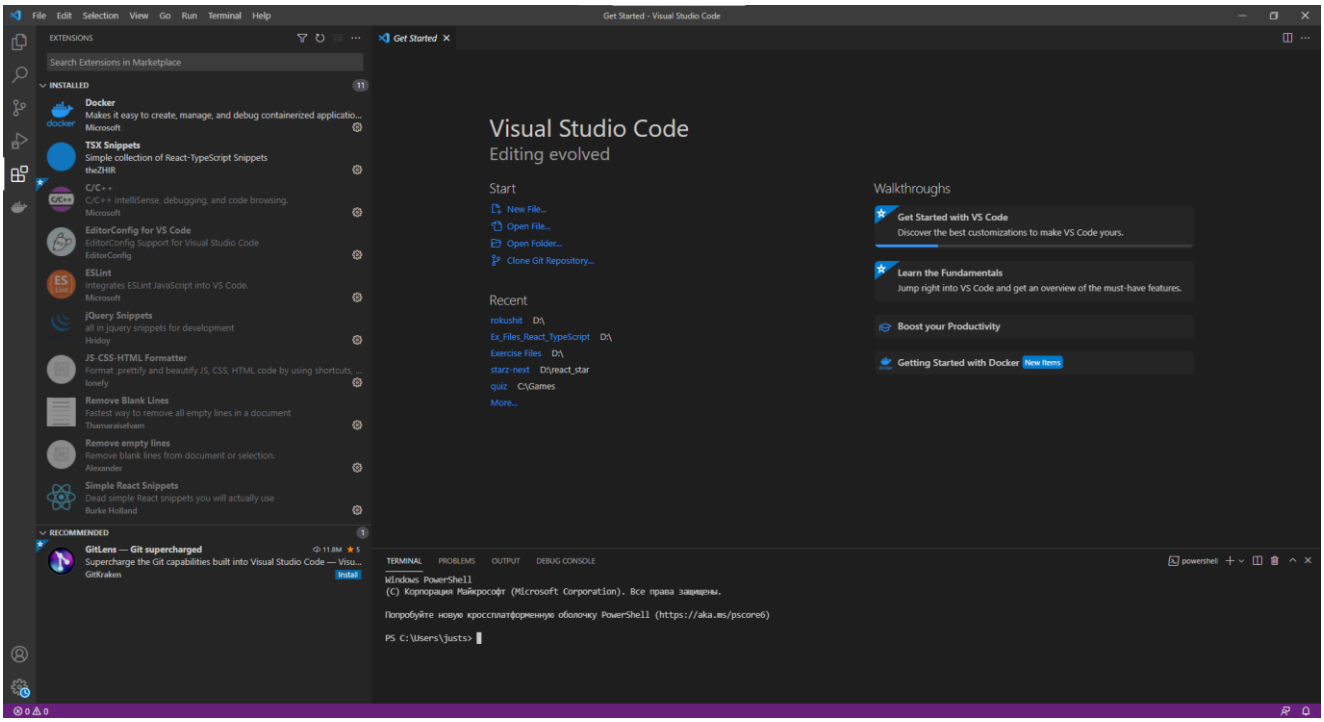


Рисунок 1.4 — Зовнішній вигляд редактору коду VSCode

Intellij idea — інтегроване середовище розробки для багатьох мов програмування таких як: Java, Python, JavaScript та інші. Створена в 2001 році і була здебільше у використанні розробниками на Java. Однак, на даний момент має величезний функціонал для розробки програм на багатьох мовах програмування. Має дві версії Community Edition та Ultimate Edition [17].

Community Edition — безкоштовна версія даного середовища розробки, має більшу частину інтеграцій та підключень з різними мовами програмування.

Ultimate Edition — є платною версією Intellij idea, має повний набір підключень до різноманітних фреймворків та API. А також, побудову UML діаграм, підрахування покриття коду та інші.

Intellij idea має такі переваги:

— розумне автодоповнення, інструменти для аналізу якості коду, зручна навігація, розширені рефакторинги та форматування для Java, Groovy, Scala, HTML, CSS, JavaScript, CoffeeScript, ActionScript, LESS, XML та багатьох інших мов;

— підтримка всіх популярних фреймворків та платформ, включаючи Java EE, Spring Framework, Grails, Play Framework, GWT, Struts, Node.js, AngularJS, Android, Flex, AIR Mobile та багато інших;

— інтеграція з серверами програм, включаючи Tomcat, TomEE, GlassFish, JBoss, WebLogic, WebSphere, Geronimo, Resin, Jetty та Virgo;

— інструменти для запуску тестів та аналізу покриття коду, включаючи підтримку всіх популярних фреймворків для тестування.

Зовнішній вигляд та частина функціоналу Intelij Idea надано на рисунку 1.6.

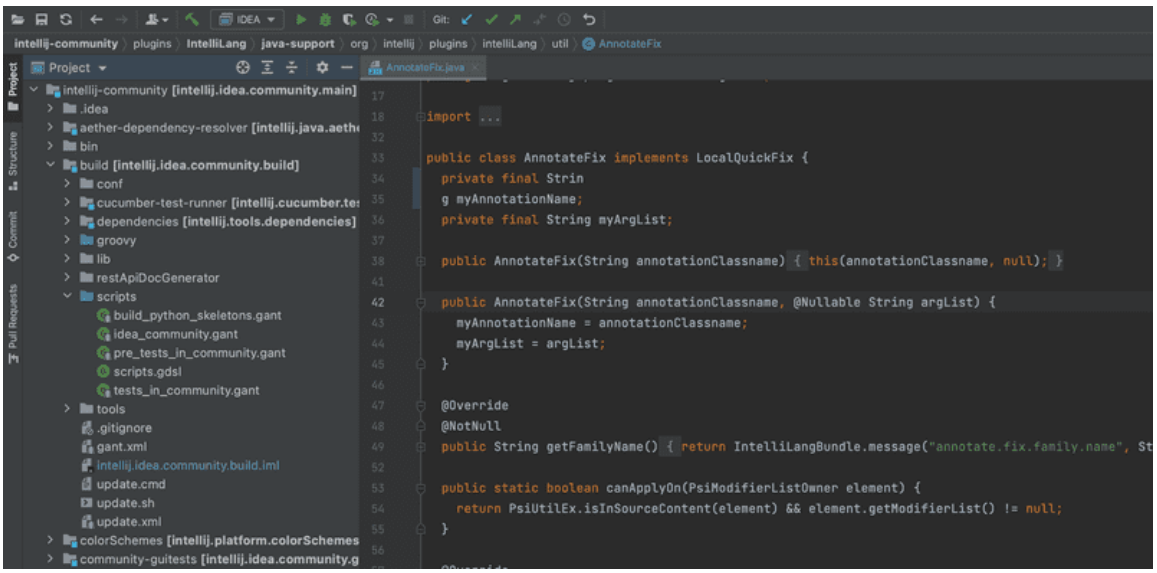


Рисунок 1.6 — Інтегроване середовище розробки Intelij Idea

Atom текстовий — редактор з відкритим вихідним кодом для Linux, macOS, Windows з підтримкою плагінів, написаних на JavaScript і вбудованих під керуванням Git.

Atom заснований на Electron (раніше відомий як Atom Shell) — фреймворку крос-платформної розробки з використанням Chromium та io.js. Редактор написаний на CoffeeScript та LESS.

Переваги Atom:

— оптимізованість;

— кросплатформенність;

— миттєве переключення файлів, в даному середовищі розробки працює система нечіткого пошуку, для більш ефективного пошуку файлів додатку;

— командний рядок, установка Атом додає дві команди командного рядка — atom та apm.

На рисунку 1.7 показано функціонал редактору коду Atom.

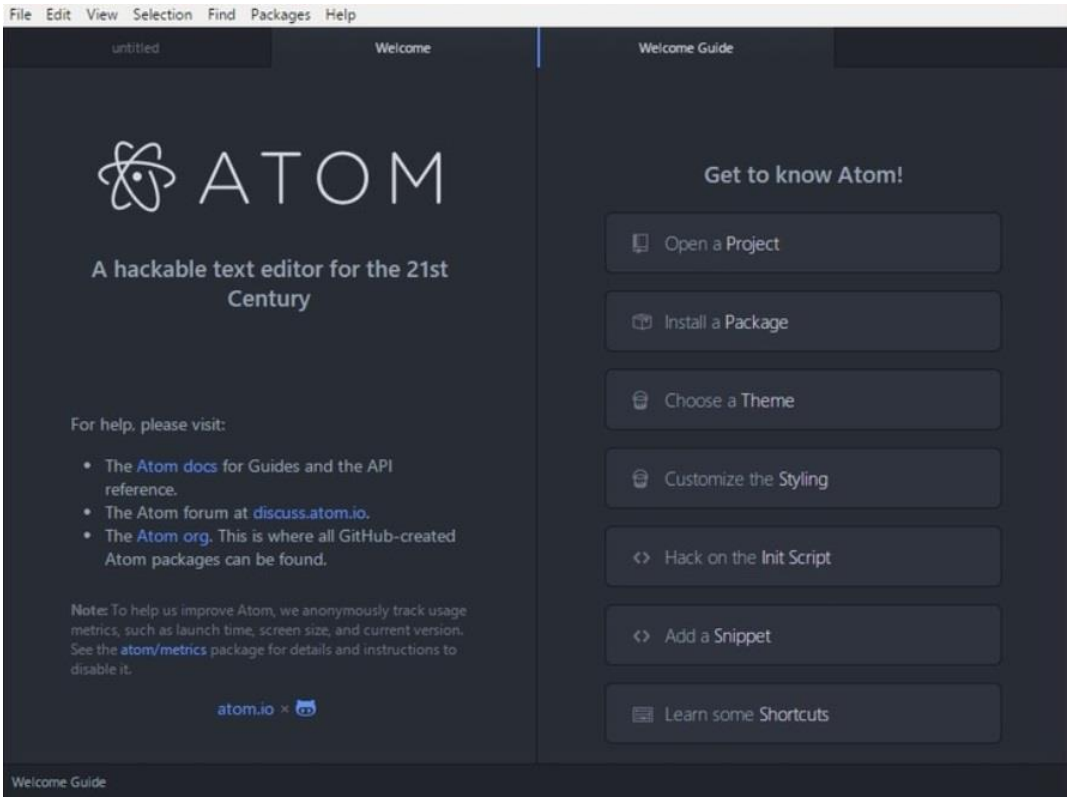


Рисунок 1.7 — Редактор коду Atom

Отже, для розробки веб-застосунку було вибрано інтегроване середовище розробки VScode, тому що він має велику кількість налаштувань, які можна використовувати щоб спростити написання коду.

1.4 Постановка задач роботи

У першому розділі було проведено аналіз методології розробки веб застосунків. Проведено порівняння таких навчальних додатків як: «coursera», «edx», «habr». Виявлено їх переваги та недоліки, та на основі отриманої

інформації визначені потреби для реалізації у веб застосунку для ретрансляції навчальних матеріалів.

Проаналізовані інтегровані середовища розробки для вирішення задачі і обгрунтовано вибір найкращого з даних інструментів.

На основі отриманої інформації та аналізу методів та інструментів розробки веб-додатків було поставлено такі задачі:

— визначення розвитку веб-застосунків з метою навчання;
— аналіз сучасних інструментів розробки веб-застосунків;
— вибір інструментів для розробки веб-застосунку;
— розробка структури веб-застосунку;
— реалізація відеоплеєру, для перегляду навчальних матеріалів;
— розробка структури проекту;
— аналіз методу розробки веб-застосунків SPA;
— створення інструкції користувача та тестування додатку.

2 АНАЛІЗ ТА ВИБІР ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ ДЛЯ РЕТРАНСЛЯЦІЇ НАВЧАЛЬНИХ МАТЕРІАЛІВ

2.1 Аналіз сучасних фреймворків для розробки веб-застосунків

2.1.1 Платформа для розробки веб-застосунків Angular

Angular — відкрита вільна платформа для розробки веб застосунків, написана мовою TypeScript, похідною мовою від JavaScript. Angular являє собою фреймворк від компанії Google для створення клієнтських додатків, який націлений на розробку SPA-рішень (Single Page Application). В цьому плані Angular є спадкоємцем другого фреймворка AngularJS. У цей же час Angular це не нова версія AngularJS, а принципово новий фреймворк.

Головною особливістю Angular є те, що він містить багато правил. Дотримання їх при написанні коду допомагає великим компаніям, в яких великі команди розробників працюють над різними частинами коду. Прикладами таких правил є те, що весь проект має бути структурований за модулями та компонентами, доступ до бек-енду здійснюється через служби, а не через компоненти тощо.

Розглянемо особливості даного фреймворку для потенційного використання у розробці веб застосунку такі як документація, підтримка компанією-розробники

Детальна документація. Даний фреймворк надає можливість скористатись своєю документацією для більш швидкого та доступного розуміння інструментів які він може надати.

Так як Angular розробила компанія Google, це само по собі є перевагою. Даний факт є певним еталоном якості для даного інструмента. Популярність Angular призвела до появи тисяч додаткових інструментів і компонентів, які можна використовувати в програмах. Це дозволяє отримати додатковий функціонал і підвищити продуктивність.

У другій версії Angular перейшов від архітектури Model-view-controller (MVC) до архітектури компонентів. Відповідно до неї програми поділяються на самостійні логічні та функціональні компоненти. Їх можна легко замінити

та від’єднати, а також повторно використовувати в інших частинах програми. Ця незалежність полегшує тестування веб-додатків і забезпечує безперебійну роботу всіх компонентів.

Компілятор Angular Ahead-of-Time (AOT) перетворює TypeScript і HTML у JavaScript під час процесу збірки. Це означає, що код компілюється до того, як браузер завантажить веб-додаток, тому він відображається набагато швидше. Компілятор AOT також набагато безпечніший, ніж компілятор Just-in-time (JIT).

Angular Universal — це серверний метод створення HTML-шаблонів, який, у свою чергу, має низку переваг. По-перше, це допомагає сканерам покращити рейтинг програми у пошукових системах. По-друге, це скорочує час завантаження сторінки та підвищує продуктивність мобільних пристроїв. Ці переваги приводять до збільшення кількості користувачів. Вона автоматизує весь процес розробки, максимально спрощуючи ініціалізацію, налаштування та розробку додатків. Angular CLI дозволяє створювати новий проект Angular, додавати до нього функції та запускати тести пристроїв та наскрізні тести за допомогою кількох простих команд. Це не лише покращує якість коду, а й значно спрощує розробку. Функція Ivy Renderer перекладає компоненти програми та шаблони на JavaScript, який можна відобразити у веб-браузері. Головною особливістю цього інструменту є техніка «струсити дерево». Під час візуалізації Ivy видаляє невикористаний код, що зменшує розмір пакету. В результаті веб-додатки завантажуються швидше.

Порівняльна характеристика фреймворків наведена на рисунку 2.1.

Однак, не зважаючи на переваги, якіх є досить багато у даному фреймворку йому також притаманні певні недоліки та складності при вивченні.

У Angular високі бар'єри для входу — його непросто вивчити через його складність.

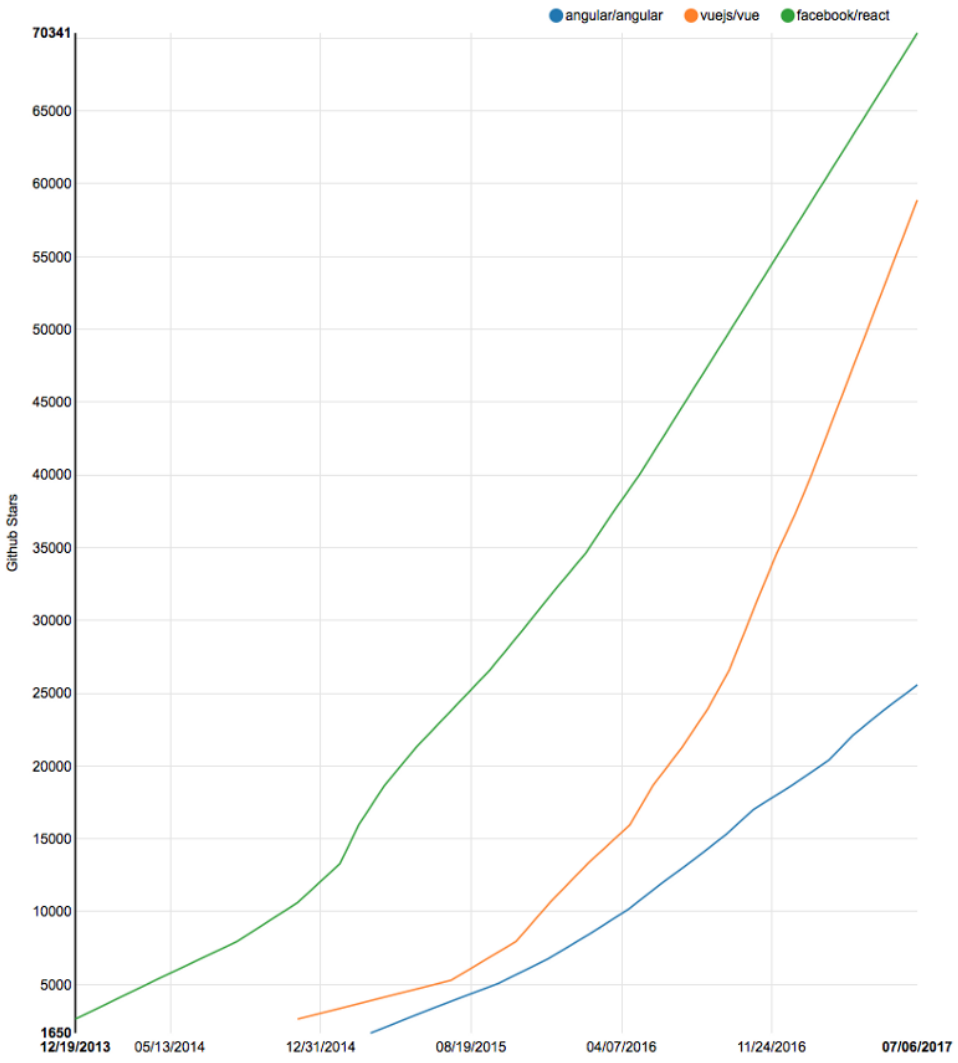


Рисунок 2.1 — Порівняльна характеристика фреймворків

Хоча компонентна архітектура робить код більш читабельним, все ж таки складно керувати залежностями компонентів. TypeScript додає додаткових труднощів, так як потрібно більше часу, щоб вивчити його [15].

У документації Angular є цілий розділ, в якому перераховані всі можливі способи вирішення проблеми міграції. Старі версії Angular повністю сумісні. Наприклад, можна легко перейти з п'ятої версії на четверту.

Також варто відзначити, що Angular пропонує рендеринг на стороні сервера, який прискорює завантаження початкової сторінки і, отже, покращує SEO, спрощуючи сканування динамічних сторінок. Швидке відображення сторінок значно покращує сприйняття веб-програм для наступного покоління, написаних в рамках Angular. Також, за допомогою даного фреймворку

можливе:

— спрощене тестування коду;

— створювення персоналізовані об'єктні моделі документу;

— моделювання даних;

— використання невеликих за обсягом моделей даних, що роблять код простим для читання і легким для тестування.

