Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи

Магістр

ступінь вищої освіти

на тему: «Методи використання BigData для оптимізації персональних фінансових витрат»

Виконав: студент II курсу, групи 1ПІ-18м

спеціальності

121 – Інженерія програмного забезпечення

(шифр і назва напряму підготовки, спеціальності)-

Цимбалюк О.О.

(прізвище та ініціали)

Керівник к.т.н., доц. каф. ПЗ Майданюк В.П.

(прізвище та ініціали)

Рецензент к.т.н., доц. каф. КН Крилик Л.В.

(прізвище та ініціали)

Вінниця – 2019 року

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Ступінь вищої освіти магістр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ_____О.Н. Романюк
“____” _____20___року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Цимбалюку Олексію Олексійовичу

1. Тема роботи: Методи використання BigData для оптимізації персональних фінансових витрат
керівник роботи: к.т.н., доцент кафедри ПЗ Майданюк В. П. затверджені наказом вищого навчального закладу від “___”_____20__року №__

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи:

Тип обробки даних – шляхом сканування онлайн ресурсів;
Тип даних – доменні;
Середовище розробки –WebStorm;
Мова програмування – Swift/Javascript;
Платформа – iOS.

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ; аналіз стану питання та постановка задач дослідження; аналіз платформи iOS; проектування та розробка програмнії модулів; висновки; додатки

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):
мета і задачі роботи; аналіз сучасного стану програм фінансових помічників; аналіз програмного середовища розробки; основні результати роботи

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Майданюк В. П., к.т.н., доцент кафедри ПЗ		
5	Бальзан М.В, к.е.н.,доцент кафедри ЕПіВМ		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задачі дослідження		
2	Розробка структури та алгоритмів додатку		
3	Вибір середовища розробки, мови програмування та архітектури додатку		
4	Розробка клієнтської частини		
5	Розробка серверної частини		
6	Тестування роботи додатку		
7	Оформлення матеріалів до захисту МДР		

Студент _____ **Цимбалюк О.О.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Майданюк В. П.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Магістерська кваліфікаційна робота є закінченою розробкою додатку для оптимізації персональних фінансових витрат.

Розроблено мобільний додаток з використанням клієнт-серверної архітектури, що дозволить у зручному вигляді передавати користувачам найновіші пропозиції, акції та відкриває можливість розробки більшої кількості клієнтів для інших плаформ без необхідності змін серверу.

ANNOTATION

Master's diploma qualification work is a complete development of an application to optimize your personal financial expenses.

Mobile application was developed a using client-server architecture that will conveniently send users the latest offers, promotions and opens the possibility of developing more clients for other platforms without the need for server changes.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. В сучасному світі торгівля є рушійною силою багатьох економік світу. Розвиток торгівлі значно зріс за останні роки, завдяки глобалізації і надзвичайно швидкому розвитку інтернету. Саме завдяки глобалізації на полицях магазинів можна знайти повне різноманіття імпортних товарів. Саме наявність імпортних конкурентів змушує вітчизняних виробляти якіснішу продукцію або всіляко зменшувати ціну. Те саме відноситься до сфери інтернет-торгівлі. З розвитком інтернету все більше покупок люди стають робити онлайн. Тенденції показують, що людям вигідніше не йти до магазинів вислуховувати поради консультантів та тратити свій час на дорогу, а вигідніше в будь-який час зайти на сайт та здійснити покупку там, а потім у зручний для себе час забрати у поштовому відділенні або навіть кур'єрською доставкою. Однією з тенденцій ринку є розширення варіантів оплати. Багато власників провідних інтернет-магазинів використовують традиційні платіжні інструменти, проте попит на інноваційні методи оплати зростає. Крок назустріч цьому попиту сприяє подальшої експансії онлайн-рітейлу, зокрема завдяки Apple Pay і іншим системам безконтактного проведення платежів.[1]

Зі зростанням рівня торгівлі збільшилось кількість магазинів і в тому числі інтернет магазинів, що надало можливість для недобросовісних магазинів впливати на ранджування видачі пошуковика та тим самим захоплювати всю увагу покупців не чесними пропозиціями. Також це дає їм змогу підвищувати ціни на 3-5% так як з великою вірогідністю покупець не стане шукати далі, а зупиниться на одній першій з п'яти посилань.[2] Зросла кількість «накручених» відгуків на сайтах, що вводить в оману багатьох людей. Для цього створюють нові сайти на яких можна відслідкувати надійність продавця, актуальність ціни, тобто чи продавець не навмисне підіймав ціни до знижок, а також справжні відгуки покупців.

Більше 30% всіх онлайн-транзакцій здійснюються за допомогою мобільних пристроїв. І ця цифра продовжує зростати.[3] Хоча вже існують сайти агрегатори

цін, що сканують найпопулярніші магазини для порівняння вартості стрімкий ритм сучасного життя робить не зручне їх використання. Адже для детального аналізу необхідно мати повноцінний монітор для таблиць порівнння.

Немає ніяких сумнівів, що ринок електронної комерції продовжить рости, адже навіть в країнах Заходу він показує позитивну динаміку, не кажучи вже про Східну Європу і країнах, що розвиваються Азії. Ще більший потенціал чекає інтернет-магазини попереду, і, якщо ви замислюєтеся про такий бізнес, наведені в цій статті цифри можуть вас по-справжньому надихнути.

Головним викликом для більшості гравців e-commerce в усьому світі є зростання вимог до технологічної складової інтернет-магазинів, необхідність мультиканального просування і забезпечення безпеки особистих і платежів покупців.

Однак, як показує практика, ринок успішно знаходить рішення всіх тих проблем, які виникають в процесі його розвитку, і найбільш ефективні рішення досить швидко стають фактичними стандартами в галузі за рахунок розробки методів BigData.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання дослідження. Метою роботи є розширення простору пошуку пропозицій при плануванні персональних витрат за рахунок використання BigData.

Основними завданнями дослідження є:

– модифікація методу обробки великих об’ємів данних, що використовує парадигму MapReduce;

– поетапна розробка кожного з модулів;

– розробка розширеного методу сканування даних з використанням VPN;

– розробка системи, яка відбирає та фільтрує потрібні пропозиції.

Об’єкт дослідження – процес обробки великих об’ємів даних.

Предмет дослідження – методи обробки та програмні засоби BigData.

Методи дослідження. У процесі досліджень використовувались: теорія нереляційних баз даних, математична модель MapReduce для проведення розподіленої паралельної обробки великих масивів даних; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод обробки великих об'ємів даних , у якому, на відміну від існуючих використовується програмна модель MapReduce, що дозволило пришвидшити агрегацію вихідних даних.

2. Вперше запропоновано метод відслідковування ціни, особливість якого полягає у розширенні сканування за рахунок використання VPN, що дозволяє відслідковувати артефактні дії продавця.

Практична цінність отриманих результатів. Практичне значення отриманих результатів полягає в тому, що на основі проведених теоретичних досліджень і отриманих наукових результатів розроблено комплекс програмних засобів для оптимізації витрат. Розроблений додаток може бути корисним для пошуку акційних пропозицій, оповіщень про майбутні розпродажі, які допоможуть людям заощадити кошти.

Структура та обсяг роботи. Робота містить вступ, п'ять розділів, перелік посилань, додатки. У першому розділі визначено загальний стан питання по розробці додатків для оптимізації витрат та розглянуто аналоги проекту, обґрунтовано вибір платформи. У другому розділі було описано методи опрацювання та вибірки даних, що було використано в проекті. У третьому розділі розроблено протестовано клієнтську частину додатку. У четвертому розділі розроблено структуру бази даних та серверну частину додатку . У п'ятому розділі проведено економічне обґрунтування доцільності та економічної вигоди даного проекту. Перелік посилань містить 14 джерел. У додатках міститься технічне завдання на роботу лістинг коду та ілюстративний матеріал до захисту роботи.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

В даному розділі проводиться огляд та аналіз стану програм для оптимізації витрат на сьогоднішній день. Обґрунтована доцільність створення власного додатку.

1.1 Актуальність теми

Глобалізація та технічний прогресс змінили те, як люди роблять покупки, та планують свої витрати. Ритм життя зачасту не дозволяє багато часу проводити у походах магазинами, хочеться обрати потрібний товар - прийти і забрати в магазині без черги і, бажано, за адекватну ціну.

Одним із підходів для оптимізації витрат є цінові аггрегатори. Веб-сайт ціновий аггрегатор, який іноді називають веб-сайтом порівняння цін, є інструментом аналізу цін, торговим агентом порівняння, shopbot або торговим механізмом порівняння, - це вертикальна пошукова система, яку покупці використовують для фільтрації та порівняння продуктів на основі ціни, особливостей, відгуків та інших критеріїв. Більшість сайтів із порівнянням торгових мереж об'єднують списки товарів багатьох торгових мереж, але безпосередньо не продають самі продукти, натомість заробляють гроші на партнерських маркетингових угодах. Інтернет-продажі вже становлять шосту частину усіх продаж по світу.

Сайти порівняння цін можуть збирати дані безпосередньо у продавців. Роздрібні торговці, які хочуть продемонструвати свою продукцію на веб-сайті, надають власні списки товарів та цін, і вони співпадають з оригінальною базою даних. Це робиться сумішшю вилучення інформації, нечіткої логіки та людської праці.

Місця порівняння можуть також збирати дані через файл подачі даних. Торговці надають інформацію в електронному вигляді у встановленому форматі. Потім ці дані імпортуються веб-сайтом для порівняння. Деякі компанії сторонніх

організацій забезпечують консолідацію каналів даних, щоб сайти порівняння не повинні імпортувати від багатьох різних продавців. Партнерські мережі агрегують канали даних від багатьох продавців та надають їх на сайти порівняння цін. Багато популярних веб-сайтів для покупок надають пряму приналежність клієнту, який хоче стати партнером. Вони надають власний API партнеру, щоб показати свої продукти зі специфікаціями на веб-сайті афілійованого партнера. Це дає змогу сайтам порівняння цін монетизувати продукти, що містяться в каналах, заробляючи комісії за клік через трафік. Інші сайти порівняння цін мають угоди з торговцями та сукупними каналами, використовуючи власну технологію.

В останні роки було розроблено багато програмних рішень на полицях, які дозволяють власникам веб-сайтів брати дані про інвентаризацію веб-сайтів для порівняння цін, щоб розміщувати ціни роздрібної торгівлі (контекстні реклами) на своєму блозі чи вмісті єдиного веб-сайту. Натомість власники веб-сайтів за вмістом отримують невелику частку доходу, отриманого веб-сайтом зі порівнянням цін. Це часто називають діловою моделлю долі.

Ще один підхід - сканувати Інтернет за цінами. Це означає, що служба порівняння сканує веб-сторінки роздрібної торгівлі, щоб отримати ціни, замість того, щоб розраховувати на їх розпродаж. Цей метод також іноді називають «вискоблюванням» інформації. Деякі, в основному менші, незалежні сайти використовують виключно цей метод, щоб отримати ціни безпосередньо з веб-сайтів, які він використовує для порівняння.

Ще один підхід - збір даних - через краудсорсинг. Це дозволяє алгоритму порівняння цін збирати дані майже з будь-якого джерела без складності створення гусеничного випромінювання або логістики налаштування каналів даних за рахунок меншої повноти покриття. Сайти, що використовують цей метод, покладаються на відвідувачів, що надають дані про ціни. На відміну від форумів для обговорень, які також збирають інформацію про відвідувачів, сайти порівняння цін, які використовують цей метод, поєднують дані з відповідними вхідними даними та додають їх до основної бази даних за допомогою спільної фільтрації, штучного інтелекту чи людської праці. Вкладники даних можуть бути

винагороджені за зусилля за допомогою призів, грошових коштів чи інших соціальних стимулів.

Однак найчастіше використовується деяка комбінація цих двох підходів. Деякі пошукові системи починають поєднувати інформацію зі стандартних каналів із інформацією з сайтів, де одиниці зберігання продуктів (SKU) недоступні.

Існують емпіричні проекти, що оцінювали функціональність та продуктивність алгоритмів SSC (боти AKA). Ці дослідження демонструють, що не існує найкращого або папрсимонічного торгового бота щодо цінової переваги.

Сайти зі порівнянням цін зазвичай не стягують з користувачів будь-які послуги. Натомість вони монетизуються за допомогою платежів від роздрібної торгівлі, які вказані на сайті. Залежно від конкретної бізнес-моделі веб-сайту аггрегатора цін, роздрібні торговці або платять фіксовану плату за включення на сайт, сплачують плату щоразу, коли користувач переходить на веб-сайт продавця, або сплачують щоразу, коли користувач виконує певну дію. - наприклад, коли вони щось купують або реєструються зі своєю електронною адресою. На сайтах порівняльння цін отримують великі канали даних про товари, що охоплюють багато різних торгових мереж із партнерських мереж, таких як LinkShare та Commission Junction. Також є компанії, які спеціалізуються на консолідації каналів даних з метою порівняння цін і які стягують користувачів із доступу до цих даних. Коли продукти з цих каналів відображаються на своїх сайтах, вони заробляють гроші щоразу, коли відвідувач переходить на сайт продавця і щось купує. Результати пошуку можуть бути відсортовані за кількістю платежів, отриманих від продавців, перелічених на веб-сайті. великі сайти порівняння цін.

Окрім порівняння матеріальних товарів, деякі сайти порівнюють ціни на послуги, такі як страхування, кредитні картки, телефонні рахунки та переказ грошей.

1.2 Порівняльний аналіз аналогів

Попри те, що в наш час покупки онлайн займають все більшу частину ринку, додатків для аггрегації найкращих цін. Основним недоліком таких програм є те, що для повного функціоналу потрібно придбати або щомісячно вносити плату за користування додатком. Також не зрозумілий та незручний інтерфейс таких додатків.

BuyVia - помічник покупок, який допомагає виявити купони чи знижки та порівняти ціни на тисячі товарів улюблених торгових мереж, таких як Amazon, Walmart, Kohls тощо.

Коли ви не ходите по магазинах, BuyVia навіть географічно пропонує вам найкращі пропозиції від сусідніх магазинів.

Ще одним аналогом є програмний продукт «ShopSavvy» [4] (рис. 1.2) – hopSavvy враховує ціни як Інтернет, так і місцевих торгових мереж, щоб представити найкращу пропозицію для товару. Щоденні або щотижневі пропозиції з ваших улюблених магазинів і великих роздрібних торговців, таких як Walmart, Macy's і BestBuy, доставляються вам автоматично через додаток відповідно до ваших уподобань.

