**МІНІ** **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВА РОБОТА**

**ТЕХНІЧНЕ ЗАВДАННЯ**

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: “Система аналізу цін споживчих товарів”

(назва теми)

|  |  |  |
| --- | --- | --- |
| **Студент групи** КП-01(02,03) | **Северин Юрій Юрійович**  (ПІБ) | \_\_\_\_\_\_\_\_\_  (підпис) |
|  |  |  |
| **Викладач**  **к.т.н, доцент кафедри СПіСКС** | **Радченко К.О.** | \_\_\_\_\_\_\_\_\_  (підпис) |

Київ – 2021

# **Анотація**

Покупка продовольчих товарів вже давно стала звичкою для кожної людини, проте далеко не кожен задумується над тим, що криється за ціноутворенням товарів.

У курсовій роботі було розглянуто створення інформаційної системи збереження та аналізу даних продовольчих товарів. Метою курсової роботи є створення такої інформаційної системи, що допомогла б її користувачам простіше зрозуміти тренди ціноутворення, по кожному товару, побачити найпопулярніші категорії товарів, та товари у певних магазинах.

Дана курсова робота складається з бази даних продовольчих товарів, та Windows Forms додатку для адміністративної взаємодії з цією базою даних.

У результаті розробки даної бази даних та даного консольного додатку було набуто практичні навички розробкисучасного програмного забезпечення, що взаємодіє з реляційними базами даних, а також здобуто навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. ули засвоєні навички роботи з інформаційними базами даних, утриманням та їх настройці.

# **Зміст**

[Анотація 2](#_Toc90405050)

[Зміст 3](#_Toc90405051)

[Вступ 4](#_Toc90405052)

[1. Аналіз інструментарію для виконання курсової роботи 5](#_Toc90405053)

[2. Структура бази даних 6](#_Toc90405054)

[3. Опис програмного забезпечення 7](#_Toc90405055)

[3.1. Загальна структура програмного забезпечення 7](#_Toc90405056)

[3.2. Опис модулів програмного забезпечення 7](#_Toc90405057)

[3.3. Опис основних алгоритмів роботи 7](#_Toc90405058)

[4. Аналіз функціонування засобів резервування/відновлення бази даних 9](#_Toc90405059)

[6. Аналіз результатів підвищення швидкодії виконання запитів 10](#_Toc90405060)

[7. Опис результатів аналізу предметної галузі 11](#_Toc90405061)

[8. Аналіз додаткового функціоналу 12](#_Toc90405062)

[Висновки 13](#_Toc90405063)

[Література 14](#_Toc90405064)

[Додатки 16](#_Toc90405065)

[А. Графічні матеріали 16](#_Toc90405066)

[Б. Фрагменти програмного коду 23](#_Toc90405067)

# **Вступ**

Дана робота присвячена дослідженням в області комп’ютерних технологій, систем зберігання даних, створення користувацьких інтерфейсів та аналізу даних.

Актуальність теми полягає в тому, що сучасне суспільство неможливо уявити без постійних покупок, проте далеко не всі замислюються про їх ефективність. Люди не здатні аналізувати таку кількість вхідних даних, і найчастіше просто не знають де їх взяти.

Додаток створений з метою мінімізації витрат, та надання користувачам можливості зручно аналізувати цінові коливання певних товарів, у певних магазинах. Розроблений мовою C# на базі Windows Forms. При проектуванні БД, була використана СУБД PosgreSQL.

База має такі сутності:

* Категорії товарів
* Товари
* Магазини
* Наявність у магазині
* Ціна наявності

База даних була спроектована відповідно до 3-ої нормальної форми,

швидкість її роботи була оптимізована шляхом додавання індексів, також вона було убезпечена додаванням механізмів реплікації та резервного копіювання. Окрім того, розроблений WindowsForms-додаток може надавати результати певного аналізу даних у базі.

