МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

НКЦ заочної освіти

Кафедра інформаційних технологій

# Бакалаврська кваліфікаційна робота

на тему: Розробка інформаційно-пошукового додатку «Market»
за технологією WPF

Виконав студент 5 курсу групи КН-5
Напрям 6.050101 комп'ютерні науки
Марцинішин Олег Вікторович

Керівник к.геогр.н., доцент
Коваленко Людмила Борисівна

Консультант д.ф.-м.н., професор
Ковальчук Володимир Володимирович

Одеса 2020

# СПИСОК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

API − Application Programing Interface;

ASP.NET − Active Server Pages для .NET;

CIL − Common Intermediate Language;

CLI − Command Line Interface;

CLR − Common Language Runtime;

GUI − Graphical User Interface;

JSON − JavaScript Object Notation;

MVC – Model View Controller;

ORM − Object-Relational Mapping;

SQL Server − Structured Query Language Server;

T−SQL − Transact structured query language;

WCF − Windows Communication Foundation;

WPF − Windows Presentation Foundation;

XAML − eXtensible Application Markup Language.

ЗМІСТ

# ВСТУП

У нашому часі величезна кількість організацій використовує персональні комп'ютери для зберігання та обробки всіх видів інформації. Ці відомості містяться в базі даних. Бази даних відіграють важливу роль у великому технологічному світі, який розвивається. Все, з чим ми кожен день взаємодіємо в житті, за всіма відомостями, фіксуються в якійсь базі даних. Робота з базами даних є важливий навиком у роботі з комп’ютером, а спеціалісти даної області стають все більш затребувані. Основні ідеї найновіших інформаційних методів базуються на представленні, відповідно до такої інформації, яка повинна бути освіченою в базових даних із заданими показниками, що динамічно змінюються, і забезпечують задоволення всіх потреб в інформації користувачів. Бази даних формуються і працюють під управлінням спеціальних програмних середовищ що називаються системами управління базами даних.

Практично кожна компанія, яка працює в Інтернеті, використовує якусь базу даних для зберігання інформації. SQL-сервери використовуються як найбільш часто використовувані інструменти для розширення та підтримки базових даних.

Сьогодні сервери SQL використовують як корпоративні, так і малі підприємствами. Вони пропонують ефективні способи зберігання інформації та працюють досить довго, щоб проникнути практично у всі галузі

Стандартний користувальницький і прикладний програмний інтерфейс (API) реляційною базою даних є язик створених запитів (SQL). Оператори SQL використовуються для інтерактивних запитів інформації з реляційної бази даних, так і для збору даних для звіту. Для забезпечення точності та доступності реляційної бази даних необхідно дотримуватись конкретних правил цілостності даних.

Реляційні бази даних корисні для всіх типів і розмірів організацій, де пов'язані точки даних, повинні постійно управлятися і захищатися. Реляційна база даних − це найчастіше розподілена модель бази даних.

Метою даної роботи є розробка бази даних магазину «MARKET» за технологією WPF. Що автоматично синхронізує інформацію про товари.

Для виконання мети, необхідно вирішити наступні задачі:

– дослідити предметну область;

– спроектувати базу даних;

– створити форми для роботи з базою;

– організувати для користувача меню;

– дана програма повинна мати простий і зручний призначений для користувача інтерфейс;

Кваліфікаційна бакалаврська робота містить вступ, 3 розділа, висновки, перелік посилань, 26 рисунків, 1 додаток

# 1 ОСНОВНИЙ АНАЛІЗ ТЕХНОЛОГІЙ

## 1.1 Аналіз додатку WebApi для роботи з базою даних

У комп'ютерному програмуванні інтерфейс прикладного програмування (API, Application Programing Interface) являє собою набір визначень підпрограм, протоколів та інструментів для створення програмного забезпечення і додатків.

Простіше кажучи, API − це свого роду інтерфейс, який має набір функцій, які дозволяють програмістам отримувати доступ до певних функцій або даними додатка, операційної системи або інших служб.

Web API, як випливає з назви, являє собою API-інтерфейс в Інтернеті, доступ до якого можна отримати за протоколом HTTP. Це концепція, а не технологія. Можна створювати Web API з використанням різних технологій, таких як Java, .NET і т.д. Наприклад, API REST Twitter надають програмний доступ для читання і запису даних, використовуючи які можливо інтегруючи можливості Twitter у власний додаток.

Web API ASP.NET являє собою структуру, (рис. 1). яка розширюється для створення служб на основі HTTP, до яких можна звертатися в різних додатках на різних платформах, таких як Інтернет, Windows, мобільні пристрої і т.д.

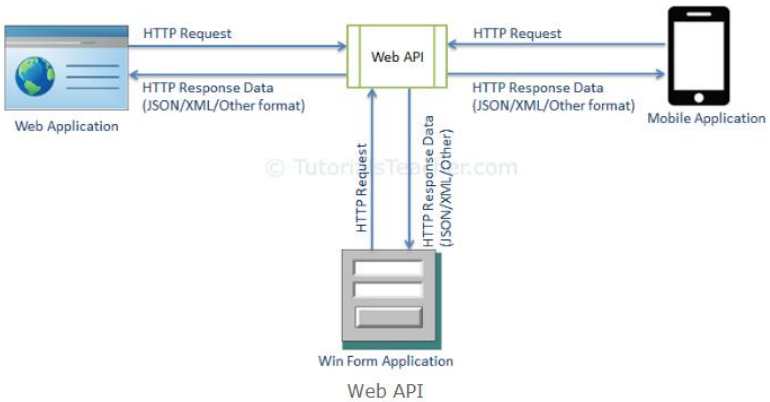



Рисунок 1 − Приклад структури Web-API

Він працює більш-менш так само, як веб-додаток ASP.NET MVC, за винятком що він відправляє дані в якості відповіді замість подання HTML. Це схоже на веб-сервіс або сервіс WCF, але виняток полягає в тому, що він підтримує тільки протокол HTTP.

### 1.1.1 Характеристики Web-API ASP.NET

Основними характеристиками Web-API ASP.NET полягає у наступному:

- є ідеальною платформою для створення сервісів RESTful;
- побудований поверх ASP.NET і підтримує конвеєр запитів / відповідей ASP.NET;
- відображає HTTP-дієслова на імена методів;
- підтримує різні формати даних відповідей, вбудована підтримка формату JSON, XML, BSON;
- може бути розміщений на IIS, самому хості або іншому веб-сервері, що підтримує .NET 4.0+;
- ASP.NET WebAPI Framework включає в себе новий HttpClient для зв'язку з сервером WebAPI (HttpClient може використовуватися на стороні сервера ASP.MVC, в додатку Windows Form, консольному додатку або інших додатках).

ASP.NET Web API − це структура для побудови HTTP-сервісів, які можуть використовуватися широким колом клієнтів, включаючи браузери, мобільні телефони, iphone і планшети. Він дуже схожий на ASP.NET MVC, оскільки містить функції MVC, такі як маршрутизація, контролери, результати дій, фільтр, прив'язки моделі, контейнер IOC або впровадження залежностей.

ASP.NET Web API є розширенням WCF REST API. Коротше кажучи, це заміна WCF REST API. Він може використовуватися з ASP.NET MVC і

іншими типами веб-додатків, такими як ASP.NET WebForms. Крім того, Web API можна використовувати в якості автономної програми веб-служб.

Сьогодні веб-додатку недостатньо для залучення клієнтів. Люди дуже розумні, вони використовують iphone, мобільні телефони, планшети, будь-які пристрої в повсякденному житті. Ці пристрої також мають безліч застосувань для полегшення життя. Насправді, весь рух від Інтернету до світу додатків.

Таким чином, якщо потрібно швидко і просто представити сервісні дані браузерам і всім цим сучасним додаткам для пристроїв, повинен бути API, сумісний з браузерами і всіма цими пристроями (рис. 2).

Наприклад, Twitter, Facebook і Google API для веб-додатків і додатків для телефонів.

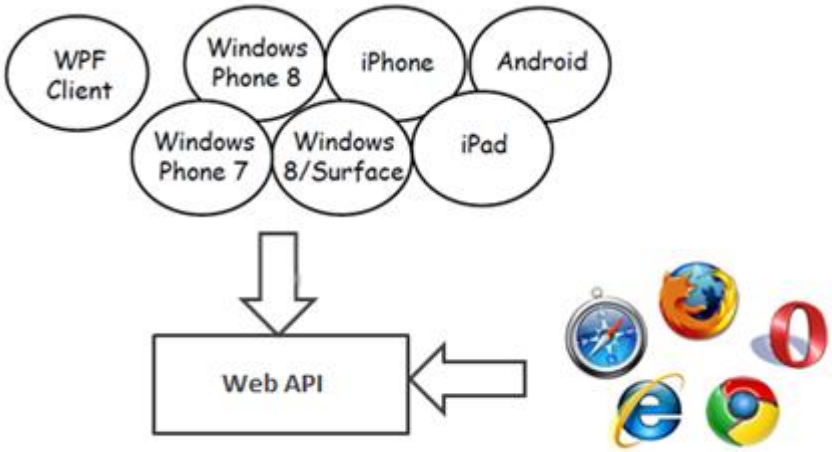



Рисунок 2 − Приклад сумісності додатків

Web API − це відмінна структура для надання ваших даних і послуг різних пристроїв. Більш того, Web API − це ідеальна платформа з відкритим вихідним кодом для створення REST-сервісів на платформі .NET Framework. На відміну від служби WCF Rest, вона використовує всі можливості HTTP (такі як URI, заголовки запитів / відповідей, кешування, управління версіями,

різні формати контенту), і вам не потрібно визначати будь-які додаткові параметри конфігурації для різних пристроїв, на відміну від служби WCF Rest.

### 1.1.2 Функції Web API

Web API підтримує засновані на угодах дії CRUD, так як він працює з HTTP-дієсловами GET, POST, PUT і DELETE.

Також підтримує декілька текстових форматів, таких як XML, JSON і т.д., або можна використовувати свій власний MediaTypeFormatter.

Може приймати і генерувати контент, який не може бути об'єктно-орієнтованим, наприклад зображення, файли PDF та інше.

Автоматична підтримка OData. Отже, помістивши новий атрибут [Queryable] в метод контролера, який повертає IQueryable, клієнти можуть використовувати цей метод для складання запиту OData.

Підтримує Self-хостинг або IIS хостинг.

Підтримує функції ASP.NET MVC, такі як маршрутизація, контролери, результати дій, фільтр, прив'язки моделей, контейнер IOC або впровадження залежностей.

Якщо потрібна веб-служба, то не потрібен SOAP, то ASP.NET Web API − кращий вибір.

Використовується для створення простих HTTP-служб без SOAP поверх існуючого конвеєра повідомлень WCF.

Проста настройка на відміну від служби WCF REST.

Просте створення сервісу з Web API. З WCF REST Services створення сервісу утруднено.

Заснований на HTTP і простий у визначенні, поданні і використанні в режимі REST.

Заснований на легкої архітектурі RESTful і хороший для пристроїв з обмеженою пропускною здатністю, таких як смартфони.

Відкритий вихідний код.

## 1.2 Аналіз десктопного додаток описаний на технології WPF

Сучасні сайти інтерактивні і динамічні, тобто вони реагують на дії користувача, обробляють його запити і видають результат. Так працюють онлайн-сервіси, наприклад, інтернет-банкінг або онлайн-кінотеатр. Для створення інтерактивних і динамічних сайтів зазвичай використовується архітектурний патерн MVC.

Model View Controller (MVC) є архітектурним шаблон, який відокремлює додаток на три основні логічні компоненти: модель, вид і контролер. Кожен з цих компонентів створений для обробки конкретних аспектів розвитку програми. MVC − це одна з найбільш часто використовуваних галузевих рамок веб-розробок для створення масштабованих та розширених проектів.

### 1.2.1 Опис модель MVC

Статична сторінка на HTML не вміє реагувати на дії користувача. Для двосторонньої взаємодії потрібні динамічні веб-сторінки. MVC − ключ до розуміння розробки динамічних веб-додатків, тому розробнику потрібно знати цю модель.

MVC розшифровується як модель − уявлення − контролер. Це спосіб організації коду, який передбачає виділення блоків, що відповідають за вирішення різних завдань. Один блок відповідає за дані додатки, інший відповідає за зовнішній вигляд, а третій контролює роботу додатка.

На рис. 3 наведені основні компоненти MVC.

Компонент Model відповідає всій логіці, пов'язаній з даними, з якою працює користувач. Це може представляти або дані, що передаються між компонентами Перегляду та Контролера, або будь-які інші дані, пов'язані з бізнес-логікою. Наприклад, об’єкт Клієнта буде отримувати інформацію про

клієнта з бази даних, маніпулювати нею та оновлювати її назад в базу даних або використовувати її для надання даних.

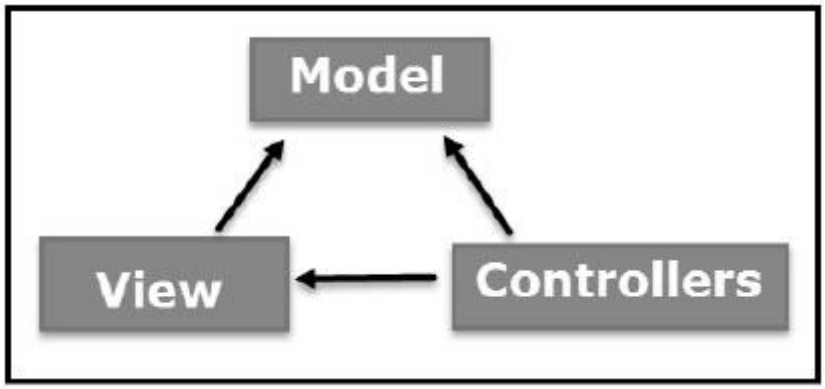



Рисунок 3 − Компоненти MVC

Компонент View використовується для всієї логіки інтерфейсу програми. Наприклад, представлення Замовника включає всі компоненти інтерфейсу користувача, такі як текстові поля, спадні місця тощо, з якими взаємодіє кінцевий користувач.

