МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Багатофункціональний сервіс для підбору товарів та аналізу інтернет магазинів»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Бабій М.С.**

Студента групи ІН – 64 – 8 **Сас І.С.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 г.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-64-8 спеціальності "Програмування інтелектуальних інформаційних систем" денної форми навчання Сас Ігора Сергійовича.

Тема: "**Багатофункціональний сервіс для підбору товарів та аналізу інтернет магазинів**"

Затверджена наказом по СумДУ

№_____ от _____2020 г.

Зміст пояснювальної записки: 1) інформаційний огляд; 2) вибір методу рішення; 3) програмна реалізація інформаційної системи;

Дата видачі завдання "_____"___2020 г.

Керівник випускної роботи _____ Бабій М.С.

Завдання приняв до виконання _____ Сас І.С.

РЕФЕРАТ

Записка: 42 стор., 12 рис., 1 додаток, 9 джерел.

Об’єкт дослідження — продажі товарів в інтернет магазинах.

Мета роботи — розробка сервісу для допомоги с вибором вигідних пропозицій на товари в різних інтернет магазинах та аналізу чесності зміни ціни в цих магазинах.

Методи дослідження — метод аналітичних випробувань.

Результати — розроблено алгоритм та програмне забезпечення системи аналізу товарів. При цьому задача оцінки честності цін на товари та оцінки роботи магазинів представлених в каталозі, розв’язана в рамках багатофункціональної інтелектуальної інформаційної системи. Розроблена інформаційна система реалізована у формі сайту каталогу, створеного за допомогою програмного середовища Visual Studio та мови програмування C#.

КАТАЛОГ ТОВАРІВ, ПІДБІР ТОВАРІВ, АНАЛІЗ ІНТЕРНЕТ МАГАЗИНІВ, СТАТИСТИКА ЦІН, ІНТЕРНЕТ МАГАЗИНИ.

ЗМІСТ

ВСТУП

З появою глобальної мережі інтернет життя багатьох людей дуже змінилося, з’явилася безліч нових можливостей. Через мережу інтернет можна забезпечити обмін інформацією між всіма віддаленими комп’ютерами. Після появи цього винаходу велика кількість користувачів змогли дуже швидко отримувати та обмінюватися інформацією з найбільш компетентних джерел.

Майже кожного дня з’являється все більше нових можливостей у користувачів та розробників. Разом з цим також розвиваються й інтернет магазини, з’являються нові, та з кожним днем все складніше обрати магазин з найкращими пропозиціями.

Тому я вирішив створити сервіс на якому буде каталог товарів та який буде шукати потрібний товар в різних магазинах, аналізувати зміни цін, перевіряти наявність товарів, наявність знижок на товари, порівнювати та пропонувати споживачеві магазин з найкращими умовами.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Вступна частина

На сьогоднішній день інтернет це настільки могутня річ, яка дуже вражає, як вона перестала вражати людей. Якими словосполученнями тільки не називають мережу інтернет: і всесвітня мережа, і глобальна мережа, та всесвітня павутина. Мільйони людей можуть дуже швидко здійснювати обмін інформацією, пошук інформації, спілкуватися неважливо від того, де вони знаходяться на земній кулі.

Інтернет це мережа мереж, що забезпечує інфраструктуру для взаємодії і спільного використання інформації. Він забезпечує ряд сервісів, такі як електронна пошта, передача файлів, підключення в режимі віддаленого терміналу, інтерактивні конференції, групи новин, і W3.

World Wide Web це вся інформація яка доступна в мережі інтернет. W3 з'явився як мережевий інформаційний проект в CERN, європейській фізичної лабораторії. W3 складається з програм, набору протоколів і угод, використовуваних для отримання доступу до інформації та її пошуку в Інтернеті. За допомогою використання гіпертекстових і мультимедійних технологій W3 дозволяє будь-якому користувачеві легко додавати нову інформацію, переглядати її та шукати. Веб-клієнти, також відомі як веб-браузери, забезпечують користувальницький інтерфейс для навігації серед інформації за допомогою методу вказівки і клацання мишкою. Веб-сервера надають браузерам HTML і інші засоби для отримання інформації за допомогою протоколу HTTP. Браузери інтерпретують, форматують і відображають документи користувачам. Кінцевим результатом є мультимедійна вистава Інтернету. На даний момент чимало організацій підтримують зовнішні веб-сайти, які описують та представляють їхні компанії а також іх сервіси. За для безпеки ці сервера звичайно знаходяться за захистом брандмауера цієї компанії.

Електронна торгівля це одна із форм доставки товарів, при якій замовлення і вибір товарів здійснюється за допомогою мережі інтернет, та розрахунки між продавцем та покупцем можуть здійснюються за допомогою електронних грошей або коштів платежу. В ролі покупців товарів можуть бути як приватні особи так і компанії.

Мережа інтернет надала можливість відкрити електронні продажі для фірм якого завгодно розміру. Раніше до цього створення електронного обміну інформацією вимагала чималих вкладень в комунікацію між відділами та була під силу лише дуже великим компаніям, та завдяки використанню мережі інтернет на сьогоднішній день в ряди "електронних комерсантів" вступили також і невеличкі фірми. Електронний каталог в інтернеті забезпечує багатьом компаніям можливість привертати покупців з усієї планети. Такий онлайн фірма створює новий ринок для продажів товарів - "віртуальний", та майже не потребує великих грошових вкладень. Якщо якісь послуги компанії (наприклад, такі як програмні продукти) можуть бути продані через інтернет, то увесь процес продажу товару (включаючи також й оплату товарів) може відбуватися в тому числі й через мережу інтернет.

Запровадження торгівлі товарами в мережі інтернет робить традиційну торгівлю менш вимогливою, так як торгівля в інтернеті має можливість маніпулювати цифровою інформацією, яка працює для створення простіших умов для спільної роботи людей. Кожен цифровий магазин має унікальну адресу в мережі інтернет, як і будь-який віртуальний сервер, та складається в дуже великої кількості веб сторінок з описами товарів.

Сайт-каталог це веб-ресурс, в якому є необхідні дані про компанію, а також представлені безпосередньо товари.

На сайті може бути як п'ятдесят, так і півтисячі найменувань товарів - обмежень немає. Важливо на стартовому етапі враховувати подальше зручність роботи не тільки для співробітників, які будуть наповнювати каталог, але і, зрозуміло, для кожного клієнта, хто зайде за посиланням.

За своєю структурою сайт-каталог в якійсь мірі схожий з інтернет-магазином. Але в порівнянні з більш масштабним проектом, каталог простіший в обслуговуванні, адже він не передбачає обробку інтернет-замовлень. Також в ньому немає функції обробки платежу, адже він надає право вибору. На сайті-каталозі клієнт самостійно вивчає і вибирає потрібну товарної позиції, ціну, а потім зв'язується з менеджером для уточнення нюансів по оплаті і доставці. Купівля відбувається виключно після спілкування з менеджером. Це досить зручно для певних товарів, в підборі яких варто враховувати безліч нюансів.

Сайт з каталогом зобов'язаний надавати клієнтам повну інформацію про товари або послуги компанії. Дуже важливо, щоб цей веб-ресурс був інформативним, а його дизайн цікавим, пошук - зрозумілим і швидким.

Компанія, яка хоче бути успішною на ринку, повинна постійно аналізувати і вдосконалювати свої продукти і пропонувати додаткові вигоди - удосконалювати «додатковий продукт», адже при інших рівних (сутності і фактичному продукті) саме додаткові вигоди є аргументом на користь покупки того чи іншого продукту одного типу. В рамках наведеного прикладу це дуже показово: як думаєте, в якому магазині купить споживач телевізор - в магазині з гнучкою системою знижок, можливістю доставки додому і вибору забарвлення, або в магазині, де всього цього не буде?

Безумовно, варто також пам'ятати і про адекватне ціноутворення - продукти будуть набагато більш затребуваними, якщо додаткові вигоди будуть пропонуватися клієнтам в якості опції. Наприклад, споживач при покупці ноутбука може доплатити певну суму і отримати за це додаткову послугу - нанесення на верхню кришку комп'ютера улюбленого малюнка або логотипу.

При складанні презентацій продукту на основі аналізу дуже важливо пам'ятати про те, що компанія пропонує «сутність продукту» (задоволення потреб і потреб споживача), а не «фактичний продукт» (набір технічних характеристик і властивостей продукту)! Важливо робити акцент на додаткові

вигоди, які отримає споживач в разі здійснення покупки, і на тому, як якісно і відмінно ваш продукт відповідає його запитам.

1.2 Огляд схожих рішень

Перше схоже рішення це E-Katalog (Рисунок 1.1) сучасний варіант універсального інтернет-ресурсу, який дає можливість потенційному покупцеві здійснити порівняння вартості самих різних видів продукції, яка пропонується до придбання в нинішніх онлайн-магазинах. E-Katalog за адресою ek.ua дає можливість ознайомитися з детальним описом і вартістю електроніки, комп'ютерної техніки, побутових варіантів сучасного обладнання, товарами автомобільної тематики, спортивним інвентарем, а також різноманітними варіантами інструменту. Під даними інтернет-ресурсом спочатку варто розуміти спеціальний каталог, а не сайт, де можна придбати якусь продукцію. Іншими словами цей портал служить відмінним помічником користувачеві, якому необхідно вибрати певну модель пристрою або ж потрібно відшукати найбільш прийнятну вартість.

Для того щоб допомогти людині в здійсненні правильного вибору, на сайті присутні спеціальні розділи, які дозволяють здійснити підбір різноманітної продукції з урахуванням певних параметрів, що дозволяє в подальшому максимально швидко порівняти різні моделі пристроїв між собою. У будь-якого виду продукції на сайті є власна сторінка з детальним описом характеристик і технічних особливостей. Остаточно визначитися з підбором тут також додатково допомагає і різна додаткова інформація, яка видається на сторінці продукції, наприклад, в блоці під назвою «Де купити?». Також тут є і такі не менш корисні блоки: відео огляди, пропозиції від магазинів про продаж певного виду продукції, а також відгуки покупців і фотознімки. На будь-якої сторінки продукції з детальним описом вироби присутній список онлайн-магазинів, де вона продається з позначенням посилання на продукцію і відповідно, вартості.