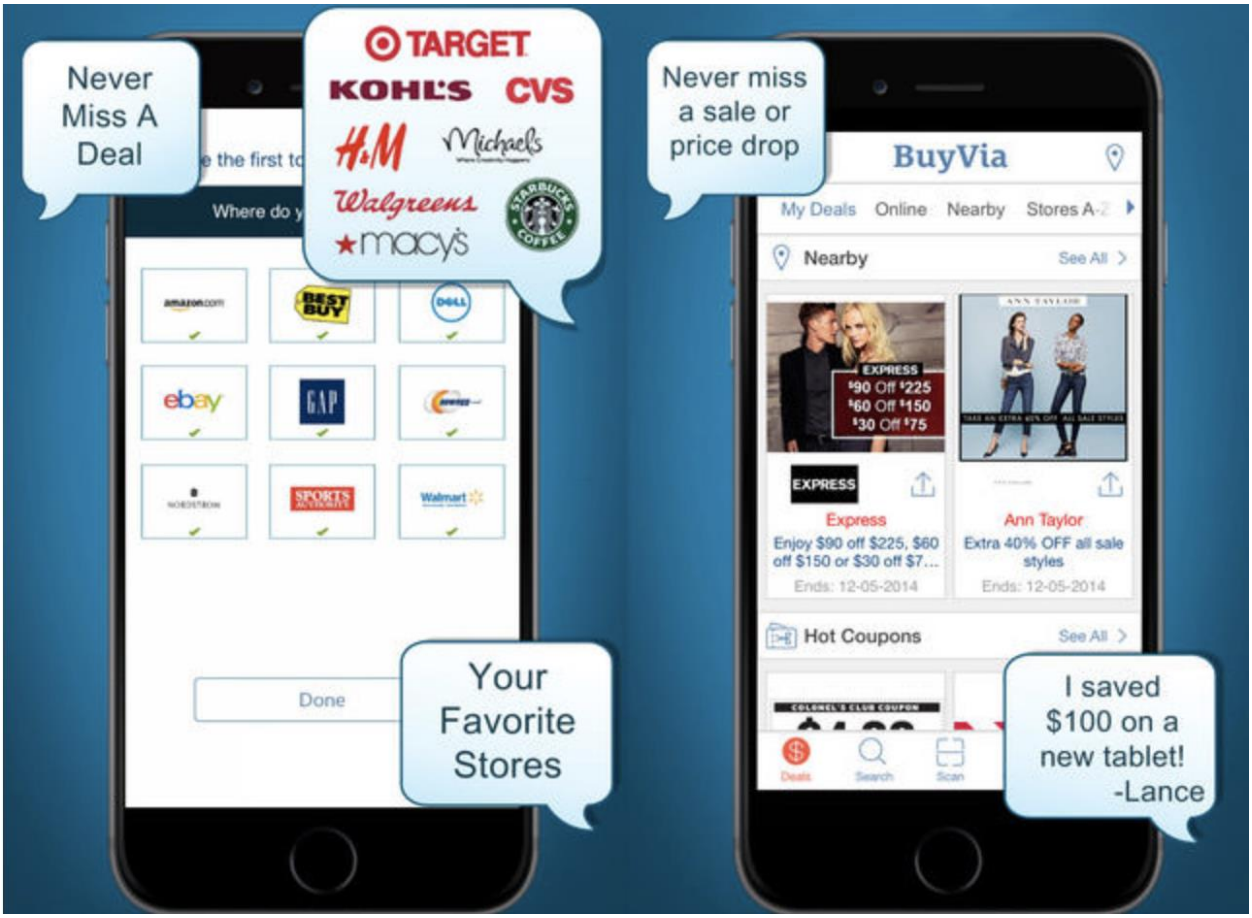


Рисунок 1.1 – Основна сторінка додатку BuyVia: зручний облік витрат

Додаток також дозволяє сканувати штрих-коди та QR-коди, щоб легко та ефективно знаходити товари.

Сканування продуктів дозволяє користувачеві сканувати предмети вдома або в магазині, щоб знайти подібні товари в сусідніх магазинах або в Інтернеті та порівняти ціни. Це також можна зробити за допомогою пошуку імені. Додаток має партнерські стосунки з понад 40 000 роздрібних торговців, серед яких Barnes & Noble, Best Buy, Nordstrom, Sears, Target та Walmart.

Огляд продуктів - показує відгуки споживачів щодо вибраних товарів та дозволяє користувачам представляти власні відгуки.

Місцеві списки - дозволяє користувачам сканувати власні продукти та продавати їх іншим користувачам.

ShopSavvy Wallet - дозволяє користувачам купувати продукти за допомогою своїх мобільних телефонів.

Пропозиції - дозволяє користувачам здійснювати пошук та перегляд пропозицій в Інтернеті та в місцевих магазинах. На вкладці "Угоди" тепер включено додаток і веб-сайт Groupon, і відображаються останні угоди Groupon

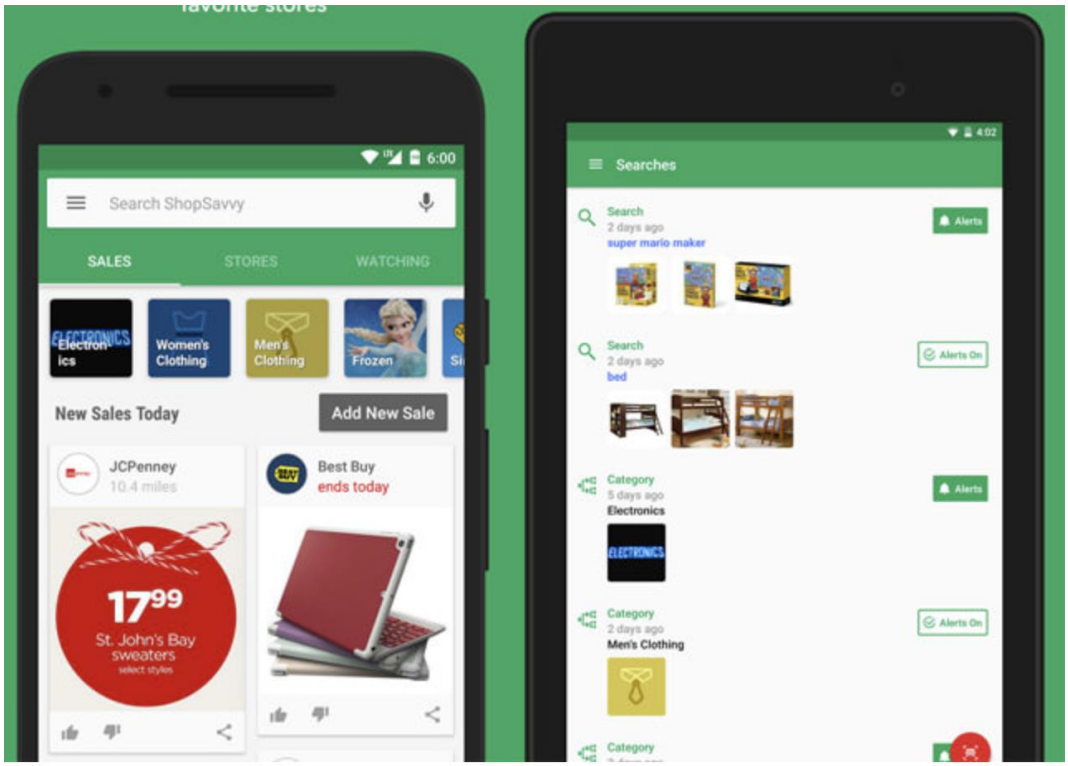


Рисунок 1.2 – Вікно програми ShopSavvy

Як і представлені аналоги, даний програмний продукт має функцію відстеження ціни проте час який іде на обробку даних та виконання власне видачі безпеосереденьо клієнту значно перевищує час, необхідний для аналогічної операції у розроблюваному додатку.

Отже, наявні системи аналоги не можуть забезпечити необхідний функціонал, повноту представлення даних, та шивдкість обробки інформації, що і обумовлює доцільність розробки власної системи для оптимізації персональних витрат.

Проаналізувавши аналоги, визначимо їхні можливості та недоліки, які враховуємо при створенні власного програмного додатку з назвою «MyFin» (табл. 1.1).

Таблиця 1.1 – Порівняльні характеристики аналогів

Критерій	BuyVia	ShopSavvy	MyDeals
Зручний інтерфейс	+	-	+
Не вимагає доступ до карток чи паролів	+	-	+
Відстеження історії ціни	-	+	+
Високий рівень безпеки	-	+	+
Повний безкоштовний функціонал	-	-	+
Bill Splitter	-	-	+

1.3 Аналіз середовищ розробки мобільного додатку

Для розробки мобільних додатків розглянемо такі середовища розробки:

- XCode;
- AppCode

AppCode - це інтегроване середовище розробки (IDE) для Swift, Objective-C, C, C ++ та JavaScript, розроблене на платформі IntelliJ IDEA JetBrains.[5] Перша загальнодоступна версія попереднього попереднього перегляду AppCode стала доступною у квітні 2011 року. Останній стабільний випуск показаний поруч зі стабільним випуском та доступний на офіційному веб-сайті JetBrains. AppCode побудований на платформі IntelliJ IDEA, яка написана на Java та Kotlin. Користувачі можуть розширити свої можливості, встановивши плагіни, створені для платформи IntelliJ, а також вони можуть писати власні плагіни.

Особлиовсті AppCode:

- Навігація по проектах та кодах: спеціальні перегляди проектів, перегляди структури файлів та швидкий перехід між файлами, класами, методами та звичками, навігація по ієрархії класів та пошук звичок.
- Перебудовування, включаючи перейменування, введення змінної, параметр вилучення / метод / параметр блоку, зміна підпису, переміщення тощо [6]
- Розробка iOS: запуск / налагодження на пристрої, симулятор iOS.
- Вбудований плагін для Reveal. теж для огляду додатка iOS з 2D / 3D візуалізацією та зміною параметрів перегляду на ходу.
- Вбудований налагоджувач з точками перерви, кадрами, годинниками та оцінкою виразів.
- Підтримка модульних тестувань: OCUnit, Kiwi, Google Test, XCTest.
- Підтримка інтернаціоналізації.

- Безшовна інтеграція CocoaPods, включаючи швидке виправлення встановлених відсутніх стручків.
- Сумісність Xcode без додаткової конфігурації: файли та зміни синхронізуються автоматично.
- Інтеграція версій управління: уніфікований інтерфейс користувача для Git, GitHub, Mercurial, Subversion, Perforce, CVS.
- Інтеграція з системами відстеження випусків: Atlassian JIRA, JetBrains YouTrack, Маяк, Pivotal Tracker, GitHub, Redmine, Trac.
- Підтримує Swift, Objective-C, C, C ++, XML, HTML, CSS, XPath, JavaScript.

Xcode - це інтегроване середовище розробки (IDE) для macOS, що містить набір інструментів розробки програмного забезпечення, розроблених Apple для розробки програмного забезпечення для macOS, iOS, iPadOS, watchOS та tvOS. Вперше випущений у 2003 році, останній стабільний реліз - версія 11.0 і доступний через Mac App Store безкоштовно для користувачів macOS Mojave. [7] Зареєстровані розробники можуть завантажувати попередні випуски та попередні версії пакету через веб-сайт Apple Developer.

Особливості XCode:

Xcode підтримує вихідний код для мов програмування C, C ++, Objective-C, Objective-C ++, Java, AppleScript, Python, Ruby, ResEdit (Rez) та Swift, з різними моделями програмування, включаючи, але не обмежуючись ними, какао, Карбон та Ява. Треті сторони додали підтримку GNU Pascal,Free Pascal, Ada, C #, Perl, та D.

Xcode може створювати жирові бінарні файли, що містять код для декількох архітектур з виконуваним форматом Mach-O. Вони називаються універсальними бінарними файлами, які дозволяють запускати програмне забезпечення як на платформах PowerPC, так і на базі Intel (x86) і які можуть включати як 32-бітний, так і 64-бітний код для обох архітектур. Використовуючи SDK iOS, Xcode також може використовуватися для компіляції та налагодження програм для iOS, які працюють на процесорах архітектури ARM.

Xcode включає інструмент GUI Instruments, який працює над динамічною рамкою трасування, DTrace, створеною Sun Microsystems і випущеною у складі OpenSolaris.

1.4 Постановка задачі для реалізації роботи

На основі проведеного аналізу стану питання додаткі для автоматизованого ведення власних фінансів у магістерській кваліфікаційній роботі потрібно виконати такі задачі:

- проаналізувати особливості та розробити структуру бази даних для мобільного додатку;
- розробити моделі та структуру мобільного додатку;
- розробити базу даних для пошуку найопитиманіших пропозицій
- розробити алгоритм функціонування мобільного додатку;
- здійснити програмну реалізацію мобільного додатку;
- провести тестування створеного програмного продукту.

1.5 Висновки

Аналіз стану додатків для оптимізації персональних витрат показав, що на сучасному етапі через великі об’єми даних обробка цих даних класичними методами практично неможлива. Тому доцільним є розробка додатку робота якого ґрунтувалась на альтернативних методах подання та обробки даних. Сформульовано основні завдання необхідні для розробки даного програмного продукту.

2 АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ ТА РІШЕНЬ

В даному розділі виконаний аналіз програмних засобів та рішень, за допомогою якого буде створено додаток. Детально розглянуто усі можливості, як додатку в цілому. Також проведено дослідження програмної частини, за допомогою якої буде виконуватись написання коду серверної частини додатку.

2.1 Використання концепції BigData в проектуванні системи

BigData (укр.“Великі дані”) - це галузь, яке обробляє способи аналізу, систематичного вилучення інформації з або іншим чином розбираються з наборами даних, які є занадто великими або складними, щоб їх можна було обробляти традиційним програмним забезпеченням для обробки даних. Дані з більшою вибіркою (рядки) пропонують більшу статистичну потужність, тоді як дані з більшою складністю (більше атрибутів або стовпців) можуть призвести до більш високої помилки виявлення [8]. Основні проблеми даних включають захоплення даних, зберігання даних, аналіз даних, пошук, обмін, передачу, візуалізація, запити, оновлення, конфіденційність інформації та джерело даних. Великі дані спочатку були пов'язані з трьома ключовими поняттями: об'єм, різноманітність та швидкість. Коли ми обробляємо великі дані, ми можемо не робити вибірку, а просто спостерігати і відстежувати, що відбувається. Тому великі дані часто включають дані з розмірами, що перевищують можливості традиційного звичайного програмного забезпечення для обробки протягом прийнятного часу та значення.

Поточне використання терміна великі дані має на увазі використання прогностичної аналітики, аналітики поведінки користувачів або деяких інших сучасних методів аналізу даних, які витягують значення з даних, і рідко до певного розміру набору даних. "Мало сумнівів у тому, що кількість наявних зараз даних дійсно велика, але це не найрелевантніша характеристика цієї нової екосистеми даних". Аналіз наборів даних може знайти нові кореляції з "точковими тенденціями бізнесу, запобіганням захворювань" боротьба зі

злочинністю тощо "Вчені, керівники бізнесу, лікарі, реклами та уряди регулярно стикаються з труднощами з великими наборами даних у таких сферах, як пошук в Інтернеті, фінтехніка, міська інформатика та бізнес-інформатика. Вчені стикаються з обмеженнями в роботі з електронної науки, включаючи метеорологію, геноміку, коннектоміку, складне фізичне моделювання, біологію та екологічні дослідження.