Контролери діють як інтерфейс між компонентами Model і View для обробки всіх бізнес логіки та вхідних запитів, маніпулювання даними за допомогою компонента Model та взаємодії з Views, щоб надати кінцевий результат. Наприклад, контролер Клієнта буде обробляти всі взаємодії та входи з Перегляду клієнтів та оновлювати базу даних, використовуючи Модель клієнта. Цей же контролер буде використовуватися для перегляду даних Клієнта. [1][1)]

### 1.2.2 Основні ідеї та принципи MVC

Основні ідеї та принципи MVC представлені далі:

[1)] [1] MVC Framework – Introduction. URL: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm (дата звернення 12.04.2020)

VC − це набір архітектурних ідей і принципів для побудови складних інформаційних систем з призначеним для користувача інтерфейсом;

MVC − це не патерн проектування. MVC − це саме набір архітектурних ідей і принципів для побудови складних систем з призначеним для користувача інтерфейсом. [2][1)]

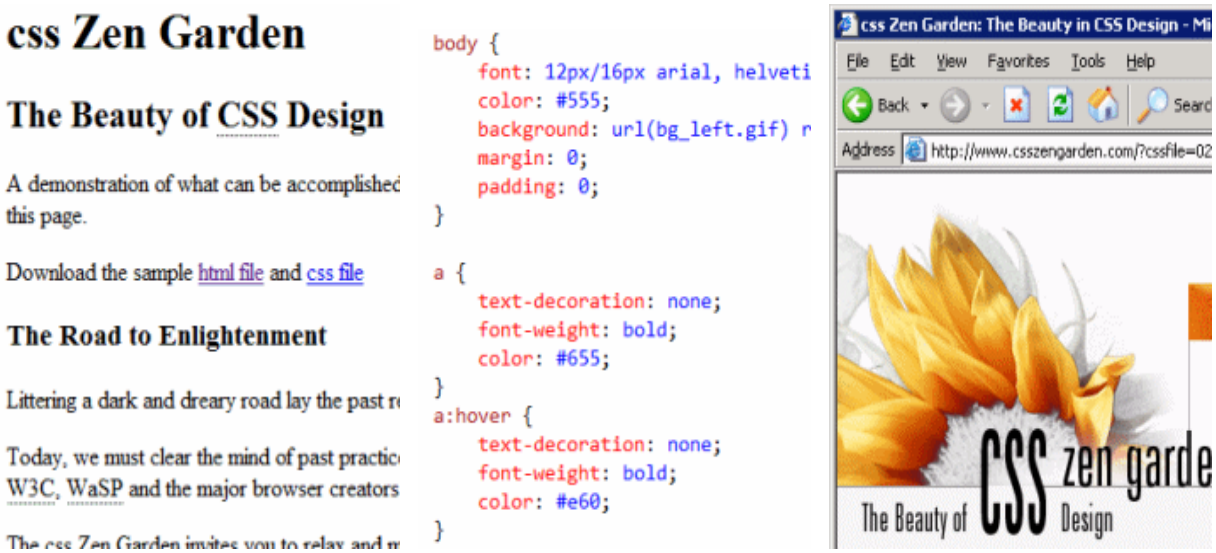
### 1.2.3 Основні моделі розробки ASP.NET MVC

ASP.NET підтримує три основні моделі розробки: веб-сторінки, веб-форми та MVC (рис. 4). ASP.NET MVC фреймворк − це легка, дуже перевірена презентаційна рамка, яка інтегрована з існуючими функціями ASP.NET, такими як головні сторінки, автентифікація тощо. У рамках .NET цей фреймворк визначений у зборі System.Web.MVC. Остання версія MVC Framework − 5.0. Використовують Visual Studio для створення програм ASP.NET MVC, які можна додати як шаблон у Visual Studio.

Особливості ASP.NET MVCASP.NET MVC надає такі функції:

- ідеально підходить для розробки складних, але не легких додатків;
  забезпечує розширюваний та підключається каркас, який можна легко замінити та налаштувати (наприклад, якщо не використовувати вбудований Razor або ASPX View Engine, дозволено використовувати будь-які інші двигуни сторонніх представників або навіть налаштувати існуючі);
- використовує дизайн додатку на основі компонентів, логічно розділяючи його на компоненти Model, View та Controller (це дає можливість розробникам керувати складністю масштабних проектів та працювати над окремими компонентами);

[1)] [2] MVC MODEL. URL: https://javarush.ru/groups/posts/2536−chastjh−7−znakomstvo−s−patternom−mvc−model−view−controller (дата звернення 12.04.2020)




а) б) в)

Рисунок 4 − Моделі розробки: а) веб-сторінки; б) веб-форми; в) MVC

- структура MVC покращує тестову розробку та довіреність програми, оскільки всі компоненти можуть бути розроблені на основі інтерфейсу та протестовані за допомогою макетних об'єктів (ASP.NET MVC Framework ідеально підходить для проектів із великою командою веб-розробників);
- підтримує всі існуючі широкі функції ASP.NET, такі як авторизація та автентифікація, основні сторінки, прив’язка даних, контроль користувачів, членство, маршрутизація ASP.NET тощо;
- не використовує поняття View State (яке присутнє в ASP.NET), це допомагає створювати додатки, які мають невелику вагу і дають повний контроль розробникам.

Таким чином, можна розглядати MVC Framework як основну основу, побудовану на базі ASP.NET, що забезпечує великий набір додаткових функціональних можливостей, орієнтованих на розробку та тестування на основі компонентів.

## 1.3 Аналіз фреймворку WPF

Windows Presentation Foundation (WPF) − це фреймворк, що створює клієнтські програми для настільних ПК. Платформа розробки WPF підтримує широкий набір функцій розробки додатків, включаючи модель програми, ресурси, елементи керування, графіку, макет, прив'язку даних, документи та безпеку. Рамка є частиною .NET, тому якщо ви раніше створювали програми з .NET за допомогою ASP.NET або Windows Forms, досвід програмування повинен бути знайомим. WPF використовує розширювану мову розмітки додатків (XAML) для надання декларативної моделі програмування додатків.

WPF, який розшифровується як фундація презентації Windows, − це останній підхід Microsoft до GUI-рамки, що використовується разом із .NET-рамкою.

GUI означає графічний інтерфейс користувача, і ви, мабуть, зараз переглядаєте один. У Windows є графічний інтерфейс для роботи з вашим комп’ютером, а у браузері, в якому, читаються документи, є графічний інтерфейс, який дозволяє переглядати Інтернет.

Основними компонентами WPF є PresentationFramework, PresentationCore, Milcore, Загальна мова виконання (CLR), User32, Kernel. Milcore пишеться некерованим кодом, щоб ввімкнути тісну інтеграцію з DirectX, який відповідає за показ. WPF чудово контролює пам'ять та виконання. Композиція в Milcore є надзвичайно чутливою до продуктивності і вимагає відмовитися від багатьох переваг загальної мови виконання для отримання продуктивності [3,4][1)].

Рамка GUI дозволяє створити додаток з широким спектром елементів графічного інтерфейсу, наприклад, міток, текстових полів та інших добре

[1)] [3] Введение в WPF. Особенности платформы WPF. URL: https://metanit.com/sharp/wpf/1.php (дата звернення 15.04.2020)

[4] Windows Presentation Foundation URL: https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation (дата звернення 15.04.2020)

відомих елементів. Без рамки GUI вам доведеться малювати ці елементи вручну та обробляти всі сценарії взаємодії з користувачем, такі як введення тексту та миші. Це багато роботи, тому натомість більшість розробників використовуватиме графічний інтерфейс GUI, який виконає всі основні роботи та дозволить розробникам зосередитися на створенні чудових додатків (рис. 5).

Рисунок 5 − Приклад стандартних елементів GUI

Тут є багато фреймворків графічного інтерфейсу, але для розробників .NET найцікавіші на даний момент − це WinForms та WPF. WPF − найновіший, але Microsoft все ще підтримує та підтримує WinForms. Як ви побачите в наступній главі, існує досить багато відмінностей між двома рамками, але їх мета однакова: полегшити створення додатків із чудовим графічним інтерфейсом.[5][1)]

## 1.4 Опис програмного забезпечення Net Framework

Net Framework − це розробка програмного забезпечення від Microsoft. Він забезпечує кероване середовище програмування, де програмне

[1)] [5] Wpf Tutorial. URL: https://www.com/about−wpf/what−is−wpf/ (дата звернення 16.04.2020)

забезпечення можна розробляти, встановлювати та виконувати в операційних системах на базі Windows.

Основні конструктивні особливості:

- інтероперабельність − дозволяє програмам, розробленим .NET, отримувати доступ до функціональних можливостей у програмах, розроблених за межами .NET.
- загальний двигун виконання (також відомий як загальна мова виконання) − це дозволяє програмам, розробленим у .NET, проявляти загальну поведінку при використанні пам'яті, обробці винятків та безпеці.
- мовна незалежність − загальні специфікації мовної інфраструктури (CLI) дозволяють обмінюватися типами даних між двома програмами, розробленими різними мовами.
- бібліотека базових класів: бібліотека коду для найбільш поширених функцій − використовується програмістами, щоб уникнути повторного переписування коду.
- простота розгортання: Існують інструменти для забезпечення простоти встановлення програм без втручання у встановлені раніше програми.
- безпека − програми, розроблені в .NET, базуються на загальній моделі безпеки.

.NET займає центральне місце в стратегії розвитку Microsoft, що займається надмірним архівуванням, і є конкуренцією організації Java. Це так важливо для розробки на платформах Windows, використання терміна залежить від контексту. Наприклад, звичайно просто говорити загалом про ".NET розробника" як програміста, який працює в середовищі розробки Microsoft З іншого боку, під час написання коду розробник посилається на те, з якою конкретною версією Framework працює − .NET 2.0, яка вийшла у

2005 році, значно відрізняється від .NET 4.0, який був поставлений у 2010 році. [6][1)]

### 1.4.1Основні компоненти .NET Framework

.NET Framework існувала майже 20 років і зазнала багатьох змін, і компоненти згорталися та згодом застаріли. На даний момент у .NET є три основні шари:

- стандартна бібліотека .NET включає компоненти, які формуватимуть інфраструктуру майже для будь-якої програми − класи та типи, які допомагають виконувати повсякденні завдання, такі як обробка струн та примітивів, створення підключень до бази даних, виконання вводу / виводу операції тощо.
- необов’язкові моделі додатків містять код сантехніки для різних платформ, на яких дозволено розгорнути свою програму .NET, існує ряд моделей додатків для додатків Windows, а також для інших платформ: наприклад, ASP.NET для веб-додатків, а також моделі для Mac і різних мобільні платформи.
- загальна інфраструктура є базовим шаром компонентів, які насправді дозволяють виконати всю систему на практиці, компіляторів для мов компонентів у час виконання (вони мають вирішальне значення для розуміння того, що може запропонувати .NET)

### 1.4.2 Опис роботи .NET Framework

Основні компоненти .NET Framework працюють разом, щоб полегшити процес написання додатків. Стандартні моделі бібліотеки та додатків

---

[1)] [6] .NET Framework. URL: https://ru.wikipedia.org/wiki/.NET_Framework (дата звернення 15.04.2020)

надають багато коду для вирішення основних завдань програмування, тому не доведеться ні чого винаходити за допомогою кожного створеного додатка. І загальна інфраструктура забезпечує значну частину роботи з розгортання цих програм.

Код, написаний будь-якою з мов .NET, збирається на мову проміжного байт-коду, яку називають загальною проміжною мовою, або CIL. Код CIL не є читабельним для людини, але його можна переносити через операційні системи та платформи. Потім CIL знову складається за допомогою загальної мови виконання або CLR. Реалізації CLR залежать від платформи, і вони компілюють код CIL у машиночитаний код, який може бути виконаний на платформі в даний момент. Різні версії CLR підтримують компіляції, що з’являються за часом та заздалегідь.

У процесі створення локального машиночитаного коду CLR також керує безліччю функціональних додатків низького рівня, таких як збирання сміття та нарізка, що є вирішальним для продуктивності додатків, але часто стомлює розробників. Спільно CIL та CLR складають загальну мовну інфраструктуру .NET.

Все це знайоме для тих, хто працював з платформою Java, оскільки це дотримується тієї ж основної парадигми − великі доступні бібліотеки класів, посередницький байт-код і специфічний для платформи час роботи, який автоматизує управління пам'яттю, − це всі функції обох пропозицій. .NET був розроблений в кінці 90-х, протягом початкового розквіту Java, і спочатку позиціонувався як конкурент платформи Java Enterprise Edition, мова Java і C#, перша і найвизначніша .NET мова, походять від C і семантично схожі.

C# оголошений після запуску .NET у 2000 році, є найвідомішою і широко використовуваною мовою програмування .NET. Вона була розроблена корпорацією Microsoft як частина ініціативи .NET, і більшість класів у стандартній бібліотеці .NET написані на C#. Мова об'єктно-орієнтована і розроблена таким чином, щоб вона була досить

схожою на C, щоб було легко для розробників C, C ++, Java та JavaScript для швидкого вивчення та використання.

В даний час Microsoft також випереджає дві інші мови програмування, які можна використовувати для запису для .NET Framework. Один − F#, функціональна мова програмування, яка є частиною сімейства мов ML, яка в кінцевому рахунку має коріння в LISP; інша − Visual Basic, поважна, легка для засвоєння мова програмування Microsoft для розробки програм клієнт-сервер. Але це лише вершина айсберга: Оскільки .NET складається з відкритих стандартів, кожен може написати мову, що компілюється в байт-код CIL і може бути виконаний CLR. У Вікіпедії є список більш ніж 20 мовних проектів CLI, які зараз підтримуються. Майже всі вони представляють .NET-порти існуючих мов, від Pascal до JavaScript і навіть COBOL.

Плюси і мінуси .Net. З позитивного боку .NET пропонує об'єктно-орієнтовану модель програмування з надійною та простою системою кешування та зрілим IDE, а також дозволяє гнучко розгорнути та просте обслуговування. Крім того, кросплатформенний характер .NET дозволяє перенести код у різні види кінцевих точок. .NET є найбільш підходящим, якщо будувати міжплатформенні програми на інфраструктурах корпоративного масштабу, для того щоб мати можливість масштабувати без необхідності повністю переоснащуватись.[7] [1)]

## 1.5 Опис Dapper

Dapper − це мікро-ORM або це проста рамка картографічних об'єктів, яка допомагає зіставити вихідний запит на доменний клас або клас C#. Це високоефективна система доступу до даних, побудована командою StackOverflow та випущена як відкритий код. Якщо ваш проект надає

[1)] [7] What is the .NET Framework (.NET). URL: https://www.techopedia.com/definition/3734/net−framework−net (Дата звернення 20.04.2020)

перевагу написанню збережених процедур або написанню нативного запиту замість використання повноцінних інструментів ORM, таких як EntityFramework або NHibernate, тоді Dapper − це очевидний вибір для вас. За допомогою Dapper дуже просто запустити SQL-запит на базу даних і отримати результат, відображений на C# доменний клас.

