МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
Савченко А.С.
«___»_____2020 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)
ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"МАГІСТР"

Тема: «Web-додаток «Рекомендаційна система»»

Виконав: Середа Владислав Валерійович

Керівник: Воронін Альберт Миколайович

Нормоконтролер з ЄСКД (ЄСПД): Райчев І.Е.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕСИТЕТ

Факультет кібербезпеки, комп’ютерної та програмної інженерії

Кафедра комп’ютерних інформаційних технологій

Спеціальність 122 “Комп’ютерні науки та інформаційні технології”

Спеціалізація «Інформаційні управляючі системи та технології (за галузями)»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Савченко А.С.

“ ” _____2019р.

З А В Д А Н Н Я

на виконання дипломної роботи студента

Середи Владислава Валерійовича

(прізвище, ім’я, по батькові)

1. Тема проекту (роботи): «Web-додаток «Рекомендаційна система»» затверджена наказом ректора №2175/ст. від 14.10.2019р..
2. Термін виконання проекту (роботи): з 14.10.2019р. по 09.02.2020р.
3. Вихідні данні до проекту (роботи): Web-додаток, системи рекомендацій
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): вступ, аналітичний огляд і постановка завдання, висновок.
5. Перелік обов’язкового графічного матеріалу:

Схематичне зображення відображення сторінки Web-додатку на різних типах дисплею;

Приклад HTML коду;

Схеми принципу роботи різних рекомендаційних систем.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1.	Проаналізувати літературу та джерела за темою дипломного проекту.	13.10.19– 20.10.19	
2.	Розроблення та затвердження плану дипломного проекту.	21.10.19– 22.10.19	
3.	Провести консультації з науковим керівником щодо створення першого розділу.	23.10.19 – 27.10.19	
4.	Розробка розділу 1: Особливості розробки Web-додатку	30.10.19 – 11.11.19	
5.	Розробка розділу 2: Аналіз технологій для створення Web-додатку	12.11.19 – 22.11.19	
6.	Розробка розділу 3: Аналіз типів рекомендаційних систем	23.11.19 – 02.12.19	
7.	Розробка розділу 4: Створення Web-додатку «Рекомендаційна система»	02.12.19 – 22.12.19	
8.	Висновки та оформлення пояснювальної записки дипломного проекту.	25.12.19 – 29.12.19	
9.	Підписання необхідних документів у встановленому порядку.	15.01.20-19.01.20	
10.	Підготовка до захисту та попередній захист дипломного проекту на випусковій кафедрі дипломного проекту	22.01.20 – 31.01.20	

7. Дата видачі завдання: 13.10.2019р.

Керівник дипломного проекту _____ Воронін А.М.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Середа В.В.
(підпис випускника) (П.І.Б.).

РЕФЕРАТ

Пояснювальна записка до дипломного проекту роботи «Web-додаток «Рекомендаційна система»» викладена на 98 с., містить 57 рис., 14 літературних джерел.

Ключові слова: WEB-ДОДАТОК, РОЗРОБКА, СИСТЕМА, РЕКОМЕНДАЦІЙ, КОРИСТУВАЧ

Об'єкт дослідження: технологія розробки Web-додатку

Предмет дослідження: технології для реалізації рекомендаційної системи

Мета роботи: створити Web-додаток, здатний підібрати користувачу якісний контент.

Методи дослідження: аналіз програмного забезпечення для створення і проектування Web-додатку

Отримані результати: реалізовано і введено в експлуатацію Web-додаток, який дозволяє зручно знаходити контент, відповідно до вподобань клієнта.

Результати дипломної роботи планується використовувати для подальшої розробки Web-додатків із більш складною системою рекомендацій.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

БД – база даних;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

HTML – HyperText Markup Language;

CSS – Cascade Style Sheets;

HTTP – HyperText Transfer Protocol;

JS – JavaScript;

HTTP – протокол передачі даних;

API – Application Programming Interface;

IDE – Інтегроване середовище розробки (Integrated Development Environment);

TF-IDF – статистичний показник, що використовується для оцінки важливості слів у контексті документа

IDF – інверсія частоти, з якою слово зустрічається в документах колекції

ВСТУП

Рекомендаційна система представляє собою систему, яка здатна передбачати речі, які могли б знадобитись конкретному користувачу у майбутньому. Одна із головних причин чому рекомендаційна система потрібна у сучасному суспільстві – це звичайно ж можливість зекономити час і надати людям більш простий вибір. Будь-який сервіс в інтернеті пропонує користувачу настільки широкий вибір контенту, що легко в ньому заплутатись. Коли існували фізичні магазини, то кількість товару в них була обмежена і була просто незрівнянно мала на фоні інтернету.

Наприклад, приходячи до звичайного магазину з фільмами на дисках, людині доводилось вибирати із декількох десятків варіантів. На просторах веб-сайту в інтернеті, кінофільмів може бути декілька десятків лише на одній сторінці. А загальна кількість може складати сотні і навіть тисячі. Таку кількість контенту потрібно якось структурувати, щоб зекономити час клієнта і надати йому якомога найкращі послуги. Одним із найпопулярніших дистрибюторів цифрового контенту являється онлайн-кінотеатр Netflix. Компанія починала із видачі кінофільмів напрокат за допомогою пошти, а зараз розповсюджує фільми з усього світу, та власні продукти за допомогою веб-сайта в мережі інтернет.

Це дозволило компанії досягти небувалих висот. Але з кожним роком база кінофільмів сервісу зростала і тому з’явилася необхідність створення рекомендаційної системи, котра змогла б допомогти клієнтам обрати найкращий кінофільм для перегляду в той чи інший момент часу.

Якщо користувачу складно розібратись в контенті який пропонує конкретний веб-сайт, то це не означає що в ньому занадто багато інформації, просто вона погано структурована.

Загалом існує два основних шляхи побудови рекомендаційної системи:

- Фільтрування на основі контенту;
- Фільтрування на основі вподобань користувача.

Рекомендаційна система представляє собою важливий інструмент, який є корисним, як для самих користувачів, так і для компанії, що надає контент. Користувачі можуть отримати цінну для них інформацію максимально швидко, а для бізнесу це можливість якісно представити свій продукт.

Для більш чіткого розуміння того, як працює рекомендаційна система, та чому вона важлива, буде проведено порівняння різних методів побудови такого фільтру. Завдяки цьому буде виявлено найбільш доцільний метод створення Web-додатку «Рекомендаційна система».

Розробка проекту складається не тільки з використання сучасних мов структуризації контенту та програмування, але також з дослідження великої кількості щодо побудови якісної рекомендаційної системи. Потрібно дослідити які переваги та недоліки містять методи створення подібного фільтра. Буде проведено порівняння шляхів реалізації системи та обрано найбільш доцільний метод для реалізації проекта.

РОЗДІЛ 1.

Аналіз особливостей розробки Web-додатку

1.1 Поняття «Web-додаток»

Web-додаток – це додаток, який отримує запит від користувача і виконує операцію, після чого відображення сторінки змінюється. Користуватись додатком можна в будь який момент часу, незалежно від пристрою за яким працює користувач. Незалежно від платформи на які відкрито додаток, контент має відображатися вірно і сайт повинен вчасно реагувати на дії користувача.

Головною відмінністю Web-додатку від Web-сайту являється принцип подачі контенту. В той час, коли звичайний Web-сайт надає користувачу можливість споглядати статичну інформацію, Web-додаток підлаштовується під потреби користувача, надаючи можливість взаємодіяти з функціональними аспектами сторінки.[Рис. 1.1]

Прикладами звичайних Web-сайтів є рекламні Web-сайти, новостні портали і т.д.. А ось Web-додатки, це онлайн-магазини, такі як Amazon, соціальні мережі Facebook, Twitter та розважальні портали, на кшталт Netflix, Hulu та багатьох інших.

Web-сайти побудовані на тому принципі, що користувач просто відвідує сторінку, зчитує з неї інформацію та закриває її. В той час як Web-додаток вимагає від користувача прямої взаємодії. Як тільки користувач виконує якусь дію на Web-сторінці, це створює запит, на який надходить відповідь від серверу до якого підключений Web-сайт.

Можна сказати що всі Web-додатки є Web-сайтами, та не всі Web-сайти є Web-додатками. Те що робить їх такими унікальними, так це той факт, що неможливо розробити Web-додаток з чистого листа самостійно. У будь-якому разі доведеться використовувати не тільки звичні мови розмітки та програмування, як HTML, CSS та JavaScript, але й цілі бібліотеки технологій від сторонніх розробників. Є велика кількість компаній, які сконцентровані на створені подібних технологій для стартапів і малого бізнесу.

Саме тому в сучасних інтернет-магазинах можна зустріти багато подібних каркасів для взаємодії з контентом на сайті. Розробники просто закупляють доступні на сьогоднішній день технології для функціонування Web-додатку у вигляді онлайн каталогу.

1.2 Переваги Web-додатку

Майже кожен сучасний Web-сайт[Рис. 1.2] можна назвати Web-додатком. У світі, де увага людей розсіяна по десяткам сторінок, які людина встигає переглянути за день, потрібно використовувати усі наявні можливості для того, щоб утримати користувача на конкретному ресурсі. Саме це і мотивує компанії створювати якомога більш функціональні сайти із використанням якомога цікавіших і сучасних технологій. Є одразу декілька переваг використання Web-додатку.

Перш за все – сучасні браузери дозволяють Web-сайтам зчитувати набагато більшу кількість дій користувача, ніж будь-коли до цього. У Web-додатка є можливість запропонувати користувачу приймати push-повідомлення, щоб той мав змогу відстежувати оновлення сервісу навіть в той час, коли сам сайт компанії закрито.

Якщо у компанії не вистачає коштів на те щоб створити повноцінний мобільний додаток, то сучасні технології Web-розробки дозволяють вирішити цю проблему настільки швидко, що в це складно повірити. Усе що колись було притаманно виключно мобільним додаткам, зараз може бути реалізовано через звичайний Web-браузер. Відмінності майже непомітні.

Прогресивні Web-додатки представляють собою ніщо інше, як повноцінні платформи для зв’язку із користувачами з усього світу. Як показала практика такої великої компанії як Alibaba, впровадження подібних технологій значно збільшує кількість потенціальних клієнтів і людей зацікавлених у Web-сайті.

Також Web-додатки прекрасно підходять для того щоб відслідковувати вподобання клієнтів і отримувати від них вірну інформацію про їх покупки. Для цього можна використовувати данні отримані із звичайного браузера, не обов’язково розробляти окремий додаток для конкретної платформи.

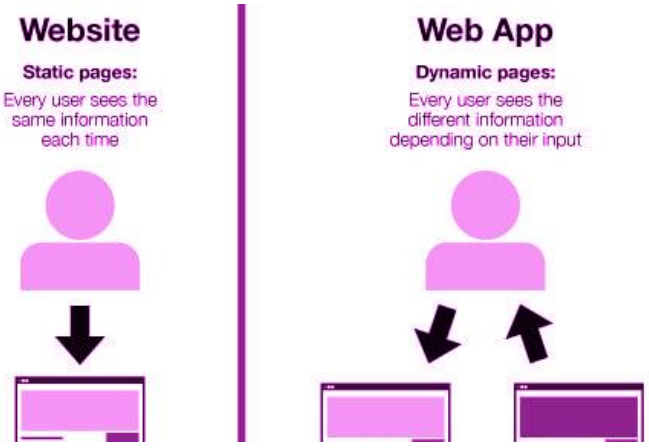

Рис. 1.1 Схематичне відображення відмінностей Web-сайту та Web-додатка

Найкращий спосіб збільшити зацікавленість користувачів у вашому бізнесі – це звісно ж створення персоналізованого сервісу. Web-додаток дає можливість інтегрувати підписку на контент Web-сайту, за допомогою якої користувач буде отримувати потрібний йому сервіс згідно із його вподобаннями.

Іноді навіть не потрібно використовувати для цього саме підписку. Бувають випадки, коли це доречно і в простому інтернет-магазині. Все залежить виключно від конкретного веб-сайту. Але як показує практика таких сервісів, як Amazon, Spotify, Netflix – використання рекомендаційної системи, це саме те що потрібно для приваблення клієнтів.

Дуже часто можна зустрітись із використанням Web-додатків у різноманітних стартапах та сервісах малого бізнесу. Пов’язано це в першу чергу з дешевезною розробки подібного роду ресурсів. А ось вплив подібного роду додатків на загальну успішність проекту – більш ніж очевидний. Тому що як можна помітити із статистики наявної в інтернеті, багато початкових нових компаній робить ставку саме на Web-додатки, а не на розробку окремих варіантів програм під кожну з платформ.

Таким чином компанії економлять максимум ресурсів, отримуючи непоганий результат. Особливо це помітно по інтернет-магазинам, яким потрібно привернути увагу користувача саме до тих продуктів, які дійсно можуть виявитись для нього корисними.

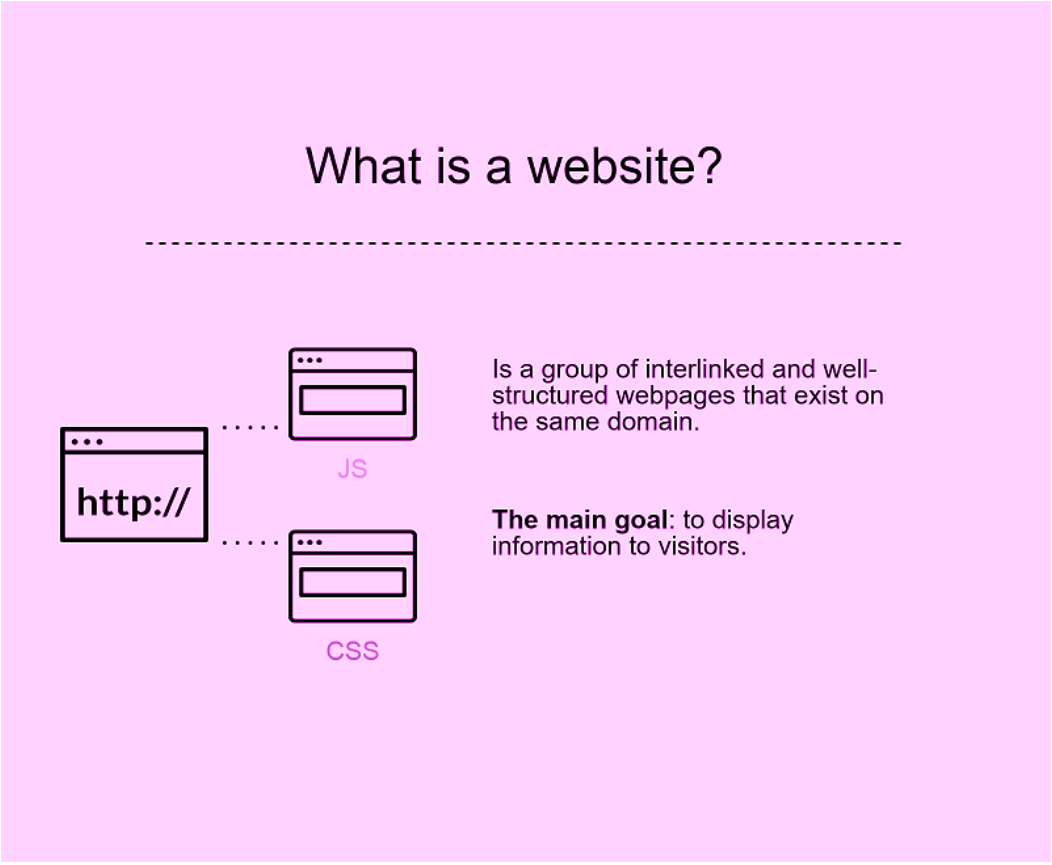

Рис. 1.2 Що таке Web-сайт

Коли стоїть вибір між Web-сайтом та Web-додатком, то потрібно зважити переваги, які ви отримуєте завдяки використанню додатка. Враховуючі всі перечисленні відмінності, вибір більш ніж очевидний. Розробка Web-додатку – це найкращий вклад у розвиток власного бізнесу в сучасних умовах, коли в центрі уваги користувачів знаходиться інформація.

Особливо це помітно по поступовій діджиталізації бізнесу. Зараз компанії зацікавлені в персоналізації сервісів і надання якомога більш якісних послуг для користувачів. Саме тому Web-додатки стрімко розвиваються і стають все більш популярнішими серед сучасних підприємців.

1.3 Технології розробки Web-додатку

Найважливіше, що потрібно враховувати під час розробки Web-додатку – це звичайно ж набір технологій. Вибір правильного набору технологій є особливо складним завданням для малого бізнесу та стартапів, оскільки вони, як правило, мають обмежений бюджет, і, таким чином, потрібен стек технологій, який забезпечить найбільш вигідний розподіл ресурсів.

Правильний стек технологій є значною мірою запорукою успіху проекту, тоді як неправильний вибір технологій розробки веб-додатків може стати причиною невдачі. Перш ніж перейти до критеріїв вибору сучасного стека веб-технологій, слід проаналізувати процес розробки Web-додатків [Рис. 1.3]. Не заглиблюючись у деталі, до розробки Web-додатків є два підходи: клієнтська та серверна. Сторона клієнта також називається Front-End. Програмування на стороні сервера включає в себе додаток (і серверну мову програмування, що ним керує), базу даних та сам сервер.

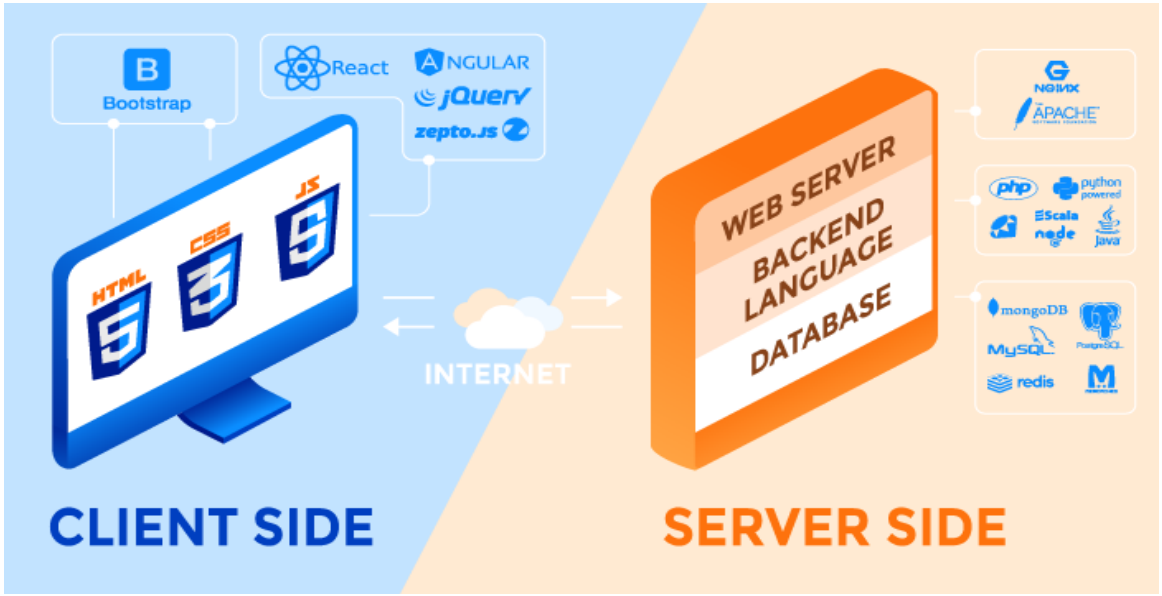

Рис.1.3 Схематичне зображення складових розробки Web-додатків

Програмування на стороні клієнта (Front-End)

Перше на що потрібно звернути увагу – це звісно ж програмування на стороні клієнта. Воно включає в себе все що буде бачити клієнт на екрані свого пристрою під час використання стандартного браузеру. Front-End технології включають в себе наступні компоненти:

- HTML;
- CSS;
- JavaScript.

HTML (Hypertext Markup Language) – це ніщо інше, як мова розмітки. За допомогою неї можна вказувати браузеру, який контент потрібно відображати на конкретних Web-сторінках.

CSS (Cascading Style Sheets) – дозволяє вказувати стиль розміщеного на сторінці контенту.

JS (JavaScript) – мова програмування, котра дозволяє зробити Web-сторінку інтерактивною. Існує багато Javascript бібліотек (такі як jQuery, React.js і Zepto.js) і фрейморки (Angular, Vue, Backbone і Ember). Завдяки їм, можна значно швидше розробляти Web-продукти.

Програмування на стороні сервера

Те, що відбувається на стороні сервера – непомітно для користувача. Але процес обробки інформації на стороні сервера є надзвичайно важливим. Сервер представляє собою аналог генератора, що постачає електроенергію для дому.

Існують мови програмування, за допомогою яких і виконується більшість операцій по взаємодії із сервером. Саме вони допомагають побудувати логіку, на якій буде базуватись Web-сайт і Web-додаток. Фреймворки для цих програмних мов пропонують багато інструментів, які дозволяють писати код значно простіше і швидше. Можна згадати одразу декілька мов програмування і їх найбільш популярні фреймворки:

- Ruby (Ruby on Rails);
- Python (Django, Flask, Pylons);
- PHP (Laravel);
- Java (Spring);
- Scala (Play).

Node.js є гарним варіантом для Back-End розробки. Це одна із бібліотек Java-Script, яка була спеціально розроблена для взаємодії з сервером.

Web-додатку потрібне місце для того, щоб зберігати інформацію. Саме для цього і існує база даних. Можна виділити два види бази даних:

- Реляційні;
- Не-реляційні.

Вони в свою чергу також діляться на декілька категорій. Можна відмітити плюси і мінуси кожної із категорій. Серед найбільш популярних баз даних у Web-розробці, можна відмітити:

- MySQL (Реляційна);
- PostgreSQL (Реляційна);
- MongoDB (Не-реляційна, документальна).

Web-додатку [Рис. 1.4] потрібна система кешу, щоб передбачити перевантаження бази даних, та впоратись із великою кількістю трафіку. Memcached і Redis представляють собою найбільш розповсюджені системи кешу.

Ну і звичайно ж, Web-додатку потрібен сервер, який би допоміг впоратись із заявками від клієнтських комп'ютерів. Зазвичай, це:

- Apache;
- Nginx.

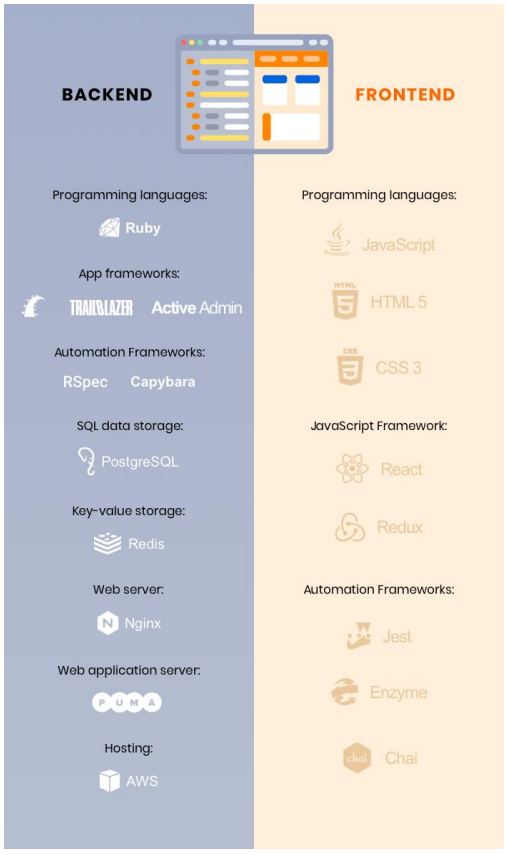

Рис. 1.4 Технології розробки Web-додатку

1.4 Критерії вибору технологій для створення Web-додатку

Щоб не прогадати з технологіями розробки Web-додатку, обов'язково потрібно одразу визначитись з критеріями вибору. Тому перш за все треба обдумати план проекту. В залежності від того, наскільки складним буде цифровий продукт, підбираються відповідні інструменти розробки. Обравши відповідний набір інструментів, можна значно покращити загально працездатність майбутнього Web-додатку.

В залежності від складності, Web-проекти можуть бути поділені на три основні типи:

- **Прості.** Такі Web-додатки створені за допомогою готових розробок (Такі як CMS інструменти для керуванням контентом) Простими прикладами являються лендінг-сторінки, які мають невеликий функціонал;
- **Середній рівень.** Такі додатки мають значно більше функцій, але все ще представляють не самі складні додатки, які створюються за допомогою фреймворків. Прикладами можуть бути: додатки для комерційних магазинів та виробництв;
- **Комплексні.** Такі додатки мають багато функцій та інтеграцій. Вони створені за допомогою різних технологій розробки Web-проектів, та з використанням різноманітних мов програмування. Прикладами таких складних додатків можуть бути соціальні мережі, великі комерційні мережі, програми по регулюванню фінансів і так далі.

Потрібно також брати до уваги той факт, що складність проекта – це далеко не все, що впливає на його розробку. Потрібно як мінімум брати до уваги ще і завдання, яке стоїть перед цим продуктом. Таким чином Web-додатки:

- **Виконують складні задачі з високим навантаженням**. Якщо на додаток буде покладено занадто навантаження, то потрібно одразу обдувати оптимізацію продукту з точки зору мов програмування і фреймворків, які будуть використовуватись в процесі розробки. Прикладами таких проектів є відео/аудіо стрімінгові додатки і сервіси розповсюдження файлів;
- **Мають високу пропускну здатність.** Різні набори технологій потрібні, щоб зробити Web-додаток якомога більш швидким. Тільки завдяки високій пропускній здатності. Найчастіше подібне можна зустріти в соціальних мережах, тому що ці веб-сайти потребують найбільшої швидкості роботи.

1.5 Приклади успішних Web-додатків

Можна навести декілька прикладів надзвичайно успішних Web-додатків, котрі існують на протязі багатьох років і встигли себе зарекомендувати. Серед них можна відзначити Shopify. Це додаток, який дозволяє підприємцям значно покращити роботу онлайн-магазинів. В основу розробки цього Web-додатку покладено технологію Ruby on Rails.

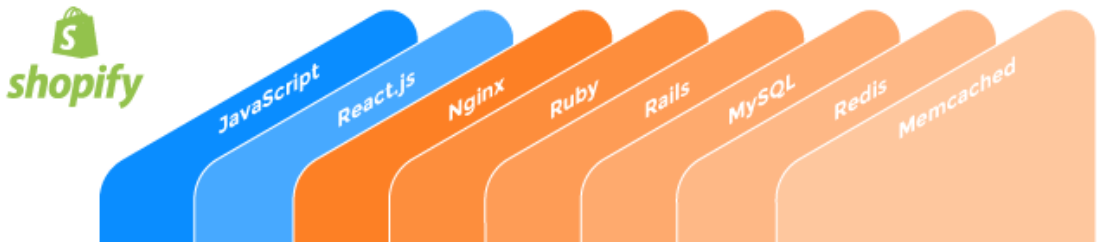

Рис. 1.5 Технології розробки Web-додатку «Shopify»

Quora – також один із найуспішніших проектів серед Web-додатків. Він допомагає людям з усього світу знаходити відповіді на питання. В цьому додатку також реалізована система рекомендацій.

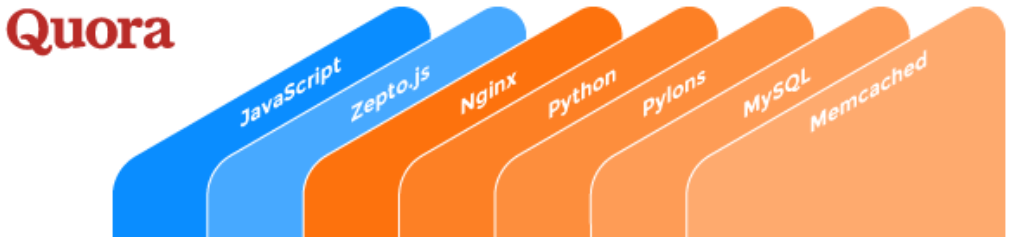

Рис. 1.6 Технології розробки Web-додатку «Quora»

Одною з найкращих платформ для обговорення новин та трендів являється Reddit. Це ще один приклад того, що Web-додаток може бути глобальним і здатним допомагати людям, незалежно від сфери їх діяльності. В основу цієї платформи покладено чимало технологій.

