Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного забезпечення для маркетингових досліджень»

Виконав: студент 4 курсу
групи 1ПІ-18Б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Наумук Д. О.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Тужанський С.Є.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. КН Колесницький О.К.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«____» _____ 2022 р.

ВНТУ – 2022

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
25 березня 2022 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Наумуку Денису Олеговичу

1. Тема роботи – Розробка програмного забезпечення для маркетингових досліджень.

Керівник роботи: Тужанський С. Є., к.т.н., доц. кафедри ПЗ, затверджений наказом вищого навчального закладу від 24 березня 2022 р. № 66.

2. Строк подання студентом роботи 13 червня 2022 р.

3. Вихідні дані до роботи: середовище розробки Visual Studio, мова розробки C#, мова структурованих запитів SQL, операційна система – Windows 11, базові алгоритми для визначення ефективності рекламних кампаній.

4. Зміст розрахунково-пояснювальної записки: вступ; обґрунтування вибору методу розробки та постановка задач дослідження; розробка структури та алгоритмів програмного продукту; розробка модулю для знаходження статистичних даних маркетингових кампаній через заявки; тестування додатку; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: мета і задачі роботи; блок-схеми алгоритмів роботи додатку; структура веб-сервісу; графічний інтерфейс додатку; тестування додатку; висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Тужанський С.Є., к.т.н, доцент кафедри ПЗ	**26.03.2022**	**10.06.2022**

7. Дата видачі завдання 25 березня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану галузі, обґрунтування доцільності розробки	26.03.2022 – 03.04.2022	Вик.
2	Розробка методів і моделей реалізації програми для аналізу маркетингових даних.	04.04.2022 – 15.04.2022	Вик.
3	Проектування бази даних.	18.04.2022 – 24.04.2022	Вик.
4	Варіантний аналіз та обґрунтування вибору засобів реалізації системи	25.04.2022 – 02.05.2022	Вик.
5	Розробка програмного коду системи для аналізу та візуалізації даних	03.05.2022 – 23.05.2022	Вик.
6	Тестування роботи системи для аналізу	24.05.2022 – 30.05.2022	Вик.
7	Оформлення матеріалів до захисту БДР	31.05.2022 – 10.06.2022	Вик.

Студент _____ Наумук Д.О.
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ Тужанський С.Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У бакалаврській дипломній роботі розроблено сервіс, який дозволяє обчислювати статистичні дані по маркетингу, зокрема, ефективність рекламних кампаній, ефективність ресурсу трафіку, ефективність відділу продажів та інші маркетингові дані.

Подальшого розвитку отримав метод дослідження статистичних даних маркетингу, у якому, на відміну від існуючих, використано комп'ютерну візуалізацію процесу дослідження та додання нових ресурсів трафіку, що дозволило підвищити рівень засвоєння навчальних матеріалів при використанні додатку в навчальному процесі.

Розробка виконана з використанням мови програмування C#, мови структурованих запитів SQL.

ABSTRACT

In the bachelor's thesis, a service has been developed that allows you to calculate marketing statistics, in particular, the effectiveness of advertising campaigns, the effectiveness of traffic, median, fashion and others.

The method of research of marketing statistics was further developed, in which, unlike the existing ones, computer visualization of the research process and addition of new traffic resources was used, which allowed to increase the level of learning materials when using the application in the educational process.

The development was performed using the C # programming language, the SQL structured query language.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Однією з умов розвитку маркетингових досліджень є широке застосування статистичної методології.

Маркетинг використовує різні інструменти загальної теорії статистики, зокрема дисперсійний і компонентний аналізи, факторний і дискримінантний аналіз, метод експертних оцінок, кореляційний і регресійний аналіз, трендові моделі, багатофакторні статистичні моделі, прогнозування та інші [1].

На базі методології статистики та завдань маркетингу формується суміжна маркетинг-статистика - розділ прикладної статистики, орієнтований на вирішення прикладних завдань маркетингових досліджень [2].

Ніякий бізнес у наші часи не може існувати без маркетингу, адже зараз у кожній категорії продуктів є хоча б декілька представників. І щоб довести клієнтам, що саме ваш продукт є найкращим і потрібен клієнту, потрібно вміло використовувати усі канали комунікації з ним [3].

Оскільки маркетинг завжди націлений на певну групу людей, об’єднаних спільною характерністю, неможливо дослідити результативність маркетингової кампанії за діями одного клієнта. Потрібно використовувати певний пул інформації, який допоможе нам зрозуміти ефективність реклами на певний сегмент аудиторії. Саме тому у маркетингу найголовніше – це статистика [4].

Завдяки різноманітним метрикам маркетологам сьогодні вдається будувати ефективні рекламні кампанії, які приносять бізнесам клієнтів за найменшими цінами [5].

Отже, тема бакалаврської дипломної роботи, присвячена розробці програмного забезпечення із удосконаленими метриками оперативного аналізу рекламної діяльності фірми із доступним та зручним інтерфейсом є актуальною і перспективною.

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

Мета та завдання дослідження. Метою роботи є розширення функціональних можливостей програмного забезпечення для аналізу маркетингових даних.

Основними задачами роботи є:

– аналіз методів оцінки ефективності рекламної компанії і маркетингу;

– розробка алгоритму знаходження ефективних ресурсів трафіку;

– вибір програмних засобів для вирішення поставлених завдань;

– розробка та тестування програмного продукту.

Об’єкт дослідження – процес розробки програмного забезпечення для формування та аналізу статистичних даних рекламної кампанії.

Предмет дослідження – методи та програмні засоби обчислення статистичних даних рекламної кампанії.

Методи дослідження. У процесі досліджень використовувались: теорія ймовірностей та математична статистики, теорія алгоритмів для розробки моделей та методів знаходження статистичних даних рекламної кампанії; комп’ютерне моделювання для перевірки отриманих результатів.

Наукова новизна отриманих результатів.

Подальшого розвитку отримав метод дослідження статистичних даних рекламної кампанії, у якому, на відміну від існуючих, використано удосконалену метрику оцінювання ефективності, що дозволило підвищити оперативність та доступність аналізу даних.

Апробація матеріалів бакалаврської дипломної роботи. Основні положення бакалаврської дипломної роботи доповідалися та обговорювалися на LI Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2022) у секції програмного забезпечення.

Публікації. Основні результати дослідження опубліковані в науковій роботі – в тезах доповіді на LI Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2022) у секції програмного забезпечення. [3].

Аналіз. У пояснювальній записці до бакалаврської дипломної роботи було розглянуто 4 розділи та було використано 20 літературних джерел.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Основні статистичні характеристики, їх властивості

Статистика - це часто єдиний спосіб вивчити наше суспільство в цілому, його проблеми, соціальне та економічне становище, крім цього вона займається вивченням суспільної думки, яка є критерієм оцінки ситуації в країні. Немає жодної сфери життя, де статистика не грала величезної ролі, тому можна сказати, що статистика є єдиним способом суспільства вивчати само себе [6].

Статистика маркетингу – це частина галузі знань статистики, у якій розглядаються загальні питання збору, виміру та аналізу масових статистичних даних про діяльність підприємства, його внутрішнього і зовнішнього середовища, з метою розробки маркетингових заходів, що сприяють розвитку підприємства в часі.

Також це сукупність даних і показників, що характеризують кількісний бік масових явищ і подій, що вивчаються в маркетинговій діяльності, у числовій формі [7].

У даній бакалаврській роботі розглядаються метрики, якими вимірюють та виявляють ефективність рекламної кампанії. Було розглянуто такі основні характеристики:

1) CPM (Cost per mile) – Вартість за 1000 показів оголошення. Де x – це CPM(Вартість оголошення), p – рекламні витрати, k – кількість переглядів.

$$X=\frac{p}{k}*1000 \quad (1.1)$$

2) CPC (Cost per click) – Вартість за 1 клік по оголошенню. Де x – це CPC (Ціна за клік), p – рекламні витрати, k – кількість кліків.

$$X=\frac{p}{k} \quad (1.2)$$

3) CTR (Click through rate) – Клікабельність оголошення. Обчислюється у відсотках. У формулі X – CTR (клікабельність оголошення), p – кількість кліків оголошення, k – враження (перегляди оголошення).

$$X=\frac{p}{k} * 100\% \quad (1.3)$$

4) CAC (Customer Acquisition Cost) – Вартість клієнта. У формулі X – CAC (Вартісь клієнта), p – витрати на маркетинг, k – кількість приведених лідів.

$$X=\frac{p}{k} * 100\% \quad (1.4)$$

5) ROI (Return of investments) – Повернення інвестицій. Обчислюється у відсотках. У формулі X – ROI (Повернення інвестицій), p – валовий дохід із приведених клієнтів, k – витрати на маркетинг (лідогенерацію).

$$X=\frac{p}{k} * 100\% - 100 \quad (1.5)$$

1.2 Порівняльний аналіз аналогів

Для визначення потреб, переваг та функціоналу розроблюваної програми перш за все необхідно проаналізувати конкурентів (Рисунок 1.1).

Roistat - сервіс наскрізної аналітики, який збирає дані з CRM, рекламних каналів, сайту, колтрекінгу в одному вікні. Можна відстежити весь шлях клієнта: від кліку реклами до покупки. На одній платформі зібрано 17 сервісів: Аналітика, Коллтрекінг, Ловець Лідів, Мовна аналітика, Онлайн-чат, А/Б тести та інші [8].

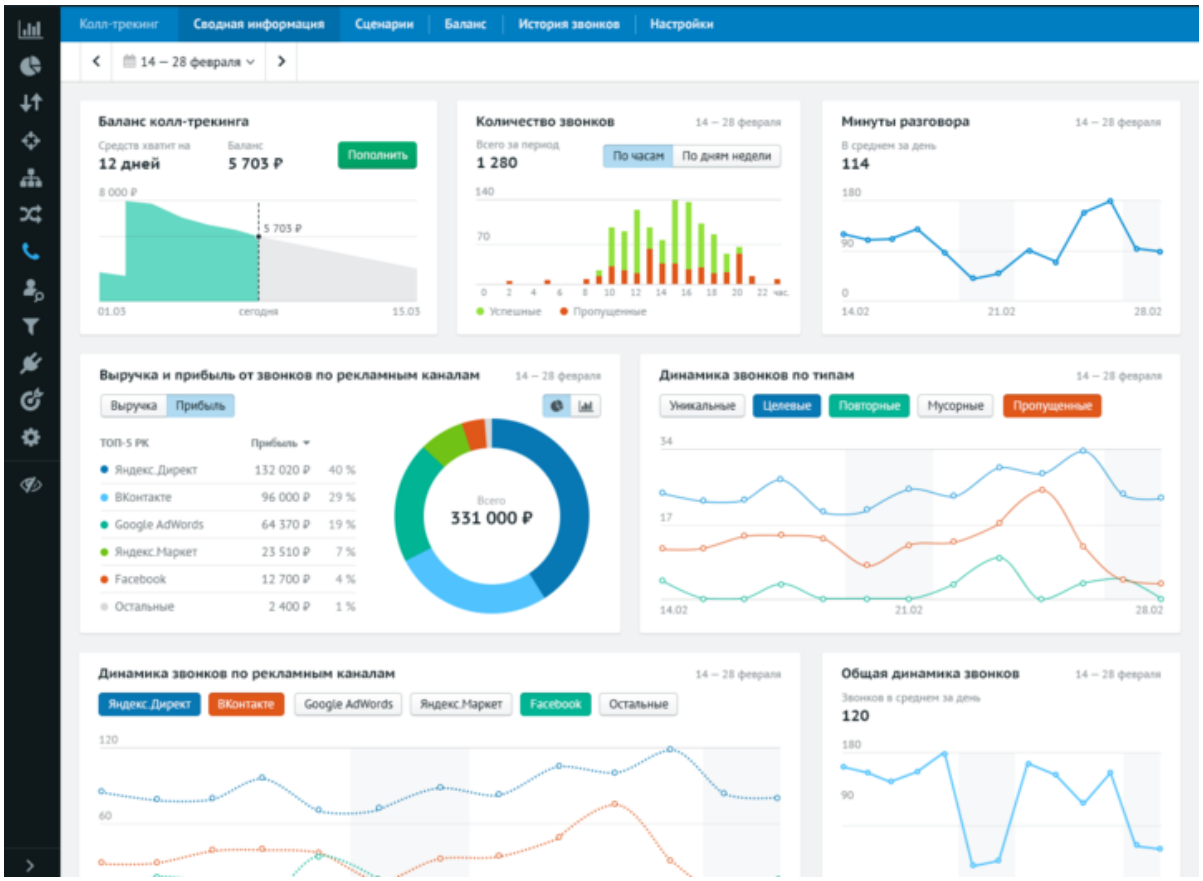


Рисунок 1.1 – Інтерфейс програми «Roistat»

Переваги програми:

– широкий функціонал;
– зрозумілий інтерфейс;
– можливість автоматично підвантажувати дані;
– є колл-трекінг.

Недоліки програми:

– відсутня можливість додавати ліди вручну;
– розробник програми – російська IT Фірма;
– програма є платною;
– більшість функціоналу може бути не потрібним користувачеві.

Сервіс «Google Analytics» є набагато професійною у порівнянні з попередньою. Її інтерфейс представлено на рисунку 1.2. Дана програма є достатньо зручною та простою у користуванні для оцінки відвідуваності та дій на сайті. Але як і у минулій програмі відсутня можливість заносити ліди вручну [9].

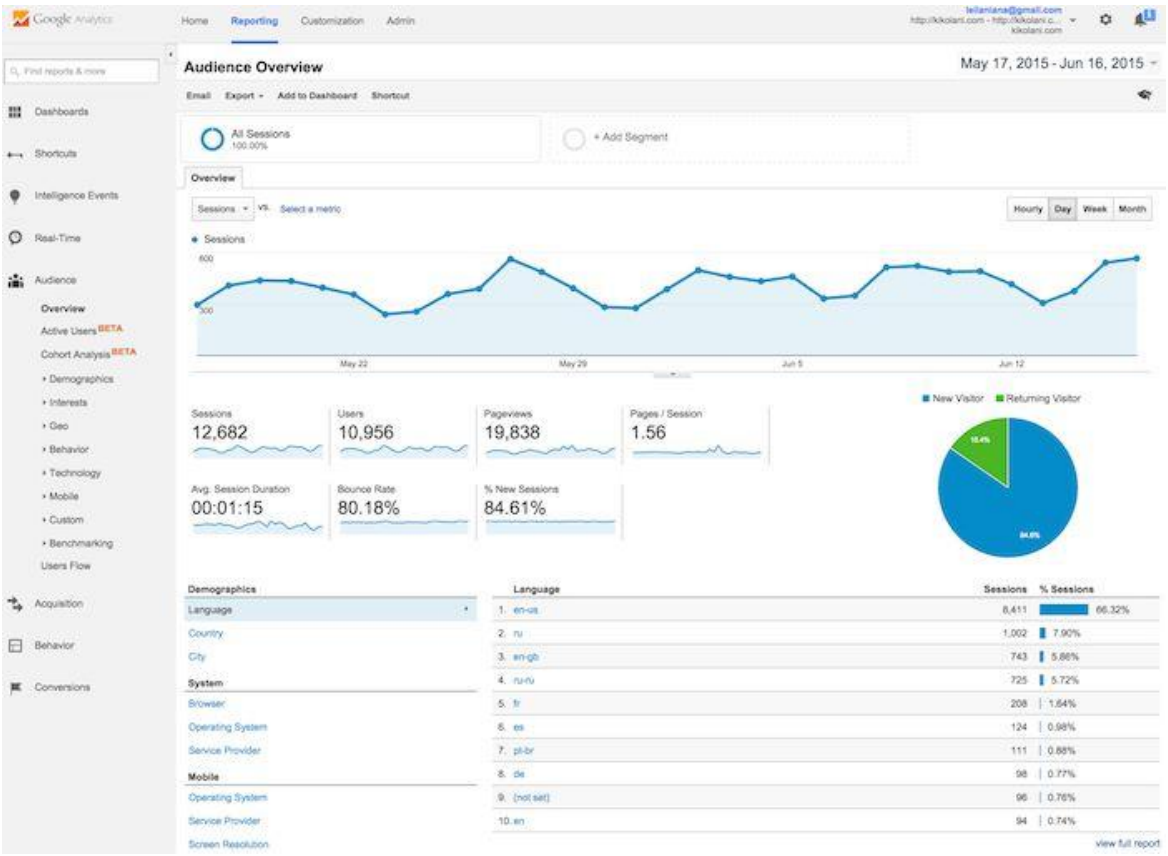


Рисунок 1.2 – Інтерфейс сервісу «Google Analytics»

Переваги програми:

– широкий функціонал;

– зрозумілий інтерфейс;

– автоматичне підвантажувати дані;

– є можливість будувати звіти.

Недоліки програми:

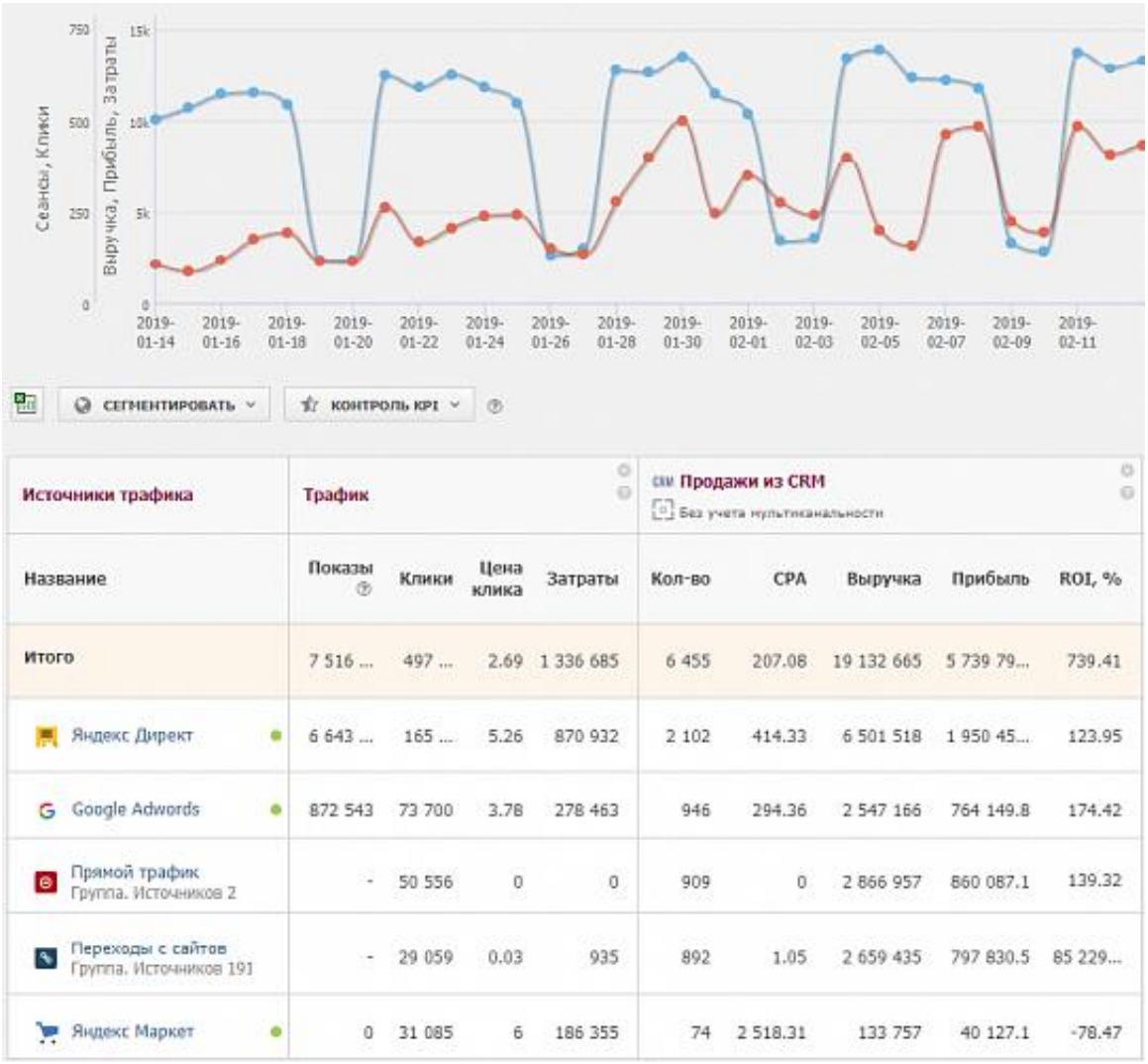
– відсутня можливість додавати ліди вручну;

– аналітика поширюється лише на сайт;

– перед користуванням варто пройти підготовку;

– більшість функціоналу може бути не потрібним користувачеві.

Наступною програмою у порівнянні стане Alytics (Рисунок 1.3). Програма є прямим аналогом Roistat з можливістю керування E-mail маркетингу.



Источники трафика	Трафик				Продажи из CRM				
					Без учета мультиканальности				
Название	Показы	Клики	Цена клика	Затраты	Кол-во	CPA	Выручка	Прибыль	ROI, %
Итого	7 516 ...	497 ...	2.69	1 336 685	6 455	207.08	19 132 665	5 739 79...	739.41
Яндекс Директ	6 643 ...	165 ...	5.26	870 932	2 102	414.33	6 501 518	1 950 45...	123.95
Google Adwords	872 543	73 700	3.78	278 463	946	294.36	2 547 166	764 149.8	174.42
Прямой трафик Группа. Источников 2	-	50 556	0	0	909	0	2 866 957	860 087.1	139.32
Переходы с сайтов Группа. Источников 191	-	29 059	0.03	935	892	1.05	2 659 435	797 830.5	85 229...
Яндекс Маркет	0	31 085	6	186 355	74	2 518.31	133 757	40 127.1	-78.47

Рисунок 1.3 – Інтерфейс програми «Alytics»

Переваги програми:

– широкий функціонал;
– зрозумілий інтерфейс;
– можливість автоматично підвантажувати дані;
– є e-mail маркетинг.

Недоліки програми:

– відсутня можливість додавати ліди вручну;
– розробник програми – російська IT Фірма;
– програма є платною;
– більшість функціоналу зав`язано на російських сервісах надання маркетингових послуг.

Виконавши аналіз схожих програм для аналізу маркетингових даних, можна скласти порівняльну таблицю (табл. 1.1), яка показує головні відмінності порівнювальних програм.

Таблиця 1.1 – Порівняння аналогів

Функціонал	Roistat	Google Analytics	Alytic	Marketing Analytics
Не прив’язано до певного сайту/ресурсу	+	-	+	+
Безкоштовний	-	+	-	+
Має функцію визначення ефективності рекламної кампанії	-	-	-	+
Має аналітику воронки для певного ресурсу трафіку	+	-	-	+
Має аналітику дослідження роботи відділу продажів	-	-	-	+

1.3 Аналіз методів розв’язання поставленої задачі

Ведення, групування та підрахунок статистичних даних розділити на 2 етапи.

– вибір способу подання аналітики;

– вибір способу обрахування аналітики;

У якості подання інформації одним із найкращих середовищ для розробки таких програм є Windows Forms.

Windows Forms (Рисунок 1.4) — інтерфейс програмування програм, який відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Цей інтерфейс спрощує доступ до елементів інтерфейсу Microsoft

Windows за рахунок створення обгортки для існуючого Win32 API в керованому коді. Причому керований код класи, що реалізують API для Windows Forms, не залежать від мови розробки. Тобто програміст однаково може використовувати Windows Forms як із написанні ПЗ на C#, З++, і VB.Net, J# та інших.

Як і Abstract Window Toolkit (AWT) (схожий API для мови Java), бібліотека Windows Forms була розроблена як частина .NET Framework для спрощення розробки компонентів графічного інтерфейсу користувача. Windows Forms побудована на основі застарілого Windows API [10].

Windows Forms надає можливість розробки кросплатформного графічного користувацького інтерфейсу. Але Windows Forms є лише обгорткою Windows API-компонентів [11].

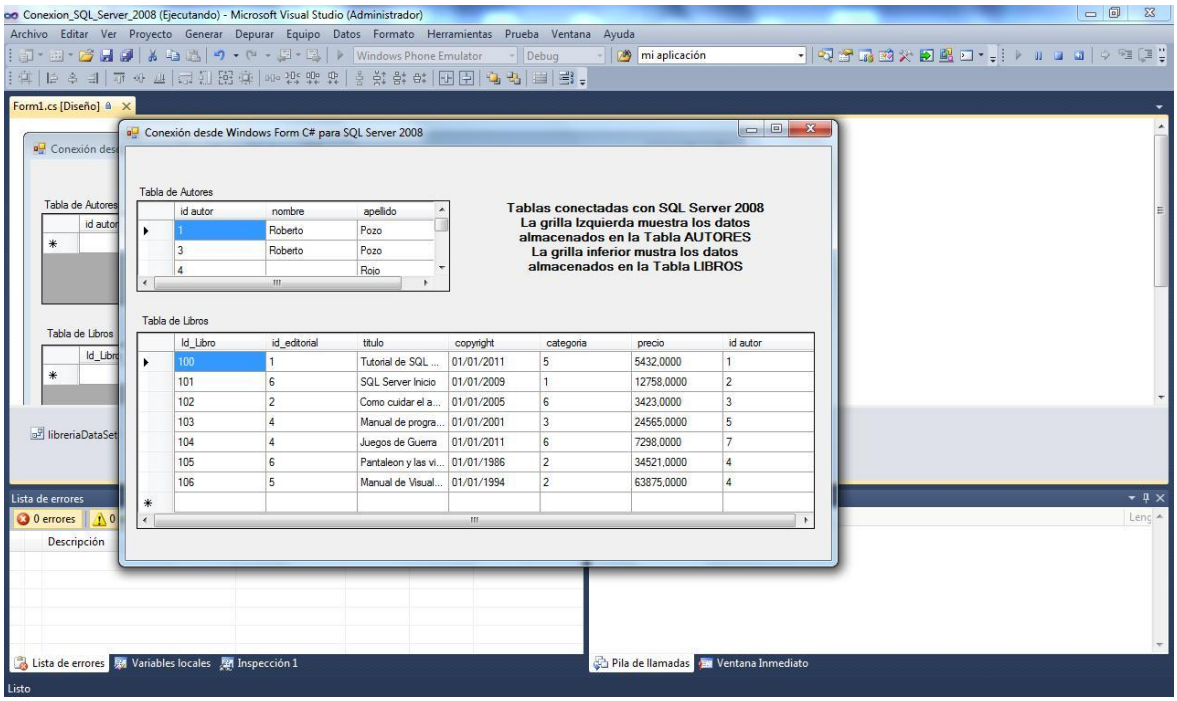


Рисунок 1.4 – Інтерфейс середовища розробки «Visual Studio» та «Windows Forms»

Отже, для проектування інтерфейсу додатку було обрано середовище розробки «Visual Studio» та середовище розробки інтерфейсів Windows Forms. Тепер необхідно обрати мову програмування, на якій буде працювати весь кістяк програми. Оскільки середовищем розробки було обрано Visual Studio, у якості розглянутих мов програмування стали C++ та C#

C++ — універсальна мова програмування високого рівня з підтримкою декількох парадигм програмування. Зокрема: об’єктно-орієнтованої та процедурної. Розроблена Б’ярном Страуструпом (англ. Bjarne Stroustrup) в AT&T Bell Laboratories (Мюррей-Хілл, Нью-Джерсі) у 1979 році та названа «C з класами». Страуструп перейменував мову у C++ у 1983 р. Базується на мові Сі. Визначена стандартом ISO/IEC 14882:2003 [12].