Наприклад, наведений нижче код отримує об’єкт Клієнта, виконуючи запит до таблиці "Клієнти", використовуючи рамку Dapper.

```
Customer cust = _db.Query <Customer> ("виберіть * з клієнтів, де CustomerId = @ CustomerId", новий {CustomerId = id}). FirstOrDefault ();
```

Для демонстрації необхідно створити новий проект Asp.Net MVC 5.0 у Visual Studio 2015. Потім необхідно, завантажити в проект пакет Dapper Nuget. Для цього потрібно натиснути правою кнопкою миші на проект у провіднику рішень та оброти "Управління Nuget Packages ..". щоб отримати вікна Nuget. Ввести Dapper та увійти у вкладку Огляд та натисніть кнопку Встановити, для встановлення Dapper.

Давайте скористаємося простою моделю службовців департаменту (рис. 6), як показано нижче, і скористаємося Dapper, щоб виконати деякі найбільш часто використовувані запити доступу до даних.

Dapper Framework фактично розширює інтерфейс IDbConnection, наявний у просторі імен System.Data. Він має багато методів розширення для доступу до даних та відображення результату до типу C# (об’єкти домену), визначеного під класом SqlMapper, знайденим у просторі імен Dapper. Отже, щоб використовувати Dapper, спочатку нам потрібно оголосити об’єкт IDbConnection та ініціалізувати його до SqlConnection для підключення бази даних. Наведений нижче клас EmployeeController оголошує IDbConnection _db з підключенням до бази даних за допомогою рядка з'єднання AppSettings.

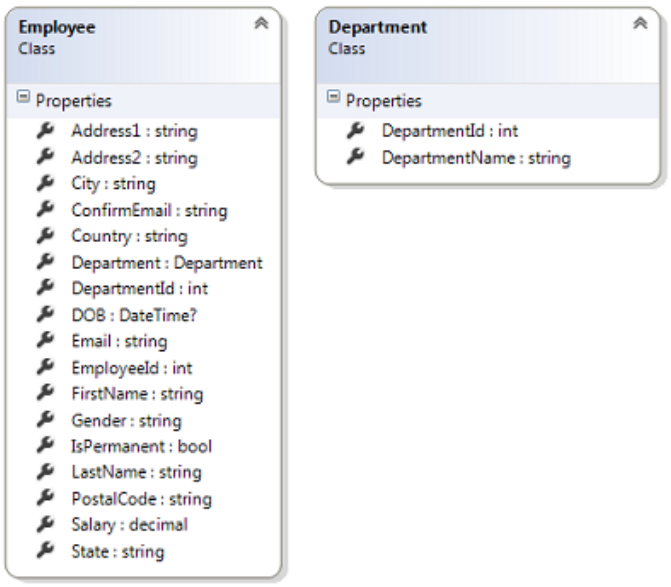



Рисунок 6 – Приклад простої моделі

```
public class EmployeeController: Controller
{ private IDbConnection _db = new SqlConnection
(ConfigurationManager.ConnectionStrings ["DBModel"]. ConnectionString);
```

Включіть System.Data, System.Data.SqlClient і Dapper простір імен.

Ви можете побачити всі методи доступу до даних під час введення _db. у способі дії. На рис.7 можна побачити швидкий список доступних методів.

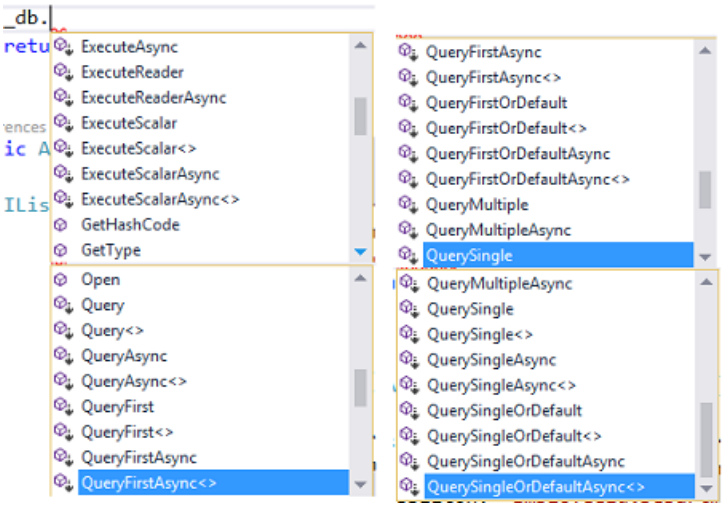



Рисунок 7 – Приклад простої моделі

### 1.5.1 Приклад виконання збереженої процедури

У наведеному нижче коді використовується метод QueryFirstOrDefault() для виконання збереженої процедури та повернення об'єкта Employee.

```
Employee emp = _db.QueryFirstOrDefault <Еслуеее> ("EmployeeGet", новий {EmployeeId = id}, commandType: CommandType.StoredProcedure);
```

```
СТВОРИТИ ПРОЦЕДУРУ [dbo]. [EmployeeGet]
    @E EmployeeeId INT
```

Крім того, ви також можете виконати запит безпосередньо, використовуючи метод Query (), як показано нижче.

```
Employee emp = _db.Query <Еслуеее> ("виберіть * з співробітників, де EmployeeId = @ EmployeeId", новий {EmployeeId = id}). FirstOrDefault ();
```

Для виконання збереженої процедури з запитами DML можна використовувати Execute() метод. Наведений нижче код виконує збережену процедуру EmployeeUpdate для оновлення рядка таблиці Employee.

```
_db.Execute ("EmployeeUpdate",
                        новий {FirstName = empModel.FirstName,
                            LastName = empModel.LastName,
                            DepartmentId = empModel.DepartmentId,
                            Адреса1 = empModel.Address1,
                            Адреса2 = empModel.Address2,
                            Місто = empModel.City,
                            State = empModel.State,
                            Країна = empModel.Country,
                            PostalCode = empModel.PostalCode,
                            Email = empModel.Email,
                            ConfirmEmail = empModel.ConfirmEmail,
                            DOB = empModel.DOB,
```

```
                    Зарплата = empModel.Sallar,
                    IsPermanent = empModel.IsPer стала,
                    Стать = empModel.Gender,
                    EmployeeId = emp.EposleeeId
                }, commandType: CommandType.StoredProcedure);
```

SP для працівника: СТВОРИТИ ПРОЦЕДУРУ [dbo]. [EmployeeUpdate]:

```
       @E EmployeeeId INT,
    @FirstName VARCHAR (50),
    @LastName VARCHAR (50),
    @DepartmentId INT,
    @ Address1 VARCHAR (50),
    @ Address2 VARCHAR (50),
    @City VARCHAR (50) ),
    @State VARCHAR (50),
    @Country VARCHAR (50),
    @PostalCode VARCHAR (50),
    @Email VARCHAR (50),
    @ConfirmEmail VARCHAR (50),
    @DOB DATETIME,
    @Gender VARCHAR (50),
    @IsPermanent BIT,
    @ SALAR DECIMAL (9,2)
ЯК
    ОНОВЛЕННЯ Співробітники
    SET FirstName = @FirstName,
        LastName = @LastName,
        DepartmentID = @DepartmentId,
        Адреса1 = @ Адреса1,
        Адрес2 = @ Адрес2,
        Місто = @City,
        State = @State,
        Країна = @country,
        PostalCode = @PostalCode,
        Email = @email,
        Confirmemail = @ConfirmEmail,
        DOB = @DOB,
        Стать = @Gender,
        IsPermanent = @IsPermanent,
        Заробітна плата = @Salary
    WHERE EmployeeId = @E zaposleeeId
```

### 1.5.2Виконання збереженої процедури для повернення складного об'єкта

Іноді у нас буде вимога приєднатися до декількох таблиць і повернути результат, який може не відображатися безпосередньо в існуючому об’єкті

домену. Наприклад, розглянемо нижче збережену процедуру, яка приєднує таблицю Співробітників та Департаменту та повертає деякі довільні стовпці.

СТВОРИТИ ПРОЦЕДУРУ [dbo]. [GetE EmployeesSummary] AS

```
SELECT
EmployeeId,
FirstName + '' + LastName As Name,
d.DepartmentId as DepartmentId,
d.DepartmentName as DepartmentName OF OF
Employees e INNER JOIN Відділи d
ON e.DepartmentId = d.DepartmentId
```

Щоб отримати результат цієї збереженої процедури, треба визначити клас C#, який відповідає результату, і який може мати інший об'єкт як член. Наведений нижче клас EmployeeSummary має відділ як дочірній об'єкт, який можна використовувати для отримання результату вищезгаданої процедури.

Публічний клас EmployeeSummary

```
{
    public int EmployeeId {get; набір; }
    ім'я публічного рядка {get; набір; }

    відділ громадського управління {get; набір; }
}
```

Щоб отримати результат збереженої процедури як об'єкта EmployeeSummary, дозволяється використовувати перевантаження нижче Запит <TFirst, TSecond, TReturn> (), як показано нижче[8][1].

```
     public ActionResult EmployeeSummary ()
{
    IList <ЕслуеeSummary> empSummary = _db.Query <EmployeeSummary,
```

[1)] [8] What is Dapper? How to use Dapper? In aspnet MVC. URL: http://www.codedigest.com/quick−start/17/what−is−dapper−how−to−use−dapper−in−aspnet−mvc (дата звернення 20.04.2020)

```
Department, EmployeeSummary> ("[GetE EmployeeSummary]",
        (emp, dept) => {emp.Department = dept; return emp;},
        splitOn: "EmployeeII, DepartmentId"). ToList ();
```

вигляд повернення (empSummary);

}

# 2 АНАЛІЗ РЕАЛІЦІЙНИХ БАЗ ДАНИХ

## 2.1 Опис системи MS SQL Server

Microsoft SQL Server − одна з інноваційних технологій, яка зробила революцію в тому, як бізнес обробляє дані. У будь-який момент у вашому бізнесі з’являться дані про постачальників, службовців, замовників та інших зацікавлених сторін. Важливо, щоб інформація була легкодоступною, але залишалася безпечною від несанкціонованого доступу. Microsoft SQL Server розроблений, щоб допомогти вашим підприємствам досягти цих цілей. Однак використання інтелектуального додатку також вимагає, щоб ви працювали з сертифікованим фахівцем, щоб переконатися, що він працює цілодобово.

SQL Server − це система управління реляційними базами даних від Microsoft. Система розроблена та побудована для управління та зберігання інформації. Система підтримує різні операції бізнес-розвідки, аналітичні операції та обробку транзакцій. Інформація, що зберігається на сервері, зберігається у реляційній базі даних. Однак, оскільки система набагато більше, ніж база даних, вона також складається з системи управління. SQL розшифровується як "Структурована мова запитів" − це комп'ютерна мова, яка управляє сервером та адмініструє його. Існує багато версій сервера SQL, кожна наступна версія − це вдосконалена модель свого попередника.

Microsoft SQL Server має численні програми у світі бізнесу. Перший і найбільш очевидний − це база даних, яка використовується для зберігання та управління інформацією. Однак підприємства, які зберігають конфіденційну інформацію про клієнтів, таку як особисті дані, дані кредитних карток та іншу конфіденційну інформацію, скористаються підвищеною безпекою. Система також дозволяє обмінюватися файлами даних комп'ютерами в одній мережі, що є фактором, що підвищує надійність.

Сервер SQL також використовується для збільшення швидкості, з якою обробляються дані, що дозволяє виконувати великі операції з легкістю. З інформацією, що зберігається в базі даних, підприємства матимуть надійну систему резервного копіювання.

Коли системи не функціонують, як очікувалося, це призведе до простоїв та втрати доходу. Це ж стосується і SQL-сервера будь-якої організації. Лише завдяки забезпеченню ефективної роботи реляційної бази даних зможе користуватися перевагами технології.[9][1)]

## 2.2 Опис T-SQL

T-SQL (Transact-SQL) − це набір розширень програмування від Sybase та Microsoft, які додають кілька функцій до мови структурованих запитів (SQL), включаючи управління транзакціями, обробку винятків та помилок, обробку рядків та оголошені змінні.

Усі програми, які спілкуються з SQL_Server, роблять це, надсилаючи T-SQL заяви на сервер. Запити T-SQL включають оператор SELECT, вибір стовпців, маркування вихідних стовпців, обмеження рядків та зміну умови пошуку.

Тим часом ідентифікатори T−SQL використовуються у всіх базах даних, серверах та об'єктах баз даних у SQL Server. Сюди входять наступні таблиці, обмеження, збережені процедури, представлення даних, стовпці та типи даних.

Кожен T-SQL-ідентифікатор повинен мати унікальне ім'я, присвоюватися під час створення об'єкта та використовуватися для ідентифікації об'єкта.[10][2)]

---

1 ) [9] What is Microsoft SQL Server and what is it used for. URL: https://www.infotectraining.com/blog/what−is−microsoft−sql−server−and−what−is−it−used−for. (дата звернення 25.04.2020)

2 ) [10] Transact SQL. URL: https://ru.wikipedia.org/wiki/Transact−SQL (дата звернення 25.04.2020)

### 2.2.1 Приклади операторів T-SQL

Найпопулярніший оператор T-SQL − це збережена процедура, яка є компільованим і збереженим кодом T-SQL. Подібно до представлень, збережені процедури генерують план виконання при першому виклику. Різниця полягає в тому, що зберігаються процедури можуть вибирати дані та виконувати будь-який код T−SQL в межах будь-яких параметрів.

Визначені користувачем функції − це ще один приклад тверджень T-SQL. Зазначені користувачем функції приймають вхідні параметри, виконують дію та повертають результати на виклик.

Іншим прикладом є тригер, який є збереженим сценарієм T-SQL, який запускається, коли оператор, відмінний від SELECT, видається проти таблиці або представлення. Два поширених тригера − ПІСЛЯ тригерів та ВСТАНОВИТИ тригери.