# **1. Аналіз інструментарію для виконання курсової роботи**

Для виконання даної роботи у якості системи керування базами даних було обрано PostgreSQl. Такий вибір був зроблений у зв’язку з такими факторами:

* Відкрите ПЗ відповідає стандарту SQL - PostgreSQL - безкоштовне ПЗ з відкритим вихідним кодом. Ця СУБД є дуже потужною системою.
* Підтримка великої кількості типів даних, включно з власними
* Цілісність даних з усіма необхідними обмеженнями
* Надійність, безпека
* PostgreSQL не просто реляційна, а об’єктно-реляційна СУБД, що надає певні переваги
* Працює з багатьма типами мереж
* Велика місткість
* Велика спільнота – просто знайти вирішення потенційних проблем при розробці
* Це повністю open-source проект
* Розширення - існує можливість розширення функціоналу за рахунок своїх процедур

Для взаємодії з базою даних було обрано бібліотеку npgsql, оскільки:

* Добре підходить для зручного використання у мові програмування C#
* Розроблена спеціально для PostgreSQL
* Найпопулярніша для взаємодії з PostgreSQL у мові програмування C#
* Має чітку, зрозумілу та вичерпну документацію з хорошими прикладами

Для візуалізації результатів аналізу даних було обрано бібліотеку Windows.Forms, оскільки:

* Вона надає зручний інтерфейс для автоматичного будування графічних об’єктів
* Для графічних об’єктів наявна можливість дуже гнучкого налаштування з великою кількістю опцій для вигляду
* Наявна можливість будувати надзвичайно різноманітні графічні об’єкти

# **2. Структура бази даних**

База даних має такі таблиці з полями:

1. categories – категорії товару
   * category\_id - цілочисельний автоінкрементний унікальний ідентифікатор
   * name - текстове поле, назва категорії
2. goods - товари
   * good\_id - цілочисельний автоінкрементний унікальний ідентифікатор
   * category\_id - зовнішній ключ до таблиці категорій, який вказує на належність набору до серії, прив’язаний до поля category\_id таблиці категорій
   * name - текстове поле, назва товару
   * comment – необов’язкове текстове поле, коментар до товару
   * price - дробове значення ціни набору
   * barcode - текстове поле, штрих-код
3. shops - магазини
   * shopid - цілочисельний автоінкрементний унікальний ідентифікатор
   * name - текстове поле, назва магазину
   * adress - текстове поле, адреса магазину
   * rating - дробове значення, рейтинг магазину
4. availability – наявність товару в магазині
   * available\_id - цілочисельний автоінкрементний унікальний ідентифікатор
   * good\_id - зовнішній ключ до таблиці товарів, прив’язаний до поля good \_id
   * shop\_id - зовнішній ключ до таблиці товарів, прив’язаний до поля shopid
   * amount – цілочисельне значення, кількість наявного товару
5. prices – ціна за певною наявністю
   * price\_id - цілочисельний автоінкрементний унікальний ідентифікатор
   * available\_id- зовнішній ключ до таблиці наявності, прив’язаний до поля available\_id
   * date – дата, дата коли була певна ціна
   * price – дробове значення, ціна одиниці товару

# **3. Опис програмного забезпечення**

## **3.1. Загальна структура програмного забезпечення**

Розроблене програмне забезпечення містить такі компоненти:

1. База даних, що зберігає інформацію про категорії товарів, товари, магазини, наявність товарів та їх ціну, у 5-ти таблицях.
2. Засоби CRUD-функціоналу
3. Засоби псевдовипадкової генерації даних
4. Засоби пошуку, фільтрації та валідації
5. Засоби реплікації з сервером-реплікою
6. Засоби резервного копіювання
7. Засоби аналізу даних
8. Засоби візуалізації даних
9. Засоби імпорту та експорту даних у форматі CSV.

## **3.2. Опис модулів програмного забезпечення**

Розроблене програмне забезпечення було розбите на такі модулі:

1. model

Даний модуль напряму взаємодіє з базою даних. В цьому модулі знаходяться прототипи сутностей, запити до бази даних, їх обробка та пов’язані з цим операції.

1. View-controller

Даний модуль відповідає за виведення даних у додаток та отримання даних від користувача з додатку. Усі використання вводу та виводу знаходяться у цьому модулі.

## **3.3. Опис основних алгоритмів роботи**

При псевдовипадковій генерації даних для кожної з таблиць процес генерації був побудований так, аби генерувалися більш-менш адекватні дані для обраної предметної галузі та для обраної структури бази даних. Зокрема, у запитах були застосовані власноруч розроблені PostgreSQL-функції для забезпечення цілісності даних, унікальності тощо.