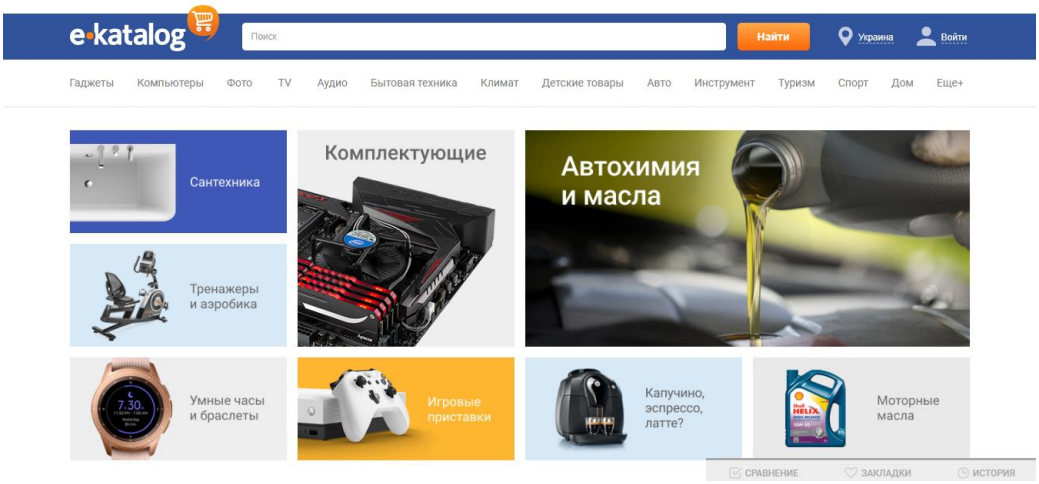


Рисунок 1.1 - Вигляд каталогу ek.ua

Наступним схожим рішенням є price.ua (Рисунок 1.2) - перший офіційний багатофункціональний сайт для порівняння цін на ринку товарів України. Він надає собою зручний механізм пошуку товарів та послуг в різних інтернет-магазинах під час порівняння цін, описів і характеристик.

Для зручності користувачів цей багаторівневий пошук товарів розроблений не тільки за категоріями, а й по виробниках, і за цінами. Тут же можна побачити мої фотографії товарів, ознайомитися з їх описом і оцінити динаміку. Завдяки наявності відгуків покупців, ви можете дізнатися про різні недоліки і переваги, які були виявлені іншими користувачами в ході експлуатації, а також рейтинг фірм, який формується користувачами і дозволяє оцінити надійність того чи іншого продавця. Розробники сайту регулярно оновлюють всі пропозиції Інтернет-магазинів і розширюють каталог споживчих категорій.

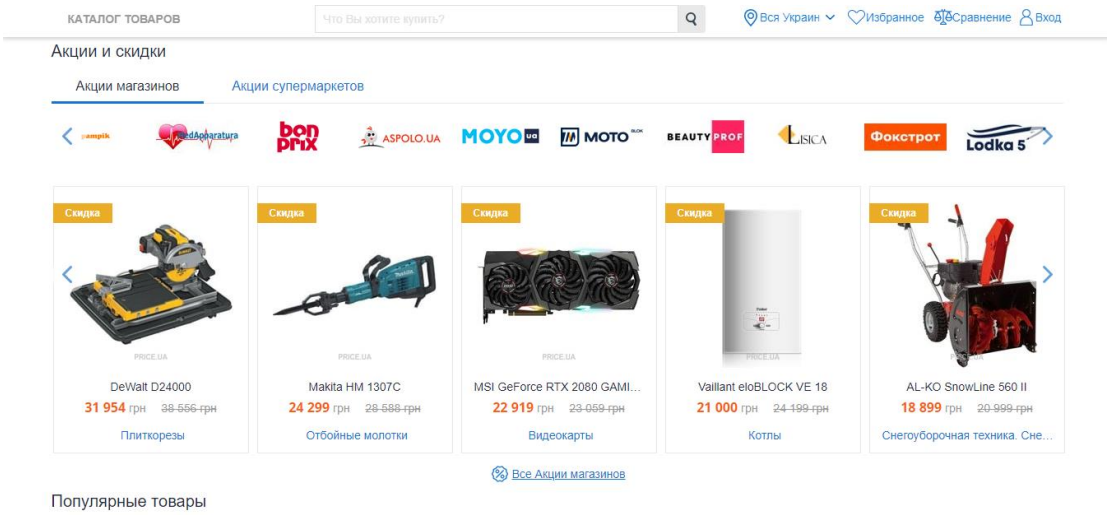


Рисунок 1.2 - Вигляд каталогу price.ua

Також є схожий сервіс Hotline (Рисунок 1.3). Це сервіс, що дозволяє розрахуватися платіжними картами за товари, представлені на hotline.ua інтернет-магазинами, учасниками Checkout. За допомогою Hotline Checkout ви можете оплатити товари з одного або декількох магазинів на сторінках hotline.ua, без переходу на інші сайти. В цьому випадку hotline виступає як "посередник" при оформленні замовлення і проведенні платежу за допомогою платіжної картки.

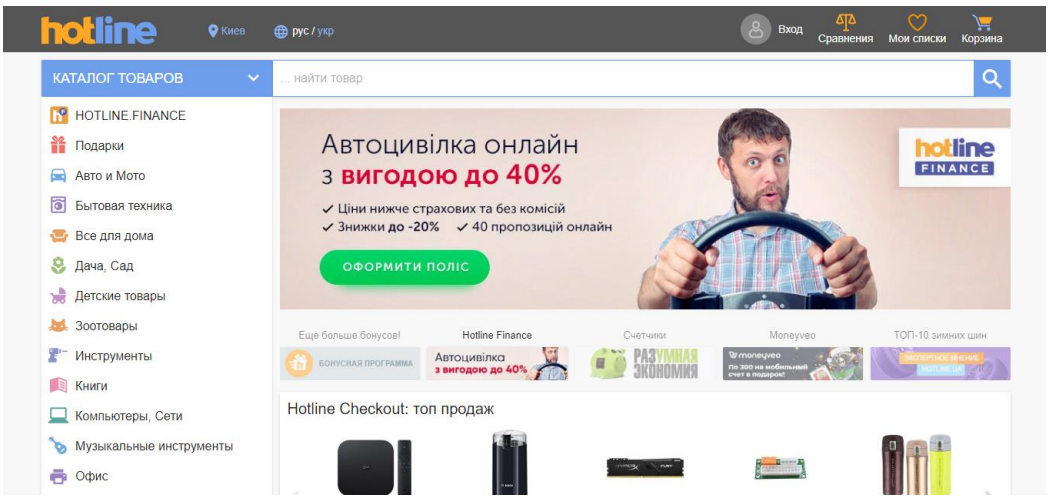


Рисунок 1.3 - Вигляд каталогу hotline.ua

1.3 Постановка задачі

Потрібно спроектувати сервіс в якому буде каталог товарів з різних магазинів, буде більш детальний опис цих товарів, статистика зміни цін на ці

товари в представлених магазинах, опис магазинів які є даному каталозі а також статистика задоволеності покупців товарів конкретним магазином.

Треба розробити веб сайт каталог, зі зручними сторінками на яких будуть представлені товари, розробити сторінки з детальним описом цих товарів та статистикою зміни цін, сторінки з описом представлених в цьому каталозі магазинів а також сторінки з описом магазинів, де буде відображена статистика товарів які представлені в каталозі.

Сервіс повинен відповідати наступним умовам:

- Інтерфейс повинен буди зручним та інтуїтивно зрозумілий користувачам;
- Зручне розділення товарів по категоріям;
- Повинен бути зручний пошук товарів в каталозі;
- У кожного товара є своя сторінка з детальним його описом;
- Зміни цін в магазинах відслідковуються кожного дня;
- У кожного магазина є своя сторінка зі статистикою та його описом.

Потрібно протестувати коректність роботи та швидкість веб сайту, роботу всіх потрібних сторінок та коректність роботи парсеру і бази даних.

2 ВИБІР МЕТОДУ РІШЕННЯ

2.1 Методи розробки

Є програми, які дозволяють автоматизувати множинні процеси інтернет-маркетингу. Вони необхідні багатьом бізнесменам, які або хочуть використовувати збір інформації з конкуруючих веб-джерел, або захистити себе від подібного «злодійства» контенту. У будь-якому випадку, працюючи з інтернет-ресурсом важливо знати про парсингу сайту. (Рисунок 2.1)

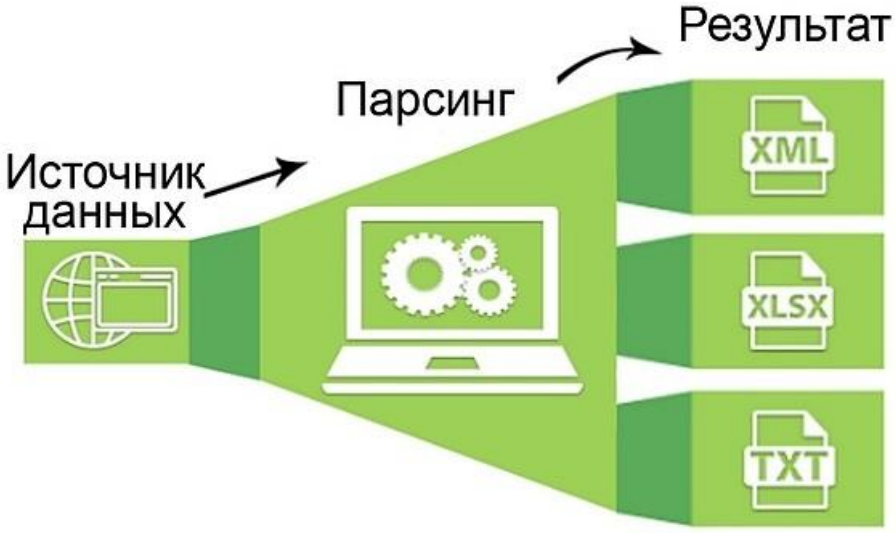


Рисунок 2.1 – Схема парсингу сайтів.

