МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ЗУРІЛОВ ІВАН МИКОЛАЙОВИЧ

УДК 004.51

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ АНАЛІЗУ ЦІН В ІНТЕРНЕТ-МАГАЗИНАХ

Автореферат кваліфікаційної роботи на здобуття ступеня вищої освіти «Бакалавр»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня кваліфікація «Бакалавр з інженерії програмного забезпечення»

Миколаїв – 2021

Кваліфікаційною роботою є рукопис.

Робота виконана в Чорноморському національному університеті імені Петра Могили Міністерства освіти і науки України на кафедрі інженерії програмного забезпечення.

Керівник: доктор технічних наук, професор
Фісун Микола Тихонович,
Чорноморський національний університет імені Петра Могили, доцент кафедри інженерії програмного забезпечення

Рецензент: кандидат педагогічних наук, доцент
Болюбаш Надія Миколаївна,
Чорноморський національний університет імені Петра Могили, доцент кафедри інтелектуальних інформаційних систем

Захист відбудеться «24» червня 2021 р. о 9:00 год. на засіданні екзаменаційної комісії (ауд. 2-309) у Чорноморському національному університеті імені Петра Могили за адресою: вул. 68 Десантників, 10, Миколаїв, 54003.

З кваліфікаційною роботою можна ознайомитися в бібліотеці Чорноморського національного університету імені Петра Могили за адресою: вул. 68 Десантників, 10, Миколаїв, 54003.

Автореферат представлений «18» червня 2021 р.

Секретар
екзаменаційної комісії І. О. Кандиба

ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Актуальність роботи зумовлена швидкими темпами збільшення кількості товарів на ринку, через що покупці не встигають слідкувати за усіма новинками та не можуть швидко знайти необхідний товар.

Завданням даної дипломної роботи було створити застосунок для порівняння цін на товари в інтернеті у форматі API із застосуванням власної бази даних. Актуальність зумовлена великим темпом зросту створення та поширення інформації і необхідністю її фільтрації для успішного пошуку необхідного товару.

Метою даної роботи є створення застосунку для порівняння цін на товари в інтернет-магазинах задля економії часу користувача шляхом виведення даних про обрані товари із різних магазинів. Об’єктом роботи виступають збір та виведення даних про ціни на товари у інтернет-магазинах. Предметом дослідження виступають технології та алгоритми збору і обробки даних про ціни на товари у інтернет-магазинах. Із науково-практичної точки зору робота є дослідженням мови програмування C# та її використання для побудови вебсервісу обробки цін на товари в інтернет-магазинах.

КРБ викладена на 55 сторінок, вона містить 4 розділи, 35 ілюстрацій, 0 таблиць, 20 джерел в переліку посилань.

ОСНОВНИЙ ЗМІСТ КРБ

1. Способи отримання даних із відкритих джерел, які використовуються для побудови аналогічних сервісів

Веб-вишкрібання (web-scraping)

Це дуже популярний спосіб отримання даних порівняння, хоча не завжди законодавчо затверджений. Йдеться про веб-сканери чи павуки, які періодично відвідують заздалегідь вибрані сайти для збору необхідної інформації. Він діє так, ніби вишкрібає дані з веб-платформи (звідси і назва – веб-вишкрібання).

Метод досить простий: спеціально навчений алгоритм, який також є автоматизованим кодом, переходить на головну сторінку Інтернет-ресурсу і починає відвідувати кожне внутрішнє посилання, аналізувати HTML-документ, шукати дані та перетворювати його у вказаний формат.

Плюси:

1. отримати дані дуже просто; вам навіть не потрібен дозвіл власника ресурсу;
2. є шанс підняти базу даних до рівня конкурентів за короткий проміжок часу;
3. рішення досить просте у реалізації.

Мінуси:

1. не можна використовувати платну модель монетизації;
2. деякі ресурси мають захист від копіювання даних;
3. вихідний код може містити посилання на однакові зображення різного розміру (скажімо, попередній перегляд);
4. сайт може визначити країну, в якій знаходиться ваш сервер, і надати вам інформацію вашою мовою (не англійською);
5. метод вишкрібання схвалений не всіма.