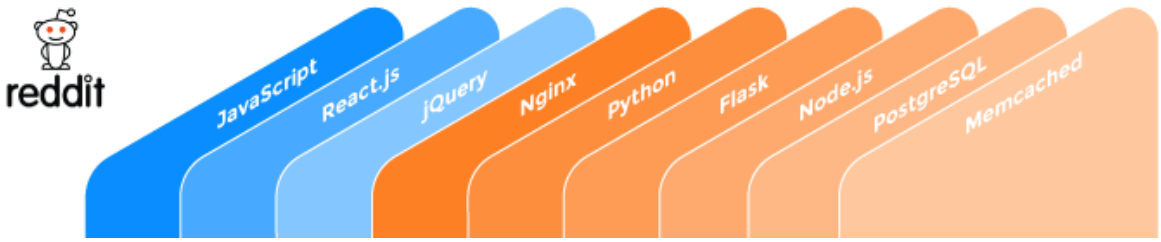

Рис. 1.7 Технології розробки Web-додатку «Reddit»

1.6 Місце Web-додатку у сучасному світі

Web-додатки стали ключовим компонентом сучасного бізнесу. Використовуючи Web-додатки, будь-яка компанія може значно спростити виконання своїх завдань. Завдяки додаткам можна охватити більшу аудиторію клієнтів. Є декілька об'єктивних причин того, що Web-додатки є більш доцільними у сучасному світі, аніж звичайні Web-сайти, якщо мова йде про розвиток мережі для охоплення якомога більшої аудиторії клієнтів.

На даний момент, комерційні компанії просто не можуть рости належним чином, якщо не будуть використовувати Web-додатки. Можна сказати що подібна розробка може зіграти найважливішу роль у побудові цілого бренду. Web-додаток виступає у ролі комунікаційного каналу між потенційним клієнтом та компанією, яка надає контент. За допомогою сучасної системи відображення контенту можна отримати значно більше інформації про вподобання клієнта і тим самим зростити успішність компанії в майбутньому.

Завдяки розробці Web-додатку [Рис. 1.8], компанія може відкрити абсолютно нові горизонти по надбанню клієнтів. Особливо це зіграє важливу в наступні роки, коли без сервісу в інтернеті неможливо буде представити ні одну серйозну компанію.

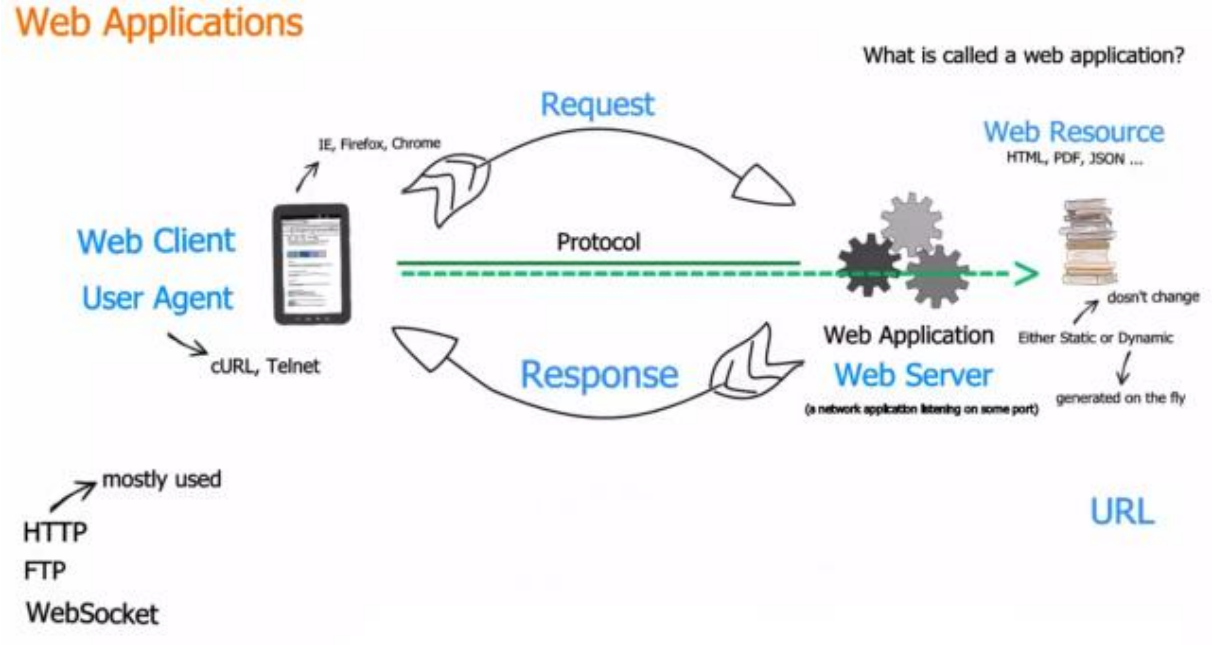

Рис. 1.8 Схема роботи сучасного Web-додатку

Ще одним великим плюсом Web-додатків являється можливість покращити службу підтримки. Тому що всередині подібної розробки можна встановити функціонал по зв'язку із клієнтом. За рахунок цього, клієнт зможе просто звернутись до співробітників компанії через форму

зворотного зв’язку. Тим самим, будь-які проблеми користувача будуть вирішені значно швидше, аніж це було раніше. Більше не існує бар’єрів, пов’язаних із відстанню. Тому що в будь-який момент часу, клієнт здатний просто задати питання, щодо товару чи послуги, яка його цікавить в конкретний момент часу.

Звісно ж не треба забувати і про те, що в сучасному світі бізнесу, дуже важливо залишатися на плаву по зрівнянню із конкурентами. Їх може бути дуже багато, зважаючи тотальний розвиток мережі інтернет. А оскільки Web-додаток – це ефективний спосіб утримання клієнтів, то немає нічого дивного в популярності подібних проектів. Завдяки крос-платформенності Web-додатків, більше не потрібно перейматися за те, яким пристроєм користується клієнт. Навіть якщо це мобільний телефон чи планшет – він все одно зможе відкрити веб-сайт компанії і тим самим отримає доступ до контенту.

Можна виділити декілька важливих причин, чому Web-додатки стали настільки важливими при побудові будь-якої сучасної компанії. Як було сказано раніше – крос-платформеність, це одна із найважливіших рис, котра виділяє такі додатки з поміж інших. Розробивши проект одного разу, його можна буде легко адаптувати під інші пристрої. Для цього не потрібно створювати декілька клієнтів для кожної з актуальних на сьогоднішній день платформ. Незалежно від того, яка саме операційна система встановлена на комп’ютері чи мобільному пристрої, додаток усе одно буде працювати максимально якісно. Тому що у кожної ОС є браузер, який дозволить запустити додаток. Вибір конкретного браузера буде залежати виключно від вподобань конкретного користувача.

Ще однією важливою перевагою Web-додатків, є робота з ресурсами пристрою. Завдяки тому, що додаток запускається через браузер, не має виникати ніяких проблем з надмірним завантаженням пам’яті чи встановленням специфічного програмного забезпечення на будь-якому девайсі. Web-додатки значно більш ефективні з цієї точки зору, тому здатні запускатись навіть на дуже слабких пристроях. Що дуже важливо – вносити зміни у додаток значно простіше, коли він представлений в мережі, тому що не потрібно окремо вказувати клієнту на необхідність встановлення нової версії. Вона може бути встановлена автоматично на фоні, а потім запущена з відкриттям нової сторінки.

Так як кількість товару, послуг чи просто інформації на ресурсі може збільшуватись, то потрібно вносити ці самі зміни і в сам додаток. У випадку Web-додатків, робити це надзвичайно просто. Тому що є безліч програм і технологій, котрі дозволяють з легкістю вносити зміни в базу даних та в презентацію тієї чи іншої сторінки сайту в режимі реального часу.

Не варто забувати і про той факт, що Web-додатки мають непоганий захист від сторонніх користувачів [Рис. 1.8]. Для цього існує адміністрування серверу, яке дозволяє виключити необхідність перевірки

сотні комп’ютерів, в той час, коли можна просто подивитись на декілька запитів, які поступали на сервер.

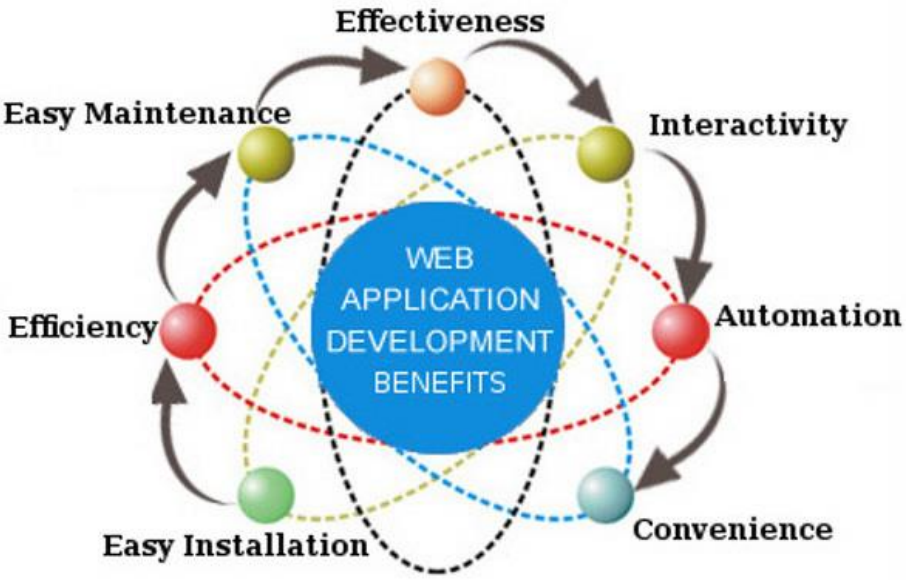

Рис. 1.9 Переваги сучасних Web-додатків

ВИСНОВОК ДО РОЗДІЛУ 1

Можна зробити висновки щодо особливостей розробки Web-додатку. Саме завдяки подібним проектам можна легко позбавитись будь-яких проблем, пов’язаних із представленням інформації сучасному користувачу. Розробка Web-додатку займає не так багато часу і ресурсів, щоб віддавати перевагу звичайним сайтам. Така розробка може бути відкрита на будь-якому пристрої, тому база потенційних користувачів автоматично збільшується. Можна забути про проблеми з оновленням програмного забезпечення, релізом нових версій сервісу. Тому що Web-додаток можна легко покращувати, не докладаючи до цього багато зусиль.

Гнучкість технологій розробки дозволяє створити Web-додаток для будь-якої сфери діяльності. А якщо додати до подібного Web-додатку ще й рекомендаційну систему, можна отримати прекрасне рішення для будь-якого сучасного інтернет-магазину, стрімінгового сервісу. Це позбавить користувача необхідності витрачати свій час дарма. А також дозволить персоналізувати сервіс, що автоматично буде сприйматися користувачем значно цінніше, аніж звичайний сайт.

Розділ 2. Аналіз технологій для створення Web-додатку

Перш ніж почати працювати над проектом, потрібно провести детальний аналіз технологій, які можуть бути використані для створення Web-додатку. У реаліях двадцять першого сторіччя існує чимало інструментів, які можуть допомогти у розробці будь-якого Web-проекту. Для ефективної побудови проекту, необхідно обов'язково розглянути усі актуальні технології для створення Web-додатків. Адже тільки так можна виділити серед них найбільш ефективні, ті які зможуть принести найбільшу користь при найменших затратах. Є декілька критеріїв вибору, які і будуть виокремлювати потрібні технології:

- Швидкість у використані;
- Складність освоєння;
- Задача, яка буде вирішуватись завдяки конкретній технології;
- Можливість подальшого покращення продукту за рахунок редагування коду;
- Мультиплатформенний підхід;
- Актуальність технології, як у конкретний момент, так і у майбутньому.

2.1 HTML. Мова розмітки гіпертекстових документів

HTML – це акронім, який розшифровується як Hyper Text Markup Language. Ця мова розмітки гіпертекстових документів була вперше представлена в 1991 році, як інструмент для організації контента у Web-документах.

HTML використовується для розробки Web-сайтів та Web-додатків. Завдяки цій мові розмітки, можна встановити контент сторінки таким чином, щоб він був найбільш зручним у використані. В назві мови не просто так вказано «Гіпер». Це означає, що завдяки властивостям цієї мови розмітки, можна легко встановити гіперпосилання на будь-який документ наявний на просторах інтернету. Тим самим відкриваються широкі можливості для створення продуктів всесвітньої мережі.

Розмітка передбачає можливість працювати з текстом всередині тегів в документі. Під час першої появи HTML, розробники були досить обмежені в своїх можливостях представлення різноманітних продуктів на просторах всесвітньої мережі. Зараз є безліч інструментів для роботи із HTML. Всі вони в першу чергу розраховані на те, щоб полегшити задачу по створенню різноманітних проектів в інтернеті. Саме тому в більшості вже задачі початкові теги Web-документу, які потрібні для відображення сторінки.

Використовуючи HTML, розробник створює інтерфейс Web-додатку. Саме HTML рекомендовано W3C як мову для розмітки Web-документів. HTML теги з обох боків закриваються кутовими дужками [Рис. 2.1].

HTML дозволяє розміщувати відео, зображення та інші формати файлів, щоб сторінка була інтерактивною. Також у HTML документ можуть бути включені скрипти та стилі. Скрипти впливають на поведінку HTML сторінки, а стилі задають вид, в якому будуть представлені ті чи інші елементи Web-сторінок.

```
01.  <h1> This Is An Example Of An HTML Heading Tag </h1>
```

Рис. 2.1 Приклад тегу «H1» заголовок у HTML

HTML 5 є найбільш актуальним на сьогоднішній день стандартом цієї мови розмітки. Саме на цьому стандарті базується найбільша кількість Web-сторінок у інтернеті. В останній версії для HTML, що представляє собою HTML 5, ми отримуємо підтримку мультимедіа (аудіо та відео).

Сьогодні основним складовим елементом веб-сторінок є HTML 5. Він надав статичний вигляд веб-сторінок. А всі інші стилі та функції додаються за допомогою CSS, PHP, Javascript. Тому перш ніж вивчати ці мови, ви повинні вивчити HTML.

Можна сказати, що HTML – це основний будівельний блок Web-розробки. Всі сучасні Web-сайти, сервіси створення Web-сайтів, такі як WordPress, Wix, Weebly використовують HTML 5 для створення Web-сайту.

Навіть якщо ви хочете створити свій Web-сайт за допомогою цих сервісів, вам також потрібні базові знання HTML 5, щоб внести деякі необхідні зміни.

HTML - мова на основі тегів. Ми описуємо структуру Web-сторінки, використовуючи позначки (теги). Елементи HTML представлені тегами. Теги містять кутову дужку між цією кутовою дужкою, ми повинні написати найменування тега. У HTML є тег відкриття та тег закриття.

Починати розробку за допомогою HTML дуже просто. Не потрібно встановлювати дороге програмне забезпечення. Усі необхідні інструменти для HTML можна знайти безкоштовно на просторах інтернету. Для початку роботи з HTML вам потрібен лише текстовий редактор для написання HTML.

- Для Mac (TextEdit, Sublime Text, Visual Studio Code)
- Для Windows (Блокнот, Блокнот ++, піднесений текст тощо)
- І браузер для запуску HTML-файлу.

Запустити HTML документ можна у будь-якому браузері, такому як Google Chrome, Safari, Mozilla Firefox, Opera. Не має значення, який саме браузер буде обрано, більша частина коду сприймається браузером однаково [Рис. 2.2].

```
<!DOCTYPE html>
<html>
  <head>
    <title> My First Web Page </title>
  </head>

<body>

<p> This is a paragraph.</p>

</body>
</html>
```

Рис. 2.2 Приклад розмітки в HTML документі

Завдяки своїм новим та потужним функціям як для розробників, так і для кінцевих користувачів, він уже використовується для кодування веб-сайтів по всьому світу. HTML5 підтримується усіма сучасними настільними та мобільними браузерами або для Web-розробок мобільних додатків.

Можна виділити одразу декілька важливих переваг HTML 5. В першу чергу використання HTML5 допомагає усунути більшість тегів і замінити їх семантичними елементами. Тепер дизайнери можуть використовувати більш чистий і акуратний код. Можна зробити більш детальне розуміння структури сторінки за допомогою HTML5.

Завдяки новим функціям та стандартам HTML5 полегшує створення додаткового додатка, такого як інструменти перетягування, вікі, дискусійні дошки та інші корисні елементи. HTML 5 дозволяє ділитись геолокацією користувача через браузер. Ви можете зробити це за допомогою своєї IP-адреси, бездротового підключення до мережі, стільникової башти, з якою розмовляє ваш телефон, або спеціального обладнання для GPS, щоб обчислити широту та довготу за інформацією, що надсилається супутниками. Завдяки новому принципу зчитування геолокації HTML5, API надають розташування безпосередньо доступним для будь-якої програми, сумісної з браузером HTML5.

HTML5 надає розширені можливості для визначення файлів, які браузер повинен кешувати. Сторінки можна правильно завантажити навіть у режимі офлайн. Замість зменшення розміру файлів cookie HTML5 дозволяє використовувати сеансове зберігання та локальне зберігання. Зберігати структуровані дані тимчасово, а не на постійній основі.

Стандартизований код елементів HTML5 збільшує смислове значення веб-сторінки. Завдяки специфічним тегам для заголовків, наві, колонтитулів тощо, дизайнери знають значення та призначення у всьому форматі.

Основними перевагами HTML5 є вишукані форми, удосконалення користувальницького інтерфейсу, зменшена потреба в JavaScript та перевірка форм, що належать до HTML. Завдяки цій версії HTML, розробнику вдається значно спростити структуру самої сторінки і зробити її більше зрозумілою навіть для тієї людини, яка бачить її вперше.

Рис. 2.3 Типова сторінка створена за допомогою HTML

HTML5 має безліч функцій, зокрема <video>, <audio>, діаграми, якісні малюнки, анімації та інтеграція SVG-вмісту. Інтегрувати мультимедійний та графічний вміст у Інтернет легко, не використовуючи якийсь сторонній інструментарій. Семантичне значення документа може бути збагачене новими елементами, такими як <section>, <article>, <header>, <nav>.

Можна відмітити ще декілька важливих нововведень, які були представлені саме в HTML 5. Серед них багато тих, що відносяться безпосередньо до розробки Web-додатків.

А саме:

- Введення елемента <figure> для асоціації підписів зі своїми аналогами зображень;
- Елемент <small> стосується дрібного друку; її можна позначати як правильну обгортку для цієї інформації;
- HTML 5 повністю видаляє атрибут <type>;
- Використання цитат може залежати від користувачів;
- <Content editable> дозволяє користувачеві редагувати будь-який текст, що міститься в елементі, включаючи його дочірні;
- HTML5 пропонує введення електронної пошти, що дозволяє вказувати браузеру дозволяти лише рядки, що підтверджують адреси електронної пошти.

2.2 CSS. Каскадні таблиці стилів

Каскадні таблиці стилів, представлені вперше в 1996 році, пропонують можливість розширити HTML у візуальному поданні Web-сторінки.

Ця мова дозволяє зв’язати елементи HTML з "шаблонами документів" (таблицями стилів або таблицями стилів), які крім вмісту топографічної інформації візуальних елементів сторінки дозволяють повністю відокремити структуру вмісту від їх поточного подання та презентація не лише на моніторі, але на будь-якому екрані (мобільний, КПК тощо), технології підтримки (екранні зчитувачі, рядки Брайля) або друкований папір. Мова (x) HTML сама була підкріплена для побудови логічних структур сторінки.

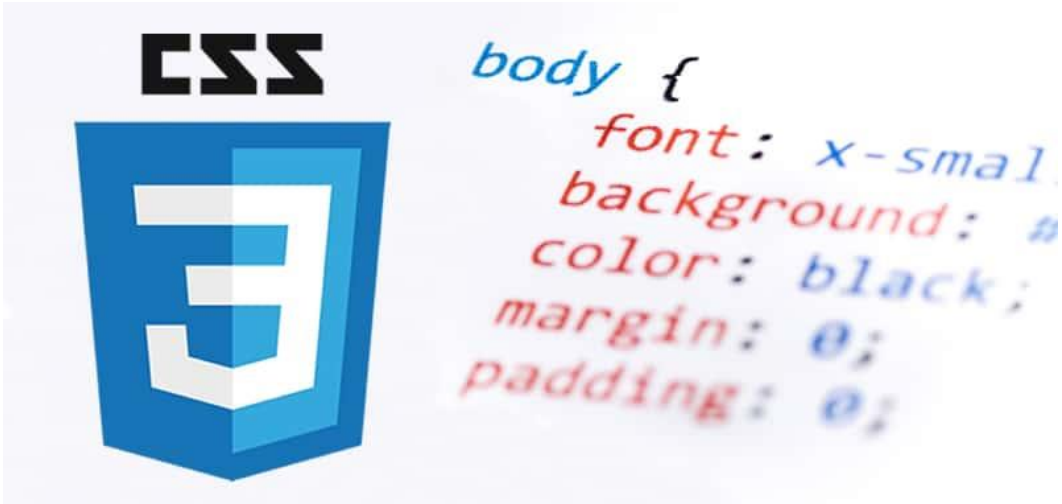

Рис 2.4 Логотип CSS 3

Можна сказати що завдяки CSS людині набагато легше сприймати інформацію представлену на Web-сайті. Це пов’язано в першу чергу з тим, що стилі дозволяють відображати сторінку правильно на будь-якому пристрої. Навіть якщо Web-сайт чи додаток буде відкрито на мобильному телефоні чи планшеті – документ усе одно буде відображатись належним чином.

CSS містить безліч функцій та методів, за допомогою яких надавати розширеним можливостям відвідувачам. Простий вихідний код та впорядкована структура також полегшують сприйняття сторінки пошуковими роботами. Найважливіші ключові слова в текстах та заголовках обробляються разом, і немає небезпеки.

Економічні переваги очевидні вже у згаданих моментах. Один економить, перш за все, час у всіх процесах. Вміст можна створити разом для всіх документів, не турбуючись про формат і форму. Аркуші стилів створюються та підтримуються централізовано. Будь-який співавтор, все ще не забуваючи про створення одних і тих же документів, може без проблем здійснити модифікації, оновлення і, навіть, повне відновлення їх.

Вихідний код, займаючи мінімальний простір, представляє негайну економію, особливо на Web-сайтах багатьох сторінок, де вартість трафіку даних буде різко знижена.

Очевидно, що сторінки, розроблені відповідно до стандарту і з максимальним зменшенням бар'єрів, передають інтелектуальний імідж бізнесу. Навіщо відмовлятися від великої кількості потенційних клієнтів або швидшої сторінки з кращим рейтингом пошукових систем? І, звичайно, це також матиме більший вплив на широку громадськість, оскільки поширеться слово про те, що вашою сторінкою дуже легко керувати, інформація про продукти легко знаходитись.

Те, як ваш Web-сайт виглядає та представленний перед користувачами, є величезним визначальним фактором User Experience (UX). Власники Web-сайтів не залишають жодних зайвих елементів, щоб їх естетичні рішення узгоджувались з функціональною частиною. Основні з цих естетичних рішень - це впровадження, переваги та недоліки CSS.

Скорочена форма для "Каскадного аркуша стилів", CSS - це те, що входить до форматування макета Web-сайту. Це набір правил форматування, які допомагають розробникам Web-сайтів контролювати зовнішній вигляд та показ Web-сайту, над яким вони працюють. CSS дозволяє їм визначати стильові рішення, такі як розташування зображень, розмір шрифту, колір тла та все інше, що впливає на те, як веб-сайт відображатиметься у Web-браузері. Крім того, за допомогою онлайн-редактора можна впорядкувати, налагодити свої стильові таблиці та легко стиснути CSS. За допомогою CSS дизайнер може реалізувати функції, які раніше не були визначені в HTML-документі сторінки (використовувались для створення веб-сторінок). З новою розробкою HTML та CSS працюють рука об руку, щоб створити чудовий веб-сайт. Найкраще в CSS - це

можливість вносити однакові зміни на всі сторінки веб-сайту. Розробники працюють над тим, щоб визначити стиль у каскадному аркуші стилів, і вони можуть використовувати стиль на декількох сторінках, просто посилаючись на цей конкретний файл CSS. Отже, кількість стильових робіт зберігається і повторення усувається. CSS вказує відображати HTML щодо того, як Web-сайт відображатиметься у користувача. У CSS є свої переваги та недоліки.

Розглянемо приклад. Ви запускаєте Web-сайт, який містить 40 і більше сторінок. Через необхідність певних стратегічних змін у змісті веб-сайту, тепер вам потрібно змінити розмір тексту з 14pt на 12pt або навпаки. Як ви думаєте, скільки часу знадобиться вручну для зміни розміру на всіх цих 40 сторінках? Багато! Тут CSS входить у сценарій як рятівник. Потім ви можете визначити зміни в одному файлі CSS і посилати всі ці 40 сторінок на той самий файл, і ваша робота буде виконана. Весь Web-сайт почне відображати зміни розміру.

Якщо доведеться зробити більш змінні зміни у своєму вмісті. Знову ж таки, за допомогою єдиного аркуша стилів ви зможете переконатися, що зміни виглядають рівномірно на всіх сторінках, а не в грудному порядку. Якби не CSS, вам доведеться робити нотатки про зміни, внесені на одну сторінку, і посилатись на неї під час внесення змін на іншу сторінку, завжди переходячи вперед і назад. Уявіть, яка кількість виснаження та мозкової активності потрібна для обгрунтування послідовності цих змін протягом усього часу. Однак у CSS ваші зміни добре застосовуються послідовно.

Щільність коду на вашому веб-сайті сприяє його швидкості. Чим більше код, тим повільніше стає веб-сайт. FYI, відвідувачі швидко відмовляються від веб-сайту, якщо на це потрібно більше 2-3 секунд. Всього кілька рядків коду - це все, що потрібно для внесення змін до великої кількості сторінок на Web-сайті за допомогою CSS. Оскільки код мінімальний, база даних веб-сайту залишається незабрудненою, таким чином, усуваючи будь-які проблеми із завантаженням веб-сайтів. Впровадження CSS разом із вибором найкращих розробників веб-сайтів - це чудовий спосіб підключити відвідувачів до Web-сайту, оскільки їм не доведеться чекати завантаження Web-сайту.

Зміни у CSS пристосовані до пристроїв. Оскільки для доступу до веб-сайтів через Інтернет користується великою кількістю різноманітних розумних пристроїв, існує потреба у чуйному веб-дизайні. CSS служить цій меті, роблячи Ваші веб-сторінки більш чуйними, щоб вони відображалися однаково на всіх пристроях.

CSS дозволяє визначати зміни в положенні веб-елементів, присутніх на сторінці. З його реалізацією розробники можуть розміщувати елементи HTML у місці, яке їм подобається, щоб узгодити естетичну привабливість сторінки чи інші подібні міркування.

CSS виконує роль SEO оптимізатора на Web-сайті. Правда полягає в тому, що боти пошукової системи дають неточні деталі, коли вони проходять через громіздку купу HTML-коду. Однак, коли CSS діє, атрибути дизайну веб-сайтів визначені, і на ньому менше коду, що робить веб-сайт SEO дружнім.