У 1990-х роках C++ стала однією з найуживаніших мов програмування загального призначення.

Нововведеннями C++ порівняно з C є:

– підтримка об'єктно-орієнтованого програмування через класи;
– підтримка узагальненого програмування через шаблони;
– доповнення до стандартної бібліотеки;
– додаткові типи даних;
– обробка винятків;
– простори імен;
– вбудовані функції;
– перевантаження операторів;
– перевантаження імен функцій;
– посилання і оператори управління вільно розподіленою пам'яттю.

C# - проста, сучасна об'єктно-орієнтована і типобезпечна мова програмування. C# відноситься до широко відомого сімейства мов C, і здається добре знайомим будь-кому, хто працював з C, C++, Java або JavaScript. Тут представлений огляд основних компонентів мови.

C# є об'єктно-орієнтованою мовою, але підтримує також компонентно-орієнтоване програмування. Розробка сучасних програм все більше тяжіє до створення програмних компонентів у формі автономних та самоописових пакетів, що реалізують окремі функціональні можливості. Важливою особливістю таких компонентів є модель програмування на основі властивостей, методів і подій. Кожен компонент має атрибути, що надають декларативні відомості про компонент та вбудовані елементи документації. C# надає мовні конструкції, які безпосередньо підтримують таку концепцію роботи. Завдяки

цьому C# чудово підходить для створення та застосування програмних компонентів.

Виконавши аналіз мов програмування, можна скласти порівняльну таблицю (табл. 1.2), яка показує головні відмінності порівнювальних мов.

Таблиця 1.2 – Порівняння аналогів

	C++	C#
Об’єктно орієнтована мова програмування	+	+
Високий рівень абстракцій	-	+
Наявність «Збирача сміття»	-	+
Автоматичне керування пам’яттю	-	+

В результаті порівняння для розробки програмного додатку для маркетингових досліджень було обрано об’єктно орієнтовану мову програмування C#.

Після того, як було обрано мову програмування, варто визначитись із виконанням основних алгоритмів програми. Є 2 варіанти як обраховувати та подавати аналітику.

– обчислення за допомогою математичних формул у C#;
– обчислення за допомогою декларативної мови програмування для взаємодії користувача з базами даних SQL.

Оскільки у маркетинговій статистиці іде мова про величезну кількість даних, необхідно вибрати саме той варіант, який своїм існуванням заточений на виконання подібних розрахунків. Даним інструментом є декларативна мова SQL.

SQL - простими словами, це мова програмування структурованих запитів, який використовується як ефективний спосіб збереження даних, пошуку їх частин, оновлення, вилучення з бази та видалення [13].

Головний інструмент оптимізації та обслуговування бази даних — ось, для чого потрібен SQL, хоча він і не обмежений цими цілями. Можливості обробки охоплюють команди визначення представлених, вказівки прав доступу, схеми

відносин (в тому числі, їх видалення та зміни), взаємодію з іншими мовами програмування, перевірку цілісності, завдання початку та завершення транзакцій.

Основні переваги мови SQLполягають у наступному:

– стандартність - як уже було сказано, використання мови SQL у програмах стандартизовано міжнародними організаціями;
– незалежність від конкретних СУБД – всі розповсюджені СУБД використають SQL, тому що реляційну базу даних можна перенести з однієї СУБД на іншу з мінімальними доробками;
– можливість переносу з однієї обчислювальної системи на іншу – СУБД може бути орієнтована на різні обчислювальні системи, однак додатка, створені за допомогою SQL, допускають використання як для локальних БД, так і для великих, багатокористувацьких систем;
– реляційна основа мови – SQL є мовою реляційних БД, тому вона сталі популярною тоді, коли одержала широке поширення реляційна модель подання даних. Таблична структура реляційної БД добре зрозуміла, а тому мова SQL проста для вивчення;
– можливість створення інтерактивних запитів – SQL забезпечує користувачам негайний доступ до даних, при цьому в інтерактивному режимі можна одержати результат запиту за дуже короткий час без написання складної програми;
– можливість програмного доступу до БД – мову SQL легко використати в додатках, яким необхідно звертатися до баз даних. Ті самі оператори SQL вживаються як для інтерактивного, так і програмного доступу, тому частини програм, що містять звертання до БД, можна спочатку перевірити в інтерактивному режимі, а потім вбудовувати в програму;
– забезпечення різного подання даних – за допомогою SQL можна представити таку структуру даних, що той або інший користувач буде бачити різні їхні подання. Крім того, дані з різних частин БД можуть бути скомбіновані й представлені у вигляді однієї простої таблиці, а

виходить, подання придатні для посилення захисту БД й її настроювання під конкретні вимоги окремих користувачів;

– можливість динамічної зміни й розширення структури БД - мова SQL дозволяє маніпулювати структурою БД, тим самим забезпечуючи гнучкість із погляду пристосованості БД до вимог, що змінюються, предметної області;
– підтримка архітектури клієнт-сервер – SQL – один із кращих засобів для реалізації додатків на платформі клієнт-сервер. SQL служить сполучною ланкою між взаємодіючою з користувачем клієнтською системою й серверною системою, що управляє БД, дозволяючи кожної з них зосередитися на виконанні своїх функцій.

Щодо методів розв’язання поставлених задач – за них відповідатиме декларативна мова SQL.

Найголовніша задача дипломної роботи – розробити алгоритм, за яким буде обчислюватись коофіцієнт та градації ефективності рекламної кампанії.

Зазвичай, компанії використовують єдиний показник ефективності – ціну за клієнта. Якщо вона більша за суму, яка закладена у мартетингові витрати на 1-го клієнта – канал трафіку є неефективним. Проще кажучи – чим більше ми витрачаємо за клієнта – тим гірше. Але до уваги не беруться такі показники як об’єми трафіку, ціна за лід. А це дуже важливі показники.

Візьмемо для прикладу таргетовану рекламу. Один продаж може коштувати занадто дорого для власника, але даний канал трафіку приносить 70% від усіх лідів. Відповідно не можна просто відмовитись від цього каналу просування, адже трафік звідти становить левову частку усіх продажів. Відповідно, справа може бути у менеджерах, що обробляють заявки, у рекламних матеріалах або у якості самого трафіку, а не у рекламній площадці.

Отже, було запропоновано використати коофіцієнт ефективності рекламної кампанії (КЕРК). Для його обчислення було використано такі параметри як частка від загального трафіку, сума, яку було витрачено на рекламну кампанію та конверсії, які принесла та чи інша кампанія.

Оскільки вартість ліда є другорядною, але необхідною частиною формули – було приняте рішення поділити відношення вартості ліда рекламної кампанії до середньої вартості ліда по усім кампаніям на 3.

Формула КЕРК виглядає наступним чином У формулі X – КЕРК (Коофіцієнт ефективності рекламної кампанії), p – вартість за лід , k – вартість за клієнта, c – кількість лідів:

$$X=\frac{p(\text{кампанії})}{p(\text{усіх кампанії за цей час})*3}+\frac{k(\text{кампанії})}{k(\text{усіх кампанії за цей час})}-\frac{c(\text{кампанії})}{c(\text{усіх кампанії за цей час})} \quad (1.6)$$

Сам коофіцієнт було створено з урахуванням об’ємів лідогенерації, вартості ліда та вартості клієнта. Тобто у одній формулі враховані усі основні показники, за якими маркетологи визначають чи працює рекламна кампанія ефективно, чи ні. Відношення із вартістю за лід ділиться на 3, щоб зменшити вплив на результат, адже іноді дорогий лід – має вищу конверсію (Наприклад у контекстній рекламі) [14].

Відповідно до результатів формули КЕРК було визначено градацію, за якою користувач може розшифрувати значення коофіцієнта. А саме, якщо коофіцієнт КЕРК дорівнює:

– 0-2 – рекламна кампанія працює ефективно;
– 2-3 – рекламна кампанія працює недостатньо ефективно. Занадто дорогий лід, або низька конверсія. Необхідні нетермінові зміни;
– 3-10 – рекламне кампанія є не ефективною. Необхідні термінові зміни, або повна відмова від ресурсу трафіку;

Порівняємо КЕРК із іншими відомими способами визначення ефективності рекламної кампанії.

Модель рекламної кампанії можна описати диференційним параметричним рівнянням:

$$\frac{dN}{dt}=[a_1(t)+a_2(t)N(t)](N_0-N(t)), \quad (1.7)$$

де dN/dt – швидкість зміни кількості споживачів, які дізналися про товар та мають намір і кошти його придбати; t – час, який минув із початку рекламної кампанії; N(t) – кількість уже проінформованих клієнтів; a1(t) – інтенсивність рекламної кампанії, яка визначається витратами на рекламу в даний момент часу; N0 – загальна кількість платоспроможних покупців.

Параметр a2(t) – ступінь інформованості покупців між собою (кожен, хто дізнався про товар, так чи інакше розповсюджує інформацію про нього серед необізнаних). Ця величина може бути встановлена анкетуванням.

Головна перевага даної формули у тому, що вона досить детально описує ефективність рекламної кампанії, беручи до уваги усі її складові. Головним недоліком ж слугує її складність та необхідність анкетуваня.

Ще одна формула, за якою можна визначити чи працює рекламна кампанія, чи ні – це коефіцієнт ROI. Головна його перевага – простота, та відображення кінцевого продукту маркетингового відділу – продажів. Але в той же час, для збору даних по цьому коефіцієнту потрібно більше часу, у формулі не враховуються об’єми лідогенерації та повна залежність від відділу продажу.

Виконавши аналіз формул для аналізу ефективності рекламної кампанії, можна скласти порівняльну таблицю (табл. 1.3), яка показує головні відмінності порівнювальних формул.

Отже, внаслідок дослідження способів визначення ефективності роботи рекламної кампанії, та провівши пряме порівняння необхідних переваг було обрано використовувати власний метод КЕРК (Коофіцієнт ефективності рекламної кампанії).

Таблиця 1.3 – Порівняння формул для визначення ефективності рекламної кампанії

	КЕРК	Визначення шляхом диференційного рівняння	Коофіцієнт ROI
Швидкий та простий у виконанні	+	-	+

Продовження таблиці 1.3

	КЕРК	Визначення шляхом диференційного рівняння	Коофіцієнт ROI
Незалежність від відділу продажів	+	+	-
Відсутність хибної аналітики за умовою малих об’ємів даних	+	+/-	-
Відсутність необхідності у додаткових даних	+	-	+

1.4 Постановка задач

Завданням бакалаврської дипломної роботи є розробка програми для маркетингових досліджень «Marketing Analytics».

Програма буде створюватися з використанням мови програмування C# на платформі Windows Forms для покращення навичок програмування та вивчення техніки розробки програм для аналізу даних. Інтерфейс програми необхідно розробляти на платформі Windows Forms.

Для зручності використання буде розроблено головний екран, на якому будуть розміщені усі необхідні кнопки для навігації у програмі.

Програма матиме можливість імпортувати та експортувати ліди, що дасть змогу швидко та безперебійно використовувати програму у реальному аналізі даних.

Потрібно забезпечити користувача про:

– Аналітику роботи відділу продажу;
– Аналітику джерела лідогенерації завдяки воронкам;
– Аналітику джерела лідогенерації завдяки маркетинговим витратам, коофіцієнту ефективності рекламної кампанії, коофіцієнту повернень інвестицій.
– Для зручності користування програмою поставленні наступні задачі:
– вибір дати огляду без використання календарів;
– зміна графіків натисканням 1-ї кнопки;

– окремі вікна, які звужують фокус користувача.

Для запуску програми апаратне забезпечення повинне відповідати мінімальним вимогам:

Комп’ютер з процесором від Pentium;

– 1 Гб оперативної пам’яті;

– внутрішня графіка Intel;

– периферія: клавіатура, миша;

– ОС Windows 10;

Розмір дискового простору, що займає програма, 30.5 Мб.

1.5 Висновки

1) Аналіз аналогів показав, що розроблюваний сервіс аналітики є необхідним рішенням за такими аспектами: швидкість, зручність у використанні, здатність самостійно вручну заводити дані, спроможність визначати ефективність рекламної кампанії;

2) Розглянуто теоретичні основи математичної статистики.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз завдання

Розробка бакалаврської дипломної роботи розпочинається з визначення завдання, а саме - з точного формулювання вимог до програмного забезпечення, що розробляється. Для цього потрібно сформулювати вимоги до програми та її інтерфейсів, визначити особливості, функції, загальні принципи побудови програмних продуктів, побудувати базу даних.

Програма для проведення маркетингових досліджень «Marketing Analysis» була створення для швидкої допомоги у аналізі бізнесу. Відповідно, інтерфейс програми повинен бути максимально простим та зрозумілим. Також, є інші вимоги, які необхідно взяти до уваги, та відтворити у програмному додатку.

Для розробки програмного забезпечення для проведення маркетингових досліджень необхідно:

- розробити користувацький інтерфейс;
- розробити базу даних;
- визначити і використати графічні елементи ;
- спроектувати користувацький інтерфейс;
- розробити основні програмні модулі додатку;
- протестувати розроблений програмний продукт.

2.2 Розробка бази даних

База даних (БД) — це організована структура, яка призначена для зберігання, зміни та обробки взаємозалежної інформації, переважно великих обсягів [15].

БД використовують для динамічних сайтів з великими обсягами (інтернет-магазин, портал, корпоративний сайт).

База даних — це певний набір даних, які пов'язані між собою спільною ознакою або властивістю, та впорядковані, наприклад, за алфавітом.

Об'єднання великої кількості даних в єдину базу дає змогу для формування безлічі варіації групування інформації — особисті дані клієнта, історія замовлень, каталог товарів та будь-що інше.

Головною перевагою БД є швидкість внесення та використання потрібної інформації. Завдяки спеціальним алгоритмам, які використовуються для баз даних, можна легко знаходити необхідні дані всього за декілька секунд. Також в базі даних існує певний взаємозв'язок інформації: зміна в одному рядку може спричинити зміни в інших рядках — це допомагає працювати з інформацією простіше і швидше.

Бази даних дають змогу зберігати інформацію, що виглядає як зв'язані між собою таблиці. Саме в БД зберігаються вся необхідна та корисна інформація для функціонування сайту (клієнтські дані, прайс-лист, список товарів).

Щоб створити запит до бази даних часто використовують Structured Query Language. SQL дає змогу додавати, редагувати та видаляти інформацію, що міститься у таблицях.

Під час програмування сайтів використовують різні системи управління БД. До основних СУБД, відносять:

– об'єктно-реляційна система управління базами даних Oracle Database;
– вільна система управління базами даних PostgreSQL;
– система керування базами даних Microsoft SQL Сервер;
– вільна система управління базами даних MySQL;

Такі системи управління відрізняються централізованою обробкою запитів, забезпечують надійність, доступність та безпеку БД.

Найбільш популярною системою управління є MySQL, вона дає зручний доступ для управління БД та підтримує велику кількість таблиць різних типів.

Для програми для проведення маркетингових досліджень було вирішено використати базу даних з однією сущністю – лідом. Саме за лідом можна індивідуальні маркетингові дані, які в майбутньому можна використати для обрахунків.

Атрибути вищеназваного об’єкта:

Lead (<Id>, PIB, Source, Category, Status, Responsible, City, Lead_Price, Creation_Date).

Оскільки база даних складається з одного об’єка, необхідність у ER моделях відсутня.

2.3 Розробка алгоритмів роботи додатку

Алгоритм — набір інструкцій, які описують порядок дій виконавця, щоб досягти результату розв'язання задачі за скінченну кількість дій; система правил виконання дискретного процесу, яка досягає поставленої мети за скінченний час. Для комп'ютерних програм алгоритм є списком деталізованих інструкцій, що реалізують процес обчислення, який, починаючи з початкового стану, відбувається через послідовність логічних станів, яка завершується кінцевим станом.

В даній бакалаврській роботі розробляється програма, що дасть змогу отримувати основні статистичні дані рекламних кампаній. Серед основних статистичних даних можна виділити:

- CPM (Cost per mile) – Вартість за 1000 показів оголошення
- CPC (Cost per click) – Вартість за 1 клік по оголошенню
- CTR (Click through rate) – Клікабельність оголошення. Обчислюється у відсотках
- CAC (Customer Acquisition Cost) – Вартість клієнта.
- ROI (Return of investments) – Повернення інвестицій. Обчислюється у відсотках.
- CPC (Cost per lead) – Ціна за зацікавленого користувача, який виконав цільову дію.

Для наглядного і зручного представлення даного набору характеристик слід використовувати засоби графічного представлення даних та зводити одержані дані до таблиць та графіків.

Для реалізації поставленого завдання перш за все необхідно виконати наступні кроки:

1. Отримання основних даних, на основі яких будуть обчислюватись вищевказані метрики;
2. Процес обчислення метрик;
3. Отримання дат для графічного виведення даних;
4. Забезпечення графічного виведення даних.

Наведений вище перелік операцій формулює алгоритм роботи розробленої програми. Сам алгоритм роботи програми зображено на блок схемі (рис. 2.1):

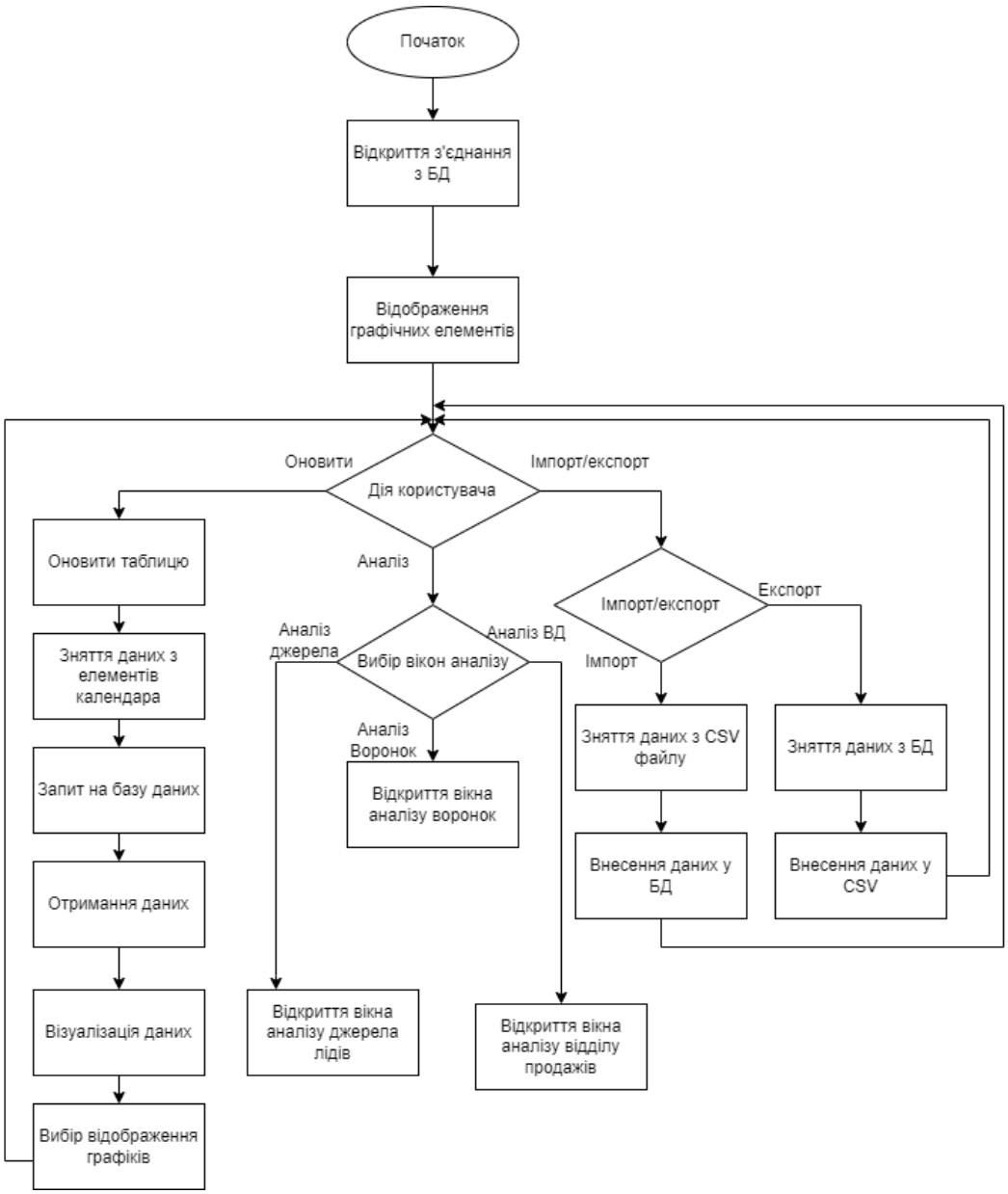


Рисунок 2.1 Алгоритм додатку

Алгоритм роботи вікна, де здійснюється аналіз джерел лідів зображено на рисунку 2.2.

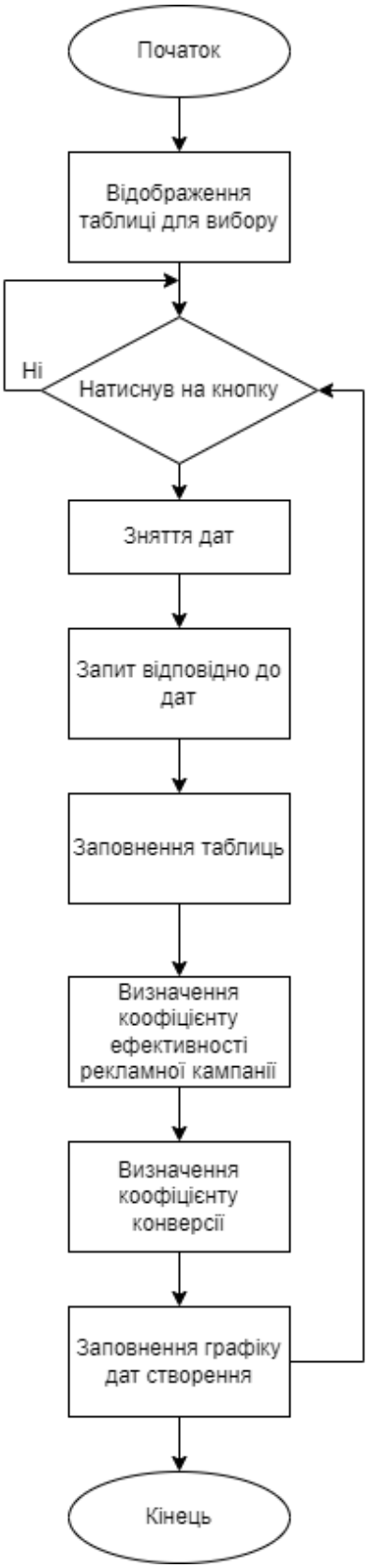


Рисунок 2.2 Алгоритм вікна аналізу джерел лідів

Алгоритм роботи вікна, де здійснюється аналіз роботи відділу продажів зображено на рисунку 2.3.

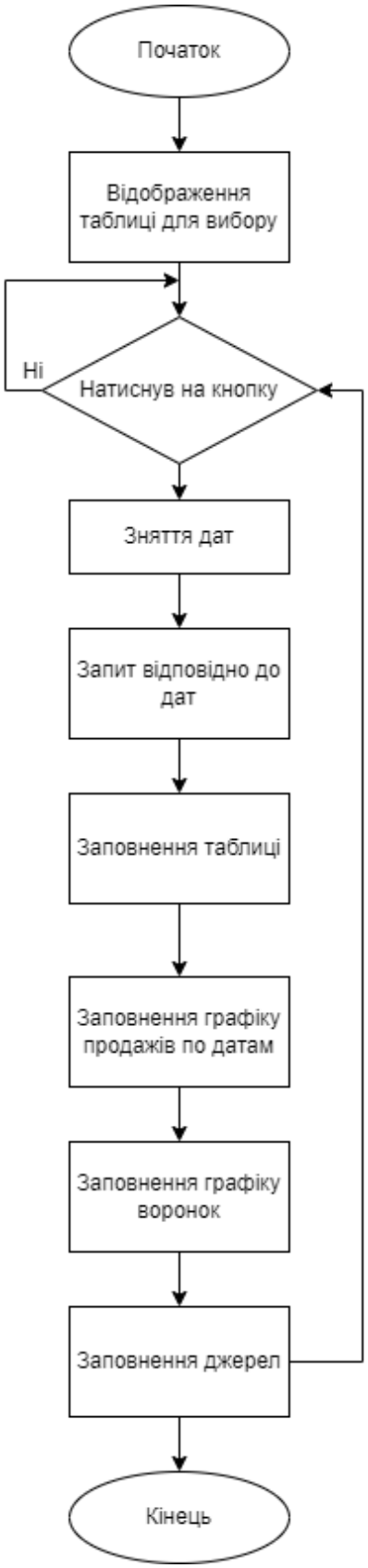


Рисунок 2.3 Алгоритм вікна аналізу роботи ВП

Алгоритм роботи вікна, де здійснюється аналіз воронки джерела трафіку зображено на рисунку 2.4.

Рисунок 2.4 Алгоритм вікна аналізу воронок

Розроблений алгоритм дозволяє визначити достатній перелік статистичних даних маркетингової кампанії, та повністю задовольняє поставленим до нього

вимогам. Даний алгоритм показав свою ефективність в знаходженні статистичних даних маркетингових кампаній.

2.4 Висновки

Аналіз показав, що обраний алгоритм є найефективнішим серед усіх існуючих алгоритмів, які спроможні обраховувати статистику маркетингових кампаній.

3 РОЗРОБКА ПРОГРАМИ ДЛЯ ПРОВЕДЕННЯ МАРКЕТИНГОВИХ ДОСЛІДЖЕНЬ

3.1 Проектування інтерфейсу користувача

Інтерфейс користувача – це сукупність засобів, за допомогою яких користувач спілкується з різними пристроями (з комп’ютером або побутовою технікою) або іншим складним інструментарієм (системою). Інтерфейс користувача - це такий різновид інтерфейсів, в якому з одного боку - людина, з іншого - машина (пристрій, програмне забезпечення). За визначенням Національного банку стандартизованих науково-технічних термінів, інтерфейс користувача - це комплекс апаратних і програмних засобів, що забезпечує взаємодію користувача з комп'ютером [16].

Інтерфейс користувача комп’ютерного додатку включає:

- засоби відображення інформації, відображувану інформацію, формати і коди;
- командні режими, мову «користувач–інтерфейс»;
- пристрої та технології введення-виведення;
- діалоги, взаємодію та транзакції між користувачем та комп’ютером, зворотній
- зв’язок з користувачем;
- підтримку прийняятя рішень в конкретній предметній області;
- порядок використання програми і документації на неї.

Інтерфейс користувача часто розуміють лише як зовнішній вигляд програми. Однак насправді користувач сприймає через нього всю програму в цілому, тобто таке розуміння є надто вузьким.

Для виконання проектування інтерфейсу користувача складено список необхідних вікон:

1. Вікно «Головне вікно» - слугує навігатором по усій програмі. Складається з 3-х областей, з яких здійснюється навігація програмою.
2. Вікно «Створення ліда вручну» - слугує контструктором для створення нового ліда, та його додання у базу даних.

3. Вікно «Видалення ліда» - слугує вікном для видалення непотрібних лідів:
4. Вікно «Про програму» - представляє користувачеві інформацію про програму, та про її автора.
5. Вікно «Аналіз роботи відділу продажів» - слугує вікном, яке відображає реальну аналітику роботи відділу продажу, а саме кожного з менеджерів.
6. Вікно «Аналіз джерел по воронках» - слугує вікном, яке відображає ситуацію по просуненню лідів по воронці у певний проміжок часу.
7. Вікно «Аналіз джерел по ефективності» - слугує вікном, яке відображає ситуацію по якості трафіку з того чи іншого джерела.