Як можна побачити, метод має більше мінусів, ніж плюсів та не є оптимальним.

API

Серед засобів цифрового порівняння широко використовується інтеграція API. Метод передбачає отримання інформації безпосередньо від самих

постачальників за допомогою API. Отримані дані автоматично вводяться у вашу базу даних.

Плюси:

1. можна отримати інформацію у правильному форматі та якісно;
2. метод легальний, затверджений усіма учасниками процесу;
3. можна монетизувати свій ресурс за рахунок комісій;
4. можна отримувати інформацію в режимі реального часу.

Мінуси:

1. метод досить ресурсомісткий;
2. є сайти, які не надають своїх API.

Модель особливо хороша, якщо більшість сайтів з ціноутворенням використовують API і готові поділитися ним з вами. І, зазвичай, вони із задоволенням надають доступ до своїх даних: врешті-решт це вигідно і їм, адже це – безкоштовна реклама їх ресурсу.

Додавання даних вручну

Звичайно, це не найзручніший спосіб створити веб-сайт для порівняння цін. Хоча, я все одно його опишу. Отже, якщо продавці не мають API, але хочуть взяти участь у вашій програмі, вони можуть надати вам свою інформацію у вигляді файлів XML та CSV. Або можна дозволити їм вводити дані вручну.

Плюси:

1. метод легко реалізувати;
2. можна отримати необхідні дані повністю легально, з дозволу постачальників.

Мінуси:

1. можуть знадобитися додаткові витрати: доведеться найняти співробітників для введення даних у вашу базу даних або

створити спеціальний інтерфейс, щоб виробники могли це робити самі;

2. метод не дуже зручний, враховуючи ручний спосіб введення даних.

Така модель підходить як доповнення до описаного вище варіанту API, але також не є зручною реалізацією.

2. Властивості REST API

Що таке REST

REST – це абревіатура від Representational State Transfer []. Це архітектурний стиль для розподілених гіпермедійних систем і вперше був представлений Роєм Філдінгом у 2000 році у своїй знаменитій дисертації.

Як і будь-який інший архітектурний стиль, у REST також є свої 6 основних обмежень, які необхідно виконати, якщо інтерфейс потрібно називати RESTful. Ці принципи наведені нижче.

Клієнт-сервер – відокремлюючи проблеми користувальницького інтерфейсу від проблем зберігання даних покращується переносимість користувальницького інтерфейсу на декількох платформах та покращуємо масштабованість, спрощуючи компоненти сервера.

Відсутність стану – кожен запит від клієнта до сервера повинен містити всю інформацію, необхідну для розуміння запиту, і не може використовувати будь-який збережений контекст на сервері. Тому стан сеансу повністю тримається на клієнті.

Кешування – обмеження кешу вимагають, щоб дані у відповіді на запит неявно або явно позначалися як кешовані чи некешовані. Якщо відповідь можна кешувати, тоді кеш-пам’ять клієнта отримує право повторно використовувати ці дані відповідей для подальших, еквівалентних запитів.

Уніфікований інтерфейс – застосовуючи принцип загальності до інтерфейсу компонента, спрощується загальна архітектура системи та покращується видимість взаємодій. Для того, щоб отримати єдиний інтерфейс, для керування поведінкою компонентів необхідні численні архітектурні обмеження. REST визначається чотирма обмеженнями інтерфейсу:

ідентифікація ресурсів; маніпулювання ресурсами через представництва; самоописові повідомлення; і, гіпермедіа як двигун стану програми.

Багатошарова система - Стиль багатошарової системи дозволяє архітектурі складатися з ієрархічних шарів, обмежуючи поведінку компонентів, так що кожен компонент не може "бачити" далі безпосереднього шару, з яким вони взаємодіють.

Код за вимогою (необов’язково) – REST дозволяє розширити функціональність клієнта, завантажуючи та виконуючи код у формі аплетів або скриптів. Це спрощує клієнтів, зменшуючи кількість функцій, необхідних для попередньої реалізації.