Здійснення початкових змін CSS на веб-сайті легко для розробника. Однак після внесення змін вам доведеться підтвердити сумісність, якщо CSS відображає подібні ефекти змін у всіх браузерах. Це просто пов'язано з тим, що CSS працює по-різному в різних браузерах.

CSS уповноважує веб-дизайнера вносити великі зміни в веб-макет всіх сторінок веб-сайту за допомогою одного файлу. Це дозволяє створити легкий, але чуйний веб-сайт, який швидко завантажується та чудово вражає аудиторію своїм дисплеєм. Отже, кожен веб-сайт повинен мати свою справедливу частку CSS незалежно від багатьох переваг та недоліків CSS.

```
#first .blue h1 {
  color: blue;
}
#second .red.bold h1 {
  color: red;
}
```

Рис 2.5 Приклад CSS коду

Каскад притаманний роботі з CSS - адже саме він надає «Каскадним таблицям стилів» їх каскадний характер. Каскад може бути потужним інструментом, але неправильне його використання може призвести до крихких таблиць стилів, які дають розробникам передусім кошмари в будь-який час, коли вони повинні внести зміни. Під час занурення в каскад ми також розглянемо кілька способів уберегти каскад від руки.

CSS Cascade - це алгоритм, за допомогою якого браузер вирішує, які стилі CSS застосувати до елемента - багато людей люблять вважати це стилем, який «перемагає».

Щоб краще зрозуміти каскад CSS, корисно подумати про декларацію CSS як "атрибути". Ці атрибути можуть бути різними частинами декларації - наприклад, селектором або властивостями CSS - або вони можуть бути пов’язані з тим, де існує декларація CSS (як-от її походження або позиція у вихідному коді).

CSS-каскад займає декілька цих атрибутів і присвоює кожному з них вагу. Якщо правило CSS виграє на рівні вищого пріоритету, це правило, яке виграє.

Однак якщо два правила все ще суперечать певній вазі, алгоритм буде продовжувати «каскадувати вниз» і перевіряти атрибути нижчого пріоритету, поки не знайде той, який виграє.

Ось атрибути, які алгоритм CSS Cascade перевіряє, перелічені в порядку від найвищої до найменшої ваги.

- Походження та значення;
- Специфікація селектора;
- Порядок явки;
- Початкові та успадковані властивості (значення за замовчуванням)

Оскільки CSS представляє собою ніщо інше як багаторазове використання одного і того ж коду на протязі сотень строк, то немає нічого незвичного в тому, що часто використовуються різноманітні допоміжні засоби, такі як LESS.

Less є попереднім процесором CSS і розширює можливості стандартного CSS. LESS має багато функцій, таких як: редагування, функціонування, комбінації, вкладені правила, операції, імпортування та здатність робити великі документи значно меншими за допомогою генерування CSS коду.

Серед переваг LESS можна відзначити:

1. LESS - це передпроцесор CSS, і після компіляції він генерує простий CSS, який працює в браузері.
2. LESS – це швидко і просто.
3. Структура очищення завдяки використанню Nesting;
4. LESS значно простіший і краще організований ніж звичайний CSS;
5. LESS підтримує сумісність між браузерами;
6. LESS економить час у порівнянні з CSS;
7. Кодування швидше, оскільки список операторів, наданий LESS;
8. Використання Mixins вирішує можливість повторного використання коду та вбудовує всі властивості класу в інший клас шляхом простого включення назви класу як одного з його властивостей;
9. Є можливість повторно використовувати цілі класи, посилаючись на них у наборі правил;
10. LESS має багато математичних та функціональних властивостей;
11. Можна імпортувати інші LESS файли у менший файли, у яких можуть бути визначені змінні, а потім використовувати ці змінні в імпортованому файлі LESS;
12. LESS має деякі заздалегідь визначені функції, і ви також можете визначити власні функції та використовувати їх у всьому коді;
13. LESS компілюється швидше, ніж будь-який інший попередній процесор CSS;
14. LESS також підтримує функцію Lazy Loading, тобто у файлі ви можете визначати свої змінні в будь-якому місці, як до або після використання змінної.

CSS - це статична мова, тоді як LESS - це динамічна мова, яка генерує CSS після компіляції коду. У CSS ви пишете статичний код для розробки веб-сторінки, але в LESS можна визначити код за допомогою змінних, і після компіляції коду значення змінних будуть проаналізовані та згенеровані в CSS. У CSS модифікація може бути досить складною, як, наприклад, якщо ми скажемо, що вам потрібно змінити колір # 000 на #fff у всьому файлі CSS, тому це може бути досить складно і потрібно змінити всі місця, де він використовувався, але в LESS змінюється значення змінної одноразово, і після компіляції це змінне значення буде змінено в цілому CSS через код, де використовується ця змінна.

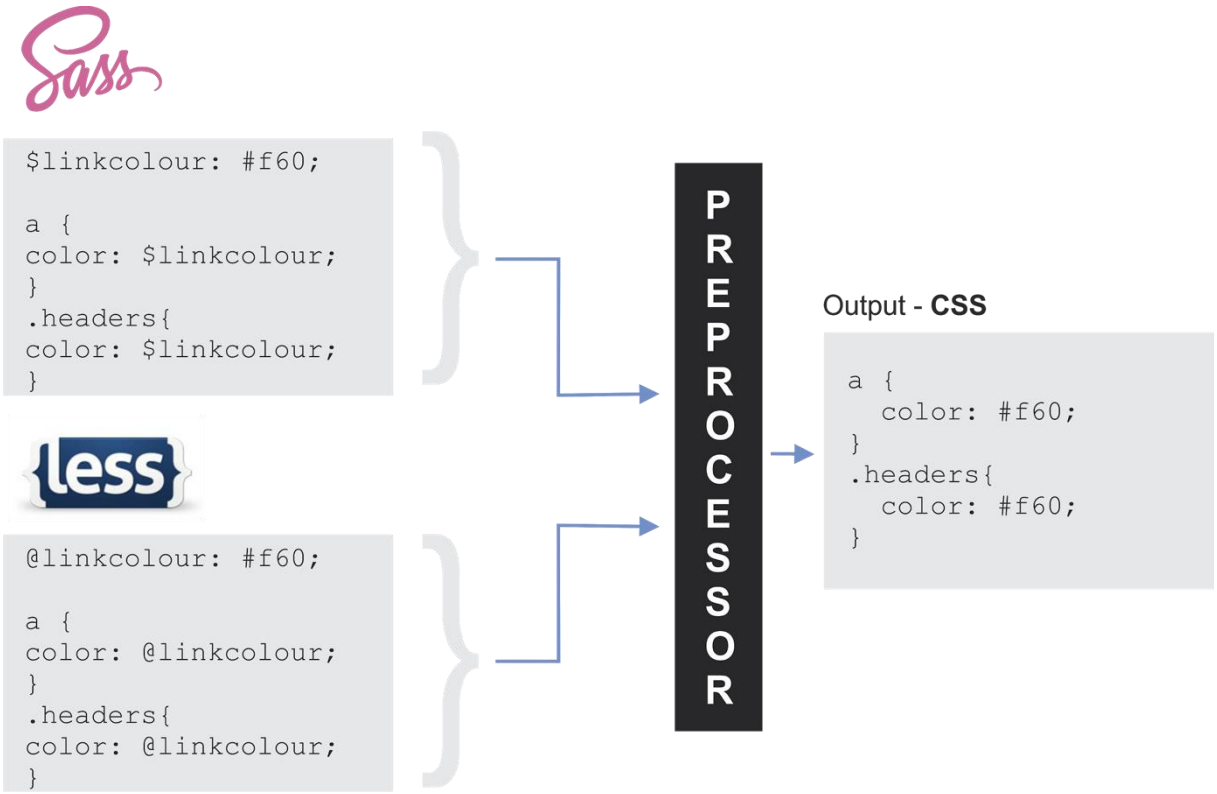

Рис 2.6 Схема роботи LESS

У CSS вам доведеться визначати один і той же клас багато разів з різними операціями, але в LESS через особливості вкладення та mixins не потрібно багато разів визначати класи для різних операцій. CSS подовжує стилізацію коду порівняно з LESS. CSS займає багато часу в порівнянні з LESS, але так, це легко вивчити та записати CSS порівняно з LESS.

А якщо використовується magento, то використовувати LESS у модулях краще, ніж використовувати CSS, оскільки в модулях потрібно написати багато коду для проектування. Отже, якщо ви використовуєте CSS там, ви можете зіткнутися з деякими труднощами, і це також збільшує розмір файлу через довжину коду, але якщо ви використовуєте LESS, код

проекту буде добре організований з меншим розміром файлу, а також збільшує повторне використання коду.

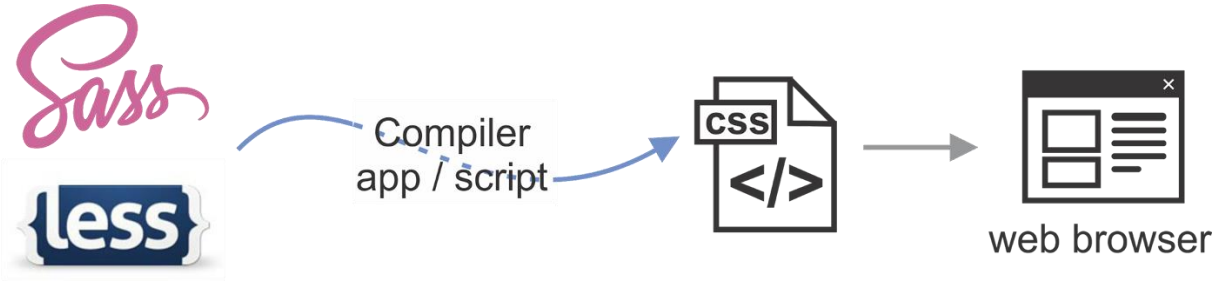

Рис. 2.7 Сценарій використання LESS

2.3 JS. Об’єктно-орієнтована мова програмування

JavaScript (JS) - це мова скриптів, призначена для надання Web-сайтам інтерактивного контенту, наприклад, меню, анімації, інтерактивних карт, прокручування тексту та інших типів динамічного контенту. Це основна технологія для сучасних Web-сайтів і працює із HTML та CSS, забезпечуючи найкращі враження від користування ресурсом в інтернеті.

Хоча більшість звичайних користувачів не знають про його роль, JavaScript відіграє основну роль у створенні та роботі інтерактивних веб-сайтів, мобільних додатків та ігор на основі браузера.

Змінні JavaScript можуть бути будь-якими з наведених нижче:

- Числа
- Струни
- Масиви
- Функції
- Об'єкти

JavaScript був вперше розроблений NetScape у 1995 році. Його не слід плутати з Java, хоча вони мають певну схожість. Існують як переваги та недоліки JavaScript у Web-технологіях. Більшість Web-дизайнерів та розробників покладаються на нього для створення не тільки звичайних сайтів, але й Web-додатків.

Особливості Javascript включають:

- Більше інтерактивних елементів Web-сайту та інтерфейсів, що також поширюється на Web-додатки. До них відносяться кнопки, а також інтерактивність наведення вказівника, функціональність меню, анімація та інші основи сучасного веб-досвіду.
- Швидкість, оскільки мова виконується на пристрої користувача, а не на сервері Web-сайту. Це мінімізує запити серверів, прискорюючи досвід роботи користувача.

- Зниження навантаження сервера, так як пристрій користувача робить більшу частину важкого обчислення, зберігаючи як пропускну здатність, так і навантаження.
- Популярність: будь-який браузер підтримує JavaScript. Сучасні браузери включають Chrome, Firefox, Safari, Edge, Internet Explorer та багато інших. Інтернет-магазини включають Amazon, PayPal, Google та багато інших. Всі вони не обходяться без підтримки JavaScript.
- Не потрібна компіляція. Браузери інтерпретують JavaScript як теги HTML.
- Код відносно простий у вивченні, запису та налагодженні. Синтаксис простий і гнучкий. Його також можна використовувати всередині скриптів, написаних іншими мовами.
- Взаємодія з іншими мовами програмування та сценаріями та JavaScript може бути вставлена на будь-яку сторінку незалежно від розширення файлу.

Жодна технологія не може бути досконалою на всі сто відсотків. Хоча майже всі Web-сайти використовують JavaScript, мова має декілька недоліків, котрі потрібно відмітити.

Недоліки JavaScript включають:

- Можливість легкого копіювання коду. JavaScript-код легко додається на Web-сторінки і його можна переглянути у браузері за допомогою вбудованого інструментарія. У кожного користувача є можливість легко скопіювати JavaScript код із іншого Web-сайту;
- Уразливість для міжсайтового сценарію (XSS) та інших видів введення зловмисного коду. Недобросовісні користувачі можуть потенційно змінити код Web-сайту чи Web-програми, яка потім виконає зловмисний код на пристрої користувача. Ці атаки часто виявляються як власником сайту, так і користувачем.
- Не працюватиме в браузерах, де користувач вручну відключив JavaScript. Така практика зустрічається досить рідко, але користувачі дійсно самостійно вимикають підтримку JavaScript у цілях власної безпеки.
- Несумісність із деякими старими браузерами. Хоча це складно назвати явним недоліком через те що це впливає на розвиток технологій. Адже люди частіше оновлюють версію свого браузеру саме для того, щоб отримати можливість використовувати конкретний Web-сайт із JavaScript.

Для розробки великих за розміром проектів використовують не тільки чистий JavaScript, але і різного роду бібліотеки. Це пов’язано із

великою кількістю переваг, які надають фреймворки. Якщо говорити про те, чому взагалі виникла потреба у використанні JavaScript, то тут насправді дуже проста схема:

- HTML структурує контент на сторінці;
- CSS стилізує контент на сторінці;
- JavaScript додає інтерактивності в контент.

Може здатись що в цій формулі немає місця для творчості, та насправді це не зовсім так. Тому що завжди є можливість внести різноманітність за допомогою сторонніх технологій, які зараз активно використовуються разом із JavaScript.

Початкова мета JS полягала в тому, щоб зробити Інтернет повноцінною платформою додатків, тобто JavaScript працюватиме як на клієнті, так і на сервері. Успішний результат прийшов не одразу, але напрямок було обрано правильний. Тому що зараз складно уявити як виглядав би інтернет, якби не було можливості використовувати мову програмування JavaScript.

З появою JavaScript можна було задовольнити різні аудиторії: 1) ентузіастів та професіоналів на рівні підприємства з Java та 2) любителів сценаріїв та дизайнерів з JavaScript. Можна сказати що це був перший відголосок тих самих сучасних Web-розробників.

Назва JavaScript походить від спроби подолати хвилю популярності Java та прискорити її прийняття. Сьогодні ви не знайдете багато подібності між обома мовами.

JavaScript приніс динамічні можливості в Інтернет. Що це означає? Ось кілька речей, які можна побачити щоразу, коли ви вмикаєте браузер та починаєте переходити із одного сайту на інший, це результат JavaScript:

- Автозаповнення;
- Завантаження нового вмісту або даних на сторінку без перезавантаження сторінки;
- Ефекти перекидання та меню, що випадає;
- Анімаційні елементи сторінки, такі як вицвітання, зміна розміру або переміщення;
- Відтворення аудіо та відео;
- Перевірка даних із форм;
- Багато інших ефектів.

JavaScript - це мова скриптів, яка вставляється безпосередньо в HTML документи. Це єдина мова програмування такого типу, яка буде зрозуміла Web-браузерам. Браузери можуть зчитувати Javascript, інтерпретувати його, а потім запускати програму, створюючи потужний User Experience.

Цього статусу мова досягла, оскільки це відкрита, стандартизована та, головне, дуже гарна мова. Він добре підходить для Web-розробки своїм динамічним характером та тісною інтеграцією з DOM.

JavaScript прекрасно підходить для використання із іншими мовами. Це дуже важливо для таких мов програмування, як PHP, Python, Ruby, Java чи .NET. Завдяки тому, що JavaScript запускається через браузер, неважливо на якому пристрої запускається додаток, він усе одно буде виглядати однаково для кожного із користувачів.

Рис. 2.8 Статистика використання JavaScript у Web-розробці

Можна сказати що JavaScript – це мова потужного, сучасного FrontEnd і деякою мірою навіть BackEnd різноманітних Тривалий час веб-сайти в основному будувались на основі PHP CMS, наприклад WordPress. Код на стороні сервера обробляв основну частину логіки. Як відомо, мода йде по круговій, тому зараз йдуть розмови про те, що статичні Web-сайти повертаються. Але це зовсім не ті ресурси до яких більшість людей могла звикнути у 90-ті роки. Зараз все стає куди цікавіше і різноманітніше. Використовуються зовсім інші технології, котрі дозволяють добитись неймовірних результатів з точки зору створення якісних ресурсів в інтернеті.

Сучасні браузери тепер мають можливість з легкістю відтворювати динамічні та інтерактивні сторінки. Логіка обробляється на стороні клієнта безпосередньо в браузері завдяки JavaScript.

Деякі з найвизначніших веб-додатків сьогодні створені за допомогою JS. Подумайте про Facebook, Gmail, Twitter та багато іншого. Якщо ми використовуємо Facebook як приклад, JavaScript дозволяє оновлювати статус та більшість інтерактивних можливостей користувачів. Без нього не можливо було б досягнути потрібного рівня привабливості Web-додатку.

Ці технічні гіганти фактично створили власні фреймворки на основі JavaScript, і тепер вони дозволяють тисячам розробників створювати власні Web-додатки. Можна згадати такі відомі фреймворки, як Angular, підкріплений Google, і React, підтриманий FaceBook. Завершує тріаду звісно ж Vue JS. Це саме той фреймворк, який нещодавно став дуже популярним серед Web-розробників.

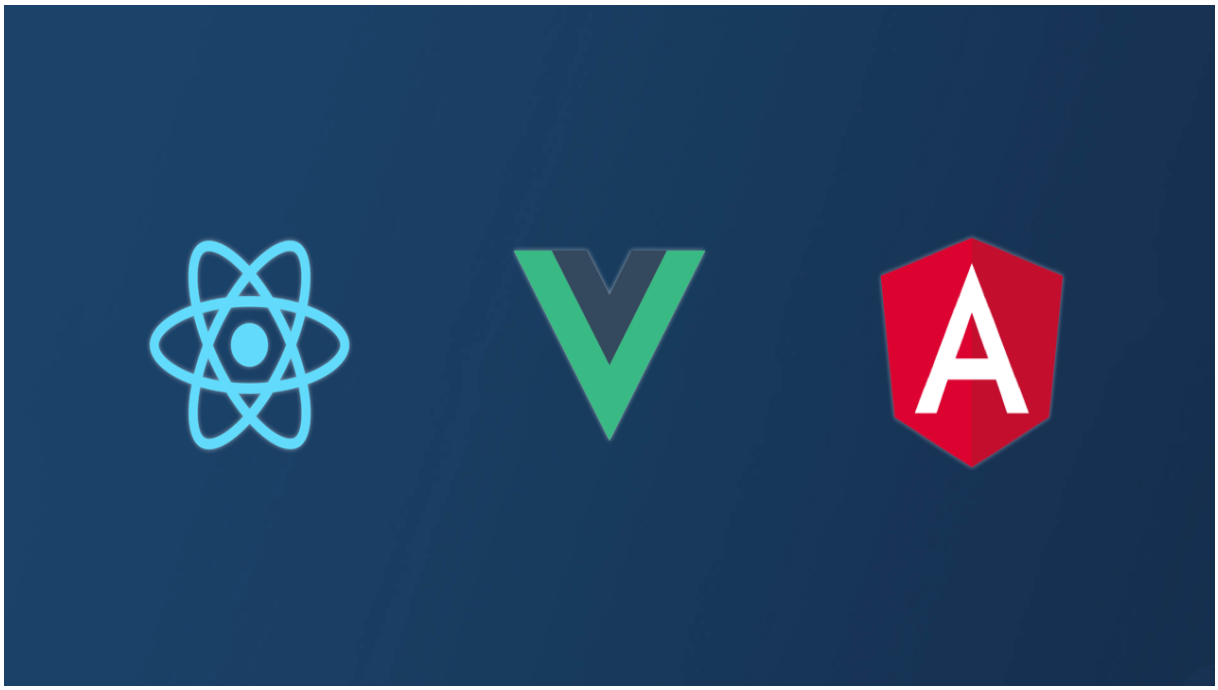

Рис. 2.9 Три найбільш популярні фреймворки JavaScript. «React JS», «Vue JS» та «Angular»

Окрім скорочення кількості часу та зусиль, необхідних для розробки Web-сайтів та додатків на базі JS, ці фреймворки допомогли сформувати новий досвід роботи в Інтернеті. Візьмемо, наприклад, програми на одній сторінці (SPA). SPA - це Web-сайт, який взаємодіє з користувачами, динамічно переписуючи сторінку у Web-переглядачі, а не завантажуючи цілі нові сторінки з сервера, змушуючи їх вести себе як настільні програми.

Javascript та робота із сервером

Були невдалі спроби змусити JavaScript працювати на сервері. Багато хто думав, що він ніколи не стане стабільною базовою мовою для роботи із серверами, аж до приходу Node.js.

Сьогодні цей час виконання JS є популярним інструментом для запуску Web-серверів. Це означає, що розробники JS можуть використовувати Node.js для запису коду на стороні клієнта та сервера в JavaScript, не покладаючись на зовнішні веб-сервери.

Мобільні додатки

Раніше вам потрібні були б інші мови для розробки чудових мобільних додатків, таких як Objective-C для iOS або Java для Android. Зараз можна використовувати JavaScript для підключення до мобільних інтерфейсів API простіше, ніж раніше це можна було собі уявити. Це означає, що ви можете використовувати функції мобільних пристроїв, такі як камера або локалізація для створення додатків, що працюють на JS.

Вкотре це відкрило розробку мобільних додатків для більш значної кількості розробників, яким більше не потрібно вивчати нову мову.

Не тільки це, але використання JavaScript у мобільних додатках навіть відкрило нові можливості, щоб зробити їх ще ефективнішими. Як приклад – Web-програми (PWA). Поєднуючи найкраще в Інтернеті та найкращі програми, PWA покращують надійність, продуктивність та взаємодію. Вони дозволяють впроваджувати нові функції, такі як офлайн-навігація.

Інтеграція із API

Розробники можуть використовувати JavaScript для отримання інформації із різних ресурсів, а потім відображати їх на Web-сайті. Однією з концепцій, яка більше, ніж будь-коли, просувається у Web-розробці, є модульність – використання різних інструментів для виконання конкретних завдань. Зараз легко створювати такі стеки завдяки API та JavaScript.

З цього випливає ще декілька переваг JavaScript, які змушують Web-розробників використовувати саме цю мову:

- Виконання логіки на стороні клієнта приносить швидший User Experience. Коли код працює безпосередньо в браузері, потреба у викликах сервера обмежується, отже, скорочення часу завантаження. Навіть при наявності сервера факт, що JS є асинхронним, означає, що він може спілкуватися з сервером у фоновому режимі, не перериваючи взаємодії з користувачем, що відбувається в інтерфейсі.
- З самого початку JavaScript впроваджував інтерактивність користувача в Інтернеті. Зараз це робиться так само для

програм будь-якого типу, допомагаючи розробити найпривабливіший UX. Сьогодні такі фреймворки, як Vue.js, виводять переходи та анімації на новий рівень.

- JavaScript стоїть за будь-яким хорошим адаптивним Web-дизайном. Дедалі більше розробників потребує адаптації свого дизайну в різних браузерах та пристроях. Поєднуючи HTML5, CSS3 та JavaScript, вони можуть робити це в форматі єдиного документу.
- Для розробників JS доволі легкий в освоєнні та одразу готовий до використаня. Його синтаксис легкий та гнучкий для новачків. Це також спрощує розробку складних додатків, дозволяючи розробникам спрощувати склад програми. Безліч фреймворків та бібліотек також полегшують життя розробників.
- JavaScript є шалено популярним. Це означає хоча б одне: можна легко знайти рішення будь-якої проблеми, яка могла виникнути під час розробки якогось серйозного проекту. У Web-розробці.

Потенційні проблеми, котрі можуть виникнути під час використання JavaScript у проекті

Досі не зрозуміло, в якій мірі пошукові системи можуть сканувати JavaScript. Навіть незважаючи на те, що Google стверджує, що сканується не весь документ, потрібно перестраховуватись і виправляти проблеми з SEO самостійно. Ця проблема ще й досі не є вирішеною і тому представляє собою одну із найнеприємніших, коли діло йде про використання JavaScript у своєму проекті.

Є можливість переборщити із використанням JavaScript. Потрібно вміло працювати з оптимізацією коду і використовувати таку кількість JavaScript, яка б не обтяжувала проект загалом. Добитись такого ефекту більш ніж можливо, якщо використовувати сучасні фреймворки та бібліотеки.

Тисячі бібліотек, що складають екосистему JS, дозволяють розробникам швидко працювати, не винаходити колесо для кожного нового завдання. Однак вони також спричиняють те, що дехто називав «пекло залежності». Потрібно навчитися боротися з цими часто необхідними залежностями, щоб вони не стали клопотом для процесу вдосконалення проекту у майбутньому.

2.4 Популярні JavaScript фреймворки

ReactJS є бібліотекою з відкритим кодом для фронтенду. Він використовується спеціально для великих, складних Web-інтерфейсів, а також для односторінкових програм. Створений Джорданом Уолком, інженером-програмістом у Facebook, він був швидко впроваджений у стрічці новин Facebook у 2011 році. Через рік пішов Instagram.com у

Facebook, і тоді все почалося. У наші дні цією бібліотекою користуються сотні тисяч (якщо не мільйони) Web-сайтів по всьому світу.

Насправді, з моменту запуску React, можна було спостерігати вибухонебезпечне зростання використання легких, але потужних бібліотек JavaScript. Користувачі все частіше хочуть користуватися швидшими, динамічнішими Web-сторінками, тоді як розробники вибирають сучасні та гнучкі середовища без купи проблем під час розробки. Ось чому ReactJS - очевидний вибір для багатьох. Можна розглянути основні переваги цього фреймворку.

Рис. 2.10 Загальноприйнятий логотип ReactJS

Найперша причина використання ReactJS – простота. Саме через це більшість розробників почали використовувати ReactJS як основну бібліотеку JavaScript. Якщо Web-розробник знайомий із функціями JavaScript, то розібратись із бібліотекою ReactJS можна буде без проблем.

За допомогою цієї бібліотеки розробники визначають інтерфейси з HTML-синтаксисом, який називається JSX. В результаті утворюється HTML та CSS-код. API React дуже маленький, але потужний, і все, що вам потрібно зробити, перш ніж почати - це вивчити кілька основних функцій.

Проблеми з вивченням бібліотеки виникають тоді, коли потрібно використовувати React з іншими бібліотеками JS, такими як Redux, Material UI або Enzyme. Хоча вони не є частиною стека React, такі бібліотеки додають додаткові можливості та дозволяють легше керувати компонентами React. Найпоширеніші бібліотеки добре задокументовані і не повинні створювати проблем жодному розробнику.

Простота React має певні переваги для бізнесу. Ті, хто вже покладається на інші рамки JavaScript, матимуть легший перехід до React. Навіть якщо ваша мета - просто спробувати бібліотеку, не здійснюючи

повноцінний перехід, це також не буде складно. І якщо вам потрібно буде масштабувати в майбутньому або змінити команду розробників, велика кількість розробників ReactJS на ринку зробить все набагато простіше.

```
const AppHeaderBlock = () => (
  <div>
    <brandedheader>Inventory</brandedheader>
    <ul classname="menu">
      <li classname="nav-item">
        <button onclick="{gotoHome}">Home</button>
      </li>
      <li classname=""nav-item"">
        <button onclick="{gotoAccount}">Account</button>
      </li>
    </ul>
  </div>
)
```

Рис. 2.11 Приклад компоненту ReactJS, імплементованого в JSX

Дослідження, проведене StateofJS.com, показує, що розробники JavaScript дуже раді використовувати React у своїх проектах, вигідно порівнюючи його з найближчими конкурентами, такими як Angular або Vue.js. Більше того, під час написання, майже дві третини запитаних відповіли, що вони використовують ReactJS і знову використовуватимуть його.

