МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

"ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

Кафедра "Обчислювальна техніка та програмування"

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

**до курсового проекту**

на тему Створення Web API

з використанням підходу Code First у Entity Framework

Виконав студент 3 курсу, групи КІТ-26Б

Керівник Шевердін І. В.

Харків 2018

РЕФЕРАТ

Пояснювальна записка до курсового проекту: 30 с., 5 рис., 3 джерел інформації.

Ключові слова: VISUAL STUDIO, INTERNET, С#, POSTMAN, WEBSITE, WPF, CONSOLE APPLICATION, MVC, WEB API.

У роботі описуються створення баз даних з зовнішнім обмеженням ключових слів (один до багатьох відносин), використовуючи метод Code First у структурі Entity, виводячи деякі зразки даних і повертаючи дані, використовуючи Web API. Доцільність даної розробки полягає в тому, щоб забезпечити зручну роботу з базами даних.

**З М І С Т**

[Перелік позначень і скорочень 4](#_Toc406602864)

[Вступ 5](#_Toc406602865)

[1   Огляд використовуваних методів і технологій 6](#_Toc406602866)

[1.1   Системи керування версіями 6](#_Toc406602867)

[1.1.1   Централізовані системи керування версіями 6](#_Toc406602868)

[1.1.2   Розподілені системи контролю версій 6](#_Toc406602869)

[1.1.3   Основи Git 7](#_Toc406602870)

[2   Розробка програмного комплексу 9](#_Toc406602871)

[2.1   Засоби, що використовувались 9](#_Toc406602872)

[2.1.1   Архітектура платформи .NET 9](#_Toc406602873)

[2.1.2   Робота з Git у Visual Studio 10](#_Toc406602874)

[2.1.3   Структура сховища 10](#_Toc406602875)

[2.1.4   Entity Framework 11](#_Toc406602875)

[2.1.5   Postman 13](#_Toc406602875)

[2.1.6   Технология ASP.NET 16](#_Toc406602875)

[2.2   Опис методології розробки ПЗ 22](#_Toc406602876)

[3   Опис програми 24](#_Toc406602878)

[Висновки 29](#_Toc406602879)

[Перелік джерел інформації 30](#_Toc406602880)

# Перелік позначень і скорочень

API – Application Programming Interface.

CMS – Content Management System.

DAO – Data Access Object.

HTML – HyperText Markup Language.

HTTP – HyperText Transfer Protocol.

IDE - Integrated Development Environment.

SSH - Secure Shell.

SSL - Secure Sockets Layer.

# Вступ

Мета роботи – продемонструвати ефективність та зручність розробки моделей баз даних за допомогою інтегрованого середовища розробки (IDE) Visual Studio та підхід по проектуванню баз даних Code First.

У пояснювальній записці наводяться технічні деталі процесу розробки програмного комплексу – від налаштування середовища до покрокової реалізації функціоналу.

# Огляд використовуваних методів і технологій

## Системи керування версіями

Система керування версіями (СКВ) – це система, яка зберігає зміни у файлі або декількох файлах, щоб потім можна було бачити потрібні старі версії. У роботі використовуються початкові коди програм, але насправді можна керувати версіями практично будь-яких типів файлів.

### Централізовані системи керування версіями

Наступним важливим питанням, з яким стикаються люди, є необхідність співпрацювати з іншими розробниками. Щоб справитися з цією проблемою, були розроблені централізовані системи контролю версій (ЦСКВ). Такі системи як CVS, Subversion і Perforce, мають єдиний сервер, який містить всі версії файлів, та деяке число клієнтів, які отримують файли з центрального місця. Протягом багатьох років, це було стандартом для систем контролю версій.

Такий підхід має безліч переваг, особливо над локальними СКВ. Наприклад, кожному учаснику проекту відомо, певною мірою, чим займаються інші. Адміністратори мають повний контроль над тим, хто і що може робити. Набагато легше адмініструвати ЦСКВ, ніж мати справу з локальними базами даних для кожного клієнта.

Але цей підхід також має деякі серйозні недоліки. Найбільш очевидним є єдина точка відмови, яким є централізований сервер. Якщо сервер виходить з ладу протягом години, то протягом цієї години ніхто не може співпрацювати або зберігати зміни над якими вони працюють під версійним контролем. Якщо жорсткий диск центральної бази даних на сервері пошкоджено, і своєчасні резервні копії не були зроблені, ви втрачаєте абсолютно все — всю історію проекту, крім одиночних знімків проекту, що збереглися на локальних машинах людей. Локальні СКВ страждають тією ж проблемою — щоразу, коли вся історія проекту зберігається в одному місці, ви ризикуєте втратити все.

### Розподілені системи контролю версій

Розподілена система контролю версії (англ. Distributed Version Control System, DVCS) — система, яка використовує замість моделі клієнт-сервер, розподілену модель зберігання файлів. Така система не потребує сервера, адже всі файли знаходяться на кожному з комп'ютерів.

Переваги:

* Кожний з розробників працює зі своїм власним репозитарієм;
* Рішення щодо злиття гілок приймається керівником проекту;
* Немає потреби в мережевому з'єднанні;

Недоліки:

* не можливо контролювати доступ до файлів;
* відсутня загальна нумерація версій файлів;
* потрібно значно більше дискового простору на сервері для зберігання всіх гілок як глобальних, так й локальних;
* відсутня можливість блокування файлів.

До розподілених систем відносять: Git, Mercurial, Bazaar, Monotone, Codeville, BitKeeper.

### Основи Git

У Git зовсім не такі поняття про інформацію і роботу з нею як в інших системах, хоча користувальницький інтерфейс дуже схожий. Проект створив Лінус Торвальдс для управління розробкою ядра Linux, а сьогодні підтримується Джуніо Хамано. Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів. Також можливе використання наступних протоколів:

* git-протокол (схема URI - git :) - відкритий протокол, що вимагає наявності на сервері запущеного git-демона (поставляється разом з Git), протокол не має коштів аутентифікації користувачів;
* SSH (ssh :) - використовує аутентифікацію користувачів за допомогою пар ключів, а також вбудований в Unix-систему «основний» SSH-сервер (sshd), з боку сервера потрібне створення облікових записів з git в якості оболонки;
* HTTP і HTTPS (http :, https :) - використовує інструмент curl (для Windows - поставляється разом з git), і його можливості HTTP-аутентифікації, як і його підтримку SSL і сертифікатів.

# Розробка програмного комплексу

## Засоби, що використовувались

### Архітектура платформи .NET

Платформа .NET складається з двох основних компонентів. Це Common Language Runtime і .NET Framework Class Library.

Common Language Runtime (скорочено CLR) можна назвати «двигуном» платформи .NET. Його завдання - забезпечити виконання додатків .NET, які, як правило, закодовані на мові CIL, розраховані на автоматичне управління пам'яттю і взагалі вимагають набагато більше турботи, ніж звичайні програми Windows. Тому CLR займається управлінням пам'яттю, компіляцією і виконанням коду, роботою з потоками управління, забезпеченням безпеки і т.п.

.NET Framework Class Library - це набір класів на всі випадки життя. По-перше, на платформі .NET реалізовані компілятори для різних мов програмування, і більшість цих мов дозволяють легко використовувати одну і ту ж бібліотеку класів. Тобто .NET Framework Class Library - це єдина бібліотека для всіх мов платформи .NET. По-друге, використання цієї бібліотеки дозволяє істотно скоротити розмір додатків, що сприяє їх поширенню через Internet.

Програма на мові C # виконується в середовищі .NET Framework - інтегрованому компоненті Windows, що містить віртуальну систему виконання (середа CLR) і уніфікований набір бібліотек класів. Середа CLR є комерційну реалізацію корпорацією Майкрософт інфраструктури CLI, яка є міжнародним стандартом, що лежить в основі створення середовищ виконання і розробки, в яких забезпечується тісний контакт між мовами і бібліотеками.

