ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Архітектура системи Web 3.0 у розподіленому середовищі»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Владислав МАГУРСЬКИЙ*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КІД-42
Владислав МАГУРСЬКИЙ
Ім'я, ПРІЗВИЩЕ
Керівник: Наталія ЛАЩЕВСЬКА
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерної інженерії

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерної інженерії
_____ Наталія ЛАЩЕВСЬКА
_____
«_____» _____ 2026р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Магурський Владислав Леонідович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка веб-застосунку каталогу фільмів з використанням технологій Web 3.0
керівник кваліфікаційної роботи Наталія ЛАЩЕВСЬКА
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2026р. № 56
2. Строк подання кваліфікаційної роботи «8» травня 2026р.
3. Вихідні дані до кваліфікаційної роботи: науково-технічна література; науково-технічна література; технології веб-розробки: HTML, CSS, JavaScript; принципи побудови веб-застосунків; архітектура Web 3.0; розподілені технології та децентралізовані системи; структури даних JavaScript; клієнтські веб-застосунки; технології пошуку та фільтрації даних у веб-інтерфейсах.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні основи архітектури Web 3.0
2. Аналіз та проєктування системи каталогу фільмів
3. Розробка веб-застосунку каталогу фільмів
4. Аналіз архітектури Web 3.0 у розробленому проєкті
5. Перелік ілюстративного матеріалу: *презентація*
6. Дата видачі завдання «20» лютого 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	20.02-10.03.2026	Виконано
2	Аналіз літератури та сучасних технологій Web 3.0	11.03-20.03.2026	Виконано
3	Дослідження принципів архітектури Web 3.0	21.03-05.04.2026	Виконано
4	Проєктування структури веб-застосунку каталогу фільмів	06.04-20.04.2026	Виконано
5	Розробка та тестування веб-застосунку каталогу фільмів	20.04-28.04.2026	Виконано
6	Оформлення пояснювальної записки	27.04-30.04.2026	Виконано
7	Розробка презентації	01.05-10.05.2026	Виконано
8	Передзахист	11.05-15.05.2026	Виконано
9	Здача в деканат	02.06-05.06.2026	Виконано

Здобувач вищої освіти _____ Владислав МАГУРСЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Наталія ЛАЩЕВСЬКА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 49 стор., 9 рис., 24 джерел.

Мета роботи – розробка веб-застосунку каталогу фільмів з аналізом архітектури Web 3.0 у розподілених середовищах.

Об'єкт дослідження – веб-застосунок каталогу фільмів.

Предмет дослідження – архітектура Web 3.0 та технології розробки клієнтських веб-застосунків.

Короткий зміст роботи: У сучасному світі інформаційні технології відіграють важливу роль у зберіганні, обробці та пошуку інформації. Веб-застосунки дозволяють користувачам швидко отримувати доступ до необхідних даних через мережу Інтернет. Одним із прикладів таких систем є веб-каталоги, які забезпечують зручну організацію та пошук інформації.

У даній роботі розглядається розробка веб-застосунку каталогу фільмів, який дозволяє користувачам переглядати інформацію про фільми, здійснювати пошук та фільтрацію даних. Для реалізації застосунку використано технології HTML, CSS та JavaScript, що забезпечують створення сучасного інтерактивного інтерфейсу користувача.

Особлива увага приділяється аналізу архітектури Web 3.0, яка передбачає використання децентралізованих технологій, розподілених систем зберігання даних та нових підходів до організації веб-застосунків.

У результаті виконання кваліфікаційної роботи було розроблено веб-застосунок каталогу фільмів, який забезпечує зручний перегляд інформації, пошук та фільтрацію даних.

КЛЮЧОВІ СЛОВА: WEB 3.0, ВЕБ-ЗАСТОСУНОК, КАТАЛОГ ФІЛЬМІВ, HTML, CSS, JAVASCRIPT, РОЗПОДІЛЕНІ ТЕХНОЛОГІЇ, КЛІЄНТСЬКІ ВЕБ-ЗАСТОСУНК

ЗМІСТ

ВСТУП…… 6
1. ТЕОРЕТИЧНІ ОСНОВИ WEB 3.0 ТА РОЗПОДІЛЕНИХ СИСТЕМ…… 9
1.1 Поняття та еволюція веб-технологій ……..... 9
1.2 Основні принципи Web 3.0…….…… 10
1.3 Розподілені системи та їх особливості…… 13
1.4 Технології Web 3.0 ……..... 12
2. АНАЛІЗ ТА ПРОЄКТУВАННЯ СИСТЕМИ КАТАЛОГУ ФІЛЬМІВ…….. 18
2.1 Опис предметної області……... 18
2.2 Аналіз існуючих аналогів ……..... 20
2.3 Вимоги до системи…….…….... 22
2.4 Архітектура системи…….…… 25
2.5 Вибір технологій та інструментів……..... 26
3. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ……..... 30
3.1 Загальна структура застосунку……..... 30
3.2 Реалізація клієнтської частини …… 31
3.3. Реалізація бази даних …….. 33
3.4 Реалізація основного функціоналу ……..... 34
4. АНАЛІЗ WEB 3.0 АРХІТЕКТУРИ В ПРОЄКТІ……... 36
4.1 Відповідність системи принципам Web 3.0…… 36
4.2 Використання розподілених технологій …… 37
4.3. Переваги та недоліки реалізованого підходу …… 39
4.4 Перспективи розвитку системи ……..... 41
ВИСНОВКИ…… 44
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ…… 46
Додаток А. Інтерфейс системи ……..... 47
Додаток Б. Структура бази даних ……..... 48
Додаток В. Фрагменти коду ……... 48
Додаток Г. Презентація ……... 51

ВСТУП

У сучасному світі інформаційні технології розвиваються надзвичайно швидкими темпами. Інтернет став невід'ємною частиною життя людини та використовується практично у всіх сферах діяльності: освіті, бізнесі, науці, розвагах та комунікації. Веб-технології постійно еволюціонують, пропонуючи нові можливості для створення ефективних інформаційних систем та веб-застосунків. Одним із нових етапів розвитку Інтернету є концепція Web 3.0, яка базується на принципах децентралізації, прозорості, безпеки та використання розподілених середовищ.

На початковому етапі розвитку мережі Інтернет домінувала концепція Web 1.0, яка характеризувалася статичними веб-сторінками та обмеженою взаємодією з користувачем. Основною функцією таких сайтів було надання інформації, а можливості взаємодії користувачів із контентом були мінімальними. Наступним етапом розвитку стала концепція Web 2.0, що принесла інтерактивність, соціальні мережі, можливість створення контенту користувачами, а також активну взаємодію між користувачами та сервісами. Проте Web 2.0 має ряд недоліків, серед яких централізація даних, залежність від великих компаній та потенційні ризики для безпеки та конфіденційності інформації.

З метою подолання цих проблем виникла нова концепція розвитку Інтернету — Web 3.0, яка передбачає використання децентралізованих технологій, таких як блокчейн, розподілені системи зберігання даних та смарт-контракти. Основною ідеєю Web 3.0 є створення середовища, у якому користувачі мають більший контроль над власними даними, а інформація зберігається у розподілених мережах, що підвищує надійність, прозорість і стійкість систем до збоїв або несанкціонованого втручання.

Одним із важливих напрямів застосування сучасних веб-технологій є створення інформаційних веб-систем, які забезпечують зручний доступ до

структурованих даних. Прикладом такої системи є каталог фільмів, який дозволяє користувачам отримувати інформацію про різні кінострічки, переглядати їх опис, рейтинг, жанр, дату випуску та інші характеристики. Подібні системи широко використовуються на різних інтернет-платформах, оскільки допомагають користувачам швидко знаходити потрібну інформацію та орієнтуватися у великій кількості медіаконтенту.

Розробка веб-застосунків подібного типу вимагає використання сучасних підходів до проєктування архітектури програмних систем. Особливу роль у цьому відіграють розподілені середовища, які дозволяють ефективно обробляти великі обсяги інформації, забезпечувати масштабованість системи та підвищувати її надійність. У розподілених системах компоненти програмного забезпечення можуть знаходитися на різних вузлах мережі та взаємодіяти між собою через мережеві протоколи. Такий підхід дозволяє підвищити продуктивність системи, забезпечити її гнучкість та можливість подальшого розширення.

Актуальність теми даної роботи обумовлена стрімким розвитком веб-технологій та зростаючою потребою у створенні ефективних інформаційних систем, які можуть працювати у розподілених середовищах. Використання принципів Web 3.0 відкриває нові можливості для побудови сучасних веб-застосунків, які забезпечують більш високий рівень безпеки, надійності та масштабованості.

Метою даної роботи є дослідження архітектури Web 3.0 систем у розподілених середовищах та розробка веб-застосунку каталогу фільмів, який демонструє принципи побудови сучасних веб-систем.

Для досягнення поставленої мети необхідно вирішити такі завдання:

— дослідити основні етапи розвитку веб-технологій;
— розглянути особливості концепції Web 3.0;
— проаналізувати принципи побудови розподілених систем;
— дослідити архітектуру сучасних веб-застосунків;
— спроєктувати структуру веб-системи каталогу фільмів;

— реалізувати основний функціонал веб-застосунку;
— провести аналіз отриманих результатів та визначити перспективи розвитку системи.

Практичне значення роботи полягає у створенні веб-застосунку каталогу фільмів, який може використовуватися для перегляду та пошуку інформації про кінофільми. Отримані результати можуть бути використані як основа для подальшого розвитку подібних інформаційних систем, а також для дослідження можливостей інтеграції Web 3.0 технологій у веб-застосунки.

Таким чином, розробка веб-систем на основі сучасних технологій та принципів розподілених середовищ є актуальним завданням у сфері інформаційних технологій. Дослідження архітектури Web 3.0 систем дозволяє глибше зрозуміти перспективи розвитку Інтернету та сприяє створенню більш ефективних і безпечних веб-рішень.

1. ТЕОРЕТИЧНІ ОСНОВИ WEB 3.0 ТА РОЗПОДІЛЕНИХ СИСТЕМ

1.1 Еволюція веб-технологій: Web 1.0, Web 2.0 та Web 3.0

Розвиток мережі Інтернет проходив декілька етапів, кожен з яких характеризувався новими можливостями взаємодії користувачів з інформацією та новими підходами до побудови веб-систем. Найчастіше виділяють три основні етапи розвитку веб-технологій: Web 1.0, Web 2.0 та Web 3.0 (рис. 1.1) .

Перший етап розвитку веб-технологій отримав назву Web 1.0. Він характеризувався використанням статичних веб-сторінок, які містили лише інформацію для читання. Користувачі могли переглядати контент, але практично не мали можливості взаємодіяти з ним. Більшість сайтів у цей період виконували функцію електронних довідників або інформаційних ресурсів. Оновлення інформації здійснювалося виключно власниками сайтів, а можливості взаємодії користувачів між собою були відсутні.