Аналіз цінових коливань допомагає нам знайти середнє значення ціни, а також побачити в якому напрямку може рухатись ціна на певні продукти. Також можна дізнатись категорію товарів, та товар, що представлений у найбільшій кількості в магазині.

# **4. Аналіз функціонування засобів резервування/відновлення бази даних**

Резервне копіювання необхідне для забезпечення безпечного та швидкого відновлення даних у разі втрати їх із бази даних. Тип резервного копіювання, який було реалізовано - повне. Це такий вид резервного копіювання, у якому щоразу копіюються повністю всі дані. Перевага такого різновиду резервного копіювання полягає у тому, що не потрібно об’єднувати різні файли для відновлення, натомість відновлюється все з одного файлу, за рахунок чого відновлення є помітно швидшим порівняно з іншими видами резервного копіювання. Було використане програмне копіювання файлів бази даних

Файли резервних копій зберігаються у папці DB\_Replica, яка знаходиться у корені проекту. Резервна копія створюється кожні 10 секунд, і файли автоматично копіюються з файлів бази даних, у кореневу папку проекту.

Також у курсовій роботі була виконана робота з тригерами, за допомогою яких було створено автоматичне резервне копіювання даних за тригером BEFORE DELETE.

# **6. Аналіз результатів підвищення швидкодії виконання запитів**

З метою підвищення швидкодії запитів для отримання деяких даних було використано індексування деяких полів 5-ти таблиць. Тип індексування, який був використаний - BTREE. Індексування було застосовано до полів id.

У випадку коли даних у таблиці багато (наприклад, 100 тисяч та більше) лінійний пошук стає заповільним, у зв’язку з чим для великих баз даних і потрібні індекси. Однак у разі малої бази даних індекси є неефективними, їхні алгоритми складніші і довші ніж просто лінійний пошук коли даних мало. У зв’язку з цим індекси і застосовуються лише для великих баз даних.

Тестування індексів було проведено на базі даних у якій у кожній з трьох проіндексованих таблиць наявно 100 тисяч записів.

Запити створення індексів:

1. CREATE INDEX ON prices USING BTREE(price\_id)
2. CREATE INDEX ON availability USING BTREE(available\_id)
3. CREATE INDEX ON shops USING BTREE(shop\_id)
4. CREATE INDEX ON goods USING BTREE(good\_id)
5. CREATE INDEX ON categories USING BTREE(category\_id)

Результати наведені на діаграмі у додатках.

# **7. Опис результатів аналізу предметної галузі**

У розробленому додатку наявний такий аналіз даних, що містяться у базі:

* Аналіз, що показує найпопулярніший продукт, у певному магазині. Виводиться в окремому вікні
* Аналіз, що показує найпопулярнішу категорію продуктів, у певному магазині. Виводиться в окремому вікні
* Аналіз, що показує середню ціну продукту, виводиться біля графіку ціни

# **8. Аналіз додаткового функціоналу**

У даній курсовій роботі при реалізації додатку для взаємодії з базою даних було також реалізовано додатковий функціонал на додачу до того, який зазначений у вимогах до даного проекту, а саме .

Також у Windows- Forms додатку була реалізована можливість виконання усіх CRUD-операцій - вставки, видалення, оновлення та читання, що надає користувачеві можливість їх зручнішого використання, аніж шляхом написання запитів.

При розробці додатку першою була створена база даних, потім за допомогою Scaffolding-у створені моделі сутностей за цією базою даних. Був використаний підхід code-first. Завдяки цьому можна зручно писати запити за допомогою C# та LINQ.

Приклад роботи імпорту та експорту наведено у додатках.

# **Висновки**

Під час виконання даної курсової роботи виконано таку роботу та отримано такі результати:

* Було розроблено базу даних, яка відповідає 3-ій нормальній формі та організована максимально зручно та просто
* Було реалізовано підхід code-frirst, за допомогою Scaffolding-у
* Було реалізовано резервування даних, за допомогою тригерів BEFORE DELETE
* Резервне копіювання було реалізовано повне, що дає можливість швидкого відновлення
* Була розроблена псевдовипадкова генерація для всіх таблиць, яка генерує реалістичні значення
* Була підвищена швидкодія запитів до бази даних шляхом індексування деяких полів таблиць
* Були розроблені засоби для аналізу даних із бази, які також надають можливість виводити графічне представлення його результату для наочності висновків
* Був розроблений зручний інтерфейс який також обробляє всі помилки, валідує дані та надає можливість виконання CRUD-операцій та фільтрації командами, а не SQL-запитами

У результаті виконання даної курсової роботи було набуто практичні навички розробкисучасного програмного забезпечення, що взаємодіє з реляційними базами даних, а також здобуто навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації.