Angular був і залишається одним з найпопулярніших фреймворків на стороні клієнта. Можна з упевненістю сказати, що навчання Angular — це не просто оволодіння методами та властивостями фреймворку, а розуміння нової моделі програмування з унікальним підходом до вирішення проблем.

Знання та застосування Angular у своїй роботі — майже обов’язкова вимога для front-end розробника.

Детальна документація приведена на рисунку 2.2.

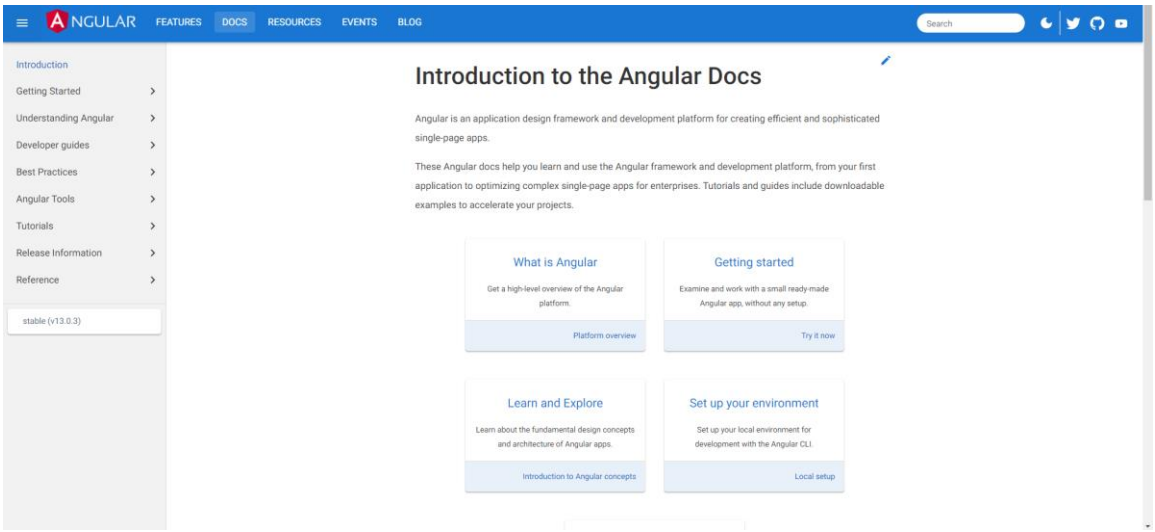


Рисунок 2.2 — Документація Angular

2.1.2 Аналіз фреймворку Ember.js

Ember.js — це фреймворк JavaScript для розробки клієнтської сторони веб-додатків. Розглянемо ключові концепції Ember.js.

Фреймворк Ember.js містить багато сучасних концепцій і технологій зі світу JavaScript.

Використання технології Babel для забезпечення підтримки ES2016.

Babel — це безкоштовний транскомпілятор JavaScript з відкритим вихідним кодом, який в основному використовується для перетворення коду ECMAScript 2015+ у зворотну сумісну версію JavaScript, яку можуть запускати старі движки JavaScript [14].

Підтримка засобів тестування Testem і QTest. Це відкриває можливості модульного, інтеграційного та прийомного тестування. Testem.js — це файл конфігурації для тестового механізму, який дозволяє вказати, які браузери використовуватимуться для тестування програми в різних середовищах. Тут зберігається логіка app/програми. QTest — це інструмент, який використовується для управління проектами, відстеження помилок та керування тестуванням. Він дотримується концепції централізованого управління тестуванням, що допомагає легко спілкуватися та допомагає швидко розробляти завдання для всієї команди QA та інших зацікавлених сторін.

Ключові особливості Qtest:

— підтримка білдера Broccoli.js;

— інтерактивне перезавантаження веб сторінок;

— підтримка шаблонізатора Handlebar;

— використання моделі розробки, при реалізації якої створюються в першу чергу URL-маршрути, що забезпечує повну підтримку глибоких посилань;

— наявність шару для роботи з даними, заснованого на JSON API.

Ember.js — це суто інтерфейсний фреймворк, він підтримує багато способів взаємодії з різними бекендами, але все, що стосується серверного коду, не входить до компетенції Ember.

Інтерфейс командного рядка Ember.js, ember-cli, надає доступ до багатьох функцій цього фреймворку. Ember-cli підтримує розробника на всіх етапах роботи. Це спрощує створення програми, розширює її функціональні можливості, тестує та запускає проект у режимі розробки.

В Ember.js використовуються такі терміни:

— маршрутизатор;

— шаблони;

— моделі;

— компоненти.

URL-адреса завантажує програму, вводячи URL-адресу в адресний рядок, і користувач натискає посилання в програмі. Ember використовує маршрутизатор, щоб зіставити URL-адресу з менеджером маршрутів. Маршрутизатор зіставляє існуючу URL-адресу з маршрутом, який потім використовується для завантаження даних, перегляду шаблонів та налаштування станів програми. Деталі процесу зображені на рисунку 2.3.

Шаблони є потужним інтерфейсом для кінцевих користувачів. Шаблон Ember надає вигляд інтерфейсу програми, який використовує синтаксис шаблону в Handlebars. Він створює інтерфейсну програму, яка виглядає як звичайний HTML. Він також підтримує регулярний вираз і динамічно оновлює вираз.

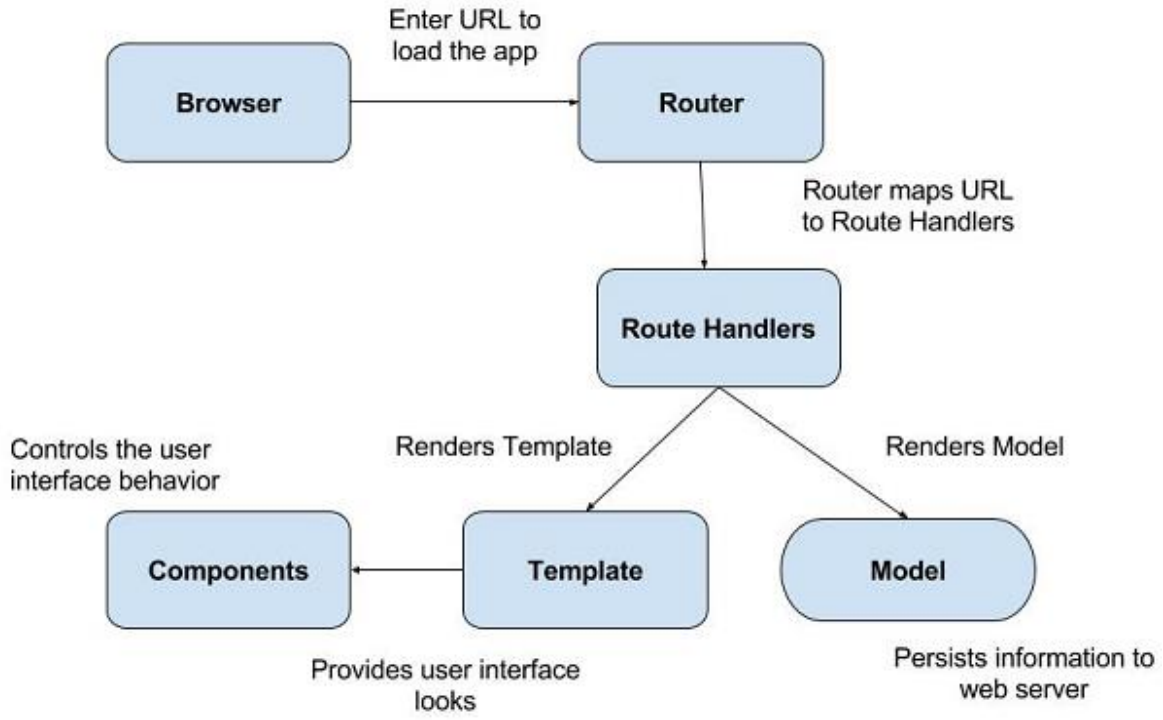


Рисунок 2.3 — Схема дії серверного обміну в Ember.js

Route Manager відтворює модель, яка зберігає інформацію на веб-сервері. Він маніпулює даними, що зберігаються в базі даних. Model — це простий клас, який розширює функціональність Ember Data. Ember Data — це бібліотека, тісно пов'язана з Ember.js для маніпулювання даними, що зберігаються в базі даних.

Документацію Ember зображено на рисунку 2.4.

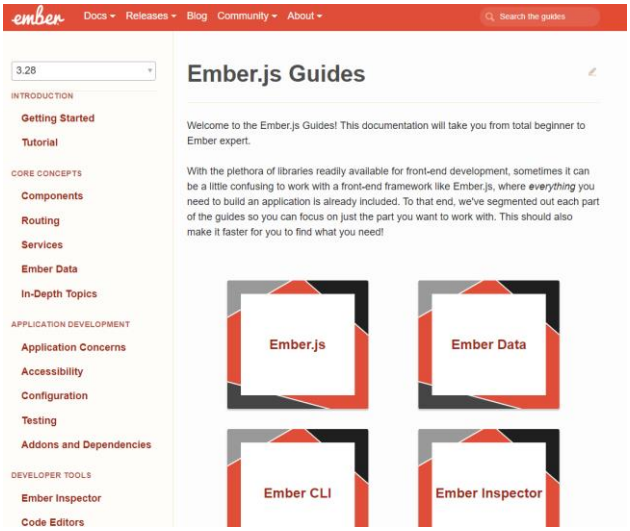


Рисунок 2.4 — Вигляд вікна документації Ember.js

2.1.3 Огляд платформи Vue.js

Vue.js — це прогресивна платформа JavaScript, яка дозволяє ефективно та легко створювати привабливі користувацькі інтерфейси.

Основний рівень розробки зосереджений на загальній побудові веб застосунку, проте він дає розробнику свободу імпортувати або інтегрувати наявні бібліотеки та інструменти. Метою є — створити неймовірно потужні та надійні односторонні (SPA) програми.

По-перше, Vue називають прогресивним фреймворком. Це означає, що він адаптується до потреб розробника.

Vue.js є, мабуть, найдоступнішим інтерфейсним фреймворком. У певній мірі Vue являється новою версією jQuery, тому що він легко входить до програми через тег сценарію і поступово отримує місце звідти. Це означає, що даний фреймворк зміг зайняти свою нішу за рахунок своєї

багатофункціональності у порівнянні з jQuery який домінував у мережі в останні роки, і він досі виконує свою роботу на великій кількості веб-сайтів.

Vue було створено шляхом вибору найкращих ідей для таких фреймворків, як Angular, React і Knockout, а також шляхом вибору найкращих варіантів, зроблених цими фреймворками.

Однією з найбільш помітних особливостей Vue є його дискретна реактивність. Моделі — це прості об'єкти JavaScript. Коли вони змінюються, перегляд даних також змінюється, що робить обробку станів програми простою та нескладною. Тим не менш, механізм реактивності має ряд функцій, знайомство з якими дозволить уникнути виникнення типових помилок. Детально даний механізм пояснюється методом реактивності даного фрейморвку (рисунок 2.5).

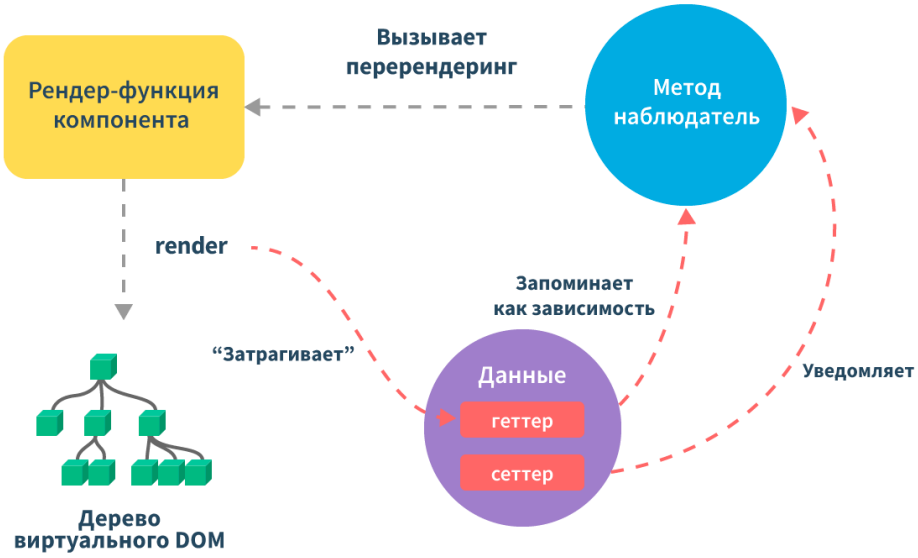


Рисунок 2.5 — Реалізація низькорівневої системи реактивності Vue

Коли простий об'єкт JavaScript передається екземпляру Vue як параметр даних, Vue обходить всі свої поля і перетворює їх на пари одержувачів/установників за допомогою Object.defineProperty. Ця функція з'явилася тільки в JavaScript у версії ES5, тому в попередніх версіях вона не емулювалася — тому Vue не підтримує IE8 і нижче.

Геттери та сеттери не помітні користувачеві, але вони є внутрішнім

механізмом, який дозволяє Vue відстежувати залежності та зміни даних. Однак при такому підході заголовки та набори, що відображаються в консолі браузера, не виглядають як звичайні об'єкти, тому для більшої наочності краще використовувати інструменти від розробника Vue-devtools [18].

Кожен екземпляр компонента додає пов'язаний екземпляр спостерігача, який зазначає всі поля, які були порушені, коли компонент був намальований як залежності. Пізніше, коли викликається установник поля, позначеного як залежний, цей регулятор інформує спостерігача, який, в свою чергу, ініціює перемальовування компонента.

Документацію Vue наведено на рисунку 2.6.

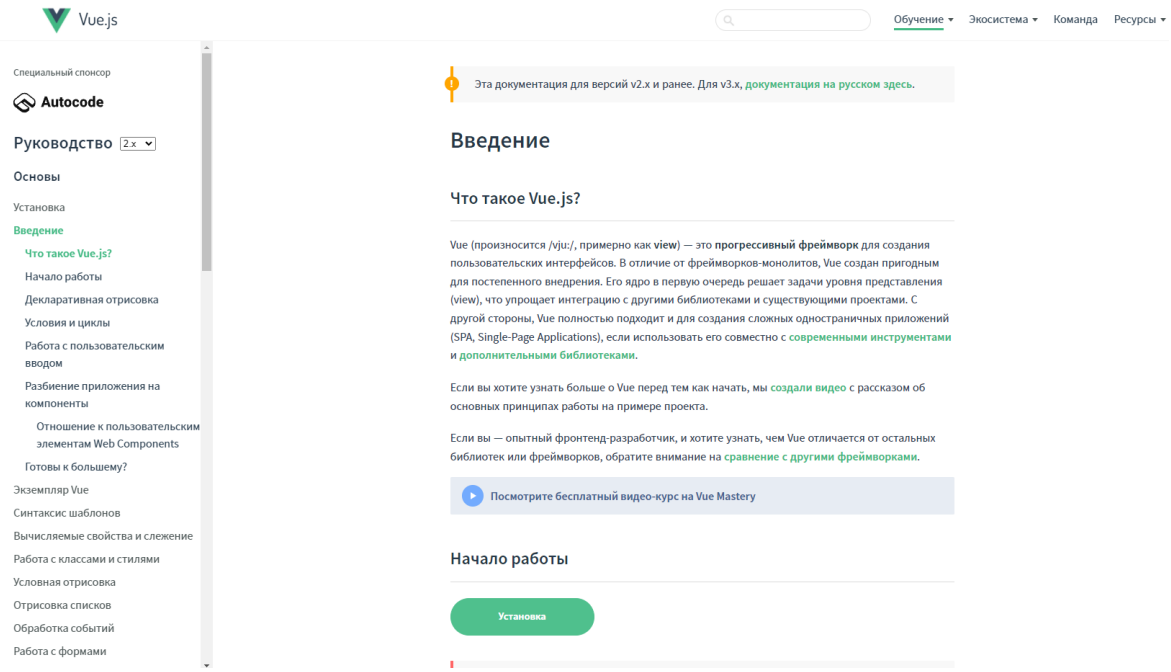


Рисунок 2.6 — Вигляд вікна документації Vue

2.2 Аналіз існуючих препроцесорів побудованих на CSS

У розробці веб застосунків дуже важливою частиною є стилізація компонентів сайту. У залежності від зовнішнього вигляду сторінки, буде визначатись популярність застосунку та зручність користування.

Стиль або CSS (каскадні таблиці стилів) відноситься до набору параметрів форматування, які застосовуються до елементів у документі для

зміни їх зовнішнього вигляду. Можливість роботи зі стилями вже давно включена в розроблені видавничі системи та текстові редактори, що дозволяє одним натисканням кнопки надати тексту заданий, заздалегідь визначений вигляд. Тепер він доступний для розробників сайтів, коли колір, розмір тексту та інші параметри зберігаються в певному місці і можуть бути легко «прикручені» до будь-якого тегу. Ще одна перевага стилів полягає в тому, що вони пропонують набагато більше варіантів форматування, ніж звичайний HTML.

2.2.1 Препроцесор стилізації SASS

Sass — це модуль, який є частиною Haml. Sass — це метамова на основі CSS, розроблена для підвищення рівня абстракції коду CSS та спрощення перекриття файлів таблиць стилів. Мова Sass має два синтаксиси: sass — відрізняється відсутністю фігурних дужок, вкладені елементи реалізовані з відступами.

Sass повністю сумісний з усіма версіями CSS. Розробники даного інструменту стилізація серйозно ставляться до сумісності між різними платформами для розробки та середовищами розробки, тому можливо легко використовувати будь-які доступні CSS бібліотеки.

Розглянемо особливості препроцесору Sass:

— мова препроцессингу із внутрішнім синтаксисом (своїм власним) CSS;

— забезпечення деяких функцій, за допомогою яких можна створювати стилі набагато ефективніше, також такі стилі найпростіше обслуговувати;

— це розширення CSS, тобто. він містить усі можливості CSS та є open source проектом на Ruby;

— він перекладає стилі документів у красивий структурований формат на відміну від плоского CSS, він використовує багаторазові методи, логічні вирази та кілька вбудованих функцій, таких як маніпулювання

кольорами, математичні списки та списки параметрів.

Для таких інструментів є важливою документація з конкретним та точним поясненням функцій та методів SASS, щоб зручно будувати веб застосунки. Документація наведена на рисунку 2.7.