На основі висунутих вимог розроблено схематичні зображення вікон програми для маркетингових досліджень

Схему вікна «Головне вікно» наведено на рисунку 3.1.

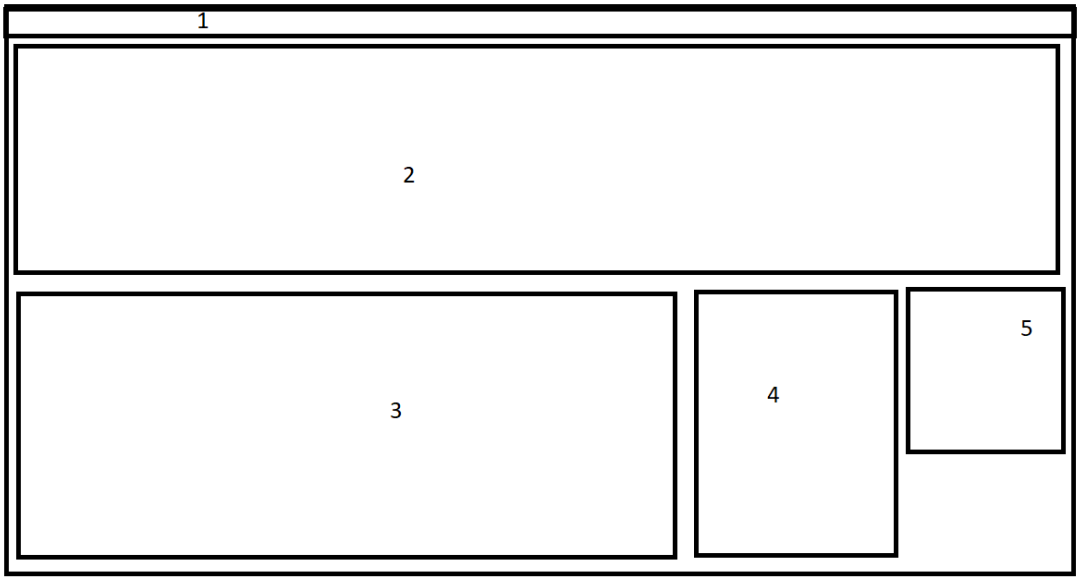


Рисунок 3.1 – Схема вікна «Головне вікно»

Опис елементів схеми відповідно до їх нумерації на рисунку 3.1:

1. Верхнє меню для навігації у програмі
2. Графік, який відображає ситуацію за завантаженою БД.
3. Таблиця, яка демонструє БД.

4. Елементи навігації у програмі.
5. Елементи керування таблицею та графіком на головному вікні

Схема вікна «Створення ліда вручну» зображена на рисунку 3.2.

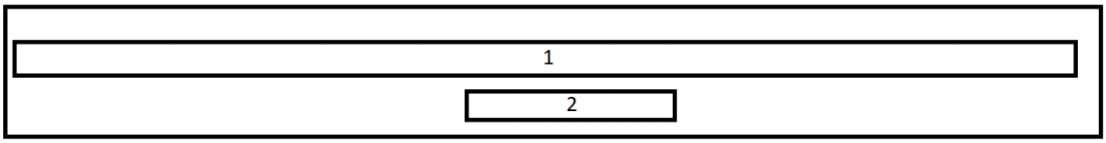


Рисунок 3.2 – Схема вікна «Створення ліда вручну»

Опис елементів схеми відповідно до їх нумерації на рисунку 3.2:

1. Поля, у які користувач заповнює дані про лід..
2. Кнопка, після натискання якої дані про лід, введені у полі 1 додаються у БД.

Схема вікна «Видалення ліда» зображена на рисунку 3.3.

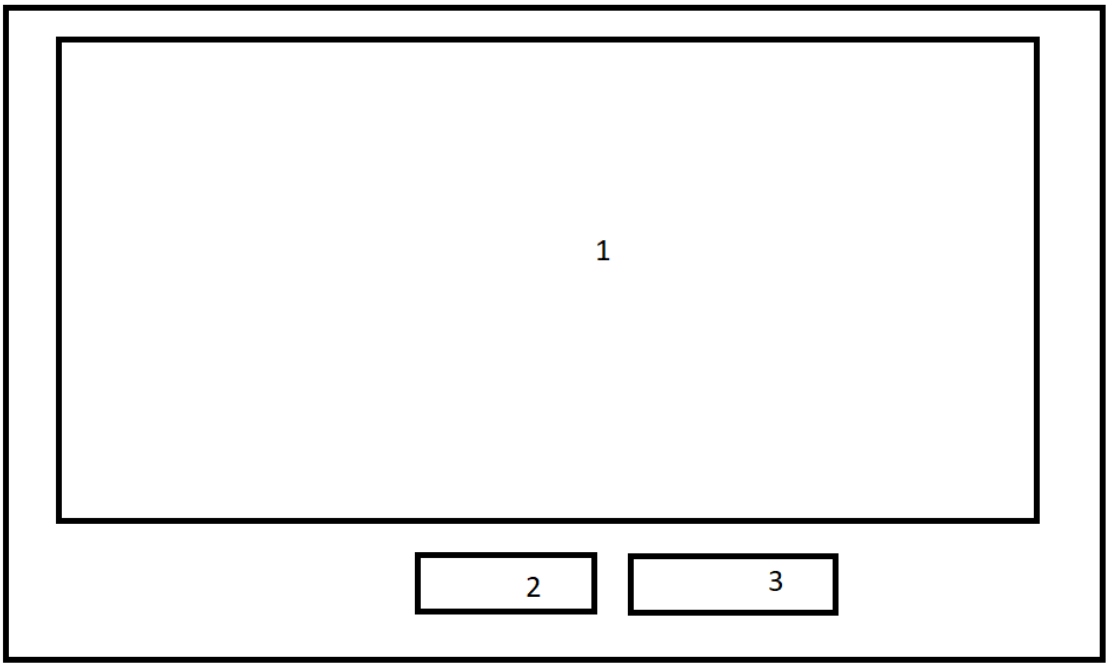


Рисунок 3.3 – Схема вікна «Видалення ліда»

Опис елементів схеми відповідно до їх нумерації на рисунку 3.3:

1. Таблиця, за якою користувач має могу звіритись під час видалення ліда.
2. Поле для вводу даних користувача, якого необхідно видалити.
3. Кнопка «Видалити», яка відшукує необхідний лід та видаляє його.

Схему «Про програму» зображено на рисунку 3.4

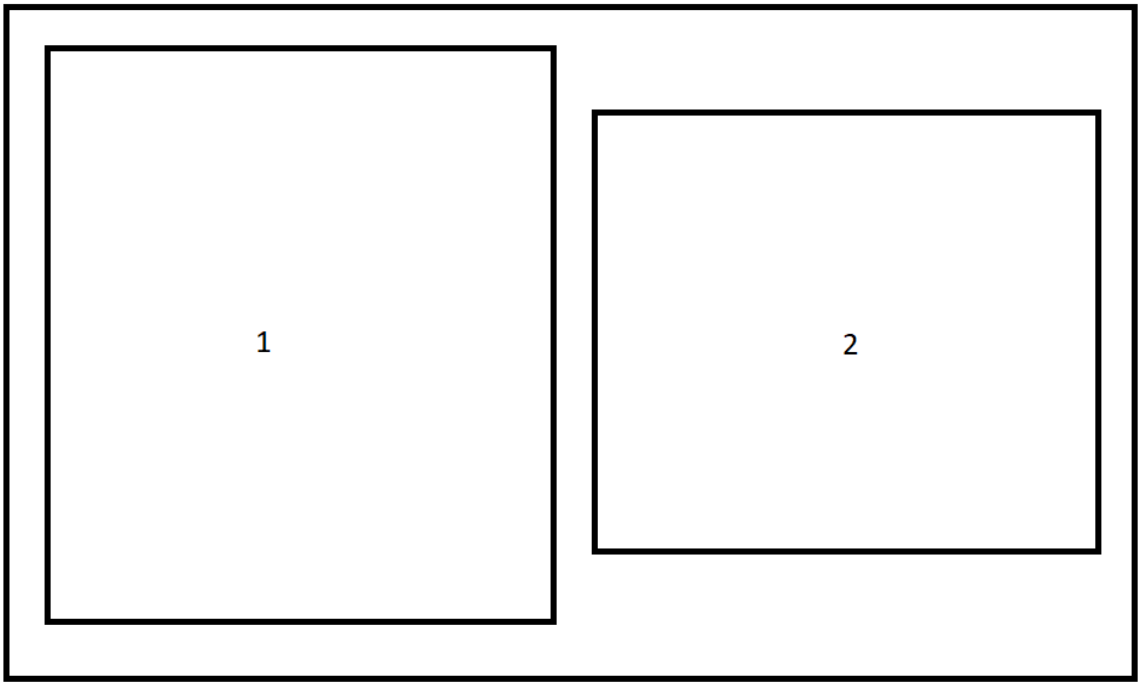


Рисунок 3.4 – Схема вікна «Про програму»

Опис елементів схеми відповідно до їх нумерації на рисунку 3.4:

1. Зображення.
2. Опис програми.

Схему «Аналіз роботи відділу продажів» зображено на рисунку 3.5

Опис елементів схеми відповідно до їх нумерації на рисунку 3.5:

1. Елемент для вибору конкретного менеджера з продажу.
2. Графік «Кількість лідів за проміжок часу».
3. Графік «Ліди та їх частки».
4. Графік «Ситуація у воронці».
5. Таблиця «Воронка менеджера»

6. Таблиця «Джерела для менеджера»
7. Таблиця «Дата обробки»

Схема вікна «Аналіз джерел по воронках» на рисунку 3.6.

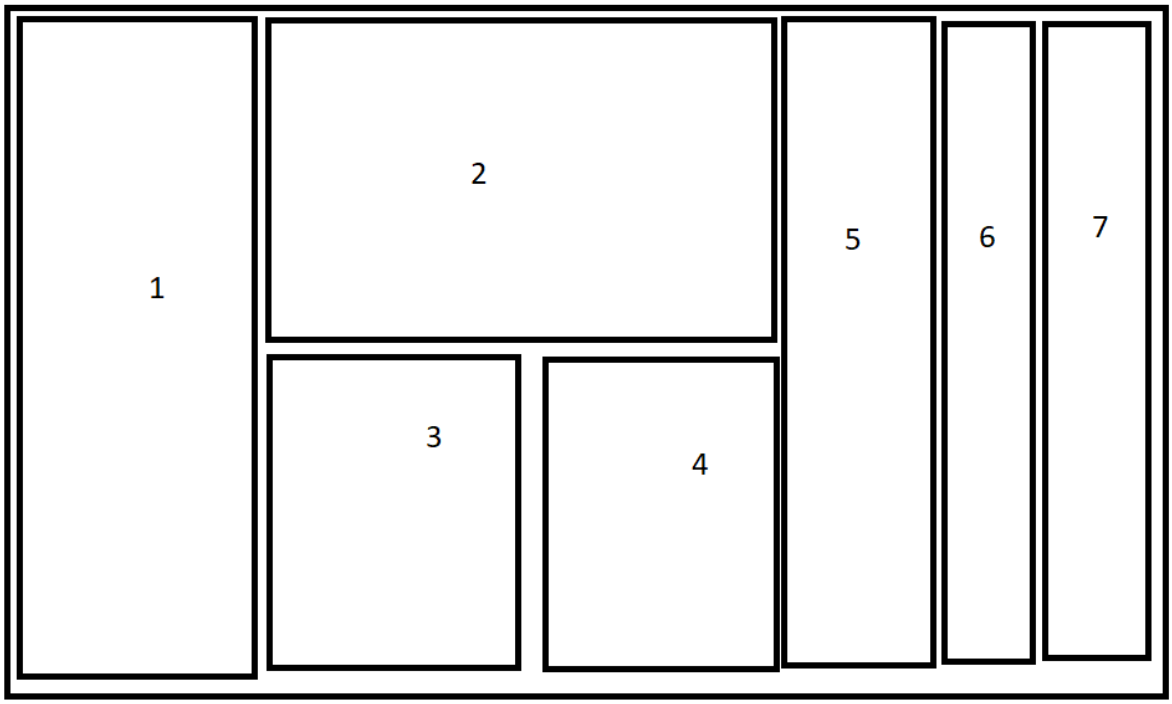


Рисунок 3.5 - Схема вікна «Аналіз роботи відділу продажів»

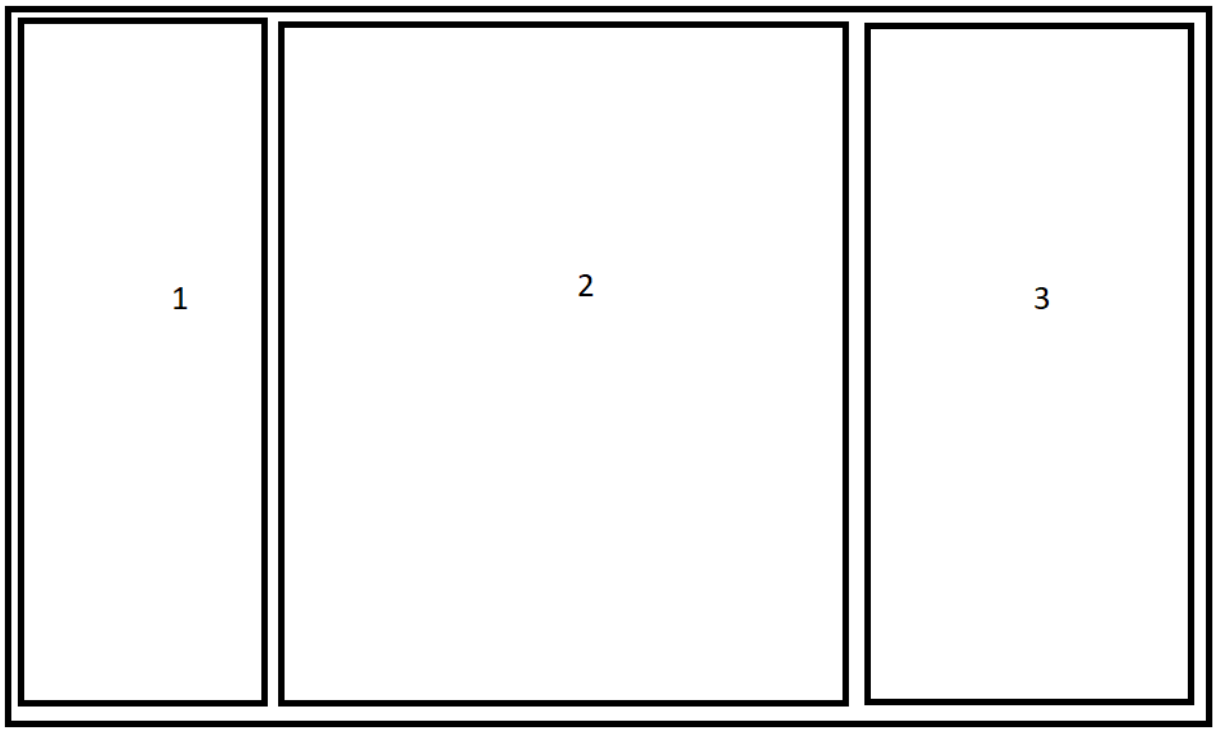


Рисунок 3.6 - Схема вікна «Аналіз джерел по воронках»

Опис елементів схеми відповідно до їх нумерації на рисунку 3.6:

1. Елемент для вибору конкретного джерела.
2. Візуалізація воронки продажу.
3. Таблиця з даними про ліди.

Схема вікна «Аналіз джерел по воронках» на рисунку 3.7.

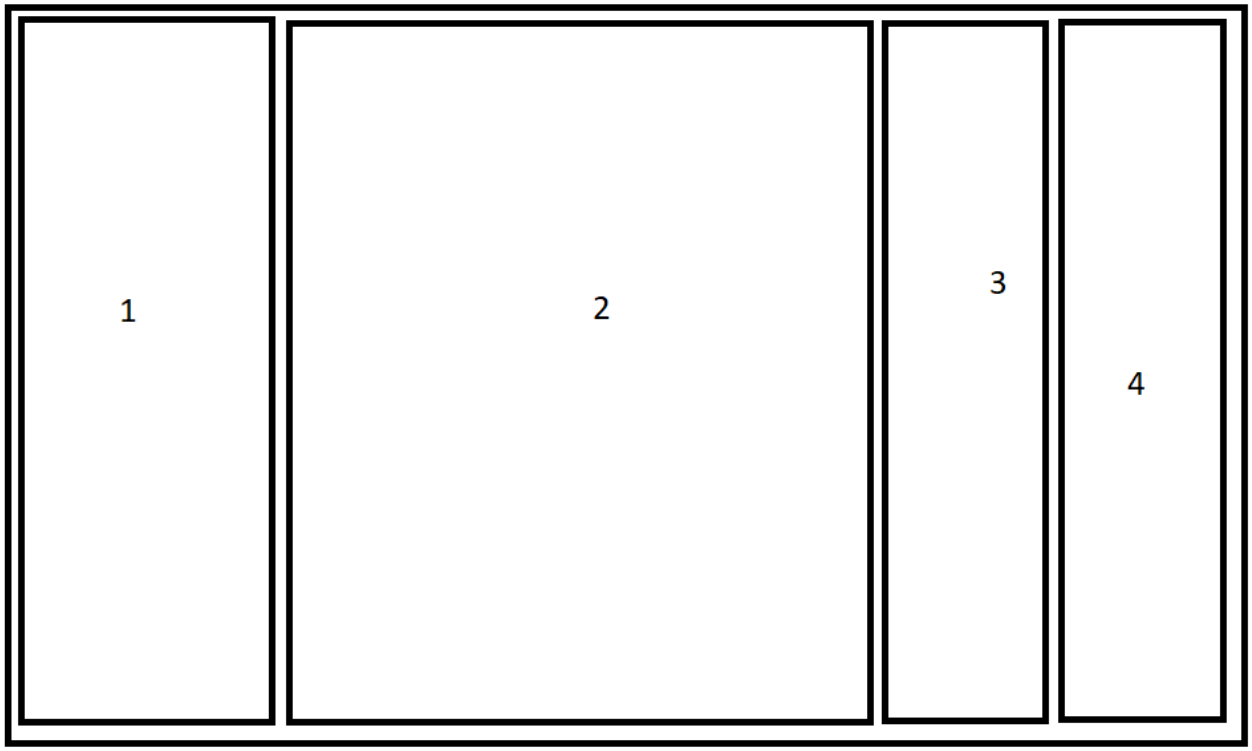


Рисунок 3.7 - Схема вікна «Аналіз джерел по ефективності»

Опис елементів схеми відповідно до їх нумерації на рисунку 3.7:

1. Елемент для вибору конкретного джерела.
2. Візуалізація статистичних даних.
3. Таблиця з даними про ліди.
4. Таблиця з даними про конверсії.

Розроблений інтерфейс користувача є макетом, який має бути заповнений графічними матеріалами, щоб забезпечити простоту використання та покращити зрозумілість користувача.

3.2 Розробка інтерфейсу програми

Важливою частиною створення інтерфейсу є простота у його створенні. Чим мінімалістичніше дизайн – тим краще як для розробника, так і для користувача. Для розробника – це значно менше часу, витраченого на дизайн. Для користувача – зручність використання.

Саме тому для проектування та створення програми для маркетингових досліджень було використано середовище для побудови інтерфейсу Windows Forms.

Windows Forms - це платформа інтерфейсу користувача для створення класичних додатків Windows. Вона забезпечує один з найефективніших способів створення класичних програм за допомогою візуального конструктора в Visual Studio. Такі функції, як розміщення візуальних елементів керування шляхом перетягування, полегшують створення класичних додатків.

У Windows Forms можна розробляти графічно складні програми, які легко розгортати, оновлювати, і з якими зручно працювати як в автономному режимі, так і в мережі. Програми Windows Forms можуть отримувати доступ до локального обладнання та файлової системи комп'ютера, на якому працює програма [17].

Розробка інтерфейсу програму починається з моделювання схем та ескізів майбутніх віком. Саме цю роботу було зроблено у минулому підрозділі.

І тепер, коли усі схеми погоджено, можна починати проектувати інтерфейс програмного додатку

Розглянемо процес розробки графічного зображення на прикладі усіх вікон.

На рисунку 3.8 наведено візуалізацію процесу створеної головного вікна програмного додатку.

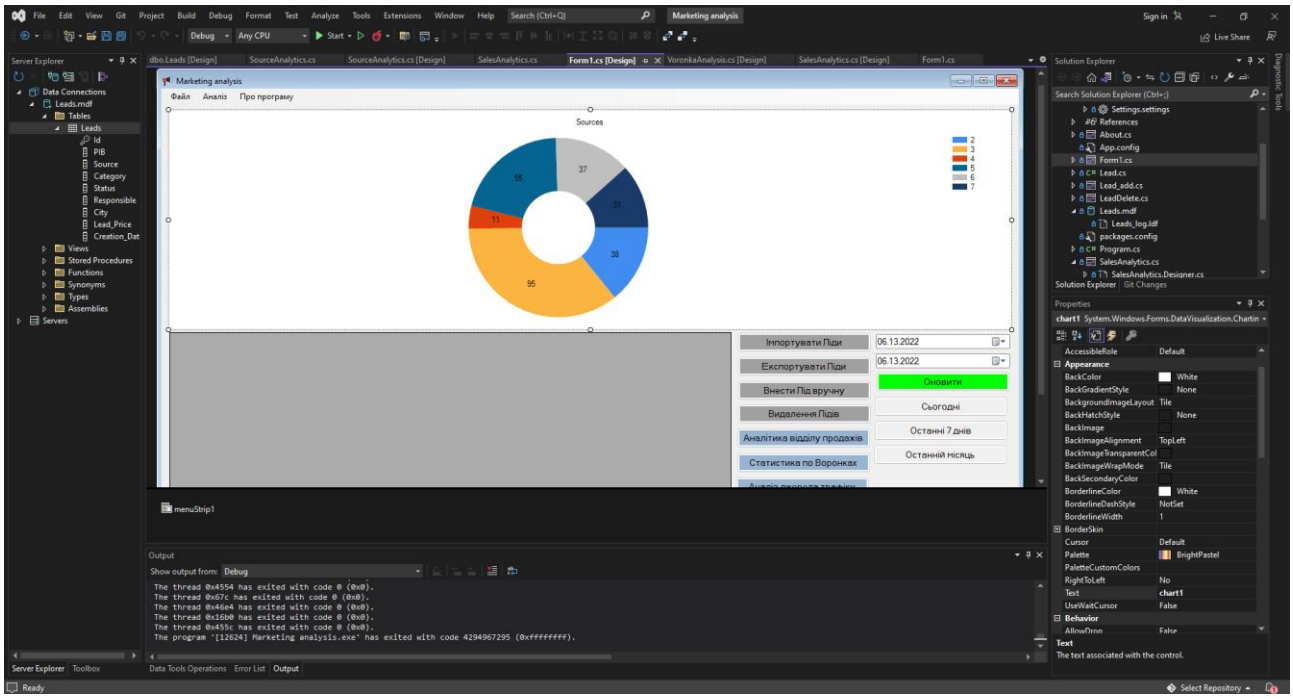


Рисунок 3.8 – Створення головного вікна за схемою у підрозділі 3.2

На рисунку 3.9 наведено вигляд вікна «Створення лідів вручну». Завдяки попередньому плануванню вдалося розробити юзер-френдлі інтерфейс, який є простим та водночас функціональним.

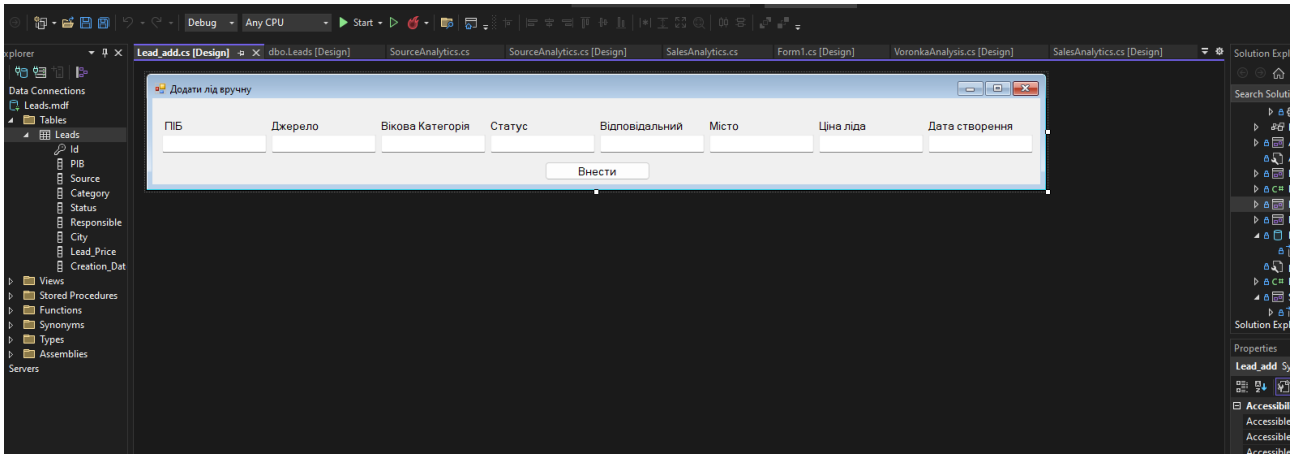


Рисунок 3.9 – Моделювання вікна «Створення лідів вручну».

На рисунку 3.10 наведено вигляд вікна «Видалення ліда».

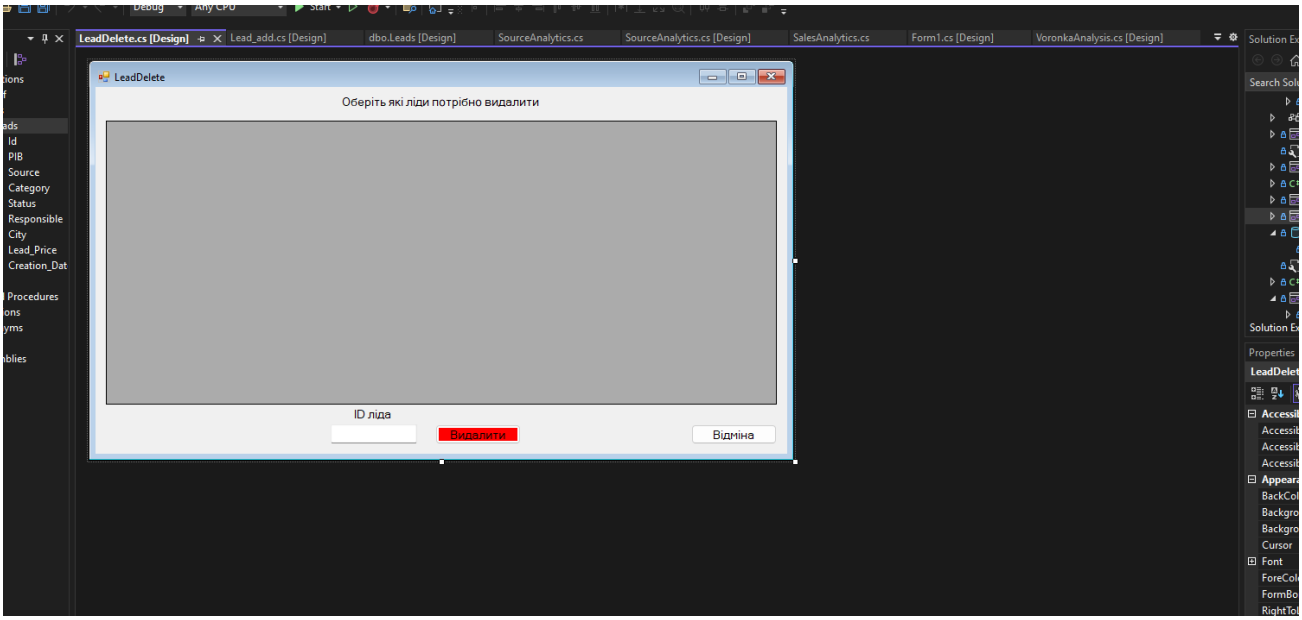


Рисунок 3.10 – Моделювання вікна «Видалення ліда»

На рисунку 3.11 наведено вигляд вікна «Про програму».