Ресурс

Ключовим абстрагуванням інформації в REST є ресурс. Будь-яка інформація, яку можна назвати, може бути ресурсом: документ або зображення, тимчасова служба, колекція інших ресурсів, невіртуальний об’єкт (наприклад, особа) тощо. REST використовує ідентифікатор ресурсу для ідентифікації конкретного ресурсу, що бере участь у взаємодії між компонентами.

Стан ресурсу на будь-якій конкретній мітці часу називається поданням ресурсу. Представлення складається з даних, метаданих, що описують дані, та гіпермедіа-посилань, які можуть допомогти клієнтам перейти до наступного бажаного стану.

Формат даних подання відомий як тип носія. Тип носія визначає специфікацію, яка визначає спосіб обробки подання. По-справжньому RESTful API виглядає як гіпертекст. Кожна адресована одиниця інформації містить адресу або явно (наприклад, атрибути посилання та ідентифікатора), або неявно (наприклад, отримана з визначення типу медіа та структури представлення).

Ресурсні методи

Інша важлива річ, пов’язана з REST, - це методи ресурсів, які слід використовувати для здійснення бажаного переходу. Велика кількість людей помилково пов'язують методи ресурсів із методами HTTP GET / PUT / POST / DELETE.

Рой Філдінг ніколи не згадував жодної рекомендації щодо того, який метод застосовувати в якому стані. Все, що він наголошує, це те, що це повинен бути єдиний інтерфейс. Якщо ви вирішите, що для оновлення ресурсу буде використовуватися HTTP POST навіть якщо більшість людей рекомендували б HTTP PUT – це нормально, і інтерфейс програми буде RESTful.

В ідеалі все, що потрібно для зміни стану ресурсу, має бути частиною відповіді API для цього ресурсу, включаючи методи та те, в якому стані вони залишать подання.

1. Технологічна основа сервісу

MongoDB – це міжплатформна, документоорієнтована база даних, яка забезпечує високу продуктивність і доступність та легку масштабованість. MongoDB працює беручи за фундамент концепції колекції та документа.

База даних - це фізичний контейнер для колекцій. Кожна база даних отримує власний набір файлів у файловій системі. Один сервер MongoDB зазвичай має кілька баз даних.

Колекція - це група документів MongoDB. Колекція існує в межах однієї бази даних. Колекції не застосовують схему. Документи в колекції можуть мати різні поля. Як правило, усі документи у колекції мають подібне або пов'язане призначення.

Документ - це набір пар ключ-значення. Документи мають динамічну схему. Динамічна схема означає, що документи в одній колекції не повинні мати однаковий набір полів або структури, а загальні поля в документах колекції можуть містити різні типи даних. Таким чином, дана база даних є оптимальним рішенням для збереження даних різних типів у різних пропорціях, що будуть поступати до неї на постійній основі.

Рисунок 1 – Логотип MongoDB

Мова програмування C# - це сучасна, об'єктно-орієнтована та строго-типізована мова програмування. C# дозволяє розробникам створювати багато типів безпечних та надійних застосунків, що працюють в екосистемі .NET. C# сягає корінням у сімейство мов C і буде одразу знайомий програмістам на C, C++, Java та JavaScript.

C# [5] - це об'єктно-орієнтована, компонентно-орієнтована мова програмування. C# забезпечує мовні конструкції для прямої підтримки цих концепцій, роблячи C# природною мовою, на якій можна створювати та використовувати програмні компоненти. З моменту свого зародження C# додав функції для підтримки нових навантажень та нових практик проєктування програмного забезпечення.

Рисунок 2 – Логотип мови C#

Кілька функцій C# допомагають створювати надійні та довговічні програми. Збір сміття автоматично відновлює пам’ять, зайняту недосяжними невикористаними об’єктами. Типи, що допускають відхилення, захищають від змінних, які не посилаються на виділені об'єкти. Обробка винятків забезпечує структурований та розширюваний підхід до виявлення та відновлення помилок. Лямбда-вирази підтримують методи функціонального програмування. Синтаксис інтегрованого мовного запиту (LINQ) створює загальний шаблон роботи з даними з будь-якого джерела. Мовна підтримка асинхронних операцій забезпечує синтаксис для побудови розподілених систем. C# має уніфіковану систему типів. Усі типи C#, включаючи примітивні типи, такі як int та double, успадковуються від одного кореневого типу об'єкта. Усі типи мають спільний набір загальних операцій. Цінності будь-якого типу можна зберігати, транспортувати та використовувати послідовно. Крім того, C# підтримує як визначені користувачем посилальні типи, так і значущі типи. C# дозволяє динамічно розподіляти об'єкти та лінійно зберігати легкі конструкції. C#