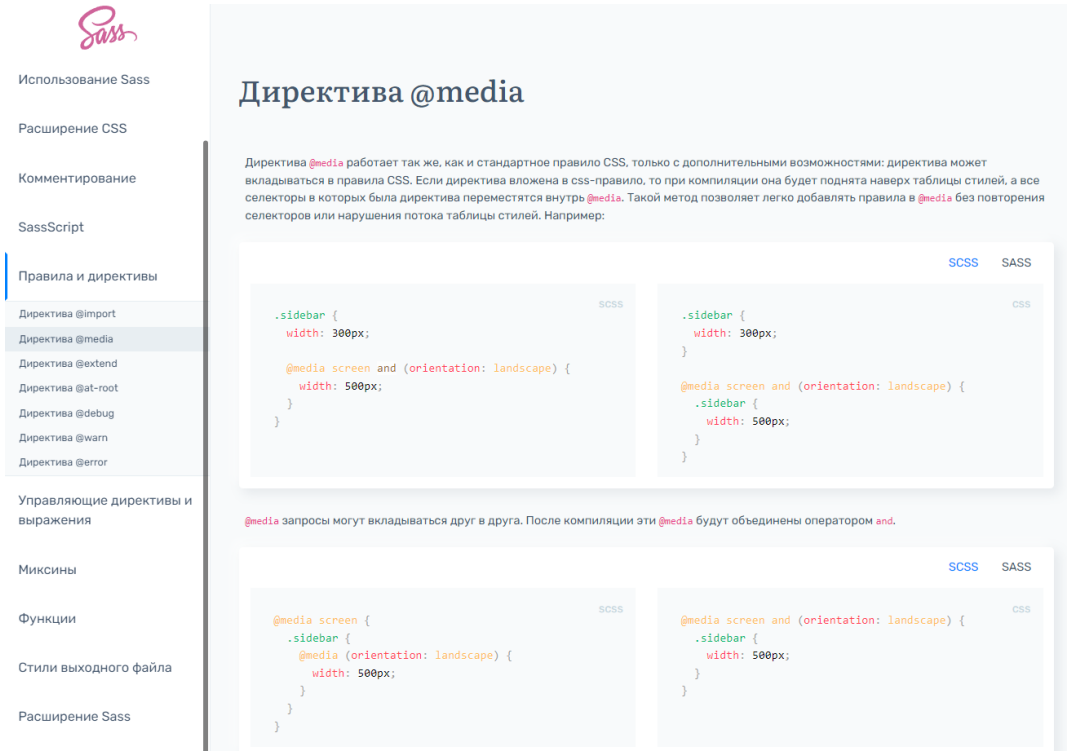


Рисунок 2.7 — Документація SASS

Розглянемо основні елементи SASS, а саме: вкладені правила, міксини, функції та селектори.

SASS має таку перевагу, як вкладені правила. Даний функціонал дозволяє зменшити чисельну кількість вихідного коду у символах, шляхом вкладення базових правил CSS в одне одне (рисунок 2.8).

Рисунок 2.8 — Вкладені правила CSS в SASS

Це допомагає уникнути повторення батьківських селекторів, і робить

складні макети CSS з великою кількістю вкладених селекторів набагато простіше.

Міксини оголошуються директивою @mixin. Слідом за ним має бути назва міксину і, за бажанням, його параметри, а також блок, що містить тіло міксину. Наприклад, можна визначити міксин великого тексту так, як показано на рисунку 2.9.

```
@mixin large-text {
  font: {
    family: Arial;
    size: 20px;
    weight: bold;
  }
  color: #ff0000;
}
```

Рисунок 2.9 — Приклад застосування mixin

Імена міксинів (і всіх інших ідентифікаторів в Sass) можуть містити дефіси та символи підкреслення як символи, що взаємозамінні. Наприклад, якщо ви визначаєте міксин add-column, можна підключати його як add_column і навпаки.

У SASS інсує свій спосіб оголошення функцій та використовувати їх у будь-якому значенні або контексті. Наприклад, як показано на рисунку 2.10.

SCSS

```
$grid-width: 40px;
$gutter-width: 10px;

@function grid-width($n) {
  @return $n * $grid-width + ($n - 1) * $gutter-width;
}

#sidebar {
  width: grid-width(5);
}
```

CSS

```
#sidebar {
  width: 240px;
}
```

Рисунок 2.11 — Приклад використання фукнцій в SASS

Як можна бачити, функції мають доступ до будь-яких глобальних змінних, а також приймають параметри, як і міксини (домішки). Функція може

містити кілька операторів, і потрібно викликати @return, щоб встановити значення функції, що повертається.

Так само, як і міксини, визначені в Sass функції можуть бути викликані іменованими аргументами.

Для того щоб вияснити як працюють селектори та директиви, розглянемо декілька на прикладі.

Sass розширює CSS @import rule1, дозволяючи вам імпортувати файли scss та sass. Усі імпортовані файли scss та sass можна об'єднати в один отриманий файл css. Крім того, будь-які змінні або міксини, оголошені в імпортованому файлі, можна використовувати в основному файлі.

Компілятор шукає інші файли sass у поточній папці та в каталозі файлів sass, якщо використовується разом із Rack, Rails або Merb. Додаткові каталоги пошуку можна вказати за допомогою параметра: load_paths або перемикача --load-path у командному рядку.

@import використовує ім'я файлу для імпорту. За замовчуванням @import шукає файли Sass, але існує кілька правил, за якими @import працює як правило CSS:

— якщо розширення файлу є .css;
— якщо ім'я файлу починається з http://;
— якщо ім'я файлу визначається через метод url();
— якщо правило @import включає в себе будь-які медіа запити.

Якщо жодне з правил вказаних вище не виконано, а розширення тотожніх файлів є .sass або .scss то дані файли будуть імпортовані.

Директива @media працює так само, як і стандартне правило CSS, тільки з додатковими функціями: директива може бути вкладена в правила CSS. Якщо директива вкладена в правило css, то під час компіляції вона буде піднята вгору таблиці стилів, а всі селектори, в які була розміщена директива, будуть переміщені всередину @media. Цей метод дозволяє легко додавати правила до @media, не повторюючи селектори або не порушуючи потоку таблиці стилів. Приклад розглянуто на рисунку 2.12.

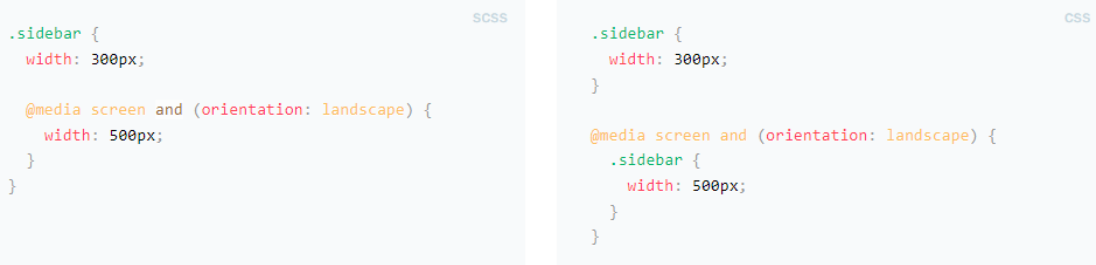


Рисунок 2.12 — Приклад застосування директиви @media

Отже, на основі проаналізованого інструментарію препроцессору SASS, розглянемо остаточні переваги та недоліки даного інструменту.

Перевагами препроцессору SASS є:

— можливість писати чистий CSS-код у мовному стилі програмування;

— можливість писати CSS швидко;

— розширеність CSS, за допомогою якого дизайнери та розробники можуть працювати набагато ефективніше і швидше;

— сумісність Sass з усіма версіями CSS, можна використовувати будь-які доступні CSS-бібліотеки;

— використання синтаксису вкладеності та такі корисні функції, як маніпуляція з кольорами, математичні та інші значення.

Недоліками препроцессору SASS є:

— необхідність вивчення можливостей препроцесора;

— втрата ефективності при використанні у великих проектах;

— втрата можливості використовувати вбудований у браузер інспектор елементів.

2.2.2 Препроцесор стилізації LESS

Less (що розшифровується як Leaner Style Sheets) — це зворотно-сумісний мовний додаток для CSS. Це офіційна документація для мови Less і Less.js, інструмента JavaScript, який перетворює ваші стилі Less у стилі CSS. Препроцесор LESS — це динамічна мова стилів, яку розробив Alexis Sellier. Він створений під впливом мови стилів Sass, і, у свою чергу, вплинув на новий

синтаксис «SCSS», в якому також використаний синтаксис, що є розширенням CSS [3].

У LESS можна створювати змінні та давати їм значення, наприклад кольори, шрифти, зображення. А потім використовувати їх у всьому файлі стилів. Це дуже зручно, тому що один і той же колір може використовуватися безліч разів і тепер, щоб поміняти його скрізь, вам достатньо буде змінити лише одну змінну [5].

Препроцесор CSS LESS розроблений на JavaScript, тому файли LESS можна використовувати на сайті, компілятор відправляє об'єднані файли у файли CSS у браузер, але насправді файл із розширенням .less буде на сервері. Зовнішній вигляд вікна документації наведено на рисунку 2.13.

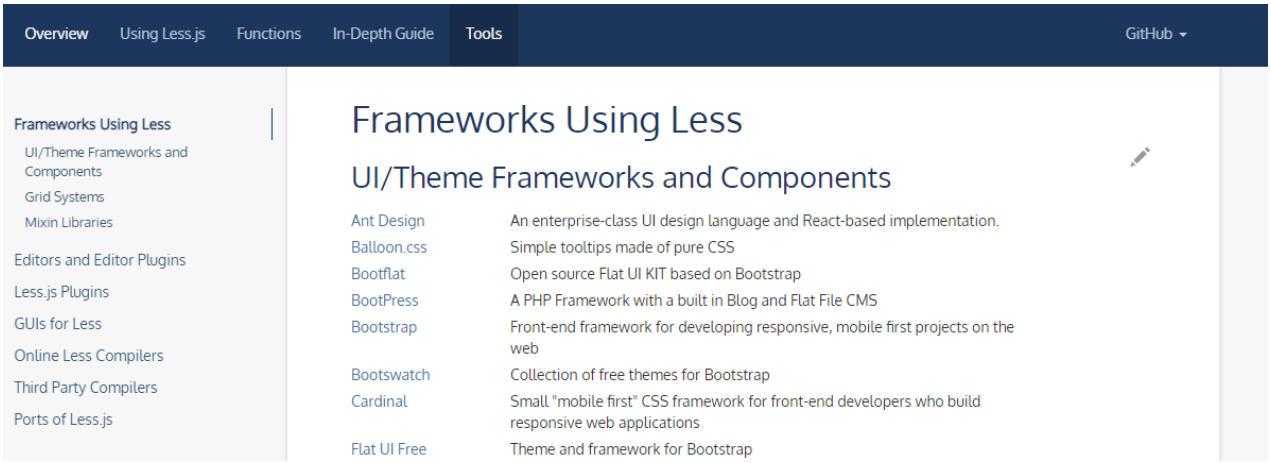


Рисунок 2.13 — Зовнішній вигляд вікна документації Less

Розглянемо деякі особливості даного інструменту стилізації такі як, оператор & представляє батьківські селектори вкладеного правила і найчастіше використовується при застосуванні модифікуючого класу або псевдокласу до наявного селектора.

Оператор «батьківські селектори» має різноманітне застосування. По суті, будь-коли вам знадобиться комбінувати селектори вкладених правил іншим способом, ніж типовим. Наприклад, ще одне типове використання & -

це створити повторювані імена класів як показано на рисунку 2.14.

```
.button {
  &-ok {
    background-image: url("ok.png");
  }
  &-cancel {
    background-image: url("cancel.png");
  }

  &-custom {
    background-image: url("custom.png");
  }
}
```

output:

```
.button-ok {
  background-image: url("ok.png");
}
.button-cancel {
  background-image: url("cancel.png");
}
.button-custom {
  background-image: url("custom.png");
}
```

Рисунок 2.14 — Приклад використання батьківського селектору в Less

Відокремлений набір правил — це група властивостей css, вкладених наборів правил, декларацій медіа чи чогось іншого, що зберігається у змінній. Ви можете включити його в набір правил або іншу структуру, і всі його властивості будуть скопійовані туди. Ви також можете використовувати його як аргумент mixin і передавати як будь-яку іншу змінну. Приклад наведено на рисунку 2.15.

```
// declare detached ruleset
@detached-ruleset: { background: red; }; // semi-colon is optional in 3.5.0+

// use detached ruleset
.top {
    @detached-ruleset();
}
```

compiles into:

```
.top {
  background: red;
}
```

Рисунок 2.16 — Приклад використання відокремленного набору правил в Less

Дужки після відокремленого виклику набору правил є обов’язковими

(крім випадків, коли за ними йде значення пошуку). Виклик @detached-ruleset; не спрацює.

Це корисно, коли ви хочете визначити міксин, який абстрагується від обгортання фрагмента коду в медіа-запиті або непідтримуваного імені класу браузера. Набори правил можна передати в mixin, щоб миксин міг обернути вміст.

Extend — це псевдоклас Less, який об'єднує селектор, на який він ставиться, з тими, які відповідають тому, на що він посилається. Приклад зображено на рисунку 2.17.

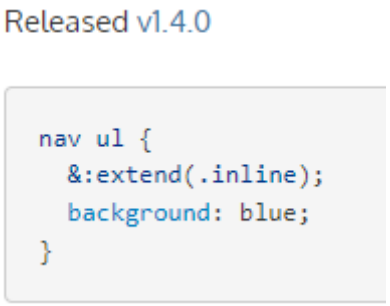


Рисунок 2.17 — Приклад псевдокласу Extend

У наведеному вище наборі правил селектор :extend застосовуватиме «розширювальний селектор» (nav ul) до класу .inline скрізь, де з’являється клас .inline. Блок оголошення буде зберігатися як є, але без посилання на розширення (оскільки extend не є css).

Перевагами інтепретатору Less є:

— можливість використовувати на стороні клієнта, для цього необхідно підключити лише один js файл;

— простота у порівнянні з аналогами;

— вбудовуваний JS;

— можливість вбудовувати класи і зручно заважати стилі;

— оптимізіація робочого процесу, велика гнучкість.

Недоліками інтерпретатору Less є:

— недостатній функціонал на стороні клієнту;

— відсутність if/for;

— необхідність у додаткових модулях на стороні серверу.

2.2.3 Препроцесор стилізації Stylus

Stylus — це препроцессор зі схожим синтаксисом до Sass, що створений для полегшення роботи зі стилізаціям веб-компонентів. Stylus орієнтований на мінімальний порог входу для розробника, різноманітну кількість інструментів для оптимізації процесу розробки та інше [4].

Препроцесор Stylus молодший, ніж його конкуренти, його заснували в 2010 році і написали на JavaScript. Йому віщують популярність через його зручність і простоту. Наприклад, у його синтаксисі відсутні «фігурні дужки» та інші розділові знаки. Хоча сама конструкція синтаксису скидається на «чистий» CSS. Для файлів використовує розширення ".styl".

Розглянемо особливості препроцессора Stylus.

Stylus підтримує інтерполяцію, використовуючи символи {} для оточування виразу, який потім стає частиною ідентифікатора. Наприклад, -webkit-{'border' + '-radius'} обчислюється як -webkit-border-radius.

Чудовим прикладом використання для цього є розширення властивостей за допомогою префіксів постачальника. Приклад наведено на рисунку 2.18.

```
vendor(prop, args)
  -webkit-{prop} args
  -moz-{prop} args
  {prop} args

border-radius()
  vendor('border-radius', arguments)

box-shadow()
  vendor('box-shadow', arguments)

button
  border-radius 1px 2px / 3px 4px
```

Рисунок 2.18 — Приклад використання інтерполяції в Stylus

Conditions забезпечують потік керування мовою, яка в іншому випадку

є статичною, забезпечуючи умовний імпорт, міксини, функції тощо.

Наведений нижче приклад умовно перевантажить властивість відступу, замінивши його на поле. Приклад наведено на рисунку 2.19

```
overload-padding = true

if overload-padding
  padding(y, x)
    margin y x

body
  padding 5px 10px
```

Another example:

```
box(x, y, margin = false)
  padding y x
  if margin
    margin y x

body
  box(5px, 10px, true)
```

Another `box()` helper:

```
box(x, y, margin-only = false)
  if margin-only
    margin y x
  else
    padding y x
```

Рисунок 2.19 — Приклад використання Condition if

У препроцессорі Stylus також наявний Condition unless. Дана умова є протилежністю умові if.

У наведеному нижче прикладі, якщо disable-padding-override значення undefined або false, padding буде замінено, замість цього відображатиметься поле. Але якщо це правда, відступ продовжить виводити відступ 5px 10px, як очікувалося. Приклад наведено на рисунку 2.20.

```
disable-padding-override = true

unless disable-padding-override is defined and disable-padding-override
  padding(x, y)
    margin y x

body
  padding 5px 10px
```

Рисунок 2.20 — Приклад використання Condition unless

Перевагами Stylus є:

— зручна підтримка та розширення проектів;

— автоматизація базових задач;

— ООП css;

— можливість створювати швидкі «теми».

Недоліками Stylus є:

— мала підтримка зі сторони користувачів даного препроцесору;

— недосконала система автоматичної генерації спрайтів;

— успадкування працює тільки з класами;

— велика кількість багів, що пов’язана здебільше з молодим віком препроцессору.

2.3 Бібліотеки

Бібліотека у програмуванні — збірник підпрограм або об'єктів, що використовуються для розробки програмного забезпечення (ПЗ).

У деяких мовах програмування те саме, що модуль, у деяких — кілька модулів. З погляду операційної системи (ОС) та прикладного ПЗ бібліотеки поділяються на динамічні та статичні.

2.3.1 Бібліотека JQuery

JQuery — це одна з найпопулярніших бібліотек JavaScript, що розширюються. Її використовують Microsoft, Netflix, Google та IBM. Найкраще застосовувати jQuery для програм або сайтів, які не керуються даними. Це з основною особливістю бібліотеки — відсутність підтримки інтерполяції змінних, маніпуляції в DOM. Вигляд вікна документації наведено на рисунку 2.21.

Одна з основних особливостей бібліотеки - можливість викликати її функцію в будь-якому місці веб-сторінки.