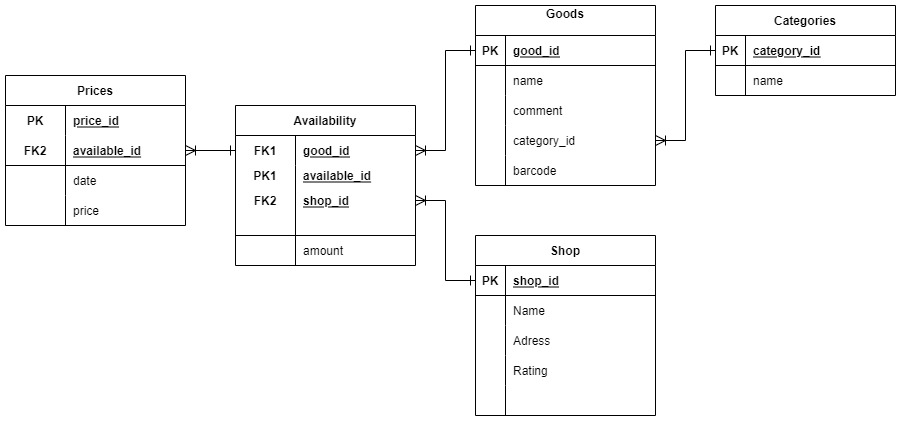
Завдяки виконанню даної роботи було здобуто вміння розробляти програмне забезпечення для реляційних баз даних, відбулося оволодіння основами використання СУБД, а також інструментальними засобами підтримки розробки додатків для подібних баз даних.

# **Література**

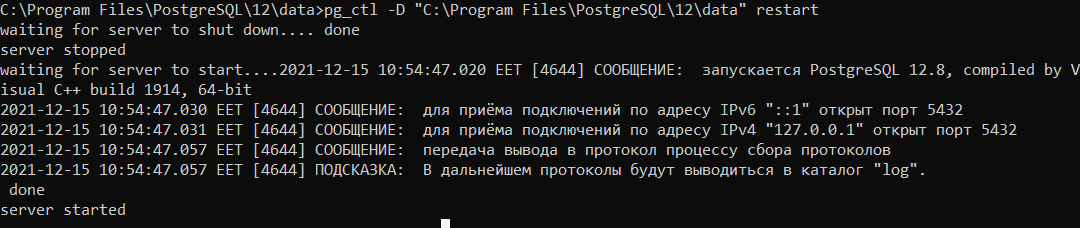
1. PostgreSQL 12.5 Documentation [Електронний ресурс] / The PostgreSQL Global Development Group // PostgreSQL: The World's Most Advanced Open Source Relational Database <https://www.postgresql.org/docs/12/index.html>.
2. Npgsql documentation [Електронний ресурс]: <https://www.npgsql.org/doc/>.
3. Windows Forms documentation: <https://docs.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-6.0>
4. DateTime — Time access and conversions [Електронний ресурс] // .Net: <https://docs.microsoft.com/en-us/dotnet/api/system.datetime?view=net-6.0>.
5. pgAdmin 4 4.28 documentation [Електронний ресурс] // pgAdmin PostgreSQL Tools: <https://www.pgadmin.org/docs/pgadmin4/4.28/index.html>.
6. Тюнинг базы Postgres [Електронний ресурс] // Highload//: <https://highload.today/tyuning-bazy-postgres/>.
7. Логическая репликация в PostgreSQL. Репликационные идентификаторы и популярные ошибки [електронний ресурс] // Habr.com //: <https://habr.com/ru/company/postgrespro/blog/489308/>
8. Half-HA cluster PostgreSQL на Windows 2012 [Електронний ресурс] // habr.com //: <https://habr.com/ru/post/308950/>
9. How to start and stop PostgreSQL server? [Електронний ресурс] // TablePlus // : <https://tableplus.com/blog/2018/10/how-to-start-stop-restart-postgresql-server.html>
10. Quick Setup Chapter 30. Logical Replication [Елекронний ресурс] // posgresql.com //: <https://www.postgresql.org/docs/12/logical-replication-quick-setup.html>