Наступним етапом розвитку став Web 2.0, який значно змінив підхід до використання Інтернету. Основною особливістю Web 2.0 стала поява інтерактивних веб-застосунків та можливість активної участі користувачів у створенні контенту. Саме в цей період з’явилися соціальні мережі, блоги, вікі-платформи та різноманітні онлайн-сервіси. Користувачі отримали можливість коментувати інформацію, створювати власний контент, ділитися ним з іншими та взаємодіяти між собою через Інтернет.

Незважаючи на значний розвиток веб-технологій, Web 2.0 має певні недоліки. Одним з основних є централізація даних. Більшість сервісів зберігають інформацію на централізованих серверах великих компаній, що може створювати ризики втрати даних, цензури або несанкціонованого доступу до інформації.

З метою вирішення цих проблем була запропонована концепція Web 3.0, яка передбачає використання децентралізованих технологій та розподілених систем.

Основною ідеєю Web 3.0 є створення середовища, у якому користувачі мають більший контроль над власними даними, а інформація зберігається не на одному сервері, а у розподіленій мережі.

Web 3.0 поєднує використання сучасних технологій, серед яких можна виділити блокчейн, смарт-контракти, розподілені системи зберігання даних та криптографічні методи захисту інформації. Завдяки цьому підвищується рівень безпеки, прозорості та надійності веб-систем.



Рисунок 1.1 – Еволюція веб-технологій (Web 1.0 → Web 2.0 → Web 3.0)

1.2 Основні принципи Web 3.0

Концепція Web 3.0 є новим етапом розвитку мережі Інтернет, який спрямований на створення більш відкритого, безпечного та децентралізованого цифрового середовища. Основною ідеєю Web 3.0 є зміна принципів взаємодії між

користувачами, сервісами та даними. Якщо у попередніх поколіннях веб-технологій контроль над інформацією здебільшого належав великим компаніям та централізованим платформам, то у Web 3.0 значна частина цього контролю переходить безпосередньо до користувачів.

Одним із ключових принципів Web 3.0 є децентралізація. У традиційних веб-системах більшість даних зберігається на централізованих серверах, які контролюються певними організаціями або компаніями. Такий підхід має певні недоліки, серед яких залежність від одного сервера, можливість цензури інформації, а також ризик втрати або викрадення даних у разі зламу системи. У Web 3.0 використовується децентралізована структура, у якій інформація може зберігатися на великій кількості вузлів мережі. Це дозволяє підвищити надійність системи, оскільки відмова одного вузла не призводить до втрати всієї інформації.

Іншим важливим принципом є прозорість та відкритість системи. У багатьох Web 3.0 платформах використовується технологія блокчейн, яка дозволяє зберігати інформацію у вигляді послідовності блоків. Кожен блок містить дані про певні операції або транзакції та пов'язаний з попереднім блоком за допомогою криптографічного хешу. Така структура забезпечує високий рівень прозорості, оскільки всі учасники мережі можуть перевірити історію змін у системі. Завдяки цьому значно зменшується ймовірність підробки або несанкціонованої зміни інформації.

Важливим аспектом Web 3.0 є підвищений рівень безпеки. Для захисту інформації використовуються сучасні криптографічні методи, які дозволяють забезпечити конфіденційність даних та їх цілісність. У децентралізованих системах дані можуть зберігатися у зашифрованому вигляді, що значно ускладнює доступ до них для сторонніх осіб. Крім того, використання розподілених мереж дозволяє зменшити ризик атак на систему, оскільки зловмиснику необхідно отримати контроль над великою кількістю вузлів одночасно.

Ще одним важливим принципом Web 3.0 є контроль користувачів над власними даними. У сучасних централізованих веб-сервісах користувачі часто змушені передавати свої персональні дані різним платформам, які можуть

використовувати їх для реклами або інших комерційних цілей. У Web 3.0 користувачі можуть самостійно визначати, які саме дані вони хочуть надати та кому саме. Це досягається за допомогою децентралізованих ідентифікаційних систем, криптографічних ключів та інших технологій.

Також важливою характеристикою Web 3.0 є використання смарт-контрактів. Смарт-контракт являє собою програмний код, який автоматично виконує певні дії при виконанні заданих умов. Такі контракти працюють у блокчейн-мережах і не потребують участі посередників. Це дозволяє значно спростити процеси взаємодії між користувачами, підвищити швидкість виконання операцій та зменшити витрати.



Рисунок 1.2 – Основні принципи Web 3.0

Крім того, Web 3.0 передбачає використання семантичних технологій, які дозволяють комп'ютерним системам краще розуміти зміст інформації, що міститься у веб-ресурсах. Це сприяє більш ефективному пошуку інформації,

автоматичній обробці даних та покращенню взаємодії між різними інформаційними системами [1].

Таким чином, основні принципи Web 3.0 спрямовані на створення більш відкритої, безпечної та децентралізованої мережі Інтернет. Використання таких підходів дозволяє підвищити надійність веб-систем, забезпечити кращий захист даних користувачів та створити нові можливості для розвитку інформаційних технологій.

1.3 Розподілені системи та їх особливості

Розподілені системи відіграють важливу роль у розвитку сучасних інформаційних технологій. Вони використовуються у багатьох сферах діяльності, зокрема у хмарних обчисленнях, великих інформаційних системах, мережевих сервісах та сучасних веб-платформах. Розвиток концепції Web 3.0 значною мірою пов’язаний саме з використанням розподілених систем, оскільки такі системи дозволяють організувати зберігання та обробку інформації без єдиного центра керування.

Розподілена система — це сукупність незалежних комп’ютерів або вузлів, які з’єднані між собою мережею та взаємодіють для виконання спільних завдань. Кожен вузол у такій системі може виконувати певні функції, а обмін інформацією між вузлами здійснюється за допомогою мережевих протоколів. Для користувача така система виглядає як єдине ціле, хоча насправді її компоненти можуть знаходитися у різних фізичних місцях.

Однією з основних характеристик розподілених систем є відсутність єдиного центрального вузла, який повністю контролює роботу всієї системи. У традиційних централізованих системах більшість процесів виконується на одному сервері, що може створювати проблеми у разі перевантаження або відмови

обладнання. У розподілених системах обчислювальні ресурси розподіляються між кількома вузлами, що дозволяє підвищити ефективність роботи та забезпечити більш стабільне функціонування системи.

Однією з важливих переваг розподілених систем є їх масштабованість. Це означає, що систему можна розширювати шляхом додавання нових вузлів без значних змін у її архітектурі. Такий підхід дозволяє збільшувати продуктивність системи відповідно до зростання кількості користувачів або обсягу оброблюваних даних.

Ще однією важливою характеристикою є відмовостійкість. У розподілених системах вихід з ладу одного вузла не обов’язково призводить до припинення роботи всієї системи. Інші вузли можуть продовжувати виконувати свої функції, що підвищує надійність системи та зменшує ризик втрати даних.

Крім того, розподілені системи забезпечують підвищену продуктивність. Завдяки розподілу обчислювальних задач між декількома вузлами система може обробляти більший обсяг інформації за менший час. Це особливо важливо для сучасних веб-застосунків, які повинні обслуговувати велику кількість користувачів одночасно.

Важливою характеристикою розподілених систем є також прозорість. Прозорість означає, що користувач не повинен знати, на якому саме вузлі виконується певна операція або де саме зберігаються дані. Система автоматично розподіляє ресурси та керує взаємодією між компонентами, забезпечуючи зручну роботу для користувача.

Разом з тим розподілені системи мають і певні складності. Однією з основних проблем є синхронізація даних між вузлами. Оскільки інформація може зберігатися на різних серверах, необхідно забезпечити її узгодженість. Також важливими питаннями є забезпечення безпеки передачі даних, управління мережею та оптимізація взаємодії між компонентами системи.



Рисунок 1.3 – Структура розподіленої системи

У сучасних веб-системах розподілені технології використовуються дуже активно. Наприклад, великі інтернет-платформи застосовують розподілені бази даних, хмарні сервіси та мережі доставки контенту. У системах Web 3.0 роль розподілених технологій ще більше зростає, оскільки вони є основою для реалізації децентралізованих платформ та сервісів [2].

Таким чином, розподілені системи є важливою складовою сучасних інформаційних технологій. Вони забезпечують високу продуктивність, надійність та масштабованість веб-систем, що робить їх необхідними для створення сучасних інтернет-сервісів та платформ.

1.4 Технології Web 3.0 (Blockchain, IPFS, смарт-контракти)

Розвиток концепції Web 3.0 став можливим завдяки появі нових інформаційних технологій, які дозволяють реалізувати принципи децентралізації,

безпеки та незалежності користувачів від централізованих платформ. До основних технологій, що використовуються у Web 3.0, належать блокчейн, смарт-контракти, розподілені системи зберігання даних та криптографічні методи захисту інформації.

Однією з ключових технологій Web 3.0 є блокчейн. Блокчейн являє собою розподілену базу даних, у якій інформація зберігається у вигляді послідовності блоків, пов'язаних між собою за допомогою криптографічних методів. Кожен блок містить певний набір даних, а також спеціальний хеш попереднього блоку. Така структура дозволяє забезпечити цілісність інформації, оскільки будь-яка спроба змінити дані у вже створеному блоці призведе до порушення всієї структури ланцюга.

Особливістю блокчейну є те, що копії бази даних можуть зберігатися на великій кількості вузлів мережі. Учасники мережі можуть перевіряти достовірність інформації, що значно підвищує прозорість та надійність системи. Саме завдяки цим властивостям блокчейн широко використовується у децентралізованих платформах та фінансових системах [3].

Ще однією важливою технологією Web 3.0 є смарт-контракти. Смарт-контракт — це програмний код, який автоматично виконує певні умови угоди між сторонами. Такий контракт зберігається у блокчейн-мережі та виконується автоматично після виконання заданих умов. Це дозволяє здійснювати різні операції без участі посередників, що знижує витрати та підвищує ефективність взаємодії між користувачами.

Смарт-контракти можуть використовуватися у різних сферах діяльності, наприклад у фінансових операціях, управлінні цифровими активами, електронній комерції та системах голосування. Завдяки автоматичному виконанню умов контракту зменшується ймовірність помилок або маніпуляцій з боку учасників угоди.

Важливу роль у Web 3.0 також відіграють розподілені системи зберігання даних. У традиційних веб-системах файли та інші дані зберігаються на централізованих серверах, що може створювати ризик втрати інформації або

обмеження доступу до неї. У розподілених системах зберігання даних інформація розміщується на багатьох вузлах мережі, що забезпечує її більшу доступність та надійність.