Цей механізм працює за певною програмою і створює деякий набір слів, з тим, що знайшлося в інтернеті. Як працювати з отриманою інформацією, написано в командному рядку, так званим «регулярним виразом». Цей рядок складається з певного набору символів і задає правила пошуку потрібної інформації.

Фактично поняття перекладається з англійської мови як семантичний аналіз або розбір. Але термін, який застосовується в технологіях створення та наповнення сайту, має ширше значення. Це процедура, дія, яка передбачає багатостороннє дослідження сторінки, документа, цілого розділу на предмет знаходження лексичних, граматичних одиниць або інших елементів (не тільки тексту, а й відео-, аудіо-контенту) з подальшою систематизацією. Шукані відомості знаходяться і перетворюються, вони готуються для подальшої роботи

з ними. Ще можна сказати, що це швидка оцінка і швидка обробка інтернет-ресурсу, даних з нього. Вручну подібний процес зайняв би багато часу, але автоматизація його значно спрощує.

Таким чином, парсер - це програма для парсинга ключових слів сайтів. Вона налаштовується, в неї вводяться параметри пошуку і інші вказівки, щоб отримати семантичне ядро або аналіз карток товарів для інтернет-магазину.

Ісходником може бути ваш власний веб-ресурс (для аналітики і прийняття подальших рішень), сайт конкурента, сторінка з соціальних мереж та ін. Отриманим результатом можна буде користуватися в подальшому на розсуд власника. Наведемо зрозумілий приклад. За таким принципом працюють пошукові системи, коли вони аналізують сторінки на релевантність, наявність ключових слів із запиту та відповідність тематиці, а потім на основі отриманих відомостей автоматично формується видача.

Визначення цінової політики - це найбільш ходова завдання для додатків. Для цього необхідно подивитися код аналізованого товару і ввести його в програму. Вона автоматично підтягне інші позиції, що відповідають запиту. Заощадити час і підвищити ефективність можна, якщо обмежити коло сторінок. Наприклад, так він не буде шукати по розділу з інформаційними статтями. Додавати варто категорії і самі картки продукції. Прописуються посилання на них в карті XML.

Для цього знадобиться вручну визначити код у кожного продукту, який вам потрібно. Потім можна підв'язати отримані відомості з автозаповненням полів у вашому інтернет-магазині. Особливо актуально підтягувати опис, коли ви займаєтеся реалізацією техніки, автомобілів, смартфонів. Часто характерні особливості беруться на сайтах виробників. Вони не можуть відрізнятися унікальністю, тому пошукові системи за це не лаються.

Процедура аналогічна - копіювання коду, а потім його введення в додаток для парсинга. Але дещо відрізняються наступні дії. Зазвичай коментарі відкриваються в той момент, коли користувач прокручує сторінку вниз, щоб

ознайомитися з ними. І тоді потрібно знову залізти в налаштування і змінити поле «Візуалізація» на JavaScript. В такому випадку програма буде себе вести точно як користувач, прокручуючи вниз контент до відгуків.

2.2 Мови програмування

За словами менеджерів фірми Microsoft, мова C # створювався в першу чергу для розробників, які використовують C і C ++, щоб дозволити їм більш ефективно створювати інтернет-додатки. Так, C # буде тісно інтегрований з мовою XML, протоколом SOAP і іншими веб-технологіями (на момент написання даного огляду деталі цієї інтеграції оголошені не були). Очевидно, що реалізувати будь-які нові можливості на рівні мови в мовах C / C ++ не можна, так як в цьому випадку був би порушений ANSI-стандарт; мова Visual Basic не надає ряду можливостей C / C ++; з мовою Java також неможливо звертатися як зі своїм власним. Тому фірма Microsoft вибрала інший шлях - створила нову мову.

Мова C # - це простий об'єктно-орієнтована мова, що нагадує C ++ і Java, але при цьому в ньому немає деяких конструкцій. Наприклад, C # не підтримує макроси, шаблонів, директив #include, а також різних способів доступу до об'єктів - замість того, щоб думати над тим, коли використовувати точку (.), Посилання (->) або оператор області дії (: :), ви завжди використовуєте точку. Для того щоб знизити можливість внесення помилок в створюваний код, в C # введений механізм збору сміття (garbage-collection): вас більше не повинні турбувати покажчики, посилання або витоку пам'яті - за всім цим стежить виконує ядро мови. У мові немає глобальних змінних, множинного спадкоємства і ряду інших конструкцій.

Мова C #, разом з Visual Basic (Visual Basic .NET, що об'єднав в собі функціональність Visual Basic і VBScript), Visual C ++ і скриптовою мовою JScript (JScript .NET), буде входити до складу Microsoft Visual Studio .NET (раніше називалася Visual Studio 7). Всі ці мови забезпечують доступ до платформи Microsoft .NET (раніше ця платформа називалася Next Generation

Windows Services NGWS), яка містить загальне виконує ядро і велику бібліотеку класів. Ядро працює на рівні спільної мови, відомого під назвою Common Language Subset (CLS, також називається Common Language Specification), який забезпечує взаємодію між усіма мовами і бібліотекою класів. Для розробників це означає, що C # буде мати доступ до всіх засобів, знайомим розробникам на Visual Basic і Visual C ++.

Більшість виразів в C # успадковано з мов C / C ++, але тут є ряд доповнень і змін. Відзначимо, що підтримуються вираження з міткою і оператор goto (можна очікувати нову хвилю обговорень з приводу цього оператора, але мені здається, що він буде існувати до тих пір, поки існує асемблерна інструкція JMP), оголошення локальних констант і локальних змінних перерахуванням (const int b = 2, c = 3;), обчислювані вирази і функції, оператори switch, while, do, for, foreach (для перебору елементів колекцій), break, continue, return, throw, try, checked / unchecked (для контролю переповнення при виконанні арифметичних операцій і перетворення цілочисельних типів) і lock.

Оголошення класів використовуються для завдання нових довідкових типів. Мова C # не підтримує множинного спадкоємства, але клас може реалізувати кілька інтерфейсів. Членами класу можуть бути константи, поля, методи, властивості, індексатори, події, оператори, конструктори, деструктори і вкладені опису типів. Кожен член класу може мати описувач доступу.

Структури багато в чому схожі з класами - вони можуть реалізовувати інтерфейси, можуть мати члени, але відрізняються від класів тим, що є значущими, а не посилальними типами і не підтримують успадкування.

Делегування дозволяє адресувати одні і ті ж покажчики на функції з мови C ++ і інших мов. На відміну від покажчиків на функції, делегати є об'єктно-орієнтованими і типізований. Делегати - це посилальні типи, створені на базі загального класу System.Delegate.

У програмах на C # використовуються простору імен. Ці простору імен служать як для внутрішньої, так і для зовнішньої організації системи - для подачі способу представлення програмних елементів для програм. У розглянутій вище демонстраційній програмі ми вже бачили, як використовується простір імен.

Властивість - це іменований атрибут, асоційований з об'єктом або класом. Прикладами властивостей можуть бути довжина рядка, розмір шрифту, заголовок вікна, ім'я клієнта і т.п. Властивості є розширеннями полів: і ті й інші - це іменовані члени з асоційованими типами, а синтаксис доступу до полів і властивостями однаковий. На відміну від полів властивості мають механізм для асоціації дій з читання і запису атрибутів об'єкта.

C # є процедурним мовою і, як всі процедурні мови, містить деякі декларативні елементи. Наприклад, доступність методу класу описується атрибутами public, protected, internal, protected internal або private. Підтримка атрибутів в мові C # дозволяє програмістам створювати нові типи декларативною інформації та витягувати цю інформацію під час роботи програми.

ASP.NET Core є останньою еволюцією популярного веб-застосування ASP.NET Microsoft, випущена в червні 2016 року. Останні версії ASP.NET побачили багато поступових оновленнь, зосереджуючись на високій продуктивності розробника та пріоритетності зворотної сумісності. ASP.NET Core витримує цю тенденцію, вносячи значні архітектурні зміни переосмислити спосіб проектування та побудови веб-рамок. ASP.NET Core багато в чому зобов’язаний своїй спадщині ASP.NET, і було використано багато можливостей вперед від раніше, але ASP.NET Core є новою основою. Вся технологія стек був переписаний, включаючи як веб-фреймворк, так і базовій платформі. В основі змін лежить філософія, яку ASP.NET повинен мати можливість голова висока, якщо вимірювати інші сучасні рамки, але ті, що існують Розробникам .NET слід і надалі залишати почуття знайомості.

Якщо ви не новачок у веб-розробці, це може бути непростим переїздом у область з такою кількістю мовних слів і безліччю продуктів, що постійно змінюються. Вам може бути цікаво, чи всі вони потрібні - наскільки важко повернути файл із сервера? Що ж, цілком можливо створити статичний веб-додаток без використання веб-рамки, але його можливості будуть обмежені. Як тільки ви захочете забезпечити будь-яку безпеку чи динамізм, ви, швидше за все, зіткнетеся з труднощами, і первісна простота, яка заманювала вас, зникне перед вашими очима! Так само, як ви, можливо, використовували рамки для розробки настільних або мобільних пристроїв для створення власних програм, ASP.NET Core робить написання веб-додатків швидшими, простішими та безпечнішими. Він містить бібліотеки для звичайних речей, таких як. Запорукою будь-якого сучасного веб-додатка є можливість генерувати динамічні веб-сторінки. Динамічна веб-сторінка відображає різні дані, наприклад, залежно від поточного користувача, який увійшов, або він може відображати вміст, поданий користувачами. Без динамічної основи неможливо було б увійти на веб-сайти чи відображати будь-які персоналізовані дані на сторінці. Коротше кажучи, веб-сайти на зразок Amazon, eBay та Stack Overflow були б неможливі.