Це має велике значення для тих, хто планує запустити будь-який програмний продукт протягом наступних кількох років. Якщо розробники так захоплюються React, це означає, що їх буде все більше і більше. Це, як наслідок, означає, що буде велика екосистема різних бібліотек та інструментів React для посилення розвитку. Як результат, ReactJS стане більш потужним з кожним розробленим Web-додатком.

Кожен проект React будується з використанням так званих компонентів для багаторазового використання. Це означає, що кожен елемент інтерфейсу, який вже був створений, може використовуватися будь-якому місці вашого проекту, просто викликавши його з інших компонентів. Написавши функцію один раз, її можна повторно використовувати у майбутньому. Більше того, можна згрупувати ці елементи у розділи чи сторінки та вставити їх у будь-яку точку проекту з однаковими функціями.

Елементи багаторазового використання в ReactJS також дотримуються простої ментальної моделі. Якщо компонент отримує однакові дані кілька разів, він завжди отримує той самий набір HTML і CSS. Така здатність, яка називається ідентифікацією, дозволяє легко міркувати і перевіряти.

ReactJS також дозволяє швидко перебирати будь-який елемент. На те щоб змінити основний колір Web-сайту потрібно всього кілька секунд.

Теж саме стосується переадресації і всіх інших елементів ресурсу. Будь-яку задачу можна виконати в рази швидше, ніж будь коли.

Логіка, що стоїть за кожним компонентом, може бути визначена розробником один раз, і React просто використовуватиме її для відображення елементів саме там, де вони б мали знаходитись згідно проекту.

Революційною річчю, що була представлена разом із React, була можливість створювати мобільні додатки для iOS та Android, не потребуючи наймання нових розробників. Порт React для мобільних пристроїв, відомий як React Native, дозволяє писати повнофункціональні мобільні додатки за допомогою Javascript. Це дозволяє розробникам повторно використовувати код з вашого веб-додатку в ReactJS, щоб прискорити розробку мобільних пристроїв. Розробник ReactJS також може використовувати різні бібліотеки JS, з якими вони вже знайомі. Після того, як спільна кодова база готова, розробник починає створювати вбудовані компоненти iOS або Android. Незважаючи на відмінності між обома платформами, хороший розробник React може обробляти всі платформи після невеликих досліджень.

ReactJS багато в чому є революційним підходом до створення як програм для десктопних приладів, так і для мобільних пристроїв. Він швидко завоював прихильність розробників JavaScript, і схоже, що так буде в усі наступні роки.

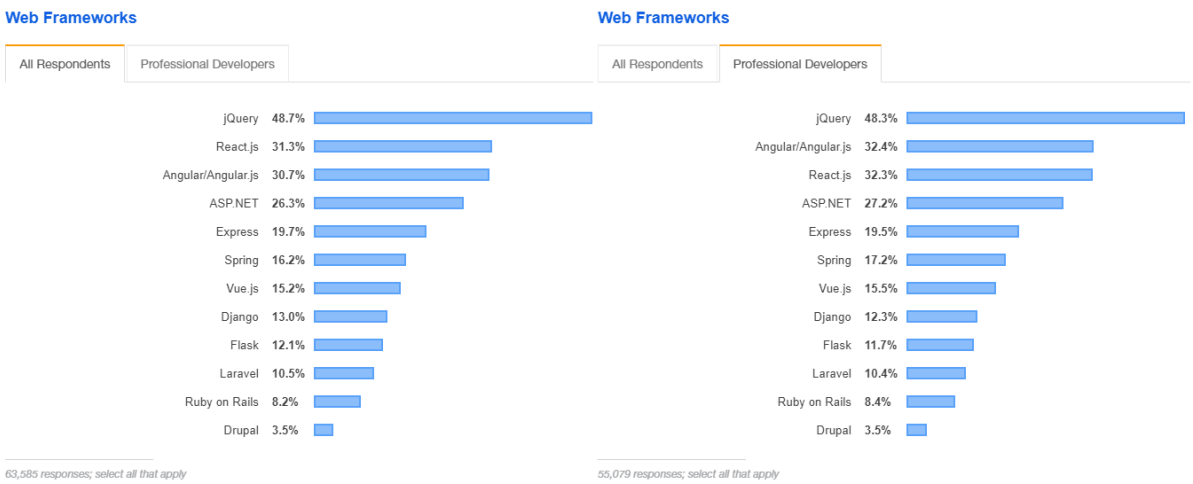

Рис. 2.12 Таблиця популярності Web-фреймворків

Vue.js - це прогресивний фреймворк JavaScript (звідси букви "JS"), який використовується для побудови інтерфейсів користувача.

На відміну від деяких інших популярних фреймворків, він не був розроблений за підтримки великої компанії. В той же час React було створено за підтримки Facebook, а Angular має підтримку Google, Vue.js повністю створений та підтримується незалежними розробниками.

На відміну від звичних фреймворків на ринку, Vue.js можна використовувати у відрізі від окремої бібліотеки.

Не обов'язково створювати проект заново, тільки щоб адаптувати його під Vue.JS. Насправді досить легко можна імплементувати цей фреймворк у вже готовий проект і тим сами досягти значно кращих результатів за короткий срок. Тому що з використанням данного фреймворку можна покращити Web-інтерфейс і зробити його більш прийнятним для сучасного користувача.

Є декілька причин чому саме Vue.JS можна вважати одним із найкращих Web-фреймворків на сьогоднішній день.

Vue.js гнучкий і зручний у використанні. На практиці це означає, що його можна використовувати для величезного, модульного SPA (Single Page Apps), а також для побудови невеликих, інтерактивних частин, які інтегруються за допомогою іншої технології. Іншими словами, це може бути все: просто бібліотека у проекті або повнофункціональний фреймворк, який використовується для створення всього продукту.

Написання типового коду – це витрата часу та ресурсів. Vue.js допомаже уникнути зайвої роботи, надаючи безліч вбудованих рішень та економію сил. Для того щоб заново не створювати колесо, використовують Vue.js, в якому вже є рішення типових проблем.

Актуальність технології в майбутньому – та частина про яку зазвичай забувають коли обговорюють переваги фреймворку.

Після того як ви створили та запустили Web-додаток, також доведеться подумати щодо виправлень помилок, нових функцій та інших удосконалень. Vue.js спрощує оновлення. Розробники бібліотеки це передбачили і тому надали підтримку винулих версій фреймворку, як тільки відбувається оновлення (90% між версіями 1 та 2). Це дозволяє не так сильно турбуватись щодо коду, який з часом може стати просто неактуальним.

Крім того, проект, що базується на Vue, не потребуватиме максимального швидкого оновлення, звільняючи розробників від необхідності щоразу вносити зміни в код побудованого Web-додатку.

Розробники люблять Vue.js не тільки тому, що це чудова технологія, а й тому, що цей фреймворк був розроблений спеціально для них.

По-перше, існує vue-cli (CLI - «інтерфейс командного рядка»), зручний інструмент, який дозволяє тривіально легко запустити (а також налаштувати, запустити, проаналізувати та протестувати) новий проект за вибором інструментів. Vue-cli набагато гнучкіший порівняно з аналогічними пропозиціями конкурентів і забезпечує безліч попередньо налаштованих параметрів.

Також доступний графічний інтерфейс (Vue GUI), який допоможе розпочати роботу над проектом без введення команд в терміналі.

Існує досить детальна документація, за рахунок якої можна досить швидко освоїти принцип роботи Vue.js. Не потрібно дуже багато часу, щоб ознайомитися з цією технологією, а також не потрібно розбирати багато інструментів.

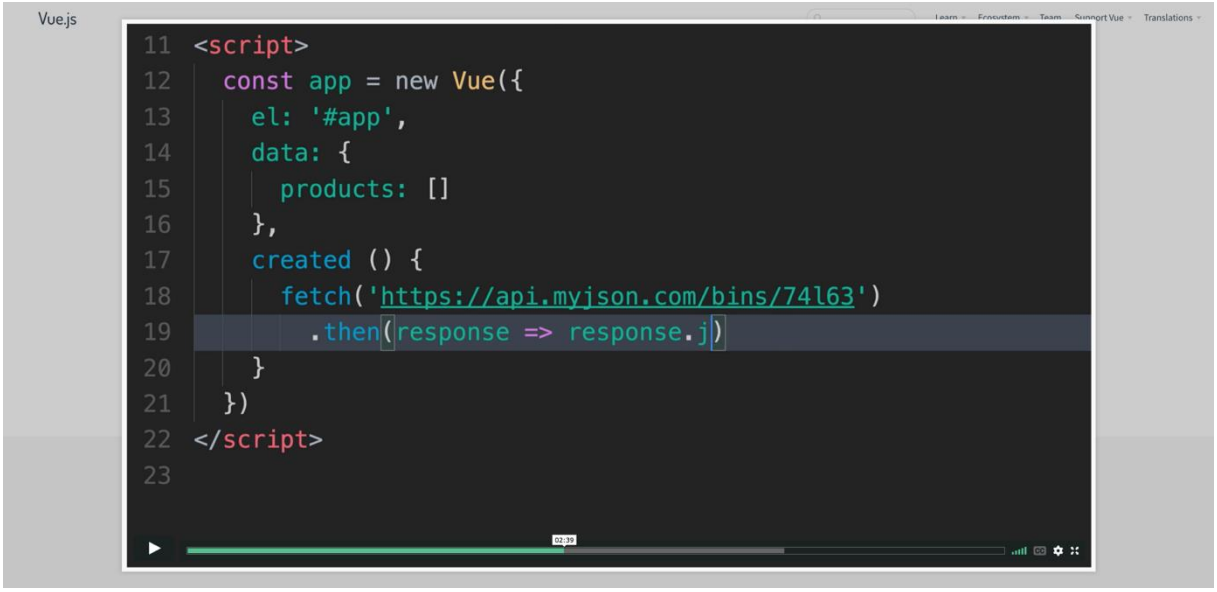

Рис. 2.13 Приклад запиту в Vue.js

Інша річ, яка робить Vue гарним інструментом для роботи - це той факт, що, оскільки нові версії випускаються за рахунок великого кола розробників, вони здатні враховувати власні помилки, та недоліки спільноти розробників при створенні нової версії.

Vue.js дійсно бере найкраще з інших фреймворків. По-перше, додатки Vue мають менші розміри (тому вони швидше завантажуються та використовують меншу пропускну здатність) і зазвичай ефективніші, ніж еквівалентні програми, створені за рахунок інших фреймворків. Це підтверджується реальними проектами, які були створені в останні кілька років.

Хоча ми розглядаємо питання продуктивності, Vue піклується про безліч оптимізацій, тому розробникам не потрібно перейматися налаштуванням програми під час її зростання. Вони можуть зосередитись на тому, щоб додати до нього нові, орієнтовані на цінність речі. Це також означає, що програми Vue є масштабованими: перенести їх із простого додатка на одній сторінці до розширеної системи доволі просто.

Якщо проект потребує збільшення продуктивності, переписання його у Vue може бути прекрасним рішенням проблеми.

Завдяки своєму прогресивному характеру, фреймворк можна вводити у код дуже просто – не потрібно переробляти все заново. Компонент за компонентом може бути поступово переведений у новий формат.

Завдяки численним перевагам Vue.js набирає популярність дуже швидко.

Vue.JS дозволяє дуже швидко писати додаток. Після цього його можна прямо з браузера. Це безумовно впливає також і на процес тестування. Складні програми, такі як ES6, JSX, компоненти, маршрутизація, пакети також можуть бути сконструйовані за допомогою Vue.js. Він пропонує безліч різних способів використання і виражає свою гнучкість, пропонуючи різні способи виразити свій код. Наприклад, можна написати простий шаблон в HTML, або в Javascript або просто використовувати JSX. Це дарує повну творчу свободу, тому що можна створювати додаток значно швидше, надаючи йому весь необхідний функціонал.

Функція Vuex може використовуватись для розробки більш великих і складних функцій потребуючих маршрутизації. Додаткові бібліотеки vue офіційно підтримуються та оновлюються з базовою бібліотекою. Redux і Flux - це розширення, які допомагають у подальшій розробці компонентів сайту чи додатку, котрі не мають реагувати на дії користувача одразу ж.

Vue.js має чудові інтеграційні можливості з існуючими програмами і є однією з причин його популярності серед розробників. Це пов’язано з тим, що він побудований на javascript і може бути легко інтегрований в інші додатки, побудовані і на javascript. Завдяки чому це дозволяє легко змінювати попередні програми, замість розробки нових додатків.

Vue – відносно свіжий фреймворк і має досить мало багажу в своїй бібліотеці. Цей мінімалізм і відсутність великої кількості минулих версій в рамках фреймворку є одним з найсильніших атрибутів. Завдяки якому vue має досить вузький варіант використання. Цей фокус дає можливість ігнорувати порив інших фреймворків. Розробники можуть легко вдосконалювати код на основі Vue.js, оскільки в бібліотеці немає нічого лишнього.

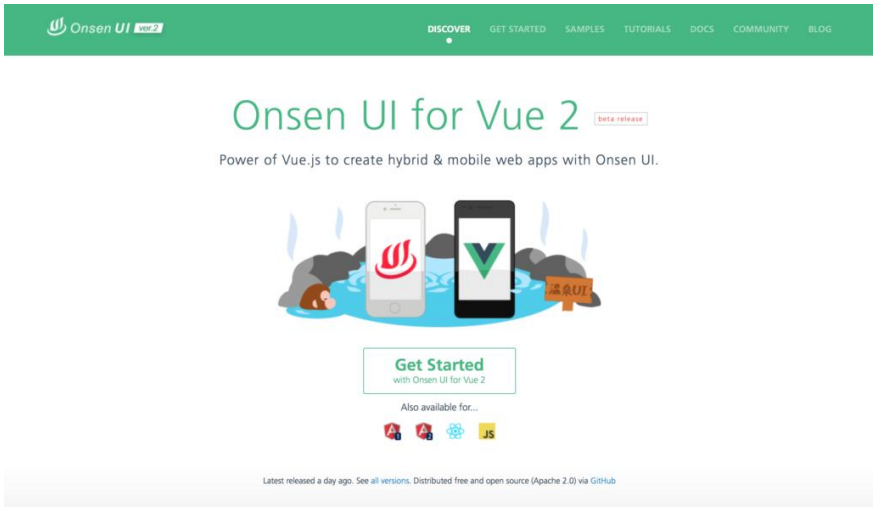

Рис. 2.14 Приклад сторінки Web-додатку, створеного завдяки Vue.js

Переваги використання Angular

Встановлюючи своє початкове середовище розробки Angular за допомогою Angular CLI, вже стає зрозуміло наскільки зручним і необхідним є використання даного фреймворку.

Сучасні Web-продукти потребують значно більше налаштувань, ніж будь-коли раніше - і це стосується не тільки Angular, але і будь-якої іншої еквівалентної екосистеми.

Angular полегшує розробку веб-додатків. Поєднуючи ін'єкцію залежностей, декларативні шаблони, комплексне інструментальне обладнання та інтегровані найкращі практики, він вирішує майже всі проблеми при створенні Web-додатку.

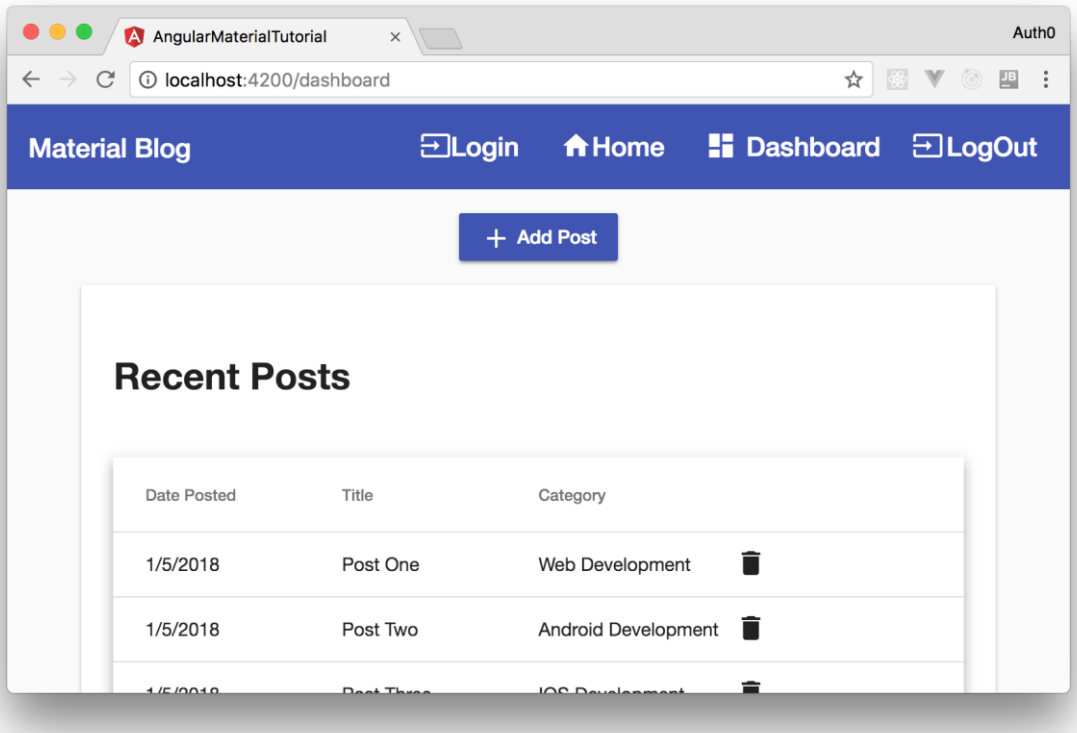

Рис. 2.15 Приклад Web-додатку створеного завдяки використанню Angular

Angular Framework включає версії від 2 до 8. 8-е оновлення було випущено в травні 2019 р. Можна виділити ключові особливості кожної із версій:

- Angular 2: Випуск Angular 2 приніс численні зміни в початковий фреймворк, оскільки він був переписаний у TypeScript. Архітектурний стиль перейшов на компонентний.
- Angular 4: Angular CLI 1.0.0 був представлений з четвертою версією, як основний елемент проекту Angular. З випуском програми Angular Universal, програми Angular можна винести за межі браузера.

- Angular 5-6: Випуск п’ятої та шостої версії сконцентрований на оптимізації Angular CLI та роботі компілятора.
- Angular 7: За допомогою Angular 7 CLI було покращено підказками, які давали поради в CLI для пояснення функцій та цілей елементів. Отже, використання CLI стало більш інтуїтивно зрозумілим. Програми отримали різні поліпшення продуктивності та розміру кодової бази.
- Angular 8: остання версія Angular: У Angular 8 були введені два елементи Ivy renderer, Bazel (інтерфейс побудови). Ще одним важливим поліпшенням є диференційоване завантаження, яке використовується для завантаження специфічних для браузера пакетів, щоб підтримувати застарілі браузери та швидше завантажувати вміст.

Кожна технологія має певні переваги та мінуси. Можна навести декілька прикладів переваг і недоліків, які присутні у Angular.

Явні переваги Angular

Впровадження архітектури MVC

Архітектура Model-View-Controller не тільки надає значення фреймворку під час створення додатка на стороні клієнта, але й встановлює основу для інших функцій, таких як прив'язка даних та сфери застосування.

За допомогою архітектури MVC можна ізолювати логіку програми від рівня інтерфейсу та підтримувати розділення проблем. Контролер отримує всі запити на додаток і працює з моделлю для підготовки будь-яких даних, необхідних для перегляда. Переглядач використовує дані, підготовлені контролером, і відображає остаточну презентабельну відповідь.

Розширена архітектура дизайну

Деякі з великих Web-додатків містять безліч компонентів. Angular спрощує спосіб управління цими компонентами, навіть якщо новий програміст приєднається до проекта після того, як процес розробки вже розпочався. Архітектура побудована таким чином, що допомагає програмісту легко знаходити та розробляти код.

Модулі

Модуль – це механізм, який об'єднує пов'язані директиви, компоненти, труби та сервіси таким чином, що їх можна комбінувати з іншими модулями для створення програми. Додаток на основі Angular може розглядатися як головоломка, де кожен модуль є необхідним, щоб мати можливість побачити повну картину. Існує ряд способів додавання різних елементів до модуля. Angular вирішує проблему глобальної експлуатації функцій, обмежуючи сферу застосування всіх функцій модулем, в якому вона була визначена та використовується.

Angular пишеться за допомогою TypeScript, що є суперсетом JavaScript. Він повністю відповідає JavaScript, а також допомагає визначити та усунути поширені помилки під час написання коду. Незважаючи на те, що невеликі проекти JavaScript не потребують такого вдосконалення, для корпоративних програм потрібні розробники, щоб зробити свій код чистішим і частіше перевіряти якість.

Явні недоліки Angular

Є одразу декілька явних недоліків, які присутні в Angular. Та це не означає що вони роблять цей фреймворк поганим. Але згадати про них потрібно, оскільки вони мають вплив на сам процес розробки.

Кутовий є багатослівним і складним

Часта скарга, яку можна почути від розробників Angular, – це різноманітність цього інструменту. І ця проблема не сильно змінилася з часу першої версії AngularJS.

Складність вивчення

Якщо потрібно найняти нових розробників для проекту, які б знали JavaScript та при цьому не мали достатнього досвіду в роботі з Angular, то навчитись їм буде доволі складно. Занадто великий об’єм тем та аспектів цього фреймворка не дозволить швидко розібратись із проектом та вникнути у розробку. Тому варто правильно підійти до процесу розробки, працюючи лише з тими спеціалістами, котрі добре володіють знаннями Angular [Рис. 2.16].

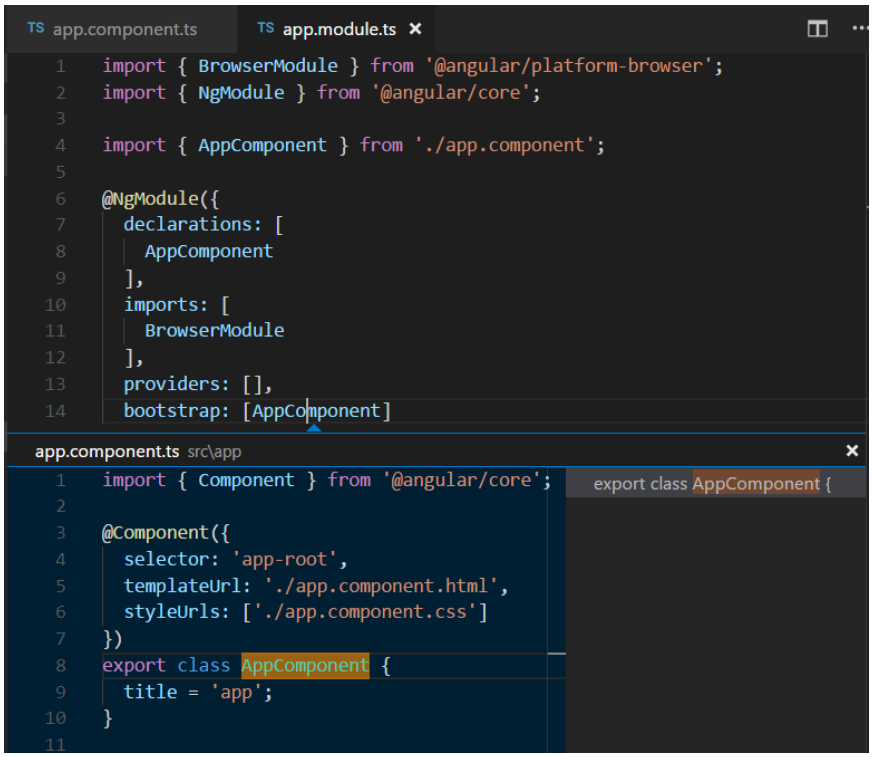

Рис. 2.16 Приклад типового коду Angular

Документація CLI не є достатньо детальною

Деякі розробники висловлюють стурбованість поточним станом документації CLI. Хоча командний рядок дуже корисний для Angular розробників, неможливо знайти достатню кількість інформації на GitHub. Доведеться витрачати лишній час на те, щоб знайти якомога кращий ресурс для вивчення Angular.

Хоча платформа має свою частку мінусів, Angular є повнофункціональною та динамічною основою для будь-якого сучасного проекту. А його зручність, гнучкість – робить його унікальним і надає шанси створити чудові та успішні Web-додатки.

ВИСНОВОК ДО РОЗДІЛУ 2

Цей розділ демонструє технології, які будуть використовуватись при розробці Web-додатку «Рекомендаційна система». Використовуючи HTML у поєднанні з CSS, JavaScript та гарним фреймворком – можна легко досягти необхідного результату досить швидко та якісно. Головне одразу правильно обрати технології, щоб у подальшому не виникало складнощів з удосконаленням проекту. Усі перераховані у розділі технології впливають на якість Web-додатку. Можна сказати, що вони закладають фундамент майбутньої розробки. У моєму випадку я постарався обрати ті технології, які б дозволили реалізувати інтерактивність Web-додатку і простоту використання на будь-якому пристрої. Адаптивність і простота проекту – ключові задачі, оскільки більш сучасних користувачів інтернету використовують саме мобільні пристрої, а не ПК.

Розділ 3. Аналіз типів рекомендаційних систем

Рекомендаційні системи визначаються як аналіз початкової інформації, що задана користувачем, оброблений системою і потім направлений до кінцевого користувача. Крім того, вона може бути визначена як система, яка виробляє індивідуалізовані рекомендації, або має ефект орієнтації користувача на персоналізований шлях до цікавих йому об'єктів у більшому просторі можливих варіантів. Система рекомендацій стане невід'ємною частиною галузі засобів масової інформації та розваг найближчим часом. В основному існує шість типів систем рекомендування, які працюють в основному в галузі засобів масової інформації та розваг: Спільна система рекомендацій, Система рекомендацій на основі контента, Система рекомендацій на основі демографічних даних, Рекомендація на основі утиліти, Рекомендаційна система на основі знань та Гібридна система рекомендацій. Потрібно розглянути основні з них, щоб зрозуміти яка має найбільше переваг у випадку реалізації Web-додатку.

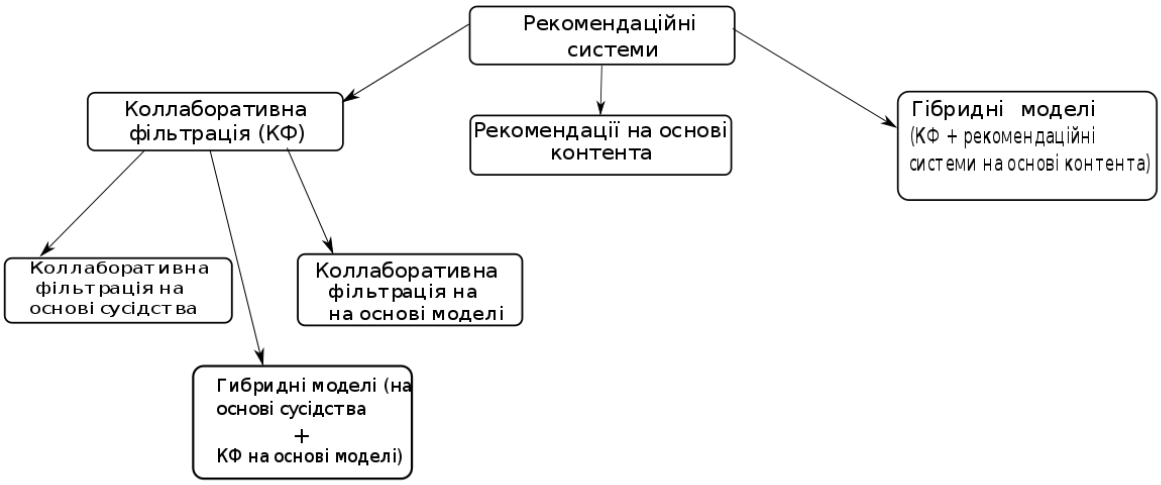

Рис. 3.1 Приклади різних рекомендаційних системи

3.1 Спільна (Коллаборативна) система рекомендацій

Системи рекомендацій використовуються в першу чергу для того, щоб користувачу не потрібно було витрачати власний час на обробку великого масиву інформації. Достатньо всього лише скористатись систему рекомендацій реалізованою у Web-додатку, щоб отримати підказку по знаходженню конкретного контенту, який би відповідав вподобанням користувача.