# **Додатки**

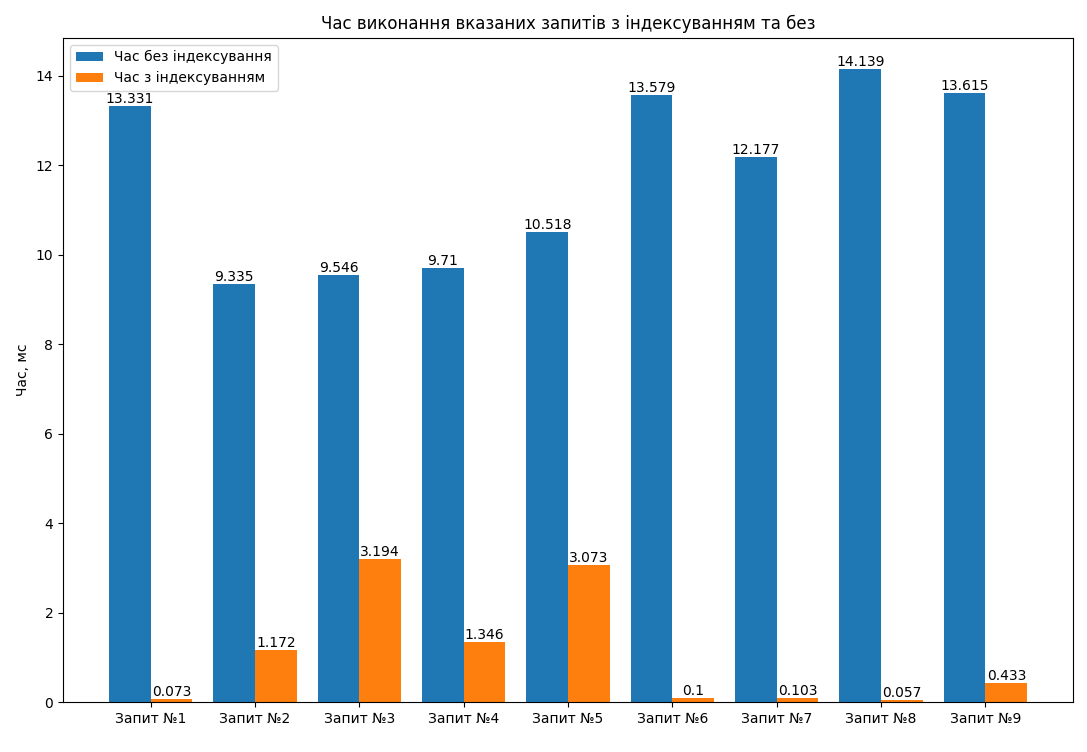
## **А. Графічні матеріали**



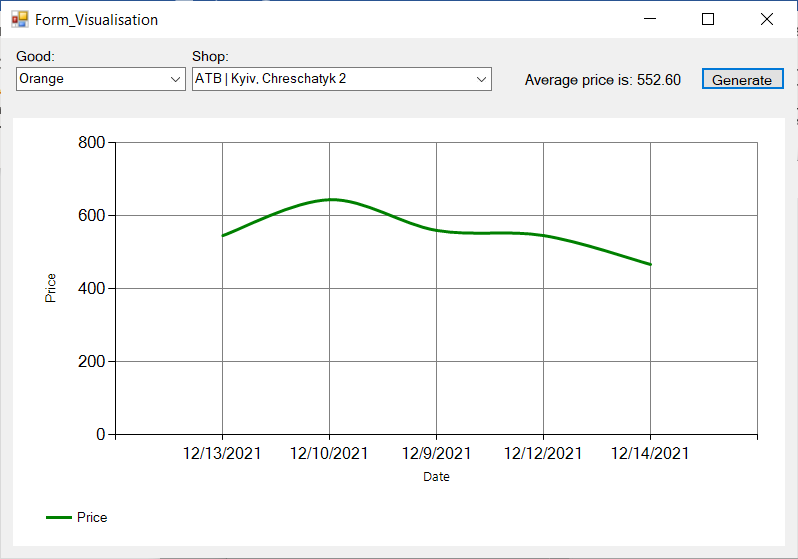
Структура бази даних



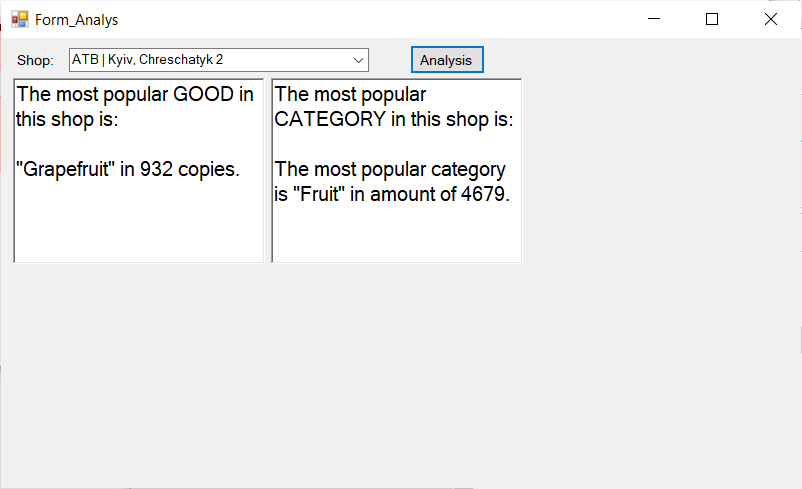
Перезапуск основного сервера