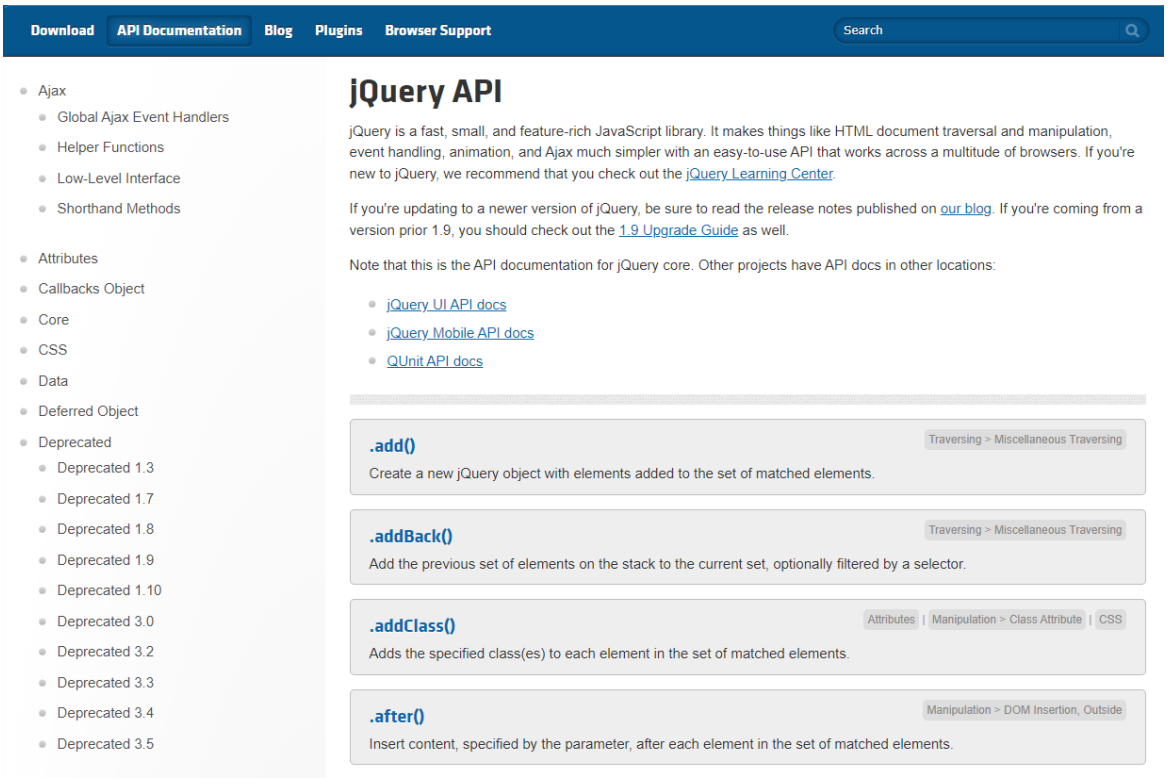


Рисунок 2.21 — . Вигляд вікна документації JQuery

Як тільки сторінка відкривається - завантажується jQuery. Потім відразу ж додається глобальна функція, що забезпечує стабільне виконання сценаріїв у будь-якому браузері. Простіше кажучи, користувачі відразу ж завантажують бібліотеку, тому проблем із переглядом контенту не виникає — не залежно від налаштувань різних браузерів, дана глобальна функція забезпечує стабільне виконання коду.

Основна мета jQuery — спростити використання JavaScript. Це виявляється у тому, що багато рядків коду JS можна замінити на метод, який, своєю чергою, можна викликати лише одним рядком коду [7]. Це значно скорочує час на розробку сторінок та окремих фічів у додатку (наприклад, анімації).

Перевагами бібліотеки JQuery є:

— оптимізація сучасного коду під застарілі версії браузерів;

— пришвидшення роботи з JavaScipt;

— використання сучасних плагінів;

— швидка відладка та тестування.

Недоліками JQuery є:

— відсутність рівню даних;

— не є оптимальною для використання у розробці складних користувальницьких інтерфейсів;

— великий об’єм файлу для імпорту;

— застарілість API.

2.3.2 Бібліотека React

React — JavaScript-бібліотека з відкритим вихідним кодом для розробки інтерфейсів користувача. React розробляється та підтримується Facebook, Instagram та спільнотою окремих розробників та корпорацій. React може використовуватися для розробки односторінкових та мобільних додатків.

React був створений Джорданом Валке, розробником програмного забезпечення із Facebook. На нього вплинув XHP - компонентний HTML-фреймворк для PHP [9]. Вперше React використовувався в стрічці новин Facebook в 2011 році і пізніше в стрічці Instagram в 2012 [10]. Вихідний код React відкрито у травні 2013 року на конференції «JSConf US».

React Native анонсований на конференції Facebook React.js Conf у лютому 2015 року, а вихідний код відкритий у березні 2015 року. Він дозволяє розробляти нативні Android-, iOS- та UWP-додатки з використанням React.

Розглянемо особливості даної бібліотеки, а саме як вона взаємодіє з кодом, видозмінює його та за допомогою яких інструментів дозволяє полегшувати та покращувати процес розробки. Документацію React наведено на рисунку 2.22.

Компоненти — найважливіша частина React.

React розроблено навколо концепції багаторазових компонентів. Веб застосунок базується на дереві великої кількості компонентів, які за допомогою інструментів React об’єднуються в більші компоненти.

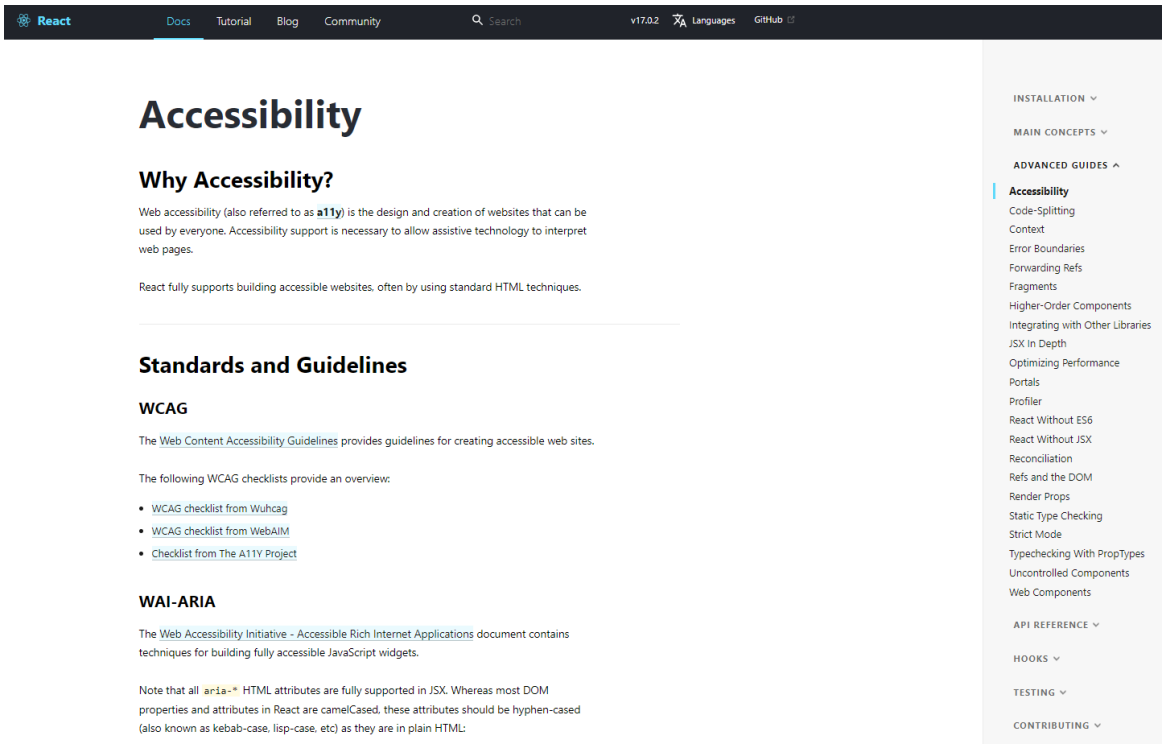


Рисунок 2.22 — Вигляд вікна документації бібліотеки React

Усі компоненти, маленькі чи великі, можуть використовуватися повторно, навіть у різних проектах.

Компонент React — у його найпростішій формі — це звичайна функція JavaScript, як зображено на рисунку 2.23.

```
class Greeting extends React.Component
 {
  constructor(props) {
    super(props);
    this.state = {
      name : "Human Friend",
      message : "You are welcome to ou
r World"
    }
  }...
```

Рисунок 2.23 — Приклад використання компоненту в React

Другий аргумент для ReactDOM.render — це елемент DOM, який React має намір захопити та контролювати. У jsComplete REPL використовується спеціальна змінна mountNode.

Назва компонента починається з великої літери. Це необхідно, тому що

ми будемо мати справу з поєднанням елементів HTML та елементів React. Назви в нижньому регістрі зарезервовані для HTML елементів. Насправді спробуйте назвати компонент React просто «button». ReactDOM проігнорує функцію та відобразить звичайну порожню кнопку HTML.

Кожен компонент отримує перелік атрибутів, як і HTML-елементи. У React цей список називається props. З функціональним компонентом ви можете назвати цей список як завгодно.

Функція render приймає щось, що виглядає як HTML, але поміщено в JavaScript. Це не JavaScript і HTML, і це навіть не React.js. Але при цьому це щось настільки популярне, що стало стандартним у програмах React. Це називається JSX, і це розширення JavaScript. Спробуйте повернути будь-який інший HTML всередині зазначеної вище функції та переконайтеся в їх підтримці (наприклад, поверніть елемент введення тексту input).

Розглянемо таку особливість бібліотеки React як JSX, що значить JavaScript and XML. Це деяка комбінація мови програмування веб-застосунків JavaScript та інструменту для стилізація XML. Перевагою такого підходу є те, що не є необхідністю використовувати окремі файли для стилів — можлива проста комбінація інструменту стилізація та мови програмування там, де це є необхідно. Приклад зображено на рисунку 2.24.

```
render() {
  const { name } = this.state;
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <h1 className="App-title">Welcome to React</h1>
      </header>
      <input type="text" onChange={this.handleChange} />
      <br />
      <b>{name}</b>

      <Home />
      |
    </div>
  );
}
```

Рисунок 2.24 — Приклад використання JSX

Функція createElement є основною функцією верхнього рівня React API. Подібно до того, як сам DOM містить функцію document.createElement для створення елемента, із вказаним ім'ям тега, функція React createElement є функцією вищого рівня, яка може робити те, що робить document.createElement, але при цьому ця функція також може бути використана для створення елемента для представлення компонента React [5].

На відміну від document.createElement, createElement React приймає необмежену кількість аргументів після другого аргументу для представлення дочірніх елементів створюваного елемента. Таким чином, createlement фактично створює дерево. Приклад зображено на рисунку 2.25.

InputForm не є компонентом React. Це лише елемент React. Ось чому ми використовували InputForm, а не <InputForm /> у викликі ReactDOM.render.

У функцію React.createElement було передано ще кілька аргументів після перших двох. Її перелік аргументів, починаючи з третього, містить перелік дочірніх елементів для створюваного елемента.

Рисунок 2.25 — Приклад використання createElement

Другий аргумент React.createElement може бути null або порожнім об'єктом, якщо елемент не потребує атрибутів або властивостей.

React API намагається бути якомога ближчим до DOM API, тому

використовується className замість класу для елемента введення.

Всередині секції JSX можливо використовувати будь-який вираз JavaScript у парі фігурних дужок.

Будь-який вираз JavaScript може бути поміщений усередину цих фігурних дужок. Це еквівалентно синтаксису інтерполяції ${} у шаблонах JavaScript.

При використанні JSX є обмеження. Всередині JSX можна використовувати лише вирази. Так, наприклад, не є можливим використовувати оператор if, але можете використати тернарний вираз.

Змінні JavaScript також є виразами, тому коли компонент отримує список властивостей (компонент RandomValue їх не використовує, властивості є необов'язковими), ви можете використовувати ці властивості всередині фігурних дужок.

Об'єкти JavaScript є також виразами. Іноді використовуються об'єкти JavaScript всередині фігурних дужок, що виглядає як подвійні фігурні дужки, але насправді це просто об'єкт усередині фігурних дужок.

React підтримує створення компонентів за допомогою синтаксису class JavaScript. Ось компонент Button, написаний за допомогою синтаксису класу, як зображено на рисунку 2.26.

```
3
4  class Button extends React.Component {
5    scream() {
6      alert('AAAAAAAAHHH!!!!!');
7    }
8
9    render() {
10     return <button onClick={this.scream}>AAAAAH!</bu
11   }
12 }
```

Рисунок 2.26 — Приклад створення компоненту Button p

Клас визначає єдину функцію render(), і ця функція повертає об'єкт

віртуального DOM. Щоразу, коли використовується компонент на основі класу Button, React буде створювати екземпляр об'єкта цього компонента на основі класу та використовувати цей об'єкт у дереві DOM.

Саме тому у прикладі використовується this.props.label всередині JSX у функції render, оскільки кожен компонент отримує спеціальний екземпляр (props), що містить всі значення, передані цьому компоненту при його створенні.

Оскільки існує екземпляр, пов'язаний з одним використанням компонента, можливо налаштувати цей екземпляр так як буде вигідно розробнику.

При обробці подій всередині елементів React необхідно врахувати такі особливості:

— всі атрибути елементів React (включно з подіями) називаються за допомогою camelCase, а не в нижньому регістрі;

— надавання фактичного посилання на JavaScript як обробник події, а не рядок.

React обгортає об'єкт події DOM у власний об'єкт (SyntheticEvent), щоб оптимізувати продуктивність обробки подій. Але всередині обробки подій ми все одно можемо отримати доступ до всіх методів, доступних в об'єкті події DOM [12]. React передає обернутий об'єкт події кожного виклику обробника. Наприклад, щоб запобігти відправленню форми за замовчуванням, можливо зробити згідно зразку, який зображено на рисунку 2.27.

Наступне стосується лише компонента класу (до тих компонентів, які розширюють React.Component). Функціональні компоненти мають трохи інше застосуання, розглянемо яке саме.

Спочатку визначається шаблон для React для створення елементів компонента.

Потім React створює екземпляр елемента та передає йому набір props, до яких ми можемо отримати доступ за допомогою this.props.

Рисунок 2.27 — Приклад роботи з обгорнутими подіями

Оскільки це все JavaScript, то буде викликаний метод конструктора. Це перший метод із тих, що ми називаємо методами життєвого циклу компонентів.

Потім React обробляє результат виклику функції render (отримує віртуальний вузол DOM).

Оскільки це перший раз, коли React виконує рендеринг елемента, React взаємодіятиме з браузером (від нашого імені, використовуючи DOM API), щоб відобразити в ньому елемент. Цей процес широко відомий як монтування.

Потім React викликає інший метод життєвого циклу, званий componentDidMount. Ми можемо використовувати цей метод, щоб, наприклад, зробити щось у DOM, який існує у браузері. До цього методу життєвого циклу DOM, з яким ми працювали, був віртуальним.

Деякі історії компонентів закінчуються тут. Компоненти демонтуються із DOM браузера з різних причин. Але перед тим, як це станеться, React викликає інший метод життєвого циклу, що складається завжди [5].

Стан будь-якого змонтованого елемента може змінюватись. Батько цього елемента може бути повторно відмальовано. Також змонтований елемент може отримати інший комплект props.

Поле класу state є спеціальним у будь-якому компоненті класу React. React контролює стан кожного компонента змін. Але для того, щоб React зробив це ефективно, ми повинні змінити поле стану за допомогою ще однієї функції API React, яку потрібно використати, це this.setState, як зображено на рисунку 2.28.

```
class Counter extends React.Component {
  state = { value: 0 };
  handleIncrementThreeTimes = () => {
    this.setState(increment);
    this.setState(increment);
    this.setState(increment);
  }
  render() {
    return (
      <div>
        <button onClick={this.handleIncrementThreeTimes}>+++</button>
        <h1>{this.state.value}</h1>
      </div>
    )
  }
}
```

Рисунок 2.28 — Приклад використання this.setState

У цьому класі є два поля. Спеціальне поле state ініціалізується об'єктом, який містить clickCounter, що починається з 0, і currentTimestamp, який починається з new Date().

Друга властивість — це функція handleClick, яку ми передали на подію onClick для елемента кнопки всередині методу render. Метод handleClick змінює стан екземпляра компонента, використовуючи setState. Зверніть увагу на це.

Інше місце, в якому змінюється стан, знаходиться всередині таймера, який ми запустили всередині методу життєвого циклу componentDidMount. Він цокає кожну секунду і виконує інший виклик this.setState.

У методі рендерингу використовувались дві властивості, які є у стані. Для цього немає спеціального API.

Отже, були розлянуті основні особливості бібліотеки React. Обгрунтовані приклади. Проаналізовано основний функціонал даної бібліотеки.

2.4 Інструменти виконання загальних задач

Gulp — це програма для виконання завдань, яка використовує Node.js як платформу. Gulp використовує лише код JavaScript і допомагає запускати зовнішні завдання та великомасштабні веб-програми. Він створює системні автоматизовані завдання, такі як мінімізація CSS та HTML, об'єднання бібліотечних файлів та компіляція файлів SASS. Ці завдання можна запустити за допомогою сценаріїв Shell або Bash у командному рядку [6].

Розглянемо особливності даного інструменту виконання загальних задач шляхом порівняння його з більш застарілим аналогом Grunt.

Grunt і Gulp є інструментами для збирання веб-додатків, створеними щоб автоматизувати процеси, що повторюються. Такі як: конкатенації файлів, стиснення картинок, таблиць стилів і файлів JavaScript.

З двох цих збирачів Gulp є новішим і, відповідно, у ньому розробники спробували позбавитися недоліків Grunt.

Grunt tasks працюють із файлами замість потоків даних: щоб запустився наступний task попередній має записати свої результати у файл. Gulp працює з потоком даних і звертається до файлової системи тільки по початку/завершення своїх task.

У Gulp немає потреби в плагіні watch, тому що можливість реагувати на зміни у файлах вже включено в ядро. Це те, що має бути в будь-якому збирачі відразу, а не виконаються за допомогою плагіна.

Конфігураційні файли Grunt нагадують об'єкт JSON, Gulp ж використовує найпростіший JavaScript-код, який у підсумку найбільш зрозумілий на високих проектах.

Тим не менш, давайте оцінимо переваги та недоліки цих збирачів для frontend-розробників з точки зору трьох параметрів: порога входження,

швидкості та продуктивності та наявності плагінів.

У Gulp використовується концепція потоків: можна провести аналогыю, що файли проходять крізь «трубу» (pipe) і в різних точках з ними виконуються ті чи інші дії, як зображено на рисунку 2.29.

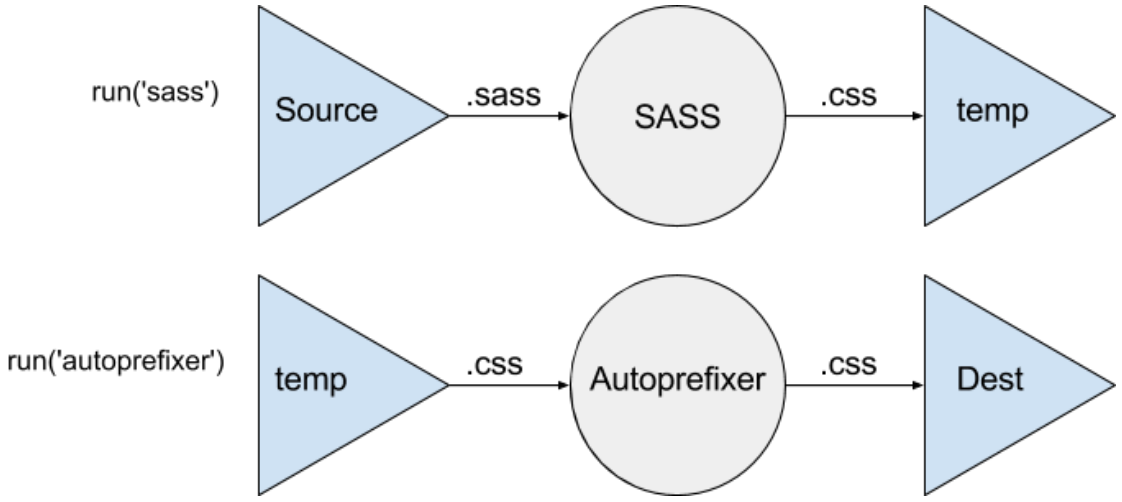


Рисунок 2.29 — Схема роботи потоків в Gulp

У такий спосіб можна, наприклад, вставити всі JavaScript-файли в pipe скриптів, який:

— поєднує всі файли в один;

— видаляє console та debugger;

— мінімізує код;

— зберігає результуючий код за вказаною адресою.