Щоб зрозуміти, чому Microsoft вирішила створити новий фреймворк, важливо зрозуміти переваги та обмеження існуючої веб-рамки ASP.NET. Перша версія ASP.NET була випущена в 2002 році як частина .NET Framework 1.0, у відповідь на тодішні традиційні сценарії середовищ класичних ASP і PHP. ASP.NET Web Forms дозволив розробникам швидко створювати веб-додатки, використовуючи графічний дизайнер та просту модель подій, що відображає методи побудови настільних додатків. Рамка ASP.NET дозволила розробникам швидко створювати нові програми, але з часом екосистема веб-розробки змінилася. Стало очевидним, що веб-форми ASP.NET страждають від багатьох проблем, особливо при створенні великих додатків. Зокрема, відсутність перевірки, складна модель стану та обмежений вплив на створений HTML (що

ускладнює розробку на стороні клієнта) змусили розробників оцінювати інші варіанти. У відповідь Microsoft випустила першу версію ASP.NET MVC в 2009 році, засновану на моделі Model-View Controller, звичайній схемі веб-дизайну, що використовується в інших структурах, таких як Ruby on Rails, Django та Java Spring. Цей фреймворк дозволив вам відокремити елементи інтерфейсу від логіки програми, полегшив тестування та забезпечив більш жорсткий контроль над процесом генерації HTML. З моменту першого випуску ASP.NET MVC пройшов ще чотири ітерації, але всі вони були побудовані на тій самій базовій основі, що надається файлом System .Web.dll. Ця бібліотека є частиною .NET Framework, тому вона попередньо встановлюється для всіх версій Windows. Він містить увесь основний код, який ASP.NET використовує під час створення веб-програми. Ця залежність приносить як переваги, так і недоліки. З одного боку, рамка ASP.NET - це надійна, протестована на бій платформа, яка є прекрасним вибором для створення сучасних додатків у Windows. Він надає широкий спектр функцій, які багато років бачили у виробництві, і добре відомий практично всім веб-розробникам Windows. З іншого боку, ця залежність обмежує - зміни в базовій системі. Це обмежує те, наскільки ASP.NET може еволюціонувати і призводить до того, що цикли випуску відбуваються лише кожні кілька років. Існує також чітка зв'язок із веб-хостом Windows, Інтернет-сервісом інформації (IIS), що виключає його використання на платформах, що не належать до Windows. В останні роки багато веб-розробників почали дивитися на міжплатформові веб-рамки, які можуть працювати на Windows, а також Linux та macOS. Microsoft відчула, що настав час створити рамку, яка більше не прив’язана до спадщини Windows, таким чином ASP.NET Core народився.

Розвиток ASP.NET Core було мотивоване бажанням створити веб-структуру з чотирма основними цілями:

- Проводити та розробляти крос-платформу
- Мати модульну архітектуру для легшого обслуговування

- Розроблено повністю як програмне забезпечення з відкритим кодом
- Бути застосовним до сучасних тенденцій розвитку веб-сторінок, таких як клієнтські програми та розгортання у хмарних середовищах

Для досягнення всіх цих цілей Microsoft потребувала платформи, яка могла б забезпечити основні бібліотеки для створення основних об'єктів, таких як списки та словники, та виконання, наприклад, простих файлових операцій. До цього часу розробка ASP.NET завжди була зосереджена і залежала від .NET Framework лише для Windows. Для ASP.NET Core Microsoft створила легку платформу, яка працює на Windows, Linux та macOS під назвою .NET Core поділяє багато тих же API, що і .NET Framework, але вона менша і на даний момент реалізує лише підмножину функцій .NET Framework забезпечує, з метою надання більш простої моделі реалізації та програмування. Це абсолютно нова платформа, а не роздріб .NET Framework, хоча вона використовує аналогічний код для багатьох своїх API. Тільки за допомогою .NET Core можна створити консольні програми, які працюють із кросплатформою. Майкрософт створив ASP.NET Core, щоб бути додатковим шаром поверх консольних програм, таким чином, щоб перетворення у веб-додаток передбачало додавання та складання бібліотек.

Додавши веб-сервер ASP.NET Core до програми .NET Core, ваша програма може працювати як веб-додаток. ASP.NET Core складається з безлічі невеликих бібліотек, які ви можете вибрати, щоб надати додатку різні функції. Вам рідко потрібні всі наявні у вас бібліотеки, і ви додасте лише те, що вам потрібно. Деякі бібліотеки є загальними і з’являться практично в усіх створених вами програмах, наприклад, для читання конфігураційних файлів або проведення журналів. Інші бібліотеки ґрунтуються на цих базових можливостях, щоб забезпечити функціональні можливості, такі як третій вхід через Facebook або Google. Більшість бібліотек, які ви будете використовувати в ASP.NET Core, можна знайти в GitHub, у сховищах організацій Microsoft

ASP.NET Core за адресою https://github.com/aspnet. Тут ви можете знайти основні бібліотеки, такі як веб-сервер Kestrel та бібліотеки реєстрації, а також багато інших периферійних бібліотек, таких як сторонні бібліотеки аутентифікації. Усі додатки ASP.NET Core дотримуватимуться аналогічного дизайну для базової конфігурації, як це пропонують загальні бібліотеки, але загалом рамки є гнучкими, що дозволяє вам створювати власні кодові угоди. Ці спільні бібліотеки, бібліотеки розширень, що будуються на них, та конвенції дизайну, які вони рекламують, складають дещо неясний термін ASP.NET Core.

Однією з найбільш значущих переваг повноцінної системи .NET є її зрілість - вона розроблялася протягом 16 років, була загартованою і широко розгорнута. Для когось ця зрілість буде важливим визначальним фактором. Він уже буде встановлений на ваших серверах, а створення ASP.NET Core на вершині передбачає (відносно) невеликий ризик для вашого існуючого середовища. Для інших, зокрема існуючих розробників ASP.NET, кросплатформна та зручна для контейнерів .NET Core не буде звертатися. Ці розробники, за необхідності, будуть використовуватися для розгортання на серверах Windows, і цілком розумно бажати продовжувати це робити, при цьому все-таки користуючись всіма пропозиціями ASP.NET Core. Найбільшою причиною дотримуватися повної .NET Framework, коли вперше було випущено .NET Core, було те, що вам потрібно було використовувати функції, характерні для Windows, такі як Реєстр або Служби каталогів. З цього часу Microsoft випустила пакет сумісності, який робить ці API доступними в .NET Core, але вони доступні лише під час роботи .NET Core в Windows, а не в Linux або macOS. Якщо ви знаєте, що ваш додаток покладається на багато функцій лише для Windows, то .NET Framework може бути найпростішим варіантом.

Існує кілька версій HTML. Стандарт мови безперервно оновлюється і доповнюється, наслідок цього - щороку виходить нова версія HTML. З іншого боку - веб-браузери, за допомогою яких користувачі переглядають html-сторінки, також мають свої відмінності в частині інтерпретації та підтримки

окремих тегів. Це обумовлено тим, що розробкою браузерів займаються різні компанії. Що з цього випливає? Попросту кажучи, один і той же html-код різні браузери будуть відображати по-різному, а деякі теги окремі браузери взагалі "не розуміють".

HTML не являє собою мову програмування, HTML є мовою розмітки веб сторінок та використовується для того щоб повідомляти веб браузерам які саме та як показувати веб-сторінки, які саме зараз відкриті в браузері. HTML складається с деякої кількості елементів розмітки веб сторінки, які використовуються для того, щоб виділяти або відрізняти різні частини тексу один від одного. Елементи розмітки більш відомі як теги, можуть зробити текст більш товстим чи тонким, можуть відобразити текст по середині або збоку сторінки.

Атрибути в мові розмітки HTML мають у собі додаткову інформацію про тег, яку не потрібно показувати користувачеві. Клас дає змогу дати елементу ідентифікаційне ім'я, яке може пізніше використовуватися, щоб звертатися до елементу з інформацією про стилі та інші речі. Ви також можете мати у своєму розпорядженні елементи всередині інших елементів – дана операція називається вкладенням.

Елементи повинні відкриватися і закриватися правильно, тому вони явно розташовуються усередині або зовні один одного. Якщо вони перекриваються, як в прикладі вище, ваш веб-браузер буде намагатися зробити найкраще припущення на основі того, що ви намагалися сказати, що може призвести до несподіваних результатів.

Javascript - це мова програмування. Перше, що потрібно розуміти це те, що Javascript - це мова програмування. Це означає те, що з його допомогою ви можете користуватися всіма основними можливостями мов програмування: ви можете застосовувати умови (якщо - зроби "то", інакше - зроби "це"), цикли, перебирати якісь значення, і.т.д . Загалом, працювати з даними, обробляти їх і виконувати якісь дії по автоматизації.

Javascript - це клієнтський мову програмування. Це означає те, що він працює на стороні клієнта. Коли ми переходимо на який-небудь сайт в мережі Інтернет, ми робимо це по протоколу HTTP. Зі свого домашнього комп'ютера ми відправляємо запит на віддалений сервер, на якому розташовується сайт. І віддалений сервер нам уже надсилає відповідь (html-сторінку, яка буде відображена на домашньому комп'ютері в браузері). Виходить клієнт-серверна структура. Клієнт - це наш локальний комп'ютер, з якого ми працюємо. Сервер - це той віддалений комп'ютер на якому розташовується який-небудь сайт. Те, що потрібно розуміти - Javascript працює на клієнта.