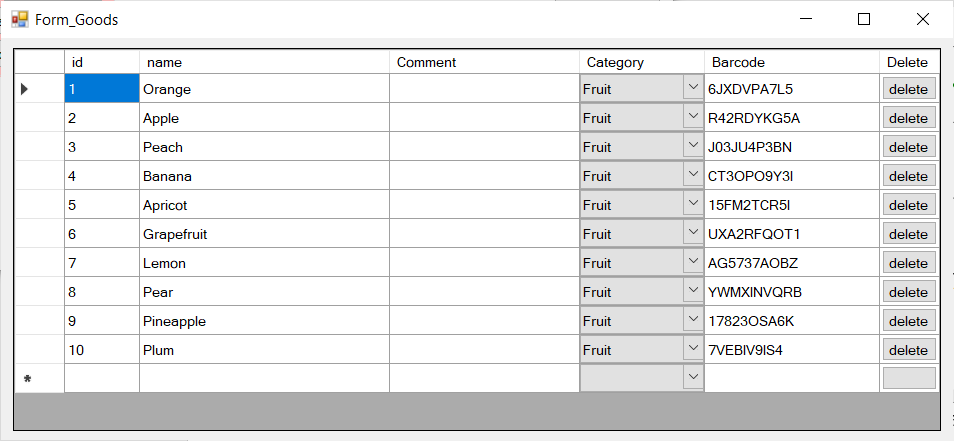
Порівняння часу виконання вказаних у відповідному розділі запитів з індексуванням та без



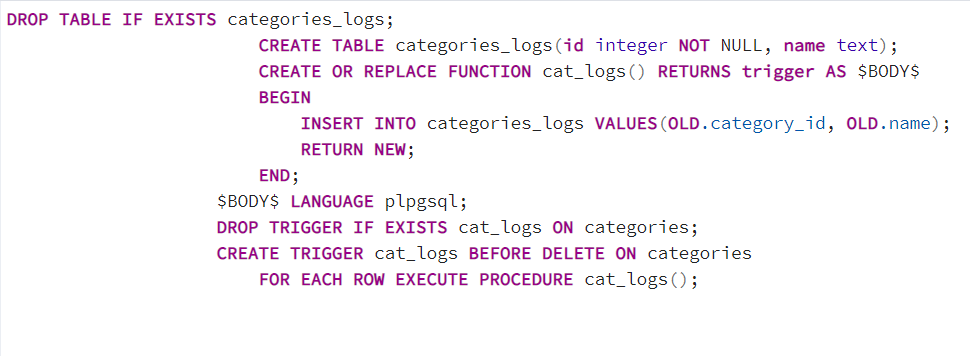
Приклад графіка до аналізу цінових коливань