Одним із прикладів такої технології є IPFS (InterPlanetary File System). Це розподілена файлова система, яка дозволяє зберігати та передавати дані у мережі без використання централізованих серверів. У системі IPFS кожен файл отримує унікальний ідентифікатор, який формується на основі криптографічного хешу. Завдяки цьому користувачі можуть отримувати доступ до файлів незалежно від їх фізичного розташування.

Крім того, у Web 3.0 широко застосовуються сучасні криптографічні технології. Вони використовуються для захисту інформації, підтвердження автентичності користувачів та забезпечення безпечної передачі даних між вузлами мережі. Криптографічні алгоритми дозволяють створювати цифрові підписи, шифрувати інформацію та перевіряти її цілісність.

Застосування таких технологій дозволяє створювати нові типи веб-застосунків, які мають більш високий рівень безпеки, надійності та незалежності від централізованих платформ. Саме ці технології формують основу сучасних Web 3.0 систем та забезпечують можливість створення децентралізованих інтернет-сервісів.

2. АНАЛІЗ ТА ПРОЄКТУВАННЯ СИСТЕМИ КАТАЛОГУ ФІЛЬМІВ

2.1 Опис предметної області

Сфера кіноіндустрії є однією з найбільш динамічних галузей сучасної індустрії розваг. Щороку у світі створюється велика кількість нових фільмів різних жанрів, стилів та тематик. У зв'язку з цим виникає потреба у системах, які дозволяють ефективно зберігати, впорядковувати та надавати користувачам доступ до інформації про кінострічки. Одним із найпоширеніших рішень для організації такої інформації є веб-системи у вигляді каталогів фільмів.

Каталог фільмів — це інформаційна система, яка містить структуровані дані про різні фільми та надає користувачам можливість переглядати, шукати та отримувати детальну інформацію про кінострічки. Основним завданням такої системи є організація зручного доступу до інформації про фільми та забезпечення швидкого пошуку необхідних даних.

У подібних системах зазвичай зберігається різноманітна інформація про фільми. До основних характеристик фільму належать назва, рік випуску, жанр, короткий опис, рейтинг, тривалість, країна виробництва, режисер та акторський склад. Наявність структурованої інформації дозволяє користувачам швидко орієнтуватися у великій кількості фільмів та знаходити ті, які їх цікавлять.

Веб-каталоги фільмів широко використовуються у сучасному Інтернеті. Вони можуть бути частиною великих кіноінформаційних порталів, онлайн-кінотеатрів або окремими інформаційними ресурсами, що надають користувачам доступ до бази даних фільмів. Подібні системи можуть виконувати різні функції, починаючи від простого перегляду інформації про фільми і закінчуючи більш

складними функціями, такими як формування рекомендацій, створення списків улюблених фільмів або залишення відгуків користувачами.

Однією з важливих характеристик таких систем є можливість пошуку та фільтрації інформації. Користувачі повинні мати можливість знаходити фільми за різними параметрами, наприклад за назвою, жанром, роком випуску або рейтингом. Це значно спрощує процес пошуку потрібної інформації та підвищує зручність використання системи.

Крім того, важливою складовою каталогу фільмів є зручний користувацький інтерфейс. Інтерфейс повинен бути зрозумілим, простим у використанні та забезпечувати швидку навігацію між сторінками сайту. Користувач повинен мати можливість легко переходити між списком фільмів, сторінками з детальною інформацією та іншими елементами системи.

Для реалізації подібної системи зазвичай використовується веб-застосунок, який складається з клієнтської частини, серверної частини та системи зберігання даних. Клієнтська частина відповідає за відображення інформації у браузері користувача, серверна частина обробляє запити та забезпечує взаємодію з базою даних, а база даних зберігає інформацію про фільми.

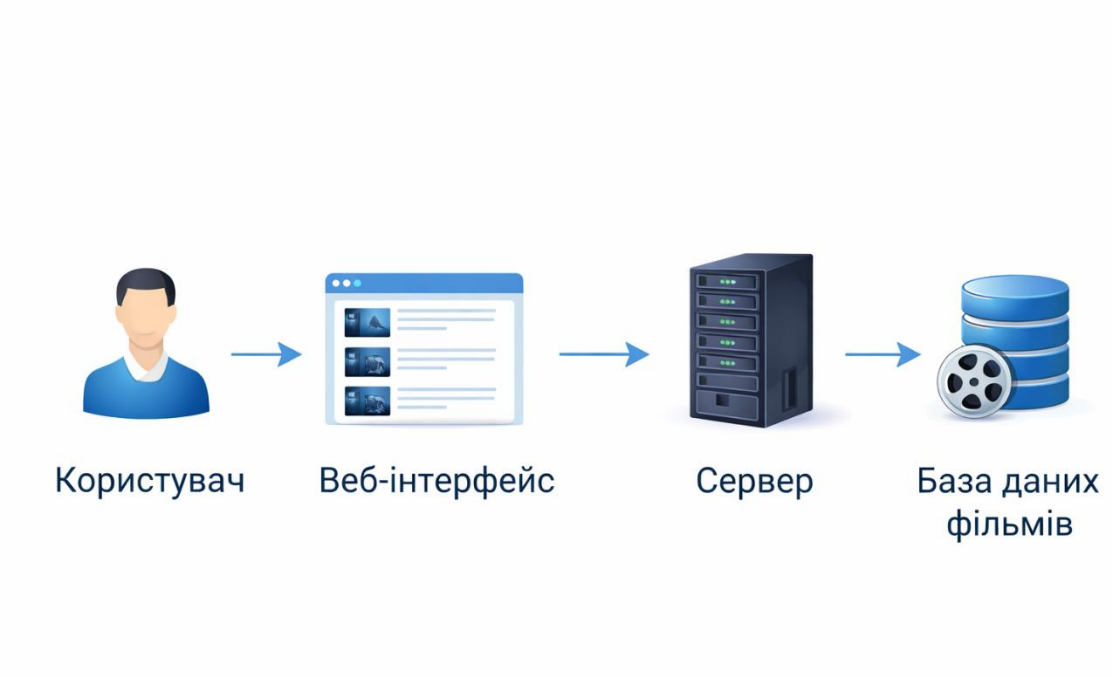


Рис. 2.1 - Загальна структура інформаційної системи каталогу фільмів

Важливим аспектом під час розробки такої системи є можливість її подальшого розвитку та розширення. Наприклад, у майбутньому до системи можуть бути додані нові функції, такі як реєстрація користувачів, система оцінювання фільмів, рекомендаційні алгоритми або інтеграція з іншими інформаційними сервісами. Тому при проєктуванні системи необхідно враховувати її масштабованість та гнучкість.

Таким чином, предметна область даної роботи охоплює процеси збирання, зберігання, обробки та представлення інформації про фільми у вигляді веб-застосунку. Розробка системи каталогу фільмів дозволяє створити зручний інструмент для доступу до інформації про кінострічки та демонструє можливості використання сучасних веб-технологій для створення інформаційних систем [4].

2.2 Аналіз існуючих аналогів

Під час розробки будь-якої інформаційної системи важливим етапом є аналіз існуючих рішень, які вже представлені на ринку. Такий аналіз дозволяє визначити основні функціональні можливості подібних систем, оцінити їх переваги та недоліки, а також врахувати найкращі підходи при проєктуванні власного програмного продукту. У випадку веб-каталогів фільмів існує велика кількість популярних платформ, які надають користувачам доступ до інформації про кінострічки.

Одним із найвідоміших прикладів таких систем є IMDb (Internet Movie Database). Це одна з найбільших у світі онлайн-баз даних, що містить інформацію про фільми, телевізійні програми, акторів, режисерів та інших учасників кіноіндустрії. Система IMDb дозволяє користувачам переглядати детальну

інформацію про фільми, включаючи опис, рейтинг, дату виходу, список акторів, трейлери та інші відомості. Крім того, користувачі можуть залишати власні оцінки та відгуки, що дозволяє формувати загальний рейтинг фільмів. Такий підхід дозволяє створити велику спільноту користувачів, які беруть участь у наповненні бази даних та оцінюванні контенту [5].

Іншим популярним прикладом є сервіс Rotten Tomatoes, який спеціалізується на збиранні рецензій критиків та користувачів. Особливістю цієї системи є агрегування оцінок із різних джерел та формування загального рейтингу фільму. Rotten Tomatoes надає користувачам інформацію про популярність фільмів, їх рейтинг серед критиків, а також відгуки глядачів. Це дозволяє користувачам отримати більш об’єктивну оцінку якості кінострічки перед її переглядом.

Ще одним прикладом є платформа **Letterboxd**, яка поєднує функції каталогу фільмів та соціальної мережі. Користувачі можуть створювати власні списки фільмів, залишати рецензії, ставити оцінки та ділитися своїми враженнями з іншими користувачами. Такий підхід дозволяє створити інтерактивне середовище, у якому користувачі можуть активно взаємодіяти між собою та обмінюватися рекомендаціями щодо перегляду фільмів.

Аналіз існуючих аналогів дозволяє виділити основні функціональні можливості, які характерні для більшості подібних систем. До таких функцій належать:

— перегляд списку фільмів;
— пошук фільмів за назвою;
— перегляд детальної інформації про фільм;
— відображення рейтингу та оцінок користувачів;
— фільтрація фільмів за жанром або роком випуску.

Окрім цього, багато сучасних систем мають додаткові функції, такі як рекомендаційні алгоритми, можливість створення персональних списків фільмів та інтеграцію з соціальними мережами (рис. 2.2) .



Рис. 2.2 - Приклади існуючих систем каталогів фільмів

Незважаючи на широкий функціонал великих кіноінформаційних платформ, для навчальних або демонстраційних проєктів зазвичай реалізується спрощена версія подібної системи. Основна мета такої системи полягає у забезпеченні базового функціоналу перегляду інформації про фільми та демонстрації принципів роботи веб-застосунків.

Таким чином, аналіз існуючих аналогів показує, що сучасні системи каталогів фільмів мають розвинений функціонал, який включає зберігання великої кількості інформації, систему рейтингів, пошук та фільтрацію даних. Отримані результати аналізу можуть бути використані під час проєктування власної системи каталогу фільмів, що дозволить врахувати основні принципи побудови подібних веб-застосунків та реалізувати необхідний функціонал.

2.3 Вимоги до системи

Під час розробки програмного забезпечення важливим етапом є визначення вимог до системи. Вимоги визначають основні функції, які повинна виконувати система, а також обмеження та характеристики, яким вона повинна відповідати. Чітке формулювання вимог дозволяє правильно спроєктувати структуру системи, визначити її функціональні можливості та забезпечити ефективну роботу програмного продукту.