Клієнтом для протоколу http є браузер. Звичайний браузер, за допомогою якого ви заходите на будь-які сайти по мережі Інтернет. Це може бути Google Chrome, Firefox, Яндекс Браузер і.т.д.

Javascript - це та мова програмування, який працює в браузерах. Можна спрощено сказати, що мова програмування Javascript вбудований в можливості браузера. При установці браузера ви вже маєте можливості для роботи з мовою Javascript. Звідси, найголовніший плюс мови Javascript - це те, що для того, щоб їм користуватися не потрібно встановлювати будь-яке додаткове програмне забезпечення. Всі вбудовано в браузер і маючи його, ви вже можете працювати з мовою Javascript. Звідси ж випливає мінус. Я нас виникають проблеми з переходом на нові версії мови Javascript. Коли мова буде оновлюватися і будуть з'являтися нові можливості, можуть виникнути деякі труднощі з підтримкою цих можливостей для всіх відвідувачів будь-якого сайту. Справа в тому, що на веб-сторінку, яка опублікована в мережі Інтернет, можуть заходити найрізноманітніші люди, з самих різних браузерів. У деяких людей встановлені свіжі версії браузерів, у деяких старі версії браузерів, звідси можуть виникати труднощі. У кого-то ці нові можливості підтримуються, у кого-то не підтримуються. Але, насправді є можливості як можна обійти цей мінус, як можна зробити можливість підтримки нових можливостей Javascript для всіх відвідувачів сайту і клієнтів, але це тема не цього відео.

Його головне завдання - внести можливості автоматизації, на веб-сторінки сайтів. В першу чергу ми можемо створювати якісь інтерактивні елементи, з якими користувач може взаємодіяти. Прикладом цього можуть бути різні калькулятори для сайтів (див. Відео). Користувач може взаємодіяти з елементами на цьому калькуляторі і за допомогою Javascript розраховуються значення, перераховуються якісь поля. Ви можете взаємодіяти з цими елементами і отримувати якийсь оброблений результат. Крім того, за допомогою мови Javascript ви можете робити різні слайдери, каруселі, картинки, які змінюються самі собою. Загалом, такі інтерактивні елементи дозволяє створювати саме Javascript. Підводячи підсумок, коли ми створюємо веб-сторінки, HTML дозволяє розмічати веб-сторінки (тобто говорити ніж є той чи інший елемент).

За допомогою технології CSS можна надавати для веб-сторінки виберіть візуальний ефект. Яким розміром, кольором, положенням повинні бути елементи. І, Javascript - це автоматизація і логіка веб-сторінки. Javascript дозволяє створювати інтерактивні елементи на веб-сторінці, обробляти події з цими елементами, створювати автоматизовані програми і.т.д.

CSS це проста мова дизайну, яка призначена для простішого відображення дизайну веб-сторінок. CSS обробляє зовнішній вигляд веб-сторінки. Використовуючи мову дизайну CSS, ви можете контролювати колір тексту, стиль шрифтів, відстань між параграфами, розміри і розташування колонок, що використовують фонові зображення і кольорів, макети дизайну, варіанти відображення на різних пристроях і розмірах екрану. А також безліч інших ефектів. CSS легко освоїти і зрозуміти, але він забезпечує потужний контроль над поданням HTML-документа. Більш часто CSS використовується разом з мовою розмітки HTML.

CSS локально за допомогою оффлайн кешу. Використовуючи це, ми можемо переглядати сайти перебуваючи оффлайн. Кеш також забезпечує швидке завантаження і кращу загальну продуктивність веб-сайту. Незалежність

від платформи. Скрипт забезпечує незалежність від платформи і підтримує новітні браузери.

2.3 База даних.

У тотальному змісті база даних - це деякий склад файлів та данних, які сформовані потрібним чином. Наприклад, ви зберігаєте всі ваші поштові листи, та вони розсортовані по відправникам, також можливо, у вас є колекція даних з інформацією по фінансовим справам: отриманим або наданими рахунками, витрат за чековою книжкою або балансом кредитної картки. Одним із типів баз даних є товари в інтернет магазинах, або в каталогах, які згруповані за виробником або ціною. Іншим типом є файли з електронними таблицями, які ви поєднуєте в групи по їхньому характеру використання. Якщо ви дуже відповідальна людина, то, використовуючи спеціальну структуру каталогів і підкаталогів, ви, можливо, впораєтеся з невеликою кількістю таблиць. В даному випадку ви будете виконувати роль диспетчера бази даних. Як переконатися, що дані вводяться правильно? Що робити, якщо одна і та ж інформація може знадобитися кільком користувачам, але при цьому не можна допустити, щоб дві людини в один і той же час коректували одні й ті ж дані? Коли ви опиняєтеся перед подібними проблемами, вам потрібна система управління базами даних (СКБД).

На сьогоднішній день майже всі прогресивні системи започатковані на реляційній (relational) моделі управління базами даних. Назва "реляційна" походить від того, що кожен окремий запис в цій базі даних містить інформацію, що відноситься (related) тільки до одного конкретного об'єкту. Між тим, з даними двох типів (наприклад, про клієнтів і замовленнях) можна працювати як з одним цілим, заснованим на значеннях пов'язаних між собою (related) даних. Наприклад, якщо включати прізвище та адресу клієнта в кожне замовлення від нього, то це призвело б до зберігання однакової інформації. Тому в реляційної системі інформація про замовлення має окреме поле даних,

куди додається ключ клієнта, за допомогою якого інформація про кожному замовленні об'єднується з даними про відповідному клієнта.

У реляційної СУБД всі використовувані дані надаються у вигляді окремих таблиць. Інформація про об'єкти певного типу представляється в табличному вигляді - в стовпчиках таблиці зосереджені різні характеристики цих об'єктів - атрибути, а рядки призначені для опису значень всіх атрибутів окремого об'єкта. Навіть в тому випадку, коли ви використовуєте функції СУБД для вибору інформації з однієї або декількох таблиць, результат представляється також в деякій табличному вигляді. Більш того, ви можете виконати запит за допомогою використання результатів іншого запиту до бази. Також ви можете об'єднати інформацію з декількох таблиць або запитів. Наприклад, щоб з'ясувати, які замовлення оформили ті чи інші клієнти, можна з'єднати інформацію про клієнтів з даними про замовлення, а для того, щоб встановити, хто із співробітників працював з даним замовленням, можна об'єднати інформацію про замовлення з інформацією про співробітників.

SQL - це не просто засіб для отримання знань із даних. Його також мова для визначення структур, які зберігають дані, щоб ми могли впорядкувати зв’язки в даних. Головне серед цих структур - це стіл. Таблиця - це сітка рядків і стовпців, що зберігають дані. Кожен рядок містить а колекція стовпців, і кожен стовпець містить дані визначеного типу: найчастіше цифри, символи та дати. Ми використовуємо SQL для визначення структура таблиці та те, як кожна таблиця може співвідноситися з іншими таблицями в база даних. Ми також використовуємо SQL для отримання або запиту даних із таблиць. Розуміння таблиць є основоположним для розуміння даних у ваших база даних. Щоразу, коли я починаю працювати зі свіжою базою даних, перше, що я робити - подивитися на таблиці всередині. Я шукаю підказки в назвах таблиці та їх колонна структура. Чи містять таблиці текст, цифри чи обидва? Скільки рядків у кожній таблиці? Далі я дивлюся, скільки таблиць знаходиться в базі даних. Найпростіший база даних може мати єдину таблицю. Повнорозмірний додаток,

яке обробляє Дані клієнтів або відстежують авіаперельоти можуть налічувати десятки чи сотні. The кількість таблиць говорить мені не тільки про те, скільки даних мені потрібно аналізувати, але також натякає, що я повинен вивчити взаємозв'язки між даними в кожному стіл.

Платформа ADO.NET Entity Framework - програмна модель, яка дозволяє заповнити прогалину між конструкціями бази даних і об'єктно-орієнтованими конструкціями. EF дозволяє взаємодіяти з реляційними базами даних, які мають справи з кодом SQL: виконуючого середовища EF генерує відповідні оператори SQL, коли розробник застосовує LINQ-запити до строго типізованим класах. Такі строго типізовані класи називаються сутностями. Суті - це концептуальна модель (Модель сутнісних даних, EDM - Entity Data Model) фізичної бази даних, яка відображається на предметну область. EDM являє собою набір класів клієнтської сторони, які відображаються на фізичну базу даних. Однак суті не зобов'язані безпосередньо відображатись на схему бази даних - сутнісні класи можна реструктурувати для відповідності існуючим потребам, і виконуючого середовища EF відобразить ці унікальні імена на конкретну схему.

Важливо відзначити, що тут, як і в багатьох випадках, назва класу відповідає назві пов'язаної з ним таблиці. варто пам'ятати, що завжди можна змінити сутність для більш точного відповідності конкретної ситуації. Для подальшої роботи з Entity Framework важливо також розуміти його структуру, з чого складається той прошарок між базою даних і кодом C #. Подібно будь-якій взаємодії ADO.NET, EF використовує постачальник даних (DDEX provider) ADO.NET для взаємодії зі сховищем даних. для окремих систем управління базами даних (СКБД), ймовірно, може потрібно буде інсталювати певне DDEX provider. Наприклад, для розглянутого далі PostgreSQL необхідний NpgsqlDdexProvider. Крім цього, API-інтерфейс EF містить служби об'єктів і клієнт суті. Служби об'єктів управляють сутностями клієнтської сторони при роботі з ними в коді: відстежують зміни сутності, керують відносинами між ними, забезпечують збереження змін в БД і збереження стану сутності за

допомогою сериализации. клієнт суті відповідає за роботу з постачальником даних для установки з'єднань, генерації SQL-операторів на основі LINQ-запитів і відображення витягнутих даних на коректні форми сутностей.