Програмування операторів T-SQL дозволяє IT-профі будувати додатки, що містяться в SQL Server. Ці програми − або об'єкти − можуть вставляти, оновлювати, видаляти чи читати дані, що зберігаються в базі даних.

Інтеграція загальної мови (CLR) є завершальним прикладом оператора T-SQL. Починаючи з SQL Server 2005, IT-профі можуть інтегруватися з .NET Framework CLR. Це дозволяє використовувати мови програмування .NET в об'єктах SQL Server для створення збережених процедур, визначених користувачем функцій і тригерів.

### 2.2.2.Функції T-SQL

Окрім вбудованих функцій SQL Server, користувачі можуть визначати функції за допомогою T-SQL.

Типи функцій T-SQL включають:

- функції сукупності, які працюють на колекції значень, але повертають одне підсумкове значення;

- функції ранжування, які повертають значення ранжування для кожного рядка в розділі;
- функції Rowset, які повертають об'єкт, який може бути використаний як посилання на таблицю в операторах SQL;
- скалярні функції, які працюють на одне значення і повертають єдине значення.

SQL Server також підтримує аналітичні функції в T-SQL для зображення складних аналітичних завдань. Ці аналітичні функції дозволяють IT-профі проводити загальний аналіз, такий як ранжування, процентні пункти, ковзаючи середні та сукупні суми, що виражаються в одному операторі SQL.

### 2.2.3Різниця між T-SQL і SQL

Існує три чіткі відмінності між ними:

- хоча T-SQL є розширенням до SQL, SQL є мовою програмування.
- T-SQL містить процедурне програмування та локальну змінну, тоді як SQL − ні.
- T-SQL є власником, а SQL − відкритим форматом.

Приєднання до T-SQL − це пропозиції, які використовуються для об'єднання рядків з двох або більше таблиць на основі відповідного стовпця між ними. Приєднання визначають, як SQL повинен використовувати дані з однієї таблиці для вибору рядків в іншій таблиці. Кілька операторів − такі як =, <,>, <>, <=,> =,! =, МІЖ, АЛЕ і НЕ − можуть використовуватися для приєднання таблиць.

У T-SQL доступні різні типи об'єднань. Вони включають, наприклад, внутрішні і зовнішні з'єднання. Внутрішнє з'єднання, яке повертає рядки, коли є збіг в обох таблицях, може бути вказане або в пунктах FROM або WHERE. Зовнішнє з'єднання, яке можна вказати лише в пункті FROM, знаходить і повертає відповідні дані та деякі різні дані з таблиць.

## 2.3 Аналіз JSON

JSON − це короткий термін для JavaScript Object Notation і є способом організованого, легкого доступу до інформації. Коротше кажучи, це дає читабельний для людини збір даних, до яких можна отримати доступ дійсно логічно.

### 2.3.1 Зберігання даних JSON

Як простий приклад, інформація про мене може бути записана в JSON наступним чином:

```
var jason = {
 "вік" : "24" ,
 "рідне місто" : "Міссула, штат Техас" ,
 "стать" : "чоловік"
} ;
```

Це створює об'єкт, до якого ми отримуємо доступ за допомогою змінної JASON. Вкладаючи значення змінної у фігурні дужки, ми вказуємо, що значення є об'єктом. Всередині об’єкта ми можемо оголосити будь-яку кількість властивостей за допомогою "name": "value"пари, розділених комами. Для доступу до інформації, що зберігається JASON, треба просто вказати назву потрібного нам ресурсу. Наприклад, щоб отримати доступ до інформації про мене, ми могли використовувати такі фрагменти:

```
документ . писати ( "Джейсон – ясон. вік ); // Вихід: Джейсону 24 роки
документ . написати ( "Джейсон – ' ясон. стать ); // Вихід: Джейсон – чоловік
```

### 2.3.2 Зберігання даних JSON в масивах

Трохи складніший приклад передбачає зберігання двох людей в одній змінній. Для цього треба скласти кілька об'єктів у квадратні дужки, що означає масив. Наприклад, якщо потрібно було включити інформацію про себе та мого брата в одну змінну, можна використовувати наступне:

```
var family = [ {
   "ім'я" : "Джейсон" ,
   "вік" : "24" ,
   "стать" : "чоловік"
} ,
{
   "ім'я" : "Кайл" ,
   "вік" : "21" ,
   "стать" : "чоловік"
} ] ;
```

Для доступу до цієї інформації нам потрібно отримати доступ до індексу масиву особи, до якої ми хочемо отримати доступ. Наприклад, ми використовуємо наступний фрагмент для доступу до інформації, що зберігається у family:

```
документ . писати ( родина [ 1 ] . ім’я ) ;  // Вихід: Кайл
документ . писати ( родина [ 0 ] . вік ) ;  // Вихід: 24
```

ПРИМІТКА**:** Це вигідно, якщо потрібно буде провести цикл через збережену інформацію, оскільки вона піддається циклу *for* з автоматично зростаючим значенням.

2.3.3 Введення даних JSON

Іншим способом зберігання кількох людей у одній змінній буде гніздування об’єктів**.** Для цього потрібно написати наступне:

```
var family = {
   "жасон" : {
      "ім'я" : "Джейсон Ленгсторф" ,
      "вік" : "24" ,
      "стать" : "чоловік"
   } ,
   "кайл" : {
      "ім'я" : "Кайл Ленгсторф" ,
      "вік" : "21" ,
```

```
        "стать" : "чоловік"
    }
}
```

Доступ до інформації в вкладених об’єктах трохи простіше зрозуміти; для доступу до інформації в об’єкті ми використовували б такий фрагмент:

```
документ . писати ( родина . язон . ім’я ) ;  // Вихід: Джейсон Ленгсторф
документ . писати ( родина . кайл . вік ) ;  // Вихід: 21
документ . писати ( родина . ясон . стать ) ;  // Вихід: чоловічий
```

Вкладені JSON та масиви можна комбінувати за необхідності для зберігання стільки даних, скільки необхідно.

З підйомом сайтів, що працюють на AJAX, стає все більш важливим для сайтів можливість завантажувати дані швидко та асинхронно, або у фоновому режимі, не затримуючи візуалізацію сторінки. Зміна вмісту певного елемента в макетах, не потребуючи оновлення сторінки, додає фактор “вау” до програм, не кажучи вже про додаткову зручність для користувачів. Через популярність та простоту соціальних медіа багато сайтів покладаються на вміст, який надають сайти, такі як Twitter, Flickr та інші. Ці сайти надають RSS-канали, які легко імпортувати та використовувати на стороні сервера, але якщо спробувати завантажити їх на AJAX, наштовхуємося на стіну: можна завантажити RSS-канал лише тоді, коли вимагає його від сервера той самий домен, на якому він розміщений. Спроба завантажити RSS-канал  облікового запису Flickr$.ajax() методом jQuery призводить до такої помилки JavaScript:

```
[ Виняток . . .  «Доступ до обмежених URI відмовлено»  код : «1012»
nsresult : "0x805303f4 (NS_ERROR_DOM_BAD_URI)"
розташування                                                    :
"http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js
Рядок: 19" ]
```

JSON дозволяє подолати проблему між домену, оскільки можна використовувати метод під назвою JSONP, який використовує функцію зворотного виклику для повернення даних JSON назад у домен. Саме ця здатність робить JSON настільки неймовірно корисною, оскільки вона відкриває безліч дверей, з якими раніше важко було обійтися.

Один з найпростіших способів завантаження даних JSON у веб-програми – це використання $.ajax() методу, доступного в бібліотеці jQuery. Легкість отримання даних буде залежати від сайту, який надає дані, але простий приклад може виглядати так:

```
$ . ajax (
    тип : "GET" ,
    URL : "http://example.com/users/feeds/" ,
    дані : "format = json & id = 123" ,
    успіх : функція ( канал )  {
        документ . писати ( подавати ) ;
    } ,
    dataType : 'jsonp'
) ;
```

Цей приклад вимагає отримання останніх новин у форматі JSON та виведення їх у браузер. Очевидно, не хочеться виводити необроблені дані JSON у браузер, але цей приклад показує основи завантаження JSON із зовнішнього джерела.[11][1)]

Незважаючи на те, що JSON є відносно стислим, гнучким форматом даних, з яким легко працювати у багатьох мовах програмування, є деякі недоліки у форматі. Ось п'ять основних обмежень:

[1)] [11] JSON what it is? How it works? How to use it? URL: https://www.copterlabs.com/json−what−it−is−how−it−works−how−to−use−it/ (дата звернення 25.04.2020)

– без схеми (з одного боку, це означає, що ви маєте повну гнучкість представляти дані будь-яким способом; з іншого боку, це означає, що ви могли випадково створити неправильні дані дуже легко);
– тільки один тип номера: формат подвійної точності з плаваючою комою IEEE-754 (це досить приголомшливо, але це просто означає, що ви не можете скористатися різноманітними і нюансованими типами номерів, доступними у багатьох мовах програмування);
– Немає типу дати (цей пропуск означає, що розробники повинні вдаватися до використання строкових уявлень дат, що призводять до форматування розбіжностей, або повинні представляти дати у вигляді мілісекунд з епохи);
– без коментарів. (це унеможливлює коментування полів в рядку, що вимагає додаткової документації та збільшує ймовірність непорозуміння);
– багатослівність (хоча JSON менш багатослівний, ніж XML, це не самий стислий формат обміну даними; для послуг з великим обсягом або спеціального призначення вам потрібно використовувати більш ефективні формати даних).[12][1)]

[1)] [12] What is JSON a better format for data exchange. URL: https: //www.infoworld. com/article/3222851/what−is−json−a−better−format−for−data− exchange.html (дата звернення 30.04.2020)

# 3 ОПИС СТРУКТУРИ “MARKET”

## 3.1 Опис архітектури бази даних

Структура бази даних представлена на рис. 8

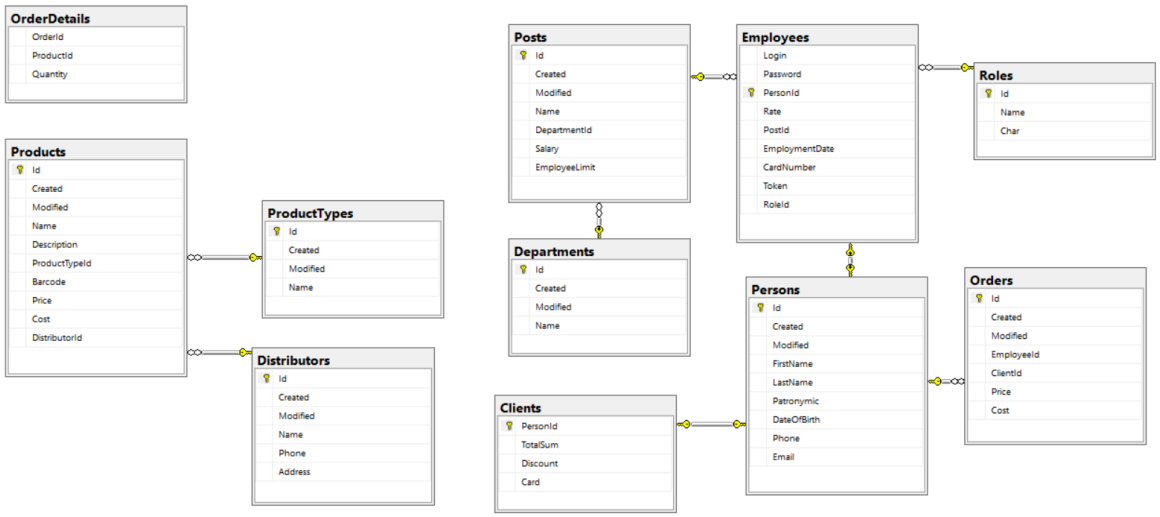



Рисунок 8 – База даних

Архитектура бази даних представлена у вигляді такої структури:

- типи продуктів;
- продукти;
- клієнти;
- робітники;
- посади;
- відділи;
- персональна інформація;
- ролі;
- закази;
- інформація о заказах;
- поставщики.

Кожна з таблиць представляє собою набір об’єктів які використовуються в програмі. Існуюча БД буде використовуватися клієнтом для мережі магазинів. Використана програма дозволяє будувати звітність для всієї мережі магазинів, яка використовує переваги розподіленої мережі. Завдяки можливості розширення поточного функціоналу можна буде додавати нові сутності і можливості для кожного з користувачів.

## 3.1 Опис програми “MARKET”

Десктопна програм повністю відповідає стандартам для роботи з обліковими даними для працівників і можливістю створення замовлень для касирів. Адміністратори можуть вносити дані про нових співробітників, додавати нових постачальників і змінити дані про продукти. Модератори також можуть виробляти дії для відкату замовлень і створювати звітні записи про продажі. А тепер докладніше о програмі.

На рис. 9 представлений вхід в систему.

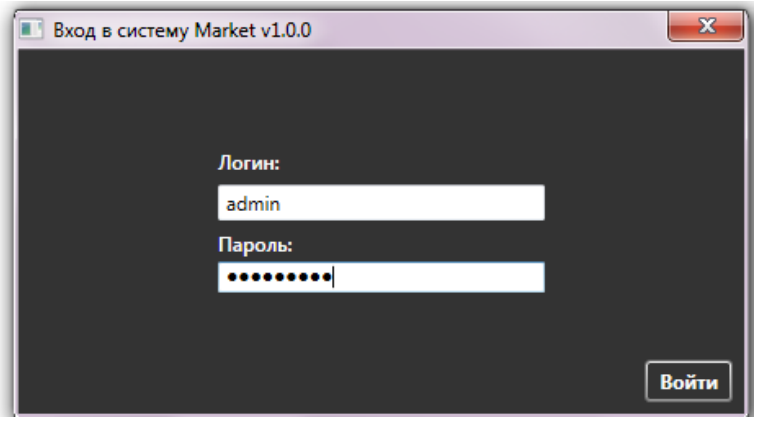



Рисунок 9 − Головне вікно програми

На рис. 10 зображена основна вкладка «Файл», в якій проходить весь процес збору інформації.