У даній роботі розробляється веб-застосунок каталогу фільмів, який повинен забезпечувати користувачам зручний доступ до інформації про різні кінострічки. Для цього необхідно визначити основні функціональні та нефункціональні вимоги до системи.

Функціональні вимоги описують основні можливості системи та дії, які вона повинна виконувати. До основних функціональних вимог системи каталогу фільмів належать:

— відображення списку фільмів на головній сторінці;
— перегляд детальної інформації про обраний фільм;
— можливість пошуку фільмів за назвою;
— відображення основних характеристик фільму (назва, жанр, рік випуску, опис, рейтинг);
— можливість переходу між сторінками системи;
— зручне відображення інформації у веб-браузері користувача.

Ці функції є базовими для більшості веб-систем, які працюють з інформаційними каталогами, оскільки вони забезпечують основну взаємодію користувача з даними системи.

Окрім функціональних вимог, важливо визначити також нефункціональні вимоги. Вони описують характеристики системи, які не пов’язані безпосередньо з виконанням певних функцій, але впливають на якість її роботи.

До основних нефункціональних вимог системи можна віднести:

— зручність використання – інтерфейс повинен бути зрозумілим та простим для користувача;
— швидкодія – система повинна швидко обробляти запити користувачів та

відображати інформацію;

— надійність – система повинна стабільно працювати без збоїв;
— масштабованість – можливість розширення функціоналу системи у майбутньому;
— сумісність – веб-застосунок повинен коректно працювати у різних браузерах та на різних пристроях.

Важливим аспектом під час визначення вимог є також структура даних, які будуть зберігатися у системі. Для каталогу фільмів необхідно передбачити зберігання інформації про кожен фільм у вигляді окремого запису, який міститиме основні характеристики. Така структура дозволить легко виконувати пошук, сортування та відображення інформації у веб-застосунку.



Рис. 2.3 - Основні функціональні вимоги системи

Крім того, під час розробки системи необхідно враховувати можливість подальшого розширення функціоналу. У майбутньому до системи можуть бути додані нові можливості, наприклад система реєстрації користувачів, створення персональних списків фільмів або можливість залишати відгуки. Тому архітектура системи повинна бути достатньо гнучкою для реалізації подібних змін.

Таким чином, визначення вимог до системи є важливим етапом проєктування програмного забезпечення. Чітко сформульовані вимоги дозволяють створити ефективну та зручну у використанні систему каталогу фільмів, яка забезпечує швидкий доступ до інформації та може бути розширена у майбутньому [6].

2.4 Архітектура системи

Архітектура програмної системи визначає загальну структуру програмного забезпечення, основні компоненти системи та принципи їх взаємодії між собою. Правильно спроєктована архітектура дозволяє забезпечити ефективну роботу системи, спрощує її підтримку та дає можливість розширювати функціональні можливості у майбутньому. Під час розробки веб-застосунків часто використовується багаторівнева архітектура, яка передбачає поділ системи на окремі логічні компоненти.

У системі каталогу фільмів доцільно використовувати класичну клієнт-серверну архітектуру. У такій архітектурі система складається з клієнтської частини, серверної частини та системи зберігання даних. Кожен із цих компонентів виконує власні функції та взаємодіє з іншими частинами системи.

Клієнтська частина системи відповідає за взаємодію з користувачем. Вона реалізується у вигляді веб-інтерфейсу, який відкривається у браузері користувача. Саме клієнтська частина забезпечує відображення списку фільмів, сторінок з детальною інформацією та інших елементів інтерфейсу. Користувач взаємодіє із системою через веб-сторінки, надсилаючи запити до серверної частини.

Серверна частина системи відповідає за обробку запитів, що надходять від клієнта. Сервер приймає запити користувача, обробляє їх, звертається до бази даних для отримання необхідної інформації та повертає результати у вигляді веб-сторінок або даних. Саме серверна частина реалізує основну логіку роботи системи, включаючи обробку пошуку фільмів, формування списку фільмів та

відображення детальної інформації про них.

Третім важливим компонентом системи є база даних, у якій зберігається вся інформація про фільми. У базі даних можуть зберігатися такі дані, як назва фільму, рік випуску, жанр, опис, рейтинг та інші характеристики. Використання бази даних дозволяє ефективно зберігати великі обсяги інформації та швидко виконувати пошук і сортування даних.

Взаємодія між компонентами системи відбувається за допомогою мережевих протоколів. Коли користувач відкриває сторінку сайту або виконує пошук фільму, браузер надсилає запит на сервер. Сервер обробляє цей запит, звертається до бази даних для отримання необхідної інформації та надсилає результат назад до клієнтської частини, де він відображається у вигляді веб-сторінки.

Використання клієнт-серверної архітектури має низку переваг. По-перше, така структура дозволяє розділити функції між різними компонентами системи, що спрощує її розробку та підтримку. По-друге, система стає більш масштабованою, оскільки за потреби можна змінювати або вдосконалювати окремі компоненти без значного впливу на інші частини системи. По-третє, така архітектура забезпечує ефективну обробку запитів великої кількості користувачів [7].

Крім того, поділ системи на окремі рівні дозволяє підвищити безпеку системи, оскільки прямий доступ користувачів до бази даних відсутній. Усі операції з даними виконуються через серверну частину, що дозволяє контролювати доступ до інформації та забезпечувати її захист.

Таким чином, архітектура системи каталогу фільмів базується на клієнт-серверному підході та включає три основні компоненти: клієнтський інтерфейс, серверну частину та базу даних. Такий підхід дозволяє забезпечити ефективну роботу системи, зручну взаємодію з користувачем та можливість подальшого розширення функціоналу.

2.5 Вибір технологій та інструментів

Під час розробки веб-застосунку важливим етапом є вибір технологій, за допомогою яких буде реалізовано функціонал системи. Вибрані технології повинні забезпечувати простоту розробки, зручність використання для користувачів та достатню функціональність для реалізації поставлених завдань. У даній роботі для створення системи каталогу фільмів використано базові веб-технології: HTML, CSS та JavaScript.

Основою будь-якого веб-сайту є HTML (HyperText Markup Language) — мова розмітки гіпертексту, яка використовується для створення структури веб-сторінок. За допомогою HTML визначаються основні елементи сторінки, такі як заголовки, текстові блоки, списки, кнопки, зображення та інші компоненти інтерфейсу. HTML дозволяє організувати логічну структуру веб-сторінки та забезпечує відображення контенту у браузері користувача. У розробленій системі HTML використовується для створення сторінок каталогу фільмів та розміщення інформації про кожен фільм [8].

Для оформлення веб-сторінок використовується CSS (Cascading Style Sheets). Ця технологія призначена для стилізації елементів, створених за допомогою HTML. За допомогою CSS можна змінювати кольори, шрифти, розміри елементів, відступи, розташування блоків та інші параметри зовнішнього вигляду сторінки. Використання CSS дозволяє зробити інтерфейс веб-сайту більш зручним та привабливим для користувачів. У даному проєкті CSS використовується для оформлення сторінок каталогу фільмів, створення структури сторінки та забезпечення зрозумілого користувацького інтерфейсу [8].

Для реалізації інтерактивних елементів використовується JavaScript — мова програмування, яка широко застосовується у веб-розробці. JavaScript дозволяє додавати динамічну поведінку веб-сторінкам, обробляти дії користувача та змінювати вміст сторінки без її повного перезавантаження. За допомогою JavaScript можна реалізувати різні функції, наприклад пошук інформації, фільтрацію даних або взаємодію з елементами інтерфейсу.

У розробленій системі використовується два файли JavaScript, які відповідають за виконання окремих функцій системи. Такий підхід дозволяє

розділити логіку роботи програми на окремі модулі, що спрощує структуру коду та полегшує його підтримку. JavaScript забезпечує взаємодію користувача з каталогом фільмів та обробку подій, які відбуваються під час роботи з веб-сторінкою [9].

Оскільки дана система є відносно простою інформаційною веб-системою, для її реалізації не використовуються складні серверні технології або бази даних. Інформація про фільми може зберігатися безпосередньо у структурі веб-сторінки або у JavaScript-файлах. Такий підхід є достатнім для демонстраційного або навчального проєкту, який має на меті показати принципи роботи веб-каталогу фільмів.

Для розробки сайту також використовуються стандартні інструменти веб-розробки, зокрема текстові редактори коду та веб-браузери для тестування роботи сайту. Веб-браузер виконує роль середовища виконання веб-застосунку, оскільки саме у браузері відображаються HTML-сторінки, застосовуються стилі CSS та виконується код JavaScript (рис. 2.5) .



Рис. 2.5 - Технологічний стек системи

Таким чином, для реалізації системи каталогу фільмів було використано базові веб-технології HTML, CSS та JavaScript. Використання цих технологій дозволяє створити простий, зрозумілий та функціональний веб-застосунок, який

забезпечує перегляд інформації про фільми та взаємодію користувача з каталогом [9].

3. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ

3.1 Загальна структура застосунку

Під час розробки веб-застосунку важливим етапом є визначення його загальної структури. Структура застосунку визначає організацію файлів, їх взаємозв’язок та роль кожного компонента у роботі системи. Правильна організація структури проєкту дозволяє зробити код більш зрозумілим, спрощує процес розробки та подальшу підтримку програмного забезпечення.

Розроблений веб-застосунок каталогу фільмів реалізований за допомогою базових веб-технологій: HTML, CSS та JavaScript. Основна логіка роботи застосунку виконується на стороні клієнта, тобто без використання серверної частини. Усі компоненти системи представлені у вигляді окремих файлів, кожен з яких виконує певну функцію.

Основним файлом веб-застосунку є index.html, який містить структуру веб-сторінки. У цьому файлі визначаються основні елементи інтерфейсу користувача, такі як заголовки, блоки з інформацією про фільми, поля пошуку, кнопки та інші елементи навігації. HTML визначає логічну структуру сторінки та забезпечує відображення інформації у браузері користувача.

Для оформлення зовнішнього вигляду веб-сторінки використовується CSS-файл, який містить стилі для всіх елементів сторінки. CSS дозволяє змінювати кольори, шрифти, розміри елементів, відступи та розташування блоків на сторінці. Завдяки використанню стилів інтерфейс сайту стає більш зручним та привабливим для користувача.

Для реалізації логіки роботи застосунку використовуються два файли JavaScript: app.js та data.js.

Файл data.js використовується для зберігання інформації про фільми. У ньому можуть міститися дані про назву фільму, рік випуску, жанр, опис та інші

характеристики. Таким чином, цей файл виконує роль простого джерела даних для застосунку.

Файл app.js відповідає за основну логіку роботи системи. У ньому реалізовано функції, які забезпечують відображення списку фільмів на сторінці, обробку дій користувача, виконання пошуку або інших інтерактивних операцій. Файл app.js взаємодіє з даними, що зберігаються у файлі data.js, та забезпечує їх відображення у веб-інтерфейсі.