Вихідний код, написаний на мові C #, компілюється в проміжний мова (IL) відповідно до специфікації CLI. Код IL і ресурси, такі як растрові зображення і рядки, зберігаються на диску в виконуваному файлі, званому складанням, з розширенням EXE або DLL в більшості випадків. Збірка містить маніфест з відомостями про типи збірки, версії, мовою і регіональних параметрах і вимогах безпеки.

При виконанні програми на C# збірка завантажується в середу CLR в залежності від відомостей в маніфесті. Далі, якщо вимоги безпеки дотримані, середа CLR виконує JIT-компіляцію для перетворення коду IL в інструкції машинного коду. Середа CLR також надає інші служби, які стосуються автоматичного збору сміття, обробці виключень і управління ресурсами. Код, що виконується середовищем CLR, іноді називають "керованим кодом" в протиставлення "некерованого коду", який компілюється в машинний код, призначений для певної системи. Далі показані відносини під час компіляції і час виконання між файлами з вихідним кодом C#, бібліотеками класів .NET Framework, збірками і середовищем CLR.

### Робота з Git у Visual Studio

Починаючи з Visual Studio 2013 Update 1, користувачам Visual Studio доступний Git-клієнт, вбудований безпосередньо в IDE. Visual Studio вже протягом досить довгого часу має вбудовані функції управління вихідним кодом, але вони були орієнтовані на централізовані системи з блокуванням файлів, і Git не дуже добре вписувався в такій робочій процес. Підтримка Git в Visual Studio 2013 була істотно перероблена в порівнянні зі старою версією, і в результаті вдалося домогтися кращої інтеграції Visual Studio і Git.

Visual Studio запам'ятовує всі проекти, керовані за допомогою Git, які Ви відкрили, і вони доступні в списку в нижній частині вікна. Якщо в списку немає проекту, який вам потрібен, натисніть кнопку "Add" ("Додати") і вкажіть шлях до робочої директорії.

### Структура сховища

Назва репозиторію – зазначене при створенні репозитория ім'я користувача (username) в англійській транслітерації.

2.1.4 **Entity Framework**

ADO.NET Entity Framework (EF) - об'єктно-орієнтована технологія доступу до даних, є object-relational mapping (ORM) рішенням для .NET Framework від Microsoft. Надає можливість взаємодії з об'єктами як за допомогою LINQ у вигляді LINQ to Entities, так і з використанням Entity SQL. Для полегшення побудови web-рішень використовується як ADO.NET Data Services (Astoria), так і зв'язка з Windows Communication Foundation і Windows Presentation Foundation, що дозволяє будувати багаторівневі додатки, реалізуючи один з шаблонів проектування MVC, MVP або MVVM.

Реліз ADO.NET Entity Framework відбувся 11 серпня 2008 року в складі .NET Framework 3.5 Service Pack 1 і Visual Studio 2008 Service Pack 1. В VS 2008 увійшов EDM Wizard для реверс-інжинірингу існуючих баз даних і EDM Designer для редагування згенерованих моделей або створення їх з нуля.

23 червня 2008 року, ще до релізу першої версії, на стадії фінальної доведення Entity Framework V1, почався процес розробки Entity Framework V2.0. За словами англ. Tim Mallalieu, менеджера програми LINQ to SQL і EF, в .NET Framework 4.0 саме Entity Framework стане рекомендованої технологією доступу до реляційних СУБД за допомогою LINQ.

12 квітня 2010 року в складі релізу Visual Studio 2010 і .NET Framework 4.0 був представлена ​​Entity Framework 4.0. Пізніше вже окремо від фреймворка були представлені версії: 4.1 (квітень 2011), 4.2 (жовтень 2011), 4.3 (лютий 2012).

11 серпня 2012 року була представлена ​​версія 5.0.0, яка була призначена для .NET Framework 4.5. А 17 жовтня 2013 року була представлена ​​версія 6.0, яка вийшла під ліцензією Apache License v2, тим самим ставши open-source проектом.

Версія 6.0 була випущена 17 жовтня 2013 року і зараз це проект з відкритим вихідним кодом під ліцензією Apache License v2. У версії 6.0 був зроблений ряд поліпшень в підтримці методу роботи Code First.

Центральної концепцією Entity Framework є поняття сутності або entity. Сутність представляє набір даних, асоційованих з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх наборами.

Будь-яка сутність, як і будь-який об'єкт з реального світу, має низку властивостей. Наприклад, якщо сутність описує людини, то ми можемо виділити такі властивості, як ім'я, прізвище, зріст, вік, вага. Властивості необов'язково представляють прості дані типу int, а й можуть представляти більш комплексні структури даних. І у кожної сутності може бути одна або кілька властивостей, які будуть відрізняти цю сутність від інших і будуть унікально визначати цю сутність. Подібні властивості називають ключами.

При цьому суті можуть бути пов'язані асоціативної зв'язком один-до-багатьох, один-до-одного і багато-до-багатьох, подібно до того, як в реальній базі даних відбувається зв'язок через зовнішні ключі.

Відмінною рисою Entity Framework є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо не тільки отримувати певні рядки, що зберігають об'єкти, з бд, а й отримувати об'єкти, пов'язані різними асоціативними зв'язками.

Іншим ключовим поняттям є Entity Data Model. Ця модель зіставляє класи сутностей з реальними таблицями в БД.

Entity Data Model складається з трьох рівнів: концептуального, рівень сховища і рівень зіставлення (маппінга).

На концептуальному рівні відбувається визначення класів сутностей, які використовуються в додатку.

Рівень сховища визначає таблиці, стовпці, відносини між таблицями і типи даних, з якими порівнюється використовувана база даних.

Рівень зіставлення (маппінга) служить посередником між попередніми двома, визначаючи зіставлення між властивостями класу суті і стовпцями таблиць.

Таким чином, ми можемо через класи, визначені у додатку, взаємодіяти з таблицями з бази даних.

Способи взаємодії з БД

Entity Framework передбачає три можливі способи взаємодії з базою даних:

* Database first: Entity Framework створює набір класів, які відображають модель конкретної бази даних
* Model first: спочатку розробник створює модель бази даних, по якій потім Entity Framework створює реальну базу даних на сервері.
* Code first: розробник створює клас моделі даних, які будуть зберігатися в бд, а потім Entity Framework за цією моделлю генерує базу даних і її таблиці

Спочатку з найпершої версії Entity Framework підтримував підхід Database First, який дозволяв по готовій базі даних згенерувати модель edmx. Потім ця модель використовувалася для підключення до бази даних. Пізніше був доданий підхід Model First. Він дозволяв створити вручну за допомогою візуального редактора модель edmx, і по ній створити базу даних. Починаючи з 5.0 кращим підходом стає Code First. Його суть - спочатку пишеться код моделі на C #, а потім по ньому генерується база даних. При цьому модель edmx вже не використовується.

2.1.5 **Postman**

Основне призначення програми - створення колекцій із запитами до вашого API. Будь-розробник або тестувальник, відкривши колекцію, зможе з легкістю розібратися в роботі вашого сервісу. До всього іншого, Postman дозволяє проектувати дизайн API і створювати на його основі Mock-сервер. Вашим розробникам більше немає необхідності витрачати час на створення "заглушок". Реалізацію сервера і клієнта можна запустити одночасно. Тестировщики можуть писати тести і виробляти автоматизоване тестування прямо з Postman. А інструменти для автоматичного документування за описами з ваших колекцій заощадять час на ще одну "корисну фічу". Є дещо і для адміністраторів - автори передбачили можливість створення колекцій для моніторингу сервісів.

Головні поняття, якими оперує Postman це Collection (колекція) на верхньому рівні, і Request (запит) на нижньому. Вся робота починається з колекції і зводиться до опису вашого API за допомогою запитів. Давайте розглянемо докладніше все по порядку.