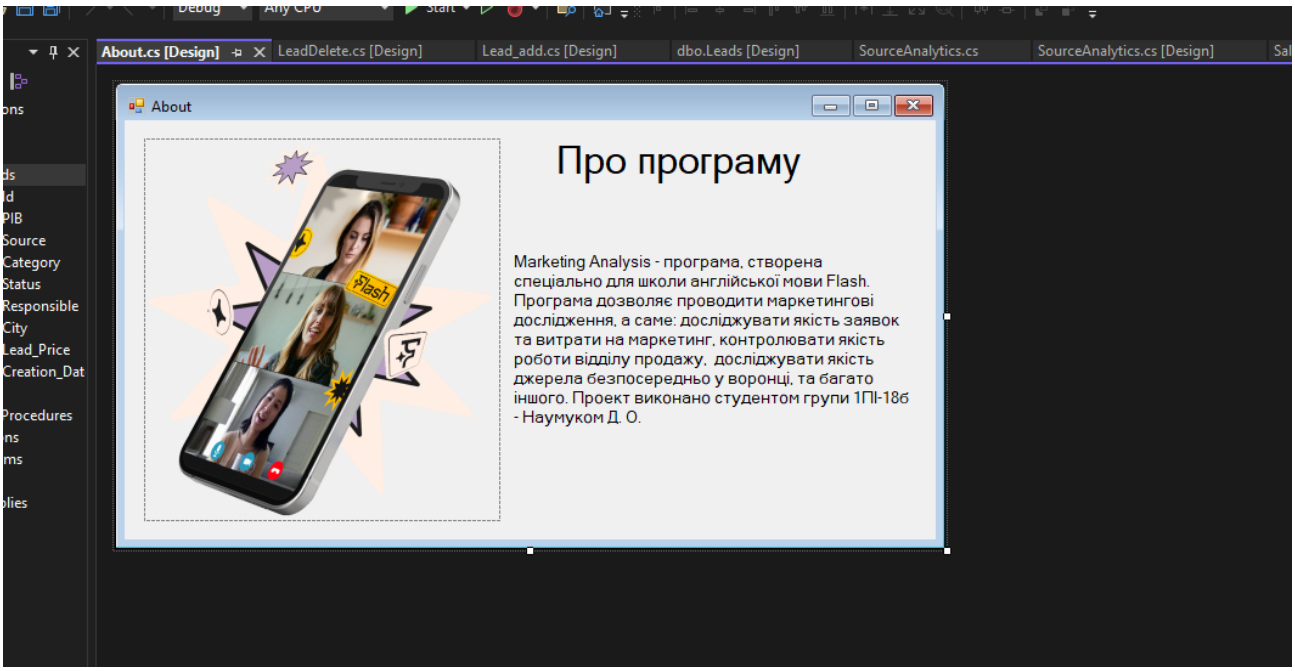


Рисунок 3.11 – Моделювання вікна «Про програму»

На рисунку 3.12 наведено вигляд вікна «Аналіз роботи відділу продажів».

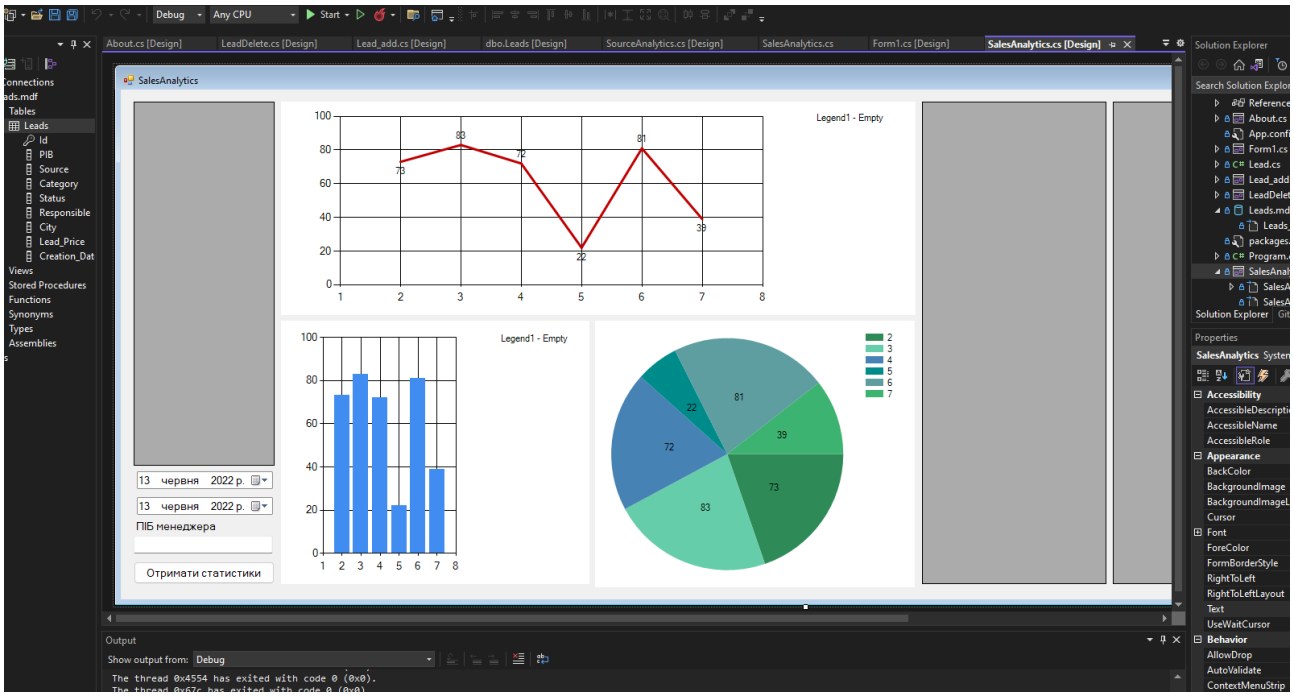


Рисунок 3.12 – Моделювання вікна «Аналіз роботи відділу продажів»

На рисунку 3.13 наведено вигляд вікна «Аналіз джерел по воронках».

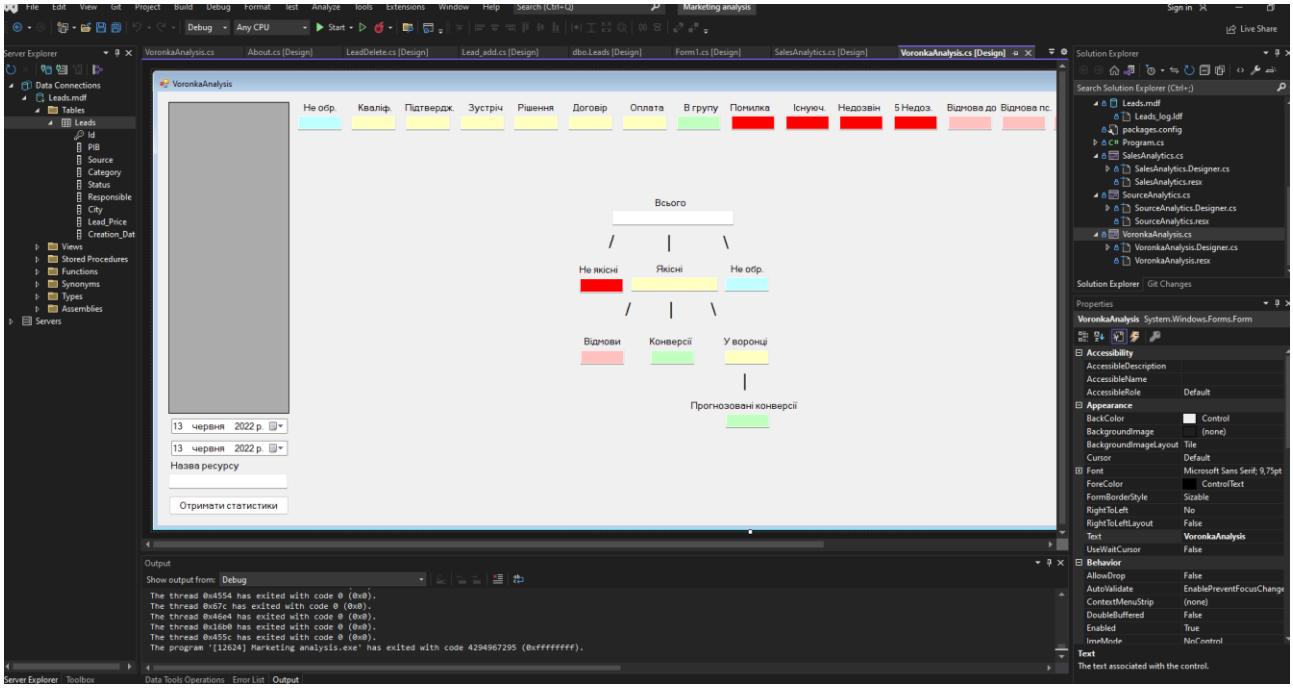


Рисунок 3.13 – Моделювання вікна «Аналіз джерел по воронках»

На рисунку 3.14 наведено вигляд вікна «Аналіз джерел по ефективності».

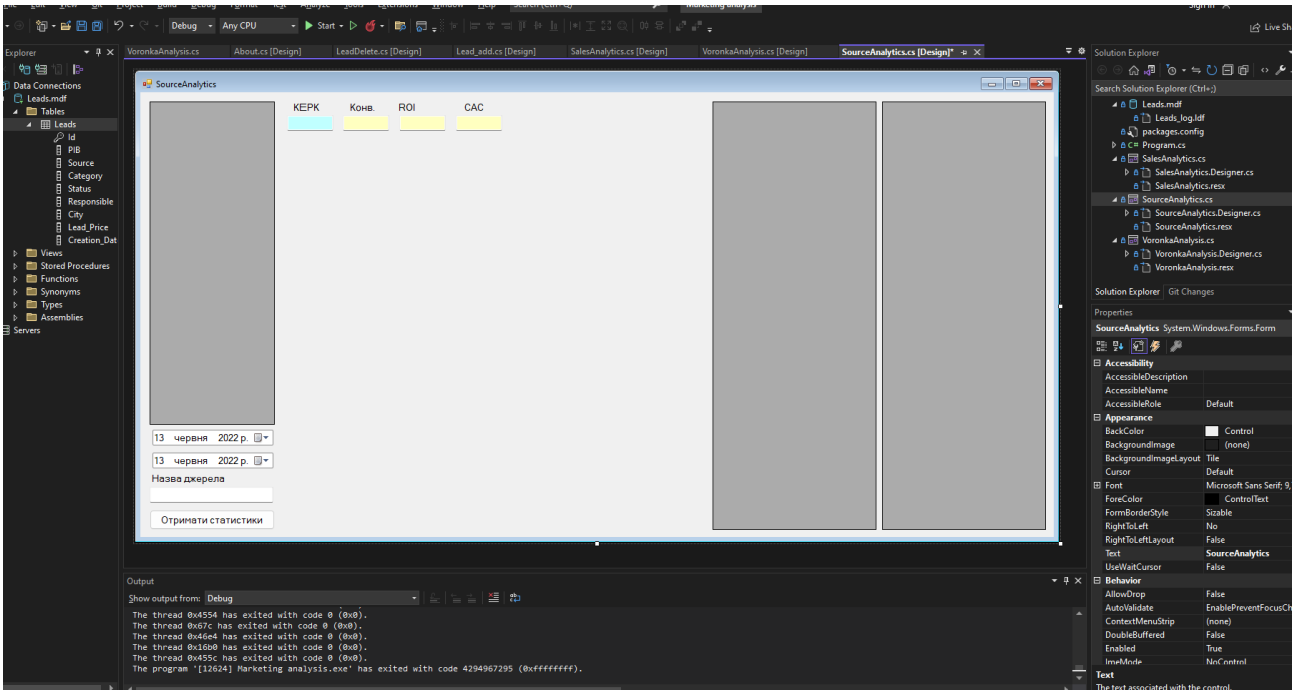


Рисунок 3.14 – Моделювання вікна «Аналіз джерел по ефективності»

Після створення макетів інтерфейсу програмного додатку, можна запрограмувати кожен елемент, який було розміщено у вікні.

3.3 Розробка основного функціоналу програми

Основним вікном програми для проведення маркетингових досліджень є клас Form1.cs, який відповідає за головне вікно програми, з якого і здійснюється уся навігація.

Перед тим як почати роботу із базою даних, необхідно імпортувати CSV файл зі всіма даними. Імпорт файлів виконується у методі ReadCSVFile, Частину коду, що реалізовує імпорт лідів у БД наведено на рисунках 3.15 та 3.16.

Оновлення таблиць та графіків головного вікна виконується після вибору необхідних дат та натискання кнопки «Оновити», що викликає метод chartUpdate. Завдяки даному методу програма бере вказані дати, та шукає ліди, які були створені поміж даних дат. А завдяки ChartUpdate ці дані потрапляють на графік також. Частину коду, що реалізовує оновлення таблиць наведено на рисунках 3.17 та 3.18.

```
2 references
private static void ReadCSVFile()
{
    var csvFileDescription = new CsvFileDescription()
    {
        FirstLineHasColumnNames = true,
        IgnoreUnknownColumns = true,
        SeparatorChar = ';',
        UseFieldIndexForReadingData = false,
        TextEncoding = Encoding.UTF8

    };

    var csvContext = new CsvContext();
    var leads = csvContext.Read<Lead>("Leads.csv", csvFileDescription);

    foreach(var lead in leads)
    {
        SqlCommand command = new SqlCommand(
        $"INSERT INTO [Leads] (PIB, Source, Category, Status, Responsible, " +
        $"City, Lead_Price, Creation_Date) VALUES" +
        $" (N'{lead.PIB}', N'{lead.Source}', N'{lead.Category}', N'{lead.Status}', " +
        $"N'{lead.Responsible}', N'{lead.City}', N'{lead.Lead_Price}', N'{lead.Creation_Date}');", conn);
        command.ExecuteNonQuery();
    }
}
```

Рисунок 3.15 – Код методу ReadCsvFile

```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    ReadCSVFile();
}
```

Рисунок 3.16 – Виклик методу ReadCsvFile

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    DateTime date1 = dateTimePicker2.Value.Date;
    DateTime date2 = dateTimePicker1.Value.Date;
    SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS LeadsCount, " +
        $"AVG(Lead_Price) AS AVGLeadPrice, SUM (Lead_Price) AS LeadPriceSum FROM Leads " +
        $"WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND " +
        $"'{date2.ToString("MM.dd.yyyy")}' GROUP BY Source;", conn);

    DataSet dataSet = new DataSet();
    adapter.Fill(dataSet);

    dataGridView1.DataSource = dataSet.Tables[0];

    ChartUpdate();
}
```

Рисунок 3.17 - Частина коду методу button1.click

Експорт лідів описано в методі Export. Частину коду, в якому реалізовано експорт у csv файл наведено на рисунку 3.19.

```
5 references
private void ChartUpdate()
{
    chart1.Series["Sources"].Points.Clear();
    for (int i = 0; i < dataGridView1.RowCount-1; i++)
    {
        if (dataGridView1.Rows[i].Cells[1].Value == null)
        {
            Console.WriteLine("NULL");

        }
        Console.WriteLine(dataGridView1.Rows[i].Cells[0].Value);
        Console.WriteLine(dataGridView1.Rows[i].Cells[1].Value);


        chart1.Series["Sources"].Points.AddXY(dataGridView1.Rows[i].Cells[0].Value, dataGridView1.Rows[i].Cells[1].Value);
    }


}
```

Рисунок 3.18 - Частина коду методу ChartUpdate();

```
private void Export()
{
    conn.Close();
    StreamWriter file;
    file = new StreamWriter("Leads.csv", false);
    using (conn)
    {
        try
        {
            var query = "SELECT * FROM Leads";
            SqlCommand command =
                 new SqlCommand(query, conn);
            conn.Open();
            SqlDataReader SQLreader =
                 command.ExecuteReader();

            file.WriteLine(
            @"""PIB"";""Source"";""Category"";""Status"";""Responsible"";""City"";""Lead_Price"";""Creation_Date""");
            if (SQLreader.HasRows)
            {
                while (SQLreader.Read())
                {
                    file.WriteLine(@""" +
                         SQLreader.GetValue(1).ToString() +
                         @""";""" +
                         SQLreader.GetValue(2).ToString() +
                         @""";""" +
                         SQLreader.GetValue(3).ToString() +
                         @""";""" +
                         SQLreader.GetValue(4).ToString() +
                         @""";""" +
                         SQLreader.GetValue(5).ToString() +
                         @""";""" +
                         SQLreader.GetValue(6).ToString() +
                         @""";""" +
                          SQLreader.GetValue(7).ToString() +
                         @""";""" +
                         SQLreader.GetValue(8).ToString() +
                         @"""");
                }
            }
            else
                file.WriteLine(
                     "No one row is in \"Leads\" table");
            file.WriteLine("End of file");
            file.Close();
            SQLreader.Close();
            conn.Dispose();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            if (conn.State == ConnectionState.Open)
                conn.Close();
            conn.Dispose();
        }
    }
}
```

Рисунок 3.19 - Частина коду методу Export

Кнопки «Сьогодні», «Останні 7 днів», «Останній місяць» дозволяють швидко змінювати дати на шаблонні. Частину кодів, що описує взаємодію із dateTimePicker зображена на рисунку 3.20.

```
1 reference
private void button12_Click(object sender, EventArgs e)
{
    dateTimePicker1.Value = DateTime.Now;
    dateTimePicker2.Value = DateTime.Now;
}

1 reference
private void button11_Click(object sender, EventArgs e)
{
    dateTimePicker2.Value = DateTime.Now.AddDays(-7);
    dateTimePicker1.Value = DateTime.Now;
}

1 reference
private void button10_Click(object sender, EventArgs e)
{
    var now = DateTime.Now;
    dateTimePicker2.Value = new DateTime(now.Year, now.Month, 1);
    dateTimePicker1.Value = new DateTime(now.Year, now.Month + 1, 1);
}
```

Рисунок 3.20 – Частина коду методів, що дозволяють швидко взаємодіяти із dateTimePicker

Також у головному вікні є кнопки, які дозволяють змінювати відображення графіків. Частини коду, що дозволяють переглянути графіки за різними критеріями зображено на 3.21, 3.22 та 3.23.

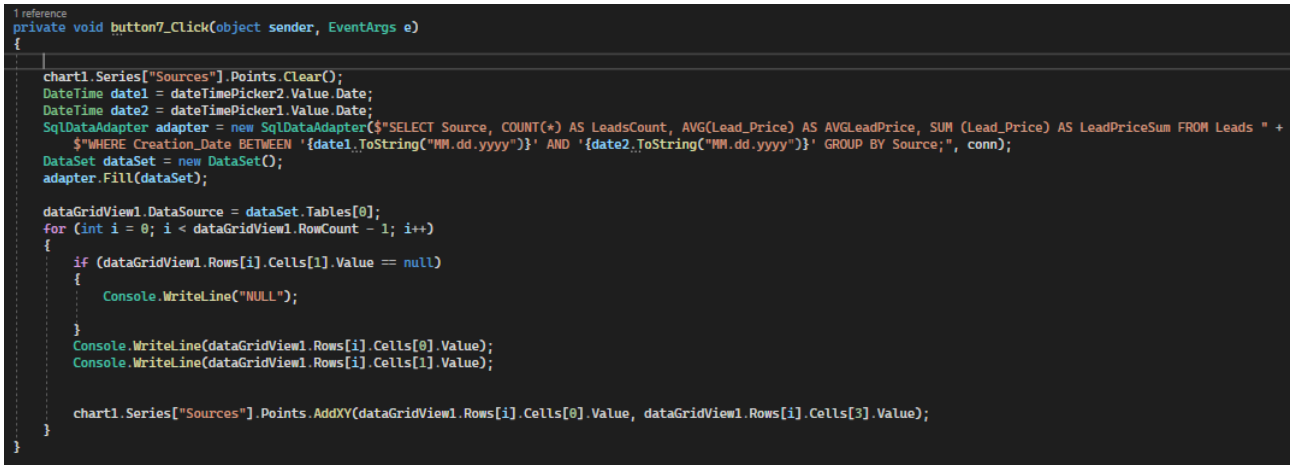
```
1 reference
private void button4_Click_1(object sender, EventArgs e)
{
    chart1.Series["Sources"].Points.Clear();
    DateTime date1 = dateTimePicker2.Value.Date;
    DateTime date2 = dateTimePicker1.Value.Date;
    SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS LeadsCount, " +
        $"AVG(Lead_Price) AS AVGLeadPrice, SUM (Lead_Price) AS LeadPriceSum FROM Leads " +
        $"WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND " +
        $"'{date2.ToString("MM.dd.yyyy")}' GROUP BY Source;", conn);

    DataSet dataSet = new DataSet();
    adapter.Fill(dataSet);

    dataGridView1.DataSource = dataSet.Tables[0];

    ChartUpdate();
}
```

Рисунок 3.21 – Частина коду, що дозволяє перемикнутись на режим перегляду графіку «Ліди по джерелам»



```
1 reference
private void button7_Click(object sender, EventArgs e)
{
    chart1.Series["Sources"].Points.Clear();
    DateTime date1 = dateTimePicker2.Value.Date;
    DateTime date2 = dateTimePicker1.Value.Date;
    SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS LeadsCount, AVG(Lead_Price) AS AVGLeadPrice, SUM (Lead_Price) AS LeadPriceSum FROM Leads " +
        $"WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}' GROUP BY Source;", conn);
    DataSet dataSet = new DataSet();
    adapter.Fill(dataSet);

    dataGridView1.DataSource = dataSet.Tables[0];
    for (int i = 0; i < dataGridView1.RowCount - 1; i++)
    {
        if (dataGridView1.Rows[i].Cells[1].Value == null)
        {
            Console.WriteLine("NULL");

        }
        Console.WriteLine(dataGridView1.Rows[i].Cells[0].Value);
        Console.WriteLine(dataGridView1.Rows[i].Cells[1].Value);


        chart1.Series["Sources"].Points.AddXY(dataGridView1.Rows[i].Cells[0].Value, dataGridView1.Rows[i].Cells[3].Value);
    }
}
```

Рисунок 3.22 – Частина коду, що дозволяє перемикнутись на режим перегляду графіку «Витрати на маркетинг»

```
1 reference
private void button13_Click(object sender, EventArgs e)
{
    chart1.Series["Sources"].Points.Clear();
    DateTime date1 = dateTimePicker2.Value.Date;
    DateTime date2 = dateTimePicker1.Value.Date;
    SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS ConversionsCount, AVG(Lead_Price) AS AVGLeadPrice FROM Leads " +
        $"WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}' AND Status='V grupu' GROUP BY Source;", conn);

    DataSet dataSet = new DataSet();
    adapter.Fill(dataSet);

    dataGridView1.DataSource = dataSet.Tables[0];