Такий підхід до організації структури проєкту дозволяє розділити дані та програмну логіку системи. Це робить код більш зрозумілим та зручним для подальшої модифікації. Наприклад, у разі додавання нових фільмів достатньо змінити файл data.js, не змінюючи логіку роботи застосунку.

Під час відкриття веб-сторінки у браузері спочатку завантажується HTML-файл, після чого підключаються файл стилів CSS та JavaScript-файли. Браузер виконує JavaScript-код, який обробляє дані та формує відображення інформації про фільми на сторінці. У результаті користувач отримує повноцінний інтерфейс для перегляду каталогу фільмів [10].

Таким чином, структура веб-застосунку каталогу фільмів складається з HTML-файлу, CSS-файлу та двох JavaScript-файлів — app.js та data.js. Така структура є простою, зрозумілою та достатньою для реалізації основного функціоналу веб-каталогу.

3.2 Реалізація клієнтської частини (Frontend)

Клієнтська частина веб-застосунку, або Frontend, відповідає за взаємодію користувача з системою та відображення інформації у браузері. Саме ця частина забезпечує інтерфейс користувача, через який здійснюється перегляд інформації про фільми, пошук та навігація по каталогу. Реалізація клієнтської частини є важливим етапом розробки веб-застосунку, оскільки від її якості залежить зручність використання системи.

У розробленому веб-застосунку клієнтська частина реалізована за допомогою технологій HTML, CSS та JavaScript. Ці технології дозволяють створити повноцінний інтерфейс користувача, забезпечити його стилізацію та додати інтерактивні елементи.

Основою клієнтської частини є HTML-сторінка, яка містить структуру веб-застосунку. У HTML-документі розміщуються основні елементи інтерфейсу, зокрема заголовок сторінки, блоки для відображення списку фільмів, поле для пошуку та інші елементи навігації. HTML визначає розташування елементів на сторінці та забезпечує відображення контенту у веб-браузері.

Для оформлення сторінки використовується CSS, який відповідає за стилізацію елементів інтерфейсу. За допомогою CSS задаються кольори, шрифти, розміри блоків, відступи та інші параметри зовнішнього вигляду сторінки. Використання стилів дозволяє створити зручний та зрозумілий інтерфейс, що покращує взаємодію користувача з системою.

Інтерактивність веб-застосунку забезпечується за допомогою JavaScript. У даному проєкті використовується два JavaScript-файли: app.js та data.js. Файл data.js містить інформацію про фільми, яка використовується для формування каталогу. У цьому файлі можуть зберігатися дані про назву фільму, жанр, рік випуску та короткий опис.

Файл app.js реалізує основну логіку роботи інтерфейсу. У ньому створені функції, які відповідають за відображення списку фільмів на сторінці, обробку дій користувача та взаємодію з даними. Наприклад, JavaScript може використовуватися для формування списку фільмів на основі даних із файлу data.js, а також для обробки подій, таких як введення тексту у поле пошуку або натискання кнопок.

Після завантаження веб-сторінки браузер виконує JavaScript-код, який зчитує дані про фільми та формує відповідні елементи на сторінці. Таким чином створюється динамічний список фільмів, який користувач може переглядати у каталозі.

Перевагою реалізації клієнтської частини за допомогою JavaScript є

можливість змінювати вміст сторінки без її повного перезавантаження. Це дозволяє зробити веб-застосунок більш зручним та швидким у використанні. Користувач може взаємодіяти з каталогом фільмів, переглядати інформацію та виконувати пошук без необхідності постійного оновлення сторінки [11].

Таким чином, клієнтська частина веб-застосунку каталогу фільмів реалізована за допомогою HTML, CSS та JavaScript. HTML забезпечує структуру сторінки, CSS відповідає за її оформлення, а JavaScript реалізує інтерактивність та обробку даних. Такий підхід дозволяє створити зручний веб-інтерфейс для перегляду каталогу фільмів.

3.3 Реалізація бази даних

У більшості сучасних веб-застосунків інформація зберігається у спеціальних системах керування базами даних, які дозволяють ефективно працювати з великими обсягами інформації. Проте у невеликих або навчальних веб-проєктах інколи використовується спрощений підхід до зберігання даних, коли інформація зберігається безпосередньо у файлах програми. Такий підхід дозволяє спростити архітектуру системи та зосередитися на реалізації основного функціоналу веб-застосунку.

У розробленому веб-застосунку каталогу фільмів повноцінна система керування базами даних не використовується. Замість цього дані про фільми зберігаються у файлі data.js, який містить структуровану інформацію про кожен фільм. Цей файл виконує роль простого сховища даних, яке використовується клієнтською частиною застосунку для відображення інформації на веб-сторінці.

Інформація про фільми у файлі data.js представлена у вигляді масиву об’єктів JavaScript. Кожен об’єкт у цьому масиві містить набір властивостей, які описують конкретний фільм. До таких властивостей можуть належати назва фільму, рік випуску, жанр, короткий опис або інші характеристики. Така

структура дозволяє зручно організувати інформацію та легко працювати з нею за допомогою JavaScript.

Після завантаження веб-сторінки JavaScript-файл app.js отримує доступ до даних, що зберігаються у файлі data.js. Далі ці дані використовуються для формування елементів веб-сторінки, які відображаються у браузері користувача. Таким чином, інформація про фільми динамічно додається на сторінку під час виконання JavaScript-коду.

Використання такого підходу має кілька переваг. По-перше, структура системи залишається простою, оскільки відсутня необхідність налаштовувати сервер або базу даних. По-друге, дані легко змінювати або доповнювати, редагуючи файл data.js. По-третє, такий підхід добре підходить для навчальних проєктів, у яких головною метою є демонстрація принципів роботи веб-застосунків.

Разом з тим слід зазначити, що подібний спосіб зберігання інформації має певні обмеження. Наприклад, він не підходить для великих інформаційних систем або веб-сайтів з великою кількістю користувачів. У таких випадках зазвичай використовуються повноцінні системи керування базами даних, які забезпечують надійне зберігання та обробку інформації.

Таким чином, у розробленому веб-застосунку каталогу фільмів дані зберігаються у JavaScript-файлі data.js у вигляді масиву об’єктів. Цей файл виконує функцію простого сховища інформації та використовується клієнтською частиною системи для формування каталогу фільмів на веб-сторінці [12].

3.4 Реалізація основного функціоналу (пошук, перегляд, фільтрація)

Основною метою розробленого веб-застосунку є забезпечення зручного перегляду інформації про фільми. Для цього у системі реалізовано базовий

функціонал, який дозволяє користувачам переглядати список фільмів, отримувати інформацію про них, а також виконувати пошук або фільтрацію даних.

Однією з основних функцій системи є перегляд каталогу фільмів. Після відкриття веб-сторінки користувач бачить список фільмів, який формується на основі даних, що зберігаються у файлі data.js. JavaScript-код у файлі app.js обробляє ці дані та створює відповідні елементи HTML, які відображаються на сторінці. Таким чином користувач може переглядати перелік доступних фільмів у каталозі.

Ще однією важливою функцією системи є пошук фільмів. Пошук дозволяє користувачам швидко знайти потрібний фільм за його назвою. Для реалізації цієї функції використовується JavaScript, який обробляє введений користувачем текст у полі пошуку та порівнює його з назвами фільмів у масиві даних. Після цього на сторінці відображаються лише ті фільми, які відповідають введеним параметрам.

Крім пошуку, у системі може бути реалізована фільтрація фільмів. Фільтрація дозволяє відображати лише ті фільми, які відповідають певним умовам, наприклад певному жанру або року випуску. Для цього JavaScript виконує перевірку даних у масиві фільмів та формує новий список елементів, які відображаються на сторінці.

Важливим елементом функціоналу системи є також динамічне оновлення інформації на сторінці. Завдяки використанню JavaScript веб-сторінка може змінювати свій вміст без повного перезавантаження. Це робить роботу користувача з системою більш зручною та швидкою.

Реалізація основного функціоналу здійснюється у файлі app.js, який містить функції для обробки даних, формування елементів інтерфейсу та взаємодії з користувачем. Файл data.js використовується як джерело даних, а HTML-сторінка відображає результати роботи JavaScript-коду.

Таким чином, основний функціонал веб-застосунку каталогу фільмів включає перегляд списку фільмів, пошук та фільтрацію інформації. Використання JavaScript дозволяє реалізувати ці функції без використання серверної частини та забезпечити зручну взаємодію користувача з системою [12].

4. АНАЛІЗ WEB 3.0 АРХІТЕКТУРИ В ПРОЄКТІ

4.1 Відповідність системи принципам Web 3.0

Web 3.0 є новим етапом розвитку Інтернету, який характеризується використанням децентралізованих технологій, підвищеною безпекою даних, прозорістю процесів та більшою автономністю користувачів. Основною ідеєю Web 3.0 є створення систем, у яких дані не контролюються одним центральним сервером, а можуть зберігатися та оброблятися у розподіленому середовищі. Такий підхід дозволяє підвищити надійність систем, забезпечити більшу відкритість та зменшити залежність від централізованих сервісів [13].

У рамках даної роботи було розроблено веб-застосунок каталогу фільмів, який демонструє принципи створення інформаційних веб-систем. Хоча реалізований застосунок є відносно простим, деякі його характеристики можуть бути розглянуті у контексті підходів Web 3.0.

Однією з характерних особливостей Web 3.0 є використання сучасних веб-технологій для створення динамічних та інтерактивних застосунків. У розробленій системі для реалізації інтерфейсу та логіки роботи застосовано HTML, CSS та JavaScript. Ці технології дозволяють створювати інтерактивні веб-інтерфейси, які забезпечують зручну взаємодію користувача з інформаційною системою.

Ще однією важливою характеристикою Web 3.0 є структурованість та семантичність даних. У розробленому веб-застосунку інформація про фільми зберігається у структурованому вигляді у файлі data.js. Дані організовані у вигляді масиву об’єктів, де кожен об’єкт містить властивості, що описують певний фільм. Така структура даних дозволяє ефективно виконувати пошук, фільтрацію та відображення інформації на веб-сторінці.

Також важливою характеристикою сучасних веб-систем є орієнтація на користувача. У розробленому застосунку реалізовано базові можливості взаємодії з користувачем, такі як перегляд списку фільмів, пошук інформації та відображення результатів без перезавантаження сторінки. Це забезпечує зручну та швидку роботу з каталогом фільмів.