підтримує загальні методи та типи, які забезпечують підвищену безпеку та продуктивність. C# надає ітератори, які дозволяють реалізаторам класів колекцій визначати власну поведінку для клієнтського коду. C# робить акцент на встановленні версій для забезпечення сумісного розвитку бібліотек та програм із часом.

Програми C# працюють на .NET - віртуальній системі виконання, яка називається середовищем виконання загальної мови (CLR) та набором бібліотек класів. CLR - це впровадження корпорацією Майкрософт загальномовної інфраструктури (CLI), міжнародного стандарту. CLI - це основа для створення середовищ виконання та розробки, в яких мова та бібліотеки працюють безперебійно.

Вихідний код, написаний на C#, компілюється на проміжну мову (IL), що відповідає специфікації CLI. Код IL та ресурси, такі як растрові зображення та рядки, зберігаються у збірці, як правило, з розширенням .dll. Збірка містить маніфест, який надає інформацію про типи, версію та культуру збірки.

Коли програма C# виконується, збірка завантажується в CLR. CLR виконує компіляцію Just-In-Time (JIT) для перетворення IL-коду в машинні інструкції. CLR надає інші послуги, що стосуються автоматичного збору сміття, обробки винятків та управління ресурсами. Код, який виконується CLR, іноді називають "керованим кодом", на відміну від "некерованого коду", який компілюється на рідну машинну мову, націлену на певну платформу.

Мовна сумісність - ключова особливість .NET. Код IL, створений компілятором C#, відповідає загальній специфікації типу (CTS). Код IL, сформований із C#, може взаємодіяти з кодом, створеним із версій .NET F#, Visual Basic, C++ або будь-якої з більш ніж 20 інших мов, сумісних із CTS.

На додаток до служб виконання, .NET також включає великі бібліотеки. Ці бібліотеки підтримують багато різних навантажень. Вони організовані в простори імен, що забезпечують широкий спектр корисних функцій для всього, від введення та виведення файлів до маніпуляцій рядками до аналізу XML, до фреймворків веб-додатків до елементів керування Windows Forms. Типовий

додаток C# широко використовує бібліотеку класів .NET для обробки загальних службових робіт.

С # використовується для:

1. Мобільні застосунки
2. Настільні програми
3. Веб- застосунки
4. Веб-сервіси
5. Веб-сайти
6. Ігри
7. VR
8. Застосунки, що взаємодіють із базами даних

З інформації про мову програмування та її екосистему можна зробити висновок, що мова C# цілком задовольняє вимоги до інструменту створення сервісу порівняння цін, адже дозволяє легко взаємодіяти із мережевими інструментами та базами даних.

2. Розробка сервісу

Для створення застосунку було використано усі попередньо описані інструменти та діаграми. Повний код буде надано у додатку. Окремо варто звернути увагу на структуру проєкту, а саме – директорії Controllers, Models та Services. У них розташовані файли класів, що виконують логіку підготовки до збору даних із колекції, схеми об’єкту колекції та збору усіх елементів колекції відповідно.