Collection

Колекція – відправна точка для нового API. Можна розглядати колекцію, як файл проекту. Колекція об'єднує в собі всі пов'язані запити. Зазвичай API описується в одній колекції, але якщо ви бажаєте, то немає ніяких обмежень зробити по-іншому. Колекція може мати свої скрипти і змінні, які ми розглянемо пізніше.

Folder

Папка - використовується для об'єднання запитів в одну групу всередині колекції. Наприклад, ви можете створити папку для першої версії свого API - "v1", а всередині згрупувати запити за змістом виконуваних дій - "Order & Checkout", "User profile" і т. П. Все обмежується лише вашою фантазією і потребами. Папка, як і колекція може мати свої скрипти, але не змінні.

Request

Запит - основна складова колекції, то заради чого все і затівалося. Запит створюється в конструкторі. Конструктор запитів це головне простір, з яким вам доведеться працювати. Postman вміє виконувати запити за допомогою всіх стандартних HTTP методів, всі параметри запиту під вашим контролем. Ви з легкістю можете поміняти або додати необхідні вам заголовки, cookie, і тіло запиту. У запиту є свої скрипти. Зверніть увагу на вкладки "Pre-request Script" і "Tests" серед параметрів запиту. Вони дозволяють додати скрипти перед виконанням запиту і після. Саме ці дві можливості роблять Postman потужним інструментом що допомагає при розробці та тестуванні.

Скрипти

"Postman Sandbox" це середовище виконання JavaScript доступна при написанні "Pre-request Script" і "Tests" скриптів. "Pre-request Script" використовується для проведення необхідних операцій перед запитом, наприклад, можна зробити запит до іншої системи і використовувати результат його виконання в основному запиті. "Tests" використовується для написання тестів, перевірки результатів, і при необхідності їх збереження в змінні.

Крім скриптів на рівні запиту, ми можемо створювати скрипти на рівні папки, і, навіть, на рівні колекції. Вони називаються також - "Pre-request Script" і "Tests", але їх відмінність в тому, що вони будуть виконуватися перед кожним і після кожного запиту в папці, або, як ви могли здогадатися, в усій колекції.

Змінні

Postman має кілька просторів і областей видимості для змінних:

* глобальні змінні,
* змінні колекції,
* змінні оточення,
* локальні змінні,
* змінні рівня даних.

Глобальні змінні і змінні оточення можна створити, якщо натиснути на шестірню в правому верхньому кутку програми. Вони існують окремо від колекцій. Змінні рівня колекції створюються безпосередньо при редагуванні параметрів колекції, а локальні змінні з виконуваних скриптів. Також існують змінні рівня даних, але вони доступні тільки з Runner.

Особливістю змінних в Postman є те, що ви можете вставляти їх в конструкторі запиту, в URL, в POST параметри, в Cookie, усюди, використовуючи фігурні дужки в якості плейсхолдера для підстановки.

Collection Runner

Призначений для тестування і виконання всіх запитів з колекції або папки, на ваш вибір. При запуску можна вказати кількість ітерацій, скільки разів буде запущена папка або колекція, оточення, а також додатковий файл зі змінними. Варто згадати, що запити виконуються послідовно, відповідно до розташування в колекції і папках. Порядок виконання можна змінити використовуючи вбудовану команду:

// Наступним виконається запит з назвою "Create order",

postman.setNextRequest ( 'Create order');

Після виконання всіх запитів формується звіт, який покаже кількість успішних і провальних перевірок з скриптів "Tests".

Console

Користуйтеся консоллю для налагодження ваших скриптів, і переглянути додаткову інформацію за запитами. Консоль працює, як під час запуску одного запиту, так і під час запуску пакета запитів через Runner. Щоб її відкрити, знайдіть іконку консолі в нижньому лівому кутку основного екрану програми.

## 2.1.6 Технология ASP.NET

Microsoft .NET Framework - це платформа для створення, розгортання і запуску Web-сервісів і додатків. Вона надає високопродуктивну, засновану на стандартах, багатомовне середовище, яке дозволяє інтегрувати існуючі застосування з додатками і сервісами наступного покоління, а також вирішувати завдання розгортання і використання інтернет-додатків. .NET Framework складається з трьох основних частин - загальномовного середовища виконання (common language runtime), ієрархічної безлічі уніфікованих бібліотек класів і компонентної версії ASP, звану ASP.NET.

ASP.NET - це частина технології .NET, використовувана для написання потужних клієнт-серверних інтернет додатків. Вона дозволяє створювати динамічні сторінки HTML. ASP.NET виникла в результаті об'єднання старішої технології ASP (активні серверні сторінки) і .NET Framework. Вона містить безліч готових елементів управління, використовуючи які можна швидко створювати інтерактивні web-сайти. Ви також можете використовувати сервіси, що надаються іншими сайтами, прозоро для користувачів вашого сайту. Загалом, можливості ASP.NET обмежені тільки вашою уявою.

Давайте обговоримо, що таке динамічні сторінки HTML і чим вони відрізняються від статичних. Статична сторінка містить код на мові гіпертекстової розмітки HTML. Коли автор сторінки пише її, він визначає, як буде виглядати сторінка для всіх користувачів сторінки. Зміст сторінки буде завжди однаковим незалежно від того, хто і коли вирішить її переглянути. Мови HTML цілком достатньо для відображення інформації, яка рідко змінюється і не залежить від того, хто її переглядає. Сторінка HTML - простий ASCII-текст, отже, клієнт може працювати в будь-якій операційній системі.

Цілком зрозуміло, що якщо сформувати web-сторінку, описавши її структуру засобами HTML, вона буде абсолютно статична в сенсі вмісту. Тобто при перегляді в браузері вона буде нести в собі точно ту ж інформацію, що була в неї записана в момент створення, і передані користувачем дані не можуть бути використані для модифікації вмісту відображаються йому: він зможе побачити тільки те, що попередньо було записано в кінцевий набір файлів.

Але що, якщо ми хочемо відобразити на сторінці поточний курс євро або прогноз погоди? Якщо ми написали сторінку HTML вчора, сьогодні вона вже застаріє. Отже, ми повинні вміти створювати динамічні сторінки. Динамічне наповнення сторінки - це інформація, зміст якої визначається тим, кому вона призначена, і яка відрізняється від перегляду до перегляду. Воно дозволяє забезпечити двосторонній обмін інформацією - від клієнта до сервера і назад.

Динамічними прийнято називати web-сторінки, які перед відправкою клієнту проходять цикл обробки на сервері. У найпростішому випадку це може бути деяка програма, яка модифікує запитувані клієнтом статичні сторінки, використовуючи параметри отриманого запиту і деяке сховище даних. Навіть при такій примітивній організації «нерозв'язна» завдання з попереднього абзацу знаходить очевидне рішення: досить підготувати всього одну статичну сторінку - шаблон - і перед відправкою сторінки програмно підставляти в нього значення, отримане сьогодні з банку або метеобюро.

Більшість сторінок на ранніх стадіях розвитку інтернету були статичними. Останні 10 років зростає кількість динамічних сторінок. І це зрозуміло, користувачі інтернет хочуть не тільки читати готову інформацію, а бути активними дійовими особами. Наприклад, вони замовляють товари в інтернет-магазині, пишуть щоденники, беруть участь в конкурсах. Інформаційні портали оновлюють новини щохвилини. Динамічні сторінки можуть підлаштовуватися під конкретного користувача, а також реагувати на його дії в браузері. Яким же чином? Для цього придумано безліч технологій. Наприклад, того, щоб ідентифікувати користувача і зберегти його налаштування для даного сайту, застосовуються файли-cookies.