Проте слід зазначити, що у даному проєкті не використовуються повноцінні технології Web 3.0, такі як блокчейн, смарт-контракти або розподілені мережі зберігання даних. Система працює як клієнтський веб-застосунок, у якому дані зберігаються локально у JavaScript-файлі. Незважаючи на це, структура застосунку може бути розширена у майбутньому для інтеграції з сучасними технологіями Web 3.0.

Таким чином, розроблений веб-застосунок каталогу фільмів частково відповідає сучасним підходам веб-розробки та демонструє принципи створення інтерактивних інформаційних систем. У подальшому система може бути модернізована шляхом інтеграції з розподіленими технологіями та іншими елементами архітектури Web 3.0 [13].

4.2 Використання розподілених технологій (або можливість їх інтеграції)

Однією з ключових особливостей архітектури Web 3.0 є використання розподілених технологій, які дозволяють зберігати та обробляти інформацію без використання централізованих серверів. У таких системах дані можуть зберігатися на великій кількості вузлів мережі, що підвищує надійність, доступність та безпеку інформації. Розподілені технології широко використовуються у сучасних децентралізованих застосунках та цифрових платформах [14].

У розробленому веб-застосунку каталогу фільмів розподілені технології безпосередньо не використовуються. Система реалізована у вигляді клієнтського

веб-застосунку, де всі дані зберігаються у JavaScript-файлі data.js, а логіка роботи реалізована у файлі app.js. Уся обробка інформації відбувається без використання серверної інфраструктури або розподілених мереж.

Незважаючи на це, архітектура розробленого застосунку може бути розширена для інтеграції з сучасними розподіленими технологіями. Наприклад, дані про фільми можуть зберігатися у розподілених файлових системах, таких як IPFS (InterPlanetary File System). У такому випадку інформація про фільми буде зберігатися не на одному сервері, а на декількох вузлах мережі, що забезпечить більшу надійність та доступність даних.

Ще одним можливим напрямком розвитку є використання блокчейн-технологій. Блокчейн може застосовуватися для зберігання інформації про рейтинги фільмів, оцінки користувачів або історію змін у каталозі. Такий підхід дозволяє забезпечити прозорість та незмінність інформації, оскільки дані у блокчейні не можуть бути змінені без підтвердження мережі.

Також можливе використання смарт-контрактів, які дозволяють автоматизувати виконання певних операцій у системі. Наприклад, смарт-контракти можуть використовуватися для реалізації системи голосування або рейтингу фільмів, де оцінки користувачів зберігатимуться у децентралізованій мережі.

Інтеграція з розподіленими технологіями дозволить значно розширити функціональні можливості веб-застосунку та зробити його ближчим до концепції Web 3.0. Зокрема, це дозволить підвищити безпеку даних, забезпечити децентралізоване зберігання інформації та створити більш прозору систему управління контентом.



Рис. 4.2 - Можлива інтеграція розподілених технологій у систему

Таким чином, хоча у поточній реалізації веб-застосунку каталогу фільмів розподілені технології не використовуються, структура системи дозволяє у майбутньому інтегрувати такі рішення. Це може бути реалізовано шляхом використання розподілених систем зберігання даних, блокчейн-платформ або інших технологій, що підтримують архітектуру Web 3.0 [14].

4.3 Переваги та недоліки реалізованого підходу

Під час розробки веб-застосунку каталогу фільмів було використано спрощений підхід до побудови архітектури системи. Застосунок реалізований як клієнтська веб-система, що працює у браузері користувача та використовує технології HTML, CSS та JavaScript. Такий підхід має як певні переваги, так і деякі обмеження, які слід враховувати при створенні інформаційних систем.

Однією з основних переваг реалізованого підходу є простота розробки та

використання. Оскільки система не використовує серверну частину або складну інфраструктуру, її розробка є відносно швидкою та зрозумілою. Для створення застосунку достатньо базових знань веб-технологій, що значно спрощує процес реалізації проєкту.

Ще однією важливою перевагою є відсутність необхідності у серверному обладнанні. Усі компоненти системи працюють на стороні клієнта, тобто у браузері користувача. Це дозволяє запускати застосунок без встановлення додаткового програмного забезпечення або налаштування серверного середовища.

Також слід відзначити швидкість роботи інтерфейсу. Оскільки дані про фільми зберігаються локально у файлі data.js, доступ до них відбувається миттєво. JavaScript може швидко обробляти інформацію та відображати результати на сторінці без необхідності надсилання запитів до сервера. Це забезпечує швидку взаємодію користувача з веб-застосунком.

Разом з тим реалізований підхід має і певні недоліки. Одним із головних недоліків є обмежені можливості зберігання та обробки даних. Оскільки у системі не використовується база даних, усі дані зберігаються у JavaScript-файлі. Це ускладнює роботу з великими обсягами інформації та не дозволяє динамічно змінювати дані під час роботи системи.

Ще одним недоліком є відсутність серверної логіки, яка могла б забезпечити додаткову обробку даних, автентифікацію користувачів або інтеграцію з іншими сервісами. У більш складних веб-застосунках серверна частина відіграє важливу роль у забезпеченні функціональності та безпеки системи.

Також варто зазначити, що система у поточному вигляді не підтримує багатокористувацьку взаємодію. Оскільки дані зберігаються локально у файлі програми, користувачі не можуть додавати нові фільми або змінювати інформацію у каталозі.

Незважаючи на зазначені обмеження, обраний підхід є цілком виправданим для навчального проєкту або невеликого інформаційного веб-застосунку. Він

дозволяє продемонструвати принципи побудови веб-інтерфейсу, організації даних та реалізації основного функціоналу системи [15].



Рис. 4.3 - Переваги та недоліки реалізованого підходу

Таким чином, розроблений веб-застосунок каталогу фільмів має низку переваг, серед яких простота реалізації, швидкість роботи та відсутність складної інфраструктури. Водночас система має певні обмеження, пов’язані з відсутністю серверної частини та повноцінної бази даних, що обмежує можливості її масштабування та подальшого розвитку.

4.4 Перспективи розвитку системи

Подальший розвиток веб-застосунку каталогу фільмів може бути спрямований на розширення функціональних можливостей системи, покращення архітектури та інтеграцію сучасних технологій. Незважаючи на те, що

розроблений застосунок є навчальним проєктом, його структура може бути використана як основа для створення більш складних інформаційних систем.

Одним із можливих напрямків розвитку є впровадження серверної частини застосунку. Додавання серверної логіки дозволить забезпечити обробку запитів користувачів, зберігання інформації у базі даних та реалізацію більш складного функціоналу. Для цього можуть використовуватися сучасні серверні технології, наприклад Node.js, Python або інші платформи для розробки веб-застосунків.

Ще одним важливим напрямком розвитку є використання повноцінної бази даних для зберігання інформації про фільми. У такому випадку дані не будуть зберігатися у JavaScript-файлі, а будуть розміщені у спеціальній системі керування базами даних. Це дозволить ефективно працювати з великими обсягами інформації, додавати нові записи та редагувати наявні дані.

Також перспективним напрямком є розширення функціоналу користувацького інтерфейсу. Наприклад, у систему можна додати можливість сортування фільмів за різними параметрами, створення списків улюблених фільмів, а також відображення більш детальної інформації про кожен фільм. Це дозволить зробити веб-застосунок більш зручним та функціональним для користувачів.

Окрему увагу можна приділити інтеграції з технологіями Web 3.0. У майбутньому система може використовувати розподілені мережі зберігання даних, блокчейн або інші децентралізовані технології. Наприклад, інформація про рейтинги фільмів або оцінки користувачів може зберігатися у блокчейні, що забезпечить прозорість та незмінність даних.

Крім того, можливим є підключення зовнішніх API, які надають інформацію про фільми. Наприклад, застосунок може отримувати дані з відкритих кіно-баз, що дозволить автоматично оновлювати інформацію та розширити каталог фільмів.

Таким чином, розроблений веб-застосунок має значний потенціал для подальшого розвитку. Удосконалення архітектури системи, використання баз даних, розширення функціоналу та інтеграція з сучасними технологіями

дозволять створити більш повноцінну інформаційну систему каталогу фільмів, що відповідатиме сучасним вимогам веб-розробки [16].

ВИСНОВКИ

У ході виконання даної роботи було досліджено особливості архітектури Web 3.0 систем у розподілених середовищах, а також розроблено веб-застосунок каталогу фільмів. Під час роботи було проаналізовано основні принципи побудови сучасних веб-систем, особливості їх архітектури та можливості використання сучасних веб-технологій для створення інформаційних застосунків.

У теоретичній частині роботи було розглянуто поняття Web 3.0, основні принципи його функціонування та роль розподілених технологій у сучасних інформаційних системах. Було встановлено, що Web 3.0 орієнтований на децентралізацію, підвищення прозорості обробки даних, а також використання нових технологій, таких як блокчейн, розподілені мережі зберігання даних та смарт-контракти.

У процесі виконання роботи було проведено аналіз предметної області та досліджено існуючі аналоги систем каталогів фільмів. На основі проведеного аналізу було сформовано основні вимоги до розроблюваної системи, визначено її функціональні можливості та спроєктовано загальну архітектуру веб-застосунку.

Практична частина роботи була присвячена розробці веб-застосунку каталогу фільмів. Для реалізації системи було використано технології HTML, CSS та JavaScript, які дозволяють створювати сучасні інтерактивні веб-інтерфейси. Було реалізовано клієнтську частину застосунку, яка забезпечує відображення каталогу фільмів, пошук інформації та взаємодію користувача з системою.

У розробленому застосунку дані про фільми зберігаються у файлі data.js у вигляді структурованого масиву об’єктів. JavaScript використовується для обробки цих даних та формування елементів інтерфейсу на веб-сторінці. Такий підхід дозволяє створити простий та зрозумілий веб-застосунок без використання серверної частини або повноцінної бази даних.

Також у роботі було проаналізовано відповідність розробленої системи принципам Web 3.0. Було встановлено, що у поточній реалізації система не

використовує повноцінні розподілені технології, проте її архітектура може бути розширена для інтеграції з такими рішеннями у майбутньому. Зокрема, можливим є використання блокчейн-технологій, розподілених систем зберігання даних та інших сучасних підходів.

Крім того, було проаналізовано переваги та недоліки реалізованого підходу. До основних переваг можна віднести простоту реалізації, швидкість роботи та відсутність складної серверної інфраструктури. Водночас серед недоліків слід відзначити обмежені можливості масштабування системи, відсутність серверної частини та повноцінної бази даних.