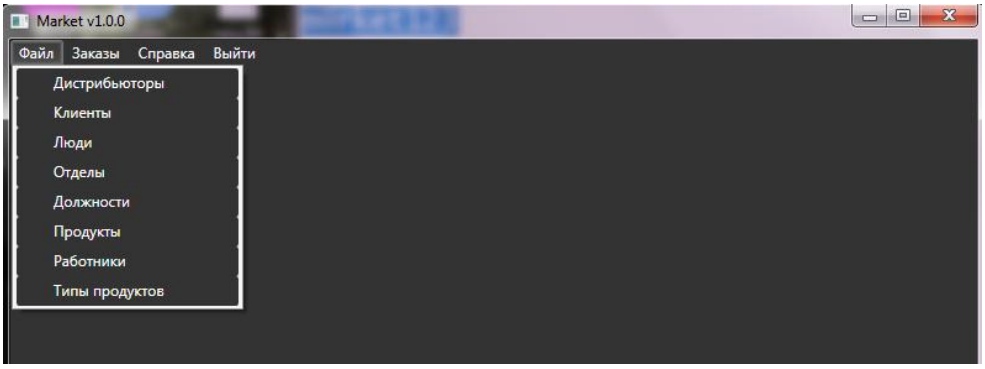


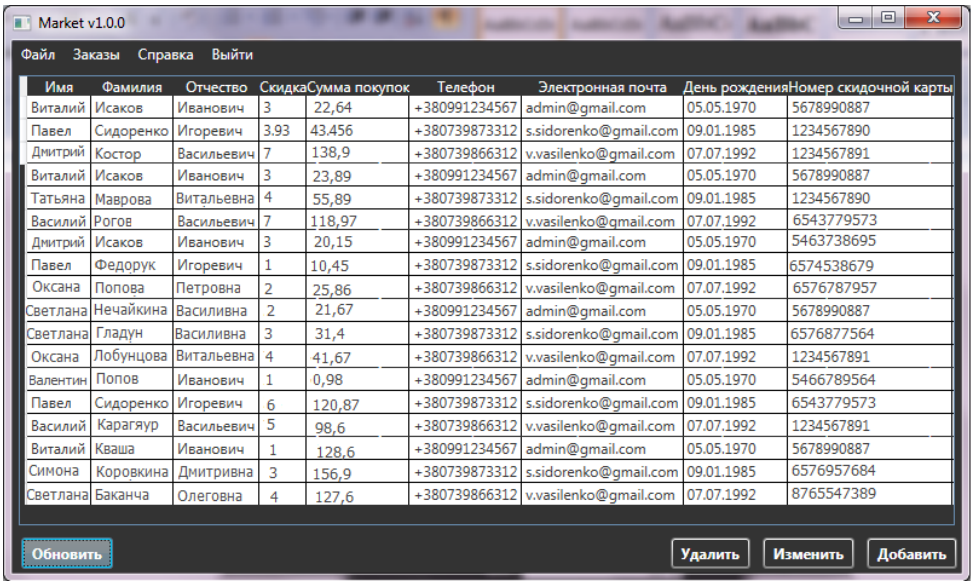

Рисунок 10 – Вкладка Файл

На рис. 11 представлений перелік фірм-дистриб'юторів з якими співпрацює магазин.


| Название дистрибьютора | Телефон | Адрес |
|---|---|---|
| ЧП Иванов | +380671235566 | г.Одесса, ул.Княжеская, 236 |
| ЧП Сахарков | +380974563278 | Одесса, ул. Лесная 15 |
| ЧП Абаржи | +380995674326 | Тарутинский, с.Веселая Долина, Светлая, дом № 15 |
| ЧП Маселков | +380995674324 | Тарутино, Ленина, дом № 228 маг. "Диана" |
| ЧП Водочка | +380995678436 | Одесса, ул. Водопроводная |
| ЧП Лакомка | +380678564783 | Кировоград, ул. Центральна, дом № 38 |
| ООЩ Ласунка | +380503401479 | Днепр, ул. Березинская, 62, 49125 |
| Авалон ЧП (буфет Прохлада) | +380995680937 | Татарбунарский, Татарбунары, Тура, дом № 19 |
| ТМ "3 медведя" | 0444992931 | м. Київ, вул. Комсомольська |
| ООО «ГАЛИЧИНА» | +380504370169 | г. Тернополь, ул. Полесская, 10 |
| КОМО | 0800503022 | Рівненська обл., м. Дубно, вул. Грушевського, 117-А |
| ТМ President | +380512765231 | м. Миколаїв, вул. Виноградна, 2 |
| ТМ Злагода | +380673330421 | м.Дніпро, вул. Журналістів, 15 |
| Колонист | +380974657847 | г. Одесса, ул Колонтаевская 15 |

Рисунок 12 – Дистриб'ютори

Перелік клієнтів зображений на рис. 13., у вигляді списку де можна побачити ПІБ клієнтів, знижка покупця, сума останньої покупки, номер телефона, електрона пошта, дата народження та номер картки.




| Имя | Фамилия | Отчество | Скидка | Сумма покупок | Телефон | Электронная почта | День рождения | Номер скидочной карты |
|---|---|---|---|---|---|---|---|---|
| Виталий | Исаков | Иванович | 3 | 22,64 | +380991234567 | admin@gmail.com | 05.05.1970 | 5678990887 |
| Павел | Сидоренко | Игоревич | 3.93 | 43,456 | +380739873312 | s.sidorenko@gmail.com | 09.01.1985 | 1234567890 |
| Дмитрий | Костор | Васильевич | 7 | 138,9 | +380739866312 | v.vasilenko@gmail.com | 07.07.1992 | 1234567891 |
| Виталий | Исаков | Иванович | 3 | 23,89 | +380991234567 | admin@gmail.com | 05.05.1970 | 5678990887 |
| Татьяна | Маврова | Витальевна | 4 | 55,89 | +380739873312 | s.sidorenko@gmail.com | 09.01.1985 | 1234567890 |
| Василий | Рогов | Васильевич | 7 | 118,97 | +380739866312 | v.vasilenko@gmail.com | 07.07.1992 | 6543779573 |
| Дмитрий | Исаков | Иванович | 3 | 20,15 | +380991234567 | admin@gmail.com | 05.05.1970 | 5463738695 |
| Павел | Федорук | Игоревич | 1 | 10,45 | +380739873312 | s.sidorenko@gmail.com | 09.01.1985 | 6574538679 |
| Оксана | Попова | Петровна | 2 | 25,86 | +380739866312 | v.vasilenko@gmail.com | 07.07.1992 | 6576787957 |
| Светлана | Нечайкина | Василивна | 2 | 21,67 | +380991234567 | admin@gmail.com | 05.05.1970 | 5678990887 |
| Светлана | Гладун | Василивна | 3 | 31,4 | +380739873312 | s.sidorenko@gmail.com | 09.01.1985 | 6576877564 |
| Оксана | Лобунцова | Витальевна | 4 | 41,67 | +380739866312 | v.vasilenko@gmail.com | 07.07.1992 | 1234567891 |
| Валентин | Попов | Иванович | 1 | 0,98 | +380991234567 | admin@gmail.com | 05.05.1970 | 5466789564 |
| Павел | Сидоренко | Игоревич | 6 | 120,87 | +380739873312 | s.sidorenko@gmail.com | 09.01.1985 | 6543779573 |
| Василий | Карагяур | Васильевич | 5 | 98,6 | +380739866312 | v.vasilenko@gmail.com | 07.07.1992 | 1234567891 |
| Виталий | Кваша | Иванович | 1 | 128,6 | +380991234567 | admin@gmail.com | 05.05.1970 | 5678990887 |
| Симона | Коровкина | Дмитривна | 3 | 156,9 | +380739873312 | s.sidorenko@gmail.com | 09.01.1985 | 6576957684 |
| Светлана | Баканча | Олеговна | 4 | 127,6 | +380739866312 | v.vasilenko@gmail.com | 07.07.1992 | 8765547389 |

Рисунок 13 – Перелік клієнтів

Також при помилці у вводі даних можна змінити потрібне значення. Для додавання клієнтів необхідно натиснути на кнопку «Добавить». Після відкриється нове вікно (рис. 14). Для додавання нових клієнтів треба заповнити всі поля. Дата народження заповнюється за допомогою календаря.

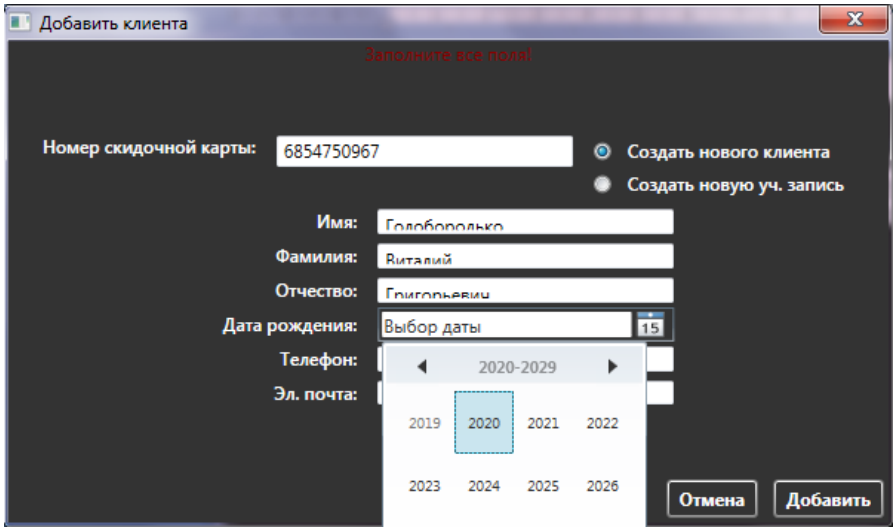



Рисунок 14 – Приклад додавання клієнтів

На рис. 15 наданий перелік основних відділів магазину.

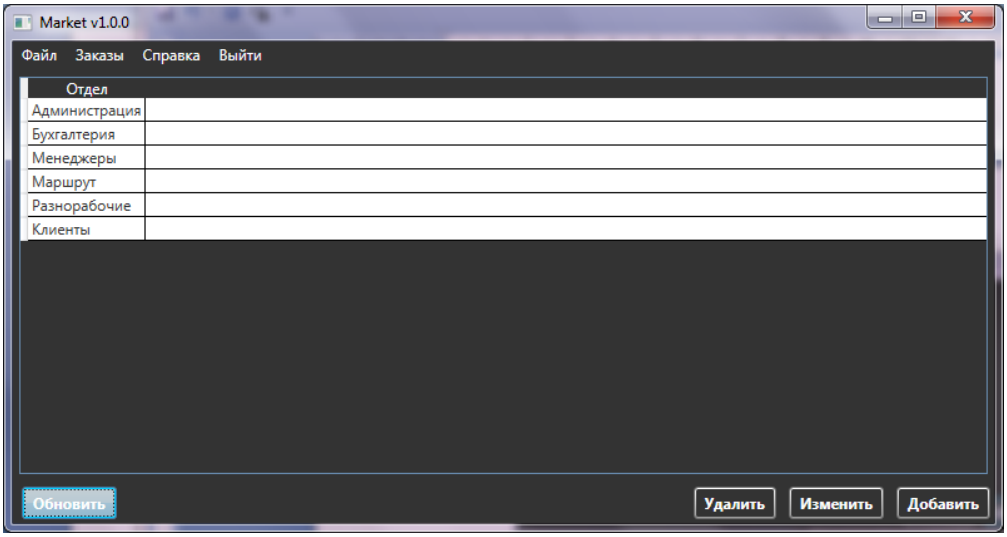



Рисунок 15 – Перелік відділів магазину

Далі на рис. 16 показано розподіл співробітників за посадами (до якого відділу відноситься, кількість місць на посаді, оклад співробітників за посадами).

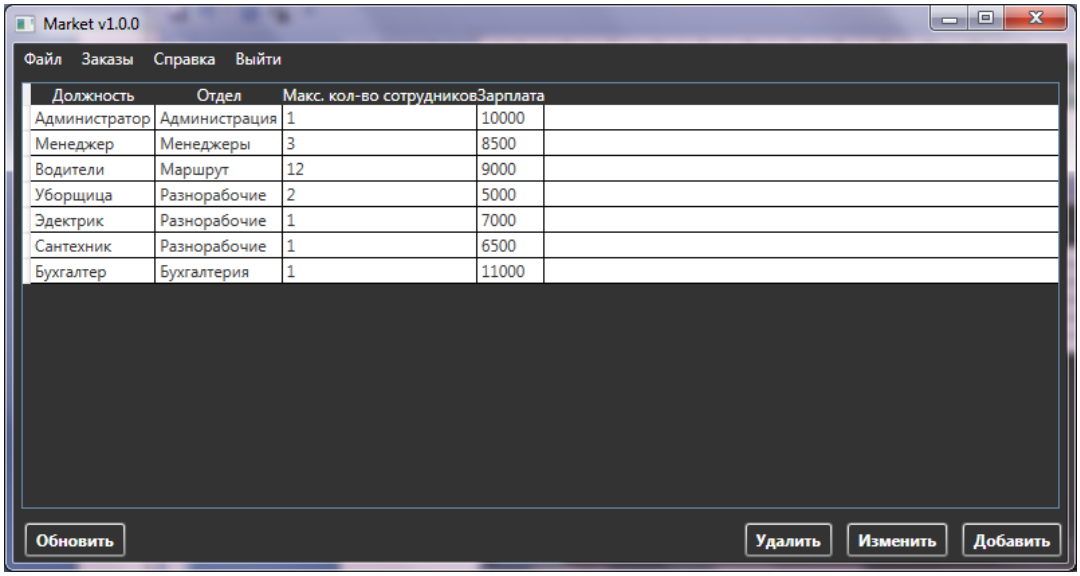
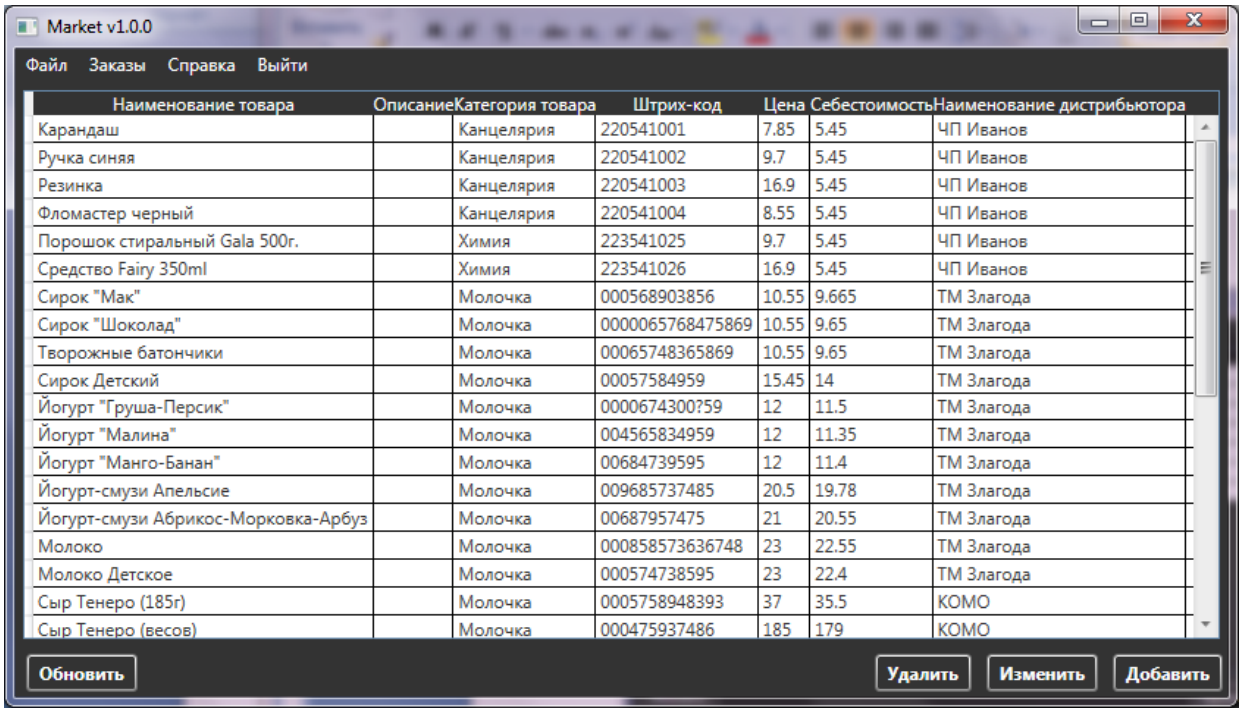