Цей термін застосовується з 90-х років, де дехто надає заслугу Джону Машею за популяризацію цього терміна. Великі дані зазвичай включають набори даних із розмірами, що перевищують можливості широко використовуваних програмних засобів для збору, обробки, управління та обробки даних протягом допустимого минулого часу. [9] Філософія великих даних охоплює неструктуровані, напівструктуровані та структуровані дані, однак головна увага приділяється неструктурованим даним. "Розмір" великих даних - це ціль, що постійно рухається, починаючи з 2012 року, починаючи від кількох десятків терабайт до багатьох зеттабайт даних. Для великих даних потрібен набір методів та технологій з новими формами інтеграції для виявлення різноманітних, складних та масових наборів наборів даних.

У визначенні 2016 року зазначено, що "Великі дані представляють інформаційні активи, що характеризуються таким високим обсягом, швидкістю та різноманітністю, що потребують конкретних технологій та аналітичних методів для їх перетворення у цінність". Аналогічно, Каплан і Хаенлейн визначають великі дані як "набори даних, що характеризуються величезною кількістю (обсягом) часто оновлюваних даних (швидкістю) у різних форматах, таких як числовий, текстовий або зображення / відео (різноманітність)". Крім того, Деякі організації додають нову V, правдивість, щоб описати її, перегляд, оскаржений деякими галузевими органами. Три Vs (об'єм, різноманітність та швидкість) було додатково розширено до інших додаткових характеристик великих даних:

Машинне навчання: великі дані часто не запитують чому і просто виявляють шаблони

Цифровий слід: великі дані часто є безоплатним побічним продуктом цифрової взаємодії

Для того аби підходи Big Data були відчутними дані вашого продукту мають відповідать таким характеристикам їх ще називають "п'ять V"[10]:

Volume (об'єм) – накопичена база даних охоплює настільки великий обсяг інформації, що його практично нереально обробляти та зберігати традиційними способами. Для них потрібен зовсім новий підхід та вдосконалені інструменти.

Velocity (швидкість) – ця характеристика вказує на швидкість накопичення даних, яка постійно збільшується. Наприклад, 90 відсотків всієї інформації, якою оперує людство, зібрано за останні два роки. Також ця характеристика має на увазі швидкість обробки даних. Останнім часом збільшується попит на технології, що дозволяють використовувати обробку даних в режимі реального часу.

Variety (різноманітність) – можливість одночасно обробляти структуровану та неструктуровану інформацію. Структурована інформація – це така, яку можна класифікувати. Наприклад, це може бути інформація з банківської бази даних, де чітко вказаний перелік клієнтів та їхні фінансові транзакції.

Неструктурована інформація охоплює різноманітні масиви даних, такі як фото, відео, текстові записи та інші дані. Найкращим прикладом є соціальні мережі. Її об'єм складає приблизно 80 відсотків від всієї інформації. Неструктурована інформація потребує комплексного аналізу перед можливістю її використання.

Veracity (достовірність) – оскільки обсяг інформації постійно збільшуються, важливе місце займає виокремлення достовірних даних. Якість зафіксованих даних може сильно відрізнятися, тим самим впливаючи на точний аналіз.

Variability (мінливість) – невідповідність інформації ускладнює та подекуди заважає процесам обробки та управління даними.

Кілька років тому типовий покупець в Інтернеті, який бажає придбати одяг або електронний гаджет, би прокручував Інтернет, щоб порівняти ціни з різних сайтів електронної комерції та шукати найкращі доступні пропозиції. Ціни на один і той же товар сильно відрізнятимуться від різних сайтів, що в свою чергу

спонукає сайти стежити за зміною цін у режимі реального часу за допомогою аналітиків. У сьогоднішньому сценарії гіперконкурентної електронної комерції гравці роздрібної торгівлі користуються всіма можливостями для залучення потенційних клієнтів. Веб-сайти для порівняння цін є однією з можливостей використання електронних роздрібних торговців для залучення клієнтів.

Веб-сайти для порівняння цін - це вигідна і для власника бізнесу, і для клієнта. Клієнти виграють завдяки вигідним пропозиціям, зручному досвіду покупки, кращому охопленню основних веб-сайтів електронної комерції та показ великої різноманітності для того ж продукту. З іншого боку, власники бізнесу отримують вигоду за рахунок отримання більшої кількості потенційних клієнтів, кращих коефіцієнтів конверсії та покращеного обслуговування клієнтів.

Веб-сайти для порівняння цін проглядають різні сайти електронної комерції, щоб зібрати дані про товари та послуги, такі як ціни, описи, функції, огляди тощо. Потім ця інформація об'єднується на веб-сайті зі порівнянням цін, а результати підбираються відповідно до запиту відвідувача.

Таким чином, коли покупець здійснює пошук товару на веб-сайті порівняння цін, сайт порівнює та відображає списки одного і того ж товару від багатьох продавців. Потім покупець може порівняти списки на основі цінових пропозицій, особливостей, витрат на доставку тощо, щоб знайти найкращу пропозицію.

Алгоритми, що регулюють функції порівняння цін, використовують дані як вхідні дані. Витяг та оновлення цих даних у режимі реального часу є складним завданням для двигунів порівняння цін. Більше того, сайти електронної комерції використовують динамічне ціноутворення, яке часто змінюється і потребує негайного оновлення за допомогою механізмів порівняння цін. Наприклад, Amazon займає рівно дві хвилини, щоб адаптуватися до зміни цін, і це на 43000 хвилин швидше, ніж в середньому по галузі.

Проблеми в отриманні та використанні даних для порівняння цін включають:

Технологія порівняння: Дані різних продавців матимуть різні структури. Побудувати двигун порівняння буде важким завданням

Обсяг даних: Ви будете обробляти великий об'єм даних, що робить його складним завданням.

Дуже багато стартапів працюють з Datahut для отримання даних для їх порівняння цін. Дані про продукти з сайтів електронної комерції збираються за допомогою наступних способів:

Оновлення від торговців. Веб-сайти порівняння цін вступають у співпрацю з магазинами або компанією для отримання даних безпосередньо з веб-каналів роздрібної торгівлі. Трафік від веб-сайтів для покупок є основним джерелом доходу для інтернет-магазинів. Якщо ви можете побудувати з ними партнерство - вони можуть надати вам доступ до їх API. Однак вам потрібно буде заплатити премію, щоб фактично використовувати API на більшому обсязі[11]. Торговці налаштовують або API, або використовують FTP для прямої доставки даних.

Примітка. Один із поширених способів монетизації сайтів порівняння цін - це стати партнером-партнером та отримати комісію за направлення.

Хоча цей спосіб збору даних забезпечує прямий доступ до даних, він має свої мінуси. Часто отримувати оновлення змін у реальному часі стає важко. Крім того, різні торговці надають дані в різних форматах, що ускладнює інтеграцію на одну платформу.

Лента продуктів від сторонніх API порівняння цін-google-shopping

Дані щодо роздрібної продукції також можна отримати через сторонні API. Інтегруючись із кошиками для покупок, деякі служби надають дані електронної комерції через запит API. У цьому випадку сторонні послуги стягують плату за обсяг вилучених даних. Ці компанії активно сканують найпоширеніші веб-сайти роздрібної торгівлі та надають ці дані в базу даних для легкого доступу на веб-сайтах порівняння цін.

Цей метод економить багато часу на розробку та дозволяє легко інтегрувати двигуни порівняння цін з кількома кошиками для покупок. Однак найбільшим завданням інтеграції API є те, що це складний і дорогий процес при розробці

декількох інтеграцій, дотримуючись специфікацій кожної торгової платформи. Покупки-гіганти, такі як Magento, WooCommerce, OpenCart, Shopify тощо, можуть бути інтегровані через API2Cart, який надає єдиний API для підключення всіх кошиків для покупок одночасно.

Веб-скребкування. У більшості випадків варіанти 1 і 2 можуть бути недоступними, коли ви тільки починаєте роботу та маєте менше коштів. У таких випадках веб-скребкування - це один із надійних та ефективних способів отримання даних про продукцію з цільових сайтів відповідно до ваших вимог.

Ви можете отримати цінові дані, побудувавши власний власний веб-сервісний інструмент для скребкування або використовуючи постачальника даних про послугу (DaaS), який надасть необхідні дані відповідно до вашої потреби. Для ефективних результатів на вашому сайті порівняння цін якість даних має бути найвищого класу, а оновлення наборів даних із затримкою стає важливою вимогою. Оскільки керування веб-сайтом порівняння цін є самим громіздким завданням, радимо звернутися до постачальника послуг веб-вискоблювання для отримання даних про ціноутворення.

Веб-сервіси зі скребку можуть використовувати скануючих ботів, щоб регулярно збирати інформацію з веб-сайтів для подальшої обробки. Це моментально забезпечує механізм порівняння цін точною та оновленою інформацією з цільових сайтів, не маючи потреби залежати від продавців для отримання даних. Однак на багатьох веб-сайтах є спеціальні механізми проти сканування, які блокують користувачів, які надто багато запитів.

2.2 Розробка метод обробки данних, що використовує парадигму MapReduce

MapReduce - це модель програмування та пов'язана з нею реалізація для обробки та генерації великих набори даних Користувачі задають функцію карти, яка обробляє а пара ключ / значення для створення набору проміжного ключа / значення пари та функція зменшення, яка об'єднує всі проміжні значення,

пов'язані з тим же проміжним ключем. Багато завданнь реального світу виразні в цій моделі, як показано в папері.

Програми, написані у цьому функціональному стилі, автоматично паралелізуються та виконуються на великому кластері машин. Система часу роботи піклується про подробиці розподілу вхідних даних, планування виконання програми на наборі машин, обробка несправностей на машині та управління необхідними між-машинами спілкування. Це дозволяє програмістам без будь-якого досвід паралельних та розподілених систем для легкого використання ресурсів великої розподіленої системи. Реалізація MapReduce працює в основному кластері машин і є високомасштабованимним.MapReduce може обробляти багато терабайт даних на тисячах машин для типових обчисленнь. Програмісти знайшли систему простою у використанні. Впроваджено сотні програм MapReduce що виконують понад тисячі завдань MapReduce у кластерах Google щодня.

Обчислення вибирає набір вхідних пар ключ / значення та виробляє набір пар виводу ключ / значення. Користувач бібліотеки MapReduce має імплементувати обчислення для двох функцій: Map та Reduce.

Map, написана користувачем, бере вхідну пару і виробляє набір проміжних пар ключів / значень. Бібліотека MapReduce об'єднує всі проміжні значення, пов'язані з тим же проміжним ключем I, і передає їх до функції Reduce.

Функція Reduce, також написана користувачем, приймає проміжний ключ I і набір значень для цього ключа. Це зливаються разом ці значення, утворюючи можливо менші набір значень. Зазвичай просто нульове або одне вихідне значення створено за зменшення виклику. Проміжні значення надходять до функції скорочення користувача через ітератор. Це дозволяє нам обробляти списки значень, які теж є великий, щоб вписатись у пам’ять

2.3 Розробка методу сканування даних з використанням VPN

VPN (скорочення від англ. Virtual Private Network — віртуальна приватна мережа) — узагальнююча назва мереж, що створюються поверх інших мереж, які мають менший рівень довіри. VPN-тунель, який створюється між двома вузлами, дозволяє приєднаному клієнту бути повноцінним учасником віддаленої мережі і користуватись її сервісами — внутрішніми сайтами, базами, принтерами, політиками виходу в Інтернет. Безпека передавання інформації через загальнодоступні мережі реалізується за допомогою шифрування, внаслідок чого створюється закритий для сторонніх канал обміну інформацією. Технологія VPN дозволяє об'єднати декілька географічно віддалених мереж (або окремих клієнтів) в єдину мережу з використанням для зв'язку між ними непідконтрольних каналів. Багато провайдерів пропонують свої послуги як з організації VPN-мереж для бізнес-клієнтів, так і для виходу в мережу Інтернет. VPN є клієнт-серверною технологією.

Однією з головних проблем, з якою ви можете зіткнутися під час спроби витягу даних з веб-сайтів за допомогою автоматизованих методів, є веб-сервер, що блокує IP вашого комп'ютера, тим самим позбавляючи доступу до завантаження його сторінок. На багатьох веб-сайтах є механізми для виявлення автоматизованого скреблінгу даних за допомогою програмного забезпечення та блокування IP-комп’ютерів з місця їх запуску. Крім того, під час скреблінгу даних, ви не можете розкривати свою особу (дані мережі) на віддалених веб-серверах.

Найкраще рішення, щоб уникнути блокування та захистити свою конфіденційність, - це використовувати проксі-сервери або VPN під час скреблінгу даних. Вони допомагають вам залишатися анонімними під час збору даних, а також уникнути їх заблокування.

2.4 Висновки

В даному розділі проведено огляд програмних засобів і методів, на основі яких виконується магістерська кваліфікаційна робота. Повністю описані її функціональні можливості та складові. Розглянуто принципи розробки мобільних додатків. Описано використання концепції BigData, її переваги і недоліки. Також проведений аналіз методів сканування з використанням VPN, що дає змогу відстежити артефактні дії продавця.

3 РОЗРОБКА МОБІЛЬНОГО КЛІЄНТА

3.1 Проектування та розробка меню та лендінг сторінки

Після аналізу технічних вимог до дипломного проекту та побудови алгоритмів серверу було розпочато розробку з меню та лендінг сторінки, адже саме її бачать користувачі коли відкривають додаток.

iOS - це мобільна ОС Apple, яка працює на пристроях iPhone, iPad, iPod Touch. Apple надає інструменти та ресурси для створення додатків та аксесуарів для iOS для цих пристроїв. Як розробник iOS ви можете програмувати на рідних мовах, таких як Swift або Objective-C, або створювати нативні додатки між платформами, використовуючи React Native (JavaScript) або Xamarin (C # & F #).

Для розробки додатків для iOS вам потрібен комп'ютер Mac з найновішою версією Xcode. Xcode - це IDE Apple (інтегроване середовище розробки) для додатків Mac та iOS. Xcode - це графічний інтерфейс, який ви використовуєте для написання програм для iOS. Xcode включає SDK для iOS, інструменти, компілятори та рамки, необхідні спеціально для розробки, розробки, написання коду та налагодження програми для iOS. Для розробки вітчизняних мобільних додатків на iOS Apple пропонує використовувати сучасну мову програмування Swift.