Таким чином, у результаті виконання роботи було створено веб-застосунок каталогу фільмів, який демонструє принципи побудови сучасних веб-систем та використання клієнтських веб-технологій. Отримані результати можуть бути використані як основа для подальшого розвитку системи та інтеграції з більш складними технологіями Web 3.0.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. CCNA: Switching, Routing, and Wireless Essentials | Networking Academy. Networking Academy. URL: https://www.netacad.com/courses/networking/ccna-switching-routing-wireless-essentials (дата звернення: 06.05.2024).
2. Berners-Lee T., Hendler J., Lassila O. The Semantic Web // Scientific American. 2001. №5. С. 28–37.
3. Allemang D., Hendler J. Semantic Web for the Working Ontologist. 2nd ed. Amsterdam : Morgan Kaufmann, 2011. 384 p.
4. Wood D., Zaidman M., Ruth L., Hausenblas M. Linked Data: Structured Data on the Web. New York : Manning Publications, 2014. 312 p.
5. Shadbolt N., Hall W., Berners-Lee T. The Semantic Web Revisited // IEEE Intelligent Systems. 2006. Vol. 21. №3. P. 96–101.
6. Hitzler P., Krötzsch M., Rudolph S. Foundations of Semantic Web Technologies. Boca Raton : CRC Press, 2009. 455 p.
7. Noy N., McGuinness D. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford : Stanford University, 2001. 25 p.
8. Hendler J. Web 3.0 Emerging // Computer. 2009. Vol. 42. №1. P. 111–113.
9. Duckett J. HTML and CSS: Design and Build Websites. Indianapolis : John Wiley & Sons, 2011. 490 p.
10. Freeman E., Robson E. Head First HTML and CSS. Sebastopol : O'Reilly Media, 2012. 768 p.
11. Mozilla Developer Network. HTML: HyperText Markup Language [Електронний ресурс]. Режим доступу: https://developer.mozilla.org
12. Freeman E., Robson E. Head First HTML and CSS. Sebastopol : O'Reilly Media, 2012. 768 p.
13. Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol : O'Reilly Media, 2020. 704 p.
14. Narayanan A., Bonneau J., Felten E., Miller A., Goldfeder S. Bitcoin and Cryptocurrency Technologies. Princeton : Princeton University Press, 2016. 336 p.

15. Antonopoulos A. Mastering Bitcoin: Programming the Open Blockchain. 2nd ed. Sebastopol : O'Reilly Media, 2017. 416 p.

16. Duckett J. JavaScript and JQuery: Interactive Front-End Web Development. Indianapolis : John Wiley & Sons, 2014. 640 p.

17. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston : Addison-Wesley, 2003. 560 p.

18. Stallings W. Cryptography and Network Security: Principles and Practice. 7th ed. Boston : Pearson, 2017. 766 p.

19. W3C. World Wide Web Consortium [Електронний ресурс]. Режим доступу: https://www.w3.org

20. Mozilla Developer Network. JavaScript Guide [Електронний ресурс]. Режим доступу: https://developer.mozilla.org

21. Руденко В.Д., Синиця Ю.П. Основи комп'ютерних мереж. – Київ: Академія, 2012. – 320 с.

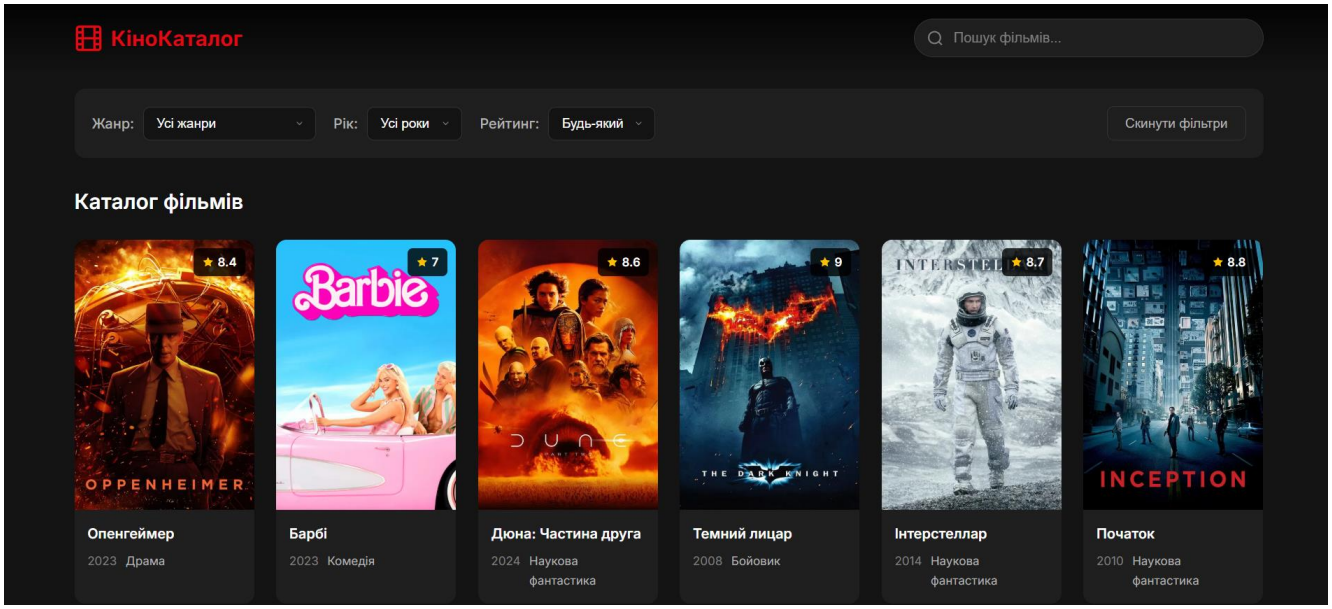
22. Ситник В.Ф., Писаренко Т.П. Основи інформаційних систем. – Київ: КНЕУ, 2011. – 420 с.

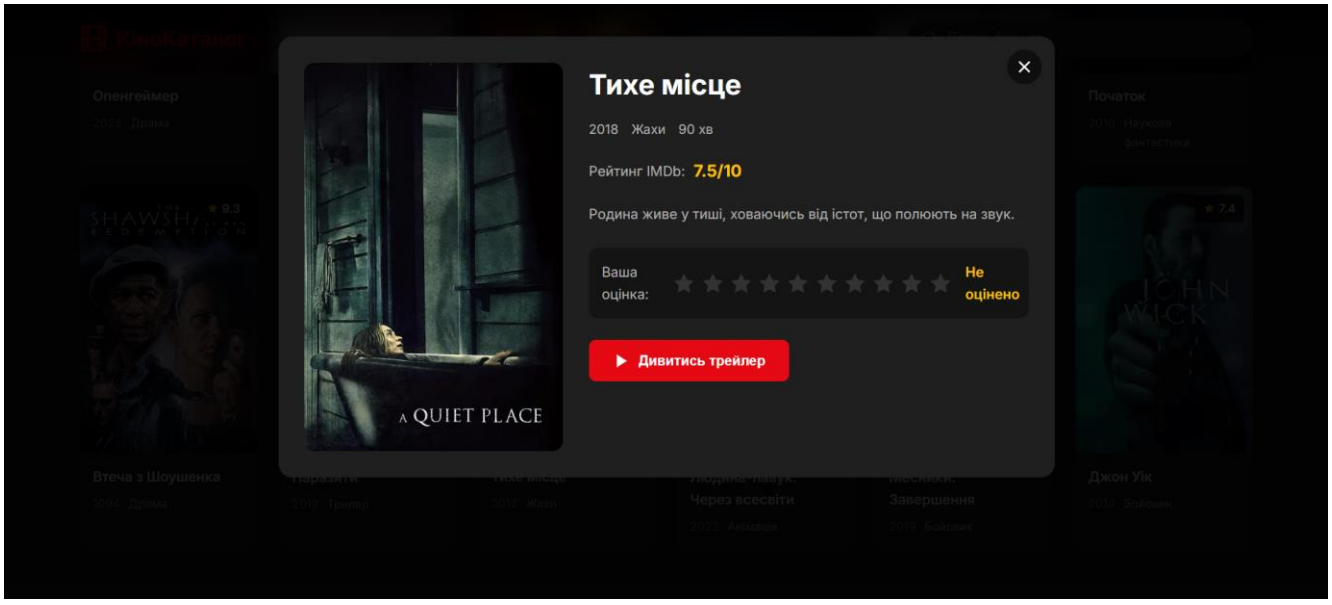
23. Глинський Я.М. Практикум з інформатики. – Львів: СПД Глинський, 2015. – 224 с.

24. Ethereum Foundation. Introduction to Web3 [Електронний ресурс]. – Режим доступу: https://ethereum.org

Додаток А

Інтерфейс системи





Додаток Б

Структура бази даних

```
-------------------------
        Movies
-------------------------
id (INT, PK)
title (VARCHAR)
genre (VARCHAR)
year (INT)
rating (FLOAT)
description (TEXT)
poster (VARCHAR)
-------------------------
```

Додаток В

Приклад логічної топології мережі

styles.css

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

:root {
    --primary: #e50914;
    --primary-hover: #f40612;
    --bg-dark: #141414;
    --bg-card: #1f1f1f;
    --bg-card-hover: #2a2a2a;
    --text-primary: #ffffff;
    --text-secondary: #b3b3b3;
    --text-muted: #808080;
    --border-color: #333;
    --star-color: #ffc107;
    --shadow: 0 4px 20px rgba(0, 0, 0, 0.5);
    --transition: all 0.3s ease;
}

body {
    font-family: 'Inter', -apple-system, BlinkMacSystemFont, sans-serif;
    background-color: var(--bg-dark);
    color: var(--text-primary);
    line-height: 1.6;
    min-height: 100vh;
}

.container {
    max-width: 1400px;
    margin: 0 auto;
    padding: 0 20px;
}

/* Header */
.header {
    background: linear-gradient(180deg, rgba(0,0,0,0.9) 0%, transparent 100%);
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    z-index: 100;
    padding: 20px 0;
    backdrop-filter: blur(10px);
}
```

app.js > ...