За кількістю плагінів Gulp помітно поступається Grunt але це цілком може бути пов'язане з тим, що Gulp молодший за свого колеги. Ось деякі плагіни для Grunt, які допомагають оптимізувати роботу:

— grunt-contrib-watch — запускає таски, коли змінюються файли, що відстежуються;

— grunt-contrib-jshint — виконує валідацію JavaScript-файлі;

— grunt-mocha — використовується для тестування за допомогою фреймворку Mocha;

— grunt-notify — автоматично відображає повідомлення при помилці

тягання;

— grunt-contrib-uglify — мінімізує файли за допомогою UglifyJS.

Всі вони є і у Gulp. Також для Gulp існує плагін gulp-grunt, який дозволяє запускати файли Grunt.

Враховуючи аналіз даних інструментів, доцільно провести певний висновок. При виборі між Grunt та Gulp необхідно врахувати потреби та ресурси що витрачаються при розробці. Grunt є легким у використанні, і вам не потрібно вникати особливо складної pipe-системи, а виконання простих завдань відбувається досить зрозуміло. Grunt - це зрілий продукт з безліччю плагінів, який використовує велику кількість розробників. При цьому у нього є недоліки на кшталт надто складних для прочитання конфігураційних файлів або уповільнення роботи за наявності великої кількості файлів через багаторазове повторення операції вводу/виводу.

У другому розділі МКР було проаналізовано інструменти для розробки веб-застосунку для перегляду навчальних матеріалів. Досліджено переваги та недоліки фреймворків Vue, Angular та Ember. Досліджені метамови для стилізації веб-застосунків Sass, Less, Stylus. Проведений аналіз інструментів збирання додатків Gulp і Grunt. Також, досліджені бібліотеки JQuery та React. Згідно виконаного аналізу інструментів, були обрані необхідні для розробки веб-застосунку. React — буде являти собою основну бібліотеку додатку, так як має великі переваги над описаними фреймворками Vue, Angular, Ember. Як метамова для стилізації було обрано Sass, за його інтегрованість, в порівнянні з іншими метамовами. Інструментом для збирання додатку було обрано Gulp, через великі переваги над аналогом Grunt.

3 РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ РЕТРАНСЛЯЦІЇ НАВЧАЛЬНИХ МАТЕРІАЛІВ

3.1 Використання методу SPA для розробки веб-застосунку

Одно сторінковий додаток — це веб-додаток або веб-сайт, який використовує єдиний HTML-документ як оболонку для всіх веб-сторінок і організує взаємодію з користувачем через HTML, CSS, JavaScript, що динамічно підвантажуються, зазвичай за допомогою AJAX [3].

SPA працює так: коли користувач відкриває сторінку, браузер завантажує одразу весь код програми. Але показує лише конкретний модуль — частина сайту, яка потрібна користувачеві. Коли користувач переходить в іншу частину програми, браузер бере вже завантажені дані та показує йому. І, якщо потрібно, динамічно підвантажує потрібний контент з сервера без оновлення сторінки.

З одного боку, такі програми працюють швидко та менше навантажують сервер. З іншого боку, вони потребують більшого завантаження на старті (рисунок 3.1).

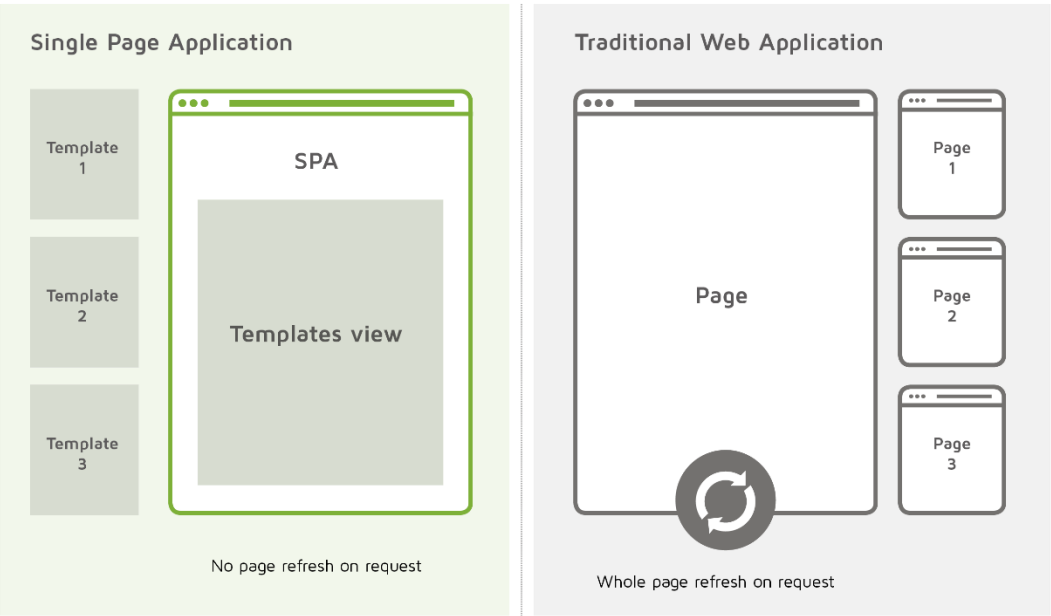


Рисунок 3.1 — Порівняння SPA з традиційним методом розробки веб-застосунків

З SPA для нових даних не потрібно повторно відображати всю сторінку, натомість через API запитуються лише нові фрагменти даних, які оновлюються на веб-сторінці [9].

Використання SPA має багато переваг, усі вони залежать від цілей додатку та призначення сайту. Деякі з цих переваг включають:

— плавний користувацький досвід;

— можливість використовувати існуючі API;

— офлайн-доступ до вмісту;

— SPA легко налагоджувати;

— просте та функціональне створення мобільних додатків;

— SPA може ефективно кешувати будь-яке локальне сховище;

— SPA розділяє інтерфейс користувача та дані.

Деякі з недоліків на даний момент використання односторінкової програми включають:

— швидкість початкового завантаження може бути нижчою, ніж на сайті, який використовує багатосторінковий додаток (MPA);

— негативно впливає на рейтинг у пошуковій системі, оскільки сторінки відображаються у браузері, а пошукові системи віддають перевагу сторінкам, створеним сервером;

— повернення до попередніх переглядів/сторінок (кнопка "Назад") може спрямовувати користувачів на фактичну попередню сторінку, яку пам'ятає браузер, а не на перегляд поточної сторінки, яку користувач міг бачити;

— витік пам’яті може з’явитися протягом тривалого періоду між перезавантаженням сторінки.

3.2 Програмна реалізація веб-застосунку

Веб-застосунок для ретрансляції, буде складатись з декількох частини, а саме з серверної та клієнтської. Серверна частина додатку буде відповідати за налаштування відеоплеєру та взаємодією з програмним забезпеченням для

трансляції [13]. Клієнтська частина — частина видима користувачеві і являє собою інтерфейс для ретрансляції, реалізований комбінуванням декількох модулів, таких як генератор ключів захисту, інформаційна панель та кнопка для початку роботи веб-застосунку.

Для роботи додатку необхідні лише комп’ютер та вихід в мережу Інтернет. Він не потребує потужної системи і його навантаження на існуючу систему мінімальне.

Система для ретрансляції являє собою веб-застосунок який напряму взаємодіє з ретрансляційним програмним забезпеченням OBS Studio або Xplit. Та при певному налаштування під’єднується до даного програмного забезпечення. Після налаштування користувач може запустити трансляцію та отримати візуальний результат на основній сторінці веб-застосунку.

Для створення веб-застосунку на React, команда create react-app створює базовий набір файлів для початку роботи. Це є перевагою бібліотеки React, тому що альтернативою було б використання webpack, а даний інструмент передбачає ручне налаштування всіх необхідних файлів та залежностей, що для недосвідченого розробника є великою проблемою.

Для початку, необхідно налаштувати базовий сервер вузлів із автентифікацією локальної стратегії за паспортом. Ми будемо використовувати MongoDB з Mongoose ODM для постійного зберігання. Ініціалізуємо новий проект, запустивши команду npm init і завантажимо залежності:

```
$ npm install axios bcrypt-nodejs body-parser bootstrap config
connect-ensure-login connect-flash cookie-parser ejs express
express-session mongoose passport passport-local request
session-file-store --save-dev
```

Пропонуємий веб-застосунок для ретрансляції навчальних матеріалів, складається з серверної та клієнтської частин. Серверна частина додатку відповідає за налаштування відеоплеєру та взаємодією з програмним забезпеченням для трансляції [12]. Клієнтська частина — частина видима

користувачеві і являє собою інтерфейс для ретрансляції, реалізований комбінуванням декількох модулів, таких як генератор ключів захисту, інформаційна панель та кнопка для початку роботи веб-застосунку.

Для роботи розроблюваного додатку необхідні лише комп’ютер та вихід в мережу Інтернет. Веб-застосунок не потребує потужної системи і його навантаження на існуючу систему мінімальне.

На основі визначеної частини збереження даних, побудуємо таблицю зі структурою проекту (таблиця 3.1).

Таблиця 3.1 — Структура файлів проекту

Файл/директорія	Призначення
/client	Клієнтська частина веб-застосунку на React
/server	Серверна частина веб-застосунку з використанням MongoDB
/public	Тека зі збереженими форматами стилів, шрифтів
package.json	Файл зі скриптами виконання файлів та підвантаженими залежностями
Webpack.config.js	Файл для збирання веб-застосунку

У теці /client розроблена візуальна частина веб-застосунку, реалізована за допомогою бібліотеки React. Користувач напряму взаємодіє з інтерфейсом, що складається з таких компонентів.

export default class Navbar extends React.Component {} — дана функція є основною у клієнтській частині веб-застосунку і слугує програмною частиною інтерфейсу. Користувач, при взаємодії з застосунком, буде користуватись інтерфейсом або панеллю навігації. За навігацією користувач напряму підключається до серверу застосунку, щоб в подальшому взаємодіяти з стороннім програмним забезпечненням.

Тека /server слугує для зберігання основної частини веб-застосунку, а

саме взаємодії з сервером. Відео, що в подальшому буде транслюватись, користується сервером та відправляє останньому запит, щоб отримати у відповідь певну адресу, яку додаток використовуватиме для підключення до програмного забезпечення.

Основним файлом у даній теці є app.js, в якому підсумована вся функціональність серверної частини і саме тут формується запит до бази даних.

Наведений нижче метод app.use відповідає за динамічне створення місця в базі даних, шляхом виконання методу MongoStore.create. Далі формується спеціальна адреса, яку отримає клієнтська частина. Ця адреса є захищеною та має термін життя 14 днів. Це термін дії ключа захисту, що формується в процесі налаштування взаємозв'язку веб-застосунку та програмного забезпечення для створення трансляцій.

```
app.use(session({
    store: MongoStore.create({
        mongoUrl: 'mongodb://127.0.0.1/nodeStream',
        ttl: 14 * 24 * 60 * 60 // = 14 days. Default
    }),
    secret: config.server.secret,
    maxAge : Date().now + (60 * 1000 * 30),
    resave : true,
    saveUninitialized : false,
}));
```

У теці /public збереженні шаблони стилів та шрифтів, що застосовуються по всьому додатку. Виділення таким чином стилів та шрифтів є дуже корисним під час розробки, код стає чистіший та легше редагується.

У файлі package.json зберігаються основні скрипти виконання файлів та залежності які використовуються під час розробки.

У файлі Webpack.config.js зберігаються загальні налаштування для збирання самого додатку. Зазвичай React сам модифікує даний файл та задає

базові налаштування для збирання. Проте, розробник може скористатись даним файлом, щоб змінити деякі параметри і створити унікальний файл для збирання.

Система для ретрансляції являє собою веб-застосунок який напряму взаємодіє з ретрансляційним програмним забезпеченням OBS Studio або Xplit. При відповідному налаштуванні є можливість під’єднатися до даного програмного забезпечення. Після налаштування користувач може запустити трансляцію та отримати візуальний результат на основній сторінці веб-застосунку.

Отже після ініціалізації застосунку, створимо та налаштуємо зв'язок з базою даних. Це необхідно для створення авторизації користувача, для більш безпечного використання додатку.

Реалізацію роботи авторизації описано у серверній частині таким кодом:

```
const passport = require('passport'),
   LocalStrategy = require('passport-local').Strategy,
   User = require('../database/Schema').User,
   shortid = require('shortid');
passport.serializeUser( (user, cb) => {
   cb(null, user);
passport.deserializeUser( (obj, cb) => {
   cb(null, obj);
passport.use('localRegister', new LocalStrategy({
      usernameField: 'email',
      passwordField: 'password',
      passReqToCallback: true
   (req, email, password, done) => {
      User.findOne({$or: [{email: email}, {username: req.body.username}]},
(err, user) => {
         if (err)
            return done(err);
```

```
            if (user) {
                if (user.email === email) {
                    req.flash('email', 'Email is already taken');
                }
                if (user.username === req.body.username) {
                    req.flash('username', 'Username is already taken');
                return done(null, false);
            } else {
                let user = new User();
                user.email = email;
                user.password = user.generateHash(password);
                user.username = req.body.username;
                user.stream_key = shortid.generate();
                user.save( (err) => {
                    if (err)
                        throw err;
                    return done(null, user);
    passport.use('localLogin', new LocalStrategy({
            usernameField: 'email',
            passwordField: 'password',
            passReqToCallback: true
        (req, email, password, done) => {
            User.findOne({'email': email}, (err, user) => {
                if (err)
                    return done(err);
                if (!user)
                    return done(null, false, req.flash('email', 'Email doesn\'t exist.'));
                if (!user.validPassword(password))
                    return done(null, false, req.flash('password', 'Oops! Wrong
password.'));
```

```
            return done(null, user);
module.exports = passport;
let mongoose = require('mongoose'),
    bcrypt   = require('bcryptjs'),
    shortid = require('shortid'),
    Schema = mongoose.Schema;
let UserSchema = new Schema({
    username: String,
    email : String,
    password: String,
    stream_key : String,
UserSchema.methods.generateHash = (password) => {
    return bcrypt.hashSync(password, bcrypt.genSaltSync(8));
UserSchema.methods.validPassword = function(password){
    return bcrypt.compareSync(password, this.password);
UserSchema.methods.generateStreamKey = () => {
    return shortid.generate();
module.exports = UserSchema;
```

Коли запускається сервер та починає роботу веб-застосунок, користувач побачить систему аутентифікації, де він повинен ввести свої особисті дані, якщо він зареєстрований у системі. У протилежному випадку, він зможе зареєструватись (рисунок 3.2).

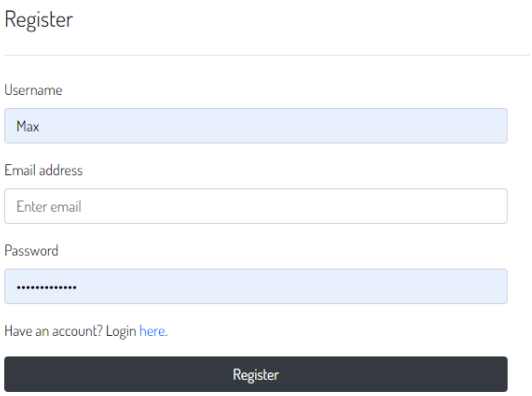


Рисунок 3.2 — Демонстрація системи аутентифікації

3.3 Розробка клієнтської частини веб-застосунку

Для відображення необхідних матеріалів, використаємо webpack і необхідні навантажувачі. Для початку завантажимо модулі:

```
$ npm install @babel/core @babel/preset-env @babel/preset-react
  babel-loader css-loader file-loader mini-css-extract-plugin
  node-sass sass-loader style-loader url-loader webpack webpack-cli
  react react-dom react-router-dom video.js jquery bootstrap history
  popper.js
```

Також додамо такі конфігурації до файлу webpack.config.js:

```
const path = require('path');
const MiniCssExtractPlugin = require("mini-css-extract-plugin");
const devMode = process.env.NODE_ENV !== 'production';
const webpack = require('webpack');
module.exports = {
   entry : './client/index.js',
   output : {
      filename : 'bundle.js',
      path : path.resolve(__dirname, 'public')
   },
   module : {
      rules : [
            test: /\.s?[ac]ss$/,
            use: [
               MiniCssExtractPlugin.loader,
               { loader: 'css-loader', options: { url: false, sourceMap: true } },
               { loader: 'sass-loader', options: { sourceMap: true } }
            test: /\.js$/,
            exclude: /node_modules/,
            use: "babel-loader"
         {
```

```
            test: /\.woff($|\?)|\.woff2($|\?)|\.ttf($|\?)|\.eot($|\?)|\.svg($|\?)/,
            loader: 'url-loader'
            test: /\.(png|jpg|gif)$/,
            use: [{
              loader: 'file-loader',
              options: {
                outputPath: '/',
    devtool: 'source-map',
    plugins: [
      new MiniCssExtractPlugin({
        filename: "style.css"
      new webpack.ProvidePlugin({
        $: 'jquery',
        jQuery: 'jquery'
    mode : devMode ? 'development' : 'production',
    watch : devMode,
    performance: {
      hints: process.env.NODE_ENV === 'production' ? "warning" : false
```

Ми використаємо react-router для маршрутизації та завантаження на інтерфейсі разом із video.js для відображення прямих трансляцій. Додамо каталог компонентів з файлом Root.js і наступний код:

```
import React from "react";
import {Router, Route} from 'react-router-dom';
import Navbar from './Navbar';
import LiveStreams from './LiveStreams';
import Settings from './Settings';
import VideoPlayer from './VideoPlayer';
const customHistory = require("history").createBrowserHistory();
export default class Root extends React.Component {
  constructor(props){
```

```
        super(props);
    render(){
        return (
            <Router history={customHistory} >
                <div>
                    <Navbar/>
                    <Route exact path="/" render={props => (
                        <LiveStreams  {...props} />
                    <Route exact path="/stream/:username" render={(props) => (
                        <VideoPlayer {...props}/>
                    <Route exact path="/settings" render={props => (
                        <Settings {...props} />
                </div>
            </Router>
}import React from "react";
import {Router, Route} from 'react-router-dom';
import Navbar from './Navbar';
import LiveStreams from './LiveStreams';
import Settings from './Settings';
import VideoPlayer from './VideoPlayer';
const customHistory = require("history").createBrowserHistory();
export default class Root extends React.Component {
    constructor(props){
        super(props);
    render(){
        return (
            <Router history={customHistory} >
                <div>
                    <Navbar/>
                    <Route exact path="/" render={props => (
```

```
            <LiveStreams  {...props} />
          )}/>
          <Route exact path="/stream/:username" render={(props) => (
            <VideoPlayer {...props}/>
          )}/>
          <Route exact path="/settings" render={props => (
            <Settings {...props} />
          )}/>
        </div>
      </Router>
```

Отже, була приведена розробка системи авторизації користувача через особисті дані. Візуальна частина була наведена та розроблена за допомогою React.