В результаті використання конструктора виходить файл з розширенням * .edmx, що містить XML-опису сутностей, фізичної БД і інструкцій по відображенню цієї інформації між концептуальної і фізичної моделями. при компіляції проекту цей файл застосовується для генерації трьох окремих XMLфайлов: для концептуальної (* .csdl), фізичної (* .ssdl) моделей і одного рівня відображення (* .msl). Дані з цих файлів потім об'єднуються з додатком у вигляді двійкових ресурсів. при компіляції збірка .NET має всі необхідні дані для викликів API-інтерфейсу EF, наявних в коді. Генерація файлу * .edmx дає в результаті сутнісні класи, які відображаються на таблиці БД, і клас, який розширює DbContext з System.Data.Entity, що є поєднанням патернів Unit Of Work і Repository, яке використовується для запитів з БД і угруповання внесених змін, які будуть додані в сховище як єдине ціле. DbContext концептуально схожий з ObjectContext з System.Data.Objects.

Таким чином, Entity Framework спрощує розробку клієнтського додатка, заповнюючи прогалину між відносинами БД і об'єктами мови програмування. Можливість генерації будь-якого з трьох рівнів клієнт-серверного додатка на основі іншого дозволяє говорити про EF як про CASE-інструменті. Автоматизація розробки необхідних для взаємодії з БД структур даних - сутнісних класів - дозволяє не тільки заощадити зусилля розробника, але і спростити його завдання в області реалізації GUI як візуального відображення даних.

2.4 Система контролю версій.

Як випливає з назви, контроль версій - це управління кількома версіями проект. Щоб керувати версією, кожна зміна (додавання, видання чи видалення) до файлів у проект слід відстежувати. Контроль версій записує кожну зміну, внесену до файлу (або групи) файлів) і пропонує спосіб скасувати або

відкрутити кожну зміну. Для ефективного контролю версій потрібно використовувати інструменти під назвою «системи контролю версій». Вони допомагають переходити між змінами та швидко повертаються до сторінки попередня версія, коли щось не так. Однією з найважливіших переваг використання контролю версій є робота в команді. Коли в проект бере участь більше однієї людини, відстеження змін стає кошмар, і це значно збільшує ймовірність перезапису чужої людини зміни. За допомогою версії управління декілька людей можуть працювати над їх копією проекту (Так звані гілки), і тільки об'єднати ці зміни в основний проект, коли вони (або інші члени команди) задоволені роботою.

Перші VCS були створені для управління вихідним кодом. Вони працювали, відстежуючи зміни, внесені до файлів у єдиній базі даних, яка зберігалася локально. Це означає, що всі зміни зберігалися в одному комп’ютері, і якщо виникли проблеми, вся робота була втрачена. Це також означає, що робота з командою не викликала сумнівів. Однією з найпопулярніших локальних ДМС була Система управління вихідним кодом або SCCS, яка була безкоштовною, але закритою. Розроблений компанією AT&T, він дико використовувався в 1970-х роках до виходу системи контролю за редакцією або RCS. RCS став більш популярним, ніж SCCS, тому що це був «Open Source», кросплатформний та набагато ефективніший. Випущений у 1982 році, RCS наразі підтримується Проектом GNU. Одним з недоліків цих двох локальних СКВ було те, що вони працювали над файлом за один раз; не було можливості відстежити з ними цілий проект. Централізовані VCS (CVCS) працюють, зберігаючи історію змін на одному сервері, до якого можуть підключитися клієнти (автори). Це пропонує спосіб працювати з командою, а також спосіб відстежувати загальний темп проекту. Вони все ще популярні, оскільки концепція така проста і її дуже легко створити. Основна проблема полягала в тому, що, як і місцевий VCS, помилка сервера може коштувати команді всю їх роботу. Також потрібно було підключення до мережі, оскільки

основний проект зберігався на віддаленому сервері. Розподілений VCS працює майже так само, як централізований VCS, але з великою різницею: немає головного сервера, який би зберігав усю історію. У кожного клієнта є копія сховища (разом із історією змін) замість перевірки одного сервера. Це значно знижує шанс втратити все, оскільки у кожного клієнта є клон проекту. З розподіленим VCS концепція наявності "основного сервера" розмивається, оскільки кожен клієнт, по суті, має всю власну владу в своєму сховищі. Це сильно заохочувало концепцію «роздрібнення» у спільноті з відкритим кодом. Форкінг - це акт клонування сховища, щоб внести власні зміни та по-іншому прийняти проект. Основна перевага розгортання полягає в тому, що ви також можете витягувати зміни з інших сховищ, якщо вважаєте за потрібне (а інші можуть робити те ж саме зі своїми змінами). Розподілена система управління версіями, як правило, швидша, ніж інші типи VCS, оскільки не потребує доступу до мережі до віддаленого сервера. Майже все робиться на місцях. Існує також невелика різниця в тому, як це працює: замість відстеження змін між версіями він відстежує всі зміни як «патчі». Це означає, що ці патчі можуть вільно обмінюватися між сховищами, тому не існує "головного" сховища, з яким можна було б іти в ногу з часом.

На відміну від багатьох систем управління версіями, Git працює зі знімками, а не з відмінностями. Це означає, що він не відстежує різницю між двома версіями файлу, але робить знімок поточного стану проекту. Ось чому Git дуже швидкий порівняно з іншими розподіленими VCS; тому також перехід між версіями та гілками настільки швидкий і простий. Пам'ятаєте, як працює централізована система управління версіями? Що ж, Git - це повна протилежність. Вам не потрібно спілкуватися з центральним сервером, щоб виконати роботу. Оскільки Git - це розподілений VCS, кожен користувач має власне повноцінне сховище зі своєю власною історією та наборами змін. Таким чином, все робиться локально, крім спільного використання патчів або наборів змін. Як було сказано раніше, центральний сервер не потрібен; але багато

розробників використовують такий варіант як умовний режим, оскільки так легше працювати. Якщо говорити про обмін патчем, як Git знає, які набори змін є чиїми? Коли Git робить знімок, він виконує контрольну суму на ньому; значить, які файли були змінені, порівнюючи контрольні суми. Ось чому Git може легко відслідковувати зміни між файлами та каталогами, а також перевіряє наявність корупційних файлів. Головною особливістю Git є його система "Три держави". Штати - це робочий каталог, область постановки та каталог git:

- Робочий каталог - це лише поточний знімок, над яким ви працюєте.
- Область постановки - це місце, де модифіковані файли позначені в їх поточній версії, готові до зберігання в базі даних.
- Каталог git - це база даних, де зберігається історія.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Розробка та реалізація вигляду сайту.

Головна сторінка сайту каталогу є одною з дуже важливих сторінок, через яку користувач визначає: чи подобається йому веб сайт та чи залишатися на ньому. На головній сторінці лежить велика відповідальність та від неї залежить перше враження користувача і його бажання і далі користуватися та чи залишатися і надалі на даному веб-сайті.

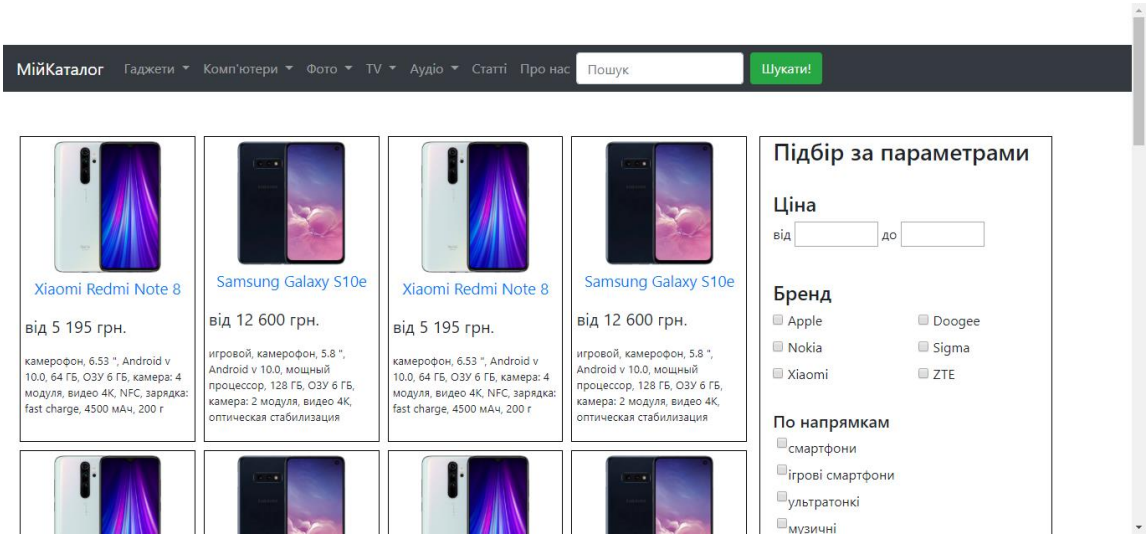


Рисунок 3.1 – Головна сторінка сервісу.

Картка товару (Рисунок 3.2-3.4) - одна з головних сторінок, що має великий вплив на рівень конверсії. Саме на цій сторінці користувач приймає рішення про покупку того чи іншого продукту.

Погляд користувача рухається зверху вниз з ліва на право. Найголовніший елемент в картці товару, це його назва, яке традиційно розміщується над фотографією по лівому краю. У деяких випадках його вставляють по центру, коли це дозволяє дизайн сайту. Начебто все логічно, проте не рідко можна побачити розташування назви під фото, що є помилкою, тому що користувач помічає його не відразу через що є відсоток тих відвідувачів, які йдуть просто не знайшовши назву.

«Краще один раз побачити, ніж сто разів почути». Картинки повинні бути якісними і надихаючими. З їх допомогою можна донести вигоду і користь від покупки даного продукту. Правильні картинки можуть значно підвищити

конверсію. Не варто забувати про розміщення додаткових зображень. Вони показують товар з різних сторін, щоб підкреслити деталі або унікальні риси.