Рисунок 16 – Перелік посад

Перелік усіх товарів, представлений на рис. 17. Де показано до якої категорії належить товар, код за яким сканується товар, ціна товару, ціна закупки товару, дистриб’ютор товару




| Наименование товара | Описание | Категория товара | Штрих-код | Цена | Себестоимость | Наименование дистрибьютора |
|---|---|---|---|---|---|---|
| Карандаш | | Канцелярия | 220541001 | 7.85 | 5.45 | ЧП Иванов |
| Ручка синяя | | Канцелярия | 220541002 | 9.7 | 5.45 | ЧП Иванов |
| Резинка | | Канцелярия | 220541003 | 16.9 | 5.45 | ЧП Иванов |
| Фломастер черный | | Канцелярия | 220541004 | 8.55 | 5.45 | ЧП Иванов |
| Порошок стиральный Gala 500г. | | Химия | 223541025 | 9.7 | 5.45 | ЧП Иванов |
| Средство Fairy 350ml | | Химия | 223541026 | 16.9 | 5.45 | ЧП Иванов |
| Сирок "Мак" | | Молочка | 000568903856 | 10.55 | 9.665 | ТМ Злагода |
| Сирок "Шоколад" | | Молочка | 0000065768475869 | 10.55 | 9.65 | ТМ Злагода |
| Творожные батончики | | Молочка | 00065748365869 | 10.55 | 9.65 | ТМ Злагода |
| Сирок Детский | | Молочка | 00057584959 | 15.45 | 14 | ТМ Злагода |
| Йогурт "Груша-Персик" | | Молочка | 0000674300?59 | 12 | 11.5 | ТМ Злагода |
| Йогурт "Малина" | | Молочка | 004565834959 | 12 | 11.35 | ТМ Злагода |
| Йогурт "Манго-Банан" | | Молочка | 00684739595 | 12 | 11.4 | ТМ Злагода |
| Йогурт-смузи Апельсие | | Молочка | 009685737485 | 20.5 | 19.78 | ТМ Злагода |
| Йогурт-смузи Абрикос-Морковка-Арбуз | | Молочка | 00687957475 | 21 | 20.55 | ТМ Злагода |
| Молоко | | Молочка | 000858573636748 | 23 | 22.55 | ТМ Злагода |
| Молоко Детское | | Молочка | 000574738595 | 23 | 22.4 | ТМ Злагода |
| Сыр Тенеро (185г) | | Молочка | 0005758948393 | 37 | 35.5 | КОМО |
| Сыр Тенеро (весов) | | Молочка | 000475937486 | 185 | 179 | КОМО |

Рисунок 17 – Перелік товарів

Приклад додавання нового товару представлений на рис. 18.

Для того. Щоб додати товар необхідно з існуючого списку типів продуктів обрати потрібний, потім вибрати дистриб’ютора, і вже після ввести всі пусті поля (назва продукту, код, ціна продажу та ціна закупівлі). Вводити потрібно всі параметри. У вітрі «Краткие сведения» можна ввести інформацію про ліцензійний продукт.

Для зручного пошуку продуктів, іх необхідно розподілити за категоріями. Розподіл продуктів за категоріями показаний на рис. 19.

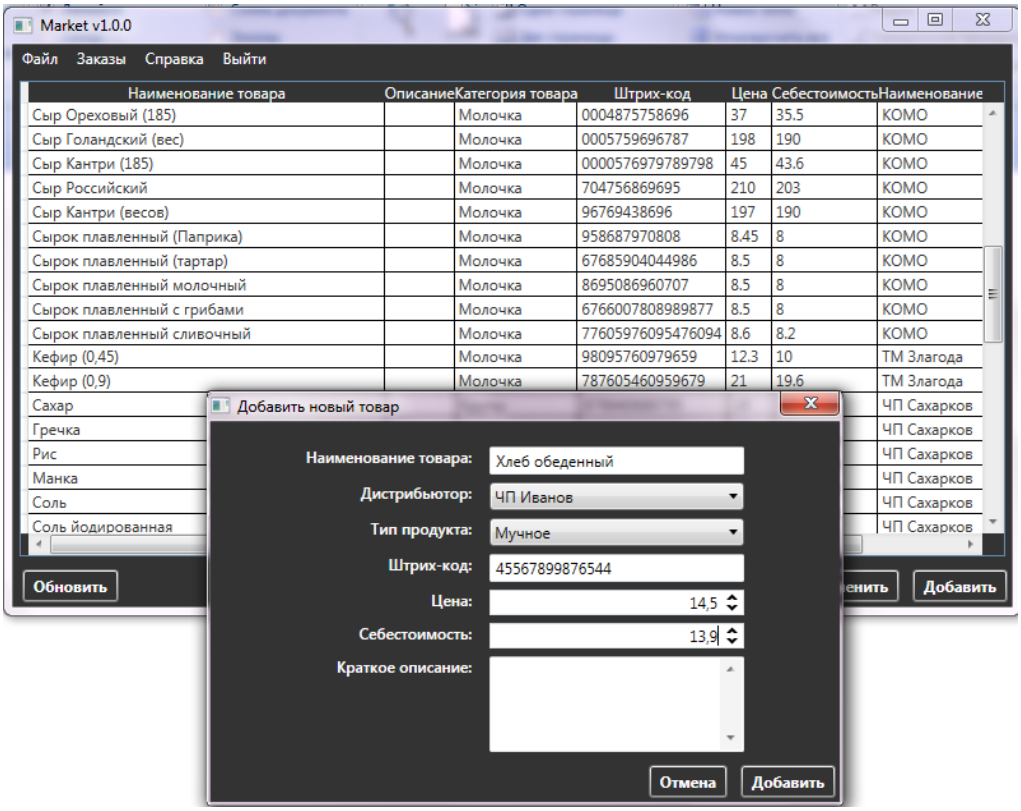



Рисунок 18 – Приклад додавання товару

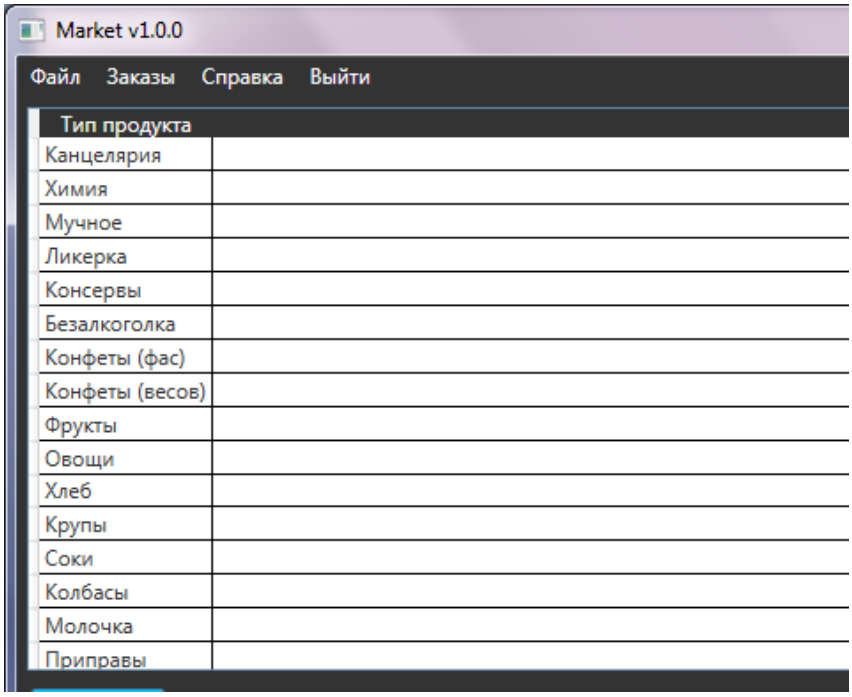
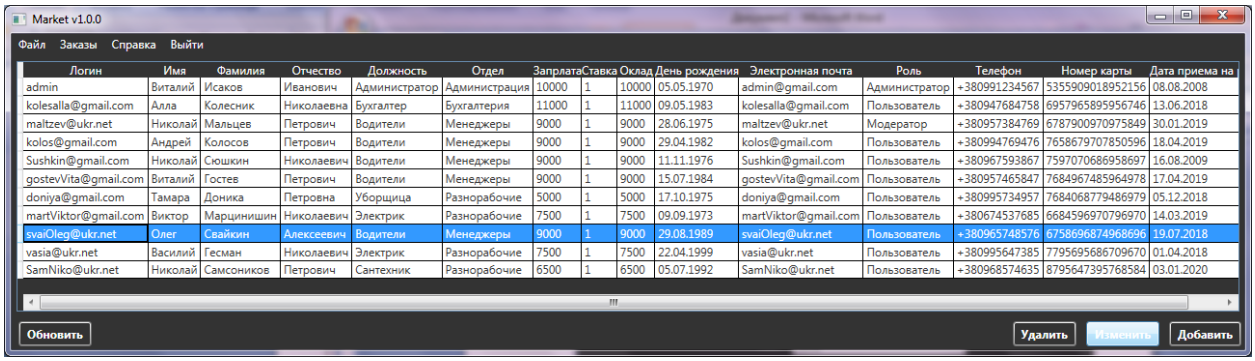



Рисунок 19 – Категорії продуктів

На рис. 20 наданий перелік робітників, які працюють в магазині. Робітників розподіляють за посадами. В переліку надається опис ПІБ, посади, відділ до якого відноситься працівник, ставка на якій він працює, оклад, дата народження, електрона пошта, номер телефону, номер картки та дата прийому на роботу.




| Логин | Имя | Фамилия | Отчество | Должность | Отдел | Запплата | Ставка | Оклад | День рождения | Электронная почта | Роль | Телефон | Номер карты | Дата приема на |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| admin | Виталий | Исаков | Иванович | Администратор | Администрация | 10000 | 1 | 10000 | 05.05.1970 | admin@gmail.com | Администратор | +380991234567 | 5355909018952156 | 08.08.2008 |
| kolesalla@gmail.com | Алла | Колесник | Николаевна | Бухгалтер | Бухгалтерия | 11000 | 1 | 11000 | 09.05.1983 | kolesalla@gmail.com | Пользователь | +380947684758 | 6957965895956746 | 13.06.2018 |
| maltzev@ukr.net | Николай | Мальцев | Петрович | Водители | Менеджеры | 9000 | 1 | 9000 | 28.06.1975 | maltzev@ukr.net | Модератор | +380957384769 | 6787900970975849 | 30.01.2019 |
| kolos@gmail.com | Андрей | Колосов | Петрович | Водители | Менеджеры | 9000 | 1 | 9000 | 29.04.1982 | kolos@gmail.com | Пользователь | +380994769476 | 7658679707850596 | 18.04.2019 |
| Sushkin@gmail.com | Николай | Сюшкин | Николаевич | Водители | Менеджеры | 9000 | 1 | 9000 | 11.11.1976 | Sushkin@gmail.com | Пользователь | +380967593867 | 7597070686958697 | 16.08.2009 |
| gostevVita@gmail.com | Виталий | Гостев | Петрович | Водители | Менеджеры | 9000 | 1 | 9000 | 15.07.1984 | gostevVita@gmail.com | Пользователь | +380957465847 | 7684967485964978 | 17.04.2019 |
| doniya@gmail.com | Тамара | Доника | Петровна | Уборщица | Разнорабочие | 5000 | 1 | 5000 | 17.10.1975 | doniya@gmail.com | Пользователь | +380995734957 | 76840687794869 79 | 05.12.2018 |
| martViktor@gmail.com | Виктор | Марцинишин | Николаевич | Электрик | Разнорабочие | 7500 | 1 | 7500 | 09.09.1973 | martViktor@gmail.com | Пользователь | +380674537685 | 6684596970796970 | 14.03.2019 |
| svaiOleg@ukr.net | Олег | Свайкин | Алексеевич | Водители | Менеджеры | 9000 | 1 | 9000 | 29.08.1989 | svaiOleg@ukr.net | Пользователь | +380965748576 | 6758696874968696 | 19.07.2018 |
| vasia@ukr.net | Василий | Гесман | Николаевич | Электрик | Разнорабочие | 7500 | 1 | 7500 | 22.04.1999 | vasia@ukr.net | Пользователь | +380995647385 | 7795695686709670 | 01.04.2018 |
| SamNiko@ukr.net | Николай | Самсоников | Петрович | Сантехник | Разнорабочие | 6500 | 1 | 6500 | 05.07.1992 | SamNiko@ukr.net | Пользователь | +380968574635 | 8795647395768584 | 03.01.2020 |

Рисунок 20 – Робітники

Для додавання співробітника, після натискання кнопки «Добавить», відкриється вікно (рис. 21), в якому необхідно ввести всі поля.