Дослідіть інструменти, технології, можливості та мови, включені в SDK для iOS, які роблять можливою розробку додатків. Деякі основні елементи iOS SDK - це структури Cocoa Touch, які включають UIKit, GameKit, PushKit, Foundation Kit та MapKit. Ці рамки та інші дозволяють вам маніпулювати камерою iPhone або iPad, додавати голосову взаємодію за допомогою SiriKit, досліджувати музику за допомогою MusicKit, розширювати перегляд та прослуховування через AirPlay 2 і навіть додавати iMessage Business Chat до своєї програми. iOS 11 додав потужність машинного навчання за допомогою Core ML та досвіду розширеної реальності (AR) з ARKit.

Важливо зазначити, що Xcode працює лише на Mac OS X і єдиний підтримуваний спосіб розробки додатків для iOS.

Ефективні методи розробки додатків нерозривно пов'язані з нативним кодом, оптимізацією, дизайном, рефракторінгом. Кожна мова програмування є унікальною і копіює методи та класи. У парадигмі програмування кожна мова пов'язана з одним поняттям, принципом та абстракцією, що визначає фундаментальний стиль програмування. Рідними мовами для iOS є мови розмітки Swift та Objective C [15].

Ефективні методи програмування під iOS:

- використання порожних елементів розмітки TextField з нульовою висотою і шириною з параметром «centerInParent» рівним «true» для вирівнювання елементів по центру екрану;
- використання елегантних способів доступу до даних, наприклад, values [SensorManager.DATA_X];
- використання розмітки RelativeLayout і властивість «fill_parent» для цієї розмітки;
- використання кількох робочих моніторів для програмування;
- уникнення створення купи об'єктів, використання існуючих об'єктів;
- зберігання даних в Cassandra, якщо їх об'єм великий;
- використання кольорової розмітки для об'єктів. Встановлення кольору фону для деяких об'єктів. Це дозволяє виділити помилки та прискорити процес їх знаходження;
- використання методів onPause()/onResume() для збереження або закриття всього того, що цього вимагає;
- використання команди Source - Format для форматування XML-файлів, щоб привести їх у читабельний вигляд;
- використання плагіну XML Tools для Notepad++ для швидкого редагування XML-файлів і розуміння структури коду;
- використання Intents за допомогою окремого методу;
- використання шаблонів, які можна швидко використати для вставки в проект.

Основні принципи розробки продуктивних додатків під iOS: економія апаратних ресурсів, протоколювати і аналізувати хід виконання додатку, , ефективно працювати з виділенням пам'яті, уникати зайвих об'єктів; для констант, класів необхідно використовувати модифікатор static final і не використовувати enum там, де модифікатор не вписується.

Ні для кого не секрет, що керівництво Apple щодо перегляду додатків Apple неймовірно строге. Apple має дуже специфічне уявлення про те, які програми вони дозволятимуть у магазині, тому корисно ознайомитись з їхніми правилами, перш ніж ви навіть спробуєте зробити свій додаток. Якщо цього не зробити, ви можете витратити свій час, роблячи щось, що Apple не дозволить в App Store.

Закінчивши додаток, надішліть його в App Store, і він буде переглянуто на основі вмісту, дизайну (докладніше про це в наступному розділі) та технічних деталей. Отже, перейдіть на сторінку рекомендацій з огляду та перейдіть до читання. У Apple також є список поширених причин, які тут відхиляються. Зазвичай це через збої, зламані посилання, рекламу чи неповну інформацію. Також відомо, що Apple блокує додатки, які містять будь-який вміст для дорослих чи політичний контент. Так само багато API Apple мають свій власний набір рекомендацій щодо огляду. Отже, якщо ви збираєтесь інтегрувати свій додаток із HealthKit або Apple Pay, добре ознайомитись і з ними. Ось вони: Apple Pay GuideLines, розширення додатків, HealthKit, HomeKit.[12]

Добре пам’ятати, що Apple, як правило, дуже консервативна у процесі перегляду додатків. Швидше за все, якщо ви робите щось навіть віддалено ризиковане, воно буде відхилено, тому пам’ятайте про це, перш ніж почати робити додаток.

Крім рекомендацій Apple щодо змісту, вони також мають набір рекомендацій щодо дизайну та інтерфейсу. Apple хоче, щоб усі додатки в їхньому магазині мали певну консистенцію, і хоча це не означає хороший дизайн, це означає, що програми використовують ті самі основні елементи інтерфейсу.

Щоб зрозуміти це, перегляньте сторінку керівництва Apple щодо людського інтерфейсу. Тут ви знайдете основи того, що шукають у програмах та дизайні іконок. У них також є набір Do's і Don'ts, які переганяють цей масивний путівник вниз, щоб було легше розпочати роботу. На щастя, Apple не залишає вас повністю в темряві щодо того, як зробити добре розроблений додаток.

Лендінг сторінка, яку ви розробляєте для свого веб-додатку, буде різницею між успіхом і невдачею вашого бізнесу. Ось чому більшість конструкцій вкладають додаткові зусилля часу та зусиль під час створення лендінг сторінки. Створення візуально переконливої та ефективної цільової сторінки, яка допоможе вам збільшити конверсії в Інтернеті, не є легким завданням.

Вам також потрібно буде пояснити, що робить ваша програма та як вона може принести користь користувачам через цільову сторінку. Ці вищезгадані речі слід робити обачним чином, не порушуючи зовнішній вигляд лендінг сторінки.

Цільова сторінка робить одне добре: фокусує увагу відвідувачів на тому, що ви рекламуєте, за допомогою єдиного заклику до дії (CTA), який дозволяє людям підписуватися на отримання додаткової інформації або безпосередньо купувати ваш товар. Цільові сторінки підтримують ваші маркетингові кампанії, допомагаючи фіксувати потенційні клієнти та представляти цільові повідомлення конкретній аудиторії.

Ви можете кодувати веб-сторінку вручну, а ви можете додати одну сторінку для свого продукту в будь-яку стандартну CMS. Але вам краще зробити щось простіше. Коли ви хочете створити цільову сторінку, щоб запустити щось нове, вам потрібно:

- Шаблони, які допоможуть вам швидко зробити сторінку з сучасним дизайном.
- Інструменти для додавання в шаблони додаткових елементів - зображень, довгоформатного тексту тощо.
- Кнопка для надсилання відвідувачів на інший сайт або магазин, де вони можуть придбати ваш товар, або форму, де вони можуть зареєструватися для отримання додаткової інформації.

- Налаштоване ім'я домену для вашого сайту.
- Відстеження Analytics, щоб знати, скільки людей відвідує вашу сторінку та як вони дісталися до неї.

Є багато інструментів, які можуть допомогти вам це зробити - ми перевірили цю статтю понад 30, відтворивши сторінку нашої книги API в кожній. Найголовніше, що створює цільова сторінка, - це перетворити вашу ідею на унікальну цільову сторінку, яка виражає вартість вашого продукту приблизно за 15 хвилин, і дозволяє зосередитись на одному заклику до дії (CTA), такому як отримання інформації про вихід із форми або стимулювання продажів. Найкраще сплачені розробники цільових сторінок також допоможуть вам проаналізувати ваш трафік та тестові сторінки A / B, щоб покращити їх. Ми зробили свій вибір з урахуванням цих особливостей, а також загальну зручність у використанні, незвичайні функції та ціни.[13]

Якщо у вас є зображення або відео вашого продукту, а також написана копія, ви зможете перетворити їх на сторінку за лічені хвилини. Коли вам потрібно швидко створити сайт, який добре продає одне, це найкращі варіанти, які виділялися в нашому тестуванні.

Панель вкладок відображається внизу екрана програми та забезпечує можливість швидкого перемикання між різними розділами програми. Панелі вкладок напівпрозорі, можуть мати фоновий відтінок, підтримувати однакову висоту у всіх орієнтаціях екрана та приховані, коли відображається клавіатура. Панель вкладок може містити будь-яку кількість вкладок, але кількість видимих вкладок залежить від розміру та орієнтації пристрою. Якщо деякі вкладки неможливо відобразити через обмежений горизонтальний простір, остаточна видима вкладка стає вкладкою Більше, яка розкриває додаткові вкладки у списку на окремому екрані.

На рисинку 3.1. зображено лендінг сторінку додатку

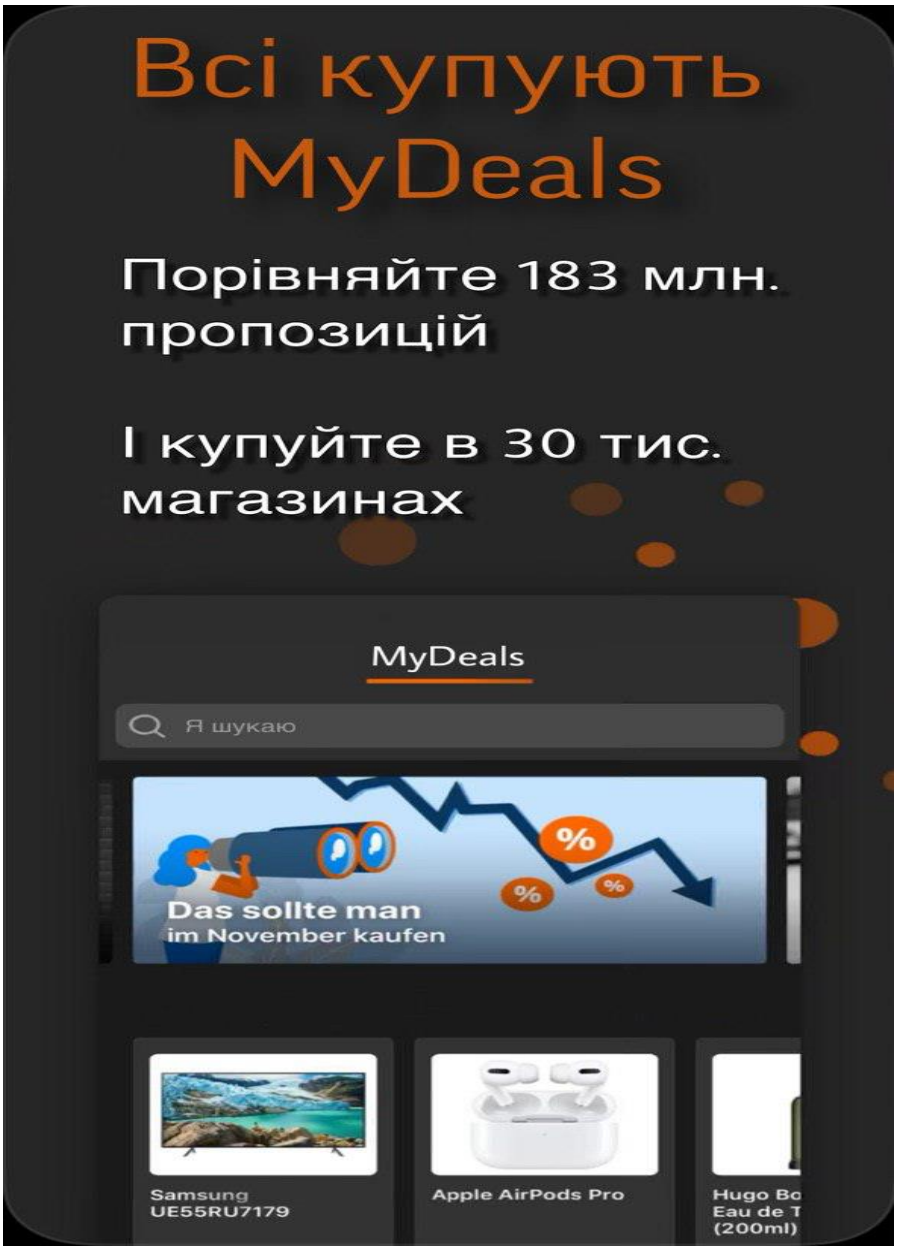


Рисунок 3.1 – Головна сторінка додатку

3.2 Розробка модуля порівняння

Для початку, нам слід визначити ключові опції при процедурі вибору товару, а це може бути наприклад: специфікація, історія ціни, ціни у інших продавців. Саме керуючись цими опціями як варіантами дії користувача було спроектовано та побудовано модуль, що являє собою по-суті сторінку товару, але ж можливістю зібрати в одному місці, все те, що скоріше за все, цікавить користувача.

Сторінка з повним описом товару, історією ціни, можливістю додати товар до відстежування, можливість додати товар в список «мені подобається», та можливістю поділитись посиланням на магазин. При прогортуванні вниз користувач може вивчити пропозиції від інших продавців, вони сортовані за ціною, а якщо ціна однакова, то першими видаються ті, у кого вищий рейтинг.

На рисинку 3.2. зображено сторінку опису і порівння товарів.

Рисунок 3.2 – Головна сторінка опису і порівння товарів

3.3 Розробка сторінки відстеження ціни

Список побажань дозволяє покупцям створювати персоналізовані колекції продуктів, які вони хочуть придбати, і зберігати їх у своєму обліковому записі користувача для подальшого використання. Списки бажань вказують на зацікавленість клієнта до товару без негайного наміру придбати.

Після того, як користувач зацікавився потенційним товаром, він може додати його до списку відстежування. Це означає, що коли сервер видасть повідомлення про знижку користувачам які додали товар до відстежування має показатись у вигляді не прочитаного повідомлення (на рівні додатку), якщо ж користувач захотів отримувати оповіщення, то його має надіслатись пуш-нотифікація з повідомлення про знижчку, тим самим спонукаючи його відкрити додаток. Тож після аналізу технічного завдання для цього функціоналу, було спроектовано та побудовано програмний інтерфейс який дозволяє користувачу інтиітивно і легко будувати список товарів ціну на які він хотів би відстежувати, також є можливість ввімкнення сповіщенньь для кожного товару індивідуально, надсилання списку відстежування друзям, та побудовано інтерфейс для подальшої можливості втілення механізму синхронізації між іншими майбутніми платформами. Окрім очевидних переваг покращення досвіду покупців від покупців, списки бажань мають можливість надавати набагато глибше стратегічне значення онлайн-роздрібним торговцям. Для клієнтів список побажань створює можливість зберегти товари "на потім", якщо вони не можуть взяти на себе зобов'язання придбати в цей момент і швидко знайти їх, коли вони повернуться у ваш магазин. Це також служить зручним способом нагадати про товари, які внесли його до списку, що особливо актуально при складанні списків подарунків або перетягуванні списків покупок для великих майбутніх подій, таких як весілля, новонароджені, новосілля тощо. Мобільні покупці вважаємо цю функцію особливо привабливою, оскільки це економить багато часу на "перегляд" на меншому екрані.

Ця функція створює величезну цінність і для інтернет-магазинів. Ті, хто зобов’язаний вийти за межі основної мети - створити безперебійну покупку, можуть використовувати потенціал списків бажань кількома способами.