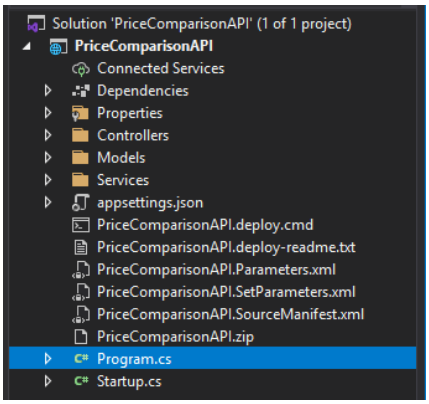

Рисунок 3 – структура проєкту

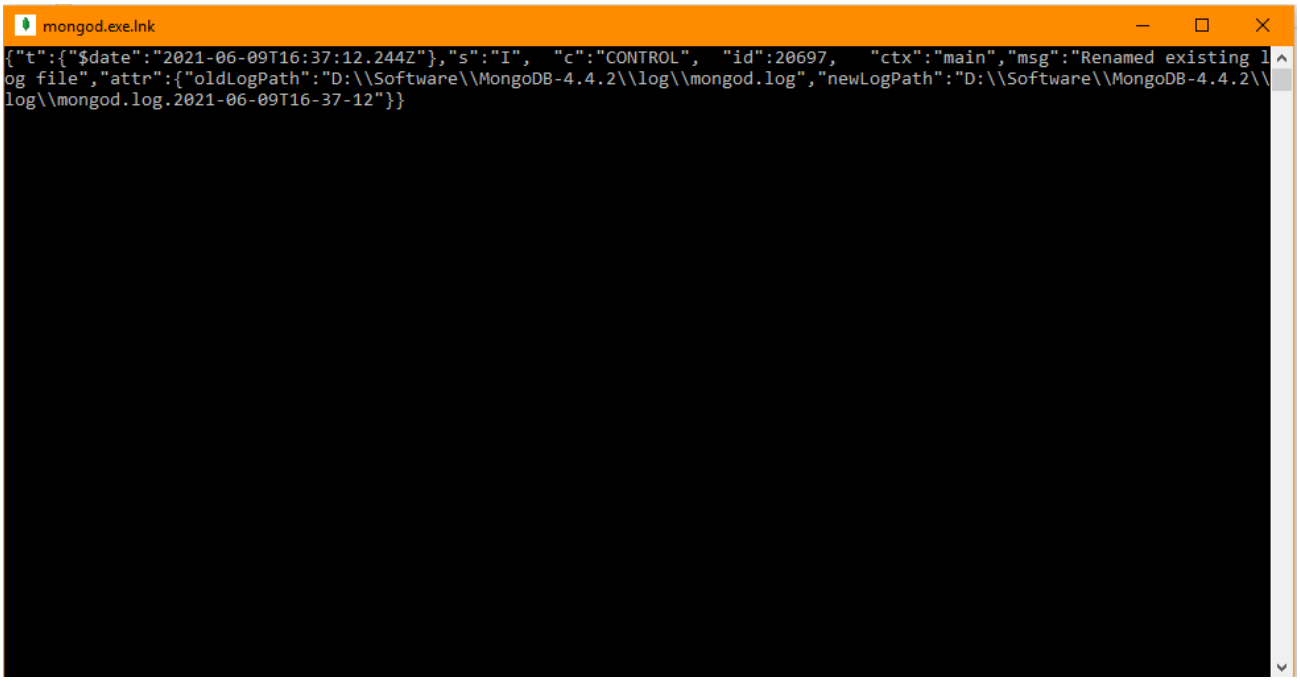

Рисунок 4 – Запуск БД

Базу даних було заповнено однією прбною колекцією із набором документів.