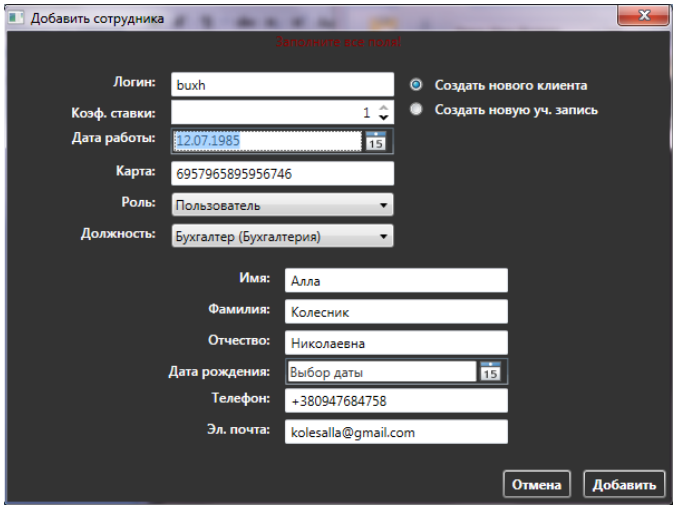



Рисунок 21 – Приклад додавання робітника

У вкладці «Закази» (рис. 22), потрібно створити заказ.

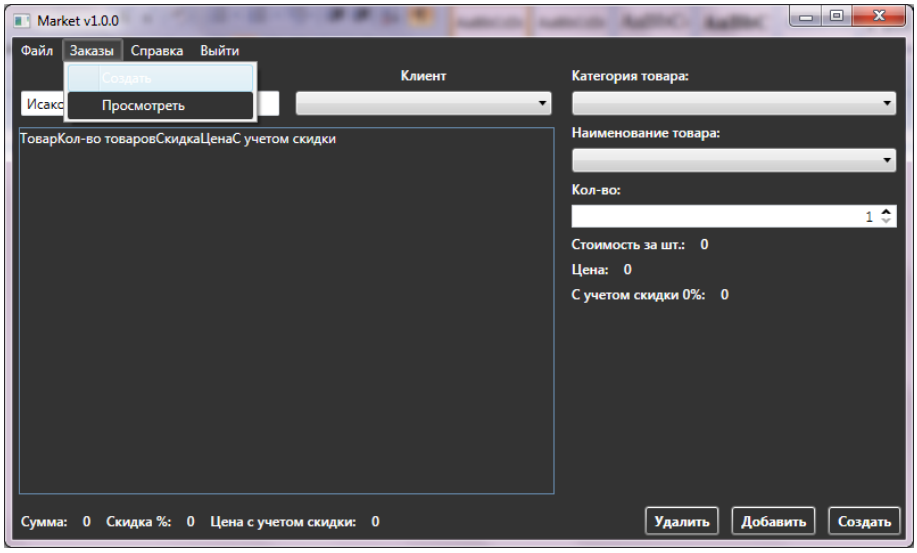



Рисунок 22 – Вкладка «Закази»

Початок оформлення заказу представлено на рис. 23, де спочатку обирається клієнт якого будуть обслуговувати і тільки потім почати сканувати товар.

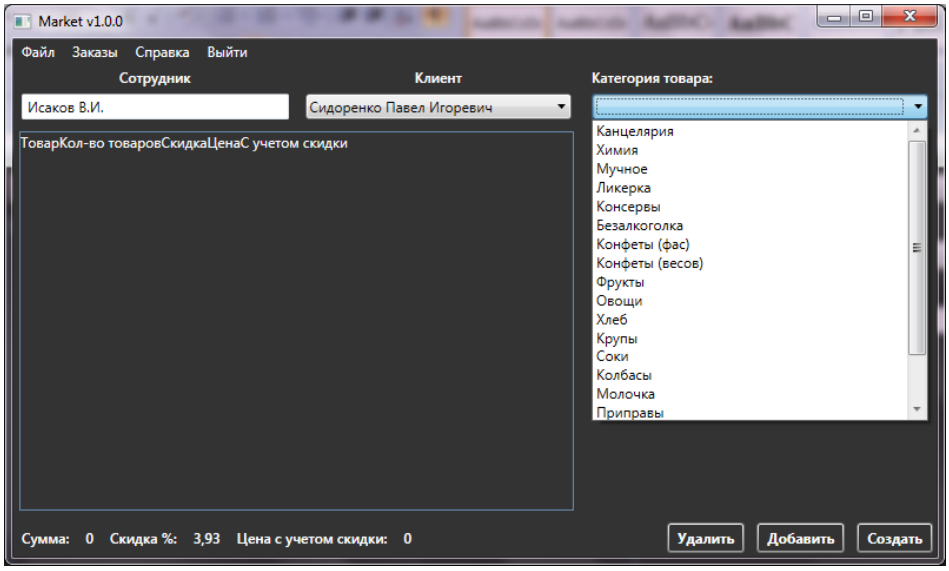


Рисунок 23 – Приклад вибору клієнта

Оформлення заказу представлено на рис. 24

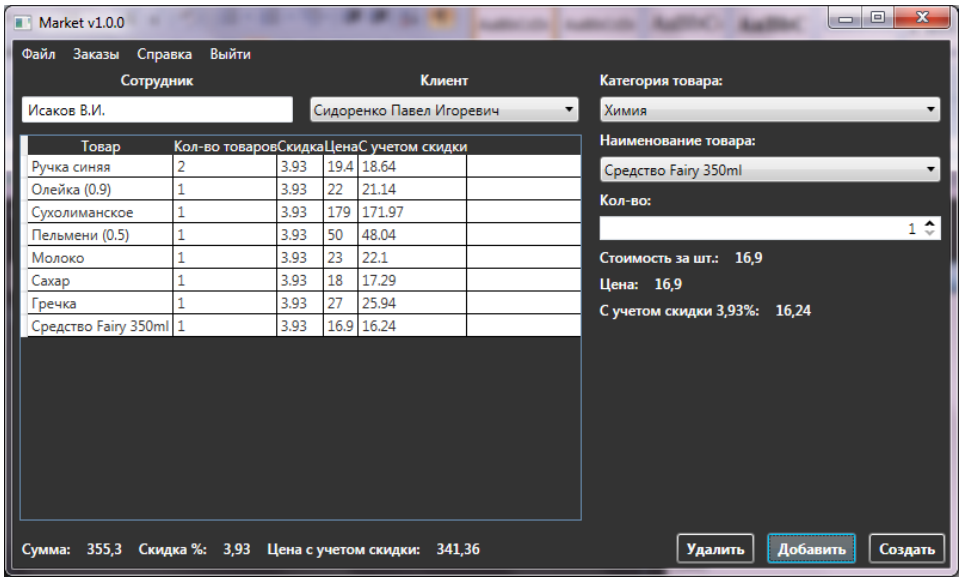



Рисунок 24 – Приклад оформлення заказу

Після необхідно натиснути на «Створити», для успішного додавання заказу (рис. 25)

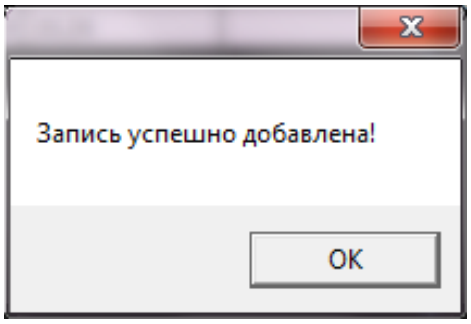



Рисунок 25 – Успішне оформлення заказу

На рис. 26 можна подивитися всі закази

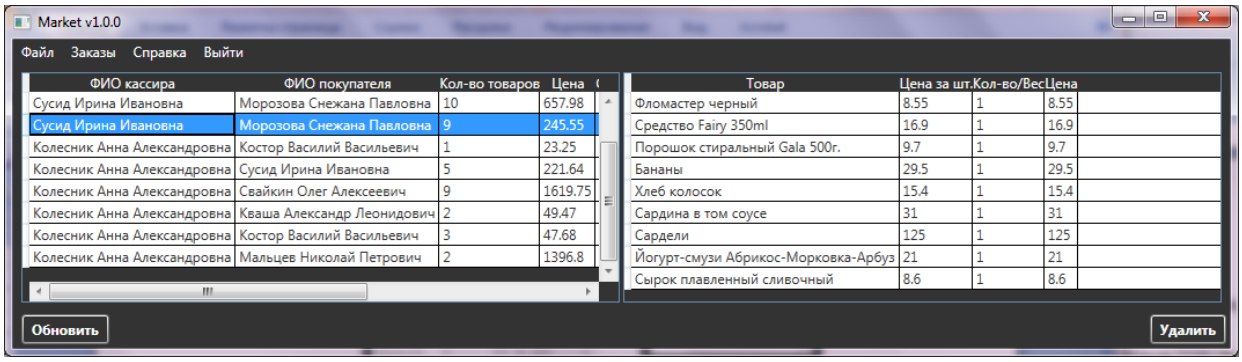



Рисунок 26 – Перегляд заказів

# ВИСНОВКИ

Була розроблена база даних магазину «MARKET» за допомогою технології WPF, що автоматично синхронізує інформацію про товари.

Для виконання мети, були вирішані наступні задачі:

– досліджено предметну область;

– спроектувано базу даних;

– створено форми для роботи з базою;

– організувано для користувача меню;

– дана програма повинна матє простий і зручний призначений для користувача інтерфейс.

Перша частина включала створення реляційної бази даних, а саме MSSQL Server, яка дозволяє працювати з даними на структурування мовою запитів T-SQL. Скрипти для створення бази даних знаходяться в каталозі DB / Scripts / .... Кожен з скриптів дозволяє створити послідовно БД, таблиці, процедури, подання, функції і злиття початкових даних. Друга частина API є сам проект WebApi, яка написана на однойменній технології від компанії Microsoft. Технологія використовує розширений шаблон MVC, який дозволяє створювати контролери для обробки логіки БД. За підсумком проект повертає JSON-об'єкти, які в майбутньому будуть використовуватися в декстопном додатку.

Вся авторизація та аутентифікація відбувається на сервері. Завдяки використанню токенов ми можемо розмежувати права серед різних користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАНЬ

1. MVC Framework – Introduction. URL: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm (дата звернення 12.04.2020)
2. MVC MODEL. URL: https://javarush.ru/groups/posts/2536− chastjh−7 −znakomstvo−s−patternom−mvc−model−view−controller (дата звернення 12.04.2020)
3. Введение в WPF. Особенности платформы WPF. URL: https://metanit.com/sharp/wpf/1.php (дата звернення 15.04.2020)
4. Windows Presentation Foundation URL: https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation (дата звернення 15.04.2020)
5. Wpf Tutorial. URL: https://www.com/about−wpf/what−is−wpf/ (дата звернення 16.04.2020)
6. .NET Framework. URL: https://ru.wikipedia.org/wiki/.NET_Framework (дата звернення 15.04.2020)
7. What is the .NET Framework (.NET). URL: https://www.techopedia.com/ definition/3734/net−framework−net (Дата звернення 20.04.2020)
8. What is Dapper? How to use Dapper? In aspnet MVC. URL: http://www.codedigest.com/ quick−start/17/what−is−dapper−how−to−use−dapper−in−aspnet−mvc (дата звернення 20.04.2020)
9. What is Microsoft SQL Server and what is it used for. URL: https://www.infotectraining.com/blog/what−is−microsoft−sql−server−and− what−is−it−used−for. (дата звернення 25.04.2020)
10. Transact SQL. URL: https://ru.wikipedia.org/wiki/Transact−SQL (дата звернення 25.04.2020)
11. JSON what it is? How it works? How to use it? URL: https://www.copterlabs.com/

json−what−it−is−how−it−works−how−to−use−it/ (дата звернення 25.04.2020)

12. JSON a better format for data exchange. URL: https: //www.infoworld.com/article/3222851/what−is−json−a−better−format−for −data− exchange.html (дата звернення 30.04.2020)

# ДОДАТОК

## Уривок лістингу програми

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection,
EntityFramework, Version=6.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" requirePermission="false" />
    <sectionGroup name="startApiGroup">
      <section name="startApiSection"
              type="DiplomaMarket.Configuration.StartApiSection"
              allowLocation="true"
              allowDefinition="Everywhere"/>
    </sectionGroup>
  </configSections>
  <startApiGroup>
    <startApiSection>
      <apiUrl url="http://market.somee.com"/>
    </startApiSection>
  </startApiGroup>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1" />
  </startup>
  <connectionStrings>
    <add name="DefaultConnection" connectionString="Initial Catalog=store.db"
providerName="System.Data.SQLite" />
  </connectionStrings>
  <entityFramework>
    <defaultConnectionFactory
type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory,
EntityFramework">
      <parameters>
        <parameter value="mssqllocaldb" />
      </parameters>
    </defaultConnectionFactory>
    <providers>
      <provider invariantName="System.Data.SQLite"
type="System.Data.SQLite.EF6.SQLiteProviderServices,
System.Data.SQLite.EF6"/>
      <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices,
EntityFramework.SqlServer" />
      <provider invariantName="System.Data.SQLite.EF6"
type="System.Data.SQLite.EF6.SQLiteProviderServices, System.Data.SQLite.EF6"
/>
    </providers>
  </entityFramework>
  <system.data>
    <DbProviderFactories>
      <add name="SQLite Data Provider (Entity Framework 6)"
invariant="System.Data.SQLite.EF6" description=".NET Framework Data Provider
for SQLite (Entity Framework 6)"
type="System.Data.SQLite.EF6.SQLiteProviderFactory, System.Data.SQLite.EF6"
/>
      <add name="SQLite Data Provider" invariant="System.Data.SQLite"
description=".NET Framework Data Provider for SQLite"
type="System.Data.SQLite.SQLiteFactory, System.Data.SQLite" />
    </DbProviderFactories>
  </system.data>
</configuration>
```

```
?xml version="1.0"?>
<doc>
    <assembly>
        <name>System.Data.SQLite</name>
    </assembly>
    <members>
        <member name="T:System.Data.SQLite.AssemblySourceIdAttribute">
            <summary>
            Defines a source code identifier custom attribute for an assembly
            manifest.
            </summary>
        </member>
        <member
name="M:System.Data.SQLite.AssemblySourceIdAttribute.#ctor(System.String)">
            <summary>
            Constructs an instance of this attribute class using the specified
            source code identifier value.
            </summary>
            <param name="value">
            The source code identifier value to use.
            </param>
        </member>
        <member
name="P:System.Data.SQLite.AssemblySourceIdAttribute.SourceId">
            <summary>
            Gets the source code identifier value.
            </summary>
        </member>
        <member name="T:System.Data.SQLite.AssemblySourceTimeStampAttribute">
            <summary>
            Defines a source code time-stamp custom attribute for an assembly
            manifest.
            </summary>
        </member>
        <member
name="M:System.Data.SQLite.AssemblySourceTimeStampAttribute.#ctor(System.Str
ing)">
            <summary>
            Constructs an instance of this attribute class using the specified
            source code time-stamp value.
            </summary>
            <param name="value">
            The source code time-stamp value to use.
            </param>
        </member>
        <member
name="P:System.Data.SQLite.AssemblySourceTimeStampAttribute.SourceTimeStamp"
>
            <summary>
            Gets the source code time-stamp value.
            </summary>
        </member>
        <member name="T:System.Data.SQLite.SQLiteLogCallback">
             <summary>
             This is the method signature for the SQLite core library logging
callback
             function for use with sqlite3_log() and the SQLITE_CONFIG_LOG.