Приклад визначення топу продуктів і категорій в магазині



Приклад графічного інтерфейсу додатку



Приклад команди на створення тригеру

## **Б. Фрагменти програмного коду**

|  |
| --- |
| Зв’язок з базою даних |
| modelBuilder.Entity<AvailabilityLog>(entity =>  {  entity.HasNoKey();  entity.ToTable("availability\_log");  entity.Property(e => e.Amount).HasColumnName("amount");  entity.Property(e => e.AvailableId).HasColumnName("available\_id").UseIdentityAlwaysColumn();  entity.Property(e => e.GoodId).HasColumnName("good\_id");  entity.Property(e => e.ShopId).HasColumnName("shop\_id");  });  modelBuilder.Entity<CategoriesLog1>(entity =>  {  entity.ToTable("categories\_log");  entity.Property(e => e.CategoryId)  .HasColumnName("category\_id")  .UseIdentityAlwaysColumn();  entity.Property(e => e.Name)  .IsRequired()  .HasColumnName("name");  });  modelBuilder.Entity<Category>(entity =>  {  entity.ToTable("categories");  entity.Property(e => e.CategoryId)  .HasColumnName("category\_id")  .UseIdentityAlwaysColumn();  entity.Property(e => e.Name)  .IsRequired()  .HasColumnName("name");  }); |

|  |
| --- |
| Приклади CRUD |
| private void button\_Analysis\_Click(object sender, EventArgs e)  {  using (dbContext db = new dbContext())  {  var popular = db.Availabilities.Include(t => t.Shop).Include(t => t.Good)  .Where(t => t.Shop.Name + " | " + t.Shop.Adress == (object)comboBoxShops.SelectedItem)  .OrderByDescending(t => t.Amount).First();  richTextBoxGood.Text += $"\n\n\"{popular.Good.Name}\" in {popular.Amount} copies.";  var vategory = db.Availabilities.Include(t => t.Shop).Include(t => t.Good)  .Where(t => t.Shop.Name + " | " + t.Shop.Adress == (object)comboBoxShops.SelectedItem)  .ToList();  var dict = new Dictionary<int, int>();  foreach(var a in vategory)  {  if(!dict.ContainsKey(a.Good.CategoryId))  {  dict.Add(a.Good.CategoryId, a.Amount);  }  else  {  dict[a.Good.CategoryId] += a.Amount;  }  }  (int, int) temp = (0, 0);  foreach(var a in dict)  {  if (a.Value > temp.Item2)  {  temp.Item2 = a.Value;  temp.Item1 = a.Key;  }  }  richTextBoxCategory.Text += $"\n\nThe most popular category is \"{db.Categories.Single(t => t.CategoryId == temp.Item1).Name}\" in amount of {temp.Item2}.";  }  richTextBoxGood.Visible = true;  richTextBoxCategory.Visible = true;  } |

|  |
| --- |
| Псевдовипадкова генерація |
| if (!db.Categories.Select(f => f.Name).ToList().Contains("Fruit"))  {  db.Categories.Add(new Category { Name = "Fruit" }); ;  }  db.SaveChanges();  int categoryID = 1;  categoryID = db.Categories.FirstOrDefault(f => f.Name == "Fruit").CategoryId;  List<string> fruitsnames = new List<string> { "Orange", "Apple", "Peach", "Banana", "Apricot", "Grapefruit", "Lemon", "Pear", "Pineapple", "Plum" };  for (int i = 0; i < 10; i++)  {  if (!db.Goods.Select(f => f.Name).ToList().Contains(fruitsnames[i]))  {  db.Goods.Add(new Good(0, fruitsnames[i], "", categoryID, RandomString(10)));  }  }  db.SaveChanges();  db.Shops.Add(new Shop(0, "ATB", "Kyiv, Chreschatyk 2", (decimal)9.9));  db.SaveChanges();  int shopID = db.Shops.FirstOrDefault(f => f.Name == "ATB").ShopId;  var avList = new List<Availability>();  for (int i = 0; i < 10; i++)  {  avList.Add(new Availability(0, db.Goods.FirstOrDefault(f => f.Name == fruitsnames[i]).GoodId, shopID, random.Next(1, 1000)));  } |