Одним із способів реалізації подібної системи є Спільна (Коллаборативна) система рекомендацій. Існує два основних підходи, за допомогою яких розробляється система рекомендацій:

1. Фільтрування на основі контенту;
2. Спільна фільтрація.

Методи, які вибірково використовують обидва підходи, називаються гібридними системами рекомендацій.

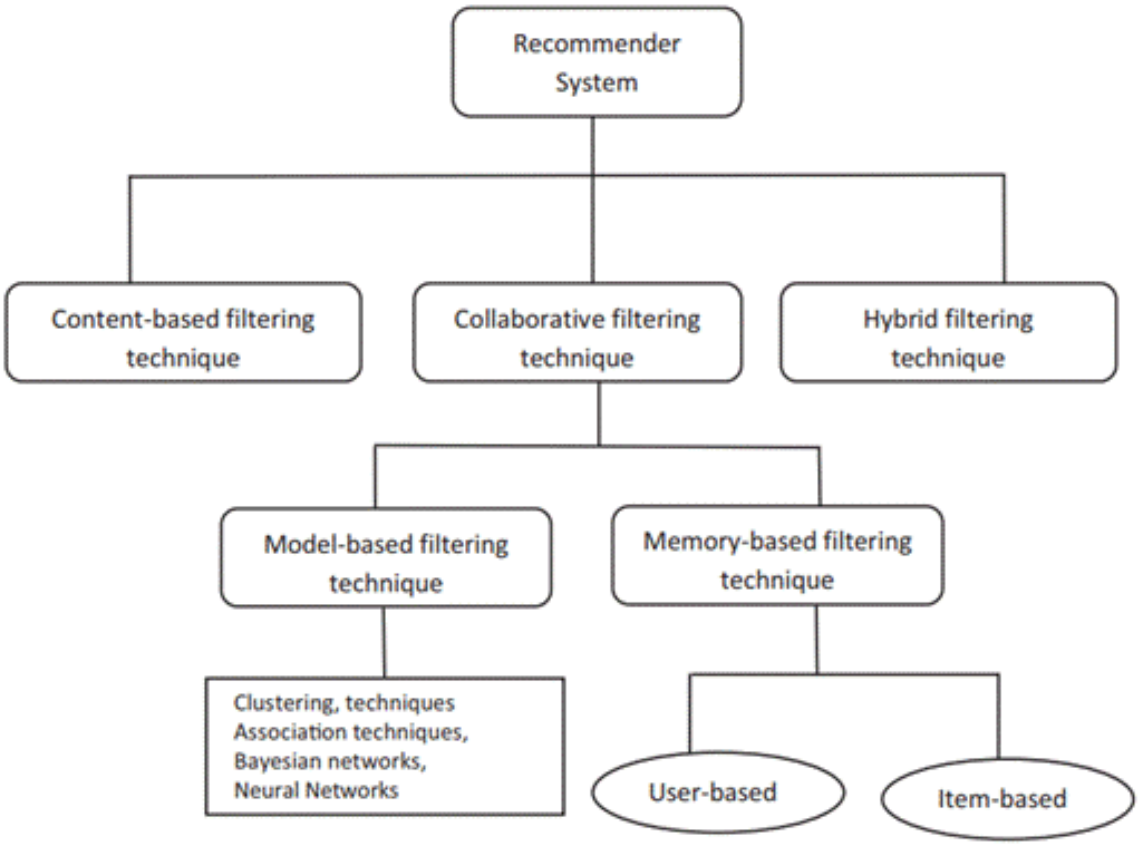

Рис. 3.2 Типи рекомендаційних систем

Розробники Xerox спочатку використовують спільну фільтрацію в системі пошуку документів. Алгоритм PageRank, використовуваний Google, є прикладом системи пошуку документів, що використовує спільну фільтрацію. Спільна фільтрація використовується для адаптації рекомендацій, що ґрунтуються на поведінці осіб з подібними інтересами. Іноді це може базуватися на предметі, купленому користувачем. Оскільки

цей метод не вимагає від самої людини завжди робити внесок у сховище даних, то порожні відділи бази даних можуть заповнюватися діями інших осіб / діями тієї ж особи щодо контенту у додатку. Кілька підходів щодо методів спільних рекомендацій щодо користувачів та предметів:

- Предметний підхід;
- Класифікаційний підхід;
- Нейронна спільна фільтрація.

Підхід від предмета до предмета

Замість використання рейтингу на основі оцінок різних користувачів ресурсу для обчислення сусідства, рейтинги використовуються для знаходження відповідностей між об’єктами.

Класифікаційний підхід

У класифікаційному підході, об’єкти представлені векторами і вони класифікуються на основі рейтингу серед активних користувачів ресурсу. Як тільки набирається необхідна кількість оцінок у кожного об’єкта формується власний клас. Таким чином реалізується класичний фільтр на основі класифікації.

Нейронна спільна фільтрація

Нейронні мережі все частіше використовуються для спільної фільтрації. Xiangnan HE та ін. досліджували використання нейронних мереж для спільної фільтрації. При цьому дані матриці взаємодії між елементами користувача трактуються як неявні дані. Хоча спостережувані записи принаймні відображають інтерес користувачів до предметів, незарезервовані записи можуть бути просто відсутніми даними, і в більшості випадків існує природний дефіцит негативних відгуків. Проблема рекомендацій із неявним зворотним зв'язком формулюється як проблема оцінки балів незабезпечених записів. Матрична факторизація використовується для оцінки прогнозованого випуску. Відсутні дані замінюються за допомогою цього вводу. Потім заповнений вхідний простір передається в багатошарову персептронну мережу для оцінки рейтингів для активного користувача.

Багато сучасних компаній використовують саме нейронну рекомендаційну систему. Вони досить точно вказує на контент який може сподобатись конкретному користувачу. Але варто відмітити той факт, що далеко не всі додатки однаково добре підходять для реалізації подібної системи. Потрібно враховувати специфіку конкретного ресурсу. Оскільки контент на основі якого буде здійснюватися прогнозування може суттєво відрізнятись. Тому потрібно серйозно підійти до цього питання і провести детальний аналіз основи для побудови системи. Інакше можна втратити можливість гідним чином створити рекомендаційну систему для свого проекту.

Рекомендаційна система онлайн сервісу YouTube є прикладом використання нейронної мережі.

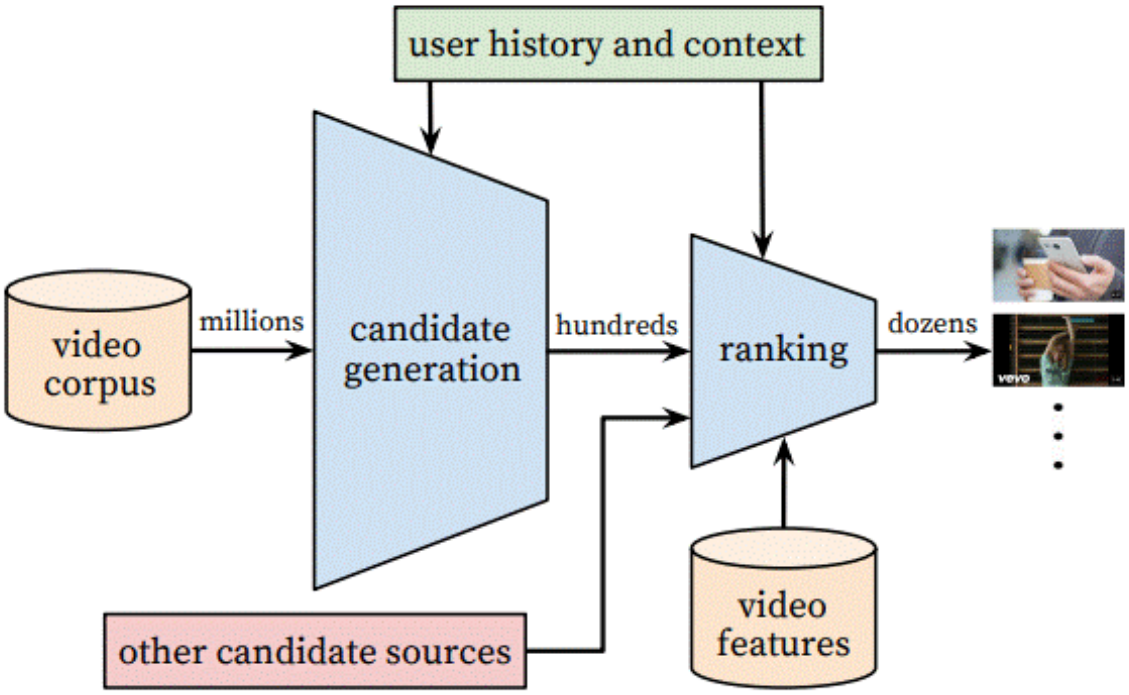

Рис. 3.3 Схема роботи рекомендаційної системи сервісу YouTube

Ця система використовує дві окремі нейронні мережі. Перша мережа використовується при генерації кандидатів, тоді як наступна мережа використовується для ранжирування. Мережа генерації кандидатів забезпечує лише широку персоналізацію за допомогою спільної фільтрації. Ця мережа використовує особисту історію користувача, щоб запропонувати користувачеві кілька сотень відео з величезного масиву відеороликів сервісу YouTube. Другий фільтр використовує розширений функціонал сервісу, включаючи опис відео та користувача для ранжування відео за пропозицією.

Підхід спільної фільтрації працює краще, ніж простий метод рекомендацій на основі вмісту, коли доступні величезні дані користувачів. Цей метод не виявляється корисним у випадку відеороликів. Це відомо як "проблема холодного старту".

Можна виділити одразу декілька переваг використання спільної системи рекомендацій:

- Чим більша база користувачів, тим краще працює система. Простіше кажучи, чим більше людей користується послугою, тим краще будуть видаватись рекомендації, без проблем самостійної обробки конкретної області знань. Достатньо того, що користувачі будуть

самостійно оброблювати данні і тим самим надавати інформацію про продукт іншим клієнтам;

- Система може використовуватись у різних областях. Підходи спільної фільтрації добре підходять для дуже різноманітних галузей. Якщо фільтри на основі контенту покладаються на метадані, спільна фільтрація базується на реальній діяльності, що дозволяє їй здійснювати зв'язки між начебто різними предметами, які можуть зацікавати одну і ту ж групу користувачів;
- Така система надає більше різновидів рекомендацій. Що стосується рекомендацій, точність не завжди є найвищим пріоритетом. Підходи до фільтрування на основі контенту, як правило, показують користувачам елементи, які дуже схожі на предмети, які їм вже сподобались, що може призвести до проблем з фільтруванням бульбашок. Навпаки, у більшості користувачів є інтереси, що охоплюють різні підмножини, що теоретично може призвести до більш різноманітних (і цікавих) рекомендацій.
- Система може враховувати більше нюансів щодо об’єктів. Навіть високо деталізована система фільтрації на основі вмісту дозволить зафіксувати лише деякі особливості певного елемента. Спираючись на фактичний досвід людини, спільна фільтрація може іноді рекомендувати предмети, які мають більшу спорідненість один з одним, ніж це дозволило б суворе порівняння їх атрибутів.

Рис. 3.4 Система рекомендацій в рітейлі

Існує два підходи до спільної фільтрації, один заснований на елементах, а другий - щодо користувачів. Спільне фільтрування «Елемент-Елемент» спочатку було розроблено Amazon і дозволяє робити висновки про взаємозв'язок між різними предметами, на основі яких товари купуються разом. Частіше два предмети (скажімо, арахісове масло і желе) з’являються в одній і тій же кошику або історії користувачів, «ближче», які вони шукають, або купували раніше. Отже, коли хтось приходить і додає арахісове масло до свого кошика, алгоритм підкаже речі, близькі по духу, як желе чи білий хліб. Такі предмети будуть виводитись в першу чергу. А от моторне масло показуватись в цій категорії взагалі не буде.

Фільтрування «Елемент-Користувач» застосовує дещо інший підхід. Тут, замість обчислення відстані між елементами, ми обчислюємо віддаленість між користувачами на основі їх оцінок (чи подобається, або будь-якого показника). Розглядаючи рекомендації для конкретного користувача, ми переглядаємо найближчих до них користувачів, а потім пропонуємо предмети, які також сподобалися користувачам, але з ними наш користувач ще не взаємодіяв. Отже, якщо переглянути декілька відео у Facebook, та поставити під ними лайк, Facebook може переглядати інших користувачів, які сподобалися цим самим відео, та порекомендувати той, який їм також сподобався, але якого ви, можливо, ще не бачили.

Важливим моментом є те, що в обох наведених вище прикладах система не має уявлення, чому будь-який із цих предметів пов'язаний один з одним, знає лише те, що вони або відображаються в одному кошику разом, або що їм подобаються люди з подібними уподобаннями. В деяких випадках, проте, це може бути особливістю, а не недоліком, особливо у випадках, коли предмети, які підлягають фільтруванню, надзвичайно неоднорідні, як у інтернет-магазинах чи соціальних мережах. Але іноді такі системи давали збій. Наприклад у Amazon нерідкими були випадки, коли покупцям хлібу пропонували купити якісь медичні препарати.

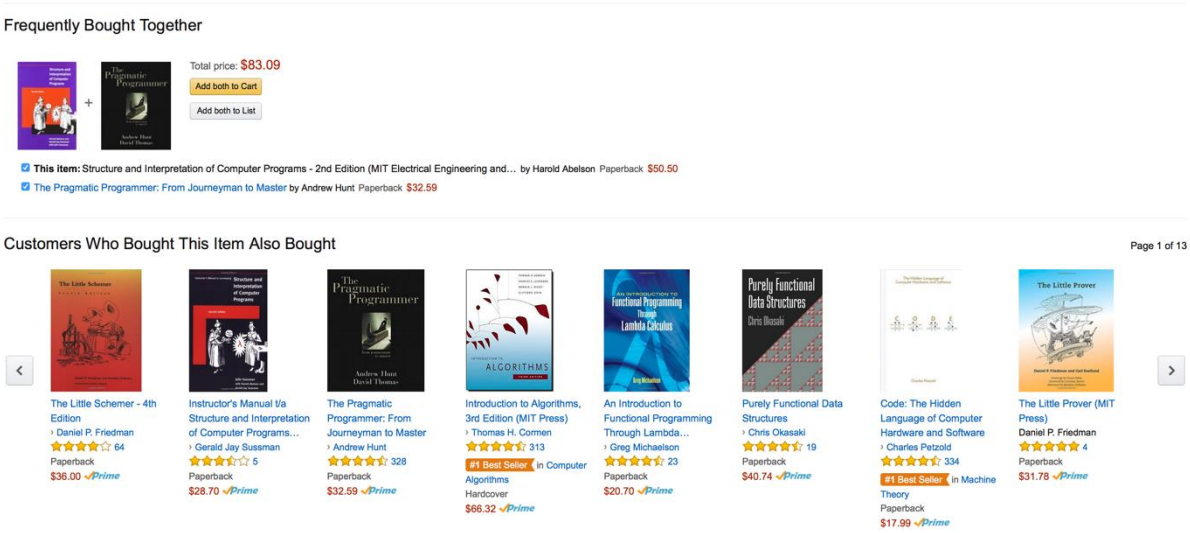

Рис. 3.5 Рекомендаційна система компанії Amazon

Наведені вище описи мають на увазі загальний огляд того, як зазвичай застосовуються методи спільної фільтрації. За кожною реалізацією існує ряд різних методик вимірювання подібності двох різних елементів або користувачів. Яка з них підходить для даної системи рекомендацій, залежить як від випадку використання, так і від характеру залучених даних.

Проблеми спільної фільтрації

Складність та витратність. Алгоритми спільної фільтрації можуть зіткнутися з проблемами масштабованості, коли кількість користувачів і елементів стає надто великою (подумайте про десятки мільйонів користувачів та сотні тисяч елементів), особливо коли рекомендації потрібно генерувати в режимі реального часу в Інтернеті. Потенційне рішення: саме тут корисні розподілені кластери машин під управлінням Hadoop або Spark. Залежно від проекту, також можна розрахувати відносини в режимі офлайн протягом ночі шляхом пакетної обробки, що робить рекомендації щодо обслуговування набагато швидшими, навіть якщо вони більше не оновлюються в режимі реального часу.

Рідкість даних. Багато зворотніх сигналів користувача неоднозначні. Просто перегляд відео не означає YouTube, чи сподобалось вам це відео чи ні, а просто поїсти в ресторані не дає данні, сподобалось вам це чи ні. Ось чому рейтинги настільки важливі в системах спільної фільтрації. Але користувачі не оцінюють кожен предмет, з яким вони взаємодіють, і багато користувачів взагалі нічого не оцінюють. Потенційне рішення: Залежно від характеру даних, можуть застосовуватися проксі-заходи, які можна використовувати. Ще одна поширена методика – припустити, що відсутні відгуки еквівалентні середнім відгукам, хоча в більшості випадків це дуже сильне припущення.

Проблема "холодного старту". Як ми бачили, спільна фільтрація може бути потужним способом рекомендувати елементи на основі історії користувачів, але що робити, якщо немає історії користувача? Це називається проблемою «холодного старту», і вона може стосуватися як нових елементів, так і нових користувачів. Елементів з великою кількістю історії рекомендується багато, тоді як ті, хто ніколи не вносити її в систему рекомендацій, призводять до позитивного циклу зворотного зв'язку. У той же час, нові користувачі не мають історії, тому система не має добрих рекомендацій. Потенційне рішення: Процеси на борту можуть дізнатися основну інформацію, щоб скористатися налаштуваннями користувачів, імпортуючи контакти у соціальній мережі.

Побудова системи рекомендацій із спільною фільтрацією є складним проектом, що включає як наукові дані, так і технічні проблеми. Для вирішення цих завдань може знадобитися досвід роботи з структурами обробки та зберігання даних, такими як Hadoop або Spark. Деякі проблеми можуть бути вирішені завдяки використанню спеціальних технологій. На

сьогоднішній день розробляється все більше способів правильної роботи з контентом.

3.2 Рекомендаційна система на основі контенту

Однією з популярних методик рекомендаційних систем є фільтрування на основі контенту. Вміст тут атрибутів продуктів, які подобаються користувачу. Отже, ідея фільтрації на основі вмісту полягає в тегуванні продуктів за допомогою певних ключових слів, розумінні того, що подобається користувачеві, пошуку цих ключових слів у базі даних та рекомендуванню різних продуктів з однаковими атрибутами.

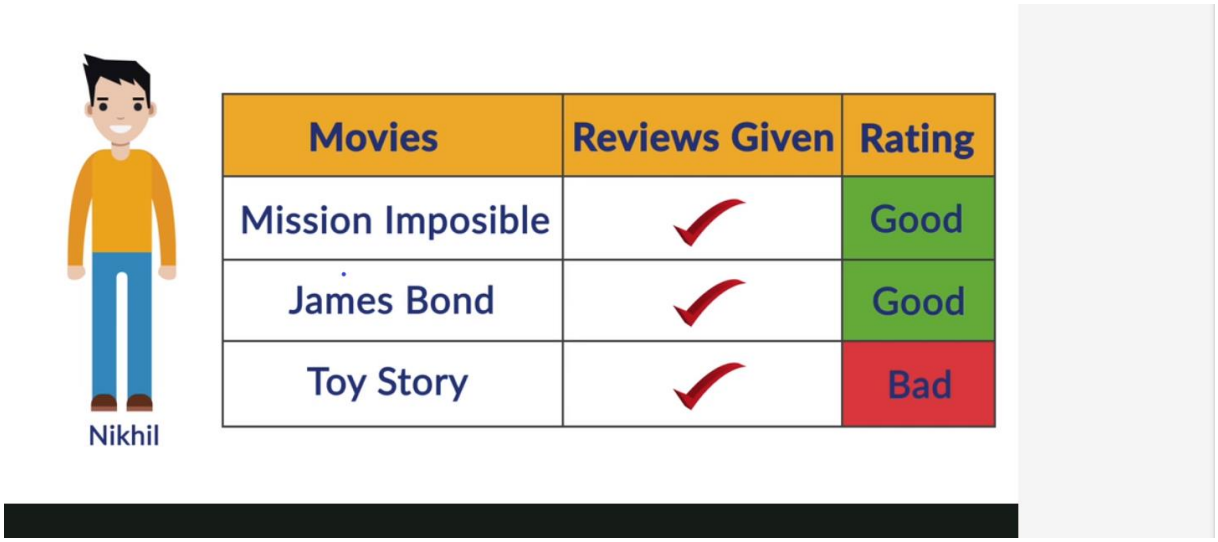

Рис. 3.6 Приклад об’єктів з їх атрибутами для вдалої роботи рекомендаційної системи онлайн кінотеатру

Фільтрування на основі контенту використовується в ряді галузей, включаючи пошук інформації (як у пошукових системах), а також у системах рекомендацій.

Спільна фільтрація може бути найсучаснішим, коли справа стосується систем машинного навчання та рекомендацій, але фільтрування на основі вмісту все ж має ряд переваг, особливо за певних обставин.

Можна виділити наступні особливості рекомендаційної системи на основі контенту:

- Результати, як правило, є дуже актуальними. Оскільки рекомендації на основі контенту покладаються на характеристики самих об'єктів, вони, ймовірно, дуже важливі для інтересів користувача. Це робить їх особливо цінними для організацій з масовими бібліотеками одного типу контенту (продумана підписка та потокові медіа-сервіси).
- Рекомендації прозорі. Ще одна перевага полягає в тому, що процес, за допомогою якого формуються будь-які рекомендації, може бути прозорим, що може збільшити довіру користувачів до їхніх рекомендацій або дозволити їх

виправити. Завдяки спільній фільтрації процес є більш чорною скринькою - алгоритм і користувачі, можливо, не дуже розуміють, чому вони бачать рекомендації.

- Користувачі можуть швидше розпочати роботу. Фільтрування на основі вмісту дозволяє уникнути проблеми «холодного старту», яка часто застосовує методи спільної фільтрації. Хоча система все ще потребує деяких початкових даних від користувачів, щоб почати давати рекомендації, якість цих ранніх рекомендацій, ймовірно, буде набагато вище, ніж у системи, яка стає надійною лише після додавання та співвіднесення мільйонів точок даних.
- Нові товари можна рекомендувати негайно. Що стосується проблеми "холодного старту", ще одна проблема спільної фільтрації полягає в тому, що нові об'єкти, додані до бібліотеки, матимуть декілька (якщо такі є) взаємодій, це означає, що їх не рекомендують дуже часто. На відміну від систем спільної фільтрації, рекомендації на основі вмісту не вимагають від інших користувачів взаємодії з об'єктом, перш ніж він почне рекомендувати його.
- Це технічно простіше втілити. Порівняно зі складною математикою, яка бере участь у створенні системи спільної фільтрації, наука даних, що стоїть на основі системи контенту, є відносно простою. Справжня робота, як ми бачили, полягає в призначенні атрибутів в першу чергу.

На самому базовому рівні фільтрація на основі контенту полягає в призначенні атрибутів елементам, щоб алгоритм знав щось про вміст кожного елемента в базі даних. Коли ви бачите колекцію Netflix із гіпер-нішею (як, скажімо, "Науково-фантастичні каперси із сильними жіночими відводами" або "Химерні інді-роми, встановлені в штаті Орегон"), ви бачите фільтрування на основі вмісту в дії.

Але звідки беруться ці атрибути? Відповідь багато в чому залежить від того, що ви намагаєтесь порекомендувати. Якщо ви працюєте з текстом (як у новинних статтях чи публікаціях блогу), можливо, ви зможете програмно витягувати ключові слова за допомогою методів обробки природних мов, хоча ці підходи мають власні підводні камені. Інші типи вмісту можуть мати різну кількість метаданих, хоча такі дані часто є неповними і охоплюють лише найосновніший вид даних, що робить їх обмеженим значенням, коли мова йде про рекомендації.

Щоб вирішити цю проблему, багато компаній звернулися до команд експертів з предметних питань, щоб вручну призначити атрибути кожному фрагменту вмісту. Як можна зрозубіти, цей процес може бути масштабною справою. Наприклад, Pandora використовує власну систему – проект «Музичний геном» - для створення своїх рекомендацій. Ці рекомендації

засновані на алгоритмі, який використовує більше 400 музичних характеристик для рекомендування пісень. Визначення цих характеристик та розробка алгоритму було головним завданням, яке вимагало десятків експертів з теорії музики та більше п'яти років розвитку - робота, яка триває і донині. Аналогічно, колекція Netflix, що містить понад 76 000 мікрогенерів, покладається на глядачів людських фільмів, присвоюючи десятки характеристик тисячам фільмів і телешоу в їх бібліотеці.

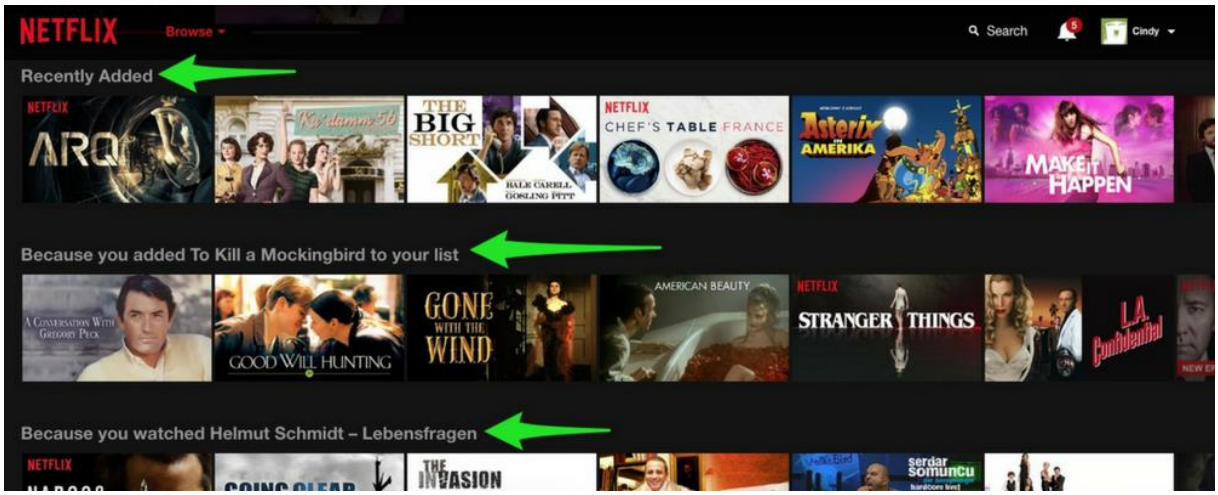

Рис. 3.7 Рекомендації у сервісі Netflix

Інша ключова частина контентної системи рекомендацій – це вподобання користувача. Ці профілі складаються з об'єктів, з якими користувач взаємодіяв, а також атрибутів цих об'єктів. Атрибути, які відображаються на ряді об'єктів, важать більше, ніж ті, які з’являються рідше. (Вдосконалені алгоритми можуть враховувати не лише предмети, які користувач переглянув, прочитав чи прослухав, але й ті, які вони переглядали минулі або навіть запропонували їм.) Підсумовуючи важливість різних елементів, відгуки користувачів мають вирішальне значення – саме тому служби, що надають рекомендації, постійно просять оцінити вміст.