    ChartUpdate();
}
```

Рисунок 3.23 – Частина коду, що дозволяє перемикнутись на режим перегляду графіку «Кількість продажів»

3.4 Висновки

У третьому розділі було спроектовано інтерфейс користувача програми для проведення маркетингових досліджень.

Розроблено інтерфейс для взаємодії із користувачем. Для розробки інтерфейсу було обрано середовище створення інтерфейсів Windows Forms.

Виконано розробку програми для проведення маркетингових досліджень, описано розробку її основних функцій.

4 ТЕСТУВАННЯ ДОДАТКУ ТА РОЗРОБКА РЕКОМЕНДАЦІЙ КОРИСТУВАЧУ

4.1 Вибір методів тестування програмного забезпечення

Тестування програмного забезпечення - це метод перевірки відповідності фактичного програмного продукту очікуваним вимогам, також необхідний, щоб переконатися, що продукт не містить дефектів. Має на увазі виконання попередньо визначених алгоритмів з використанням ручних або автоматизованих інструментів для оцінки одного або декількох властивостей, що цікавлять. Метою тестування є виявлення помилок, прогалин або відсутніх вимог, заданих на етапі проектування продукту [18].

Існують різні способи, які можна використовувати для тестування програмного забезпечення.

Методика тестування без будь-яких знань про внутрішню роботу програми називається «чорною скринькою». Тестер не звертає уваги на архітектуру системи та не має доступу до вихідного коду. Як правило, при виконанні тесту з «чорною скринькою» тестер буде взаємодіяти з інтерфейсом системи, надаючи вхідні дані та аналізуючи виходи, не знаючи, як і де обробляються входи [19].

Переваги методу тестування «Чорна скринька»:

– добре підходить і ефективний для великих сегментів коду;
– кодовий доступ не потрібний;
– чіткий поділ перспективи користувача з погляду розробника з допомогою певних ролей;
– велика кількість помірно кваліфікованих тестувальників може протестувати програму без будь-яких знань про реалізацію, мову програмування або операційні системи.

Недоліки методу тестування «Чорна скринька»:

– обмежене покриття, оскільки насправді виконується лише обрана кількість тестових сценаріїв;
– неефективне тестування, тому що тестер тільки має обмежені знання про додаток;

– сліпе охоплення, оскільки тестер не може орієнтуватися на певні сегменти коду або області помилок;
– тестові приклади важко розробити.

Перевірка білої скриньки – це докладне дослідження внутрішньої логіки та структури коду. Тестування з використанням білої скриньки також називається тестуванням скла або відкритим тестуванням. Щоб виконати тестування білої скриньки у програмі, тестер повинен знати внутрішню роботу коду [20].

Тестер повинен заглянути всередину вихідного коду та з'ясувати, який пристрій/блок коду поводиться некоректно.

Переваги методу тестування «Біла скринька»:

– оскільки тестер знає вихідний код, дуже легко дізнатися, який тип даних може допомогти в ефективному тестуванні програми;
– це допомагає в оптимізації коду;
– додаткові рядки коду можуть бути видалені, що може призвести до прихованих дефектів;
– завдяки знанням тестера про код, максимальне охоплення досягається при написанні сценарію сценарію.

Недоліки методу тестування «Біла скринька»:

– у зв'язку з тим, що для тестування білих ящиків потрібен кваліфікований тестер, витрати збільшуються;
– іноді неможливо заглянути в кожен куточок та кут, щоб виявити приховані помилки, які можуть створювати проблеми, оскільки багато шляхів будуть неперевірені;
– важко підтримувати тестування білих ящиків, оскільки для цього потрібні спеціалізовані інструменти, такі як аналізатори коду та інструменти налагодження.

Тестування сірій скринці – це метод тестування програми з обмеженим знанням внутрішньої роботи програми. При тестуванні програмного забезпечення фраза, чим більше ви знаєте, тим краще переносить масу під час тестування програми.

Освоєння домену системи завжди дає тестеру перевагу над кимось із обмеженими знаннями домену. На відміну від тестування чорної скриньки, де тестер тестує тільки інтерфейс користувача програми; при тестуванні в сірій скринці тестер має доступ до проектної документації та бази даних. Маючи ці знання, тестер може підготувати найкращі тестові дані та сценарії тестування при складанні плану тестування.

Переваги методу тестування «Сіра скринька»:

– пропонує комбіновані переваги тестування чорної скриньки та білої скриньки, де це можливо;
– тестувальники сірої скриньки не покладаються на вихідний код; натомість вони покладаються на визначення інтерфейсу та функціональні специфікації;
– на основі наявної обмеженої інформації, тестер сірої скриньки може розробити відмінні сценарії тестування, особливо щодо протоколів зв'язку та обробки даних;
– тест виконується з погляду користувача, а чи не дизайнера.

Недоліки методу тестування «Сіра скринька»:

– оскільки доступ до вихідного коду недоступний, можливість пройти через код та зону тестування обмежена;
– тести можуть бути надмірними, якщо розробник програмного забезпечення вже виконав тестовий приклад;
– тестування всіх можливих вхідних потоків нереально, оскільки для цього знадобиться необґрунтована кількість часу; тому багато програмних шляхів будуть неперевірені.

Внаслідок порівняння сильних та слабких сторін методів тестування було проаналізовано переваги та недоліки методів тестування «чорної скриньки», «білої скриньки» та «сірої скринки». В результаті було обрано метод тестування «чорної скриньки» для тестування програмного забезпечення з урахуванням мети програми.

4.2 Тестування розробленої програми

Після запуску програми користувача зустрічає головне вікно програми, у якому відображається основні статистики та елементи навігації (рисунок 4.1).

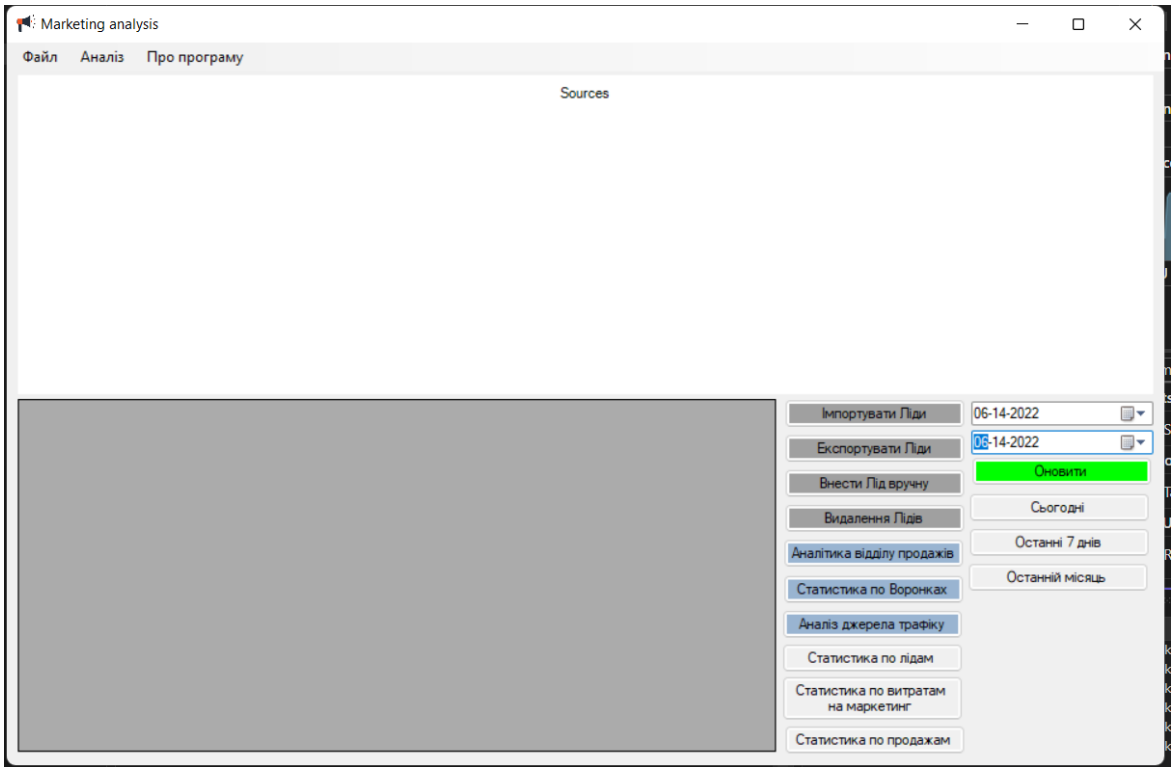


Рисунок 4.1 – Старт роботи програми та головне вікно

Для того, щоб відобразити статистики та таблиці – необхідно обрати дати створення лідів та натиснути кнопку «Оновити». Для прикладу візьмемо період з 1-го травня по 14-те червня (рисунок 4.2).

Після оновлення даних, користувач може здійснювати навігацію по основній інформації, що була отримана в наслідок оновлення. Однією з таких дій є сортування таблиці за певним елементом. Для цього, користувачу потрібно натиснути на пункт, сортування за яким його цікавить. У прикладі було використано сортування в алфавітному порядку за стовпцем «Source» (Рисунок 4.3).

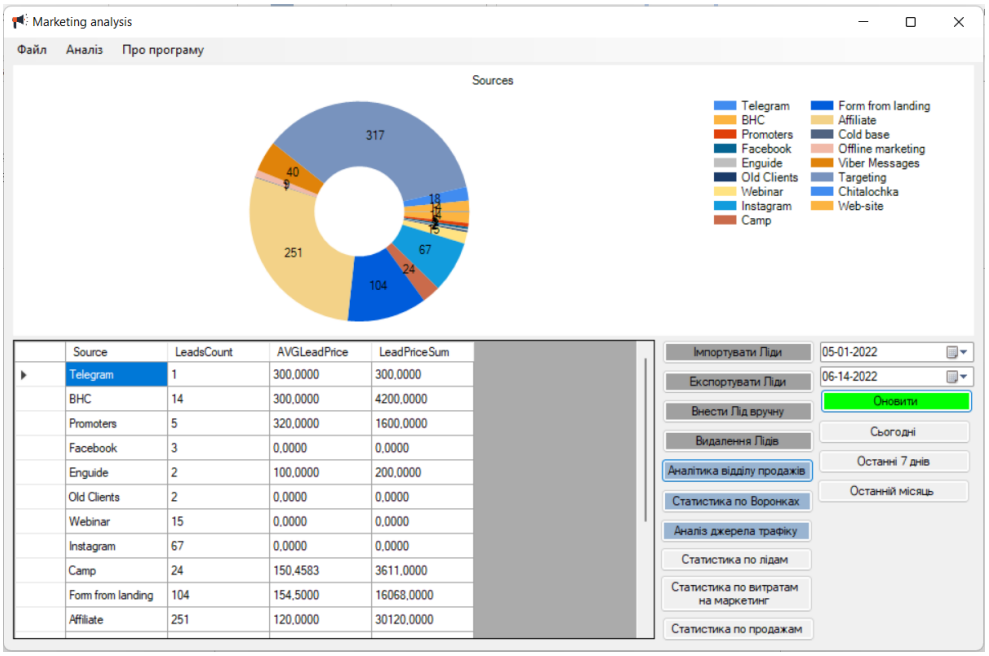


Рисунок 4.2 – Оновлення даних у головному вікні

Також, користувач має змогу вибрати параметри, за якими будуть відображатись графіки. Для прикладу, було обрано режим відображення «Статистика по продажам». Після натискання кнопки таблицю було змінено на більш релевантну, а графік змінив режим відображення, і тепер вказує користувачеві на кількість продажів (Рисунок 4.4).

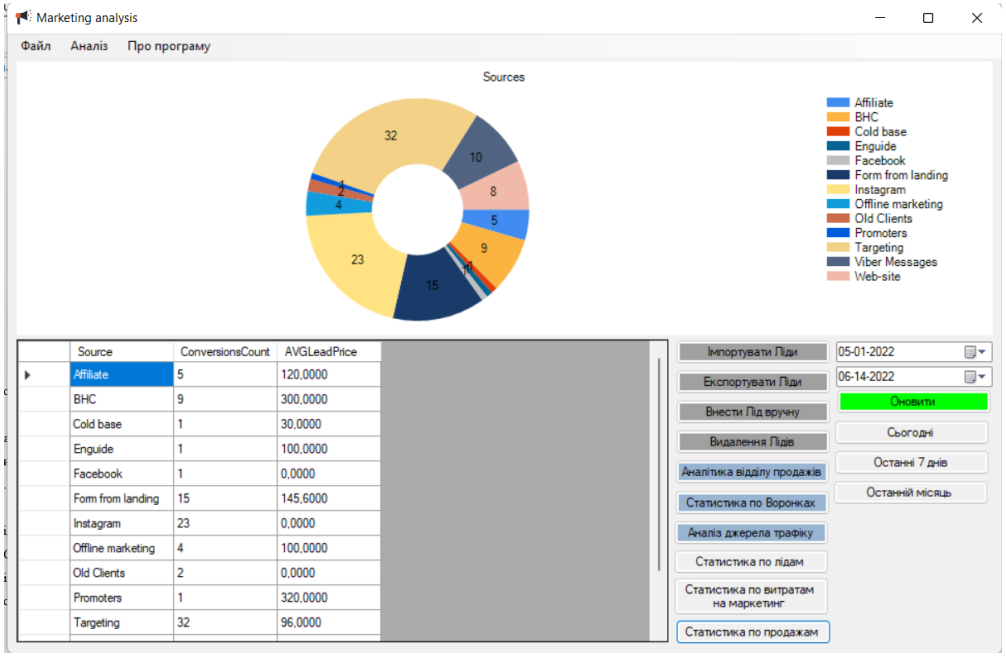


Рисунок 4.3 – Режим відображення «Статистика по продажам» у головному вікні

Існує ще один режим відображення статистичних даних. А саме, відображення маркетингових витрат. Після натискання кнопки таблицю було змінено на більш релевантну, а графік змінив режим відображення, і тепер вказує користувачеві на маркетингові витрати (Рисунок 4.4).

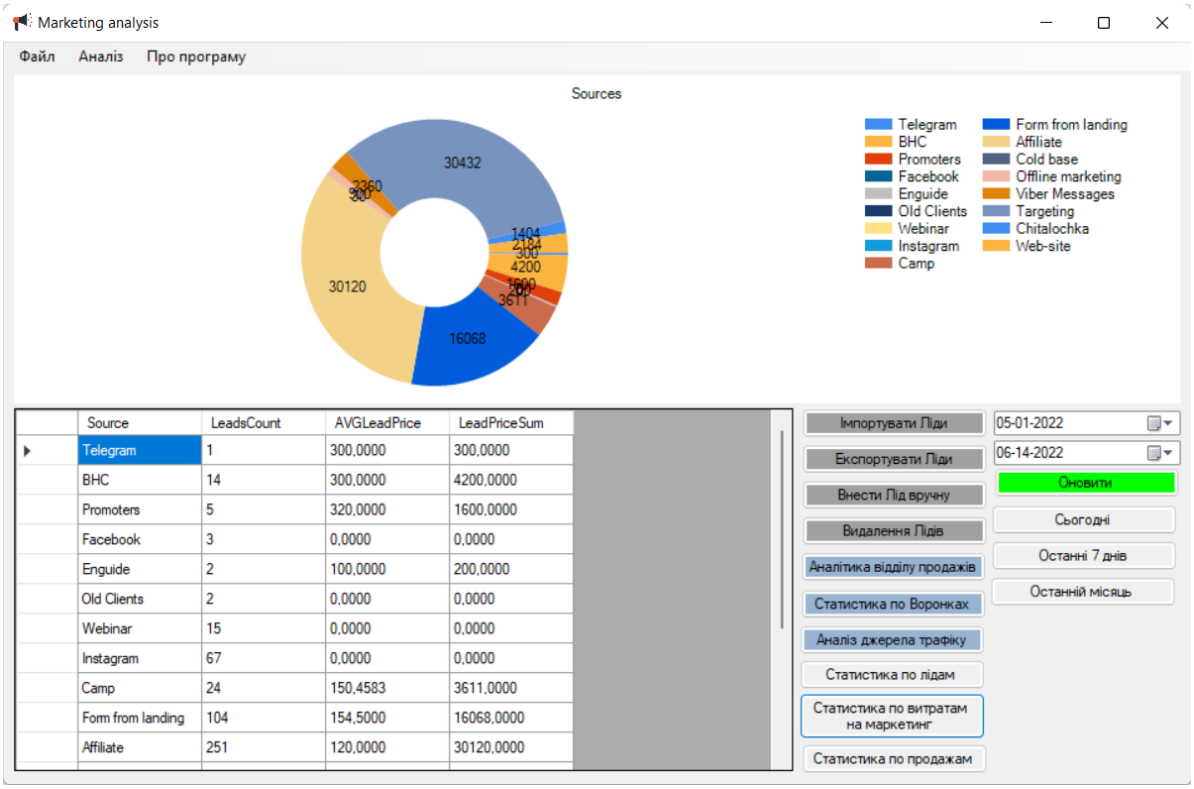


Рисунок 4.4 – Режим відображення «Маркетингові витрати» у головному вікні

Користувач має змогу дізнатись відомості про програму та її розробника. Для цього йому потрібно натиснути на кнопку «Про програму» у верхньому меню. Після натискання на кнопку з’явиться вікно «Про програму» (Рисунок 4.5).

Перейдемо до функції внесення ліда вручну. Для цього у головному вікні натискаємо кнопку «Додати лід вручну» після чого відображається вікно внесення ліда (Рисунок 4.6).

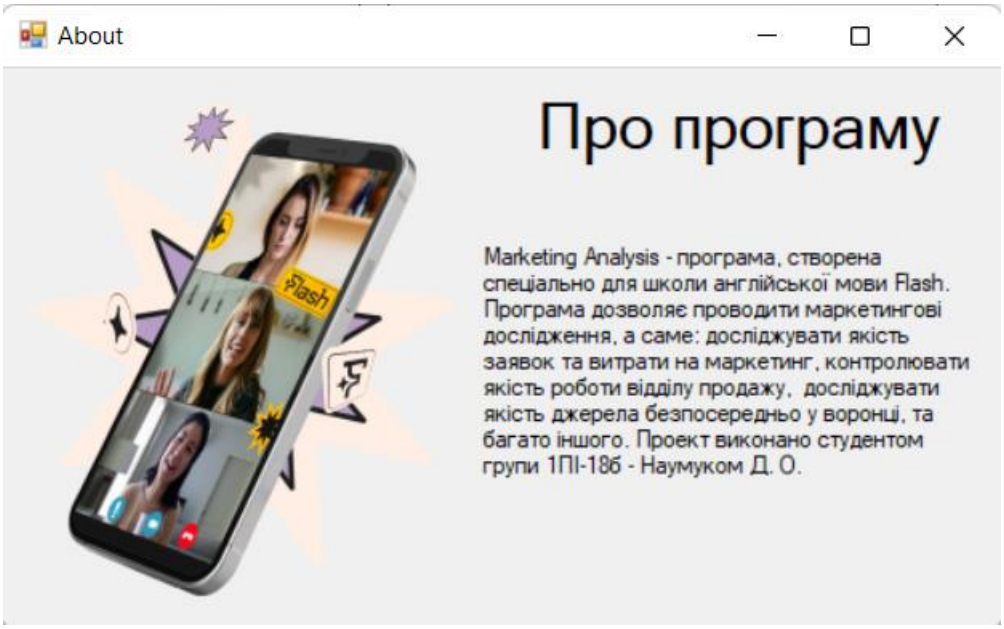


Рисунок 4.5 – Вікно «Про програму»

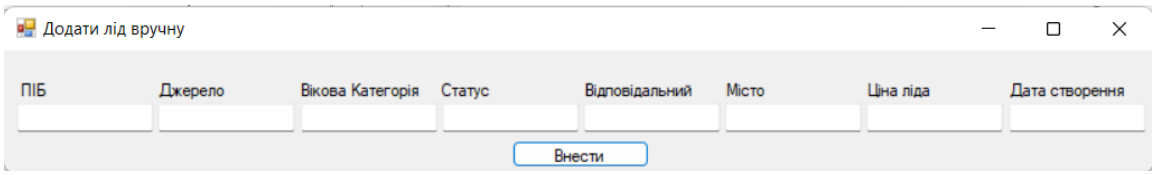


Рисунок 4.6 – Вікно «Додати лід вручну»

Створимо лід з ім’ям «ForTest», щоб перевірити роботу функції внесення лідів. Щоб внести лід потрібно заповнити усі поля та натиснути кнопку «Внести». Після успішного внесення з’являється повідомлення «Лід додано коректно» (Рисунок 4.7).

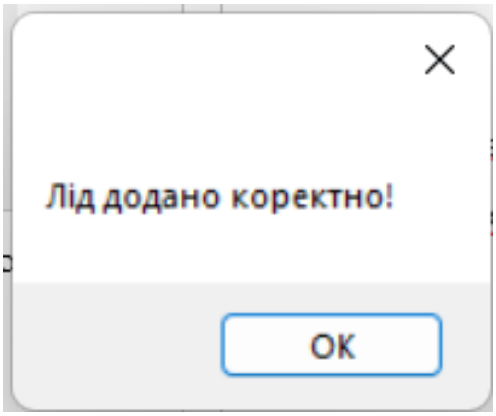


Рисунок 4.7 – Вікно «успішного додання ліда»

Наступною дією виконаємо видалення ліда. Для цього на головному вікні потрібно натиснути на кнопку «Видалення ліда». Після натискання на кнопку з’являється вікно видалення лідів (Рисунок 4.8).

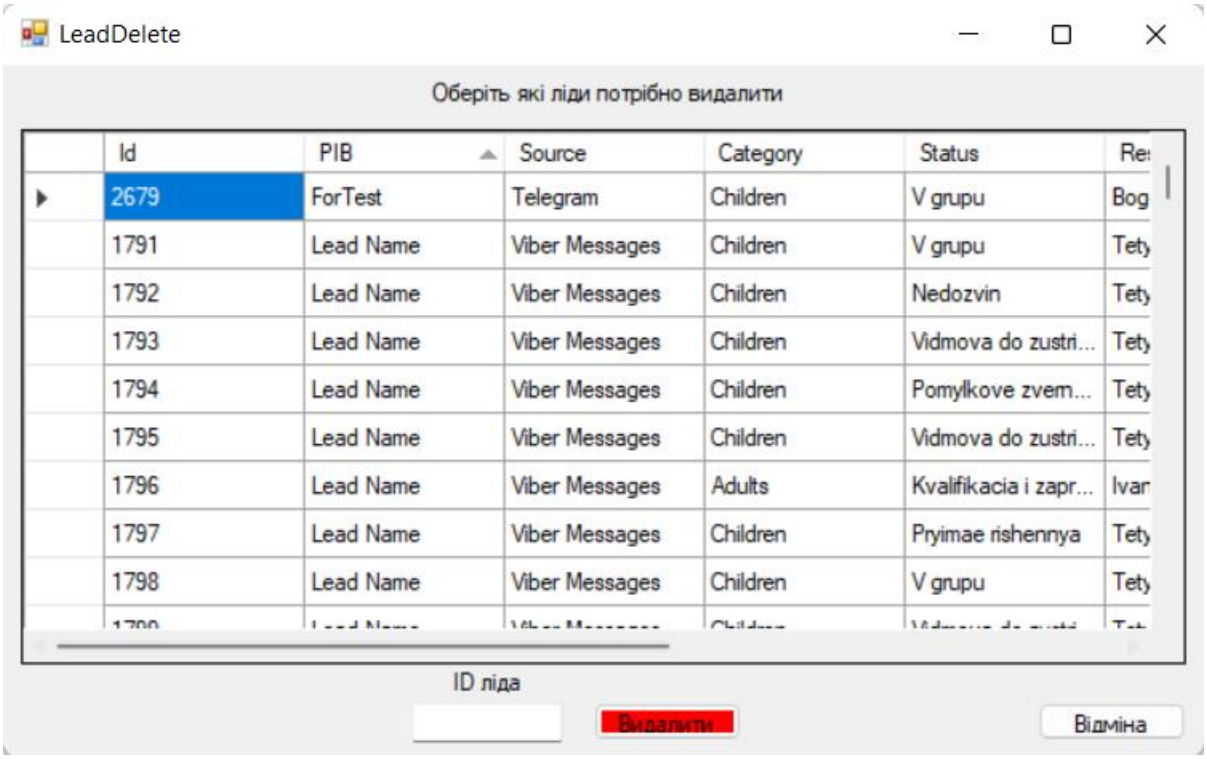


Рисунок 4.8 – Вікно «Видалення лідів»

У цьому вікні можна побачити свіжестворений лід “ForTest”. Щоб видалити лід потрібно ввести ID ліда у текстбокс, та натиснути кнопку «Видалити». Після успішного видалення з’являється повідомлення «Лід видалено коректно» (Рисунок 4.9).

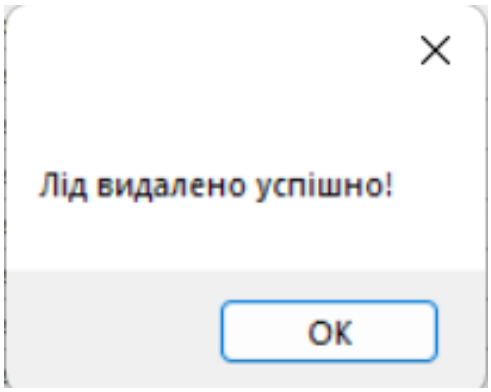


Рисунок 4.9 – Вікно «Лід видалено успішно»

Перейдемо до аналітики. Для того, щоб виконати аналіз роботи відділу продажу необхідно натиснути кнопку «Аналітика відділу продажу». Після натискання відкривається вікно, у якому розміщено список менеджерів з продажу та місце для відображення аналітик (Рисунок 4.10).

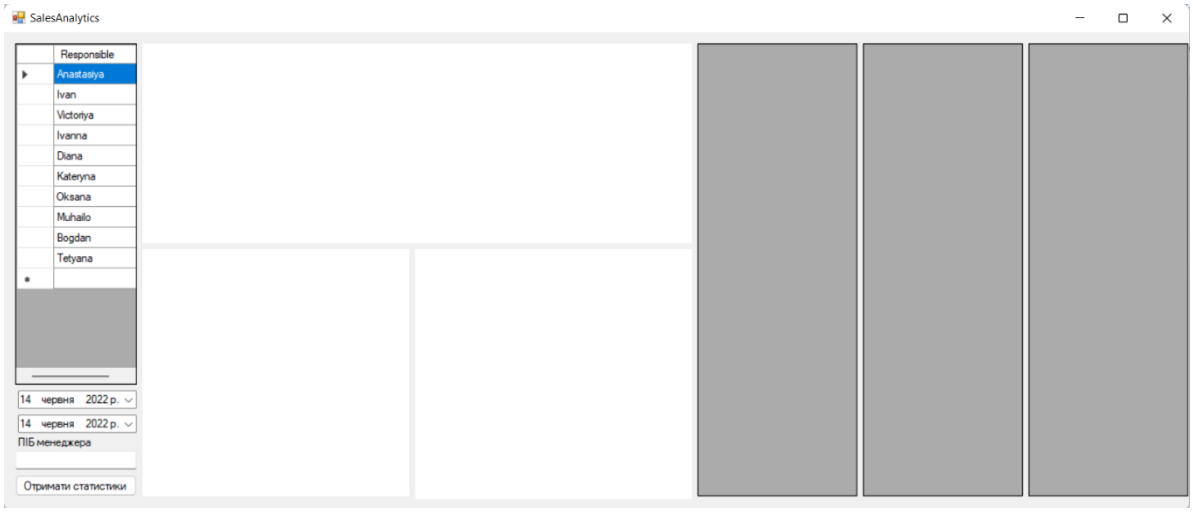


Рисунок 4.10 – Вікно «Аналітика відділу продажу»

Щоб отримати дані по менеджеру, потрібно вписати його ПІБ у текстбокс, обрати дати та натиснути кнопку «Отримати статистики». Після цього програма усю інформацію по роботі вибраного менеджера (Рисунок 4.11).

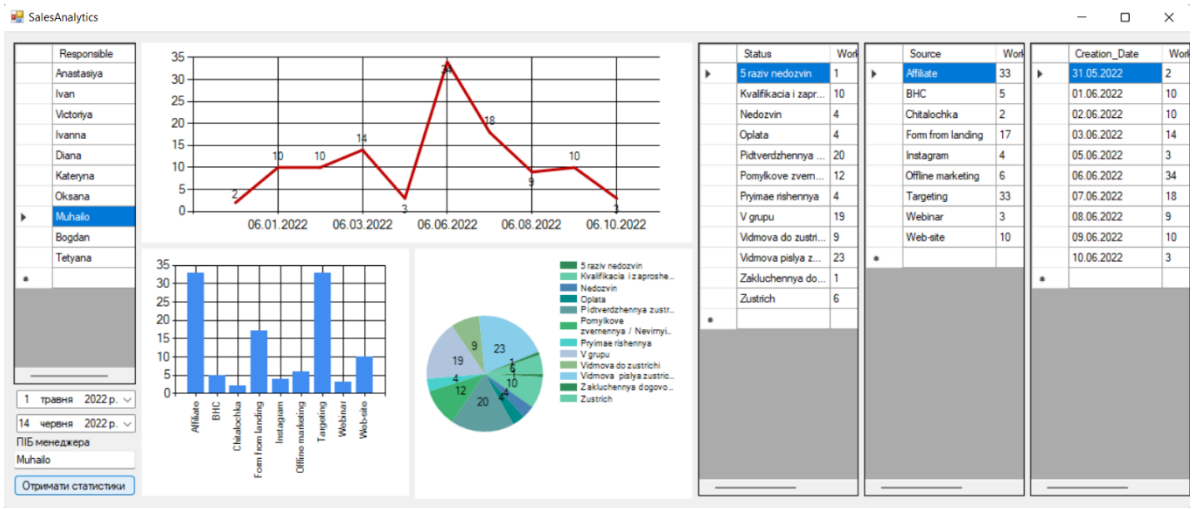


Рисунок 4.11 – Вікно «Аналітика відділу продажу» з оновленими даними

Для того, щоб виконати аналіз джерела по воронках необхідно натиснути кнопку «Статистика по воронках». Після натискання відкривається вікно, у якому розміщено список джерел з продажу та місце для відображення аналітик (Рисунок 4.12).

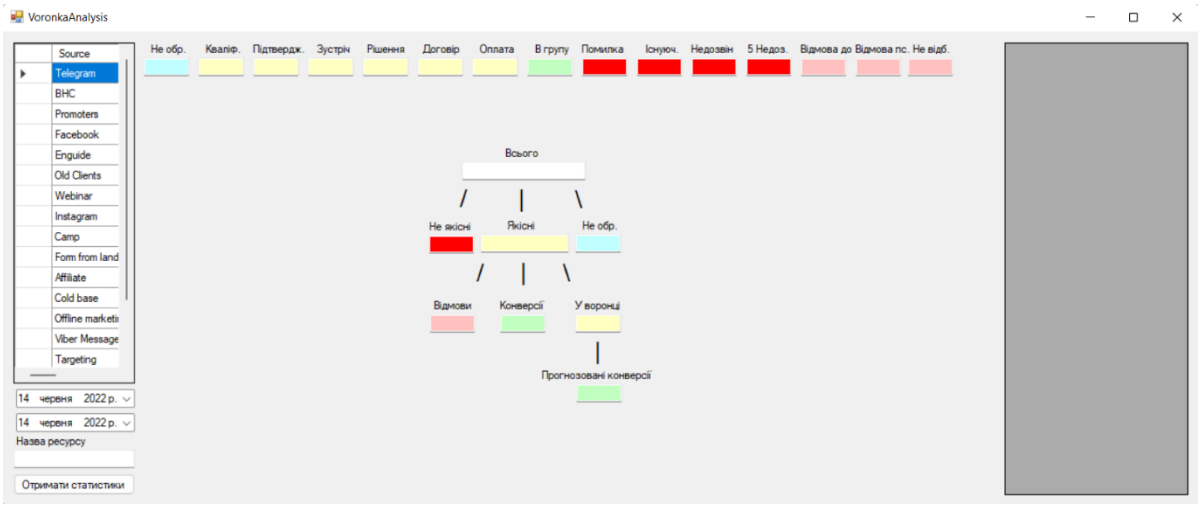


Рисунок 4.12 – Вікно «Статистика по воронках»

Щоб отримати дані по джерелу, потрібно вписати його назву у текстбокс, обрати дати та натиснути кнопку «Отримати статистики». Після цього програма усю інформацію по роботі вибране джерело (Рисунок 4.13).

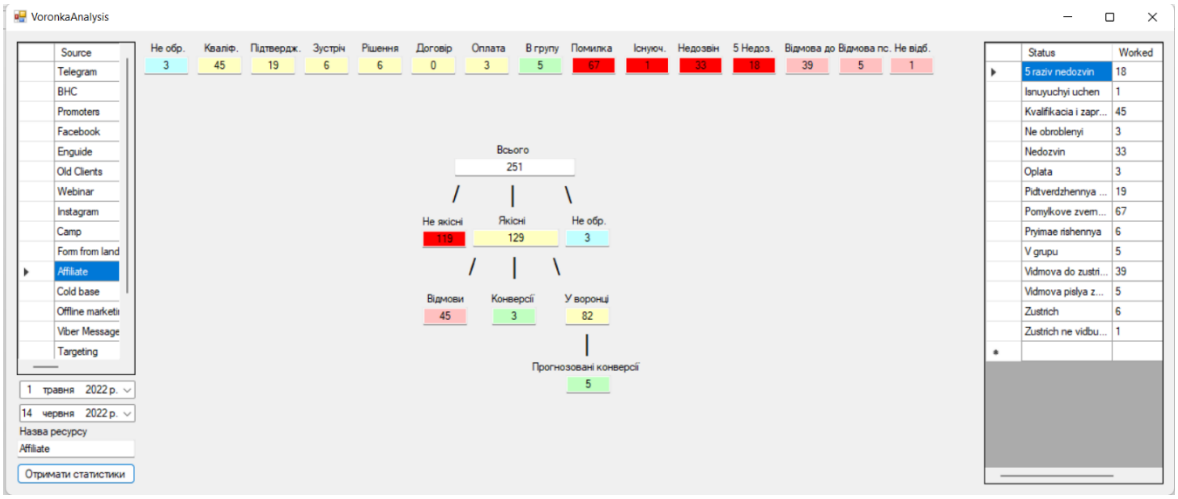


Рисунок 4.13 – Вікно «Статистика по воронках» з оновленими даними

Для того, щоб виконати аналіз джерела по конверсії, ROI та коофіцієнту ефективності рекламної кампанії необхідно натиснути кнопку «Аналіз джерела

трафіку». Після натискання відкривається вікно, у якому розміщено список джерел та місце для відображення аналітик (Рисунок 4.14).

Рисунок 4.14 – Вікно «Аналіз джерела трафіку»

Щоб отримати дані по джерелу, потрібно вписати його назву у текстбокс, обрати дати та натиснути кнопку «Отримати статистики». Після цього програма усю інформацію по роботі вибране джерело (Рисунок 4.15).

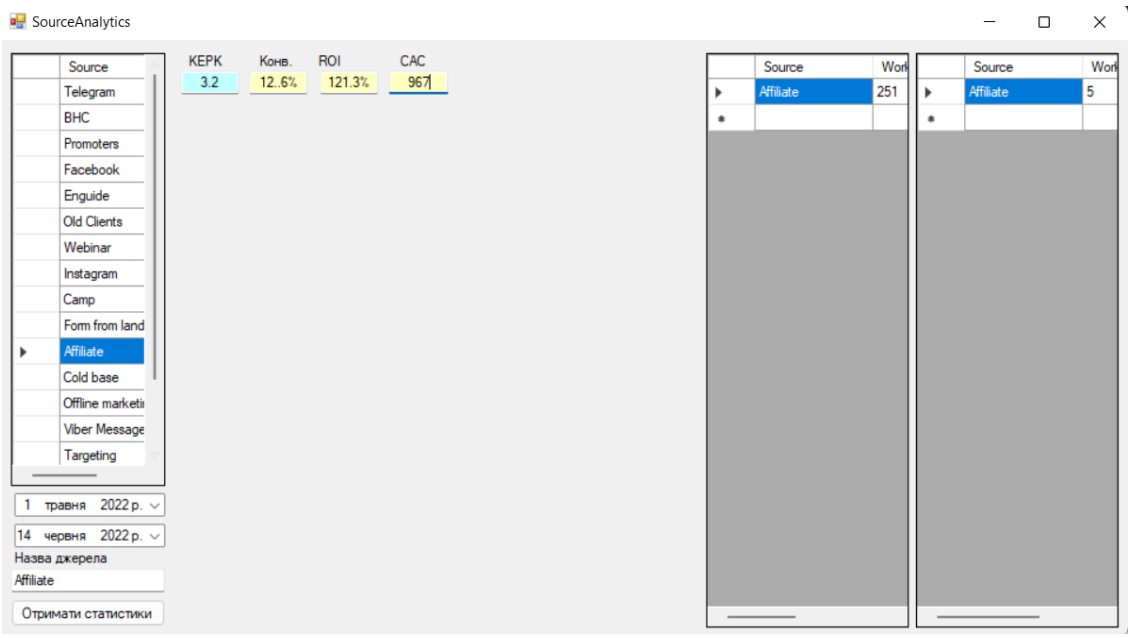


Рисунок 4.15 – Вікно «Аналіз джерела трафіку» з оновленими даними

Для завершення тесту необхідно натиснути кнопку «Вихід» у головному вікні.

4.3 Висновки

За результатами тестування програми підтверджено працездатність усіх заявлених функцій. Програма справно та швидко виконує усі поставлені задачі.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено програму для маркетингових досліджень. Для розробки було використано середовище програмної розробки Microsoft Visual Studio, мова програмування C# та декларативну мову програмування SQL.

У бакалаврській дипломній роботі проаналізовано стан проблем на сьогоднішній день. Розглянуто основні аналоги програмних продуктів та у порівнянні з власними програмним продуктом було виявлено їхні недоліки. На основі порівняння визначено основні завдання дипломної роботи для розробки програми для проведення аналізу маркетингових даних.

Під час аналізу технологій розробки було обґрунтовано вибір мови програмування C#. Також було розглянуто переваги використання декларативної мови програмування SQL.

Запропоновано метод реалізації аналізу маркетингових даних, який, на відміну від існуючих, пропонує користувачу коофіцієнт, за яким він зможе визначати ефективність рекламної кампанії.

Спроектовано користувацький інтерфейс програми. Виконано розробку програмного коду програми для маркетингових досліджень, яка допоможе користувачу ефективніше керувати рекламними кампаніями та відділом продажу.

Виконано аналіз методів тестування, обрано метод тестування «чорної скриньки». За обраним методом проведено тестування розробленої програми згідно поставлених задач, яке підтвердило його повну працездатність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Маркетинг [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Marketing

2. Маркетингова аналітика [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Analytics.

3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЦИФРОВОГО МАРКЕТИНГУ / Д. О. Наумук, С. Є. Тужанський Матеріали LI Науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2022), Секція – Програмне забезпечення. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15621

4. Г. Крамер “Математичні методи статистики”—М.: “Мир”, 1967р.—456с.

5. Алгоритм [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Algorithm

6. Маркетингові метрики [Електронний ресурс] – Режим доступу до ресурсу: https://ideadigital.agency/blog/osnovni-metriki-i-kpi-v-internet-marketingu/

7. Статистика [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Statistics

8. Google Analytics [Електронний ресурс] – Режим доступу до ресурсу: https://analytics.google.com/analytics/web/

9. Roistat [Електронний ресурс] – Режим доступу до ресурсу: https://roistat.com/

10. Програмування. Принципа та практика використання C++. / Бьярн Страуструп – Діалектика, 2018. – 320 ст.

11. Andrew Troelsen Pro C# 8 with .NET Core 3 : навч. посіб. / Andrew Troelsen, Phil Japikse – Apress, 2020. – 1881 ст.

12. Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://visualstudio.microsoft.com/.

13. Windows Forms [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/getting-started-with-windows-forms?view=netframeworkdesktop-4.8

14. SQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/sql/.

15. Designing for the Digital Age: How to Create Human-Centered Products and Services. / Kim Goodwin – Wiley, 2009. – 768 ст.

16. C# Guide [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/

17. Інтерфейс додатку. Як правильно створити? [Електронний ресурс] – Режим доступу до ресурсу: http://www.kievoit.ippo.kubg.edu.ua/kievoit/2016/83/index1.html

18. Мистецтво тестування програм / Гленфорд Майерс, Диалектика, 2012. – 270 ст.

19. Типи тестування програмного додатку. Чорна скринька [Електронний ресурс] – Режим доступу до ресурсу: https://uk.myservername.com/black-box-testing-an-depth-tutorial-with-examples

20. What is BLACK Box Testing? Techniques, Example & Types [Електронний ресурс] – Режим доступу до ресурсу: https://www.guru99.com/black-box-testing.html.

ДОДАТКИ

Додаток А

Технічне завдання

Розробка програмного забезпечення для маркетингових досліджень

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

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
31 березня 2022 р.

Технічне завдання
на бакалаврську дипломну роботу
«Розробка програми для маркетингових досліджень»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____к.т.н., доц. каф. ПЗ С. Є. Тужанський
"_31_"_____березня___2022 р.
Виконав:
_____студент гр. 1ПІ-18б Д.О. Наумук
"_31_"_____березня___2022 р.

Вінниця – 2022 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка програми для маркетингових досліджень».

Галузь застосування – програма для ПК.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол № 12 від « 07 » лютого 2022 р.

3. Мета та призначення розробки.

Метою бакалаврської дипломної роботи є групування, подача та аналіз маркетингових даних, що допоможе користувачеві ефективніше приймати рішення щодо рекламних кампаній.

Призначення роботи – програма для групування, візуалізації та аналізу маркетингових даних.

4. Вихідні дані для проведення НДР

1. Andrew Troelsen Pro C# 8 with .NET Core 3 : навч. посіб. / Andrew Troelsen, Phil Japikse – Apress, 2020. – 1881 ст.

2. Designing for the Digital Age: How to Create Human-Centered Products and Services. / Kim Goodwin – Wiley, 2009. – 768 ст.

3. The art of software Testing / Glandorf Myers , Professional, 2012. – 270 ст.

5. Технічні вимоги

Склад комплексу *технічних засобів*

Для розв’язку задачі буде використовуватися сучасний комп’ютер на операційній системі Windows 10.

Вимоги до складових *частин комплексу технічних засобів.*

Для запуску програми апаратне забезпечення повинне відповідати мінімальним вимогам:

Комп’ютер з процесором від Pentium;

- 1 Гб оперативної пам’яті;
- внутрішня графіка Intel;
- периферія: клавіатура, миша;
- ОС Windows 10;

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред’являється по закінченню робіт:

– пояснювальна записка до бакалаврської дипломної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

Додаток Б

Протокол перевірки на плагіат

ПРОТОКОЛ

ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програми для маркетингових досліджень

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доц. каф. ПЗ Тужанський С. Є.

Оригінальність	87,3%
Схожість	12,7%

Аналіз звіту подібності

■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Наумук Д. О.

Керівник роботи _____ Тужанський С. Є.

Додаток В

Лістинг програмного забезпечення для маркетингових досліджень

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Data.OleDb;
using LINQtoCSV;

namespace Marketing_analysis
{
   public partial class Form1 : Form
   {
      private static SqlConnection conn = null;
      public Form1()
      {
         InitializeComponent();
      }

      private void Form1_Load(object sender, EventArgs e)
      {
         dateTimePicker1.CustomFormat = "MM-dd-yyyy";
         dateTimePicker2.CustomFormat = "MM-dd-yyyy";

         conn = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);
         conn.Open();

      }
```

```
        private void dateTimePicker2_ValueChanged(object sender, EventArgs e)
        {

        }

        private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
        {

        }

        private void chart1_Click(object sender, EventArgs e)
        {

        }

        private void ChartUpdate()
        {
            chart1.Series["Sources"].Points.Clear();
            for (int i = 0; i < dataGridView1.RowCount-1; i++)
            {
                if (dataGridView1.Rows[i].Cells[1].Value == null)
                {
                    Console.WriteLine("NULL");

                }
                Console.WriteLine(dataGridView1.Rows[i].Cells[0].Value);
                Console.WriteLine(dataGridView1.Rows[i].Cells[1].Value);


                chart1.Series["Sources"].Points.AddXY(dataGridView1.Rows[i].Cells[0].Value,
dataGridView1.Rows[i].Cells[1].Value);
            }


        }

        private void button1_Click(object sender, EventArgs e)
        {
            DateTime date1 = dateTimePicker2.Value.Date;
            DateTime date2 = dateTimePicker1.Value.Date;
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS LeadsCount, " +
                $"AVG(Lead_Price) AS AVGLeadPrice, SUM (Lead_Price) AS LeadPriceSum FROM Leads " +
```

```
        $"WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND " +
        $"'{date2.ToString("MM.dd.yyyy")}' GROUP BY Source;", conn);

    DataSet dataSet = new DataSet();
    adapter.Fill(dataSet);

    dataGridView1.DataSource = dataSet.Tables[0];

    ChartUpdate();
}

private static void ReadCSVFile()
{
    var csvFileDescription = new CsvFileDescription()
    {
        FirstLineHasColumnNames = true,
        IgnoreUnknownColumns = true,
        SeparatorChar = ';',
        UseFieldIndexForReadingData = false,
        TextEncoding = Encoding.UTF8

    };

    var csvContext = new CsvContext();
    var leads = csvContext.Read<Lead>("Leads.csv", csvFileDescription);

    foreach(var lead in leads)
    {
        SqlCommand command = new SqlCommand(
        $"INSERT INTO [Leads] (PIB, Source, Category, Status, Responsible, " +
        $"City, Lead_Price, Creation_Date) VALUES" +
        $" (N'{lead.PIB}', N'{lead.Source}', N'{lead.Category}', N'{lead.Status}', " +
        $"N'{lead.Responsible}', N'{lead.City}', N'{lead.Lead_Price}', N'{lead.Creation_Date}');", conn);
        command.ExecuteNonQuery();
    }
}

private void button2_Click(object sender, EventArgs e)
{
    ReadCSVFile();
}

private void Export()
```

```
{
    conn.Close();
    StreamWriter file;
    file = new StreamWriter("Leads.csv", false);
    using (conn)
    {
        try
        {
            var query = "SELECT * FROM Leads";
            SqlCommand command =
                new SqlCommand(query, conn);
            conn.Open();
            SqlDataReader SQLreader =
                command.ExecuteReader();

            file.WriteLine(

@""""PIB"";""Source"";""Category"";""Status"";""Responsible"";""City"";""Lead_Price"";""Creation_Date""");
            if (SQLreader.HasRows)
            {
                while (SQLreader.Read())
                {
                    file.WriteLine(@""""" +
                        SQLreader.GetValue(1).ToString() +
                        @""";""" +
                        SQLreader.GetValue(2).ToString() +
                        @""";""" +
                        SQLreader.GetValue(3).ToString() +
                        @""";""" +
                        SQLreader.GetValue(4).ToString() +
                        @""";""" +
                        SQLreader.GetValue(5).ToString() +
                        @""";""" +
                        SQLreader.GetValue(6).ToString() +
                        @""";""" +
                         SQLreader.GetValue(7).ToString() +
                        @""";""" +
                        SQLreader.GetValue(8).ToString() +
                        @""""");
                }
            }
            else
                file.WriteLine(
```

```
                    "No one row is in \"Leads\" table");
                file.WriteLine("End of file");
                file.Close();
                SQLreader.Close();
                conn.Dispose();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                if (conn.State == ConnectionState.Open)
                    conn.Close();
                conn.Dispose();
            }
        }
    }
    private void button3_Click(object sender, EventArgs e)
    {
        Export();
    }

    class ExcelReader
    {
        public DataSet GetCVSFile(string pathName, string fileName)
        {
            using (OleDbConnection ExcelConn =
            new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;
            Data Source=" + pathName + ";Extended Properties=Text;"))
            {
                OleDbCommand ExcelCommand = new OleDbCommand(@"SELECT * FROM " + fileName,
ExcelConn);
                OleDbDataAdapter ExcelAdapter =  new OleDbDataAdapter(ExcelCommand);

                ExcelConn.Open();

                DataSet ExcelDataSet = new DataSet("MyDataSet");
                ExcelAdapter.Fill(ExcelDataSet);

                return ExcelDataSet;
            }
        }
```

```
}

private void button4_Click(object sender, EventArgs e)
{
   Lead_add lead_Add = new Lead_add();
   lead_Add.Show();
}

private void button12_Click(object sender, EventArgs e)
{
   dateTimePicker1.Value = DateTime.Now;
   dateTimePicker2.Value = DateTime.Now;
}

private void button11_Click(object sender, EventArgs e)
{
   dateTimePicker2.Value = DateTime.Now.AddDays(-7);
   dateTimePicker1.Value = DateTime.Now;
}

private void button10_Click(object sender, EventArgs e)
{
   var now = DateTime.Now;
   dateTimePicker2.Value = new DateTime(now.Year, now.Month, 1);
   dateTimePicker1.Value = new DateTime(now.Year, now.Month + 1, 1);
}

private void button5_Click(object sender, EventArgs e)
{
   LeadDelete leadDelete = new LeadDelete();
   leadDelete.Show();
}

private void button9_Click(object sender, EventArgs e)
{
   SalesAnalytics salesAnalytics = new SalesAnalytics();
   salesAnalytics.Show();
}

private void button4_Click_1(object sender, EventArgs e)
{
   chart1.Series["Sources"].Points.Clear();
   DateTime date1 = dateTimePicker2.Value.Date;
```

```
            DateTime date2 = dateTimePicker1.Value.Date;
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS LeadsCount, " +
              $"AVG(Lead_Price) AS AVGLeadPrice, SUM (Lead_Price) AS LeadPriceSum FROM Leads " +
              $"WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND " +
              $"'{date2.ToString("MM.dd.yyyy")}' GROUP BY Source;", conn);

            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);

            dataGridView1.DataSource = dataSet.Tables[0];

            ChartUpdate();
        }

        private void button7_Click(object sender, EventArgs e)
        {

            chart1.Series["Sources"].Points.Clear();
            DateTime date1 = dateTimePicker2.Value.Date;
            DateTime date2 = dateTimePicker1.Value.Date;
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS LeadsCount,
AVG(Lead_Price) AS AVGLeadPrice, SUM (Lead_Price) AS LeadPriceSum FROM Leads " +
              $"WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND
'{date2.ToString("MM.dd.yyyy")}' GROUP BY Source;", conn);
            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);

            dataGridView1.DataSource = dataSet.Tables[0];
            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
              if (dataGridView1.Rows[i].Cells[1].Value == null)
              {
                Console.WriteLine("NULL");

              }
              Console.WriteLine(dataGridView1.Rows[i].Cells[0].Value);
              Console.WriteLine(dataGridView1.Rows[i].Cells[1].Value);


              chart1.Series["Sources"].Points.AddXY(dataGridView1.Rows[i].Cells[0].Value,
dataGridView1.Rows[i].Cells[3].Value);
            }
        }
```

```
        private void button13_Click(object sender, EventArgs e)
        {
            chart1.Series["Sources"].Points.Clear();
            DateTime date1 = dateTimePicker2.Value.Date;
            DateTime date2 = dateTimePicker1.Value.Date;
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS ConversionsCount,
AVG(Lead_Price) AS AVGLeadPrice FROM Leads " +
                $"WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND
'{date2.ToString("MM.dd.yyyy")}' AND Status='V grupu' GROUP BY Source;", conn);

            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);

            dataGridView1.DataSource = dataSet.Tables[0];

            ChartUpdate();
        }

        private void button8_Click(object sender, EventArgs e)
        {
            VoronkaAnalysis voronka = new VoronkaAnalysis();
            voronka.Show();
        }

        private void проПрограмуToolStripMenuItem_Click(object sender, EventArgs e)
        {
            About about = new About();
            about.Show();
        }

        private void імпортуватиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            ReadCSVFile();
        }

        private void експортуватиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            conn.Close();
            StreamWriter file;
            file = new StreamWriter("Leads.csv", false);
            using (conn)
            {
```

```
                try
                {
                    var query = "SELECT * FROM Leads";
                    SqlCommand command =
                        new SqlCommand(query, conn);
                    conn.Open();
                    SqlDataReader SQLreader =
                        command.ExecuteReader();

                    file.WriteLine(

@""""PIB"";""Source"";""Category"";""Status"";""Responsible"";""City"";""Lead_Price"";""Creation_Date""");
                    if (SQLreader.HasRows)
                    {
                        while (SQLreader.Read())
                        {
                            file.WriteLine(@""""" +
                                SQLreader.GetValue(1).ToString() +
                                @""";""" +
                                SQLreader.GetValue(2).ToString() +
                                @""";""" +
                                SQLreader.GetValue(3).ToString() +
                                @""";""" +
                                SQLreader.GetValue(4).ToString() +
                                @""";""" +
                                SQLreader.GetValue(5).ToString() +
                                @""";""" +
                                SQLreader.GetValue(6).ToString() +
                                @""";""" +
                                 SQLreader.GetValue(7).ToString() +
                                @""";""" +
                                SQLreader.GetValue(8).ToString() +
                                @""""");
                        }
                    }
                    else
                        file.WriteLine(
                            "No one row is in \"Leads\" table");
                    file.WriteLine("End of file");
                    file.Close();
                    SQLreader.Close();
                    conn.Dispose();
                }
```

```
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            if (conn.State == ConnectionState.Open)
                conn.Close();
            conn.Dispose();
        }
    }
}

private void внестиЛідВручнуToolStripMenuItem_Click(object sender, EventArgs e)
{
    Lead_add lead_Add = new Lead_add();
    lead_Add.Show();
}

private void аналітикаВідділуПродажівToolStripMenuItem_Click(object sender, EventArgs e)
{
    SalesAnalytics salesAnalytics = new SalesAnalytics();
    salesAnalytics.Show();
}

private void статистикаПоВоронкахToolStripMenuItem_Click(object sender, EventArgs e)
{
    VoronkaAnalysis voronka = new VoronkaAnalysis();
    voronka.Show();
}

private void button6_Click(object sender, EventArgs e)
{
    SourceAnalytics sourceAnalytics = new SourceAnalytics();
    sourceAnalytics.Show();
}

private void статистикаПоЛідамToolStripMenuItem_Click(object sender, EventArgs e)
{
    chart1.Series["Sources"].Points.Clear();
    DateTime date1 = dateTimePicker2.Value.Date;
    DateTime date2 = dateTimePicker1.Value.Date;
```

```
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS LeadsCount, AVG(Lead_Price) AS AVGLeadPrice, SUM (Lead_Price) AS LeadPriceSum FROM Leads " +
                $"WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}' GROUP BY Source;", conn);

            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);

            dataGridView1.DataSource = dataSet.Tables[0];

            ChartUpdate();
        }

        private void статистикаПоВитратамToolStripMenuItem_Click(object sender, EventArgs e)
        {

            chart1.Series["Sources"].Points.Clear();
            DateTime date1 = dateTimePicker2.Value.Date;
            DateTime date2 = dateTimePicker1.Value.Date;
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS LeadsCount, AVG(Lead_Price) AS AVGLeadPrice, SUM (Lead_Price) AS LeadPriceSum FROM Leads " +
                $"WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}' GROUP BY Source;", conn);
            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);

            dataGridView1.DataSource = dataSet.Tables[0];
            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                if (dataGridView1.Rows[i].Cells[1].Value == null)
                {
                    Console.WriteLine("NULL");

                }
                Console.WriteLine(dataGridView1.Rows[i].Cells[0].Value);
                Console.WriteLine(dataGridView1.Rows[i].Cells[1].Value);


                chart1.Series["Sources"].Points.AddXY(dataGridView1.Rows[i].Cells[0].Value, dataGridView1.Rows[i].Cells[3].Value);
            }
        }
```

```
        private void статистикаПоПрожажамToolStripMenuItem_Click(object sender, EventArgs e)
        {
            chart1.Series["Sources"].Points.Clear();
            DateTime date1 = dateTimePicker2.Value.Date;
            DateTime date2 = dateTimePicker1.Value.Date;
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS ConversionsCount,
AVG(Lead_Price) AS AVGLeadPrice FROM Leads " +
                $"WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND
'{date2.ToString("MM.dd.yyyy")}' AND Status='V grupu' GROUP BY Source;", conn);

            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);

            dataGridView1.DataSource = dataSet.Tables[0];

            ChartUpdate();
        }
    }
}
```