Існують мови, здатні динамічно змінювати вміст веб-сторінки. З одного боку, це мови скриптів, що виконуються безпосередньо у клієнта. Приклади скриптових мов - JavaScript і VBScript. Скрипти на цих мовах вбудовані в код HTML, який сервер посилає браузеру. Сценарії, що виконуються на стороні клієнта, виділяються тегами і. Браузер інтерпретує цей код і показує користувачеві результат. Сам код можна переглянути через View Source браузера. Природно, ці програми не можуть бути більшими. Наприклад, якщо потрібно виконати пошук в базі даних, ми давно б відправив користувачеві весь її вміст. Але скрипти можуть перевірити правильність запиту, введеного в форму, тоді не доведеться перезавантажувати сервер обробкою неправильних запитів. Деякі програмісти створюють на JavaScript анімаційні ефекти. Одна студентка intuit.ru бажала знайти скрипт, який би відправляв SMS-повідомлення. Нажаль це неможливо. Виконуваних на стороні клієнта сценаріїв недостатньо для створення повноцінних динамічних сторінок. Навіть якщо на сторінці використовується JavaScript, анімовані картинки .gif, вона називається статичною.

Динамічна веб-сторінці повинна бути створена «на льоту» програмою, що виконується на інтернет-сервері. Широко застосовуються механізм шлюзів CGI (Common Gateway Interface). Спочатку користувач отримує статичну сторінку з формою. Вам відомо, що в тезі FORM існує атрибут ACTION. Саме він задає адресу (URL) виконуваного додатка. На сервері знаходяться виконувані файли програм, написаних, наприклад на C / С++ або Дельфі, які по протоколу HTTP приймають дані з вхідного потоку або з змінних оточення і записують в стандартний вихідний потік готову сторінку.

Користувачеві у відповідь на запит надсилається HTML код, який був спеціально згенерований для нього. Це може бути, наприклад, результат пошуку в пошуковій системі. CGI -скріпти можуть бути написані на що інтерпретується мовою (Perl) або навіть скрипті командного рядка. Вхідний і вихідний потоки перепризначувалися. На вхід інтернет-сервер приймає дані, введені користувачем. Після обробки отриманих даних, користувачеві повертається результуюча сторінка. При виконанні cgi-програма завантажується в пам'ять сервера, а при завершенні - видаляється. Коли 100 клієнтів одночасно звертаються до сервера, в пам'яті створюються 100 процесів, для розміщення коду кожного з яких потрібна пам'ять. Це негативно позначається на масштабованості. Нагадаємо, що масштабованість - це можливість плавного зростання часу відповіді програмної системи на запит з ростом числа одночасно працюючих користувачів.

Для вирішення це проблеми Microsoft була запропонована альтернатива - ISAPI (Internet Server Application Programming Interface) -Розширені і фільтри. Замість виконуваних файлів використовуються DLL - бібліотеки. Код DLL знаходиться в пам'яті весь час і для кожного запиту створює не процеси, а нитки виконання. Всі нитки використовують один і той же програмний код. ISAPI-додаток виконується в процесі IIS-сервера. Це дозволяє підвищити продуктивність і масштабованість.

ISAPI-розширення можна створювати в Visual Studio C ++ 6.0, користуючись майстром.

У ISAPI теж є недоліки, що відносяться до розробки. Якщо ми змінюємо вихідний код dll, ми повинні його відкомпілювати і помістити у виконувану директорію сервера. Але так як попередній варіант dll знаходиться в пам'яті, необхідно зупинити сервер, щоб отримати доступ на зміну файлу. В цей час клієнти не зможуть отримати в сервера жоден документ, і, звичайно, будуть не задоволені.

Скриптові мови, яка крутиться на стороні сервера - php і asp. Технологія asp була розроблена Microsoft в 90-х роках.

Виконання коду asp підтримується ISAPI-розширенням сервера. У діалозі конфігурації сервера IIS визначаються способи обробки файлів з різними розширеннями. Для обробки URL-адреси з розширенням в установках сервера визначено файл asp.dll. Файли asp відправляються до нього на обробку. На вхід надходить asp, а на виході маємо потік HTML-коду.

Технологія asp була обмежена за своїми можливостями. Він використовував скриптові мови, які мають менше можливостей, ніж повнофункціональні мови програмування. Код asp був вбудований в HTML у вигляді спеціальних тегів, що створювало плутанину. Шматочки asp були розкидані по ньому, як родзинки в булці. Але HTML код зазвичай створюють дизайнери, які вміють «робити красиво», а asp - програмісти, які змушують все це працювати. У ASP.NET ви можете тримати код asp і HTML в різних файлах.

Скриптові мови не підтримують строгу типізацію. Що це означає? Ви можете не описувати змінну до її використання і можете привласнювати їй значення різних типів. Це зручно, але створює ґрунт для помилок. Наприклад, у вас є змінна x1, і ви привласнюєте їй значення 1, але ви зробили помилку і помилково написали x2 = 1. Буде створена нова змінна x2, а значення x1 не зміниться. У мові зі строгою типізацією компілятор помітить, що змінна x2 не описує, і видає помилку.

У 2000 році на конференції розробників в якості частини нової технології .NET Microsoft представила ASP +. З виходом .NET Framework 1.0 вона стала називатися ASP.NET.

ASP.NET - це не продовження ASP. Це концептуально нова технологія Microsoft, створена в рамках ідеології .NET. У ASP.NET закладено все, для того, щоб зробити весь цикл розробки веб-додатки більш швидким, а підтримку більш простий. ASP.NET заснована на об'єктно-орієнтованої технології, але зберегла модель розробки asp: ви створюєте програму і ставите її в директорію, виділену сервером, і вона буде працювати. У ASP.NET з'явилося багато нових функцій, а існували в asp значно вдосконалені.

У ASP.NET використовуються компільовані мови. Під час компіляції перевіряється синтаксична коректність вихідного тексту. Скомпільований в проміжний мова код виконується швидше, і він буде таким же незалежно від мови, який ми використовуємо. Компільовані мови підтримують строгу типізацію.

Компіляція відбувається на сервері в момент першого звернення користувача до сторінки. Якщо програміст змінив текст сторінки, програма перекомпілюється автоматично. При написанні коду можна використовувати набір компонентів, що поставляються з .NET.

Платформа .NET Framework надає додаткам середовище виконання, сама безпосередньо взаємодіючи з операційною системою. Вище лежить інтерфейс ASP.NET додатків, на якому в свою чергу базуються веб-форми (ASP.NET сторінки) і веб-сервіси. Інтерфейс .NET Framework дозволяє стандартизувати звернення до системних викликів і надає середовище для більш швидкої і зручної розробки. CLR забезпечує єдиний набір сервісів для всіх мов.

ASP.NET використовує технологію доступу до даних ADO.NET, яка забезпечує єдиний інтерфейс для доступу до баз даних SQL Server і файлам XML. Крім того, посилена модель безпеки дозволяє забезпечувати захист клієнта і сервера від несанкціонованого доступу.

У 2004 році з'явилася версія ASP.NET 2.0 (бета-версія, остаточний вихід - кінець 2005-початок 2006). Як стверджується, ця версія дозволяє скоротити обсяг кодування на 70%. Нові можливості версії 2.0 - наприклад, використання шаблонів дизайну сторінок (Master Page), спрощена локалізація Web-додатків, більше 50 нових серверних елементів управління. Цілі, які переслідували розробники нової версії - підвищити швидкість розробки сайтів, масштабованість, легкість підтримки і адміністрування сайтів, швидкість роботи сервера. З'явилася панель остнасткі MMC (консоль управління Microsoft), що надає графічний інтерфейс для управління настройками ASP.NET. Змінювати настройки проекту тепер можна і через web-інтерфейс. ASP.NET 2.0 підтримує роботу на 64-бітових процесорах. Сервіс персоналізації (personalization) надає готове рішення для зберігання персональних даних, безпосередньо характеризують користувача сайту, так званого профілю користувача (Profile).