|  |
| --- |
| Приклад фільтрації |
| var popular = db.Availabilities.Include(t => t.Shop).Include(t => t.Good)  .Where(t => t.Shop.Name + " | " + t.Shop.Adress == (object)comboBoxShops.SelectedItem)  .OrderByDescending(t => t.Amount).First();  richTextBoxGood.Text += $"\n\n\"{popular.Good.Name}\" in {popular.Amount} copies.";  var vategory = db.Availabilities.Include(t => t.Shop).Include(t => t.Good)  .Where(t => t.Shop.Name + " | " + t.Shop.Adress == (object)comboBoxShops.SelectedItem)  .ToList(); |

|  |
| --- |
| Резервне копіювання та відновлення |
| private void Replication()  {  try  {  if (Directory.Exists(@".\DB\_replica(16466)"))  {  Directory.Delete(@".\DB\_replica(16466)", true);  }  Copy(@"C:\Program Files\PostgreSQL\12\data\base\16466", @".\DB\_replica(16466)");  }  catch{ }  Thread.Sleep(10\_000);  }  public static void Copy(string sourceDirectory, string targetDirectory)  {  var diSource = new DirectoryInfo(sourceDirectory);  var diTarget = new DirectoryInfo(targetDirectory);  CopyAll(diSource, diTarget);  }  public static void CopyAll(DirectoryInfo source, DirectoryInfo target)  {  Directory.CreateDirectory(target.FullName);  foreach (FileInfo fi in source.GetFiles())  {  Console.WriteLine(@"Copying {0}\{1}", target.FullName, fi.Name);  fi.CopyTo(Path.Combine(target.FullName, fi.Name), true);  }  foreach (DirectoryInfo diSourceSubDir in source.GetDirectories())  {  DirectoryInfo nextTargetSubDir =  target.CreateSubdirectory(diSourceSubDir.Name);  CopyAll(diSourceSubDir, nextTargetSubDir);  }  } |

|  |
| --- |
| Аналіз популярності |
| var dict = new Dictionary<int, int>();  foreach(var a in vategory)  {  if(!dict.ContainsKey(a.Good.CategoryId))  {  dict.Add(a.Good.CategoryId, a.Amount);  }  else  {  dict[a.Good.CategoryId] += a.Amount;  }  }  (int, int) temp = (0, 0);  foreach(var a in dict)  {  if (a.Value > temp.Item2)  {  temp.Item2 = a.Value;  temp.Item1 = a.Key;  }  }  richTextBoxCategory.Text += $"\n\nThe most popular category is \"{db.Categories.Single(t => t.CategoryId == temp.Item1).Name}\" in amount of {temp.Item2}.";  }  richTextBoxGood.Visible = true;  richTextBoxCategory.Visible = true; |

|  |
| --- |
| Створення графіку |
| private void buttonGenerate\_Click(object sender, EventArgs e)  {  chart1.Series.Clear();  chart1.Series.Add("Price");  chart1.Series["Price"].BorderWidth = 3;  chart1.Series["Price"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;  decimal avg = 0;  int c = 0;  using (dbContext db = new dbContext())  {  var data = db.Prices.Include(i => i.Available.Good).Include(i => i.Available.Shop)  .Where(f => f.Available.Good.Name == (string)comboBoxGood.SelectedItem)  .Where(f => (f.Available.Shop.Name + " | " + f.Available.Shop.Adress) == (string)comboBoxShop.SelectedItem).ToList();  c = data.Count;  foreach (var a in data)  {  avg += a.Price1;  chart1.Series["Price"].Points.AddXY(a.Date.ToShortDateString(), a.Price1.ToString());  }  }  avg /= c;  label3.Visible = true;  label3.Text = "Average price is: " + avg.ToString();  } |

|  |
| --- |
| Приклад деяких методів view-controller |
| void GetData()  {  using (dbContext db = new dbContext())  {  var goodsList = db.Goods.ToList();  dataGridView.DataSource = null;  try  {  if (dataGridView.CurrentCell != null)  {  dataGridView.CurrentCell.Selected = false;  }  dataGridView.Rows.Clear();  }  catch (InvalidOperationException)  {  MessageBox.Show("Press Enter to finish editing.", "Enter", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);  Close();  return;  }  foreach (var a in goodsList)  {  dataGridView.Rows.Add(a.GoodId, a.Name, a.Comment, (string)db.Categories.ToList().Find(c => c.CategoryId == a.CategoryId).Name, a.Barcode, "Delete");  }  }  } |