SourseAnalysis.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Marketing_analysis
{
    public partial class SourceAnalytics : Form
    {
        private SqlConnection conn = null;

        public SourceAnalytics()
        {
            InitializeComponent();
        }
```

```
        private void SourceAnalytics_Load(object sender, EventArgs e)
        {
            dateTimePicker1.CustomFormat = "MM-dd-yyyy";
            dateTimePicker2.CustomFormat = "MM-dd-yyyy";
            conn = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);
            conn.Open();
            TableUpdate();
        }

        private void TableUpdate()
        {
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source FROM Leads GROUP BY Source",
conn);

            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);

            dataGridView1.DataSource = dataSet.Tables[0];
        }

        private void button1_Click(object sender, EventArgs e)
        {
            var id = textBox1.Text.ToString();
            LeadsUpdate(id);
            ConversionUpdate(id);
            //KERKExecute();
        }

        private void KERKExecute()
        {
            DateTime date1 = dateTimePicker2.Value.Date;
            DateTime date2 = dateTimePicker1.Value.Date;

            //Executing first value
            var cpl = (Convert.ToInt32(dataGridView2.Rows[0].Cells[2].Value));
            var allLeads = Convert.ToInt32(dataGridView2.Rows[0].Cells[1].Value);
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT '{cpl}'/AVG(Lead_Price) FROM Leads
WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}'
GROUP BY Source", conn);
            DataSet value1 = new DataSet();
            adapter.Fill(value1);
```