Відгуки найчастіше виступають ключовим елементом в здійсненні покупки користувачем. Найчастіше розміщуються під описом або поруч з ним. У деяких випадках, якщо дозволяє дизайн, розміщуються в лінійку після фотографії та характеристик товару. Даний варіант розміщення дуже ефективний, так як погляд користувача відразу зустрічає даний елемент без прокрутки сторінки.

Система рейтингу (зірочки товару) дозволяє відвідувачеві визначити його якість без витрачання часу на прочитання відгуків. Ефективний елемент, який спрощує вибір користувачеві і робить його зручніше.

В цьому розділі відбувається відповідь на питання: «Навіщо мені це потрібно?», «Як це спростить мені життя?» і т.д. Завдання - розкрити переваги продукту і вигоди від його використання. Це можна робити як за допомогою текстової так і візуальної інформації (ролики, інфографіка, таблиці і т.д.).

Якщо є якась важлива інформація, яка не підійшла ні під який з елементів вище, то не забудьте згадати про неї окремо в спеціальному блоці.

У разі величезного асортименту даний елемент може виявитися необхідним. Після перегляду кількох релевантних товарів, користувач може прийти до висновку, що перший з них є кращим, і щоб знову не шукати даний продукт в каталозі, його можна швидко повернути через цей блок.

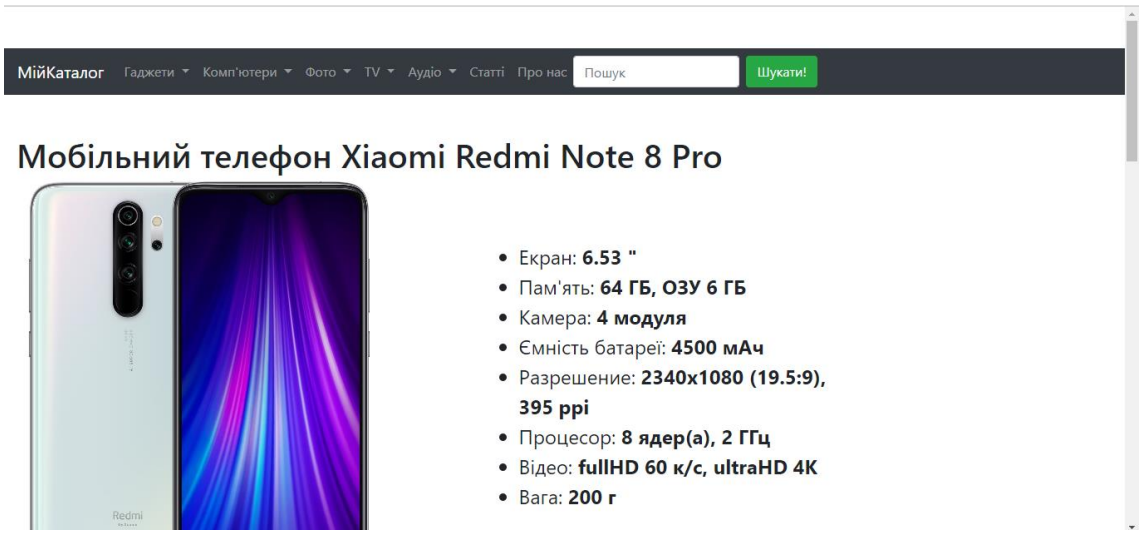


Рисунок 3.2 – Сторінка товару.

#	Зображення	Назва	Ціна	Магазин
1		Смартфон Xiaomi Redmi Note 8 Pro 6/128GB White 524142	6 499 грн.	MOYO
2		Xiaomi Мобильный телефон Xiaomi Redmi Note 8 Pro 6/128GB White	6 499 грн.	ROZETKA

Рисунок 3.3 – Опис магазинів де продається даний товар.

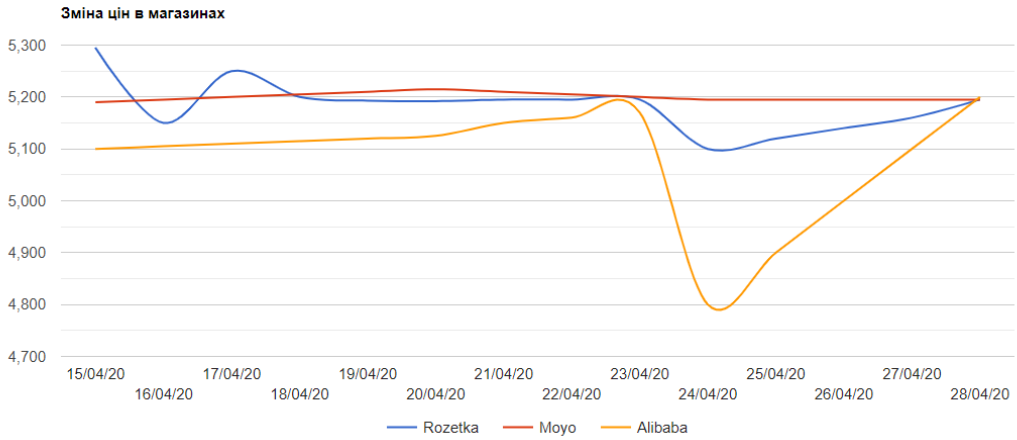


Рисунок 3.4 – Статистика зміни цій на цей товар.

Також було розроблено сторінку на якій показується інформація з описом магазинів (Рисунок 3.5), кількістю товарів в каталозі, та рейтингом довіри до них.

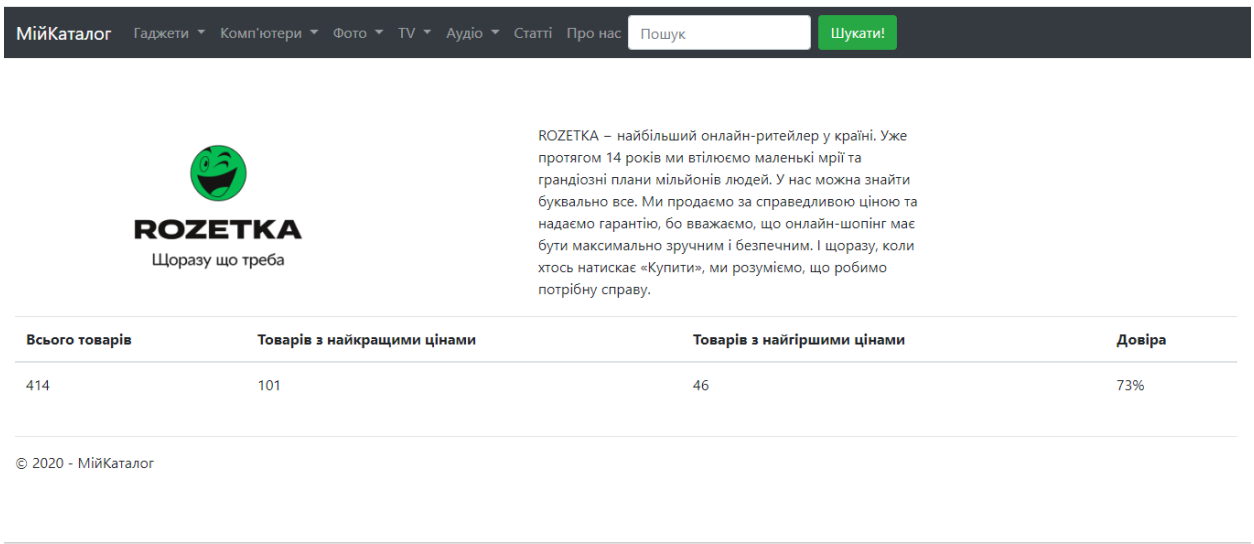
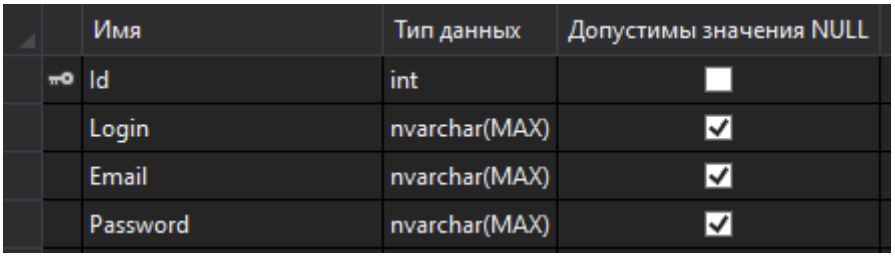


Рисунок 3.5 – Сторінка з інформацією про магазин.

3.2 Розробка бази даних.

При розробці бази даних було використано Entity Framework та створено декілька необхідних таблиць за допомогою нього.

Першою таблицею є таблиця з Користувачами, яка графічно зображена на малюнку 3.5.



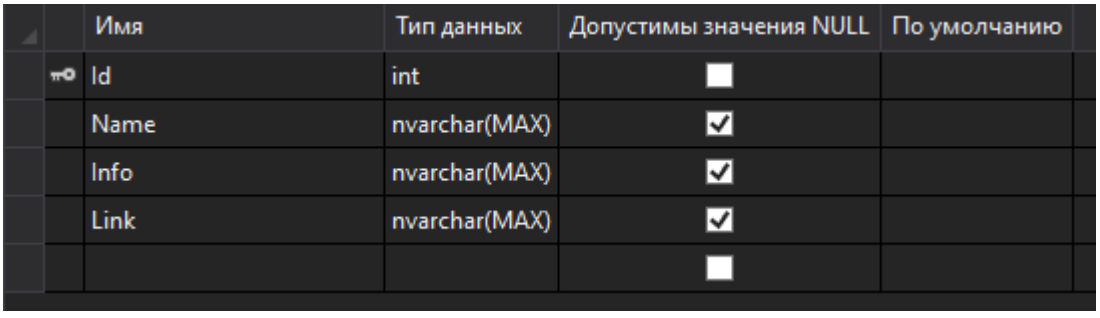
	Имя	Тип данных	Допустимы значения NULL
🔑	Id	int	☐
	Login	nvarchar(MAX)	☑
	Email	nvarchar(MAX)	☑
	Password	nvarchar(MAX)	☑

Рисунок 3.5 – Вигляд таблиці з користувачами

Та клас взаємодії таблиці з інформаційним додатком виглядає приблизно так:

```
namespace MyWebProject.DataBase
{
    public class User
    {
        public int Id { get; set; }
        public string Login { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
    }
}
```

Наступною таблицею є таблиця з інформацією про магазини, яка графічно зображена на малюнку 3.6.