Шаблони дизайну, теми і скіни дозволяють незалежно дизайн всього сайту окремо від його функціональності, теми включають графіку і каскадні таблиці стилів.

Попередні версії Visual Studio для проектів ASP.NET вимагали наявності на машині розробника сервера IIS. Тепер сервер вбудований в середу розробки.

ASP.NET 2.0 і Visual Studio 2005 надає інструменти для легкого побудови локалізуємих сайтів, які визначають вибір мови користувача і посилають йому сторінки на його мові.

Можливість перекомпіляції дозволяє виявити помилки до завантаження сторінок на сервер. Можна не зберігати на сервері вихідні сторінки aspx, тим самим захищаючи свою інтелектуальну власність.

У ASP.NET 2.0 вбудована технологія автоматичного оновлення кешування баз даних. Дані, отримані з бази, зберігаються на сервері і він не звертається до бази для обробки повторного запиту. При зміні бази даних кеш оновлює свій вміст.

ASP.NET - це технологія, а не мова, і дозволяє програмувати на різних мовах - С#, Visual Basic, J#. У платформі .NET всі мови рівні, але деякі рівніші (Дж. Оруелл). Ось такою мовою і є С#, тому що він був спеціально створений для цієї платформи. Програмування C# дозволяє повною мірою використовувати концепції, методи і патерни об'єктно-орієнтованої розробки. Мова Visual Basic 8.0 наділений майже тими ж можливостями. Щоб навчитися ASP.NET, вам потрібно знати основи HTML, а знання asp не обов'язково. Воно може навіть перешкодити, так як доведеться міняти спосіб мислення. Також для розуміння багатьох бажано знати CSS і JavaScript.

## Опис методології розробки ПЗ

Була використана гнучка методологія, тобто на кожному кроці було можливість повернутися назад, на одну ітерацію. Для даного проекту спочатку був визначений алгоритм, написана програма, виконані тести, доопрацьована програма та написана документація

# Опис програми

Створюємо проект ASP.NET Web Application для роботи з базою даних “ComputersDBCW”. Вибираємо Web API. Тоді вже створюються автоматично деякі класи. Запускаємо проект та отримуємо інформацію про наш проект на сайті «[http://localhost:62494/api/computers» (рис](http://localhost:62494/api/computers). 1).

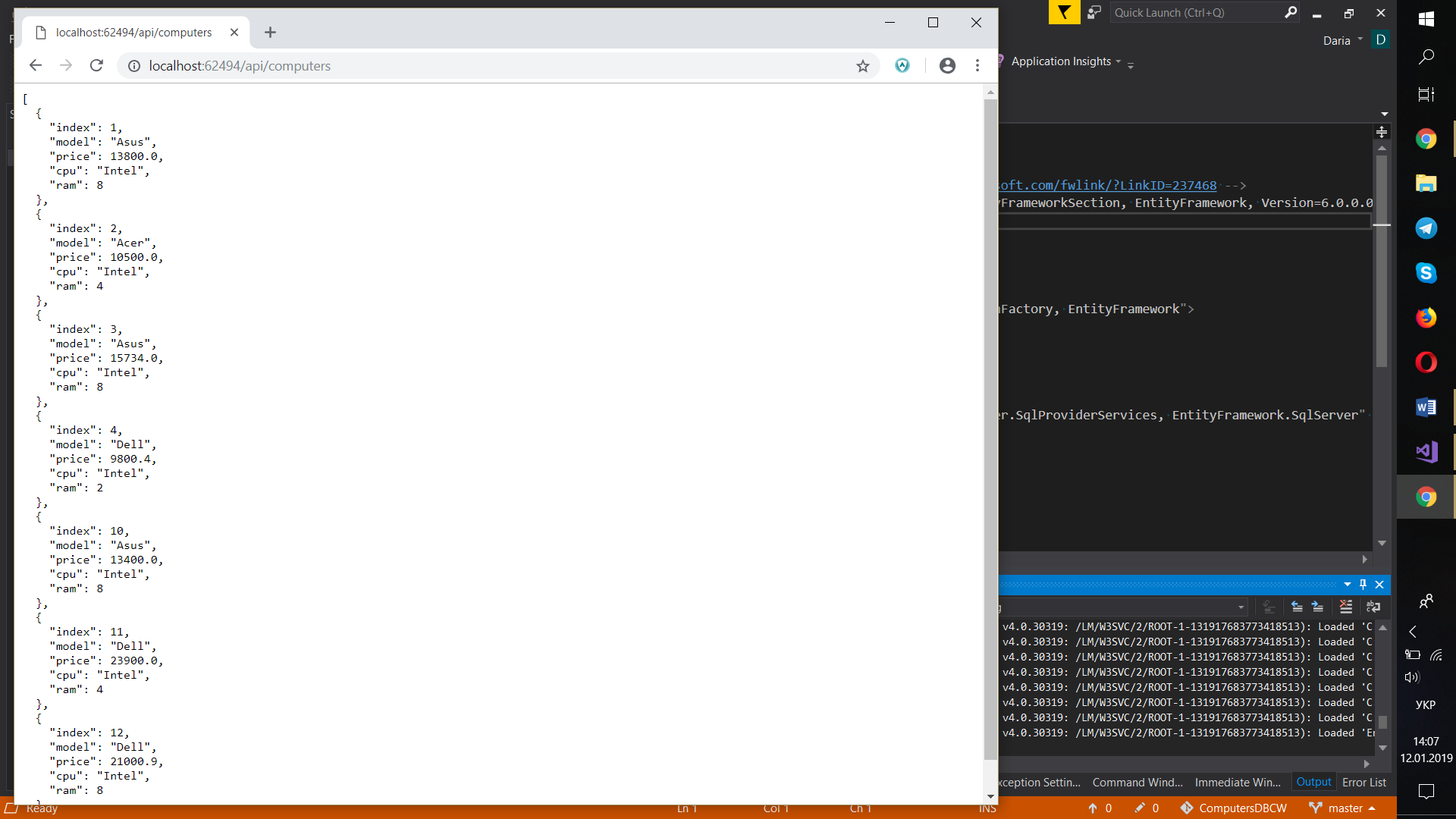


Рисунок 1 – Результат роботи Web API.

Перевірку цього проекту ми виконуємо за допомогою програми Postman (рис. 2, 3).