```
                var cac = (allLeads * cpl /Convert.ToInt32(dataGridView3.Rows[0].Cells[1].Value));
                adapter = new SqlDataAdapter($"SELECT '{cac}'/(SELECT SUM(Lead_Price) FROM Leads WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}')/(SELECT COUNT(*) FROM Leads WHERE Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}') FROM Leads ", conn);
                DataSet value2 = new DataSet();
                adapter.Fill(value2);
                Console.WriteLine(value2.Tables[0].Rows[0].ToString());
                Console.WriteLine(value2.Tables[0].Rows[0].ToString());


                var kerk = (Convert.ToInt32(dataGridView2.Rows[0].Cells[2].Value));
            }

            private void LeadsUpdate(string id)
            {
                DateTime date1 = dateTimePicker2.Value.Date;
                DateTime date2 = dateTimePicker1.Value.Date;
                SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS Worked, AVG(Lead_Price) FROM Leads WHERE Source='{id}' AND Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}' GROUP BY Source", conn);
                DataSet dataSet = new DataSet();
                adapter.Fill(dataSet);
                dataGridView2.DataSource = dataSet.Tables[0];
            }

            private void ConversionUpdate(string id)
            {
                DateTime date1 = dateTimePicker2.Value.Date;
                DateTime date2 = dateTimePicker1.Value.Date;
                SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS Worked FROM Leads WHERE Source='{id}' AND Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}' AND Status='V grupu'  GROUP BY Source", conn);
                DataSet dataSet = new DataSet();
                adapter.Fill(dataSet);
                dataGridView3.DataSource = dataSet.Tables[0];
            }
        }
    }
```

LeadAdd.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

namespace Marketing_analysis
{
   public partial class Lead_add : Form
   {
      private SqlConnection conn = null;
      public Lead_add()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         SqlCommand command = new SqlCommand(
            $"INSERT INTO [Leads] (PIB, Source, Category, Status, Responsible, City, Lead_Price, Creation_Date) VALUES (N'{textBox1.Text}', N'{textBox2.Text}', N'{textBox3.Text}', N'{textBox4.Text}', N'{textBox5.Text}', N'{textBox6.Text}', N'{textBox7.Text}', N'{textBox8.Text}')", conn);

         if (command.ExecuteNonQuery() == 1 )
         {
            MessageBox.Show("Лід додано коректно!");
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
            textBox4.Text = "";
            textBox5.Text = "";
            textBox6.Text = "";
            textBox7.Text = "";
            textBox8.Text = "";
         }
         else
```

```
            {
                MessageBox.Show("Данні введено некоректно. Перевірте правильність написання!");
            }
            conn.Close();
        }

        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void Lead_add_Load(object sender, EventArgs e)
        {


            conn = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);
            conn.Open();

        }
    }
}
```

LeadDelete.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Marketing_analysis
{
    public partial class LeadDelete : Form
    {
        private SqlConnection conn;

        public LeadDelete()
```

```
{
    InitializeComponent();
}

private void label1_Click(object sender, EventArgs e)
{

}

private void LeadDelete_Load(object sender, EventArgs e)
{
    conn = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);
    conn.Open();
    TableUpdate();
}

private void TableUpdate()
{
    SqlDataAdapter adapter = new SqlDataAdapter($"SELECT * FROM Leads", conn);

    DataSet dataSet = new DataSet();
    adapter.Fill(dataSet);

    dataGridView1.DataSource = dataSet.Tables[0];
}

private void label2_Click(object sender, EventArgs e)
{

}

private void button2_Click(object sender, EventArgs e)
{
    var id = textBox1.Text.ToString();
    SqlCommand command = new SqlCommand(
        $"DELETE FROM Leads WHERE Id={id}", conn);

    if (command.ExecuteNonQuery() > 0)
    {
        MessageBox.Show("Лід видалено успішно!");
        textBox1.Text = "";
    }
    else
```

```
            {
                MessageBox.Show("Данні введено некоректно. Перевірте правильність написання!");
            }
            TableUpdate();
        }
    }
}
```

About.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Marketing_analysis
{
    public partial class About : Form
    {
        public About()
        {
            InitializeComponent();
        }
    }
}
```

Sales Analysys.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace Marketing_analysis
{
    public partial class SalesAnalytics : Form
    {
        private SqlConnection conn = null;

        public SalesAnalytics()
        {
            InitializeComponent();
        }

        private void SalesAnalytics_Load(object sender, EventArgs e)
        {
            dateTimePicker1.CustomFormat = "MM-dd-yyyy";
            dateTimePicker2.CustomFormat = "MM-dd-yyyy";
            conn = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);
            conn.Open();
            TableUpdate();
        }

        private void TableUpdate()
        {
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Responsible FROM Leads GROUP BY
Responsible", conn);

            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);

            dataGridView1.DataSource = dataSet.Tables[0];
        }

        private void button1_Click(object sender, EventArgs e)
        {
            var id = textBox1.Text.ToString();
            ChartVoronkaUpdate(id);
            ChartSoursesUpdate(id);
            ChartTimeUpdate(id);
        }

        private void ChartVoronkaUpdate(string id)
        {
            chart3.Series[0].Points.Clear();
            DateTime date1 = dateTimePicker2.Value.Date;
```

```
            DateTime date2 = dateTimePicker1.Value.Date;
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Status, COUNT(*) AS Worked FROM Leads WHERE Responsible='{id}' AND Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}' GROUP BY Status", conn);
            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);
            dataGridView2.DataSource = dataSet.Tables[0];
            for (int i = 0; i < dataGridView2.RowCount - 1; i++)
            {
                if (dataGridView2.Rows[i].Cells[0].Value == null)
                {
                    Console.WriteLine("NULL");

                }
                Console.WriteLine(dataGridView2.Rows[i].Cells[0].Value);


                chart3.Series[0].Points.AddXY(dataGridView2.Rows[i].Cells[0].Value, dataGridView2.Rows[i].Cells[1].Value);
            }
        }

        private void ChartSoursesUpdate(string id)
        {
            chart2.Series[0].Points.Clear();
            DateTime date1 = dateTimePicker2.Value.Date;
            DateTime date2 = dateTimePicker1.Value.Date;
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) AS Worked FROM Leads WHERE Responsible='{id}' AND Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}' GROUP BY Source", conn);
            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);
            dataGridView3.DataSource = dataSet.Tables[0];
            for (int i = 0; i < dataGridView3.RowCount - 1; i++)
            {
                if (dataGridView3.Rows[i].Cells[0].Value == null || dataGridView3.Rows[i].Cells[1].Value == null)
                {

                }
                else
                {
                    chart2.Series[0].Points.AddXY(dataGridView3.Rows[i].Cells[0].Value, dataGridView3.Rows[i].Cells[1].Value);
```

```
            }
                }
        }

        private void ChartTimeUpdate(string id)
        {
            chart1.Series[0].Points.Clear();
            DateTime date1 = dateTimePicker2.Value.Date;
            DateTime date2 = dateTimePicker1.Value.Date;
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Creation_Date, COUNT(*) AS Worked FROM Leads WHERE Responsible='{id}' AND Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}' GROUP BY Creation_Date", conn);
            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);
            dataGridView4.DataSource = dataSet.Tables[0];
            for (int i = 0; i < dataGridView4.RowCount - 1; i++)
            {
                if (dataGridView4.Rows[i].Cells[0].Value == null || dataGridView4.Rows[i].Cells[1].Value == null)
                {