             WARNING: This delegate is used more-or-less directly by native code,
do
                    not modify its type signature.
             </summary>
             <param name="pUserData">
             The extra data associated with this message, if any.
             </param>
             <param name="errorCode">
             The error code associated with this message.
             </param>
```

```
            <param name="pMessage">
            The message string to be logged.
            </param>
        </member>
        <member name="T:System.Data.SQLite.SQLite3">
            <summary>
            This class implements SQLiteBase completely, and is the guts of the
code that interop's SQLite with .NET
            </summary>
        </member>
        <member name="F:System.Data.SQLite.SQLite3.dbName">
            <summary>
            This field is used to refer to memory allocated for the
            SQLITE_DBCONFIG_MAINDBNAME value used with the native
            "sqlite3_db_config" API.  If allocated, the associated
            memeory will be freed when the underlying connection is
            closed.
            </summary>
        </member>
        <member name="F:System.Data.SQLite.SQLite3._sql">
            <summary>
            The opaque pointer returned to us by the sqlite provider
            </summary>
        </member>
        <member name="F:System.Data.SQLite.SQLite3._functions">
            <summary>
            The user-defined functions registered on this connection
            </summary>
        </member>
        <member name="F:System.Data.SQLite.SQLite3._shimExtensionFileName">
            <summary>
            This is the name of the native library file that contains the
            "vtshim" extension [wrapper].
            </summary>
        </member>
        <member name="F:System.Data.SQLite.SQLite3._shimIsLoadNeeded">
            <summary>
            This is the flag indicate whether the native library file that
            contains the "vtshim" extension must be dynamically loaded by
            this class prior to use.
            </summary>
        </member>
        <member name="F:System.Data.SQLite.SQLite3._shimExtensionProcName">
            <summary>
            This is the name of the native entry point for the "vtshim"
            extension [wrapper].
            </summary>
        </member>
        <member name="F:System.Data.SQLite.SQLite3._modules">
            <summary>
            The modules created using this connection.
            </summary>
        </member>
        <member
name="M:System.Data.SQLite.SQLite3.#ctor(System.Data.SQLite.SQLiteDateFormat
s,System.DateTimeKind,System.String,System.IntPtr,System.String,System.Boole
an)">
            <summary>
            Constructs the object used to interact with the SQLite core library
            using the UTF-8 text encoding.
            </summary>
            <param name="fmt">
            The DateTime format to be used when converting string values to a
            DateTime and binding DateTime parameters.
            </param>
            <param name="kind">
            The <see cref="T:System.DateTimeKind" /> to be used when creating
DateTime
            values.
```

```
            </param>
            <param name="fmtString">
            The format string to be used when parsing and formatting DateTime
            values.
            </param>
            <param name="db">
            The native handle to be associated with the database connection.
            </param>
            <param name="fileName">
            The fully qualified file name associated with <paramref name="db "/>.
            </param>
            <param name="ownHandle">
            Non-zero if the newly created object instance will need to dispose
            of <paramref name="db" /> when it is no longer needed.
            </param>
        </member>
        <member name="M:System.Data.SQLite.SQLite3.DisposeModules">
            <summary>
            This method attempts to dispose of all the <see
cref="T:System.Data.SQLite.SQLiteModule" /> derived
            object instances currently associated with the native database
connection.
            </summary>
        </member>
        <member name="M:System.Data.SQLite.SQLite3.GetCancelCount">
            <summary>
            Returns the number of times the <see
cref="M:System.Data.SQLite.SQLite3.Cancel" /> method has been
            called.
            </summary>
        </member>
        <member name="M:System.Data.SQLite.SQLite3.ShouldThrowForCancel">
            <summary>
            This method determines whether or not a <see
cref="T:System.Data.SQLite.SQLiteException" />
            with a return code of <see
cref="F:System.Data.SQLite.SQLiteErrorCode.Interrupt" /> should
            be thrown after making a call into the SQLite core library.
            </summary>
            <returns>
            Non-zero if a <see cref="T:System.Data.SQLite.SQLiteException" /> to
be thrown.  This method
            will only return non-zero if the <see
cref="M:System.Data.SQLite.SQLite3.Cancel" /> method was called
            one or more times during a call into the SQLite core library (e.g. when
            the sqlite3_prepare*() or sqlite3_step() APIs are used).
            </returns>
        </member>
        <member name="M:System.Data.SQLite.SQLite3.ResetCancelCount">
            <summary>
            Resets the value of the <see
cref="F:System.Data.SQLite.SQLite3._cancelCount" /> field.
            </summary>
        </member>
        <member name="M:System.Data.SQLite.SQLite3.Cancel">
            <summary>
            Attempts to interrupt the query currently executing on the associated
            native database connection.
            </summary>
        </member>
        <member
name="M:System.Data.SQLite.SQLite3.BindFunction(System.Data.SQLite.SQLiteFun
ctionAttribute,System.Data.SQLite.SQLiteFunction,System.Data.SQLite.SQLiteCo
nnectionFlags)">
            <summary>
            This function binds a user-defined function to the connection.
            </summary>
            <param name="functionAttribute">
```

```
          The <see cref="T:System.Data.SQLite.SQLiteFunctionAttribute"/>
object instance containing
          the metadata for the function to be bound.
          </param>
          <param name="function">
          The <see cref="T:System.Data.SQLite.SQLiteFunction"/> object instance
that implements the
          function to be bound.
          </param>
          <param name="flags">
          The flags associated with the parent connection object.
          </param>
       </member>
       <member
name="M:System.Data.SQLite.SQLite3.UnbindFunction(System.Data.SQLite.SQLiteF
unctionAttribute,System.Data.SQLite.SQLiteConnectionFlags)">
          <summary>
          This function binds a user-defined function to the connection.
          </summary>
          <param name="functionAttribute">
          The <see cref="T:System.Data.SQLite.SQLiteFunctionAttribute"/>
object instance containing
          the metadata for the function to be unbound.
          </param>
          <param name="flags">
          The flags associated with the parent connection object.
          </param>
          <returns>Non-zero if the function was unbound and removed.</returns>
       </member>
       <member name="P:System.Data.SQLite.SQLite3.OwnHandle">
          <summary>
          Returns non-zero if the underlying native connection handle is owned
          by this instance.
          </summary>
       </member>
       <member name="P:System.Data.SQLite.SQLite3.Functions">
          <summary>
          Returns the logical list of functions associated with this connection.
          </summary>
       </member>
       <member name="M:System.Data.SQLite.SQLite3.ReleaseMemory">
          <summary>
          Attempts to free as much heap memory as possible for the database
connection.
          </summary>
          <returns>A standard SQLite return code (i.e. zero for success and
non-zero for failure).</returns>
       </member>
       <member
name="M:System.Data.SQLite.SQLite3.StaticReleaseMemory(System.Int32,System.B
oolean,System.Boolean,System.Int32@,System.Boolean@,System.UInt32@)">
          <summary>
          Attempts to free N bytes of heap memory by deallocating non-essential
memory
          allocations held by the database library. Memory used to cache database
pages
          to improve performance is an example of non-essential memory.  This is
a no-op
          returning zero if the SQLite core library was not compiled with the
compile-time
          option SQLITE_ENABLE_MEMORY_MANAGEMENT.  Optionally, attempts to
reset and/or
          compact the Win32 native heap, if applicable.
          </summary>
          <param name="nBytes">
          The requested number of bytes to free.
          </param>
          <param name="reset">
          Non-zero to attempt a heap reset.
```

```
            </param>
            <param name="compact">
            Non-zero to attempt heap compaction.
            </param>
            <param name="nFree">
            The number of bytes actually freed.  This value may be zero.
            </param>
            <param name="resetOk">
            This value will be non-zero if the heap reset was successful.
            </param>
            <param name="nLargest">
            The size of the largest committed free block in the heap, in bytes.
            This value will be zero unless heap compaction is enabled.
            </param>
            <returns>
            A standard SQLite return code (i.e. zero for success and non-zero
            for failure).
            </returns>
        </member>
        <member name="M:System.Data.SQLite.SQLite3.Shutdown">
            <summary>
            Shutdown the SQLite engine so that it can be restarted with different
            configuration options.  We depend on auto initialization to recover.
            </summary>
            <returns>Returns a standard SQLite result code.</returns>
        </member>
        <member
name="M:System.Data.SQLite.SQLite3.StaticShutdown(System.Boolean)">
            <summary>
            Shutdown the SQLite engine so that it can be restarted with different
            configuration options.  We depend on auto initialization to recover.
            </summary>
            <param name="directories">
            Non-zero to reset the database and temporary directories to their
            default values, which should be null for both.  This parameter has no
            effect on non-Windows operating systems.
            </param>
            <returns>Returns a standard SQLite result code.</returns>
        </member>
        <member name="M:System.Data.SQLite.SQLite3.IsOpen">
            <summary>
            Determines if the associated native connection handle is open.
            </summary>
            <returns>
            Non-zero if the associated native connection handle is open.
            </returns>
        </member>
        <member
name="M:System.Data.SQLite.SQLite3.GetFileName(System.String)">
            <summary>
            Returns the fully qualified path and file name for the currently open
            database, if any.
            </summary>
            <param name="dbName">
            The name of the attached database to query.
            </param>
            <returns>
       <member
name="M:System.Data.SQLite.SQLiteChangeSetMetadataItem.GetOldValue(System.In
t32)">
            <summary>
            Queries and returns the original value of a given column for this
            change.  This method may only be called when the
            <see
cref="P:System.Data.SQLite.SQLiteChangeSetMetadataItem.OperationCode" /> has a
value of
            <see cref="F:System.Data.SQLite.SQLiteAuthorizerActionCode.Update"
/> or
```

```
            <see cref="F:System.Data.SQLite.SQLiteAuthorizerActionCode.Delete"
/>.
            </summary>
            <param name="columnIndex">
            The index for the column.  This value must be between zero and one
            less than the total number of columns for this table.
            </param>
            <returns>
            The original value of a given column for this change.
            </returns>
        </member>
        <member
name="M:System.Data.SQLite.SQLiteChangeSetMetadataItem.GetNewValue(System.In
t32)">
            <summary>
            Queries and returns the updated value of a given column for this
            change.  This method may only be called when the
            <see
cref="P:System.Data.SQLite.SQLiteChangeSetMetadataItem.OperationCode" /> has a
value of
            <see cref="F:System.Data.SQLite.SQLiteAuthorizerActionCode.Insert"
/> or
            <see cref="F:System.Data.SQLite.SQLiteAuthorizerActionCode.Update"
/>.
            </summary>
            <param name="columnIndex">
            The index for the column.  This value must be between zero and one
            less than the total number of columns for this table.
            </param>
            <returns>
            The updated value of a given column for this change.
            </returns>
        </member>
        <member
name="M:System.Data.SQLite.SQLiteChangeSetMetadataItem.GetConflictValue(Syst
em.Int32)">
            <summary>
            Queries and returns the conflicting value of a given column for
            this change.  This method may only be called from within a
            <see cref="T:System.Data.SQLite.SessionConflictCallback" /> delegate
when the conflict
            type is <see
cref="F:System.Data.SQLite.SQLiteChangeSetConflictType.Data" /> or
            <see
cref="F:System.Data.SQLite.SQLiteChangeSetConflictType.Conflict" />.
            </summary>
            <param name="columnIndex">
            The index for the column.  This value must be between zero and one
            less than the total number of columns for this table.
            </param>
            <returns>
            The conflicting value of a given column for this change.
            </returns>
        </member>
        <member
name="M:System.Data.SQLite.SQLiteChangeSetMetadataItem.Dispose">
            <summary>
            Disposes of this object instance.
            </summary>
        </member>
        <member
name="F:System.Data.SQLite.SQLiteChangeSetMetadataItem.disposed">
            <summary>
            Non-zero if this object instance has been disposed.
            </summary>
        </member>
        <member
name="M:System.Data.SQLite.SQLiteChangeSetMetadataItem.CheckDisposed">
            <summary>
```

```
            Throws an exception if this object instance has been disposed.
            </summary>
        </member>
        <member
name="M:System.Data.SQLite.SQLiteChangeSetMetadataItem.Dispose(System.Boolea
n)">
            <summary>
            Disposes or finalizes this object instance.
            </summary>
            <param name="disposing">
            Non-zero if this object is being disposed; otherwise, this object
            is being finalized.
            </param>
        </member>
        <member
name="M:System.Data.SQLite.SQLiteChangeSetMetadataItem.Finalize">
            <summary>
            Finalizes this object instance.
            </summary>
        </member>
    </members>
</doc>
```