Виходячи з цих атрибутів і загальної історії, система виробляє унікальну модель уподобань кожного користувача, часто використовуючи методи машинного навчання. Ці моделі складаються з атрибутів, які кожному користувачеві подобаються (або не подобаються), зважених за важливістю. Потім ці моделі порівнюються з усіма об'єктами в базі даних і присвоюються балами на основі їх подібності з профілем користувача.

Ось приклад. Скажімо, ви переглядали і сподобалися кінофільми «Врятувати Рядового Райана», «Марсіанин» та «Людина-Павук», система може визнати, що вам подобаються блокбастери (які описують усі три фільми), фільми про Стівена Спілберга (де описано два) та фільми, де люди повинні рятувати Метт Деймон (теж два). Виходячи з вашого інтересу до фільмів про порятунок Метта Деймона, а також вашого

інтересу до наукової фантастики, Інтерстеллар, ймовірно, отримає високу рекомендаційну оцінку, тоді як фільм «Вихідні у Берні», який має менше спільного з іншими назвами, мабуть, отримає нижчий бал.

Проблеми фільтрування на основі вмісту

Відсутність новизни та різноманітності. Актуальність важлива, але це ще не все. Якщо ви дивилися та полюбили оригінальні «Зоряні війни», шанси досить викокі, що вам також сподобається «Імперія Наносить Зворотній Удар», але вам, мабуть, не потрібен механізм рекомендацій, щоб сказати це. Також важливо, щоб механізм рекомендацій створив нові результати (тобто речі, яких користувач не очікував) та різноманітні (тобто речі, що представляють широкий вибір їхніх інтересів).

Масштабованість - це виклик. Як ми бачили, ключовою вимогою, що стосується фільтрування на основі вмісту, є виняткове знання, яке залежить від домену. Наймання експертів з предметних питань може бути трудомістким і дорогим процесом, що робить непрактичним для багатьох підприємств, які просто намагаються побудувати МВП. Крім того, додавання нового контенту до ручного тегування атрибутів має тривати.

Атрибути можуть бути застосовані неправильно або непослідовно. Рекомендації, що базуються на вмісті, настільки ж хороші, як і спеціалісти з тематики, які позначають теги. Якщо у вас є сотні тисяч (або мільйонів) предметів, це може бути складним завданням, щоб атрибути застосовувалися послідовно або точно по призначенню.

Шляхи реалізації рекомендаційної системи на основі контенту

Фільтрування на основі контенту – один із поширених методів побудови систем рекомендацій. Є два підходи:

- Аналіз опису лише контенту;
- Створення профілю користувача та профілю об’єкта з контенту, оціненого користувачем.

Перший підхід аналогічний спільному фільтруванню на основі об’єктів. Коротше кажучи, система порекомендує все, що схоже на предмет, який сподобався користувачеві раніше.

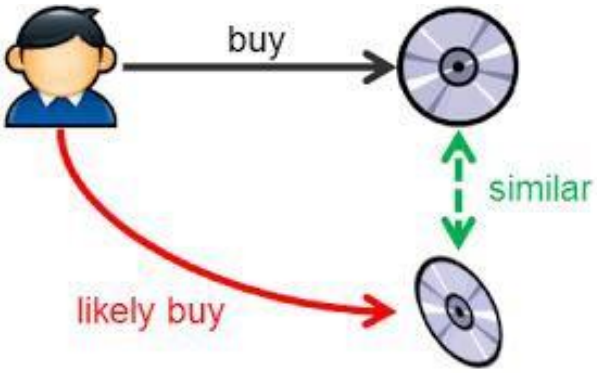

Рис. 3.8 Схема аналізу лише контенту

На етапі побудови моделі система спочатку знаходить схожість між усіма парами елементів, а потім використовує найбільш схожі елементи з уже оціненими елементами користувача для формування списку рекомендацій на етапі рекомендацій.

Зазвичай подібність буде отримана з опису предмета і буде введено поняття TF-IDF. Тоді кожен елемент буде представлений вектором TF-IDF. TF-IDF знаходиться в підрозділі обробки природних мов (NLP). Він використовується в пошуку інформації для вилучення функцій. Коротше кажучи, підраховується кількість кожного слова в документі і зважуєте важливість кожного слова, і обчислюєте бал за цей документ.

Частота вживаності слова в поточному документі порівнюється із загальною кількістю слів у документі. Чим більше входжень слова з'являється у документі, тим більш важливим він стає.

Загальна кількість документів для частоти зустрічі документів, що містять слово. Це означає рідкість слова, оскільки слово, що зустрічається в документі, менше, ніж зростає IDF. Це допомагає давати більш високі бали за рідкісні терміни в документах.

Зрештою, TF-IDF – це міра, яка використовується для оцінки того, наскільки важливим є слово для документа в рамках документа. Важливість слова збільшується пропорційно кількості разів, коли слово з'являється в документі, але компенсується частотою слова в корпусі. Оскільки метод значною мірою покладається на опис для розрізнення кожного елемента, опис повинен заглиблюватися в деталі продукту, тобто назву, резюме, мітки, жанр, так що він забезпечує набагато більше інформації про предмет.

Щоб обчислити, наскільки схожими є елементи векторів, можна використовувати різні методи, такі як:

- Подібність косину;
- Евклідова відстань;
- Піресова кореляція.

Тоді рекомендаційна система дасть рекомендації на основі найбільш подібних пунктів. Суттєвої різниці між усіма методами – немає.

Другий підхід заснований на описі або атрибутах предметів, з якими користувач взаємодіяв, щоб рекомендувати подібні елементи. Це залежить лише від попереднього вибору користувача, що робить цей метод надійним, щоб уникнути проблеми з холодним запуском. Для текстових елементів, як-от статей, новин та книг, просто використовувати категорію статей чи необроблений текст для створення профілів предметів та профілів користувачів.

Переглядаючи конкретний жанровий фільм, будуть рекомендовані фільми відповідно до цього конкретного жанру. Назва, рік випуску, режисер, ролі також корисні для виявлення подібного вмісту фільму.

У цьому підході вміст об’єкта вже оцінюється залежно від уподобань користувача (профіль користувача), тоді як жанр елемента - це неявні функції, які будуть використані для створення профілю елемента. Потім прогнозується оцінка позиції за допомогою обох профілів, і рекомендації можуть бути зроблені. Подібно до першого підходу, у цьому підході буде також використана методика TF-IDF.

Саме такий метод рекомендації контенту можна зустріти на багатьох Web-сайтах із можливістю перегляду фільмів в режимі онлайн. Реалізувати таку систему значно простіше, оскільки можна скористатись системою обробки інформації, яка буде розміщена на віддаленому сервері і на основі отриманих даних можна видати користувачеві об’єкт, який скоріше за все йому сподобається. Особливо беручи до уваги подібні об’єкти із історії його перегляду.

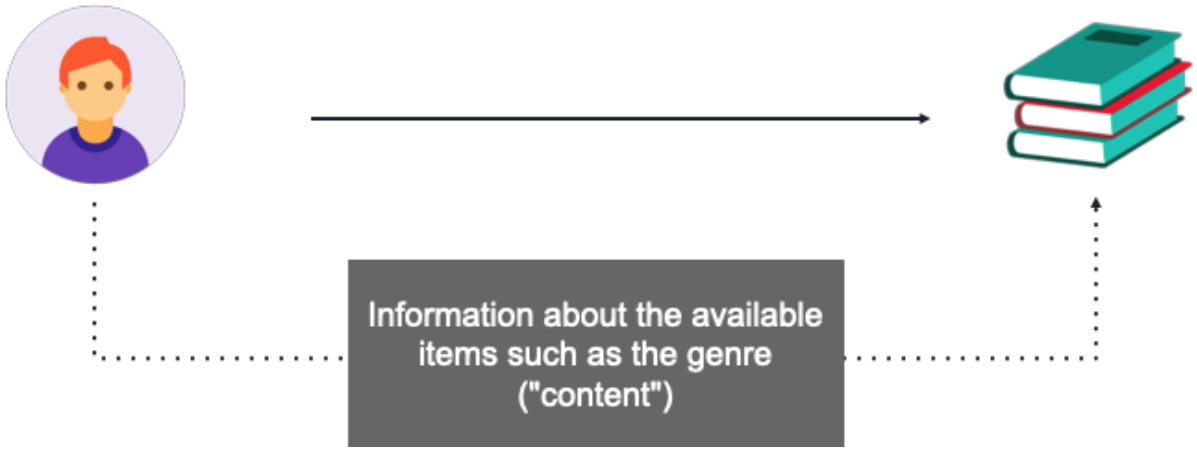

Рис. 3.9 Схема аналізу характеристик контенту

Чому часто обирають рекомендаційну систему на основі контенту?

Усі інші моделі страждають від того, що називається «проблемою холодного старту». Оскільки рекомендації обчислюються за допомогою набору даних відгуків користувачів про елементи, вони не можуть рекомендувати елементи без жодного або лише кількох відгуків, наприклад нових елементів. Так само вони не можуть нічого рекомендувати новому користувачеві, перш ніж почати завантажувати, даючи відгуки про достатню кількість елементівю. Ці проблеми усуваються за допомогою моделей на основі вмісту. Підхід ідентичний попереднім алгоритмам «Користувач-користувач» або «Елемент-Елемент», за винятком того, що подібність обчислюється за допомогою лише функцій, що базуються на вмісті.

Для підготовки моделі до вирішення елемента «холодний старт» (відповідно, користувальницький холодний старт) вам потрібен набір даних, включаючи докладні описи ваших елементів (відповідно ваших

користувачів), таких як жанр фільму, його бюджет, його тривалість, або будь-яку змінну, яка може допомогти рекомендації. Нещодавній прогрес розпізнавання шаблонів у машинному навчанні відкрив значні вдосконалення моделі, що базується на вмісті, використовуючи інформацію, витягнуту з неочищених зображень або опису необробленого тексту. В Інтернеті можна знайти безліч інструментів та заздалегідь навчених моделей глибокого навчання Комп'ютерного зору чи обробки природних мов. Очевидною перевагою використання заздалегідь підготовленої моделі є те, що вам не потрібно величезних наборів даних і дорогих серверів для навчання вашого механізму рекомендацій.

Переваги та недоліки рекомендаційної системи на основі контенту

В першу чергу потрібно відмітити переваги системи:

- Незалежність користувача: Змістовий метод повинен аналізувати лише елементи та профіль одного користувача на предмет рекомендації, що робить процес менш громіздким. Таким чином, фільтрування на основі вмісту дає більш надійні результати при меншій кількості користувачів у системі.
- Прозорість: спільне фільтрування дає рекомендації на основі інших невідомих користувачів, які мають той самий смак, як у певного користувача, але при фільтруванні на основі вмісту елементи рекомендуються на рівні функцій.
- Немає холодного початку: на відміну від спільної фільтрації, нові елементи можна запропонувати перед оцінкою значної кількості користувачів.

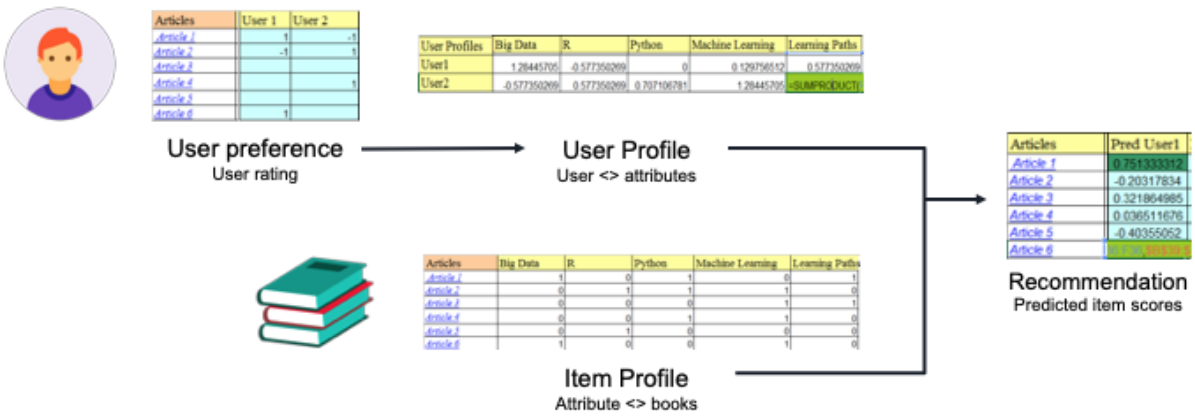

Рис. 3.10 Схема профіля користувача і профіля об’єкту

Зазвичай таблиця рейтингу (оцінка користувачів), профіль статті (жанри книг) – це єдиний матеріал, який ми маємо.

- рейтингова таблиця: співвідношення користувач-книга
- профіль предмета: відношення атрибутів до книги

Articles	Big Data	R	Python	Machine Learning	Learning Paths		Total attributes		User 1	User 2
Article 1	1	0	1	0	1		3		1	-1
Article 2	0	1	1	1	0		3		-1	1
Article 3	0	0	0	1	1		2			
Article 4	0	0	1	1	0		2			1
Article 5	0	1	0	0	0		1			
Article 6	1	0	0	1	0		2		1	
DF	2	2	3	4	2					

Рис. 3.11 Приклад стандартної таблиці атрибутів об’єкту

Є декілька переваг та недоліків того чи іншого підходу до створення рекомендаційної системи на основі контенту. Слід виділити плюси і мінуси першого та другого підходу до реалізації системи.

Плюси першого методу:

- На відміну від спільної фільтрації, якщо елементи містять достатню кількість описів, уникається «проблема нового елементу».
- Представлення контенту різноманітне, і воно відкриває варіанти використання різних підходів, таких як: техніка обробки тексту, використання смислової інформації, тощо.
- Зробити більш прозору систему легко: ми використовуємо той самий контент для пояснення рекомендацій.

Мінуси першого методу:

- Вміст RecSys схильний до надмірно спеціалізації: вони рекомендуватимуть предмети, подібні до тих, що вже вживались, із тенденцією до створення «фільтруючої бульбашки».

Плюси другого методу:

- Незалежність користувача: спільне фільтрування потребує рейтингу інших користувачів, щоб знайти схожість між користувачами, а потім дати пропозицію. Натомість контент-метод повинен аналізувати лише елементи та профіль користувача за рекомендацією.
- Прозорість: метод спільної роботи дає вам рекомендацію, оскільки деякі невідомі користувачі мають такий самий смак, як і ви, але на основі контентного методу можна сказати, що вони рекомендують вам предмети, залежно від особливостей саме ваших вподобань.
- Немає холодного початку: на відміну від спільної фільтрації, нові елементи можуть бути запропоновані, перш ніж оцінювати значну кількість користувачів.

Мінуси другого методу:

- Обмежений аналіз контенту: якщо вміст не містить достатньо інформації для точної дискримінації елементів, рекомендація не буде точно в кінці.
- Надспеціалізація: контент-заснований метод забезпечує обмежений ступінь новизни, оскільки він повинен відповідати особливостям профілю та предметів. Цілком досконала фільтрація на основі вмісту може нічого не запропонувати «здивувати».
- Новий користувач: коли недостатньо інформації для створення надійного профілю для користувача, рекомендацію не можна було б надати правильно.

3.3 Гібридна рекомендаційна система

Як фільтрування на основі вмісту, так і спільне фільтрування мають сильні та слабкі сторони. Для фільтрування на основі вмісту можна виділити три конкретні проблеми:

- Опис контенту. У деяких областях створити корисний опис вмісту може бути дуже складно. Наприклад, у доменах, де елементи складаються з музики чи відео, наприклад, подання вмісту не завжди можливо за допомогою сучасної технології.
- Надспеціалізація. Система фільтрації на основі вмісту не буде вибирати елементи, якщо попередня поведінка користувача не підтверджує це. Необхідно додати додаткові методи, щоб надати системі можливість висловлювати пропозиції поза межами того, до чого користувач уже виявив інтерес.
- Суб'єктивна проблема домену. Методи фільтрації на основі вмісту мають труднощі розрізняти суб'єктивну інформацію, таку як точки зору та гумор.

Спільна система фільтрації не має цих недоліків. Оскільки немає необхідності в описі елементів, що рекомендуються, система може обробляти будь-яку інформацію. Крім того, система здатна рекомендувати користувачеві елементи, які можуть мати зовсім інший вміст, ніж те, що користувач раніше зацікавив. Нарешті, оскільки рекомендації ґрунтуються на думці інших, вони добре підходять для суб'єктивних областей, як мистецтво.

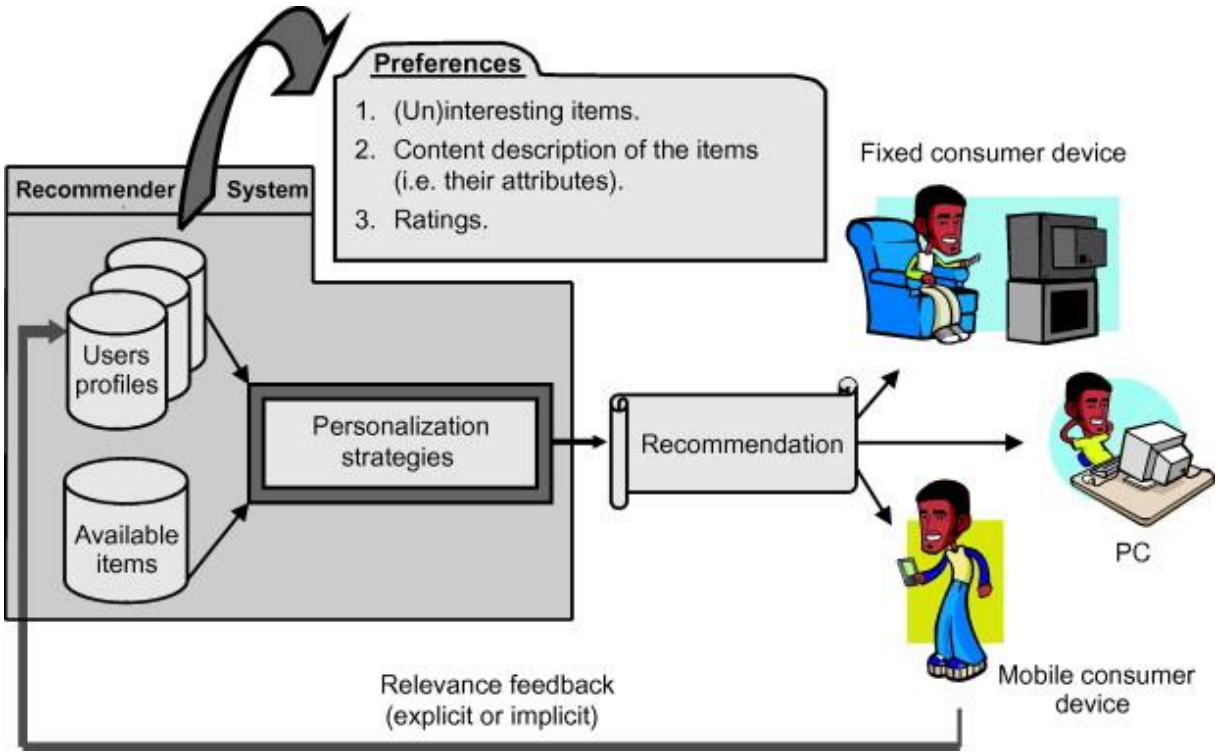

Рис. 3.11 Приклад результатів роботи гібридної рекомендаційної системи

Однак спільна фільтрація справді створює певні проблеми:

- Рання проблема з рейтингами. Системи спільної фільтрації не можуть надати рекомендації щодо нових елементів, оскільки немає рейтингів користувачів, на яких можна базувати прогноз. Навіть якщо користувачі почнуть оцінювати товар, пройде якийсь час, перш ніж товар отримає достатню кількість оцінок, щоб дати точні рекомендації. Аналогічно, рекомендації також будуть неточними для нових користувачів, які оцінили мало елементів.
- Проблема зрідженості. У багатьох інформаційних сферах наявна кількість предметів перевищує кількість, яку людина здатна (і бажає) дослідити на сьогоднішній день. Це ускладнює пошук предметів, оцінених достатньою кількістю людей, на яких можна базувати прогнози.
- Сірі вівці. Групи користувачів потрібні з перекриваючими характеристиками. Навіть якщо такі групи існують, люди, які не погоджуються або не погоджуються ні з якою групою людей, отримуватимуть неточні рекомендації.

Система, що поєднує фільтрацію на основі контенту та спільну фільтрацію, може скористатись як представленням вмісту, так і подібністю серед користувачів. Хоча існує декілька способів поєднання двох методів, можна розрізняти два базові підходи. Гібридний підхід поєднує в собі два типи інформації, хоча також можна використовувати рекомендації двох методів фільтрації незалежно один від одного.

Порівняння спільної рекомендаційної системи та рекомендаційної системи на основі контенту

Спільна фільтрація шукає співвідношення між рейтингами користувачів для прогнозування. Таке співвідношення є найбільш значимим, коли користувачі мають багато спільних вподобань. Як було сказано раніше, у великих областях з багатьма предметами це не завжди так. Крім того, відсутність доступу до вмісту елементів перешкоджає збігу подібних користувачів, якщо вони не оцінили один і той самий об’єкт. Наприклад, якщо одному користувачеві сподобався фільм «Роккі», а іншому сподобався фільм «Рокі II», вони не обов'язково будуть узгоджуватися разом. Гібридний підхід, який називається співпрацею через вміст, займається цими питаннями, включаючи як інформацію, що використовується при фільтрації на основі вмісту, так і за допомогою спільної фільтрації.

У співпраці за допомогою вмісту для створення профілю користувача використовуються як рейтингові елементи, так і вміст елементів. Вибір термінів, які описують зміст предметів, здійснюється за допомогою контентних методик. Вага характеристик вказує на те, наскільки вони важливі для користувача [Рис. 3.11].

Рис. 3.12 Схема роботи гібридної рекомендаційної системи

Замість визначення кореляції з рейтингами користувачів, однак використовуються терміни ваги. Оскільки у цього методу є більша кількість елементів, з яких визначити схожість, ніж спільна фільтрація, проблема користувачів, які не мають достатньо загальних позицій, вже не є проблемою. Крім того, на відміну від фільтрування на основі вмісту, прогнози ґрунтуються на враженнях інших користувачів, які можуть призвести до рекомендацій поза звичайним середовищем користувача. Однак для надання рекомендацій щодо товарів все-таки необхідно, щоб було достатньо користувачів, які оцінили товар. Так само, як і спільну фільтрацію нових елементів, не можна рекомендувати, якщо немає користувачів, які оцінили новий елемент.

Інший підхід до поєднання спільної та контентної фільтрації полягає у складанні прогнозів на основі середньозваженої середньої рекомендації на основі вмісту та рекомендації щодо спільної роботи. Ранг кожного рекомендованого предмета може бути мірою для ваги. Таким чином найвища рекомендація отримує найбільший пріоритет.

ВИСНОВОК ДО РОЗДІЛУ 3

У цьому розділі було розглянуто основні типи рекомендаційних систем та методи їх реалізації. Загалом можна виділити три основних типи рекомендаційних систем, серед яких спільна рекомендаційна системи, система рекомендацій на основі контенту та гібридна (змішана) рекомендаційна система. У кожної з них є свої переваги та недоліки. В залежності від мети яка поставлена перед проектом, потрібно обирати відповідний методі реалізації рекомендаційної системи. Найбільш явною і простою є система рекомендацій на основі контенту. Існує можливість використання віддаленої системи підрахунку збігів у характеристик того чи іншого об’єкту. Тим самим спрощується можливість використання подібної системи на будь-яких пристроях.

Розділ 4. Створення Web-додатку "Рекомендаційна система"

Розібравшись із технологіями створеня Web-додатків та основними типами рекомендаційних систем, можно приступити до створення проекту. Дотримуючись послідовності, можна створити Web-додаток, який би продемонстрував можливості рекомендації контенту на основі бази фільмів. Такий додаток допоміг би людині підібрати гарний кінофільм, зважаючи на його вподобання.

Оптимальною рекомендаційною системою для розробки проекту було обрано систему рекомендацій на основі контенту. Така система дозволяє продемонструвати можливості рекомендацій в інтернеті і при цьому не потребує від користувача занадто великої кількості маніпуляцій. Також вона дозволяє позбутись проблеми нульового результату, коли інші користувачі не пов’язані з конкретним користувачем, або надали занадто мало фактів щодо перегляду того чи іншого фільму.

4.1 Необхідне програмне забезпечення

Для того щоб розпочати роботу над створенням Web-додатку, потрібно обрати вдале середовище розробки. Найпростішим інструментом можна назвати Visual Studio Code від компанії Microsoft. Це програмне забезпечення є безкоштовним і його може завантажити будь який користувач, незалежно від платформи на якій здійснюється розробка продукту. Це може бути як операційна система Windows від Microsoft, так і будь яка Unix система.

За допомогою цієї програми можна легко редагувати код, або ж створювати документи HTML, CSS чи JavaScript. Основна версія програми є безкоштовною, та при бажанні можна придбати версію із розширеним функціоналом для створення комерційних продуктів чи використання віддаленого сховища.

Серед плюсів програмного забезпечення Visual Studio Code можна відзначити той факт, що воно легко встановлюється і не потребує значних ресурсів технічного засобу на якому буде використовуватись. Є версія не тільки для персональних комп’ютерів, але і для мобільних телефонів чи планшетів. Це дозволяє отримати можливість продовжувати розробку, незалежно від пристрою, часу, та місцезнаходження користувача.

Для даного проекту вистачить безкоштовної версії програмного забезпечення, тому що розробка буде виконуватись локально і розширений функціонал попросту не знадобиться.

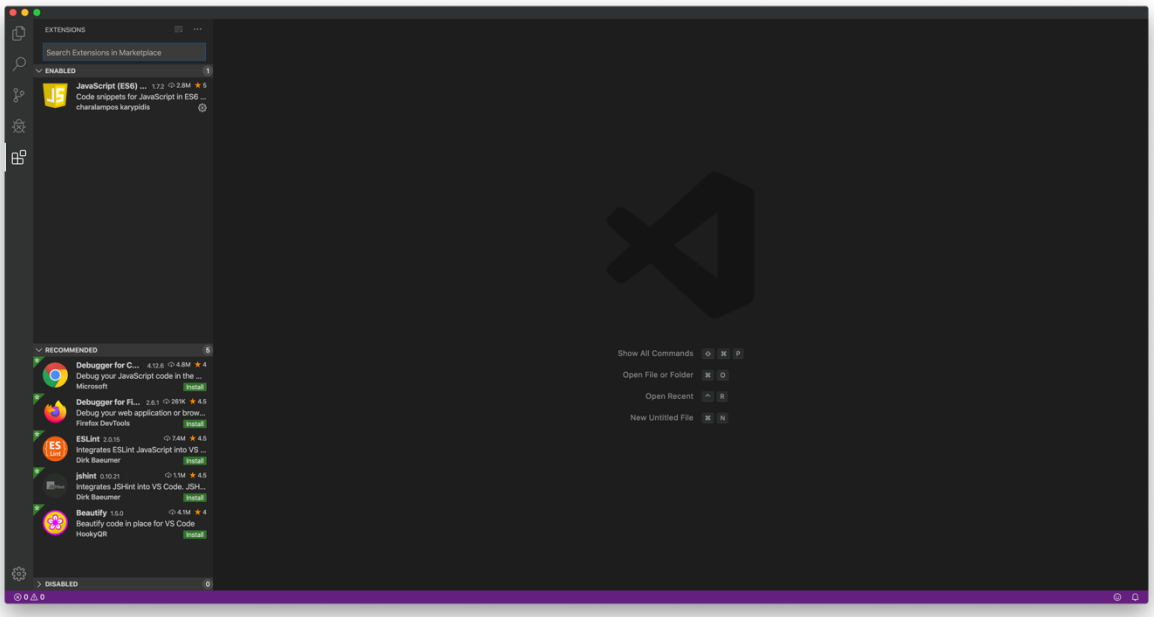

Рис. 4.1 Початковий екран програми Visual Studio Code

З лівого боку початкового екрану можна обрати необхідні розширення для роботи з різними мовами програмування, у тому числі JavaScript. Також можна обрати корінну папку в якій і будуть розміщені файли з якими буде відбуватися подальша робота над створенням Web-додатку.