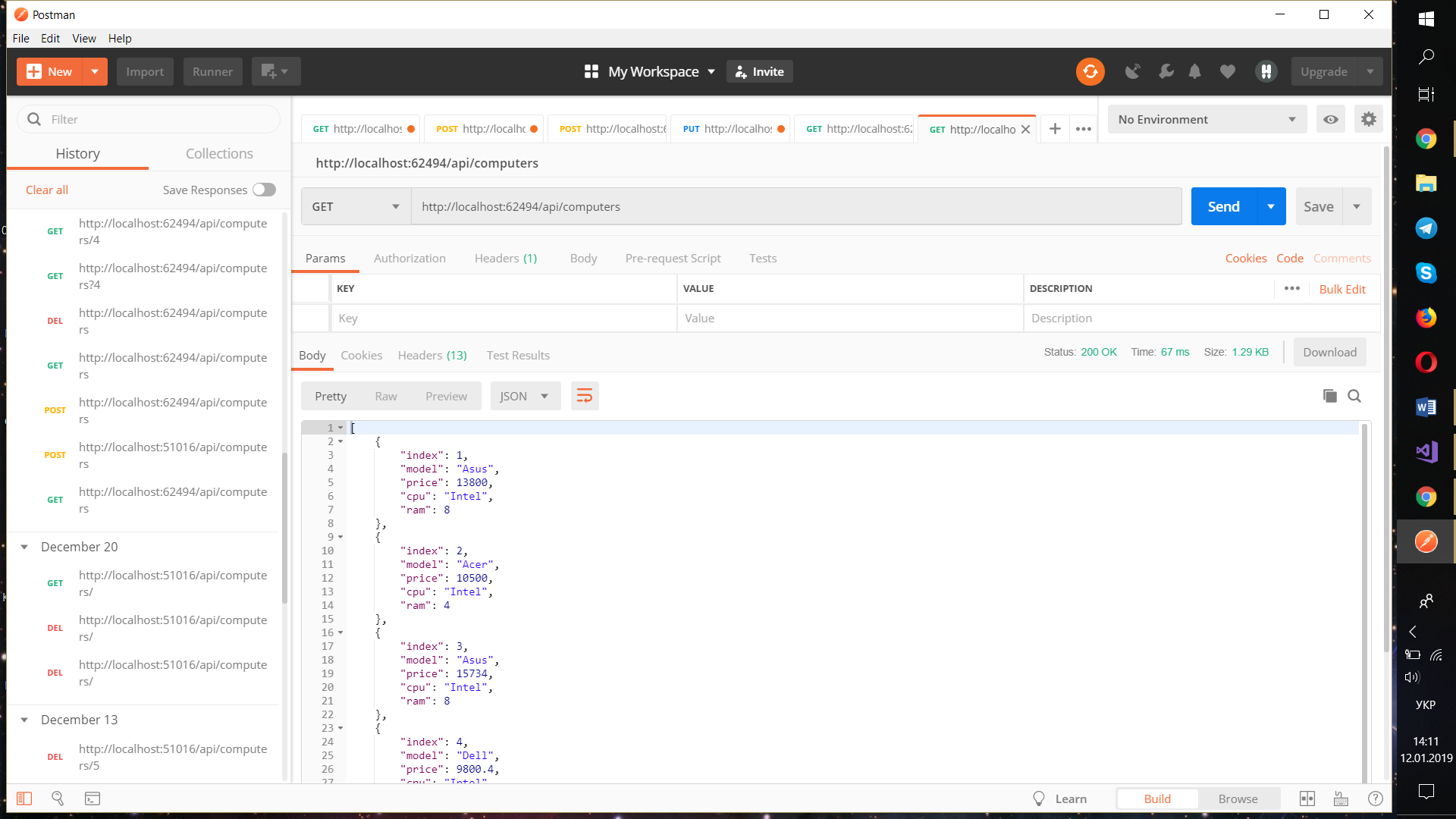


Рисунок 2 – Виконання запросу GET

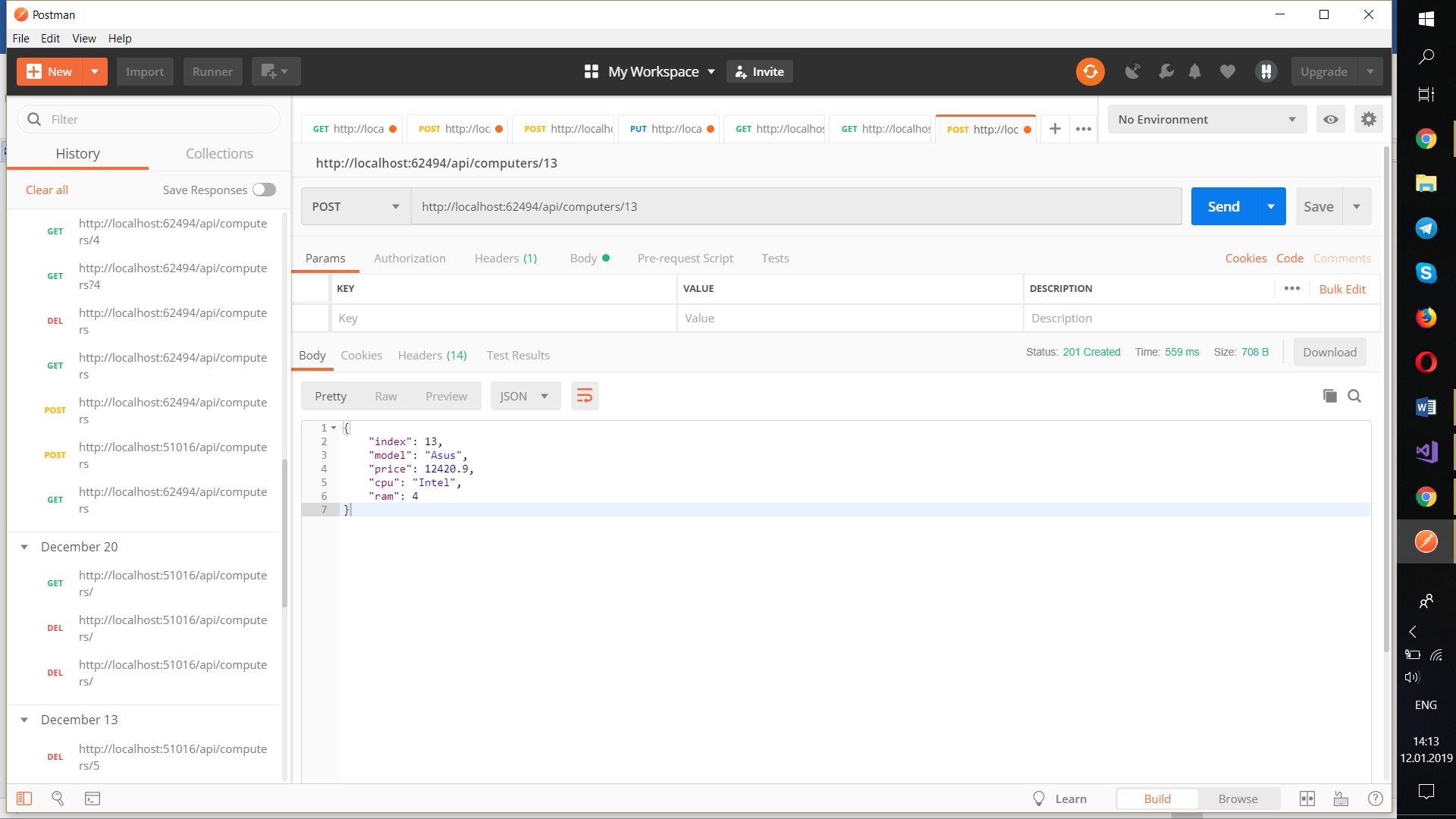


Рисунок 3 – Виконання запросу POST

Наступний проект ASP.NET Web Application – MVC. Автоматично створюється деякі файли. Ми додаємо деякі файли Views/Home. Додамо такі файли до цієї папки Edit.cshtml, Delete.cshtml, Add.cshtml та змінюємо файл Index.cshtml. Додаємо 2 контролера, запускаємо проект і за посиланням «http://localhost:52044/Computers» отримуємо результат роботи програми(рис. 4,5).