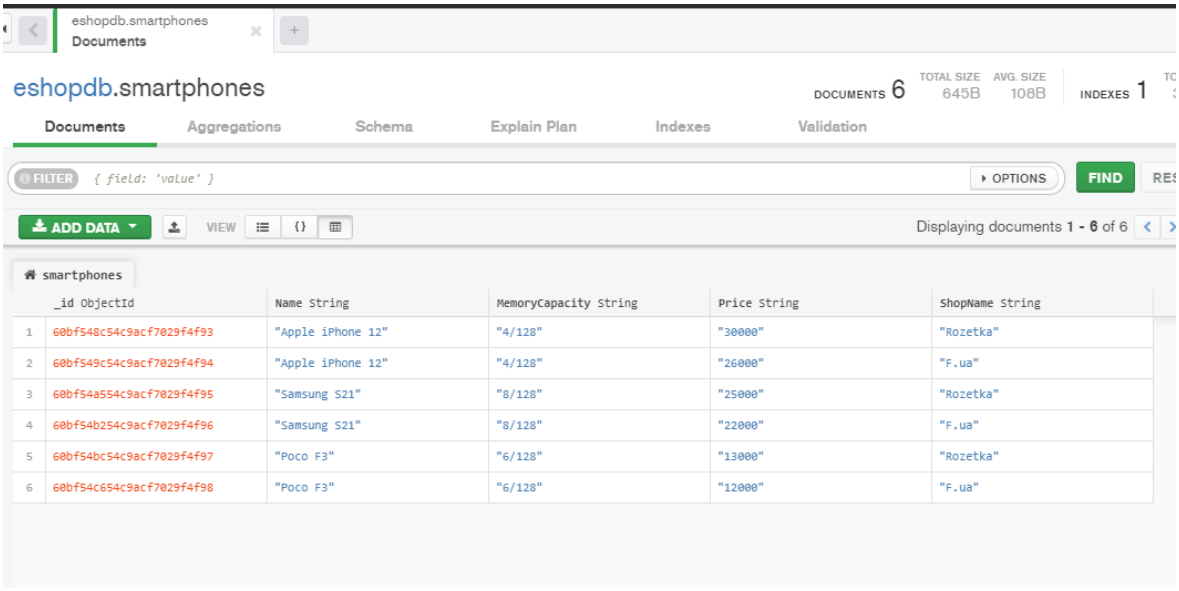

Рисунок 5 – Заповнена БД

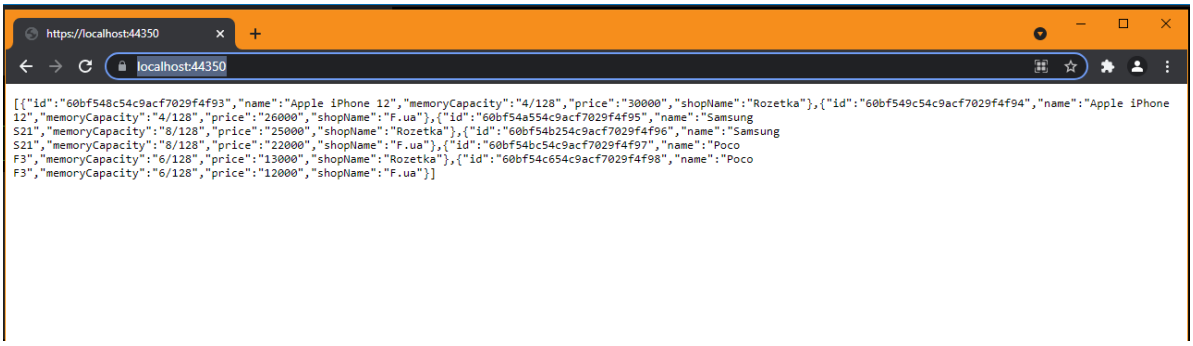

Рисунок 6 – Запуск застосунку

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було засвоєно навички розробки API мовою C# із використанням БД MongoDB та хмарного середовища MS Azure.

Було досліджено механізми роботи сервісів порівняння цін в цілому та на ринку України. Було досліджено переваги та недоліки підходів, які використовують такі сервіси разом із інтернет-магазинами, а саме:

1. надання інформації про товари напряму сервісам порівняння цін;
2. відсутність взаємодії із БД магазину шляхом використання API;
3. встановлені системи антипарсингу вебсторінок, що унеможливлюють зчитування напряму зі сторінки.

Були виявлені і недоліки підходів до створення сервісів порівняння цін:

1. метод парсингу виявився занадто громіздким;
2. метод підключення API виявився складним та неможливим у реалізації в умовах ринку України та українських інтернет-магазинів;
3. метод додавання даних вручну виявився надто довгим та з великою кількістю зайвих дій, до того ж магазини часто неправильно заповнюють БД перед відправкою.

В результаті було створено API, що взаємодіє із власною БД, у яку можна завантажувати дані, та дозволяє скористатися обмеженим набором даних з БД шляхом запитів у веббраузері. Таким чином створено потенційно новий застосунок, що обходить обмеження аналогів – дає доступ до даних БД, але не розкриває їх повністю.

Сервіс можна масштабувати, додаючи нові реалізації методів взаємодії із БД та підключивши API до клієнтського застосунку, використовуючи посилання, створене за допомогою сервісу Azure.