Після того, як користувач вводить свої особисті дані, він може використовувати веб-застосунок. А саме, веб-застосунок дозволяє згенерувати ключ захисту, а також можливість зберігати завчасно записані за допомогою ретрансляційних застосунків навчальні матеріали.

У третьому розділі магістерської кваліфікаційної роботи було приведено розробку базового функціоналу та базову структуру клієнтської та серверної частини веб-застосунку. Описано переваги та недоліки SPA. Розроблено структуру веб-застосунку. Розроблено клієнтську та серверну частини веб-застосунку.

4 ТЕСТУВАННЯ ВЕБ-ЗАСТОСУНКУ ТА РЕКОМЕНДАЦІЇ КОРИСТУВАЧУ

4.1 Опис технологій тестування веб застосунків

Веб-тестування — це ретельна перевірка сайту на наявність потенційних помилок. Це повне тестування веб-застосунків перед запуском. Веб-система повинна бути повністю перевірена від початку до кінця, перш ніж вона буде запущена для кінцевих користувачів. Також, під час тестування визначається зручність та привабливість сайту для користувача, швидкість отримання та доступність потрібної інформації, надійність та безпека використання. І, найголовніше, чи він допомагає досягти тих бізнес-цілей, які були поставлені на початку проекту. Загальна мета веб-тестування — знайти проблеми, також відомі як помилки, проблеми або дефекти, які можуть вплинути на веб-сайт або програму. Веб-тестування також можна використовувати для виявлення певних областей або аспектів веб-сайту, які можна покращити та отримати кращі результати, будь то більше запитів, більше продажів, більше постійних відвідувачів, та інше.

До того як розпочати тестування, дуже важливо визначити стратегію тестування. Насамперед це необхідно для організації процесу тестування в умовах, які є на даний момент або визначення потреби в залученні інших ресурсів. По-друге, не менш важливим є розуміння всіх учасників процесу тестування що відбувається, рівні очікування та виявлення проблем, які можуть виникнути в процесі тестування. Це є частини життєвого циклу тестування програмного забезбечення, який зображено на рисунку 4.1.

Стратегії тестування також описують, як ризики знижуються на рівні тестування, які типи тестування повинні виконуватись, та які критерії входу та виходу застосовуються. Вони створюються з урахуванням розробки проектної документації. Проектні документи описують функціональні можливості програмного забезпечення, яке буде включено до наступного випуску. Для кожного етапу розробки проекту має бути створено відповідну

стратегію тестування для тестування нових наборів функцій.

Продумана стратегія тестування стає основою розробки більш докладної документації, зокрема, тест-плана. Насамперед у побудові стратегії беруть участь менеджер проекту та спеціаліст контролю якості розробки, тобто QA (quality assurance) або QC (quality control) спеціалісти [11].

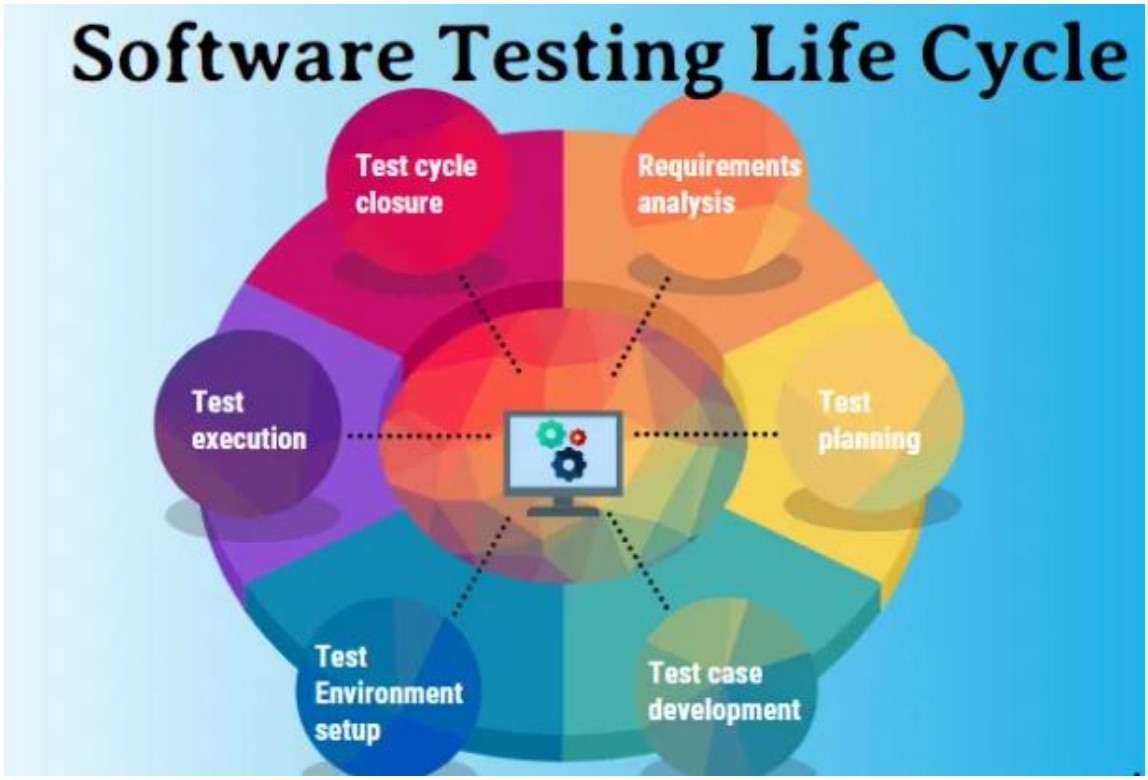


Рисунок 4.1 — Життєвий цикл тестування програмного забезпечення

Стратегії тестування також описують, як ризики знижуються на рівні тестування, які типи тестування повинні виконуватись, та які критерії входу та виходу застосовуються [8]. Вони створюються з урахуванням розробки проектної документації. Проектні документи описують функціональні можливості програмного забезпечення, яке буде включено до наступного випуску. Для кожного етапу розробки проекту має бути створено відповідну стратегію тестування для тестування нових наборів функцій.

Продумана стратегія тестування стає основою розробки більш докладної документації, зокрема, тест-плана. Насамперед у побудові стратегії беруть участь менеджер проекту та спеціаліст контролю якості розробки, тобто QA

(quality assurance) або QC (quality control) спеціалісти.

Якщо говорити про основні правила тестування веб-сайтів, то можна сказати, що йдеться про обов'язкове проходження декількох етапів тестування, які забезпечують повну працездатність програми, її безперебійну та безпечну роботу. Це важлива частина веб-розробки, яка забезпечує правильну роботу програми до його випуску.

На певних етапах тестування веб-сайтів використовуються такі види або типи тестування, що дозволяють повністю досягти поставлених цілей тестування. Цілі можуть бути різні, такі як перевірка важливих елементів: зручність використання безпеки, надійність, системи загалом, так і на перевірку цих елементів після змін, наприклад, повторне тестування [10].

Також важливою частиною процесу тестування програмного забезпечення є тестування продуктивності, що детально зображено на рисунку 2.31.

Рисунок 4.2 — Тестування продуктивності програмного забезпечення

4.2 Тестування веб-застосунку

Тестування починається з підключення директив для відкриття веб-застосунку: npm run watch — для автоматичного оновлення компонентів веб-застосунку, npm start — для запуску клієнту, npm run start — для запуску веб-серверу.

Далі користувач побачить сторінку авторизації (рисунок 4.3).

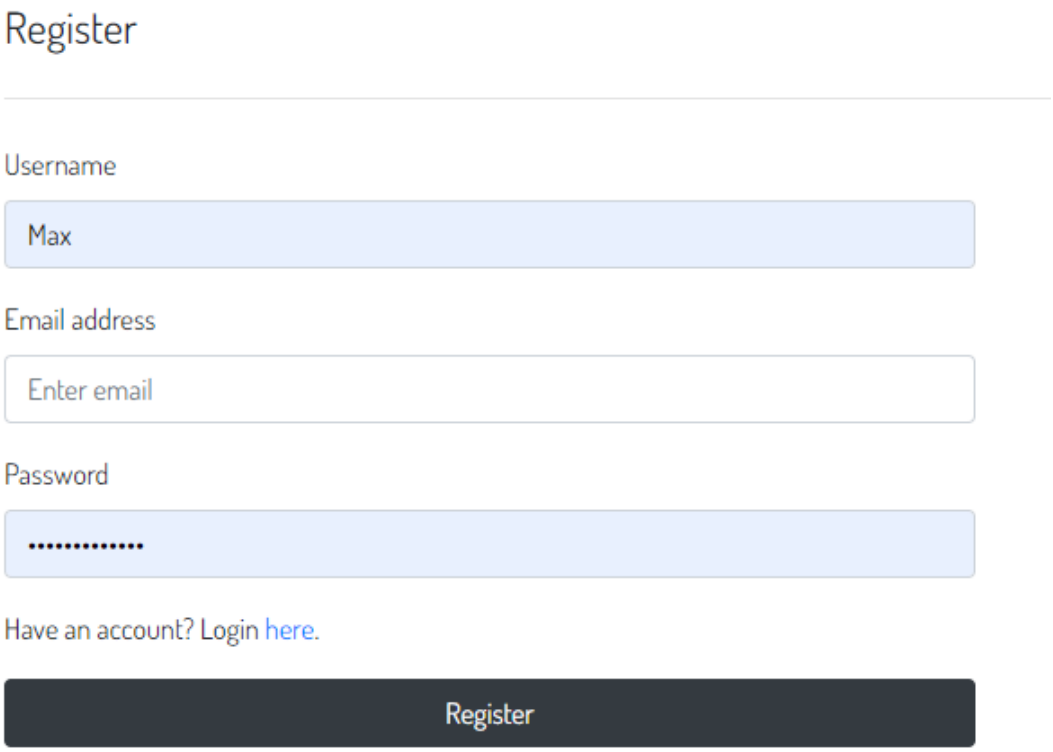


Рисунок 4.3 — Сторінка реєстрації користувача

Користувач вводить свої дані щоб зареєструватись, далі він повинен використати ці дані щоб скористатить функціоналом веб-застосунку (рисунок 4.4).

Після переходу за посиланням “Go Live”, користувач перейде на сторінку, де зможе згенерувати унікальний ретрансляційний ключ, щоб в подальшому використати його для застосування в програмному забезпеченні OBS (рисунок 4.6).

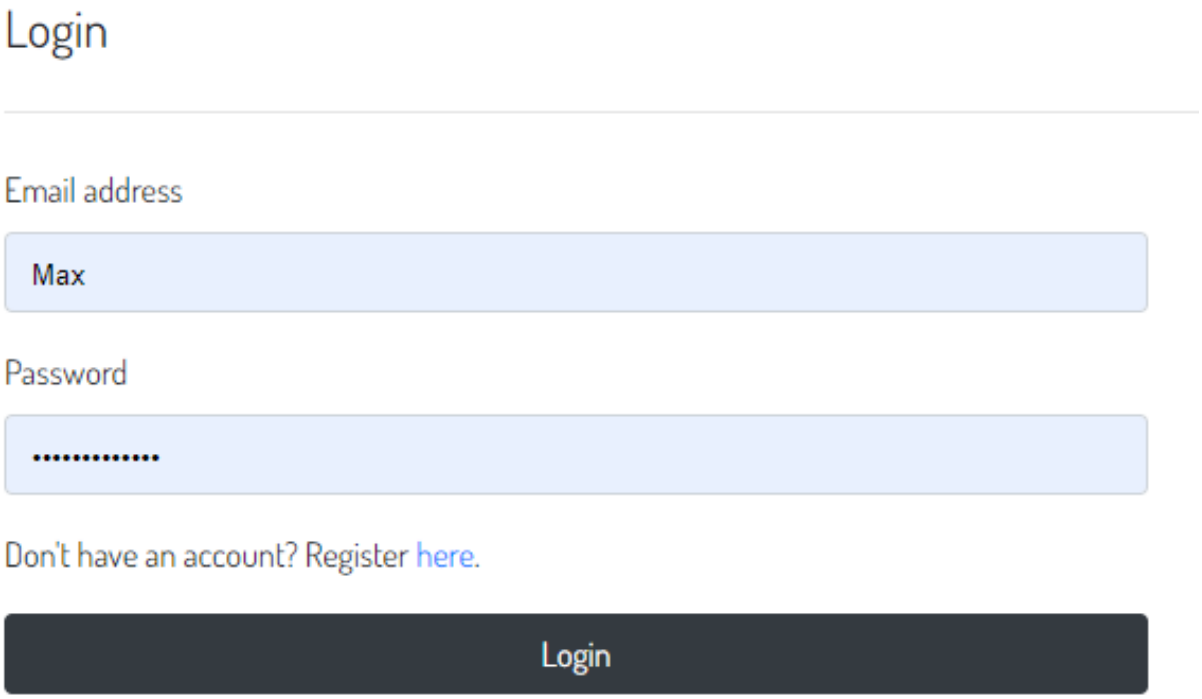


Рисунок 4.4 — Сторінка авторизації користувача

Користувач побачить майже порожню сторінку де в подальшому будуть ретранслюватись матеріали (рисунок 4.5).

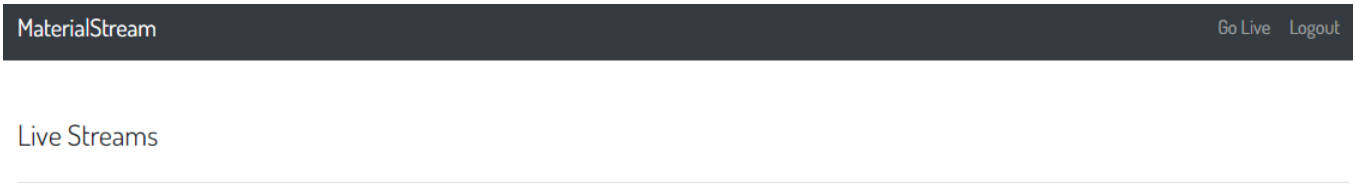


Рисунок 4.5 — Початкова сторінка з трансляцією

Далі користувач повинен оглянути текст, що розташований в нижній частині сторінки та перейти за посиланням на сайт OBS або Xplit.

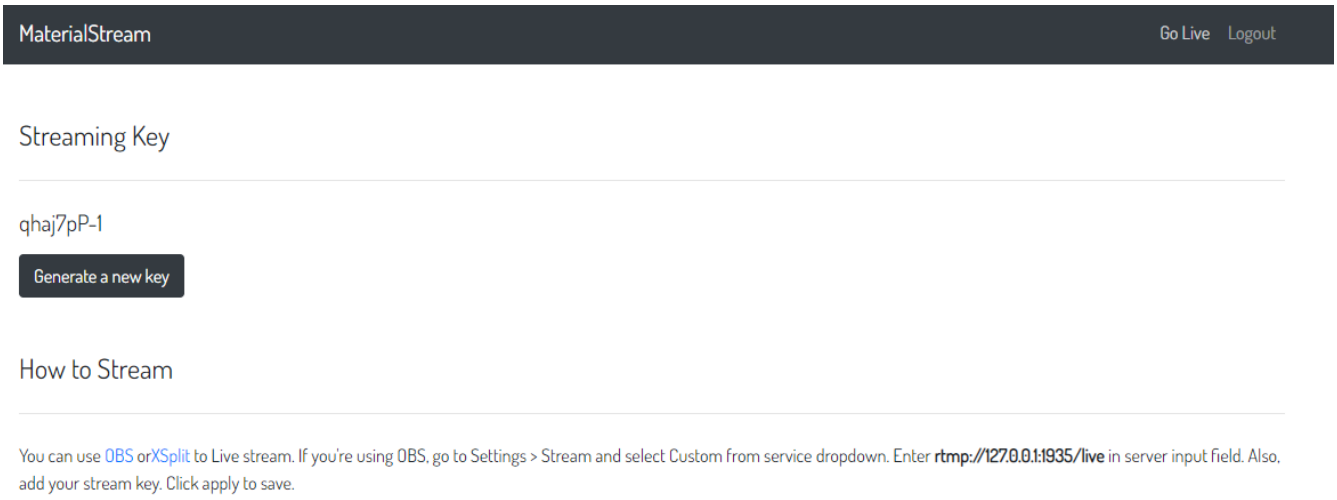


Рисунок 4.6 — Сторінка генерації ретрансляційного ключа

Там користувач зможе отримати необхідну інформацію для завантаження ретрансляційного додатку на систему яка буде йому зручна, щоб в подальшому використати цей застосунок для створення своєї трансляції, як показано на рисунку 4.7

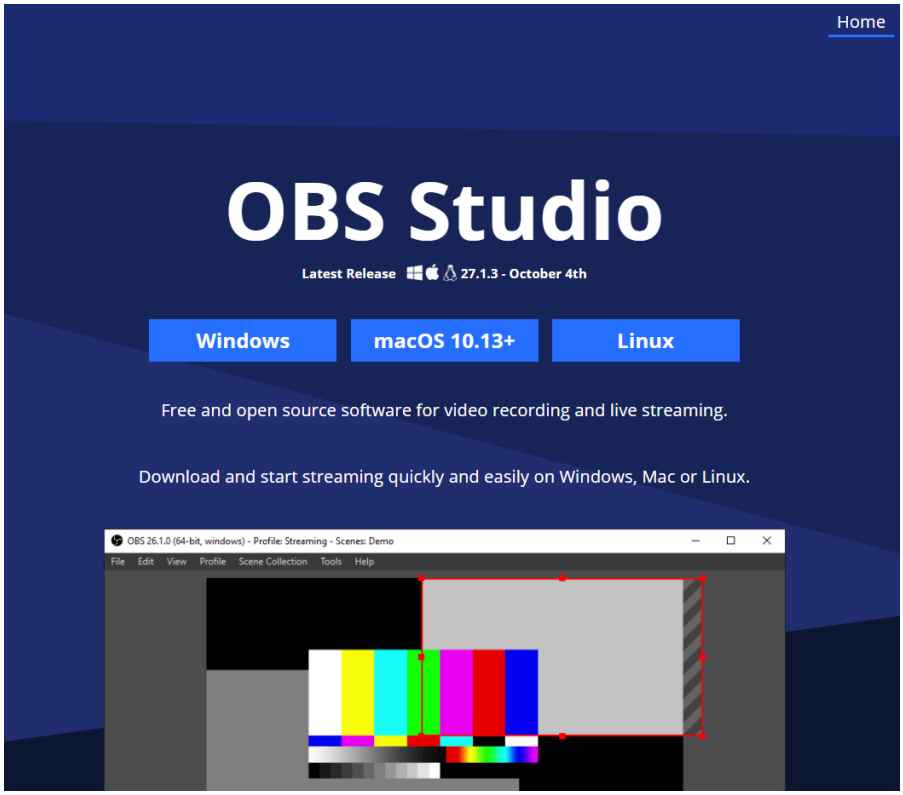


Рисунок 4.7 — Зовнішній вигляд сторінки застосунку для створення трансляцій OBS Studio

Далі необхідно буде скористатись ретрансляційним програмним забезпеченням OBS, для збереження матеріалів (рисунок 4.8).