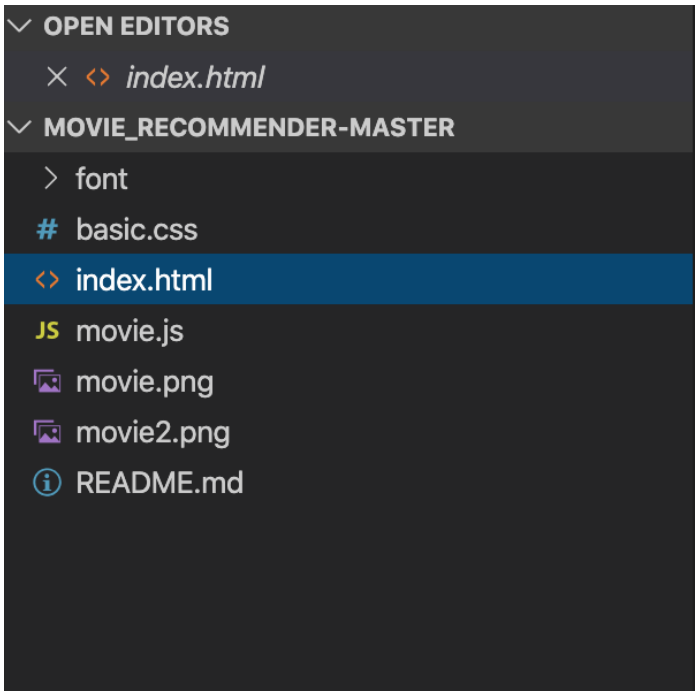

Рис 4.2 Відображення корневої папки з файлами Web-додатку у Visual Studio Code

4.2 Створення Web-додатку «Рекомендаційна система»

Додаток створюється для користувача, який бажає переглянути кінофільм, та не може визначитись із тим, який би йому міг сподобатись. Щоб полегшити вибір, рекомендаційна система Web-додатку буде надавати користувачу список із фільмами. В цілому на одній сторінці буде відображатись до трьох кінофільмів різних жанрів. У випадку якщо ні один із них не підійде користувачу, він може вказати на це за допомогою функції представленої в шапці Web-додатку і сторінка автоматично видає наступну «порцію» кінофільмів.

Для початку створюється документ index.html, в якому буде розміщена основна інформація по розташуванню контенту на сторінці. Елементи будуть розподілені таким чином щоб користувачу не потрібно було докладати лишніх зусиль для вибору гарного фільму для перегляду.

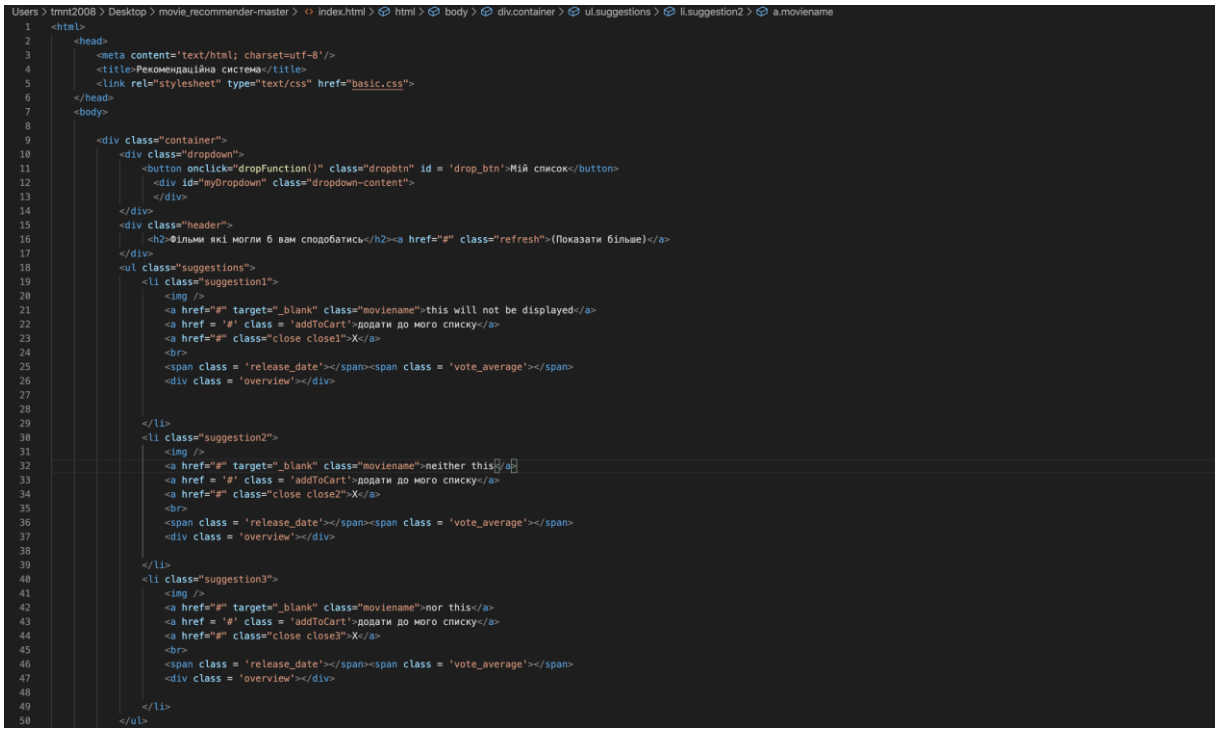

Рис. 4.3 Документ «index.html». Основа Web-додатку

Але тільки HTML не допоможе у вдалому представлені Web-додатку перед користувачем. Потрібно розробити в міру простий інтерфейс, який дозволив би умістити на одній сторінці максимум інформації. Саме тому було використано стандартний CSS. Створивши файл basic.css, в Web-додаток було додано стилі. Завдяки цьому, елементи проекту не будуть розкидані по сторінці і будуть мати чітке відображення на призначених для цього місцях.

Було обрано варіант при якому на сторінці в цілому буде відображатись до трьох кінофільмів. Таким чином можна прикувати увагу користувача до запропонованих йому картин.

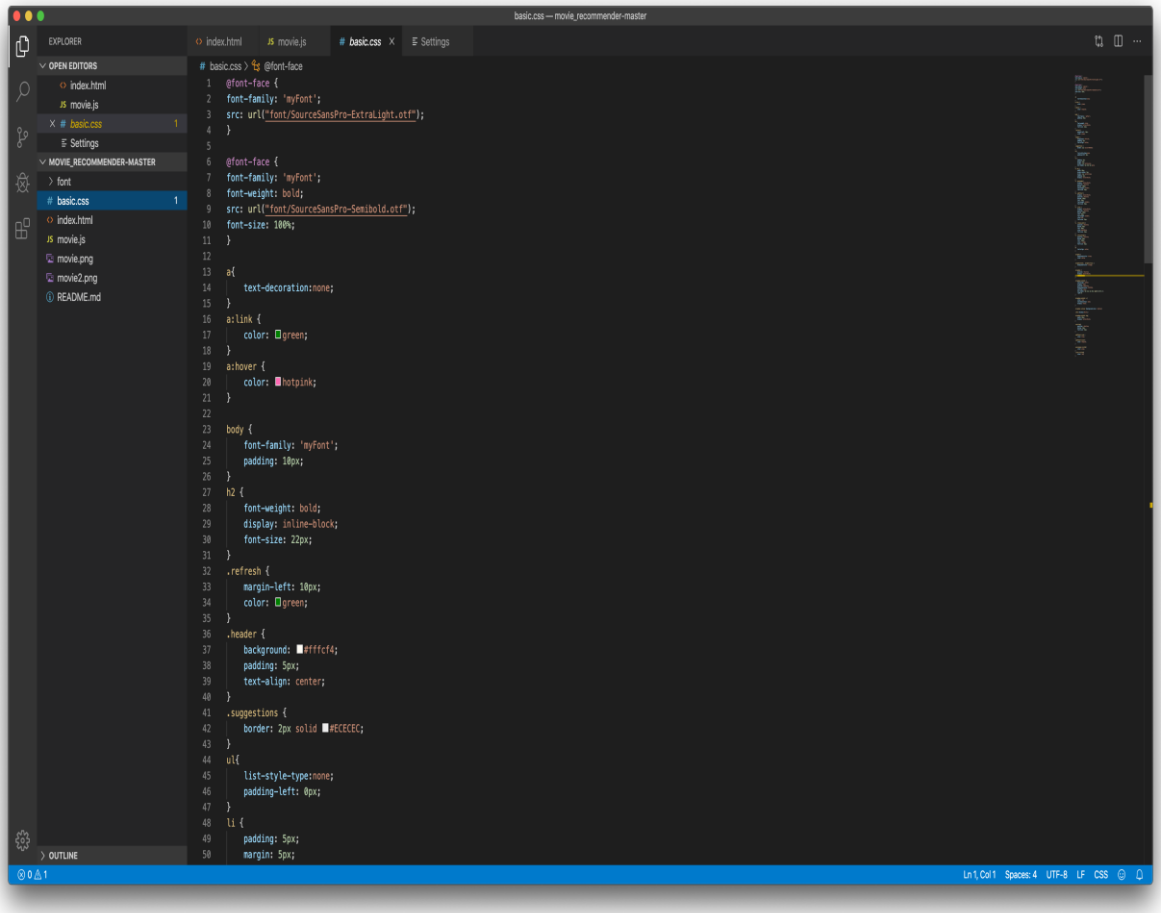

Рис. 4.4 Документ «basic.css». Стилі елементів Web-додатку

Для створення інтерактивності сторінки і перетворити звичайний Web-сайт на Web-додаток, потрібно використати JavaScript. Тим сами можна буде створити сценарій згідно якого користувач буде взаємодіяти з додатком і отримувати рекомендації щодо перегляду кінофільмів.

Web-додаток має генерувати список із кінофільмами для користувача, використовуючи для цього концепт асинхронної передачі інформації. Краще всього для цього підійде бібліотека ReactiveX. Це спеціальна бібліотека, яка дозволяє створити асинхронну і засновану на діях програму, використовуючи для цього послідовності.

Рис. 4.5 Бібліотека Reactive X для JavaScript

Потокова передача даних представляє собою послідовність подій. Можна передавати певний тип значень, помилок та сигналів. У випадку цього Web-додатку можна фіксувани певні події асинхронним способом. Сигналів про помилки в цьому виникнути не має, бо вся увага буде зосереджена на обробці значень.

Моніторинг потоку називається підписом. Функція для обробки потоку називається спостерігачем. Таким чином, потоки в ReactiveX також називають спостережними.

План задач які потрібно визначити завдяки JavaScript та ReactiveX наступний:

- Коли користувач відкриває сторінку, додаток надає три пропозиції щодо фільму, вибравши їх з кінцевих точок API.
- Коли користувач натискає кнопку "оновити", він відображає 3 нові пропозиції щодо фільму.
- Коли користувач натискає кнопку "x" на одній із пропозицій фільму, вона заміняється новою пропозицією.
- Коли користувач натисне заголовок фільму, з’явиться нова вкладка домашньої сторінки фільму.
- Коли користувач натисне "додати в список" на одне з пропозицій фільму, фільм буде доданий до кошика (колекції) користувача, який знаходиться в правому верхньому куті сторінки.

- Список користувача зберігається в локальній пам’яті (офлайн), щоб користувач все ще мав доступ до нього під час перегляду сторінки.
- Коли користувач обрав певний кінофільм, система рекомендацій підказує йому наступний фільм, який міг би йому сподобатись судячи з оцінки і параметрів стрічки.

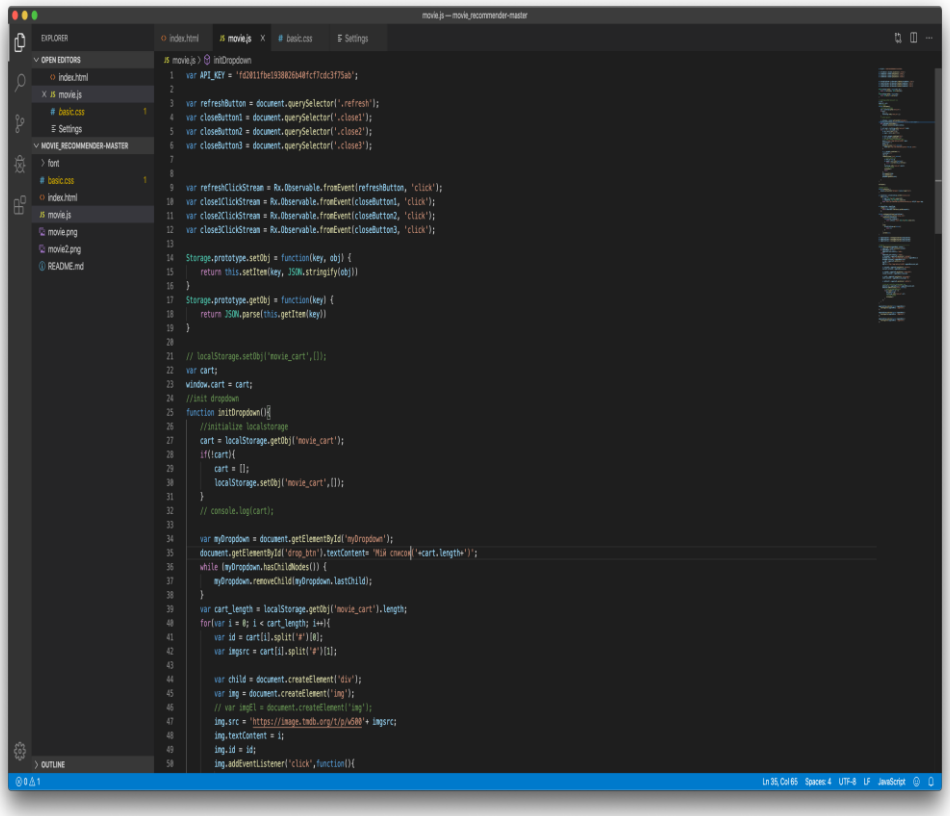

Рис 4.6 JavaScript документ «movie.js», в якому прописаний сценарій дій Web-додатку

Можна виділити декілька основних етапів розробки правильної видачі кінофільмів у Web-додаткі.

Для початку створюється спостереження від події клацання по трьом кнопкам закриття та на одну кнопку оновлення.

```
var closeButton1 = document.querySelector('.close1');
var closeButton2 = document.querySelector('.close2');
var closeButton3 = document.querySelector('.close3');


var refreshClickStream = Rx.Observable.fromEvent(refreshButton, 'click');
var close1ClickStream = Rx.Observable.fromEvent(closeButton1, 'click');
var close2ClickStream = Rx.Observable.fromEvent(closeButton2, 'click');
var close3ClickStream = Rx.Observable.fromEvent(closeButton3, 'click');
```

Рис. 4.7 Створення спостерігача для подій клацання по кнопкам додатку

Другим кроком є визначення потоку запиту та відповідь. Коли сторінка завантажується вперше або натискається кнопка оновлення, вона запустить потік запитів, який містить запит API GET. Для потоку відповідей він відображає кожен потік запиту на запрошений об'єкт.

```
var requestStream = refreshClickStream.startWith('startup click')
    .map(function(){
        var page = Math.floor(Math.random()*100)+1
        var randomOffset = Math.floor(Math.random()*20);
        return 'https://api.themoviedb.org/3/discover/movie?
                api_key='+API_KEY+'&page='+page;
    })

var responseStream = requestStream
    .flatMap(function (requestUrl) {
        return Rx.Observable.fromPromise($.getJSON(requestUrl));
    });
```

Рис. 4.8 створення запиту до API

Створюється потік пропозицій. Ідея полягає в тому, що потрібно відобразити три потоки пропозицій, що є випадковим вибором зі списку повернутих фільмів довжиною 20. Можна поєднати потік closeclick (користувач натиснув «x») з останнім потоком відповідей. Отже, коли користувач натискає "x", це пропозиція замінюється іншим елементом із останнього поверненого списку фільмів, не викликаючи API знову. merge () використовується для об'єднання декількох потоків в один потік.

```
function createSuggestionStream(closeClickStream) {
    return closeClickStream.startWith('startup click')
        .combineLatest(responseStream,
            function(click, listMovies) {
                return listMovies['results'][Math.floor(Math.random()*20)];
            }
        )
        .merge(
            refreshClickStream.map(function(){
                return null;
            })
        )
        .startWith(null);
}
```

Рис. 4.9 створення потоку пропозицій

Останнім кроком є відображення потоку даних. Це в основному відображення ключових значень файлу JSON.

```
// Rendering -----------------------------------------------
function renderSuggestion(suggestedMovie, selector) {
    var suggestionEl = document.querySelector(selector);
    if (suggestedMovie === null) {
        suggestionEl.style.visibility = 'hidden';
    } else {
        suggestionEl.style.visibility = 'visible';
        var movienameEl = suggestionEl.querySelector('.moviename');
        movienameEl.href = 'https://www.themoviedb.org/movie/'+suggestedMovie.id;
        movienameEl.textContent = suggestedMovie.title;
        var imgEl = suggestionEl.querySelector('img');
        imgEl.src = "";
        imgEl.src = 'https://image.tmdb.org/t/p/w500'+ suggestedMovie.poster_path;

        var overviewEl = suggestionEl.querySelector('.overview');
        overviewEl.textContent = suggestedMovie.overview;

        var releaseEL = suggestionEl.querySelector('.release_date');
        releaseEL.textContent = suggestedMovie.release_date;

        var voteEL = suggestionEl.querySelector('.vote_average');
        voteEL.textContent = suggestedMovie.vote_average+'/10';

        var addToCartEl = suggestionEl.querySelector('.addToCart');

        // dirty way to solve the 'overlap-element' bug
        addToCartEl.id = suggestedMovie.id+'#'+suggestedMovie.poster_path;
        addToCartEl.addEventListener("click", function(){
            // console.log(suggestedMovie.id);
            if(cart.indexOf(this.id) < 0){
                cart.push(this.id);
                localStorage.setObj('movie_cart',cart);
                //re-render dropdown
                initDropdown();
            }
        });
    }
}
```

Рис. 4.10 відображення ключових значень JSON файлу

В якості бази даних з якої береться інформація по фільмам що відображаються в Web-додатку, було обрано відомий агрегатор кінофільмів «MovieLens». Він дозволяє отримати інформацію по рейтингам більше ніж 20 мільйонів кінострічок. А також в ньому наявні теги по тисячам фільмів, що також дозволяє значно спростити їх порівняння. База даних регулярно оновлюється починаючи з 2015 року, що дозволяє досить точно відобразити інформацію для користувача додатку.

Використовуючи набір даних MovieLens, розроблена система рекомендованих елементів (фільм до фільму), яка рекомендує фільми, подібні до тих, які обрав сам користувач. Для створення гібридної моделі ми зібрали результати автоенкодера, який вивчає вкладені на основі вмісту фільми з даних тегів, і глибокої нейромережі, що вкладає фільми на основі спільної роботи, з даних рейтингів.

До даних MovieLens входить набір близько 500 000 тегів фільмів, створених користувачами. Відповідно до документації MovieLens: «Кожен тег – це зазвичай одне слово або коротка фраза. Значення та призначення певного тегу визначає кожен користувач»

Ці дані групуються за фільмами, а теги об'єднуються для отримання розділу документів. У цьому розділі документи – це поєднання всіх тегів для певного фільму.

```
adventure characters epic fantasy world fighting photography Action
adventure atmospheric based on a book based on book beautifully
filmed ensemble cast fantasy fantasy world high fantasy imdb top 250
magic music nature nothing at all Oscar (Best Cinematography) Oscar
(Best Effects - Visual Effects) scenic stylized Tolkien wizards
adventure atmospheric ensemble cast fantasy fantasy world magic
stylized wizards Watched adapted from:book author:J. R. R. Tolkein
based on book epic fantasy middle earth faithful to book fantasy
good versus evil high fantasy joseph campbell's study of mythology
influenced magic atmospheric boring high fantasy Action adventure
atmospheric based on a book beautifully filmed fantasy high fantasy
magic music mythology romance stylized time travel Viggo Mortensen
wizards Peter Jackson Peter Jackson music must see Tolkien high
fantasy Myth Tolkien wizards Ian McKellen bast background universe
```

Рис. 4.11 Приклад того, як відображаються теги фільмів у MovieLens

Як можна бачити, слова / фрази на зразок "пригода", "фентезі" та "засновано на книзі" з'являються досить часто. Ці дані також містять імена акторів та автора фільму.

Далі документ перетворюється на теги в представлення терміна TF-IDF (Частота-зворотна частота документа). TF-IDF перетворює неструктуровані текстові дані в числові функції, які можна легше обробити алгоритмами машинного навчання. Грубо кажучи, частота кожного терміна в документі масштабується за кількістю документів, що містять цей термін. В результаті слова, які розрізняють фільми (наприклад, "чужорідне", "фентезі"), будуть більш пріоритетними, ніж слова, які використовуються для опису всіх фільмів (наприклад, "фільм", "акторський склад").

```
1  from sklearn.feature_extraction.text import TfidfVectorizer
2  tfidf = TfidfVectorizer(ngram_range=(1, 1), min_df=0.0001, stop_words='english')
3  tfidf_matrix = tfidf.fit_transform(tags['document'])
```

Рис. 4.11 Приклад того, як система відбирає слова в тегах

У просторі TF-IDF кожен вимір представляє, наскільки важливим є певне слово для фільму. Таке представлення є ідеальним, оскільки кодування єдиного поняття (наприклад, чужорідного / позаземного) є фрагментарним та розпорошеним по декількох вимірах. Ми хотіли б стиснути дані TF-IDF у простір нижнього розміру, де поняття консолідуються у загальні розміри. У такому виді буде значно легше обробити інформацію.

Для виконання стиснення використовується автоенкодер. Автоенкодери – це нейронні мережі, де виходи ідентичні входам. У цій архітектурі вхідний об'єм TF-IDF поступово стискається в 100 розмірний центральний прихований шар. Ця перша половина мережі є "кодером".

Друга половина мережі, «декодер», намагається відновити вихідний вхід. Встановлюючи функцію втрати середньої квадратичної помилки і виконуючи зворотне розповсюдження, мережа (особливо кодер) вивчає функцію, яка переносить дані в простір нижнього розміру таким чином, щоб збереглася більша частина інформації.

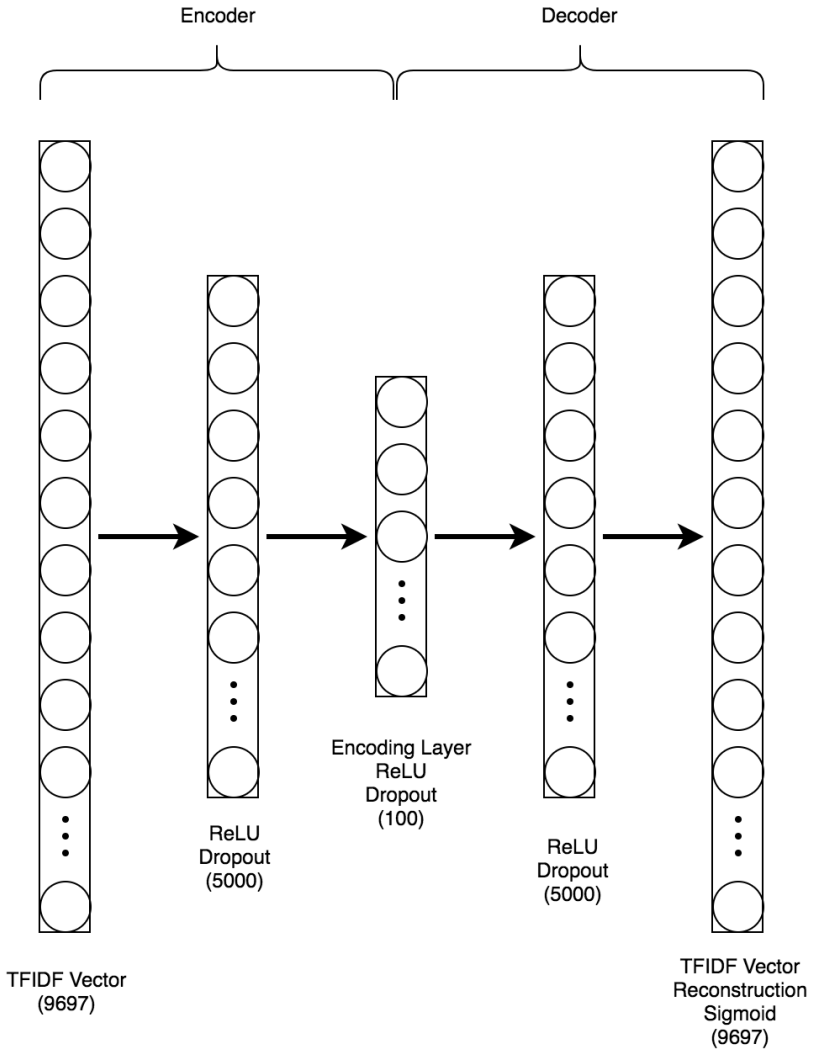

Рис. 4.12 Архітектура автоенкодеру

Таким чином з 20 фільмів у рейтинговому списку, буде вижиматись три з найкращими оцінками і які б відповідали вподобанням користувача. В залежності від тегів, картина може опинитись у відображенні. Тому же не завжди блокбастер може знаходитись на вершині рейтингу, оскільки його оцінка серед критиків та користувачів може різко відрізнятись і не буде такою високою. При цьому сам кінофільм може потенціально сподобатись користувачеві. Саме тому необхідно поступово видавати кінофільми на розсуд користувачеві. Він почне відзначати яка із картин є найбільш цікавою саме для нього і тим самим зможе поступово відібрати

цілу колекцію фільмів. Список зберігається офлайні, тому ніяких проблем із доступом до колекції бути не може.

	title	genres
1	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Sci-Fi
2	Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	Action|Adventure
3	Lord of the Rings: The Return of the King, The (2003)	Action|Adventure|Drama|Fantasy
4	Star Wars: Episode V - The Empire Strikes Back (1980)	Action|Adventure|Sci-Fi
5	Schindler's List (1993)	Drama|War
6	Lord of the Rings: The Two Towers, The (2002)	Adventure|Fantasy
7	Shawshank Redemption, The (1994)	Crime|Drama
8	Saving Private Ryan (1998)	Action|Drama|War
9	Good Will Hunting (1997)	Drama|Romance
10	Matrix, The (1999)	Action|Sci-Fi|Thriller
11	Inception (2010)	Action|Crime|Drama|Mystery|Sci-
12	Dark Knight, The (2008)	Action|Crime|Drama|IMAX
13	Princess Bride, The (1987)	Action|Adventure|Comedy|Fantas
14	Silence of the Lambs, The (1991)	Crime|Horror|Thriller
15	Batman Begins (2005)	Action|Crime|IMAX
16	Slumdog Millionaire (2008)	Crime|Drama|Romance
17	Toy Story (1995)	Adventure|Animation|Children|Co
18	Star Wars: Episode VI - Return of the Jedi (1983)	Action|Adventure|Sci-Fi
19	Life Is Beautiful (La Vita è bella) (1997)	Comedy|Drama|Romance|War

Рис. 4.13 Приклад того як виглядає рейтинг із двадцяти фільмів для відбору користувачем

Таким чином опрацювання логіки Web-додатку закінчено. Функціонал цього проекту виглядає наступним чином.