Списки бажань створюють короткий огляд мислення ваших клієнтів. Ви вважаєте, що особистий список бажань одного клієнта є надто специфічним, щоб дати вам більше розуміння. Насправді це, дійсно, складено для задоволення конкретних потреб однієї людини. Однак, коли ви розміщуєте всі дані із сотень чи тисяч списків бажань для інтелектуального аналізу, з’являється цілий ряд різних цінних даних: ви можете легко визначити тенденції та оптимізувати свою тактику мерчандайзингу, щоб використовувати їх. Ви можете швидко та легко виміряти ефективність своїх наступних маркетингових кампаній електронною поштою та визначити, яка тактика дає найкращі результати: чи реагують люди на флеш-розпродажі, купони чи пропозиції безкоштовної доставки?

Ви можете легко виміряти ефективність ваших загальних маркетингових зусиль, відстежуючи, які товари потрапляють у списки бажань ваших клієнтів.

Ви можете проаналізувати дані, щоб побачити, які товари зберігають на пізніше та які предмети вони насправді купують, що допоможе вам визначити основні проблеми, що гальмують ваші продажі: чи потрібна клієнтам додаткова інформація? Правильна ціна? Вони чекають спеціальних пропозицій? І т.д.

Списки бажань дають глибокий погляд на світ вашого клієнта. Успішний бізнес в галузі електронної комерції має чітке розуміння своєї цільової аудиторії і часто використовує персонал покупців для тонкої настройки маркетингових та мерчандайзингових кампаній. Чіткий показник того, що клієнти хочуть "в даний момент", списки бажань також можуть дати унікальне уявлення про світ, мрії та прагнення клієнта. Ці дані можуть допомогти підприємствам повідомити про свої брендингові та маркетингові стратегії, встановити більш глибоку, емоційну зв’язок зі своїми клієнтами і, зрештою, стимулювати більше продажів.

Списки бажань можуть залучати багато трафіку. Здійснення спільного доступу до списку бажань - це надійний спосіб залучити багато нових відвідувачів. Це чудовий спосіб підвищити обізнаність про ваш бренд і просувати

свою продукцію, а найкраще в цьому - це безкоштовно. Загальне правило полягає в тому, що в кінцевому рахунку більше відвідувачів повинні перетворитись на більше клієнтів, тому реалізація цієї функції у вашому магазині також є надійною тактикою оптимізації доходу.

На рисинку 3.3. зображено сторінку «Зберігай і відстежуй»

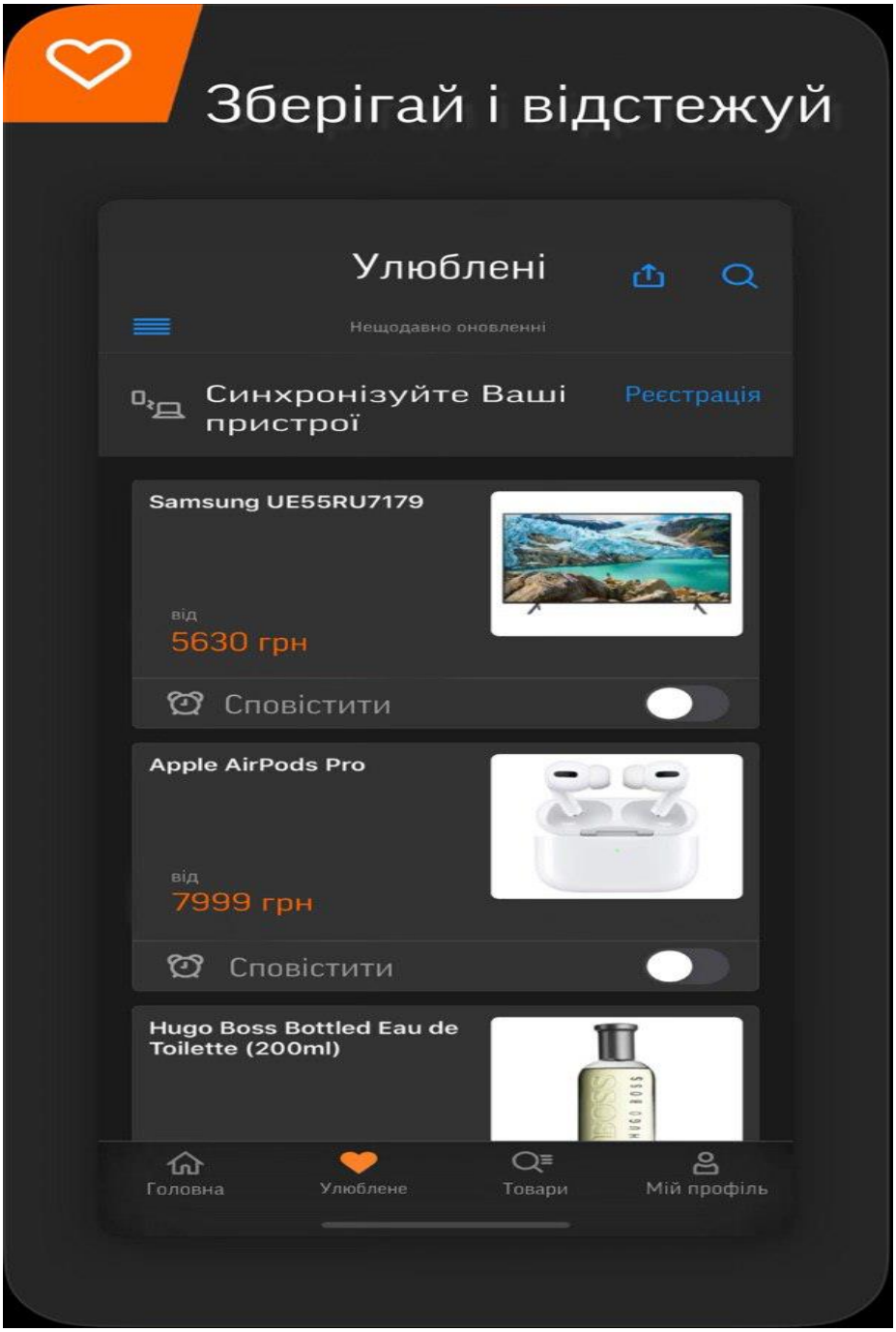


Рисунок 3.3 – Сторінка «Зберігай і відстежуй»

3.4 Розробка сторінки історії ціни

Представлення даних про історію ціни за допомогою графіків та діаграм - одна з найпомітніших особливостей мобільних додатків для трекінгу цін сьогодні правильно об’єктивно та презинтативно показати графік та правдиву ситуацією з ціною на товар. Графіки iOS роблять додатки більш красивими та помітно привабливішими.В дипломний проекту було інтегровано Swift Charts, яка місти багато шаблонів зручних графіків для мобільних додатків, таких як Шаблон програми iOS для приладної панелі і Шаблон додатків для персональних фінансів iOS.

В даний час це стало однією з найпомітніших особливостей більшості фітнес-програм або будь-яких корпоративних програм, які забезпечують відстеження, перевірку та порівняння даних користувачів. Відображення даних у табличному або колективному перегляді може бути нудним і дає обмежені можливості візуалізації даних. І з того часу інтеграція діаграм у додатки iOS - це річ, яку слід робити в цих випадках.

Залежно від мети, обираємо, який тип діаграми Swift найбільш підходить для наших функцій iOS:

Штрихові діаграми: Коли дані мають різні категорії та потребують порівняння (наприклад, щомісячні цінності продажу), то гістограма забезпечує хорошу та просту візуалізацію.

Кругові діаграми: коли категоричні дані потрібно відображати у відсотках, хороший вибір будуть кругові діаграми (а.к.а кругові діаграми). Кожна діаграма завжди являє собою ціле, таким чином, повинна дорівнювати 100%.

Лінійні діаграми: для візуального представлення змін за певний проміжок часу, наприклад, інвентар, немає нічого кращого, ніж використання лінійних діаграм. Він показує максимальний і мінімальний бали та показує поточний стан компанії.

На рисинку 3.4. зображено сторінку відстеження історії ціни

Рисунок 3.4 – Сторінка «Історія ціни»

3.5 Висновки

В даному розділі описано процес розробки клієнтського додатку для платформи iOS.

Процес розробки клієнтського додатку складався з таких етапів:

- проектування та розробка меню і навігації;
- побудова головної сторінки;
- побудова сторінки товару і порівняння;
- побудова сторінки для збереження товарів та відслідковування ціни;
- побудована сторінка історії ціни.

В процесі розробки клієнт тестувався на версії iOS 13.1.2, також основний функціонал перевірено в емуляторі на всіх версіях від 9 до 13.1.1, що покриває 99.2% пристроїв.

Робоча версія мобільного додатку відповідає технічному завданню. Інтерфейс додатку є зручним та інтуїтивно зрозумілим.

На етапі проектування клієнтської частини аналізувались особливості моделей iPhone з метою визначення версій ОС на яких даний додаток буде працювати плавно і відображатись належним чином, а на яких ні. Слід зазначити, що проект не є прив’язаним до української мови, адже всі компоненти підтримують автоматичний переклад. Крім того проаналізовано і продемонстровано використання інструментів для відображення історії ціни товарів.

4 РОЗРОБКА ТА ТЕСТУВАННЯ СЕРВЕРУ

4.1 Розробка алгоритмів роботи

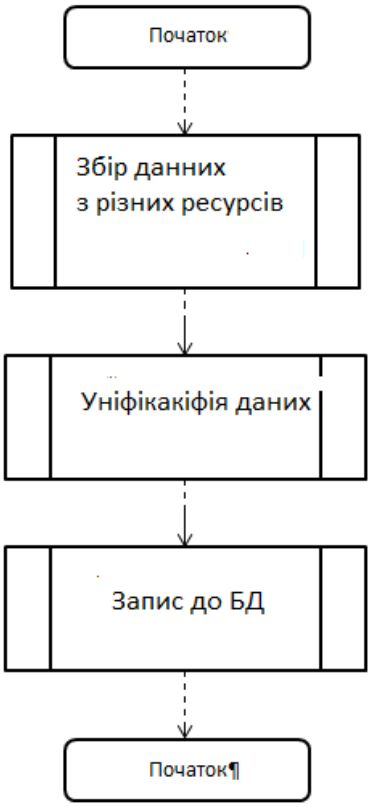


Рисунок 4.1 – Загальний алгоритм роботи зі збору та уніфікації даних

Загальна алгоритм роботи складається з періодичного сканування даних за попередньо написаними скріптами для кожного магазину. Потім уніфікується і записується в базу даних для подальшого використання. (рисунок 4.1)

Після того, дані було зібрано та аггреговано, нам слід визначити, чи є з них товари, які суттєво подешевшели з минулого сканування, якщо такі є, а вони як правило є, то викликається sendNotifications (рисунок 4.2). У цьому методі товари з суттєвою знижкою помічаються як «гарячі», тобто вони будуть відображатись на головній формі додатку. Далі для кожного з таких товарів визначаються цільові групи, це в цільові групи можна віднести користувачів, що задовільняють хоча б одній з наступних умов: додали товар до списку відтежуваних товарів, товар схожий на той, що ви раніше шукали, регіональна пропозиція в вашому регіоні, або промоушн від магазинів, що хочуть потрапити в топ. Коли ми

визначили групи користувачів, для кожної з них нам слід відправити відповідне кастомізоване повідомлення, оскільке саме відчуття турботи та індивідуально підходу цінує користувач, у подібного роду додатках.

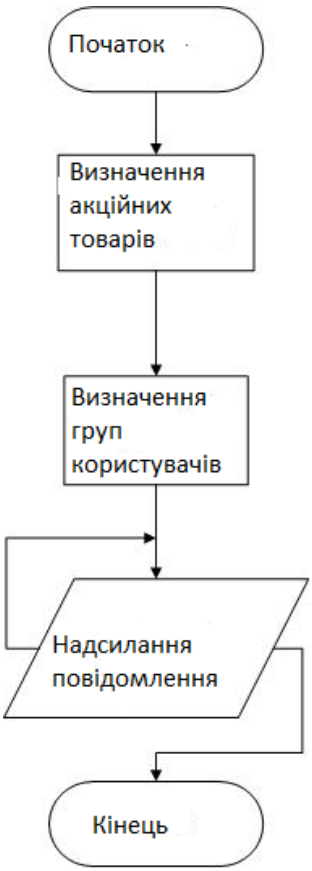


Рисунок 4.2 – Робота методу sendNotifications()

4.2 Варіантний аналіз і обґрунтування вибору фреймворку для сервера

Програмне забезпечення з відкритим кодом взагалі, і JavaScript зокрема, може здатися місцем, де бурхливість - це верховенство закону: швидке зростання, перш ніж кожен перейде до наступної великої справи. Але Node.js відрізняється. Хоча це, безумовно, не можна було б охарактеризувати як нове, і його зростання жодним чином не було драматичним, але за останні кілька років йому вдалося просунутися вперед як один з найбільш широко використовуваних інструментів JavaScript на планеті. Node.js варто вивчити, оскільки він дозволяє писати JavaScript на сервері. Це, певно, перетворило те, як ми думаємо про JavaScript.

Якщо раніше це була мова, спеціально написана на клієнті, підкріплена симпатіями PHP та Java, тепер це мова, якою ви можете користуватися у серверій частині вашого додатку.[7]

Це важливо, оскільки це означає, що команди можуть працювати набагато ефективніше разом. Використання різних мов для бекенда та фронтену, як правило, є основним джерелом незручностей і несімісності. Якщо у вас є дуже хороші розробники поліглоти, команда обмежується своїми основними навичками, тоді як інструментарій також є більш негнучким. Якщо ви використовуєте JavaScript через стек, простіше використовувати послідовний ланцюжок інструментів.

З особистої точки зору, вивчення Node.js є чудовою відправною точкою для повної розробки стеків. По суті, це як доповнення, яке неабияк розширює те, що можна зробити з JavaScript. Node.js дозволяє створювати складні та потужні програми без написання складного коду

Ще один вагомий аргумент для Node.js - це те, що він побудований для продуктивності.

Це пояснюється двома важливими речами - асинхронною архітектурою Node.js і тим, що вона використовує двигун JavaScript V8. Важливість цього полягає в тому, що V8 є однією з найшвидших реалізацій JavaScript, яка використовується для живлення багатьох надзвичайно популярних продуктів Google в браузері (наприклад, Gmail).

Node.js потужний тим, що використовує асинхронну парадигму для обробки даних між клієнтом і сервером. Щоб уточнити, що це означає, варто порівнювати з типовою моделлю сервера додатків, яка використовує блокуючи введення / виведення - у цьому випадку програма повинна обробляти кожен запит послідовно, зупиняючи потоки, поки вони не можуть бути оброблені. Це може додавати додатку складність і, звичайно, уповільнювати програму.