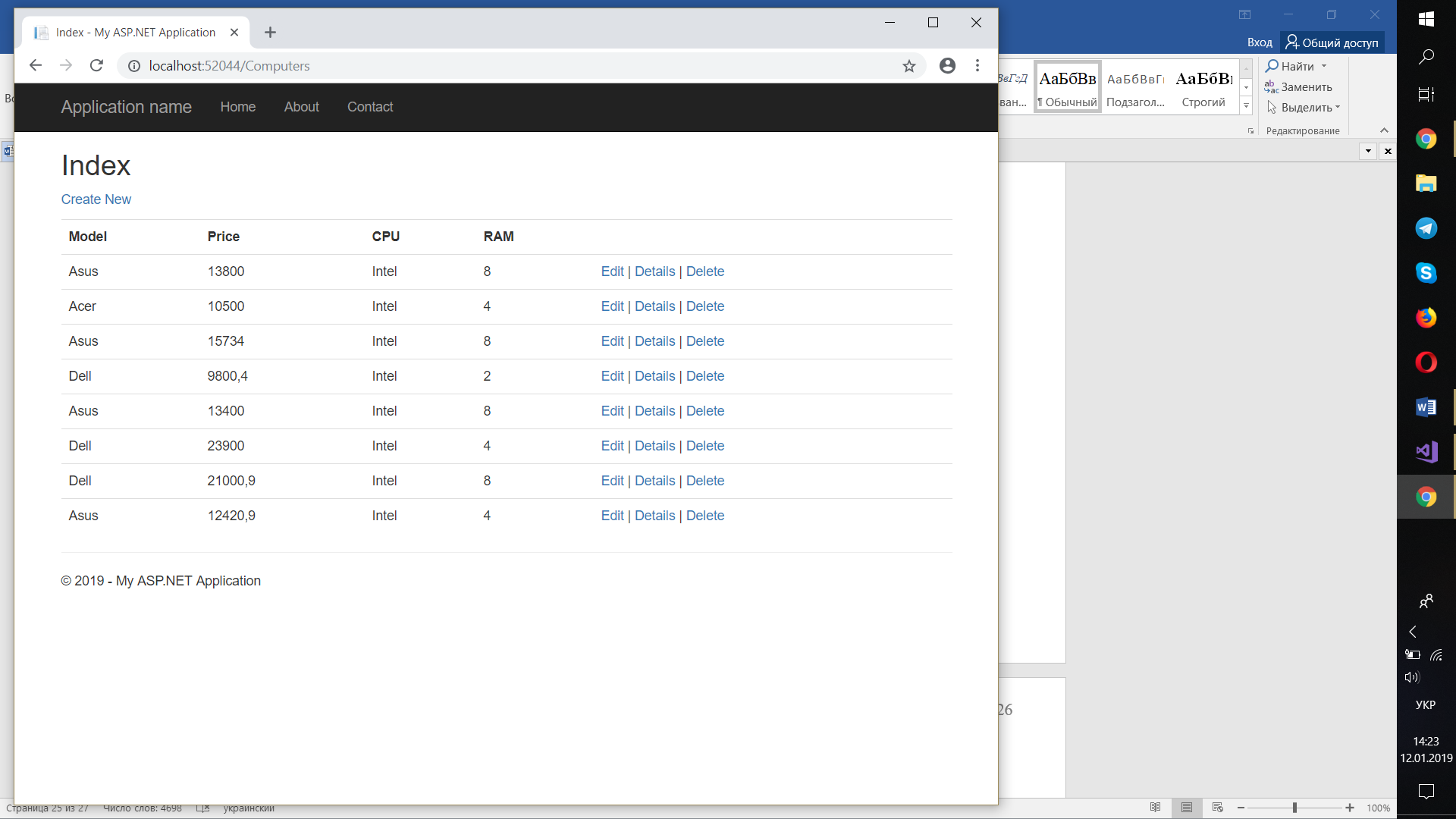


Рисунок 4 – Результат роботи програми

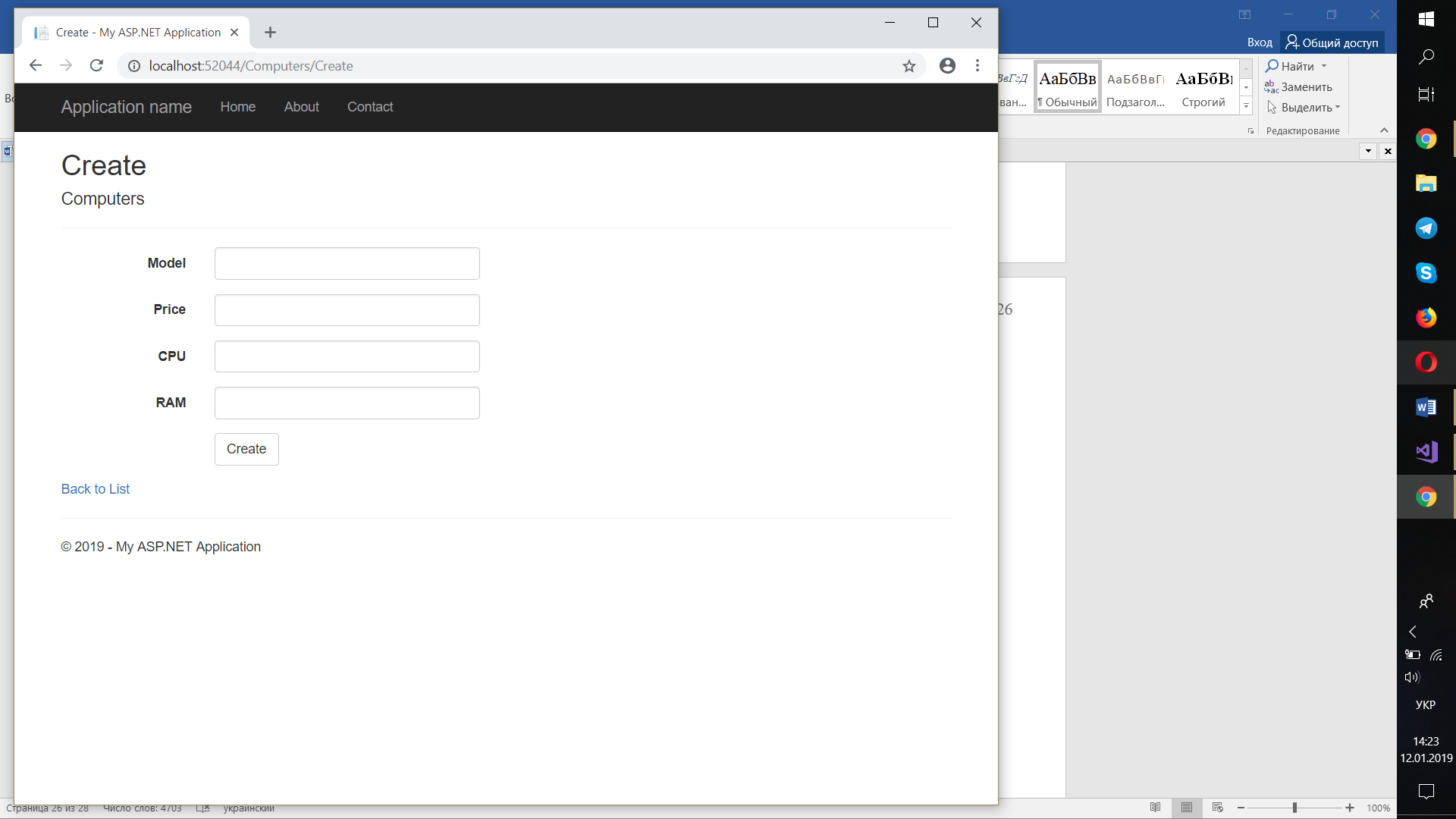


Рисунок 5 – Додавання нового елементу до бази даних

Наступним створюємо WPF проект, створюємо клієнтський додаток Windows, у якому є можливість додавання, видалення, оновлення та виклик Консольного додатку(рис. 6).