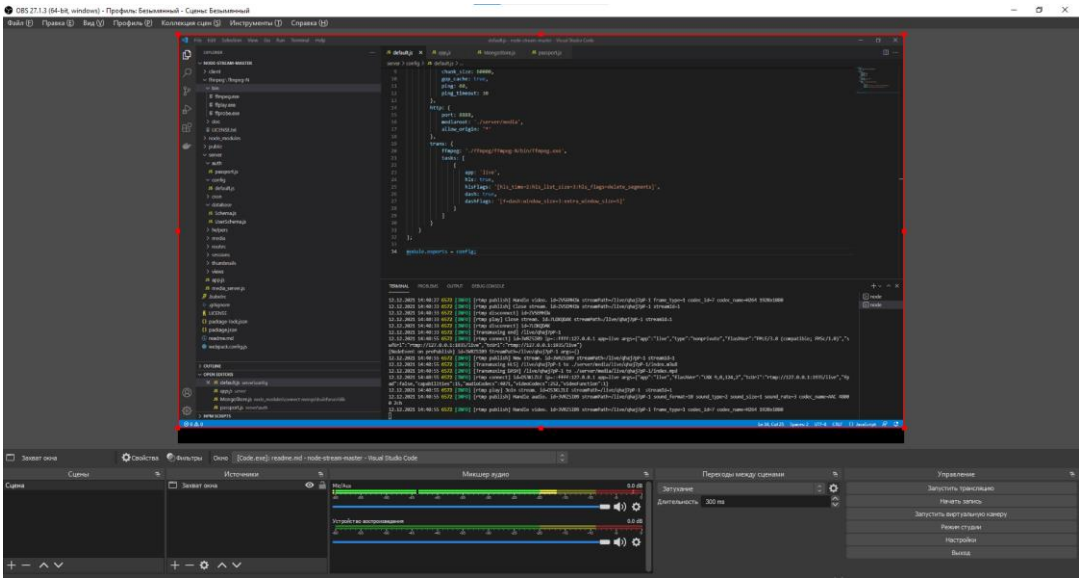


Рисунок 4.8 — Вигляд програмного забезпечення для ретрансляцій OBS

Після цього користувач повинен зайти в налаштування даного додатку та вибрати опцію “Трансляция” (рисунок 4.9).

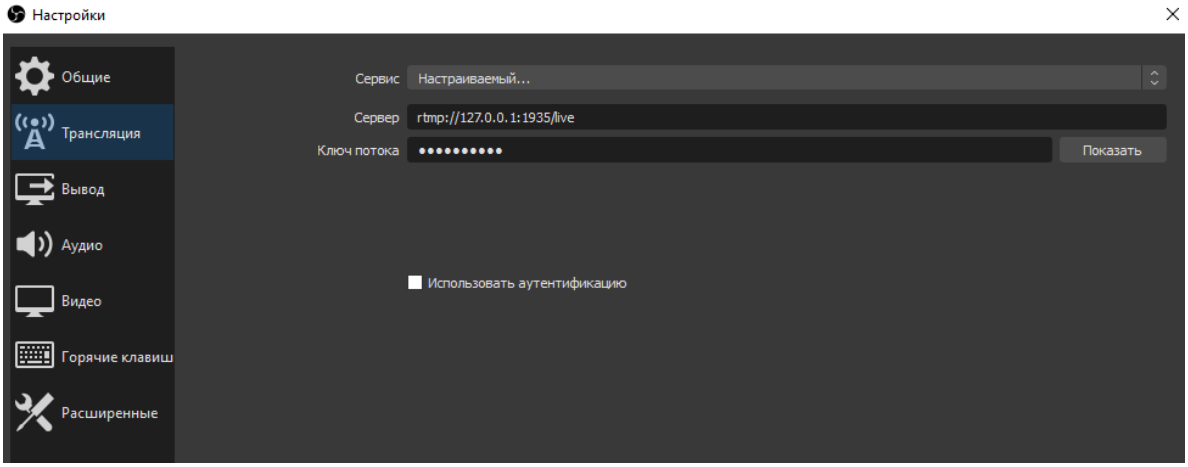


Рисунок 4.9 — Налаштування трансляції в додатку OBS Studio

Для того щоб сервер зміг записати трансляцію, потрібно натиснути кнопку “Начать трансляцию”. Після цього користувач зможе скористуватись веб-застосунком щоб отримати доступ до трансляції(рисунок 4.10).

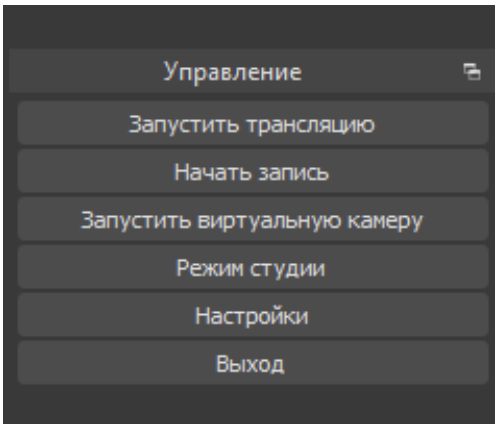


Рисунок 4.10 — Вигляд інтерфейсу додатку OBS Studio

Пересвідчившись, що додаток працює коректно, користувач відкриває веб-застосунок та натискає кнопку “Go Live” (рисунок 4.11).

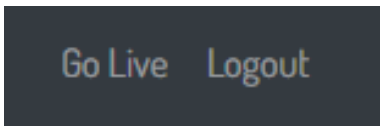


Рисунок 4.11 — Інтерфейс веб-застосунку для ретрансляції

Після натискання, користувач побачить вікно з своєю трансляцією в центрі екрану веб-застосунку, як можна бачити на рисунку 4.12.

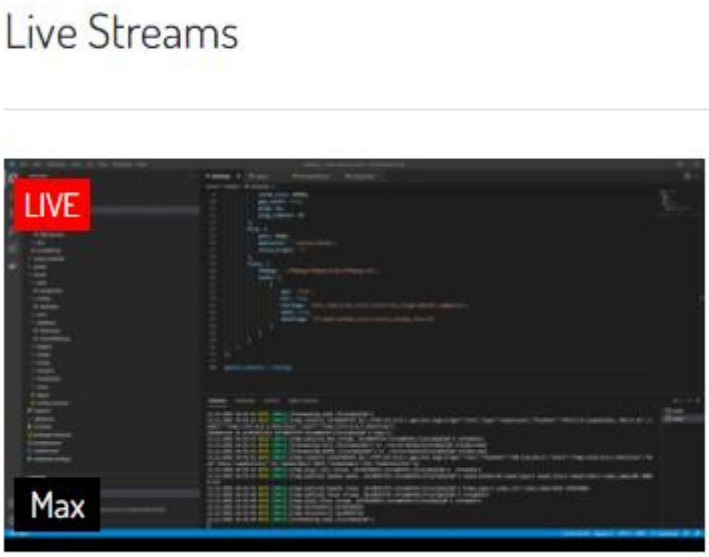


Рисунок 4.12 — Перед показ трансляції на головній сторінці

Коли користувач натисне на вікно що вспливає, він зможе побачити

відео плеєр з трансляцією (рисунок 4.13).

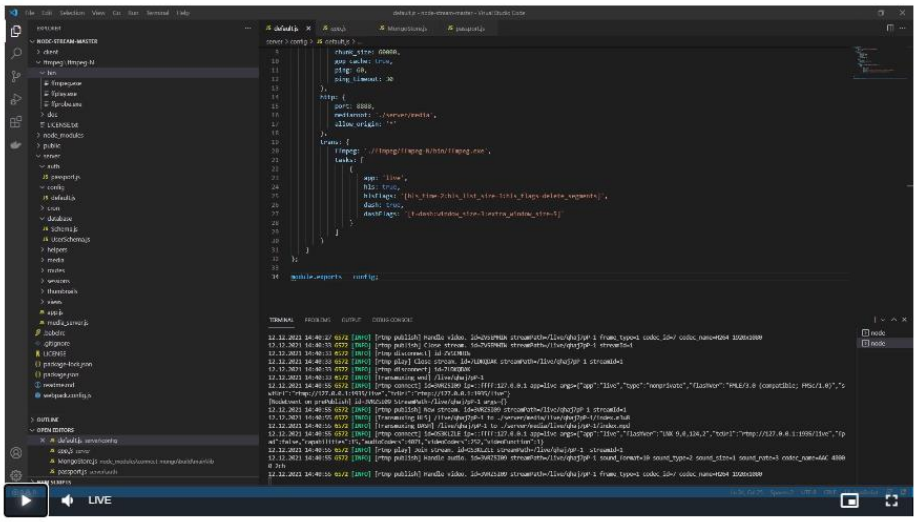


Рисунок 4.13 — Відео плеєр трансляції

У четвертому розділі магістерської кваліфікаційної роботи було проведено аналіз технологій тестування. Протестовано веб-застосунок і доведено що його функціонал працездатний та оптимальний для використання у технології розробки веб-застосунків. Проаналізовані підходи та життєвий цикл в тестуванні веб-застосунків. Проведено тестування веб-застосунку. Розроблено інструкцію та рекомендації користувачу.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота з розробки та дослідження на тему «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

— проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

— розраховано витрати на здійснення науково-технічної розробки;

— розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень» є оцінювання науково-технічного

рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [20].

Таблиця 5.1 — Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з	Необхідно наймати фахівців або витрачати	Необхідне незначне навчання фахівців та	Необхідне незначне навчання	Є фахівці з питань як з технічної, так і з

Закінчення таблиці 5.1

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 — Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	5	5
2. Ринкові переваги (наявність аналогів)	2	2	2
3. Ринкові переваги (ціна продукту)	4	3	3
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	0	0	0
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	4	4	4

Закінчення таблиці 5.2

8. Практична здійсненність (наявність фахівців)	4	5	4
9. Практична здійсненність (наявність фінансів)	3	4	4
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	3	4
12. Практична здійсненність (розробка документів)	4	3	3
Сума балів	40	39	39
Середньоарифметична сума балів $СБ_c$	39,3		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [19]:

Таблиця 5.3 — Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ ,розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень» становить 39,3 бала, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Одиничний параметричний індекс розраховуємо за формулою [19]:

$$q_i = \frac{P_i}{P_{баз\,i}}. \quad (5.1)$$

де q_i — одиничний параметричний індекс, розрахований за i-м параметром;

P_i — значення i-го параметра виробу;

$P_{базі}$ — аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 5.4.

Таблиця 5.4 — Основні техніко-економічні показники аналога та розробки, що проектується

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектоване ПЗ	Відношення параметрів нової розробки до аналога	Питома вага показника
Кількість базових ОС	шт.	2	4	2	0,25
Кількість каналів ретрансляції	шт.	8	16	2	0,12
Доступність інтерфейсу	бал	7	9	1,28	0,15
Кількість функцій навчального процесу	шт.	12	18	1,5	0,3
Захищеність від зовнішнього впливу (до 10 балів)	бал	7	7	1	0,18
Експлуатаційні витрати	грн	15	15	1	0,45
Ціна (тарифний план)	грн	150	110	0,73	0,55

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 — пристрій відповідає нормам і стандартам, 0 — не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою [19]:

$$I_{НП} = \prod_{i=1}^{n} q_i \ , \qquad (5.2)$$

де $I_{нп}$ — загальний показник конкурентоспроможності за нормативними параметрами;

q_i — одиничний (частинний) показник за i-м нормативним параметром;

n — кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра [19]:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \ , \qquad (5.3)$$

де $I_{тп}$ — груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i — одиничний параметричний показник i-го параметра;

α_i — вагомість i-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n — кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 4.4.

$$I_{тп} = 2 \cdot 0{,}25 + 2 \cdot 0{,}12 + 1{,}28 \cdot 0{,}15 + 1{,}5 \cdot 0{,}3 + 1 \cdot 0{,}18 = 1{,}56.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [19]:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \,, \tag{5.4}$$

де $I_{ЕП}$ — груповий параметричний індекс за економічними показниками;

q_i — економічний параметр i-го виду;

β_i — частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m — кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці .

$$I_{ЕП}=1 \cdot 0{,}45 + 0{,}73 \cdot 0{,}55 = 0{,}85.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [19]:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \tag{5.5}$$

$$K_{ІНТ} = 1 \cdot 1{,}56 / 0{,}85 = 1{,}83.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників.

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [19]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.6)$$

де k — кількість посад дослідників залучених до процесу досліджень;

M_{ni} — місячний посадовий оклад конкретного дослідника, грн;

t_i — число днів роботи конкретного дослідника, дн.;

T_p — середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 12250{,}00 \cdot 22 / 22 = 12250{,}00 \text{ грн.}$$

Проведені розрахунки наведені на таблиці 5.5.

Таблиця 5.5 — Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	12250,00	556,82	22	12250,00
Інженер-розробник інформаційних веб-застосунків	11800,00	536,36	14	7509,09

Закінчення таблиці 5.5

Інженер-розробник програмного забезпечення	11500,00	522,73	15	7840,91
Консультант (викладач ЗВО, д.т.н., професор)	12200,00	554,55	8	4436,36
Технік	7500,00	340,91	10	3409,09
Всього				35445,45

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \qquad (5.7)$$

де C_i — погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i — час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \qquad (5.8)$$

де M_M — розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=2379,00 грн;

K_i — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [19];

$К_с$ — мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ — середнє число робочих днів в місяці, приблизно $Т_р$ = 22 дн;

$t_{зм}$ — тривалість зміни, год.

$С_1$ = 2379,00 · 1,10 · 1,65 / (22 · 8) = 24,53 *грн.*
$З_{р1}$ = 24,53 · 7,90 = 193,81 *грн.*

Таблиця 5.6 — Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка обладнання для проектування веб-застосунка	7,90	2	1,10	24,53	193,81
Підготовка робочого місця розробника програмного забезпечення	4,10	2	1,10	24,53	100,59
Підготовка серверного обладнання	5,11	5	1,70	37,92	193,75
Інсталяція програмного забезпечення для моделювання та розробки веб-застосунка	3,50	4	1,50	33,45	117,09
Компіляція програмних блоків	3,65	3	1,35	30,11	109,90
Налагодження програмних блоків	4,78	6	2,00	44,61	213,22
Тестування застосунка	5,00	4	1,50	33,45	167,27
Всього					1095,63

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \qquad (5.9)$$

де $Н_{дод}$ — норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод}$ = (35445,45 + 1095,63) · 10 / 100% = 3654,11 грн.

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.10)$$

де $Н_{зп}$ — норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (35445,45 + 1095,63 + 3654,11) · 22 / 100% = 8842,94 грн.

5.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень».

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{в\,j}\,, \quad (5.11)$$

де $Н_j$ — норма витрат матеріалу j-го найменування, кг;

n — кількість видів матеріалів;

$Ц_j$ — вартість матеріалу j-го найменування, грн/кг;

$К_j$ — коефіцієнт транспортних витрат, ($К_j = 1,1 \ldots 1,15$);

$В_j$ — маса відходів j-го найменування, кг;

$Ц_{вj}$ — вартість відходів j-го найменування, грн/кг.

$М_1 = 2,00 \cdot 125,00 \cdot 1,1 - 0,000 \cdot 0,00 = 275,00$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.7 — Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір VERDE Ultra	125,00	2,00	-	-	275,00
Папір для записів VERDE Light A5	66,00	3,00	-	-	217,80
Органайзер офісний VERDE OFFICE	172,00	2,00	-	-	378,40
Канцелярське приладдя (набір офісного працівника)	202,00	4,00	-	-	888,80
Картридж для принтера Canon LBP6000	1010,00	1,00	-	-	1111,00
Диск оптичний NewCODE CD-R	12,25	3,00	-	-	40,43
Flesh-пам'ять Kingston 32 GB	230,00	1,00	-	-	253,00
Тека для паперів CARBONIX BOX-ZX	114,00	2,00	-	-	250,80
Всього					3415,23

5.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень» відсутні.

5.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (5.12)$$

де $Ц_i$ — ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ —кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i — коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k — кількість найменувань устаткування.

$B_{спец} = 24820,00 \cdot 1 \cdot 1,1 = 27302,00$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.8 — Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Кросплатформений сервер на основі Dell Inspiron 15 3857	1	24820,00	27302,00
Всього			27302,00

5.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.13)$$

де $Ц_{iпрг}$ — ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ — кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i — коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1{,}10…1{,}12$);

k — кількість найменувань програмних засобів.

$B_{прг} = 8520{,}00 \cdot 1 \cdot 1{,}1 = 9372{,}00$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.9 — Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Середовище розробки VisualStudioCode	1	8520,00	9372,00
Емулятор серверу для моделювання поведінки веб-застосунку	1	4635,00	5098,50
Всього			14470,50

5.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.14)$$

де $Ц_б$ — балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ — термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ — строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (27250{,}00 \cdot 1) / (2 \cdot 12) = 1135{,}42$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.10 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер розробника ПЗ	27250,00	2	1	1135,42
Робоче місце інженера-програміста	7620,00	5	1	127,00
Пристрої передачі даних	7320,00	4	1	152,50
Оргтехніка	8560,00	4	1	178,33
Приміщення лабораторії розробки ІС	316000,00	25	1	1053,33
ОС Windows 11	7852,00	2	1	327,17

Закінчення таблиці 5.10

Прикладний пакет Microsoft Office 2019	7210,00	2	1	300,42
Всього				3274,17

5.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.15)$$

де W_{yi} — встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ — вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 4,50 грн;

$K_{вni}$ — коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i — коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,45 \cdot 160,0 \cdot 4,50 \cdot 0,95 / 0,97 = 324,00$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.11 — Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер розробника ПЗ	0,45	160,0	324,00
Робоче місце інженера-програміста	0,25	160,0	180,00
Пристрої передачі даних	0,03	11,0	1,49
Оргтехніка	0,65	25,0	73,13
Кросплатформений сервер на основі Dell Inspiron 15 3857	0,50	150,0	337,50
Всього			916,11

5.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів тощо. Витрати за даною статтею відсутні.

5.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» відсутні.

5.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.16)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$I_в$ = (35445,45 + 1095,63) · 50 / 100% = 18270,54 грн.

5.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів;

витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.17)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$B_{нзв}$ = (35445,45 + 1095,63) · 100 / 100% = 36541,08 грн.