Навпаки, Node.js дозволяє використовувати неблокуючі введення / виведення, в яких потоки (в даному випадку послідовні, а не одночасні), які можуть керувати декількома запитами. Якщо обробляти неможливо, це ефективно

"утримується" як обіцянка. Це означає, що він може бути виконаний пізніше, не затримуючи інші потоки.

Це означає, що Node.js може допомогти вам створити додатки значної складності, не додаючи складності вашого коду. Node.js добре підходить для створення мікросервісів.

Мікросервіси стали архітектурним стилем, що швидко розвивається, що пропонує підвищену спритність та гнучкість у порівнянні з традиційним монолітом. Переваги мікросервісів добре зафіксовані, і незалежно від того, чи підходять вони вам зараз, ймовірно, вони будуть домінувати в програмному ландшафті, коли світ відходить від монолітної архітектури. [10]

Цей факт слугує лише зміцненню аргументу про те, що слід вивчити Node.js, оскільки бібліотека настільки придатна для розвитку таким чином. Це тому, що це спонукає вас розвиватися модульно і цілеспрямовано, досить буквально, використовуючи конкретні модулі для розробки програми. Це виразно і майже не суперечить монолітному підходу до архітектури програмного забезпечення.

4.3 Написання коду серверу в середовищі WebStorm IDE

В даному випадку використовувалось середовище розробки WebStorm IDE, яке дає змогу писати програмний код на мові JavaScript. Для початку визначаються з ключовими та необхідними бібліотеками, налаштувати сервер для прийому запитів, для цього необхідно описати повідінку відповідних енпойнтів. Звісно, усе це можна описати, аж до найбазовішого рівня самому, але у сьогодення, коли час це гроші, доцільніше використовучати перевірені часом серверні фреймворки. Більшість з них уже мають свої ком’юніті, тести, що покривають близько 90-98 відсотків коду. Одним з таких фреймворків є Express.
Express – це мінімалістичний і гнучкий веб-фреймворк для додатків Node.js, що надає великий набір функцій для мобільних і веб-додатків. Маючи в своєму розпорядженні безліч службових методів HTTP і проміжних оброблювачів,

створити надійний API можна швидко і легко. Express надає тонкий шар фундаментальних функцій веб-додатків, які не заважають вам працювати з давно знайомими і улюбленими вами функціями Node.js.

Перш за все слід визначити базову модель користувача, код наведено нижче:

```
const cassandra = require(cassandra);
const bcrypt = require('bcrypt');

const environment = process.env.NODE_ENV;
const stage = require('./config')[environment];

// schema maps to a collection
const Schema = cassandra.Schema;

const userSchema = new Schema({
  name: {
    type: 'String',
    required: true,
    trim: true,
    unique: true
  },
  password: {
    type: 'String',
    required: true,
    trim: true
  }
});

module.exports = cassandra.model('User', userSchema);
```

Бузумовно одним з базових але в той же час найважливішим є питання безпеки. У рамках дипломного проекту було прийнято рішення використовувати bcrypt для хешування паролів користувачів, перш ніж ми їх збережемо.

Наведений нижче код, демонструє алгоритм попереднього хешування паролю при створенні користувача. Завдяки зв’язці цього методу і схеми користучаів, що дозволяє зробити Express, немає необхідності налаштовувати особливості такого алгоритму.\

```
userSchema.pre('save', function(next) {
  const user = this;
  if(!user.isModified || !user.isNew) { // don't rehash if it's an old user
    next();
  } else {
    bcrypt.hash(user.password, stage.saltingRounds, function(err, hash) {
      if (err) {
        console.log('Error hashing password for user', user.name);
        next(err);
      } else {
        user.password = hash;
        next();
      }
    });
  }
});
```

Після цього додамо контролер входу, який буде обробляти наші запити до маршруту / login.

```
login: (req, res) => {
   const { name, password } = req.body;

    cassandra.connect(connUri, { useNewUrlParser: true }, (err) => {
```

```
let result = {};
let status = 200;
if(!err) {
  User.findOne({name}, (err, user) => {
    if (!err && user) {
      // We could compare passwords in our model instead of below
      bcrypt.compare(password, user.password).then(match => {
        if (match) {
          result.status = status;
          result.result = user;
        } else {
          status = 401;
          result.status = status;
          result.error = 'Authentication error';
        }
        res.status(status).send(result);
      }).catch(err => {
        status = 500;
        result.status = status;
        result.error = err;
        res.status(status).send(result);
      });
    } else {
      status = 404;
      result.status = status;
      result.error = err;
      res.status(status).send(result);
    }
  });
} else {
```

```
      status = 500;
      result.status = status;
      result.error = err;
      res.status(status).send(result);
    }
  });
}
```

Після того, як користувачі можуть реєструватись та авторизуватись в додатку, необхідним є написання алгоритмів пошуку та сканування ціни. Для цього було прийнято рішення використовувати бібліотеку Scrape. Scrape дозволяє сканувати найкращі сайти електронної комерції (Amazon, Walmart, Target, Best Buy), повернутати основну інформацію про товар (назва, ціна, зображення, опис) та має простий у використанні API. Нижче наведено приклад отримання інформації про продукт на Amazon.

```
import Scraper from "@jonstuebe/scraper";

3 (async () => {
  const data = await Scraper('http://www.amazon.com/gp/product/B00X4WHP5E/');
  console.log(data);
})();
```

4.4 Тестування серверу

Для тестування серверу було використано комплексний підхід який полягає в умовному розділенні плану тестування на такі: модульне тестування та інтеграційне тестування.

Найменші частини програми називаються модулів, тестування цих модулів необхідне, щоб перевірити, чи придатні вони для використання, чи ні, називається модульний тестуванням (unit testing). Якщо ми збираємося створити тест на будь-яку функцію, то нам потрібно переконатися, що функція сама по собі, окремо від усього навколо, повинна робити те, що вона призначена, не більше, не менше і знущатися над рештою, що є не піддається тестуванню. І це основний принцип одиничного тестування. У контексті тестування модулів тестування взаємодій між

двома модулямими називається інтеграційним тестуванням. Сценарії на зразок функції під тестом викликають іншу функцію з деяким контекстом. Нам все ж слід знущатися над зовнішніми ресурсами, але потрібно перевірити ці інтеграційні посилання.

Для тестування сучасних веб-сервісів не має сенсу використовувати самописні рішення, адже уже існують фреймовки в яких передбаченні всі особилості мови, та спрощені більшість базових перевірок. Для модульного тестування серверної частини дипломного проекту було обрано фреймворк Mocha.

Для встановлення використано команди:

npm install --global mocha

npm install --save-dev mocha

Приклад тесту з Mocha

```
var assert = require('assert');


describe('Basic Mocha String Test', function () {
 it('should return number of charachters in a string', function () {
        assert.equal("Hello".length, 4);
    });


 it('should return first charachter of the string', function () {
        assert.equal("Hello".charAt(0), 'H');
    });
});
```

Рисунок 4.5 – Приклад тесту з Mocha

У наведеному вище фрагменті тесту

assert допомагає визначити статус тесту, воно визначає збій тесту.

describe - функція, яка містить збір тестів. Він має два параметри, перший - значуще ім'я функціональності, що перевіряється, а другий - функція, яка містить один або кілька тестів. Ми можемо вкласти також опис.

It - також знову функція, яка фактично є самим тестом і приймає два параметри, перший параметр - це ім'я тесту, а другий параметр - це функція, яка містить тіло тесту.

Кожна функція виконує конкретне завдання. Щоб перевірити функцію, функцію потрібно викликати з тестового або специфікаційного файлу з необхідними входами. Тоді ми поставимо ствердження, щоб перевірити вихід або завдання функції.

Далі проводиться тестування методу isValidUserId (рисунок 4.6).

```
function LoginController() {

function isValidUserId(userList, user) {
   return userList.indexOf(user) >= 0;
}

return {
    isValidUserId
  }
}
module.exports = LoginController();

/* Test */
it('should return true if valid user id', function(){
      var isValid =
loginController.isValidUserId(['abc123','xyz321'], 'abc123')
      assert.equal(isValid, true);
});
```

Рисунок 4.6 – Тестування методу isValidUserId

Тестування асинхронної функції (колбеків)

Під час тестування колбеків єдина головна відмінність, ми повинні сказати Mocha, що тест завершений через асинхронний характер перевіреної функції[11]. У наведеному на рисунку 4.7 прикладі Mocha чекає, коли буде викликана функція done () для завершення тесту.

```
function isValidUserIdAsync(userList, user, callback) {
    setTimeout(function(){
      callback(userList.indexOf(user) >= 0)
    }, 1);
}
Note: setTimeout has been used to simulate the async behavior.

/* Test */
it('should return true if valid user id', function(done){
  loginController.isValidUserIdAsync(['abc123','xyz321'], 'abc123',
     function(isValid){
      assert.equal(isValid, true);
      done();
  });
});
```

Рисунок 4.7 – Тестування методу isValidUserIdAsync

На рисунку 4.8 наведено результати тестування контроллеру авторизації.

```
> mocha './test/**/*.spec.js'



  LoginController
    isValidUserId
      - should return true if valid user id
      ✓ should return false if invalid user id
    isValidUserIdAsync
      - should return true if valid user id
    isAuthorizedPromise
      ✓ should return true if valid user id


  2 passing (24ms)
  2 pending
```

Рисунок 4.8 – Результати тестування контроллеру авторизації

Деякі пише тести після написання коду, деякі перед написанням коду, а деякі паралельно з кодом. Дискусійно, який підхід кращий, але врешті-решт всі згодні з тим, що тестування одиниць є критичною частиною розвитку. Отже, ми повинні знати про всі інструменти та методи модульного тестування.

Як видно з результатів тестування в методи описані раніше відповідають необхідним вимогам, зокрема це було перевірено на позитивних і негативних

данних. Тобто можна зробити висновок, що методи контроллера самі по собі є дієзаними і такими що відповідають вимогам технічного завдання.

4.5 Основні несправності та методи їх усунення

В процесі написання та тестування коду серверної дипломного проекту виникали помилки з конвертацією валют. Для повної точності у визначенні ціни, слід зберіграти курс на кожен день тому, що слід відображати фактичну ціну на кожен день, інакше графік буде кожного разу мінятись, що приведе до розбіжності. Одним з проблематичних моментів виявилось написання та налагодження скриптів для парсинсу найбільш популярних ресурсів на відкритих довідників. Річ у тім, що крім очевидної різниці у структурі даних кожного з ресурсів, деякі з них замість JSON повертають XML. Це було неочевидно на перших кроках в розробці, тому знадобилось дописати відповідний код, що перетворював би XML в JSON. Також проблемою вияловилось те, що не лише достатньо повернути користувачу інформацію про акційний товар, але й ще й метадані магазину, зокрема: умови акції, дати проведення, кількість акційного товару та інше. Було вирішено надсилати ці дані опціональним полем properties та показувати його на клієнті за наявністю.

4.6 Висновки

В середовищі WebStorm мовою програмування JavaScript розроблено серверну частину додатку. Платформою для сервера було обрано NodeJs. Розроблено основні архітектурні сутності додатку, наведено приклад захисту даних користувача. Сервер періодично проводить сканування ресурсів, уніфікує та зберігає дані для подальшого використання з метою вибору найбільш оптимальної ціни. Проведене комплексне тестування показало працездатність програмного забезпечення і повну його відповідність завданню на роботу.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Майданюк В.П. та Войтко В.В.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	3	4
4	4	4
5	3	3
6	3	4
7	4	3
8	3	4
9	4	4
10	4	4
11	3	4
12	3	4
Сума балів	$СБ_1 = 42$	$СБ_2 = 45$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 43{,}5$	

Отже, з отриманих даних таблиці 5.1 видно, що нова розробка має високий рівень комерційного потенціалу.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (5.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (5.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 21 день;

t - число днів роботи розробника, t = 40 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 5.2.

Таблиця 5.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5000	238,1	5	1190,5
Інженер-програміст	3500	166,66	40	6666,4
Всього:				7856,9

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 7856,9 = 785,69 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп}= (З_о + З_р)\cdot\frac{\beta}{100},$$

$$Н_{зп}= (785{,}69+ 7856{,}9)\cdot\frac{36{,}3}{100} = 3137{,}26\ (грн.). \quad (5.2)$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot\frac{Т}{12}, \quad (5.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 5.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп’ютер	10000	25	3	625
Всього:				625

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_i \cdot Ц_i \cdot К_i, \quad (5.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 5.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	150	1	150
Пачка паперу	уп.	100	1	100
Ручка	шт.	10	1	10
Всього з урахуванням транспортних витрат				286

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п ; \quad (5.5)$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=180 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0,8$).

$$В_е= 1,7 \cdot 0,6 \cdot 180 \cdot 0,8 = 146,88 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (5.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (7856,9 + 785,69) = 8642,59 \text{ (грн).}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в,$$

В = 7856,9+785,69+3137,26+625+286+146,88+8642,59 = 21480,32 (грн.).

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha}, \quad (5.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{21480,32}{1} = 21480,32,$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (5.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{21480,32}{0,9} = 23867,02 \text{ (грн.)}$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i , \quad (5.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 20 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 20 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 300 користувачів, протягом другого року – на 200 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 1500 користувачів, а

прибуток, що отримував розробник до впровадження результатів наукової розробки – 500 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 20 \cdot 1500 + (500 + 20) \cdot 300 = 186000 \text{ (грн)}.$$

Протягом другого року:

$$\Delta П_2 = 20 \cdot 1500 + (500 + 20) \cdot (300 + 200) = 290000 \text{ (грн)}.$$

Протягом третього року:

$$\Delta П_3 = 20 \cdot 1500 + (500 + 20) \cdot (300 + 200 + 100) = 342000 \text{ (грн)}.$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$ вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \quad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 5.1.