```js
// Глобальні змінні
let currentMovie = null;
let userRatings = {};

// Ініціалізація
document.addEventListener('DOMContentLoaded', () => {
    loadUserRatings();
    renderMovies(moviesData);
    setupEventListeners();
});

// Завантаження оцінок з LocalStorage
function loadUserRatings() {
    const saved = localStorage.getItem('movieRatings');
    if (saved) userRatings = JSON.parse(saved);
}

// Збереження оцінок
function saveUserRatings() {
    localStorage.setItem('movieRatings', JSON.stringify(userRatings));
}

// Рендеринг фільмів
function renderMovies(movies) {
    const grid = document.getElementById('moviesGrid');
    const noResults = document.getElementById('noResults');

    if (movies.length === 0) {
        grid.innerHTML = '';
        noResults.classList.remove('hidden');
        return;
    }

    noResults.classList.add('hidden');

    grid.innerHTML = movies.map(movie => {
        const userRating = userRatings[movie.id];
        const userRatingHtml = userRating ? `
            <div class="movie-user-rating">
                <svg viewBox="0 0 24 24"><polygon points="12 2 15.09 8.26 22 9.27 17 14.14 18.18 21.02 12 17.77 5.82 21.02 7 14.14 2 9.27 8.91 8.26 12 2"/></svg>
                <span>Ваша: ${userRating}/10</span>
            </div>
        ` : '';
```

```json
// Movies Database - Movie Catalog
const moviesData = [
  {
    id: 1,
    title: "Оппенгеймер",
    titleEn: "Oppenheimer",
    year: 2023,
    genre: "drama",
    genreUa: "Драма",
    duration: "180 хв",
    rating: 8.4,
    poster: "https://image.tmdb.org/t/p/w500/8Gxv8gSFCU0XGDykEGv7zR1n2ua.jpg",
    description:
      "Історія американського фізика Роберта Оппенгеймера та його ролі у розробці атомної бомби.",
    trailer: "https://www.youtube.com/embed/bK6ldnjE3Y0",
  },
  {
    id: 2,
    title: "Барбі",
    titleEn: "Barbie",
    year: 2023,
    genre: "comedy",
    genreUa: "Комедія",
    duration: "114 хв",
    rating: 7.0,
    poster: "https://image.tmdb.org/t/p/w500/iuFNMS8U5cb6xfzi51Dbkovj7vM.jpg",
    description:
      "Барбі та Кен насолоджуються життям у Барбіленді, поки не потрапляють у реальний світ.",
    trailer: "https://www.youtube.com/embed/pBk4NYhWNMM",
  },
  {
    id: 3,
    title: "Дюна: Частина друга",
    titleEn: "Dune: Part Two",
    year: 2024,
    genre: "scifi",
    genreUa: "Наукова фантастика",
    duration: "166 хв",
    rating: 8.6,
    poster: "https://image.tmdb.org/t/p/w500/8b8R8188Qje9dn9OE8PY05Nxl1X.jpg",
    description: "Пол Атрідес об'єднується з фременами на шляху помсти.",
    trailer: "https://www.youtube.com/embed/Way9Dexny3w",
  },
```

ДОДАТОК Г. ПРЕЗЕНТАЦІЯ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА
на тему: «Архітектура системи Web 3.0 у розподіленому середовищі »

Виконав студент групи КІД-42
Магуський Владислав Леонідович
Керівник кваліфікаційної роботи:
Лащевська Наталія Олександрівна
завідувач кафедри КІ

Київ 2026

Слайд Г.1 – Слайд 1



Слайд Г.2 – Слайд 2

Мета та завдання

- Розробити веб-застосунок каталогу фільмів та проаналізувати використання принципів Web 3.0.
- дослідити технології Web 3.0
- проаналізувати існуючі системи
- спроєктувати структуру застосунку
- розробити веб-застосунок
- проаналізувати його архітектуру

3

Слайд Г.3 – Слайд 3



Рисунок 1.0

Web 3.0

- децентралізація
- розподілені системи
- семантичний веб
- блокчейн

4

Слайд Г.4 – Слайд 4



Рисунок 1.1

5

Слайд Г.5 – Слайд 5



Рисунок 1.2

6

Слайд Г.6 – Слайд 6

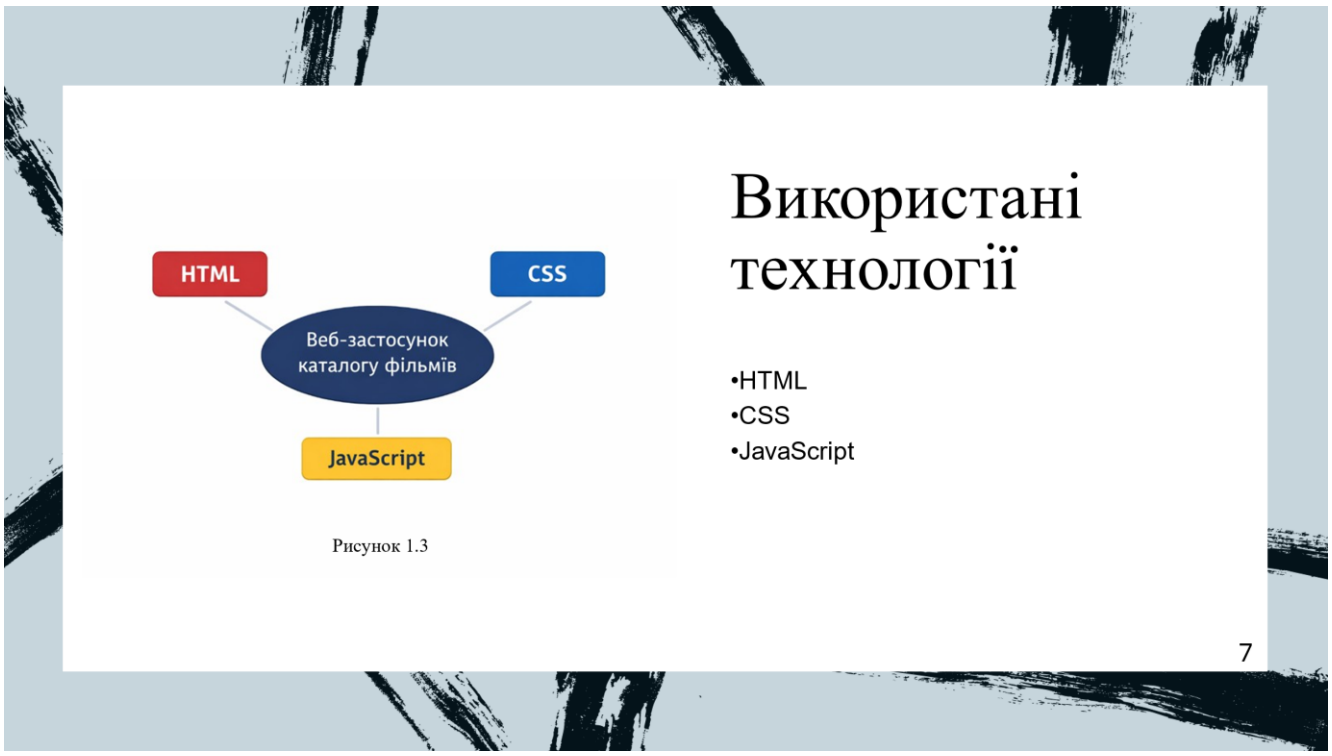


Рисунок 1.3

Слайд Г.7 – Слайд 7

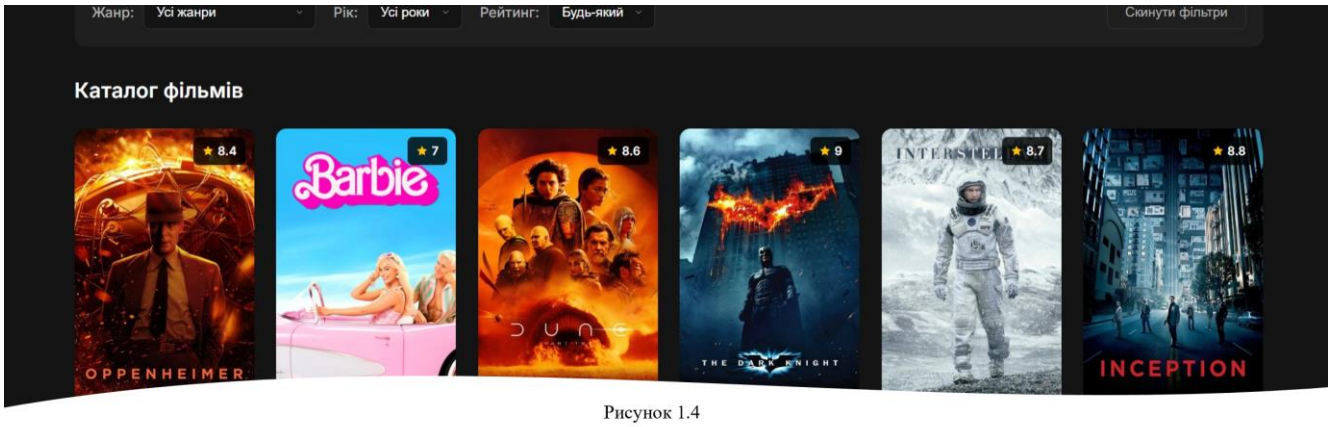


Рисунок 1.4

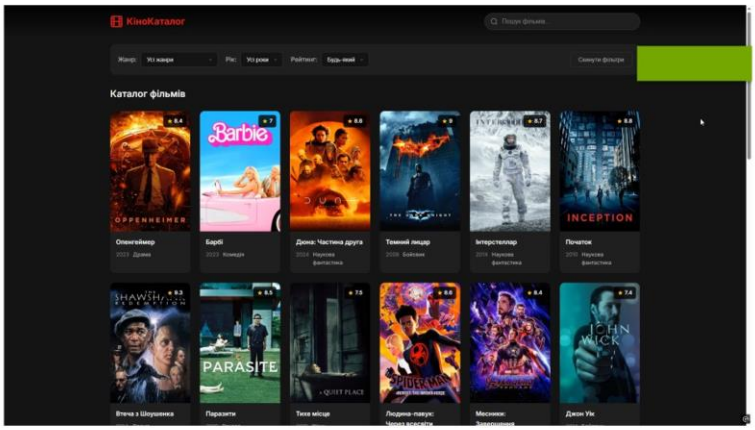
Функціонал системи

- перегляд каталогу
- пошук фільмів
- фільтрація
- відображення інформації

8

Слайд Г.8 – Слайд 8

Демонстрація роботи сайту



Демонстрація 1 — Робота основного функціоналу сайту

9

Слайд Г.9 – Слайд 9

Висновки

- У результаті виконання роботи було досліджено архітектуру Web 3.0 та розроблено веб-застосунок каталогу фільмів. Реалізована система забезпечує зручний перегляд та пошук інформації про фільми.
- Розглянуто основні принципи Web 3.0: децентралізація, прозорість обробки даних, використання блокчейну, розподілених мереж зберігання даних та смарт-контрактів.
- Проведено аналіз предметної області та існуючих аналогів систем каталогів фільмів, на основі чого сформовано вимоги та спроєктовано архітектуру застосунку.
- Для реалізації використано технології HTML, CSS та JavaScript. Дані про фільми зберігаються у файлі data.js у вигляді структурованого масиву об'єктів, що забезпечує роботу без серверної частини.
- До переваг системи належать: простота реалізації, швидкість роботи та відсутність складної серверної інфраструктури. Серед недоліків — обмежені можливості масштабування та відсутність повноцінної бази даних.

10

Слайд Г.10 – Слайд 10