                }
                else
                {
                    DateTime date = Convert.ToDateTime(dataGridView4.Rows[i].Cells[0].Value);
                    string s1 = date.ToString("MM/dd/yyyy");
                    chart1.Series[0].Points.AddXY(s1, dataGridView4.Rows[i].Cells[1].Value);
                }
            }
        }
    }
}
```

VoronkaAnalysis.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Marketing_analysis
{
    public partial class VoronkaAnalysis : Form
    {

        private SqlConnection conn = null;
        public VoronkaAnalysis()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            var id = textBox1.Text.ToString();
            VoronkaFill(id);
        }

        private void VoronkaFill(string id)
        {
            DateTime date1 = dateTimePicker2.Value.Date;
            DateTime date2 = dateTimePicker1.Value.Date;
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Status, COUNT(*) AS Worked FROM Leads WHERE Source='{id}' AND Creation_Date BETWEEN '{date1.ToString("MM.dd.yyyy")}' AND '{date2.ToString("MM.dd.yyyy")}' GROUP BY Status", conn);
            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);
            dataGridView2.DataSource = dataSet.Tables[0];
            for (int i = 0; i < dataGridView2.Rows.Count; i++)
            {
                if (dataGridView2.Rows[i].Cells[1].Value != null)
                {
                    if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Ne obroblenyi")
                        textBox2.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                    if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Kvalifikacia i zaproshennya")
                        textBox3.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                    if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Pidtverdzhennya zustrichi")
                        textBox4.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();
```

```
                if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Zustrich")
                    textBox5.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Pryimae rishennya")
                    textBox6.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Zakluchennya dogovoru")
                    textBox7.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Oplata")
                    textBox8.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "V grupu")
                    textBox9.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Pomylkove zvernennya / Nevirnyi
nomer")
                    textBox10.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Isnuyuchyi uchen")
                    textBox11.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                if (dataGridView2.Rows[i].Cells[0].Value.ToString().Contains("raziv nedozvin"))
                    textBox13.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Nedozvin")
                    textBox12.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Vidmova do zustrichi")
                    textBox14.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Vidmova pislya zustrichi")
                    textBox15.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();

                if (dataGridView2.Rows[i].Cells[0].Value.ToString() == "Zustrich ne vidbulasya")
                    textBox16.Text = dataGridView2.Rows[i].Cells[1].Value.ToString();
            }
            else
                Console.WriteLine(i + " is Null");
        }

        if (textBox2.Text == "")
            textBox2.Text = "0";
```

```
            if (textBox3.Text == "")
                textBox3.Text = "0";
            if (textBox4.Text == "")
                textBox4.Text = "0";
            if (textBox5.Text == "")
                textBox5.Text = "0";
            if (textBox6.Text == "")
                textBox6.Text = "0";
            if (textBox7.Text == "")
                textBox7.Text = "0";
            if (textBox8.Text == "")
                textBox8.Text = "0";
            if (textBox9.Text == "")
                textBox9.Text = "0";
            if (textBox10.Text == "")
                textBox10.Text = "0";
            if (textBox11.Text == "")
                textBox11.Text = "0";
            if (textBox12.Text == "")
                textBox12.Text = "0";
            if (textBox13.Text == "")
                textBox13.Text = "0";
            if (textBox14.Text == "")
                textBox14.Text = "0";
            if (textBox15.Text == "")
                textBox15.Text = "0";
            if (textBox16.Text == "")
                textBox16.Text = "0";

            all.Text = (Convert.ToInt32(textBox2.Text) + Convert.ToInt32(textBox3.Text) +
Convert.ToInt32(textBox4.Text) + Convert.ToInt32(textBox5.Text) + Convert.ToInt32(textBox6.Text) +
Convert.ToInt32(textBox7.Text) + Convert.ToInt32(textBox8.Text) + Convert.ToInt32(textBox9.Text) +
Convert.ToInt32(textBox10.Text) + Convert.ToInt32(textBox11.Text) + Convert.ToInt32(textBox12.Text) +
Convert.ToInt32(textBox13.Text) + Convert.ToInt32(textBox14.Text) + Convert.ToInt32(textBox15.Text) +
Convert.ToInt32(textBox16.Text)).ToString();

            textBox18.Text = (Convert.ToInt32(textBox3.Text) + Convert.ToInt32(textBox4.Text) +
Convert.ToInt32(textBox5.Text) + Convert.ToInt32(textBox6.Text) + Convert.ToInt32(textBox7.Text) +
Convert.ToInt32(textBox8.Text) + Convert.ToInt32(textBox9.Text) + Convert.ToInt32(textBox14.Text) +
Convert.ToInt32(textBox15.Text) + Convert.ToInt32(textBox16.Text)).ToString();

            textBox17.Text = (Convert.ToInt32(textBox10.Text) + Convert.ToInt32(textBox11.Text) +
Convert.ToInt32(textBox12.Text) + Convert.ToInt32(textBox13.Text)).ToString();
```

```
            textBox19.Text = (Convert.ToInt32(textBox2.Text)).ToString();

            textBox21.Text = (Convert.ToInt32(textBox14.Text) + Convert.ToInt32(textBox15.Text) + Convert.ToInt32(textBox16.Text)).ToString();

            textBox20.Text = (Convert.ToInt32(textBox8.Text)).ToString();

            textBox22.Text = (Convert.ToInt32(textBox2.Text) + Convert.ToInt32(textBox3.Text) + Convert.ToInt32(textBox4.Text) + Convert.ToInt32(textBox5.Text) + Convert.ToInt32(textBox6.Text) + Convert.ToInt32(textBox7.Text) + Convert.ToInt32(textBox8.Text)).ToString();

            textBox23.Text = (Convert.ToInt32(textBox22.Text) * Convert.ToInt32(textBox20.Text) / (Convert.ToInt32(textBox20.Text) + Convert.ToInt32(textBox21.Text))).ToString();



        }

        private void VoronkaAnalysis_Load(object sender, EventArgs e)
        {
            dateTimePicker1.CustomFormat = "MM-dd-yyyy";
            dateTimePicker2.CustomFormat = "MM-dd-yyyy";
            conn = new SqlConnection(ConfigurationManager.ConnectionStrings["DB"].ConnectionString);
            conn.Open();
            TableUpdate();
        }

        private void TableUpdate()
        {
            SqlDataAdapter adapter = new SqlDataAdapter($"SELECT Source, COUNT(*) FROM Leads GROUP BY Source", conn);

            DataSet dataSet = new DataSet();
            adapter.Fill(dataSet);

            dataGridView1.DataSource = dataSet.Tables[0];
        }

        private void label2_Click(object sender, EventArgs e)
        {

        }
```

```
        private void label17_Click(object sender, EventArgs e)
        {

        }

        private void label20_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Додаток Г

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МАРКЕТИНГОВИХ ДОСЛІДЖЕНЬ

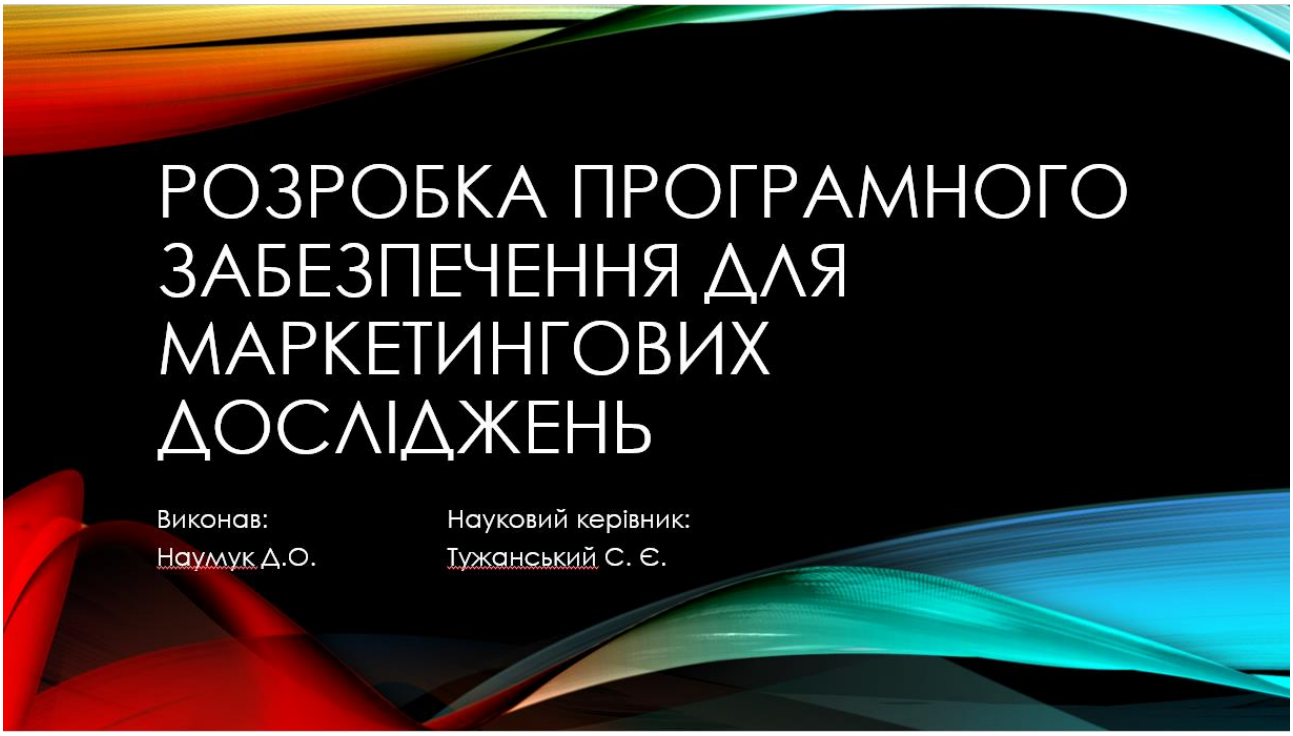


Рисунок Г.1 – Назва роботи

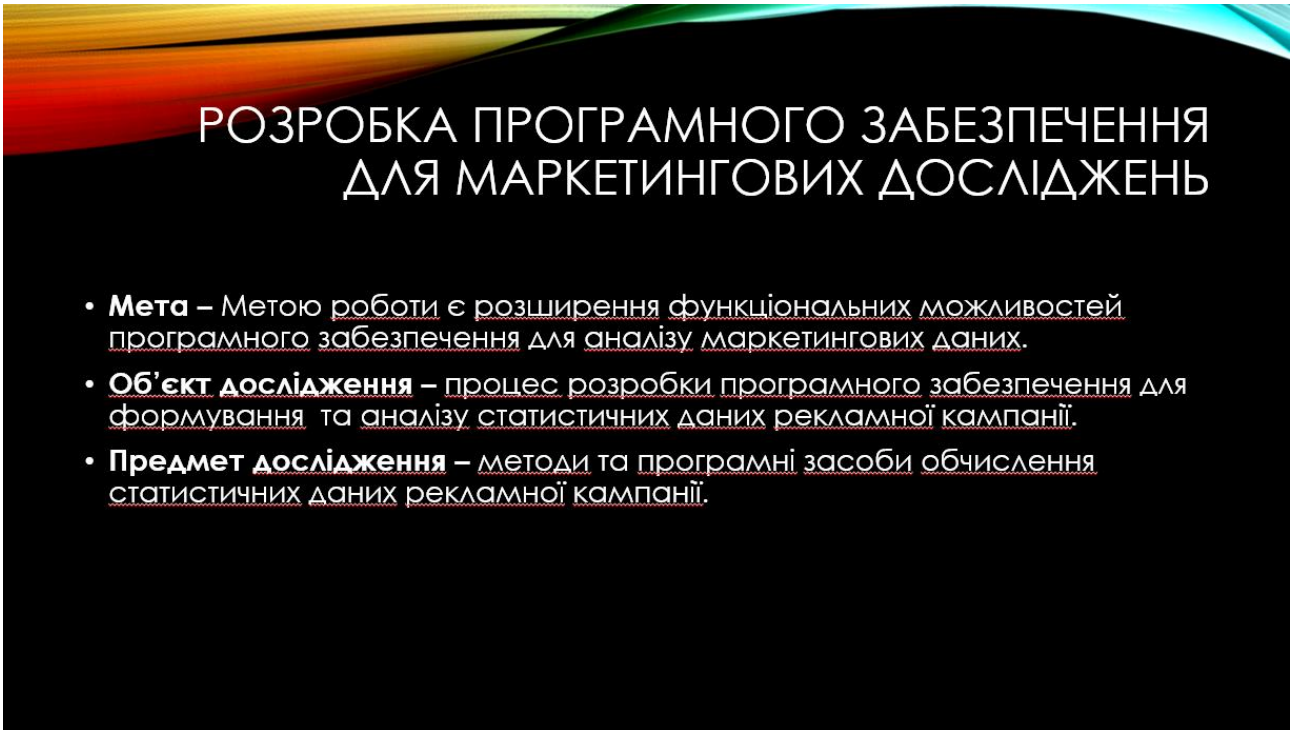


Рисунок Г.2 – Мета, об'єкт і предмет дослідження

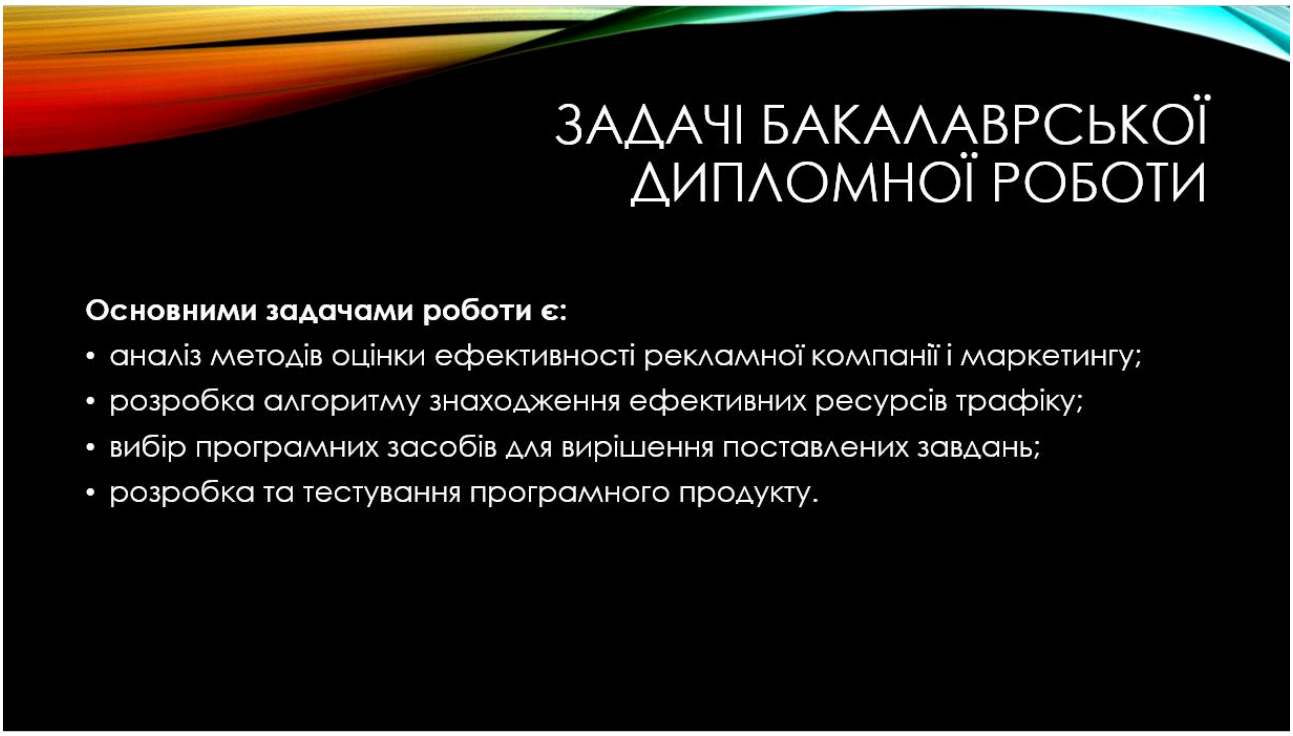


Рисунок Г.3 – Задачі бакалаврської дипломної роботи

Рисунок Г.4 – Наукова новизна

Рисунок Г.5 – Аналоги

Функціонал	Roistat	Google Analytics	Alytic	Marketing Analytics
Не прив'язано до певного сайту/ресурсу	+	-	+	+
Безкоштовний	-	+	-	+
Має функцію визначення ефективності рекламної кампанії	-	-	-	+
Має аналітику воронки для певного ресурсу трафіку	+	-	-	+
Має аналітику дослідження роботи відділу продажів	-	-	-	+

Рисунок Г.6 – Порівняння аналогів

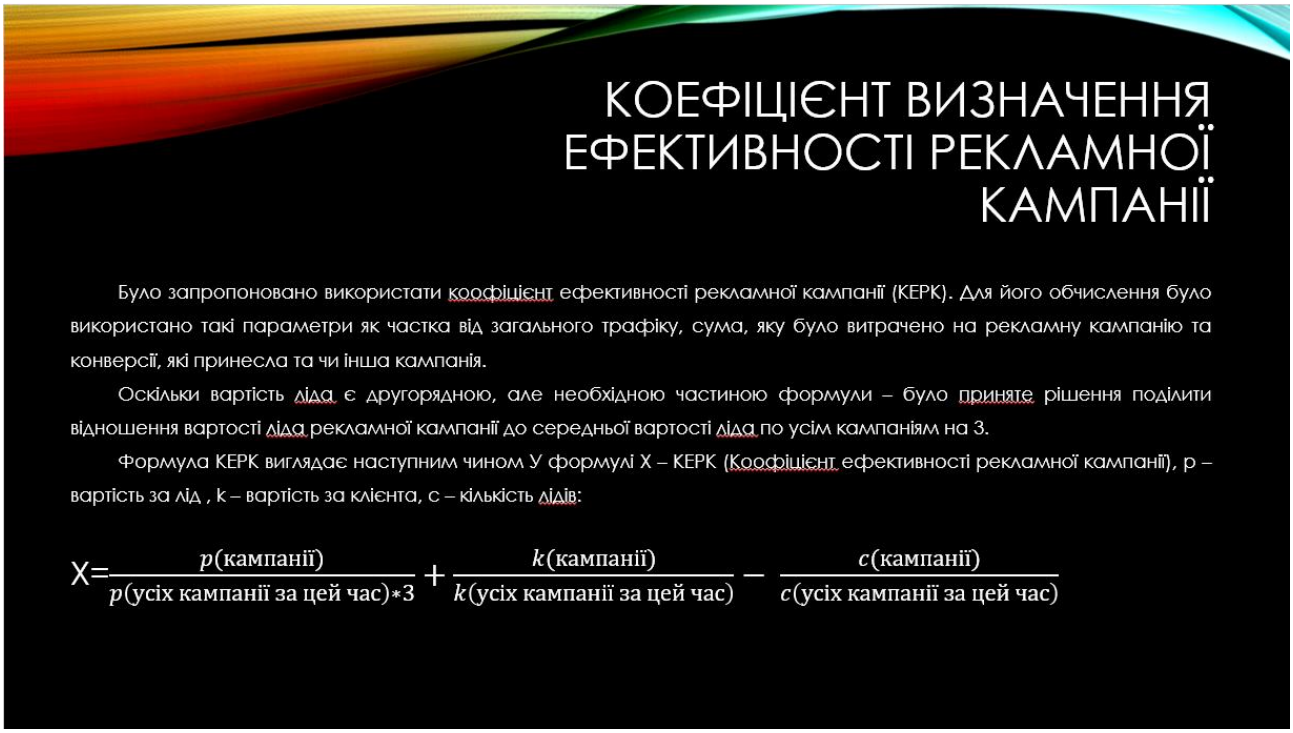


Рисунок Г.7 – Коефіцієнт визначення ефективності рекламної кампанії

	КЕРК	Визначення шляхом диференційного рівняння	Коофіцієнт ROI
Незалежність від відділу продажів	+	+	-
Відсутність хибної аналітики за умовою малих об'ємів даних	+	+/-	-
Відсутність необхідності у додаткових даних	+	-	+
Швидкий та простий у виконанні	+	-	+

Рисунок Г.8 – Порівняння коефіцієнту з іншими способами визначити ефективність рекламної кампанії

Рисунок Г.9 – Алгоритм роботи додатку

Рисунок Г.10 – Схематичне проектування дизайну вікон

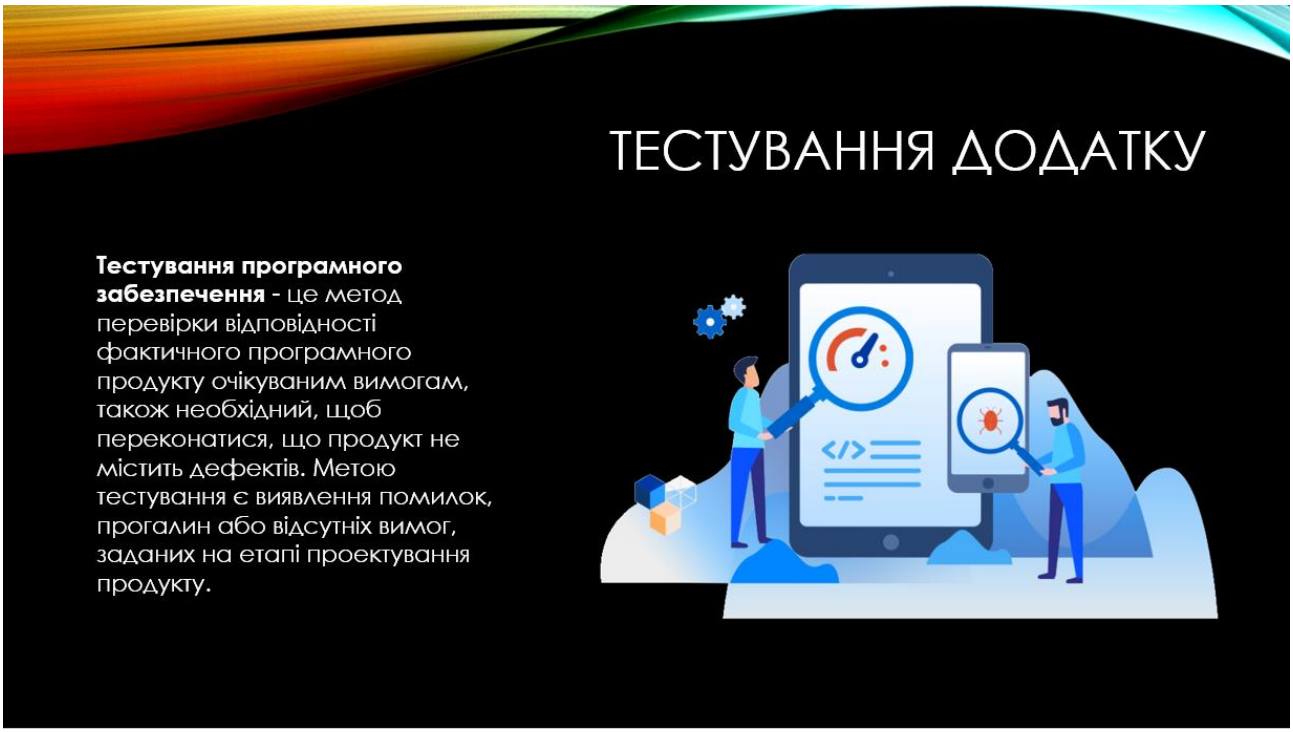


Рисунок Г.11 – Тестування

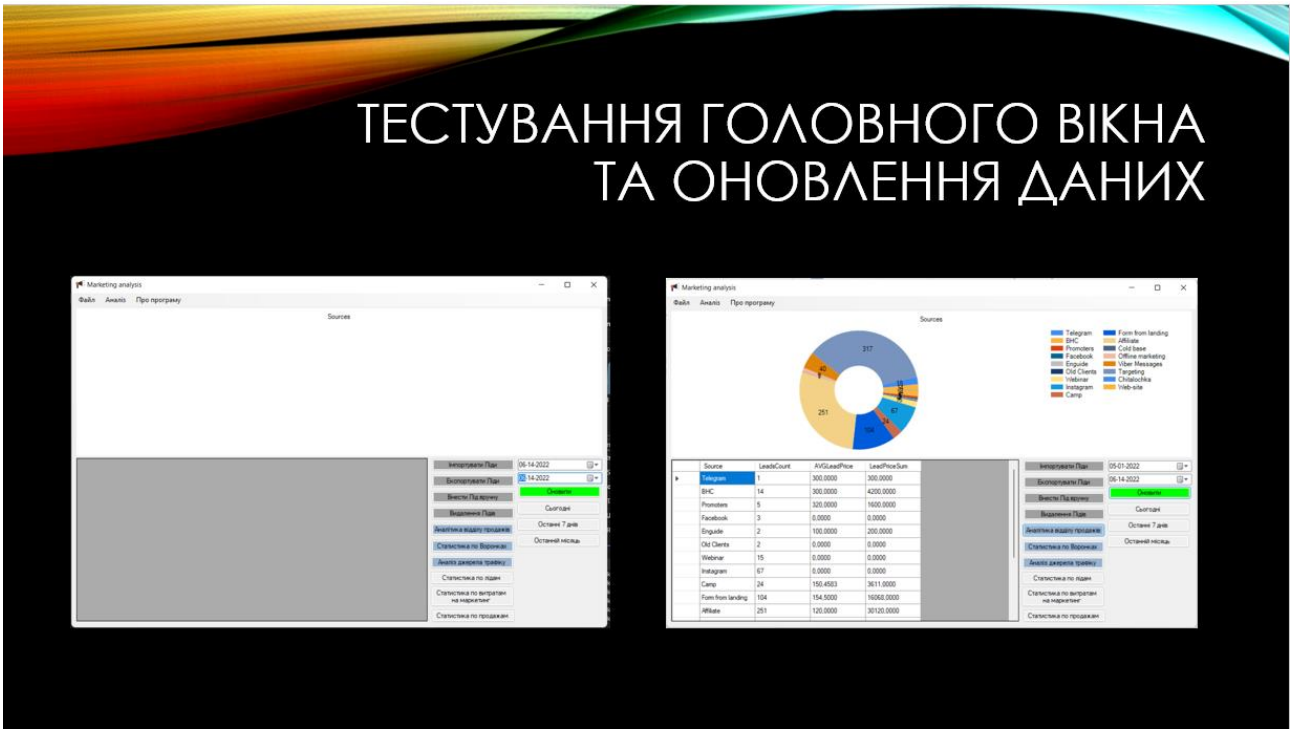


Рисунок Г.12 – Тестування: головне вікно

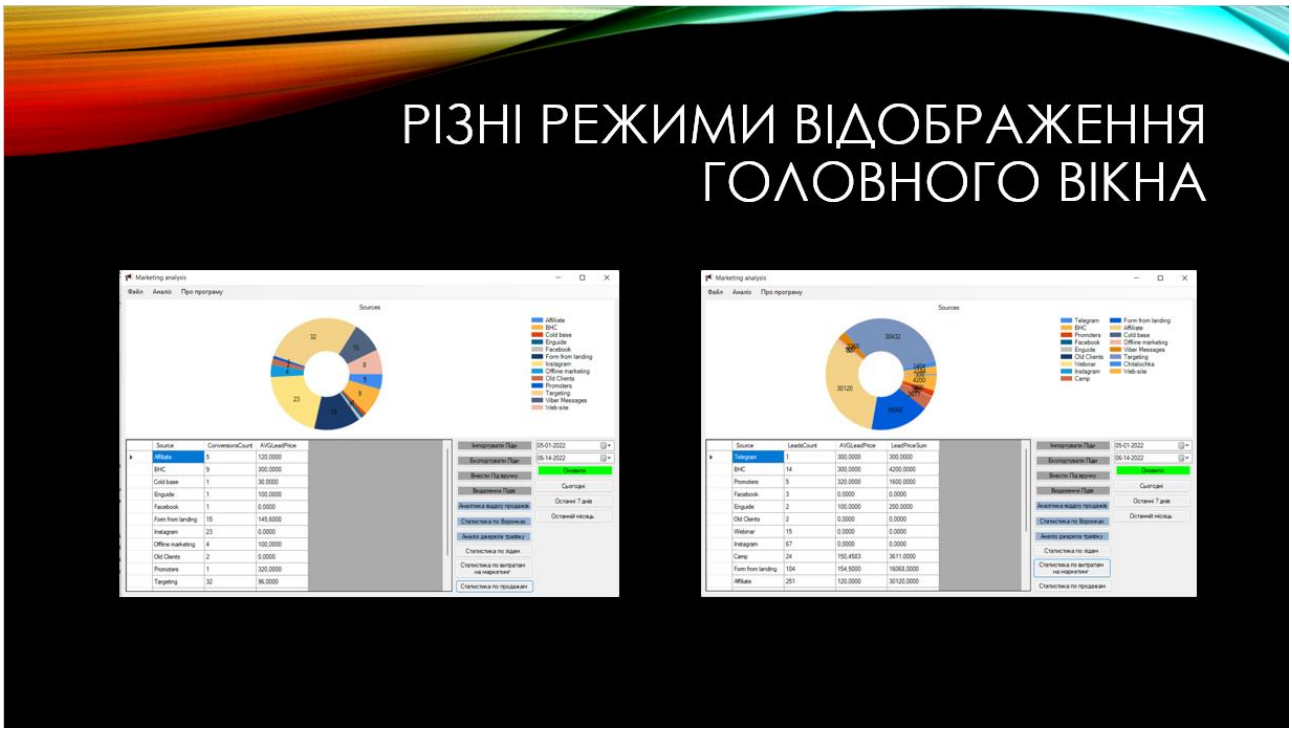


Рисунок Г.13 – Тестування: варіації головного вікна

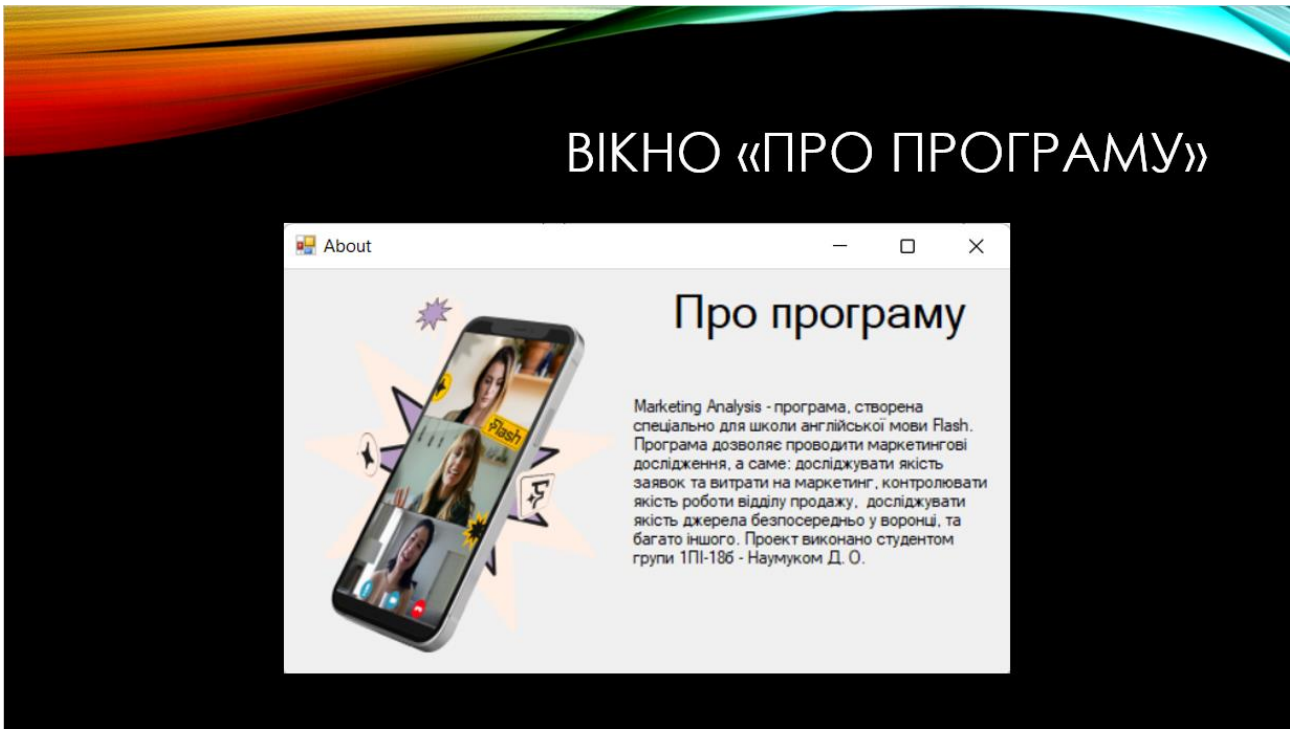


Рисунок Г.14 – Тестування: вікно про програму

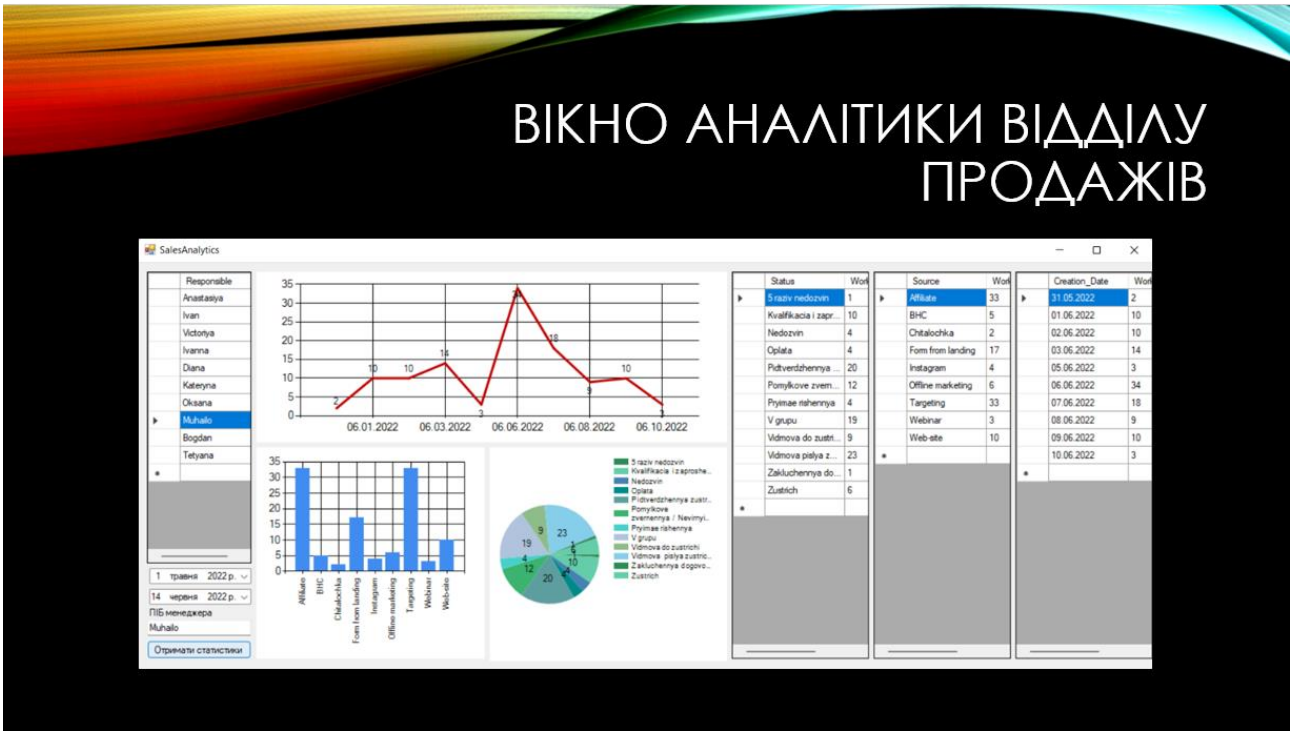


Рисунок Г.15 – Тестування: Вікно аналітики відділу продажів

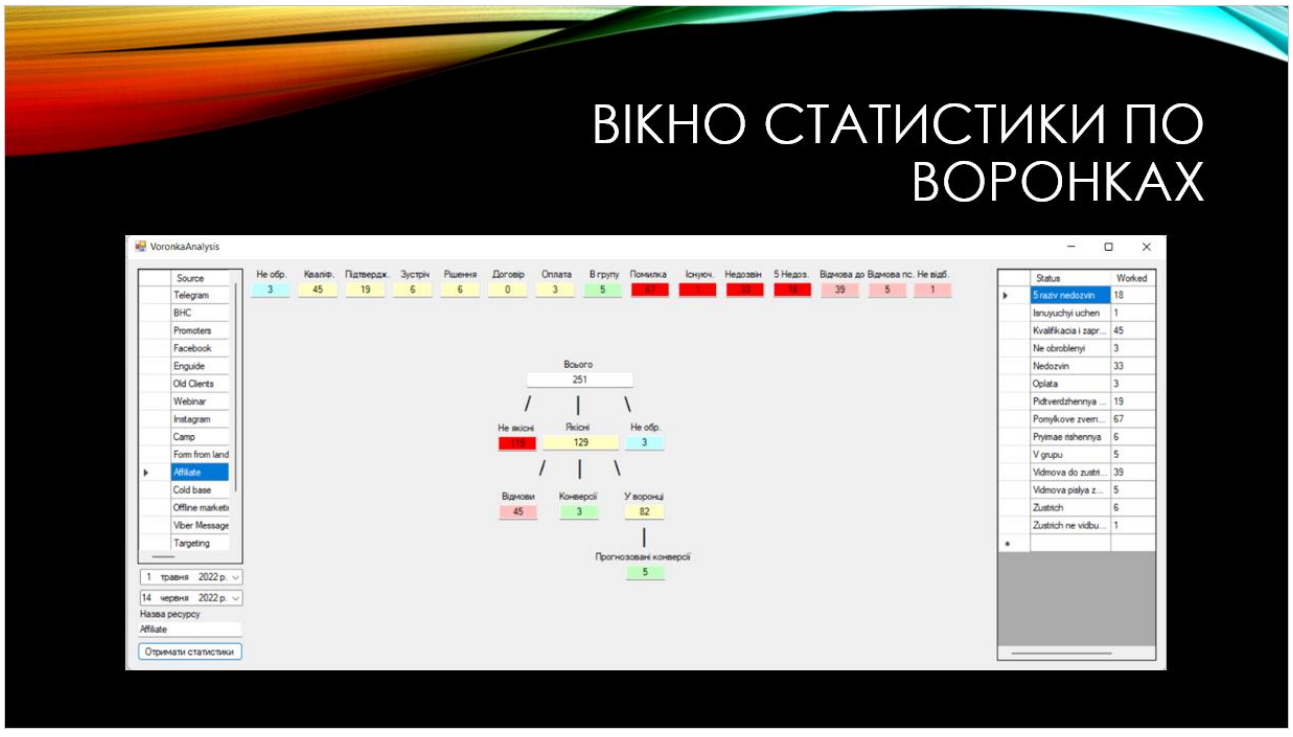


Рисунок Г.16 – Тестування: Вікно статистик по воронках

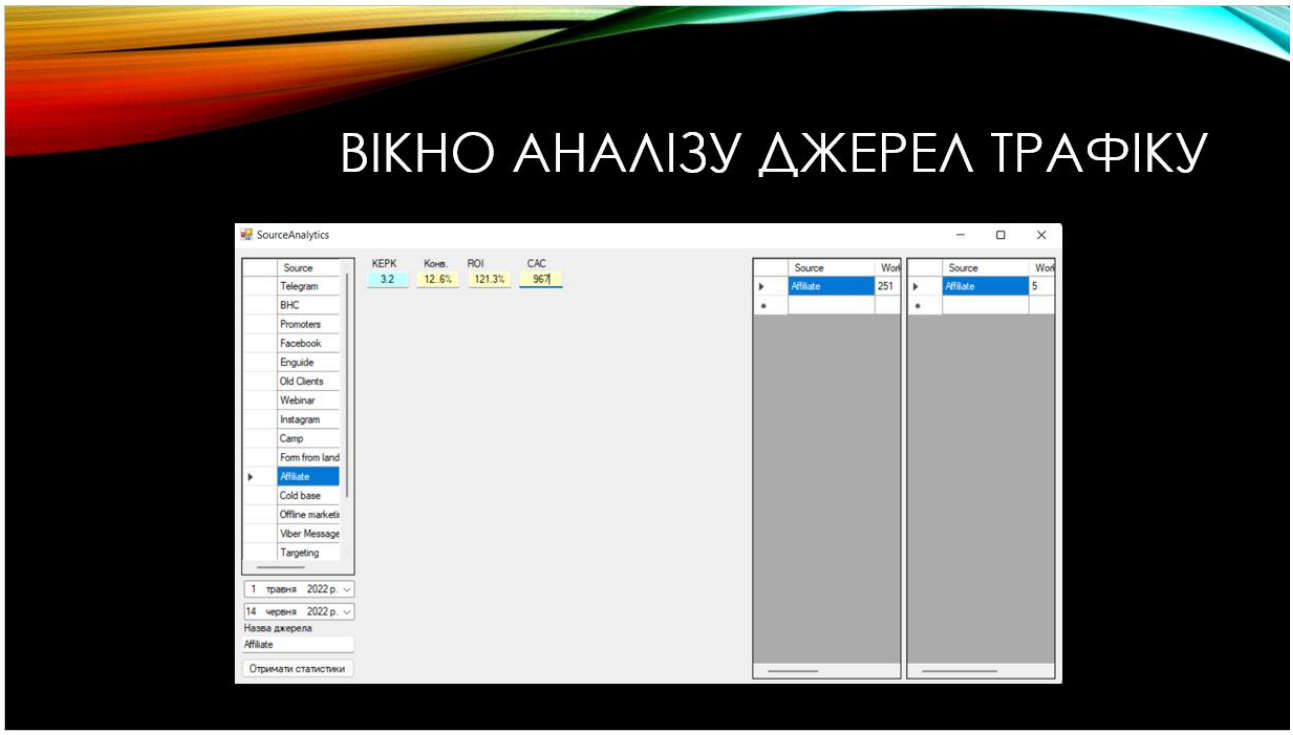


Рисунок Г.17 – Тестування: вікно аналізу джерел траіфку

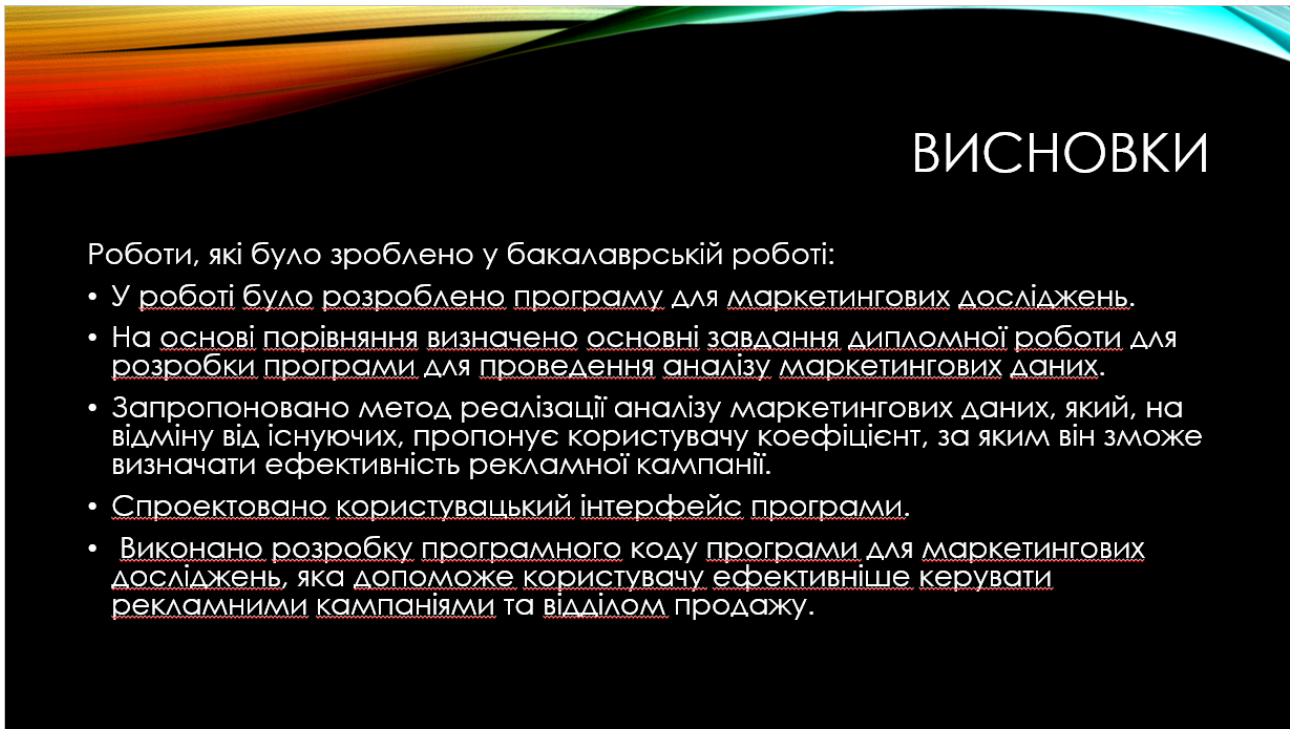


Рисунок Г.18 – Висновки

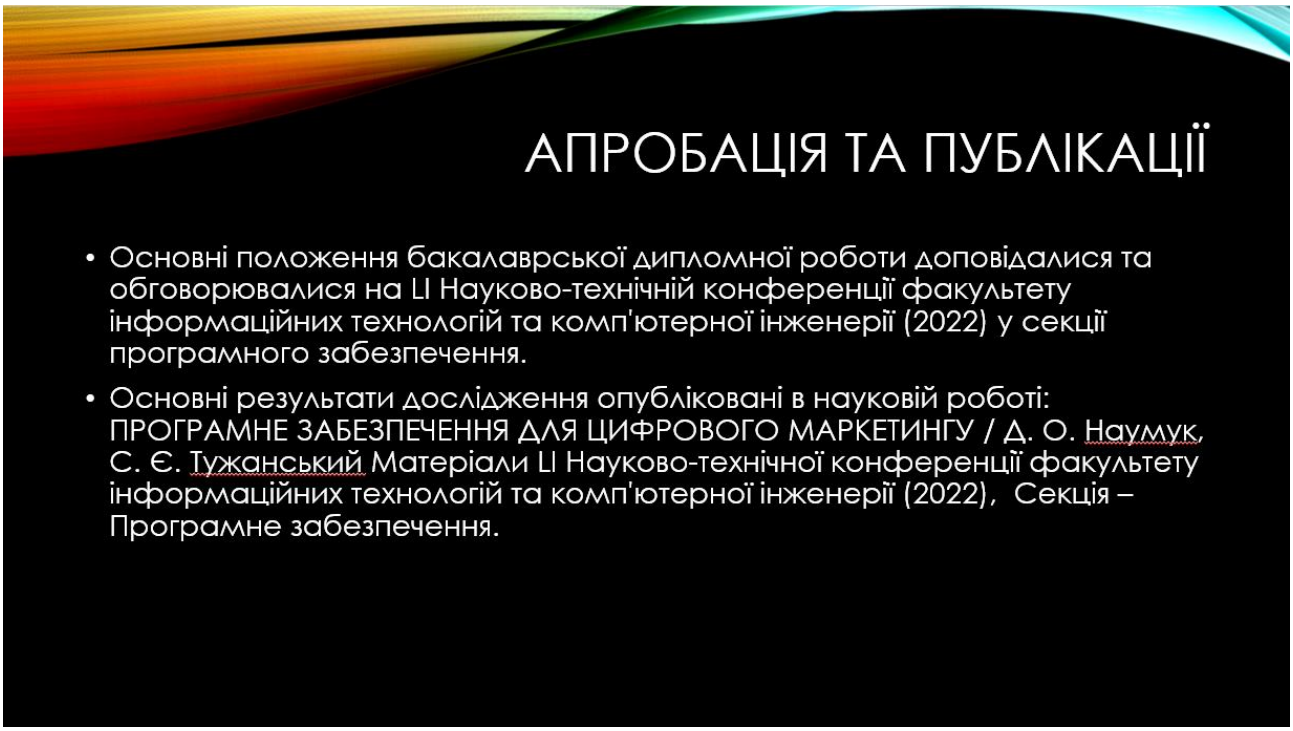


Рисунок Г.19 – Апробація та публікації