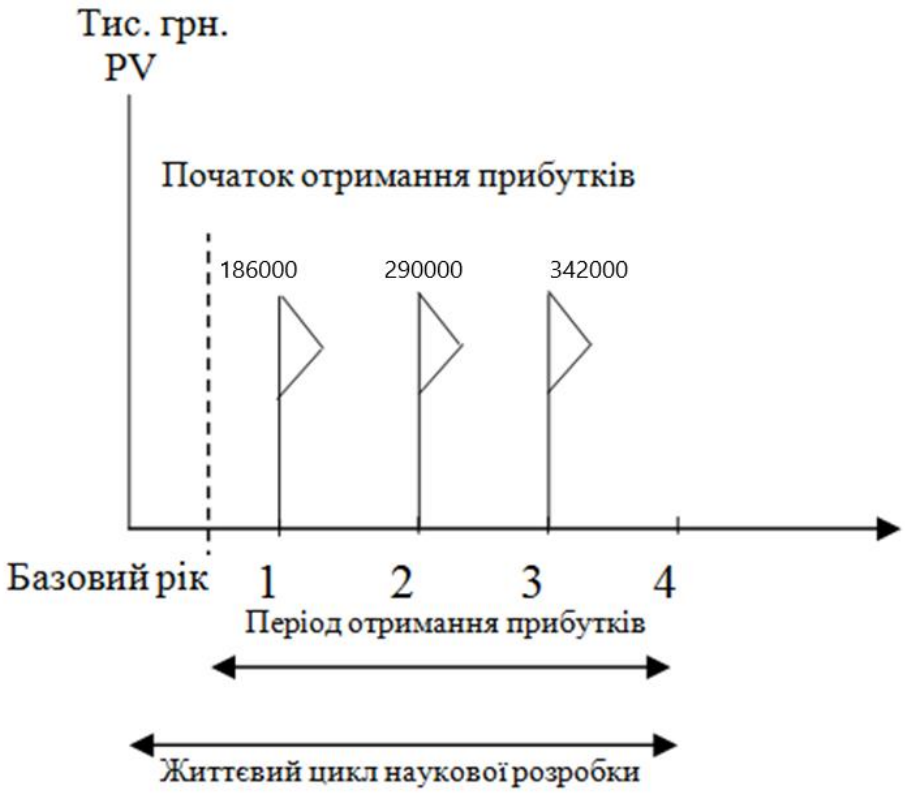


Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{23867,02}{(1+0,1)^0} + \frac{186000}{(1+0,1)^2} + \frac{290000}{(1+0,1)^3} = 629057,92 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$E_{абс} = 629057,92 - 23867,02 = 605190,9 \text{ (грн.)}.$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[T]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (5.12)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{605190,9}{23867,02}} - 1 = 1,97 \text{ або } 197\ \%.$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3 \text{ ,}$$

Оскільки $E_в$ = 197% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в},$$

$$T_{ок} = \frac{1}{1{,}97} = 0{,}50(\text{року}).$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

В даній магістерській кваліфікаційній роботі розроблено додаток, що використовує методи BigData для оптимізації персональних фінансових витрат.

Основні результати отримані під час виконання магістерської кваліфікаційної роботи такі:

1. Аналіз стану додатків для оптимізації персональних витрат показав, що на сучасному етапі через великі об’єми даних обробка цих даних класичними методами практично неможлива. Тому доцільним є розробка додатку робота якого ґрунтувалась б на альтернативних методах подання та обробки даних.
2. Огляд програмних засобів та методів, на основі яких виконується магістерська кваліфікаційна робота показав, що найбільш прийнятною є форма зберігання даних у вигляді колонок, використання програмної парадигми MapReduce, що дозволяє ефектно уніфікувати та зберігати дані.
3. Описано використання концепції BigData, її переваги і недоліки. Показано, що використання цієї конценпції дає приріст в швидкості обробки від 25% до 45% на великих об’ємах даних.
4. Аналіз методу сканування з використанням VPN, показав що він дає змогу відстежити артефактні дії продавця.
5. Показано, що процес розробки клієнтського додатку для iOS складався з таких етапів:

– проектування та розробка меню і навігації;
– побудова головної сторінки;
– побудова сторінки товару і порівняння;
– побудова сторінки для збереження товарів та відслідковування ціни;
– побудована сторінка історії ціни.

6. На етапі проектування клієнтської частини аналізувались особливості моделей iPhone з метою визначення версій ОС на яких даний додаток буде працювати плавно і відображатись належним чином, а на яких ні.
7. Розгляд питань локалізації додатку показав, що проект не є прив’язаним до української мови, адже всі компоненти підтримують автоматичний переклад.
8. Проаналізовано і продемонстровано використання інструментів для відображення історії ціни товарів.
9. В середовищі WebStorm мовою програмування JavaScript розроблено серверну частину додатку. Платформою для сервера було обрано NodeJs. Розроблено основні архітектурні сутності додатку, наведено приклад захисту даних користувача. Сервер періодично проводить сканування ресурсів, уніфікує та зберігає дані для подальшого використання з метою вибору найбільш оптимальної ціни.
10. В процесі розробки клієнт тестувався на версії iOS 13.1.2, також основний функціонал перевірено в емуляторі на всіх версіях від 9 до 13.1.1, що покриває 99.2% пристроїв. Робоча версія мобільного додатку відповідає технічному завданню. Інтерфейс додатку є зручним та інтуїтивно зрозумілим.
11. Проведене комплексне тестування показало працездатність програмного забезпечення і повну його відповідність завданню на роботу.
12. Економічний розрахунки показали доцільність розробки додатку і його конкурентну здатність на ринку програмних продуктів. Термін окупності розробки додатку складає 0.5 року.

ПЕРЕЛІК ПОСИЛАНЬ

1. Буда А. Г. Методичні вказівки до оформлення курсових проектів (робіт) у Вінницькому національному технічному університеті / Уклад. Г.Л. Лисенко, А.Г. Буда, Р.Р. Обертюх. В.: ВНТУ, 2006 р., 60 с
2. Grady B. Object-Oriented Analysis and Design with Applications/ B. Grady. – Amsterdam: Addison-Wesley Professional, 2009. – 720 p.
3. Horton's I. Beginning JavaSctipt/I. Horton's. –Amsterdam: Wrox, 2015. – 988 p.
4. Stellman A. Head First NodeJS / A. Stellman. – Boston: O'Reilly Media, 2014. – 784 p
5. Мови програмування — Stackoverflow. [Електронний ресурс]. – Режим доступу: https://ru.stackoverflow.com/tags/info
6. Нечітка логіка, нечіткі множини, м'які обчислення — часті питання Режим доступу http://www.znannya.org/?view=fuzzy-logic-q
7. Нечітка логіка: лекція – Режим доступу: https://www.victoria.lviv.ua/html/oio/html/theme11.htm
8. Бази знань. Інтелектуальні інформаційні системи [Електронний ресурс]. – Режим доступу: https://studfiles.net/preview/5474324/
9. Nodejs Documentation [Електронний ресурс]. Режим доступу до ресурсу: https://nodejs.org/en.
10. Fleder D., Hosanagar K. Blockbuster Culture's Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity (журнал) // Management Science, Vol. 55, No. 5, May 2009, pp. 697-712. — 2009. — P. 1 - 49.
11. Xiaoyuan Su and Taghi M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques A Survey of Collaborative Filtering Techniques (журнал) // Hindawi Publishing Corporation, Advances in Artificial Intelligence archive, USA. — 2009. — P. 1 - 19.
12. Zadeh, L. A. (1965). Fuzzy sets. *Information and Control* **8** (3): 338. doi:10.1016/S0019-9958(65)90241-X

13. Козловський В.О. Техніко-економічні обґрунтування та економічні розрахунки в дипломних проектах та роботах. Навчальний посібник. / В.О. Козловський – Вінниця: ВДТУ, 2003. – 75 с.

14. Козловський В.О. Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт / В.О. Козловський – Вінниця: ВНТУ, 2012. – 22 с.

Додаток А Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"____ "_____2019 р.

Технічне завдання
на магістерську кваліфікаційну роботу «Методи використання BigData для оптимізації персональних фінансових витрат» за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____ д.т.н., проф. В.П.Майданюк
"____"_____2019 р.
Виконав:
_____студент гр.1ПІ-18м О.О. Цимбалюк
"____"_____2019 р.

Вінниця – 2019 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Методи використання BigData для оптимізації персональних фінансових витрат».

Галузь застосування – персональні фінансові помічники.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є розширення простору пошуку пропрозицій при плануванні персональних витрат за рахунок використання BiData.

Призначення роботи – розробка методів і засобів обробки великих об’ємів даних для вибору та відстеження великої кількості товарів.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Horton's I. Beginning JavaSctipt/I. Horton's. –Amsterdam: Wrox, 2015. – 988 p.
2. Stellman A. Head First NodeJS / A. Stellman. – Boston: O'Reilly Media, 2014. – 784 p
3. Бази знань. Інтелектуальні інформаційні системи [Електронний ресурс]. – Режим доступу: https://studfiles.net/preview/5474324/
4. Nodejs Documentation [Електронний ресурс]. Режим доступу до ресурсу: https://nodejs.org/en.

4. Технічні вимоги

Розробити серверний додаток, що сканував би відомі інтернет-магазини і агрегував дані використовуючи програмні парадигми BigData. Вхідні дані – звіти для кожного з сайтів. Вихідні дані – аггреговані пропозиції, надіслані повідомлення користувачам, для яких товар є відстежуваним.

Розробити мобільний додаток на платформі iOS, що дозволить користувачу у зручній та інтуітивній формі взаємодіяти з агрегованими даними, переглядати історію ціни для кожного з товарів, додавати у список для вістеження, та ділитися посиланням на магазин.

5. Конструктивні вимоги.

Зовнішній вигляд додатку повинен бути приємним та естетичним, анімації плавними, робота додатку має корректно починатись навіть після блокування телефону.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Обґрунтування вибору методу розробки та постановка задачі дослідження	
2	Аналіз програмних засобів та рішень	
3	Розробка клієнтської частини додатку	
4	Розробка серверної частини додатку	
5	Економічна частина	

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

Додаток Б Текст серверних модулів

wishListController.js

```
const WishList = require('../models/wishListModel');

exports.add_new_wishlist = function(req, res) {

  var wishListId = req.params.wishListId;
  var sem3_id = req.body.sem3_id;
  var name = req.body.name;
  var description = req.body.description;
  var brand = req.body.brand;
  var price = req.body.price;
  var images_uri = req.body.images_uri;
  var site_details = req.body.site_details
  var image = req.body.image

  var wishlist = new WishList(req.body);

  wishlist.save(function(err, wishlist) {
    console.log(wishlist)

        if (err)
        res.send(err);
        res.json({ message: 'wishlist successfully added' });
      });
};

exports.list_all_wishlist = function(req, res) {
  WishList.find({}, function(err, wishlist) {
    if (err)
      res.send(err);
    res.json(wishlist);
  });
};
```

```
exports.view_a_wishlist = function(req, res) {
  var wishListId = req.params.wishListId;

  console.log(wishListId)

  WishList.findOne({_id: wishListId}, function(err, wishlist) {
    if (err)
      res.send(err);
    res.json(wishlist);
  });
};

exports.delete_a_wishlist = function(req, res) {

  console.log(req.params.wishlistid)

    WishList.remove({_id: req.params.wishlistid}, (err, todo) => {

    if (err) return res.status(500).send(err);
    const response = {
        message: "wishlist successfully deleted",
        id: todo._id
    };
    return res.status(200).send(response);
});
}
```

userController.js

```
                                    };
const jwt = require('jwt-simple');
const User = require('../models/userModel');
const config = require('../config/main');

function tokenForUser(user) {
    const timestamp = new Date().getTime();
```

```
    return jwt.encode({sub: user.id, iat: timestamp}, config.secret);
}

exports.signin = function (req, res, next) {
    res.status(200).json({
      message: "Successful",
      token: tokenForUser(req.user),
      success: 1,
      user: req.user
    })
}

exports.signup = function (req, res, next) {
    const email = req.body.email;
    const password = req.body.password;
    const first_name = req.body.first_name;
    const last_name = req.body.last_name;
    const address = req.body.address;
    const state = req.body.state;
    const city = req.body.city;
    const zip = req.body.zip;
    const usertype = req.body.usertype

    if (!email || !password) {
        return res.status(422).send({error: 'You must provide email and password'});
    }

    User.findOne({ email: email}, function(err, existingUser) {
        if (err) { return next(err); }

        if (existingUser) {
            return res.status(422).send({error: 'Email is in use'});
        }

        const user = new User({
```

```
            email: email,
            password: password,
            first_name: first_name,
            last_name: last_name,
            address: address,
            city: city,
            state: state,
            zip: zip,
            usertype: usertype
        });

        user.save(function(err){
            if (err) { return next(err); }

            res.json({token: tokenForUser(user)});
        });
    });
exports.list_all_users = function(req, res) {
  User.find({}, function(err, user) {
    if (err)
      res.send(err);
    res.json(user);
  });
};

exports.user_info = function(req, res) {
  const email = req.params.email;

  User.findOne( {email: email}, { "password": 0 }, function(err, user) {
    if (err)
      res.send(err);
    res.json(user);
  });
};
```

```
exports.isAuthenticated = function(req, res, next) {

  if (req.user)
   return next();

  res.redirect('/signin');
 };amazonCtrl.js var util = require('util');var OperationHelper =
require('apac').OperationHelper;var _ = require('underscore');

var parseServ = require('../services/parseServ.js');
// var secret = require('../../secrets.js');

module.exports = {
  getProducts: function(req, res) {
  var opHelper = new OperationHelper({
    awsId:     process.env.ID,        awsSecret: process.env.AWSSECRET,
    assocId:   process.env.ASSOCID version:   '2013-08-01'
  }) opHelper.execute('ItemSearch', {
    'SearchIndex': req.query.search,
    'Keywords': req.query.item,
    //'ResponseGroup':
'ItemAttributes,Offers,SalesRank,Reviews,Images,VariationSummary',
    'ResponseGroup': 'Large',
    'ItemPage': req.query.page
    // 'Sort': 'reviewrank_authority'
  }, function(err, results) {
    parseServ.parseResults(results).then(function(response) {
      res.status(200).json(response);
    });
    // res.status(200).send(results);
  })}};
```

ДОДАТОК В Ілюстративний матеріал до захисту
Магістерської кваліфікаційної роботи

Ілюстративний матеріал до захисту магістерської кваліфікаційної роботи

Завідувач кафедри ПЗ, д. т. н., професор _____Романюк О.Н.

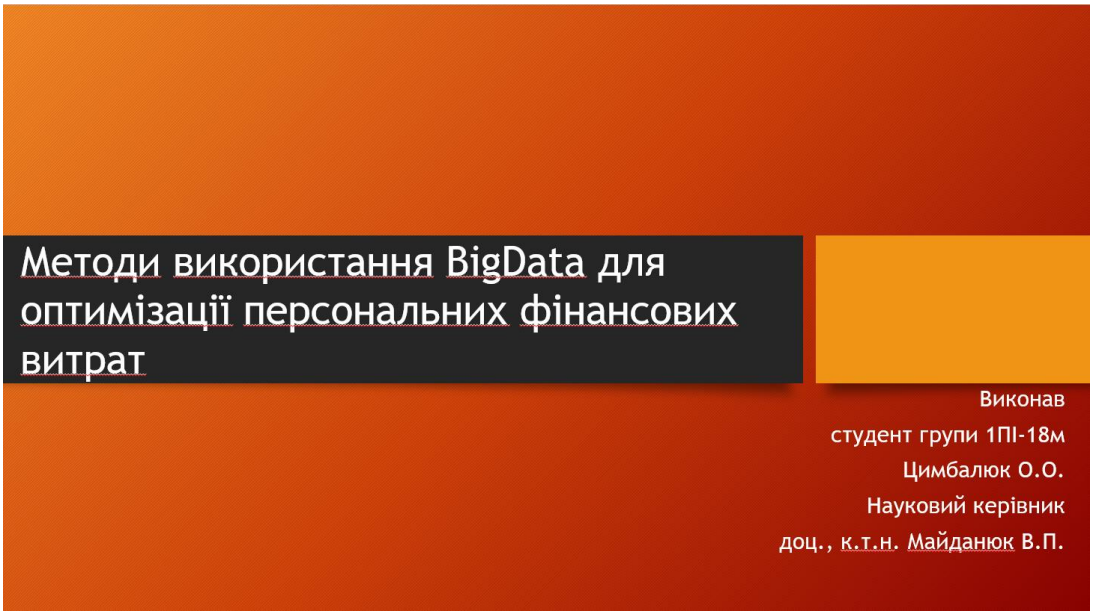
Науковий керівник, к. т. н., професор _____МайданюкВ.П.