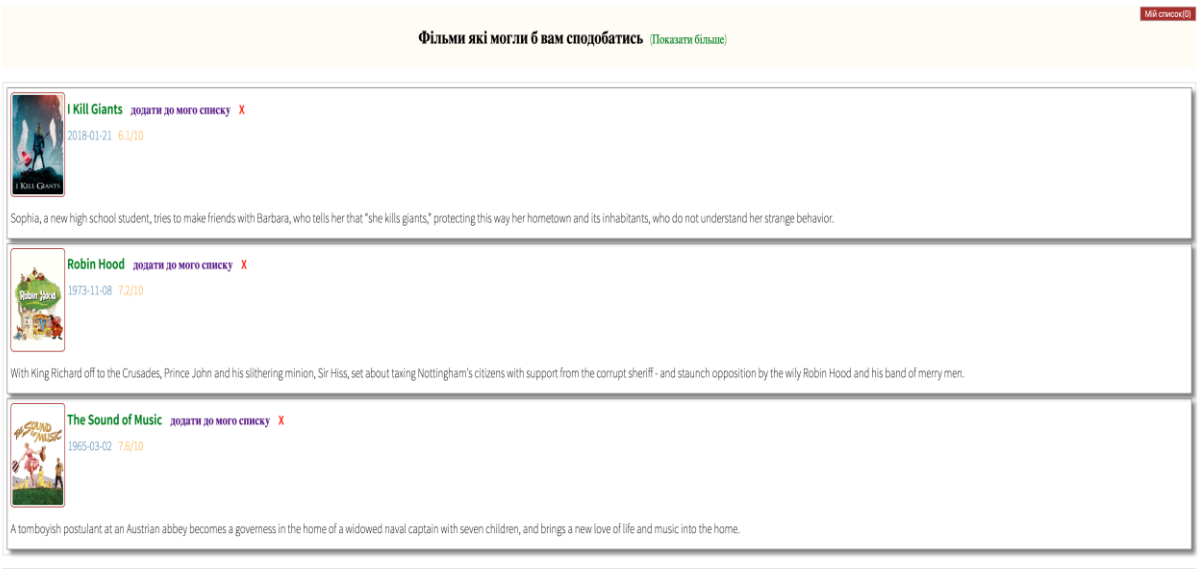

Рис. 4.13 Головна сторінка Web-додатку «Рекомендаційна система»

Таким чином на головній сторінці відображається три кінофільма відібрані із двадцяти стрічок у базі. Для початку вони добираються випадковим чином і тим самим дозволяють користувачеві підібрати початкові стрічки, які йому подобались. Оскільки добираються стрічки із найбільшим рейтингом, то на вибір початкових даних не піде багато часу.

Якщо серед запропонованої трійки не буде кінофільма, який би сподобався користувачеві, то можна натиснути на кнопку «Показати більше» і це згенерує нову трійку рекомендованих кінофільмів.

Рис. 4.14 Функціональне посилання для генерування нової трійки фільмів

Після натискання на функціональне посилання «Показати більше», трійка кінофільмів буде повністю змінена:

Рис. 4.15 Нова трійка фільмів після обнуління

Користувач здатний взаємодіяти з Web-додатком у декілька способів. Для того щоб система зрозуміла вподобання користувача, йому необхідно спочатку відібрати ті кінофільми які йому подобаються. Завдяки використанню API MovieLense для отримання інформації щодо кінофільму, користувач здатен поглянути на дату виходу картини, її світовий рейтинг на короткий опис. Цієї інформації цілком достатньо, щоб система мала змогу рекомендувати кінофільм користувачу у разі якщо він додав один із них до свого списку перегляду.

Додавання кінофільму до списку перегляду виконується дуже просто. Для цього потрібно використати функціональну кнопку «Додати до мого списку», яка знаходиться поруч із найменуванням кінофільму.

Рис. 4.16 Додавання кінофільму до персонального списку користувача

Якщо кінофільм користувачеві не сподобався, то він може натиснути на «X» поруч із найменуванням фільму і його буде автоматично замінено додатком на іншу стрічку.

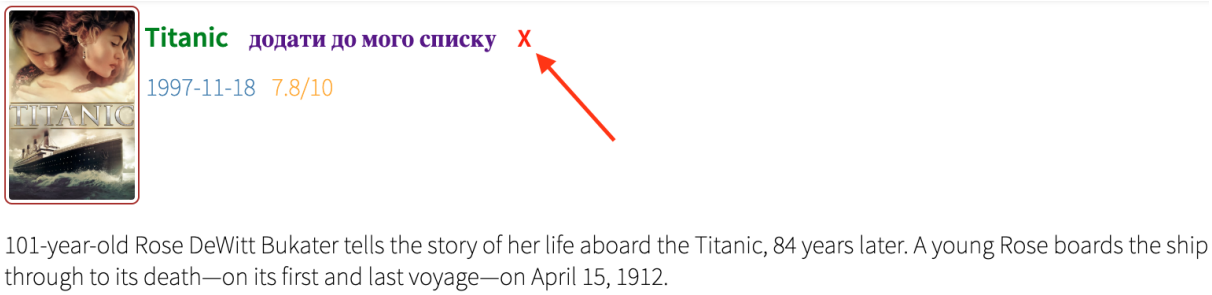

Рис. 4.17 Видалення кінофільму із рекомендацій

Користувачеві не потрібно перезавантажувати додаток, щоб отримати нові рекомендації по фільмам. Система розуміє що фільми були додані до списку користувача і підбирає кінофільми, які б відповідали рейтингу, тегам, найменуванню.

Тим самим користувач зможе економити свій час підбираючи кінофільми для перегляду в майбутньому. Важливо відмітити той факт що база фільмів постійно поповнюється. Оскільки кінокартини виходять на щоденній основі, дуже важливо щоб вони регулярно з'являлись на просторах Web-додатку.

Завдяки використанню стороннього API MovieLense, не потрібно поповнювати базу фільмів самостійно. Вона оновлюється завдяки відгукам користувачів. Кожному фільму присвоюється оцінка і він в свою чергу потрапляє на простори Web-додатку, так як проект підтягує всі актуальні кінофільми за виставленим їм рейтингом користувачів. Достатньо щоб користувач обрав цікаві йому кінофільми, щоб система продовжувала рекомендації фільмів.

Як тільки кінофільми було обрано користувачем, вони попадають у персональний список користувача.

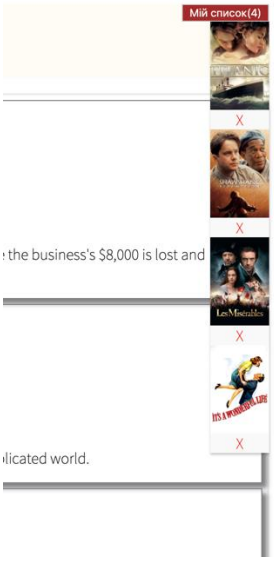

Рис. 4.18 Персональний список кінофільмів користувача у Web-додатку

Зрозуміло що користувач хотів би дізнатись більше інформації щодо кінофільма який йому було запропоновано системою рекомендацій. Саме тому в додатку реалізовано можливість відкрити окрему сторінку з детальним описом конкретного кінофільма. Натискаючи на найменування картини, користувач автоматично переходить на окрему сторінку з повним описом фільму, іменами акторів які його грають, та авторами цієї картини.

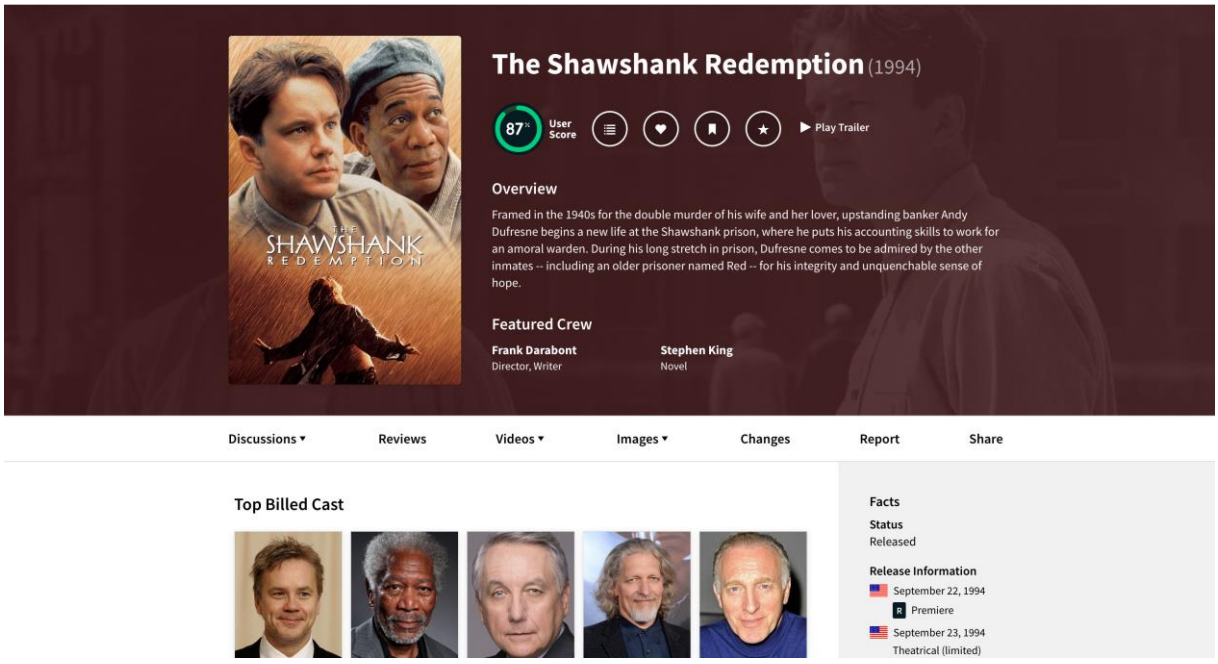

Рис. 4.19 Окрема сторінка кінофільма після натискання на його найменування в додатку

Такий функціонал є можливим завдяки використанню бази даних фільмів «The Movie DB». Вона дозволяє генерувати сторінку для кожного рекомендованого кінофільма в додатку. Таким чином користувачеві буде значно простіше обирати прийнятну для нього стрічку.

Оскільки більшість сучасних користувачів інтернету використовує мобільний телефон чи планшет в якості пристрою для виходу в інтернет, то Web-додаток також було адаптовано для використання на портативних пристроях.

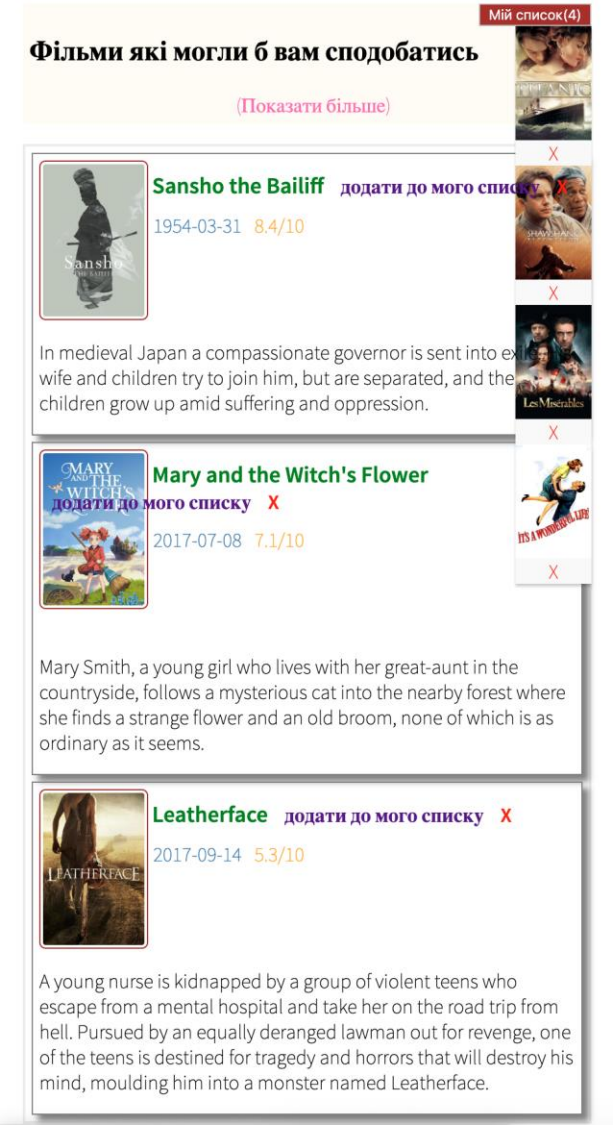

Рис. 4.20 приклад відображення Web-додатку на екрані мобільного пристрою.

Для адаптації Web-додатку під мобільні пристрої використовувались медіа-запити. Завдяки цьому можна спокійно використовувати додаток, незалежно від платформи на якій він буде відкритий. Контент буде автоматично підлаштований під розміри екрану приладу користувача.

ВИСНОВКИ ДО РОЗДІЛУ 4

Цей розділ демонструє як можна реалізувати просту систему рекомендацій у вигляді Web-додатку для вибору кінофільму. Використовуючи якісне програмне забезпечення та правильні технології, побудова додатку не стане проблемою. Було використано велику кількість технологій. В першу чергу це мова розмітки HTML, таблиці стилей CSS та об’єтно-орієнтована мова програмування JavaScript.

Web-додаток орієнтований в першу чергу на демонстрацію можливостей рекомендаційної системи, але при цьому буде зрозумілим у використанні звичайними людьми. Саме тому було використано максимально простий дизайн та інтерфейс. Його однаково просто використовувати як на персональному комп’ютері, так і на мобільному телефоні чи планшеті.

ВИСНОВКИ

У ході виконання дипломного проекту були досліджені та випробувані методи створення Web-додатку з рекомендаційною системою контенту. Проведено дослідження ефективності використання різних технологій для реалізації функціонального і готового до експлуатації Web-додатку. Для цього було проаналізовано переваги програмного забезпечення, редакторів вихідного коду, фреймворків та засобів, які допомагають полегшити розробку.

Було виявлено причини для адаптації Web-додатку під портативні пристрої. На сьогодні, більша частина користувачів всесвітньої мережі отримує доступ до мережі за допомогою смартфона, планшета і інших ґаджетів здатних підключатись до мобільного інтернету.

Таким чином, якщо кінцевий продукт (Web-додаток) буде здатним належним чином відображати контент на екрані переносного пристрою це в рази збільшить кількість активних відвідувачів.

З цього випливає висновок – в сучасних умовах адаптація Web-додатку під різні типи дисплею є не тільки гарним доповненням до вже існуючого ресурсу, а і являється необхідністю. Але для того щоб користувачу було цікаво та зручно використовувати додаток, він має бути ще й практичним.

Тому обійтися без рекомендаційних систем просто неможливо. Вони обов’язково мають бути наявні ресурсах які безпосередньо надають людям контент. Ті ж онлайн кінотеатри, інтернет-магазини беруть до уваги можливості реалізації системи рекомендацій, оскільки тільки завдяки їм можна затримати користувача на більш довгий період і допомогти користувачеві зорієнтуватися при виборі продукту який його більше всього може зацікавити.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. agilethought.com [Електронний ресурс]. – Режим доступу: https://agilethought.com/blog/articles/how-to-build-hybrid-recommender-system/
2. cmsmagazine.ru [Електронний ресурс]. – Режим доступу: http://www.cmsmagazine.ru/library/items/graphical_design/top-responsive-web-design-problems-and-how-avoid-them/
3. medium.com [Електронний ресурс]. – Режим доступу: https://medium.com/@valkyrie_be/flexbox-an-overview-4cf819742b6a
4. prodesign.in.ua [Електронний ресурс]. – Режим доступу: https://prodesign.in.ua/2011/12/yak-stvoryty-dyzajn-sajtu-scho-orijentovanyj-na-mobilni-prystroji/
5. wikipedia.org [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Adobe_Photoshop
6. yourhtmlsource.com [Електронний ресурс]. – Режим доступу: http://www.yourhtmlsource.com/starthere/whatishtml.html
7. medium.com [Електронний ресурс]. – Режим доступу: https://medium.com/actualize-network/modern-css-explained-for-dinosaurs-5226febe3525
8. medium.com [Електронний ресурс]. – Режим доступу: https://medium.com/topic/javascript
9. wikipedia.org [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Редактор_исходного_кода
10. medium.com [Електронний ресурс]. – Режим доступу: https://medium.com/build-acl/visual-studio-code-tips-and-tricks-e37e8e1efe57
11. medium.com [Електронний ресурс]. – Режим доступу: https://medium.com/@samdias9792/introduction-to-bootstrap-4-57616767a81

12. UpWork [Електронний ресурс]. – Режим доступу: https://www.upwork.com/hiring/data/how-collaborative-filtering-works/
13.medium.com [Електронний ресурс]. – Режим доступу: https://medium.com/@abhishekj/an-intro-to-git-and-github-1a0e2c7e3a2f
14. quora.com [Електронний ресурс]. – Режим доступу: https://www.quora.com/What-are-the-types-of-recommender-system

Додаток А

Код сторінки index.html

```
<html>
    <head>
        <meta content='text/html; charset=utf-8'/>
        <title>Рекомендаційна система</title>
        <link rel="stylesheet" type="text/css"
href="basic.css">
    </head>
    <body>

        <div class="container">
            <div class="dropdown">
                <button onclick="dropFunction()"
class="dropbtn" id = 'drop_btn'>Мій список</button>
                  <div id="myDropdown"
class="dropdown-content">
                   </div>
            </div>
            <div class="header">
                 <h2>Фільми які могли б вам
сподобатись</h2><a href="#" class="refresh">(Показати
більше)</a>
            </div>
            <ul class="suggestions">
                <li class="suggestion1">
                    <img />
                    <a href="#" target="_blank"
class="moviename">this will not be displayed</a>
                    <a href = '#' class =
'addToCart'>додати до мого списку</a>
                    <a href="#" class="close
close1">X</a>
                    <br>
                    <span class =
'release_date'></span><span class =
'vote_average'></span>
                    <div class = 'overview'></div>


                </li>
                <li class="suggestion2">
                    <img />
```

```
                    <a href="#" target="_blank"
class="moviename">neither this</a>
                    <a href = '#' class =
'addToCart'>додати до мого списку</a>
                    <a href="#" class="close
close2">X</a>
                    <br>
                    <span class =
'release_date'></span><span class =
'vote_average'></span>
                    <div class = 'overview'></div>

                </li>
                <li class="suggestion3">
                    <img />
                    <a href="#" target="_blank"
class="moviename">nor this</a>
                    <a href = '#' class =
'addToCart'>додати до мого списку</a>
                    <a href="#" class="close
close3">X</a>
                    <br>
                    <span class =
'release_date'></span><span class =
'vote_average'></span>
                    <div class = 'overview'></div>

                </li>
            </ul>
            <hr />

        </div>
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1
.0/jquery.min.js"></script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/rxjs/2.2.
26/rx.js"></script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/rxjs/2.2.
26/rx.async.js"></script>
```

```
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/rxjs/2.2.
26/rx.coincidence.js"></script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/rxjs/2.2.
26/rx.binding.js"></script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/rxjs/2.2.
26/rx.time.js"></script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/rxjs-
dom/2.0.7/rx.dom.js"></script>
        <script type="text/javascript" src =
'movie.js'></script>
        <script>

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r
;i[r]=i[r]||function(){

(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
Date();a=s.createElement(o),

m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.pa
rentNode.insertBefore(a,m)

})(window,document,'script','https://www.google-
analytics.com/analytics.js','ga');

          ga('create', 'UA-77776468-7', 'auto');
          ga('send', 'pageview');

        </script>
    </body>
</html>
```

Код стилей basic.css

```
@font-face {
font-family: 'myFont';
src: url("font/SourceSansPro-ExtraLight.otf");
}

@font-face {
font-family: 'myFont';
font-weight: bold;
src: url("font/SourceSansPro-Semibold.otf");
font-size: 100%;
}

a{
    text-decoration:none;
}
a:link {
    color: green;
}
a:hover {
    color: hotpink;
}

body {
    font-family: 'myFont';
    padding: 10px;
}
h2 {
    font-weight: bold;
    display: inline-block;
    font-size: 22px;
}
.refresh {
    margin-left: 10px;
    color: green;
}
.header {
    background: #fffcf4;
    padding: 5px;
    text-align: center;
}
.suggestions {
    border: 2px solid #ECECEC;
}
ul{
    list-style-type:none;
    padding-left: 0px;
}
li {
    padding: 5px;
    margin: 5px;
    border: 1px solid gray;
    box-shadow: 5px 5px 5px gray;
}
li img {
    width: 80px;
    border-radius: 5px;
    border: 1px solid brown;
    padding: 2px;
    display: inline-block;
```

```
}
li .moviename {
    display: inline-block;
    position: relative;
    bottom: 100px;
    font-weight: bolder;
    font-size: 18px;
}
li .addToCart {
    display: inline-block;
    position: relative;
    bottom: 100px;
    left: 10px;
    font-weight: bolder;
    font-size: 16px;
}
li .close {
    display: inline-block;
    position: relative;
    bottom: 100px;
    left: 20px;
    font-weight: bolder;
    color:red;
    font-size: 16px;
}
li .release_date {
    position: relative;
    bottom: 90px;
    left: 90px;
    color:steelblue;
    font-size: 16px;
}
li .vote_average {
    position: relative;
    bottom: 90px;
    left: 100px;
    color: orange;
    font-size: 16px;
}
p{
    text-align: center;
}

.dropbtn {
    background-color: brown;
    color: white;
}

.dropbtn:hover, .dropbtn:focus {
    background-color: orange;
}

.dropdown {
    position: relative;
    display: inline-block;
    float: right;
}

.dropdown-content {
```

```
    text-align: center;
    display: none;
    position: absolute;
    background-color: #f9f9f9;
    overflow: auto;
    box-shadow: 0px 2px 2px 0px rgba(0,0,0,0.2);
    right:0;
}

.dropdown-content a {
    color: red;
    text-decoration: none;
    display: block;
}

.dropdown a:hover {background-color: skyblue}

.show {display:block;}

.dropdown-content img{
    width: 60px;
    display: inline-block;
}

.overview{
    position: relative;
    bottom: 10px;
    font-size: 16px;
}

.addToCart:link {
    color: blue;
}
.addToCart:hover{
    color: hotpink;
}

.moviename:visited{
    color: gray;
}
.close.visited{
    color: red;
}
```

Додаток В

Код сценаріїв файлу movie.js

```
var API_KEY = 'fd2011fbe1938026b40fcf7cdc3f75ab';

var refreshButton =
document.querySelector('.refresh');
var closeButton1 = document.querySelector('.close1');
var closeButton2 = document.querySelector('.close2');
var closeButton3 = document.querySelector('.close3');


var refreshClickStream =
Rx.Observable.fromEvent(refreshButton, 'click');
var close1ClickStream =
Rx.Observable.fromEvent(closeButton1, 'click');
var close2ClickStream =
Rx.Observable.fromEvent(closeButton2, 'click');
var close3ClickStream =
Rx.Observable.fromEvent(closeButton3, 'click');

Storage.prototype.setObj = function(key, obj) {
    return this.setItem(key, JSON.stringify(obj))
}
Storage.prototype.getObj = function(key) {
    return JSON.parse(this.getItem(key))
}

// localStorage.setObj('movie_cart',[]);
var cart;
window.cart = cart;
//init dropdown
function initDropdown(){
    //initialize localstorage
    cart = localStorage.getObj('movie_cart');
    if(!cart){
        cart = [];
        localStorage.setObj('movie_cart',[]);
    }
    // console.log(cart);

    var myDropdown =
document.getElementById('myDropdown');
    document.getElementById('drop_btn').textContent=
'Мій список('+cart.length+')';
```

```
    while (myDropdown.hasChildNodes()) {
        myDropdown.removeChild(myDropdown.lastChild);
    }
    var cart_length =
localStorage.getObj('movie_cart').length;
    for(var i = 0; i < cart_length; i++){
        var id = cart[i].split('#')[0];
        var imgsrc = cart[i].split('#')[1];

        var child = document.createElement('div');
        var img = document.createElement('img');
        // var imgEl = document.createElement('img');
        img.src = 'https://image.tmdb.org/t/p/w500'+
imgsrc;
        img.textContent = i;
        img.id = id;
        img.addEventListener('click',function(){

window.open('https://www.themoviedb.org/movie/'+this.
id,'_blank');
        })

        var x = document.createElement('a');
        x.textContent = 'X';
        x.id = id;
        x.addEventListener('click',function(){
            // console.log(this.id);
            var this_id = this.id;
            var newcart = cart.filter(function(c){
                return
!c.startsWith(this_id.toString());
            })

localStorage.setObj('movie_cart',newcart);
            //re-render dropdown
            initDropdown();
            return;
        })
        child.appendChild(img);
        child.appendChild(x);
        myDropdown.appendChild(child);
```

```
    }
}

initDropdown();

//dropdown function
function dropFunction() {

document.getElementById("myDropdown").classList.toggl
e("show");
}

var requestStream =
refreshClickStream.startWith('startup click')
    .map(function(){
        var page = Math.floor(Math.random()*100)+1
        var randomOffset =
Math.floor(Math.random()*20);
        return
'https://api.themoviedb.org/3/discover/movie?api_key=
'+API_KEY+'&page='+page;
    })

var responseStream = requestStream
    .flatMap(function (requestUrl) {
        return
Rx.Observable.fromPromise($.getJSON(requestUrl));
    });

function createSuggestionStream(closeClickStream) {
    return closeClickStream.startWith('startup
click')
        .combineLatest(responseStream,
            function(click, listMovies) {
                return
listMovies['results'][Math.floor(Math.random()*20)];
            }
        )
        .merge(
            refreshClickStream.map(function(){
                return null;
            })
```

```
        )
        .startWith(null);
}

var suggestion1Stream =
createSuggestionStream(close1ClickStream);
var suggestion2Stream =
createSuggestionStream(close2ClickStream);
var suggestion3Stream =
createSuggestionStream(close3ClickStream);


// Rendering ------------------------------------
-----------
function renderSuggestion(suggestedMovie, selector) {
    var suggestionEl =
document.querySelector(selector);
    if (suggestedMovie === null) {
        suggestionEl.style.visibility = 'hidden';
    } else {
        suggestionEl.style.visibility = 'visible';
        var movienameEl =
suggestionEl.querySelector('.moviename');
        movienameEl.href =
'https://www.themoviedb.org/movie/'+suggestedMovie.id
;
        movienameEl.textContent =
suggestedMovie.title;
        var imgEl =
suggestionEl.querySelector('img');
        imgEl.src = "";
        imgEl.src =
'https://image.tmdb.org/t/p/w500'+
suggestedMovie.poster_path;

        var overviewEl =
suggestionEl.querySelector('.overview');
        overviewEl.textContent =
suggestedMovie.overview;

        var releaseEL =
suggestionEl.querySelector('.release_date');
```

```
        releaseEL.textContent =
suggestedMovie.release_date;

        var voteEL =
suggestionEl.querySelector('.vote_average');
        voteEL.textContent =
suggestedMovie.vote_average+'/10';

        var addToCartEl =
suggestionEl.querySelector('.addToCart');

        // dirty way to solve the 'overlap-element'
bug
        addToCartEl.id =
suggestedMovie.id+'#'+suggestedMovie.poster_path;
        addToCartEl.addEventListener("click",
function(){
            // console.log(suggestedMovie.id);
            if(cart.indexOf(this.id) < 0){
                cart.push(this.id);

localStorage.setObj('movie_cart',cart);
                //re-render dropdown
                initDropdown();
            }
        });
    }
}

suggestion1Stream.subscribe(function (suggestedMovie)
{
    renderSuggestion(suggestedMovie, '.suggestion1');
});

suggestion2Stream.subscribe(function (suggestedMovie)
{
    renderSuggestion(suggestedMovie, '.suggestion2');
});

suggestion3Stream.subscribe(function (suggestedMovie)
{
    renderSuggestion(suggestedMovie, '.suggestion3');
```