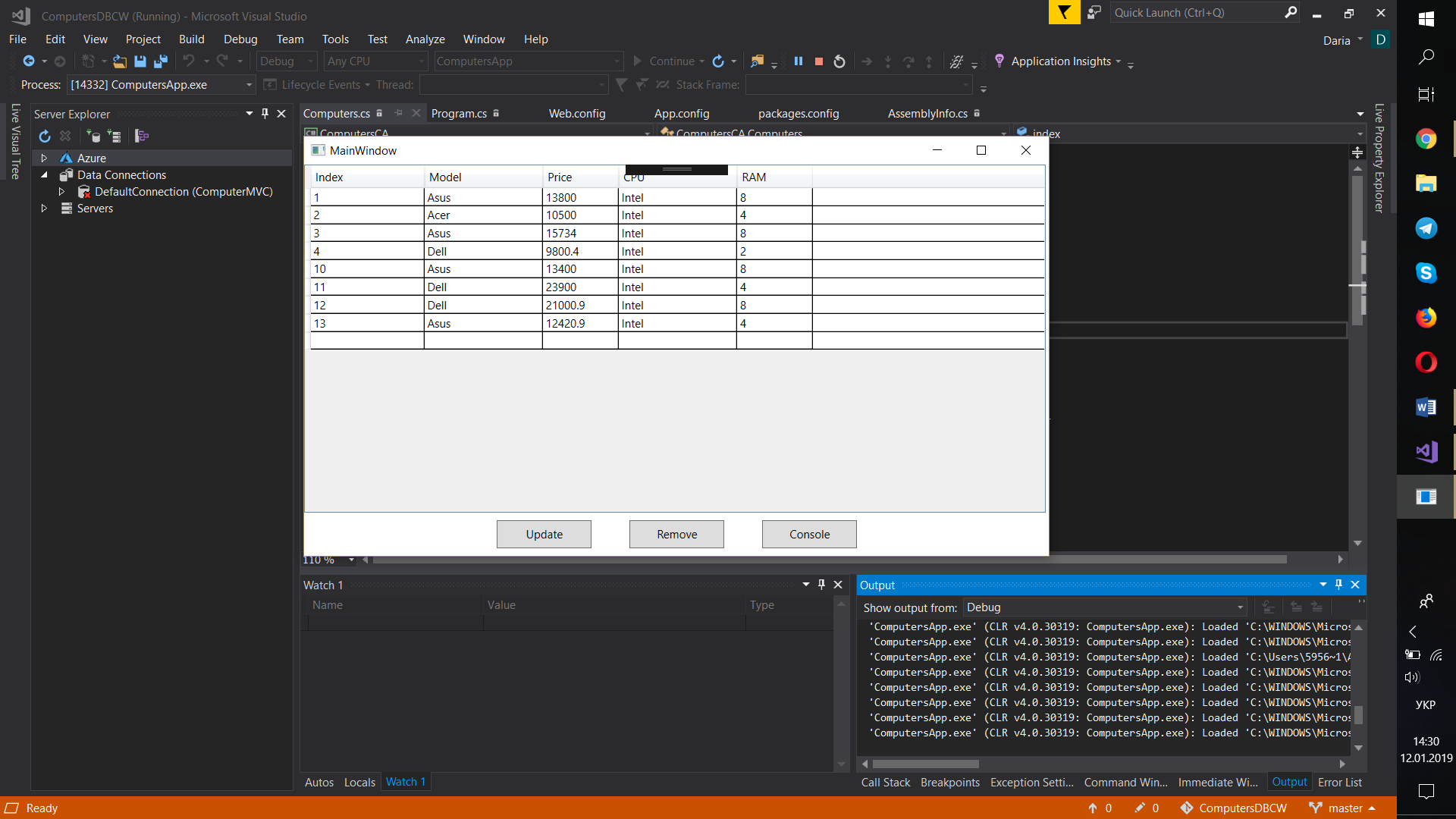


Рисунок 6 – Результат роботи WPF додатку

Через кнопку Console викличемо наш Console Application, у якому ми маємо можливість додавати, видаляти та редагувати базу даних (рис. 7).

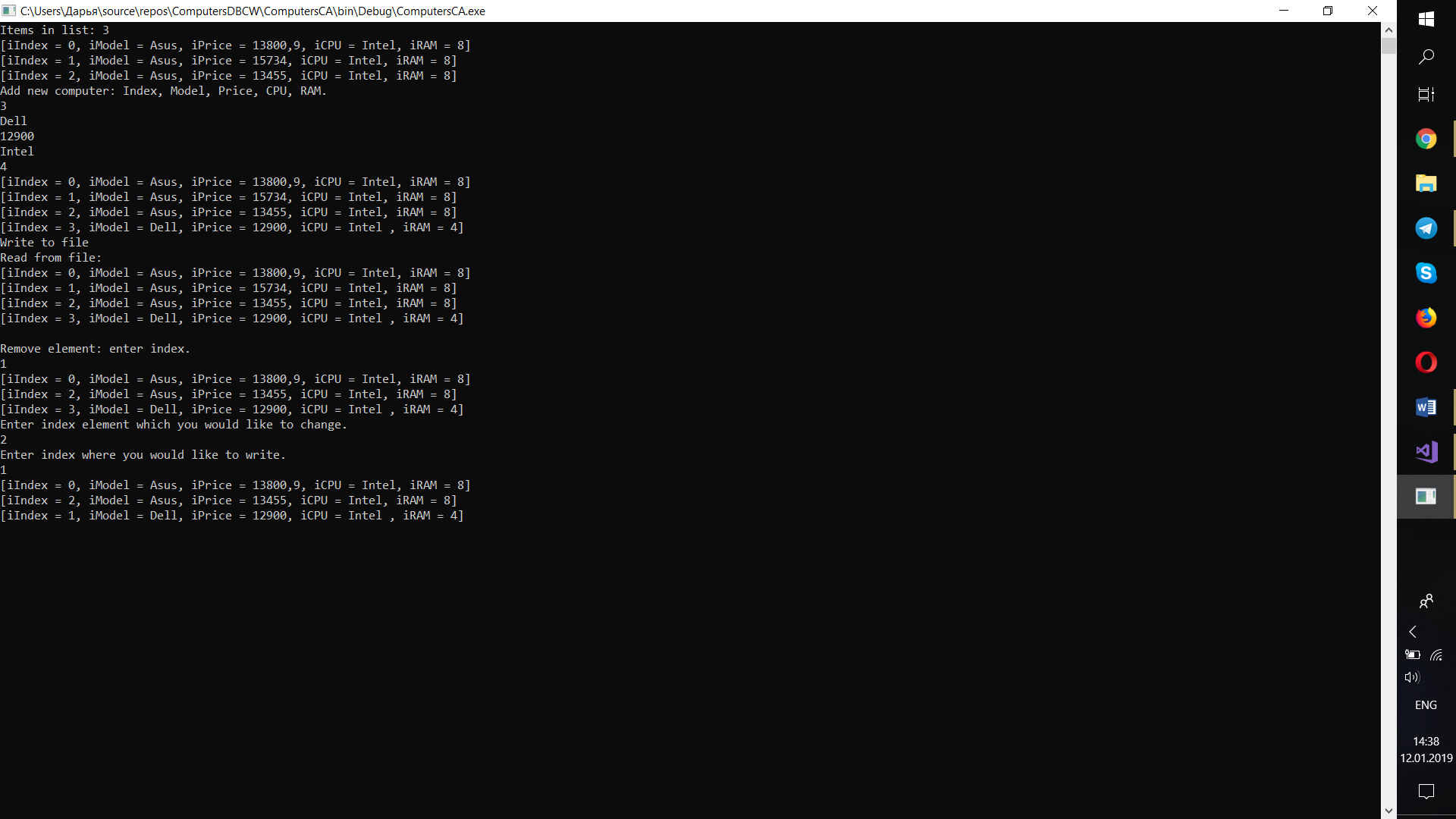


Рисунок 7 – Результат роботи проекту Console Application

# Висновки

В ході виконання роботи була розгорнута та налаштована система, що демонструє, як створити основний Web API проект у Visual Studio, та як написати основні операції CRUD за допомогою структури Entity.

Розроблений програмний комплекс є незалежним від апаратної реалізації обчислювальних засобів та операційної системи під управлінням якої він функціонує.

Архітектура програмного комплексу відповідає патерну проектування MVC, а модульна структура дозволяє додавати в міру необхідності додаткові компоненти.

# Перелік джерел інформації

1. Албахари Д. С# 6.0 Справочник. Полное описание языка - Вильямс, 2016.
2. Троелсен Э. Язык программирования на С# - Вильямс, 6 издание, 2013.
3. Рихтер Дж. Программирование на платформе Microsoft .NET Framework 4.0. на языке C# - СПб.: Питер, 2012.