Рецензент, к.т.н., доцент _____ Крилик Л.В.

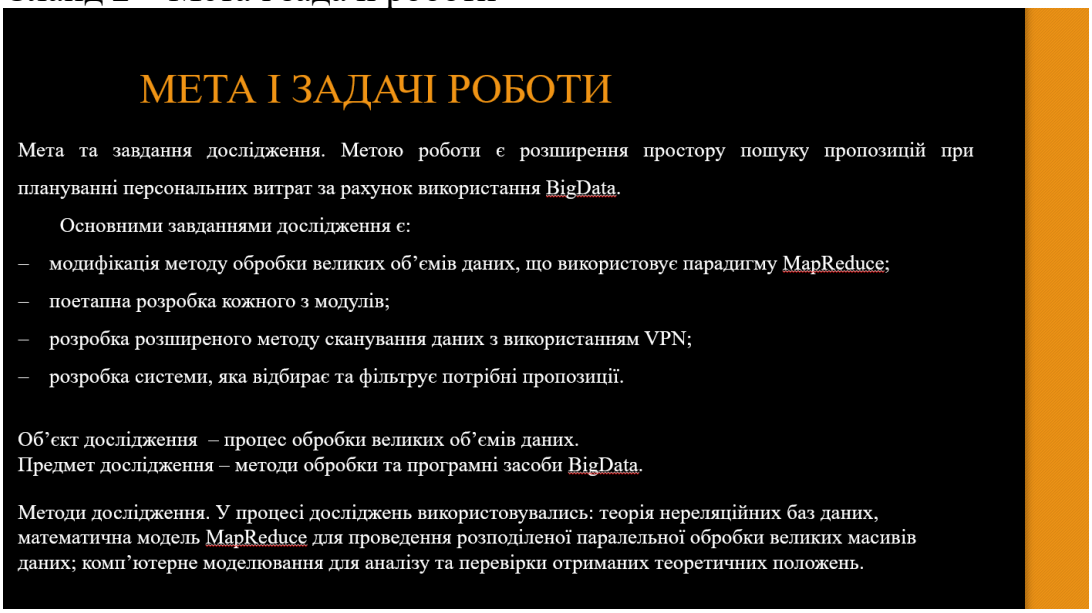
Нормоконтроль, к. т. н., доцент _____Майданюк В.П.

Виконавець, студент групи 1ПІ-18м _____Цимбалюк О.О.

Слайд 1 – Тема і автор МКР



Слайд 2 – Мета і задачі роботи



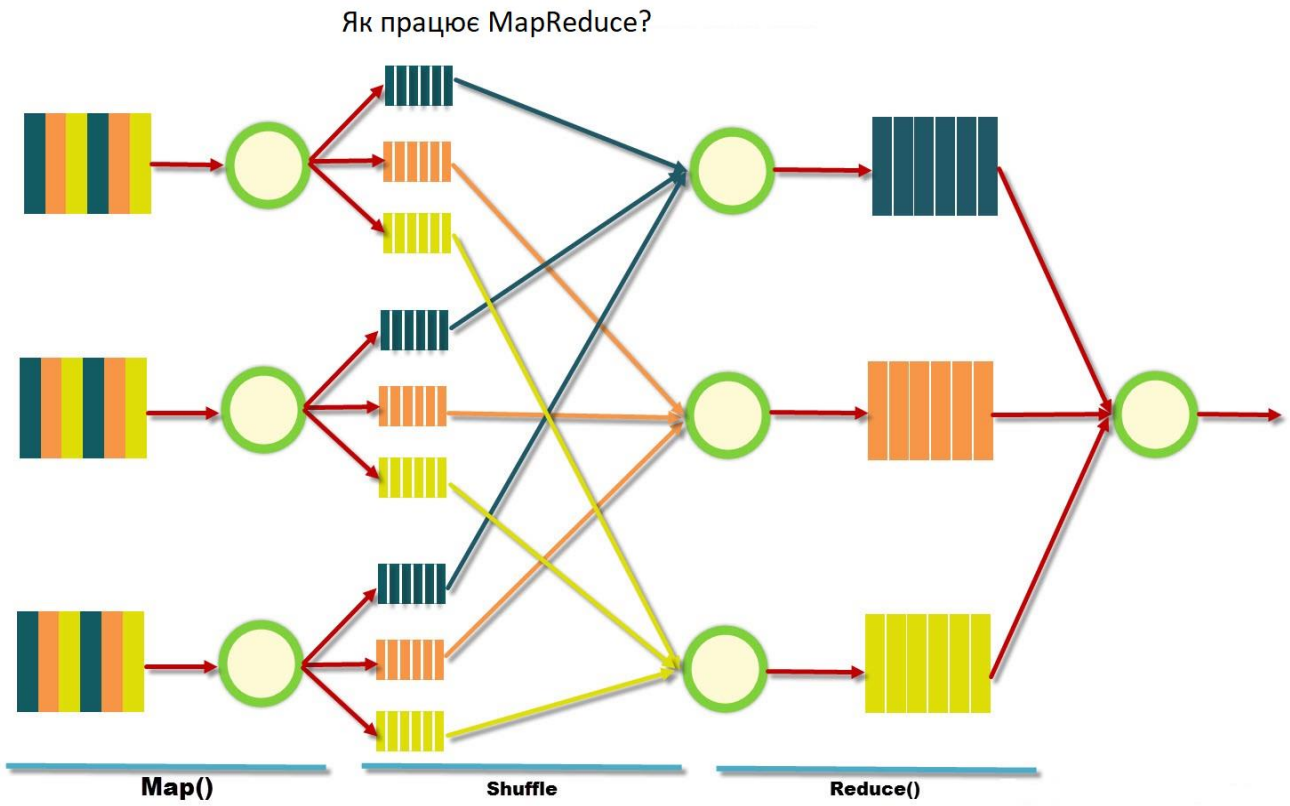
Слайд 3 – Наукова новизна

Наукова новизна

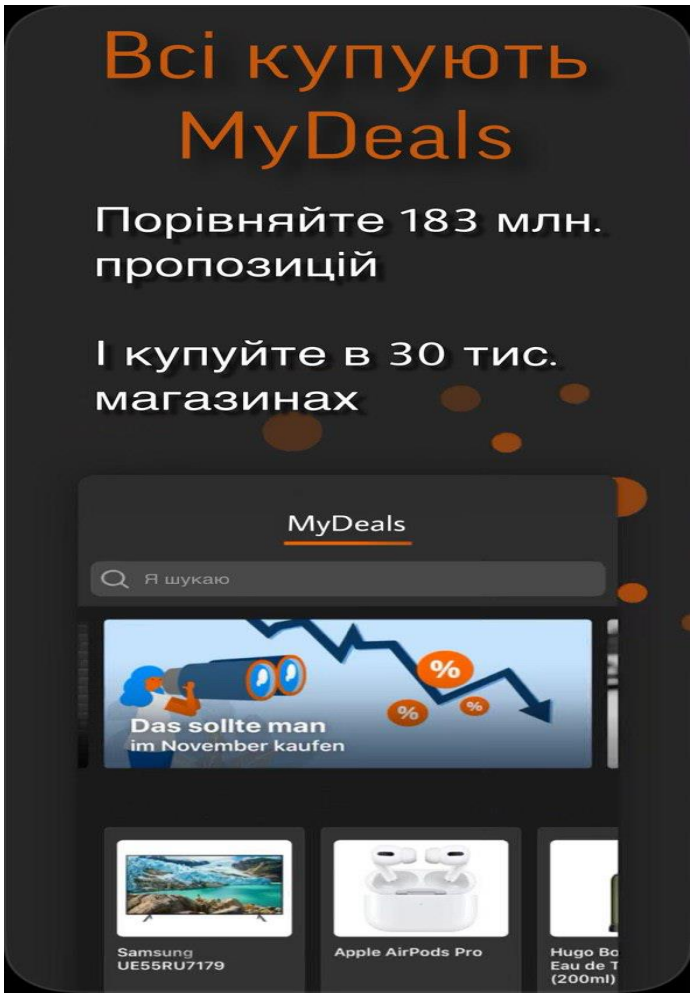
Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод обробки великих об’ємів даних , у якому, на відміну від існуючих використовується програмна модель MapReduce, що дозволило пришвидшити агрегацію вихідних даних.
2. Вперше запропоновано метод відслідковування ціни, особливість якого полягає у розширенні сканування за рахунок використання VPN, що дозволяє відслідковувати артефактні дії продавця.

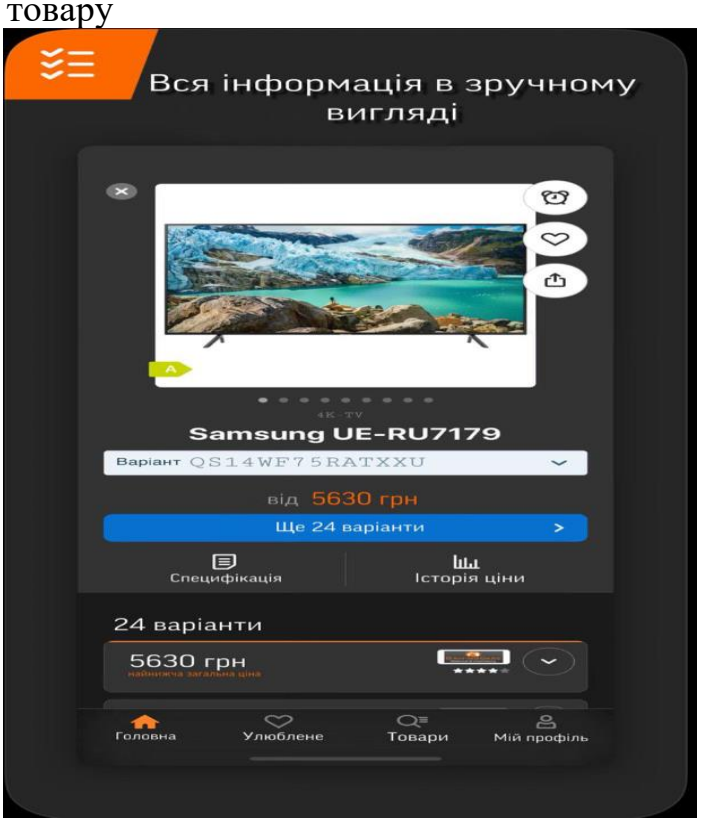
Слайд 4 – Як працює MapReduce



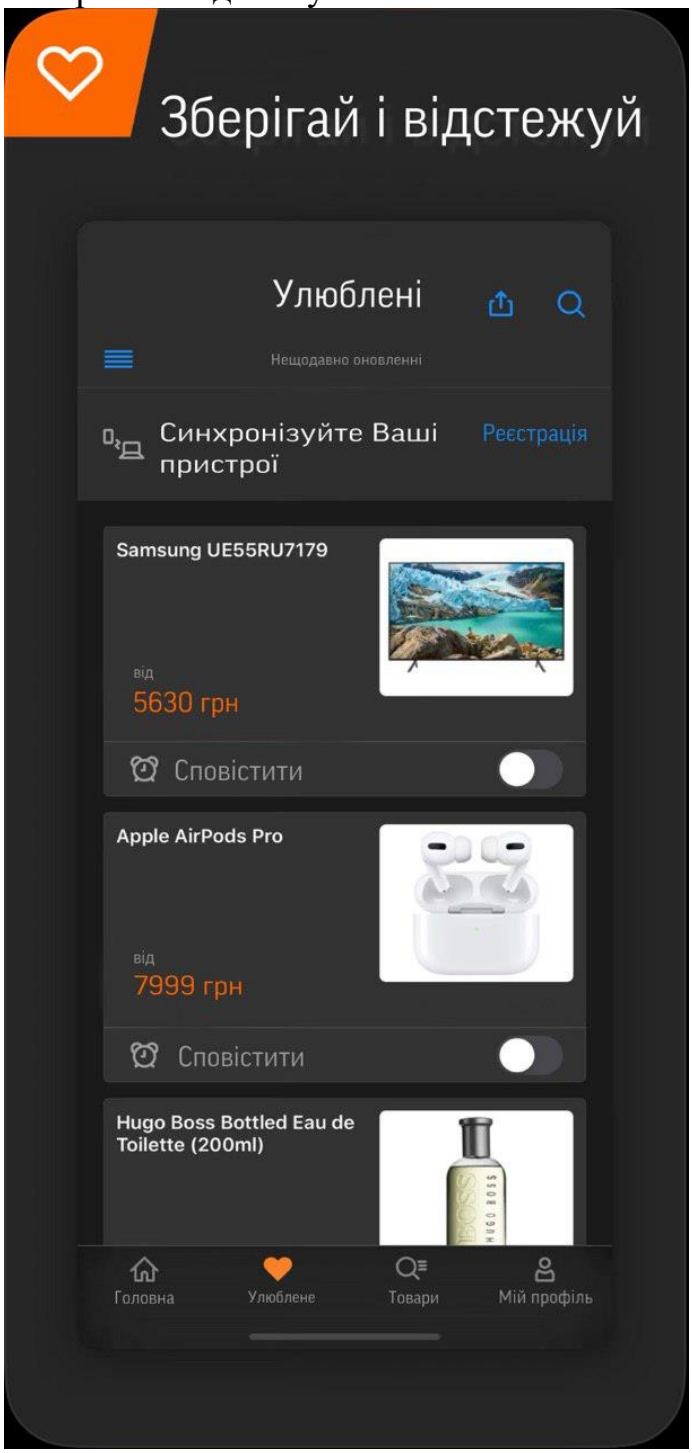
Слайд 5 – Головна сторінка системи



Слайд 6 – Сторінка товару



Слайд 7 – Сторінка «Зберігай і відстежуй»



Слайд 8 - Сторінка «Історія Ціни»