	Имя	Тип данных	Допустимы значения NULL	По умолчанию
🔑	Id	int	☐	
	Name	nvarchar(MAX)	☑	
	Info	nvarchar(MAX)	☑	
	Link	nvarchar(MAX)	☑	
			☐	

Рисунок 3.6 – Вигляд таблиці з інформацією про магазини

Та клас взаємодії таблиці з інформаційним додатком виглядає приблизно так:

```
namespace MyWebProject.DataBase
{
```

```
    public class Shop
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Info { get; set; }
        public string Link { get; set; }
    }
}
```

Далі йде таблиця з інформацією про товари, яка графічно зображена на малюнку 3.7.

	Имя	Тип данных	Допустимы значения NULL	По умолчанию
🔑	Id	int	☐	
	Name	nvarchar(MAX)	☑	
	CategoryId	int	☐	
	Photo	nvarchar(MAX)	☑	
			☐	

Рисунок 3.7 – Вигляд таблиці з інформацією про товари

Та клас взаємодії таблиці з інформаційним додатком виглядає приблизно так:

```
namespace MyWebProject.DataBase
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int CategoryId { get; set; }
        public string Photo { get; set; }
    }
}
```

Наступною йде таблиця з інформацією про телефони, яка графічно зображена на малюнку 3.8.

	Имя	Тип данных	Допустимы значения NULL	По умолчанию
🔑	Id	int	☐	
	ProductId	int	☐	
	Screen	nvarchar(MAX)	☑	
	Memory	nvarchar(MAX)	☑	
	Capacity	nvarchar(MAX)	☑	
	Resolution	nvarchar(MAX)	☑	
	Processor	nvarchar(MAX)	☑	
	Video	nvarchar(MAX)	☑	
	Weight	nvarchar(MAX)	☑	

Рисунок 3.8 – Вигляд таблиці з інформацією про телефони

Та клас взаємодії таблиці з інформаційним додатком виглядає приблизно так:

```
namespace MyWebProject.DataBase
{
    public class Phones
    {
        public int Id { get; set; }
        public int ProductId { get; set; }
        public string Screen { get; set; }
        public string Memory { get; set; }
        public string Capacity { get; set; }
        public string Resolution { get; set; }
        public string Processor { get; set; }
        public string Video { get; set; }
        public string Weight { get; set; }

    }
}
```

Наступною йде таблиця з інформацією про категорії продуктів, яка графічно зображена на малюнку 3.9.

	Имя	Тип данных	Допустимы значения NULL	По умолчанию
🔑	Id	int	☐	
	ParentId	int	☐	
	Name	nvarchar(MAX)	☑	
			☐	

Рисунок 3.9 – Вигляд таблиці з категоріями

Та клас взаємодії таблиці з інформаційним додатком виглядає приблизно так:

```
namespace MyWebProject.DataBase
{
    public class Category
    {
        public int Id { get; set; }
        public int ParentId { get; set; }
        public string Name { get; set; }
    }
}
```

Останньою йде таблиця з інформацією про зміну цін на різні товари в магазинах, яка графічно зображена на малюнку 3.10.

	Имя	Тип данных	Допустимы значения NULL	По умолчанию
🔑	Id	int	☐	
	ShopId	int	☐	
	ProductId	int	☐	
	ShopLink	nvarchar(MAX)	☑	
	Price	int	☐	
	Date	datetime	☐	
			☐	

Рисунок 3.10 – Вигляд інформацією про ціни на продукти

Та клас взаємодії таблиці з інформаційним додатком виглядає приблизно так:

```
namespace MyWebProject.DataBase
{
    public class ProductShop
    {
        public int Id { get; set; }
        public int ShopId { get; set; }
        public int ProductId { get; set; }
        public string ShopLink { get; set; }
        public int Price { get; set; }
        public DateTime Date { get; set; }
    }
}
```

ВИСНОВКИ

Під час розробки дипломного проекту було спроектовано сервіс в якому є інтернет каталог товарів з різних інтернет магазинів, детальний опис товарів, статистика зміни цін на ці товари та статистика магазинів.

Було розроблено веб каталог товарів зі зручними сторінками на яких представлені товари, розроблено сторінки з детальним описом цих товарів та статистикою зміни цін, сторінки з описом представлених в цьому каталозі магазинів а також сторінки з описом магазинів, де відображена статистика товарів які представлені в каталозі, реалізована база даних за допомогою EntityFramework.

Сервіс відповідає наступним умовам:

- Інтерфейс є зручним та інтуїтивно зрозумілий користувачам;
- Зручне розділення товарів по категоріям;
- Розроблений зручний пошук товарів в каталозі;
- У кожного товара є своя сторінка з детальним його описом;
- Зміни цін в магазинах відслідковуються кожного дня;
- У кожного магазина є своя сторінка зі статистикою та його описом.

Було протестовано коректність роботи та швидкість веб сайту, роботу всіх потрібних сторінок та коректність роботи парсеру і бази даних.

Планується розширення функціоналу та корегування уже реалізованих функцій.

СПИСОК ЛІТЕРАТУРИ

1. Cleary S. Concurrency in C# Cookbook / Stephen Cleary. – United States of America: O`Reilly, 2019. – 254 с.
2. Miles R. The Department of Computer Science / Rob Miles. – UK, 2014. – 222 с.
3. Vystavěl R. C# Programming for Absolute Beginners / Radek Vystavěl., 2017. – 385 с.
4. Wright H. Software Engineering at Google / H. Wright, T. Manshreck, T. Winters., 2020. – 602 с.
5. McGrath M. C# Programming in easy steps / Mike McGrath., 2017. – 192 с.
6. Griffiths I. Programming C# 8.0: Build Cloud, Web, and Desktop Applications / Ian Griffiths. – United States of America: O`Reilly, 2020. – 800 с.
7. McGrath M. HTML5 in easy steps, Second Edition / Mike McGrath. – Warwickshire: In Easy Steps Limited, 2017. – 240 с.
8. Filipova O. Software Development From A to Z / O. Filipova, R. Vilão. – Berlin, Germany: apress, 2018. – 308 с.
9. Dean J. Web programming with HTML5, CSS, and JavaScript / Jogn Dean. – Burlington: Jones & Bartlett Learning, 2019. – 699 с.

ДОДАТОК А

Структура меню та футера

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - МійКаталог</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <nav class="navbar navbar-expand-sm bg-dark navbar-dark">
        <!-- Brand -->
        <a class="navbar-brand" href="/Home/Index">МійКаталог</a>

        <!-- Links -->
        <ul class="navbar-nav">

            <!-- Dropdown -->
            <li class="nav-item dropdown">
                <a  class="nav-link  dropdown-toggle"  href="#"  id="navbardrop"
data-toggle="dropdown">
                    Гаджети
                </a>
                <div class="dropdown-menu">
                    <a class="dropdown-item" href="#">Мобільні</a>
                    <a class="dropdown-item" href="#">Чохли</a>
                    <a class="dropdown-item" href="#">Планшети</a>
                    <a class="dropdown-item" href="#">Приставки</a>
                    <a class="dropdown-item" href="#">Наушники</a>
                </div>
            </li>

            <li class="nav-item dropdown">
                <a  class="nav-link  dropdown-toggle"  href="#"  id="navbardrop"
data-toggle="dropdown">
                    Комп'ютери
                </a>
                <div class="dropdown-menu">
                    <a class="dropdown-item" href="#">Ноутбуки</a>
                    <a class="dropdown-item" href="#">Монітори</a>
                    <a class="dropdown-item" href="#">ПК</a>
                    <a class="dropdown-item" href="#">Комплектуючі</a>
                </div>
            </li>

            <li class="nav-item dropdown">
                <a  class="nav-link  dropdown-toggle"  href="#"  id="navbardrop"
data-toggle="dropdown">
                    Фото
                </a>
                <div class="dropdown-menu">
                    <a class="dropdown-item" href="#">Фотоапарати</a>
                    <a class="dropdown-item" href="#">Об'єктиви</a>
                    <a class="dropdown-item" href="#">Спалахи</a>
                    <a class="dropdown-item" href="#">Штативи</a>
                </div>
```

```
            </li>

            <li class="nav-item dropdown">
                <a  class="nav-link  dropdown-toggle"  href="#"  id="navbardrop"
data-toggle="dropdown">
                    TV
                </a>
                <div class="dropdown-menu">
                    <a class="dropdown-item" href="#">Телевізори</a>
                    <a class="dropdown-item" href="#">ТВ тюнери</a>
                </div>
            </li>

            <li class="nav-item dropdown">
                <a  class="nav-link  dropdown-toggle"  href="#"  id="navbardrop"
data-toggle="dropdown">
                    Аудіо
                </a>
                <div class="dropdown-menu">
                    <a class="dropdown-item" href="#">Навушники</a>
                    <a class="dropdown-item" href="#">Гарнітури</a>
                    <a class="dropdown-item" href="#">Акустика</a>
                </div>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Статті</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Про нас</a>
            </li>
        </ul>
        <form class="form-inline" action="/action_page.php">
            <input class="form-control mr-sm-2" type="text" placeholder="Пошук">
            <button class="btn btn-success" type="submit">Шукати!</button>
        </form>
    </nav>
    <div class="container-fluid body-content">
        <br />
        <br />
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - МійКаталог</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```