Витрати на проведення науково-дослідної роботи на тему «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (5.18)$$

$B_{заг}$= 35445,45 +1095,63 +3654,11 +8842,942427 +3415,23 +0,00 +27302,00 + 14470,50 + 3274,17 +916,11 +0,00 +0,00 +18270,54 +36541,08 = 153227,76 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.19)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 153227,76 / 0,9 = 170253,07 грн.

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN — збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	5000	7000	10000	7000

N — кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 60000 осіб;

$Ц_б$ — вартість послуги у році до впровадження інформаційної системи, приймемо 100,00 грн;

$\pm\Delta Ц_о$ — зміна вартості послуги від впровадження результатів, приймемо 11,20 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів

від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [19]:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.20)$$

де λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ=38%;

ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2021 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(11,20·60000,00+111,20·5000)·0,83·0,38·(1-0,18/100%)=317595,18 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(11,20·60000,00+111,20·12000)·0,83·0,38·(1-0,18/100%)=518911,22 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(11,20·60000,00+111,20·22000)·0,83·0,38·(1-0,18/100%)=806505,56 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4$ =(11,20·60000,00+111,20·29000)·0,83·0,38·(1-0,18/100%)=1007821,59 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.21)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,15;

t — період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 317595{,}18/(1+0{,}15)^1+518911{,}22/(1+0{,}15)^2+806505{,}56/(1+0{,}15)^3+$$
$$+1007821{,}59/(1+0{,}15)^4=276169{,}73+392371{,}43+530290{,}49+576225{,}27=1775056{,}92 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \qquad (5.22)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 170253,07 грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 170253{,}07 = 340506{,}14$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \qquad (5.23)$$

де $ПП$ — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 1775056,92 грн;

PV — теперішня вартість початкових інвестицій, 340506,14 грн.

$E_{абс} = ПП - PV$ = 1775056,92 - 340506,14 = 1434550,78 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.24)$$

де $E_{абс}$ — абсолютний економічний ефект вкладених інвестицій, 1434550,78 грн;

PV — теперішня вартість початкових інвестицій, 340506,14 грн;

$T_ж$ — життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+1434550,78/340506,14)^{1/4} = 0,51.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (5.25)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d =0,09;

f — показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,2.

$\tau_{мін}$ = 0,09+0,2 = 0,29 < 0,51 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у

впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{5.26}$$

де $E_в$ — внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,51 = 1,96 р.

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

У п’ятому розділі МКР згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень» становить 39,3 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,83 рази.

Також термін окупності становить 1,96 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень».

ВИСНОВКИ

У магістерській кваліфікаційній роботі було створено кросплатформений веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень.

У першому розділі було проведено аналіз методології розробки веб-застосунків. Проведено порівняння таких навчальних додатків як: «coursera», «edx», «habr». Виявлено їх переваги та недоліки, та на основі отриманої інформації визначені потреби для реалізації у веб застосунку для ретрансляції навчальних матеріалів.

У другому розділі МКР було проаналізовано інструменти для розробки веб-застосунку для перегляду навчальних матеріалів. Досліджено переваги та недоліки фреймворків Vue, Angular та Ember. Досліджені метамови для стилізації веб-застосунків Sass, Less, Stylus. Проведений аналіз інструментів збирання додатків Gulp і Grunt. Також, досліджені бібліотеки JQuery та React. Згідно виконаного аналізу інструментів, були обрані необхідні для розробки веб-застосунку. React — буде являти собою основну бібліотеку додатку, так як має великі переваги над описаними фреймворками Vue, Angular, Ember. Як метамова для стилізації було обрано Sass, за його інтегрованість, в порівнянні з іншими метамовами. Інструментом для збирання додатку було обрано Gulp, через великі переваги над аналогом Grunt.

У третьому розділі магістерської кваліфікаційної роботи було приведено розробку базового функціоналу та базову структуру клієнтської та серверної частини веб-застосунку. Описано переваги та недоліки SPA. Розроблено структуру веб-застосунку. Розроблено клієнтську та серверну частини веб-застосунку.

У четвертому розділі магістерської кваліфікаційної роботи було проведено аналіз технологій тестування. Протестовано веб-застосунок і доведено що його функціонал працездатний та оптимальний для використання у технології розробки веб-застосунків. Проаналізовані підходи та життєвий цикл в тестуванні веб-застосунків. Проведено тестування веб-

застосунку. Розроблено інструкцію та рекомендації користувачу.

В п’ятому розділі роботи проведено дослідження рівня комерційного потенціалу розробки, який становить 39,3 бала, що, свідчить про комерційну важливість проведення даних досліджень. За рівнем конкурентоспроможності науково-технічна розробка переважає існуючі аналоги приблизно в 1,83 рази. Термін окупності складає 1,96 р., що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Даний веб-застосунок є перспективним рішенням, завдяки іноваційному методу ретрансляції навчальних матеріалів шляхом взаємодії з ретрансляційним інструментом OBS. В подальшому даний веб-застосунок може бути покращений для трансляції матеріалів в навчальних закладах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Молодь в науці: дослідження, проблеми, перспективи. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2022
2. MIME types — HTTP | MDN. URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Basics_of_HTTP/MIME_types.
3. EdX | Free Online Courses by Harvard, MIT, & more | edX. URL: https://www.edx.org/.
4. Coursera | Онлайн курси та сертифікати. URL: https://ru.coursera.org/.
5. Всі публікації підряд / Хабр. URL: https://habr.com/ru/all/.
6. Хаген Граф. Створення сайтів за допомогою Joomla 1.5. Діалектика-Вільямс, 2009. 204 с.
7. Яков Файн, Антон Моісеєв. Angular і TypeScript. Manning, 2018. 464 с.
8. Тіленс Т. М. React в дії. Manning, 2019. 368 с.
9. Алекс Кіріакидіс, Костас Міниатис. The Majesty of Vue.js. Kindle, 2016. 240 с.
10. Дін Хум, Джейсон Грігсбі. Progressive web apps. Manning, 2018. 303 с.
11. Gulp або Grunt. URL: https://www.google.com/search?q=gulp+grunt&oq=gulp+gr&aqs=chrome.2.69i57j0i512l6j0i22i30l3.3973j1j7&sourceid=chrome&ie=UTF-8
12. Джералд Д.Е. Тестування програмного забезпечення. FTP, 2006. 280.
13. Тревіс Мейнард. Getting started with Gulp. Packt, 2015. 132 с.
14. Розробка Web-додатків для бізнесу [Електронний ресурс] — Режим доступу до ресурсу: https://tqm.com.ua/ua/rozrobka-veb-dodatkiv-servisiv-web-application-development/rozrobka-web-dodatkiv-dlia-bizniesu.
15. Тузовський А. Ф. Проектування і розробка web-додатків / А. Ф. Тузовський. — М.: Юрайт, 2019. — 218 с.
16. Монолитная vs Микросервисная архитектура [Електронний ресурс]. — 2019. — Режим доступу до ресурсу: https://proglib.io/p/monolitnaya-vs-mikroservisnaya-arhitektura-2019-09-16.

17. Flanagan D. JavaScript - The definitive guide / David Flanagan., 2011. — 1100 с. — (6).
18. Годун В. М. Інформаційні системи і технології в статистиці / В. М. Годун, Н. С. Орленко, М. А. Сендзюк. — К.: КНЕУ, 2003. — 267 с.
19. WebAssembly: что и как [Електронний ресурс] / nzeemin. — 2019. — Режим доступу до ресурсу: https://habr.com/en/post/475778/.
20. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. — Вінниця : ВНТУ, 2021. — 42 с.
21. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа — Вінниця : ВНТУ, 2016. — 113 с.

ДОДАТОК А

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ ВНТУ

д.т.н., проф.

_____Азаров О.Д.

“__” _____ 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«Кросплатформенний веб-застосунок для ретрансляції навчальних матеріалів в умовах карантинних обмежень»

08-23.МКР.007.00.000 ПЗ

Науковий керівник к.т.н., доц. каф. ОТ

_____ Войцеховська О.В.

Студент групи 1КІ-20м

_____ Коваленко М.М.

Вінниця 2021

Підставою для виконання МКР є потреба в покращенні методу під’єднання веб-застосунку до програмного забезпечення для створення трансляцій.

Актуальність роботи полягає в розробці веб-застосунку для ретрансляції навчальних матеріалів в умовах карантинних обмежень, що будуть напряму підключатись до стороннього програмного забезпечення для створення трансляцій, зокрема без використання сторонніх сервісів. Це дасть можливість педагогічним працівникам записувати матеріал та використовувати його для показу студентам у прямому ефірі.

Мета проекту — створення веб-застосунку для ретрансляції навчальних матеріалів, задля покращення комунікації в умовах карантинних обмежень.;

Призначення розробки — використання інтегрованого середовища розробки та мови JavaScript для розробки веб-застосунку.

Джерелами розробки МКР є: інтегроване середовище розробки VScode, мова програмування JavaScript з бібліотекою React.

Технічною вимогою до виконання МКР є наявність веб-застосунку, розробленого в VScode;

Етапи виконання МКР та очікувані результати показані в таблиці 5.1.

Таблиця 5.1 — Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасного стану досліджень			Аналітичний огляд літературних джерел, задачі досліджень
2	Практичне застосування та оцінка ефективності розроблених моделей			розділ
3	Підготовка економічної частини			розділ

Закінчення таблиці 5.1

4	Апробація та/або впровадження результатів дослідження			тези доповідей /акт впровадження
5	Опублікування результатів досліджень			стаття
6	Оформлення пояснювальної записки, графічного матеріалу і/або презентації			пояснювальна записка, графічний матеріал і/або презентація
7	Розробка веб-застосунку			розділ
8	Тестування веб-застосунку			розділ

Матеріалами, що подаються до захисту МКР є: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відзив наукового керівника, відзив опонента, анотації до МКР українською та іноземною мовами.

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

Вимоги до оформлення МКР викладені в Положеннях про випускну роботу ВНТУ, ДСТУ 2021, ДСТУ 3974-2000 «Правила виконання дослідно-конструкторських робіт. Загальні положення» та діючого ГОСТ 2.104-95 ЕСКД. Технічне завдання до виконання отримав _____ Коваленко М.М.

ДОДАТОК Б

Структура веб-застосунку для ретрансляції навчальних матеріалів

Таблиця Б.1 — Структура веб-застосунку.

Файл/директорія	Призначення
/client	Клієнтська частина веб-застосунку на React
/server	Серверна частина веб-застосунку з використанням MongoDB
/public	Тека зі збереженими форматами стилів, шрифтів
package.json	Файл зі скриптами виконання файлів та підвантаженими залежностями
Webpack.config.js	Файл для збирання веб-застосунку

ДОДАТОК В

Сторінка генерації ретрансляційного ключа

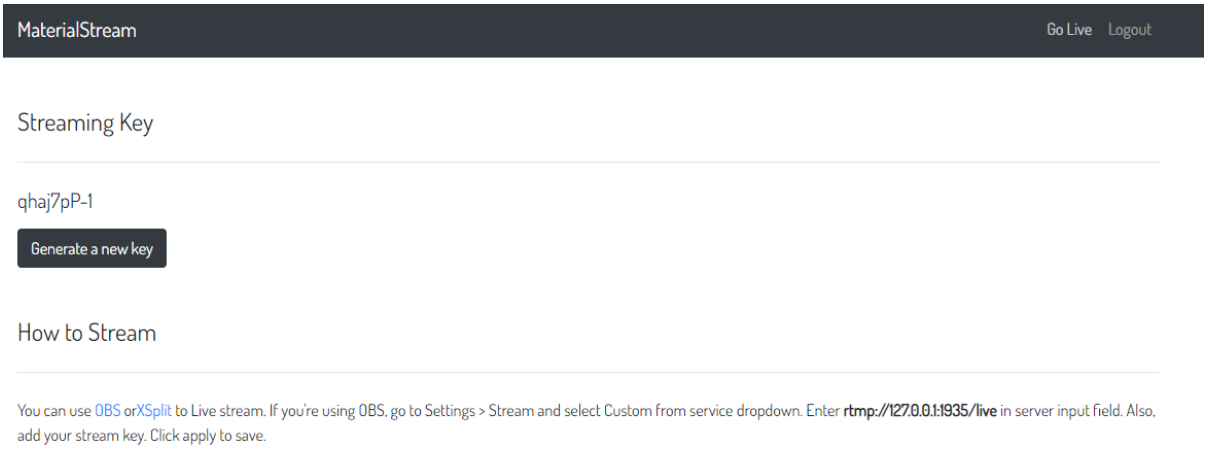


Рисунок В.1 — Генерація ретрансляційного ключа

ДОДАТОК Г

Структурна схема серверного обміну

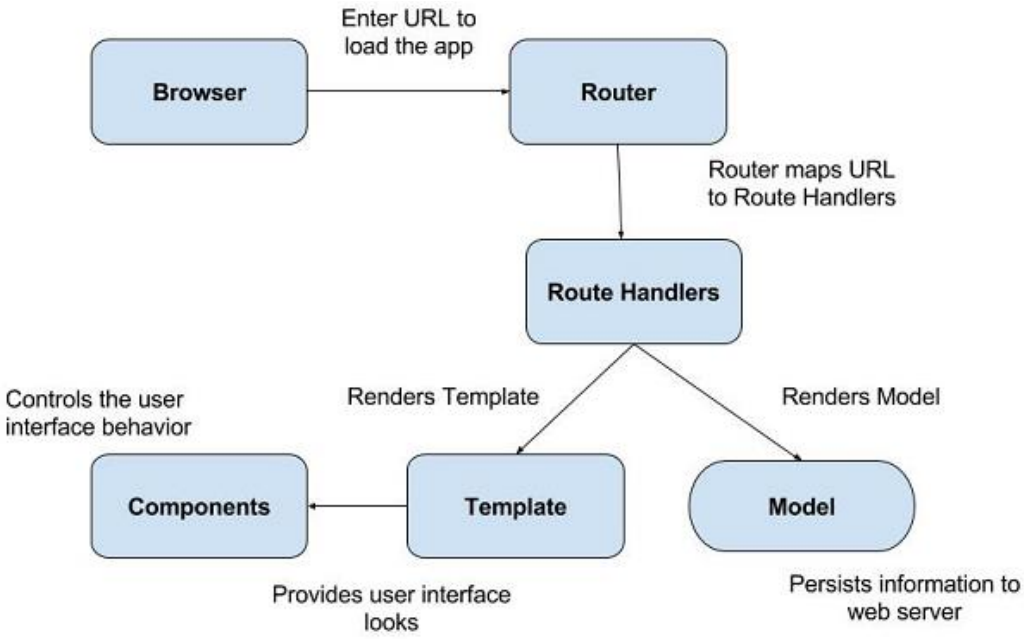
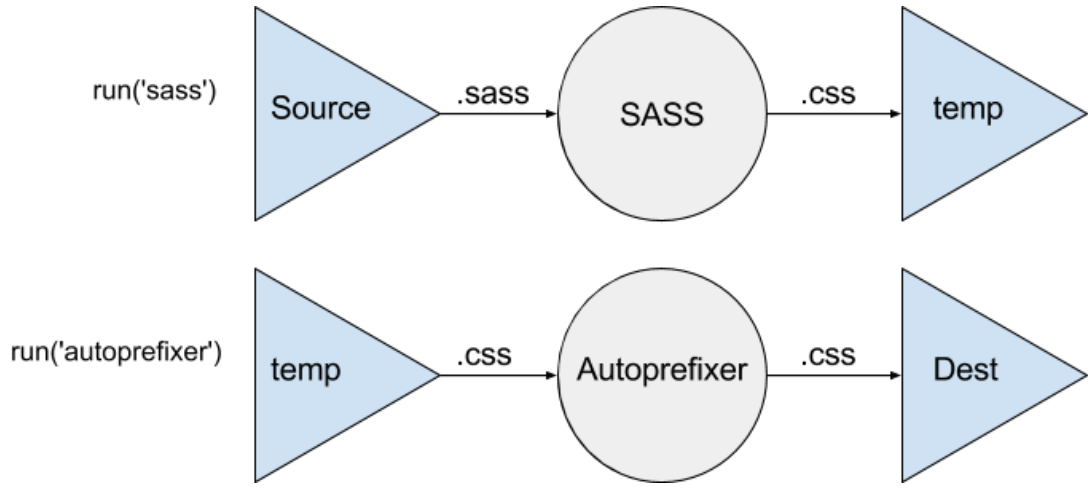


Рисунок Г.1 — Структурна схема серверного обміну в Gulp

ДОДАТОК Д

Схема роботи потоків в Gulp



1. Рисунок Д.1 — Структурна схема потоків

ДОДАТОК Е

Відео плеєр ретрансляції

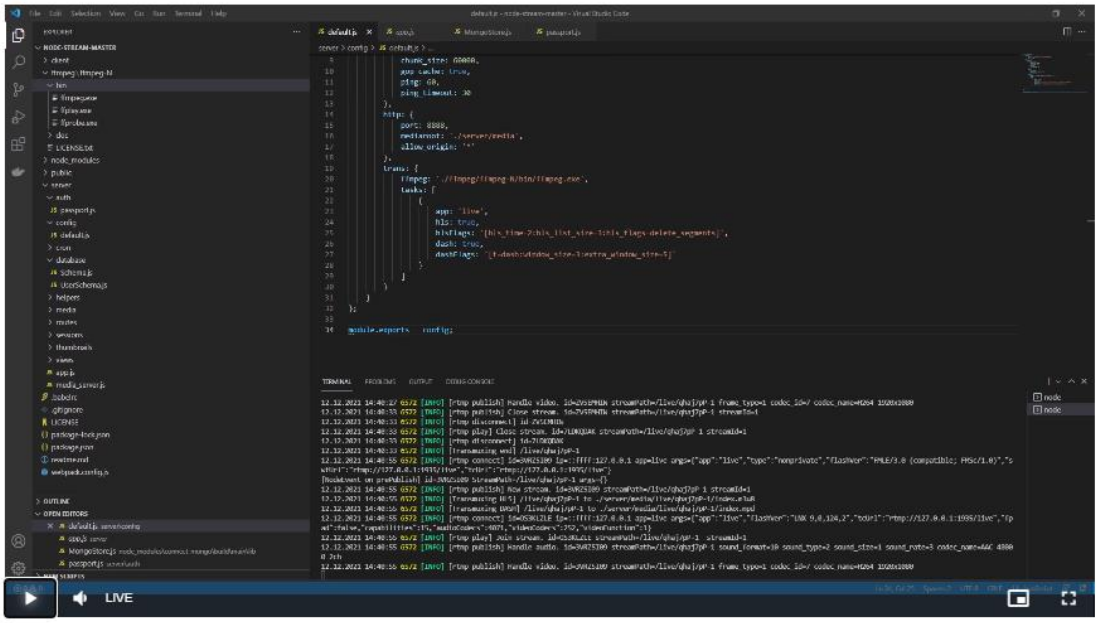


Рисунок Е.1 — Зовнішній вигляд вікна відео плеєру трансляції