**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

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: “База даних наборів LEGO”

**Студентка**
**групи** КП-01 **Максименко Дмитро Юрійович** _____ (підпис)

**Асистент кафедри**
**СПіСКС** **Радченко К.О.** _____ (підпис)

Захищено з оцінкою _____

Київ – 2021

## Анотація

Курсова робота складається з бази даних відомостей про дорожню обстановку з камер (дані камери, про автівки та їх власників) та консольного додатку для адміністративної взаємодії з цією базою даних, написаного мовою програмування C#, який дозволяє виконувати певні операції над нею (читання, оновлення, запис, видалення, псевдовипадкова генерація записів, аналіз даних тощо).

У результаті розробки даної бази даних та даного консольного додатку було набуто практичні навички розробки сучасного програмного забезпечення, що взаємодіє з реляційними базами даних, а також здобуто навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проєктної документації.

**Зміст**

## Вступ

Галузь застосування розробки це система організації дорожнього руху: зчитування зображень доріг або номерних знаків автомобілів з метою подальшого зберігання та аналізу (заторів, пошуку автомобілів з певним номерним знаком тощо). Метою курсової роботи є набуття практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з реляційними базами даних, а також здобуття навичок оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проєктної документації, на прикладі розробки системи, що допоможе в організації дорожнього руху, за для аналізу стану на дорогах та вичислення за номерними знаками. База має такі сутності:

- Камери
- Автомобілі
- Власники автомобілів

База даних була спроєктована відповідно до 3-ої нормальної форми, швидкість її роботи була оптимізована шляхом додавання індексів, також вона було убезпечена додаванням механізму резервного копіювання. Окрім того, розроблений консольний додаток може надавати результати певного аналізу даних у базі..

## 1. Аналіз інструментарію для виконання курсової роботи

1. 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 опен-сорсний проєкт;
    - розширення - існує можливість розширення функціоналу за рахунок своїх процедур.
2. 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.

3. Для візуалізації результатів аналізу даних було обрано бібліотеку ScottPlot, оскільки:
    - вона надає зручний інтерфейс для автоматичного будування графічних об’єктів;
    - наявна можливість будувати надзвичайно різноманітні графічні об’єкти.

## 2. Структура бази даних

База даних має такі таблиці з полями:

1. cameras - камери
    - id - цілочисельний автоінкрементний унікальний ідентифікатор, що позначає номер камери;
    - locationaddress - текстове поле, назва місця розташування камери.
2. cars - машини, зафіксовані на камерах
    - id - цілочисельний автоінкрементний унікальний ідентифікатор;
    - carnumber - значення номеру автівки за стандартами МВС України;
    - ownerid - зовнішній ключ до таблиці власники автівок, який вказує на належність машини до власника, прив’язаний до поля id таблиці власники автівок;
    - cameraid - зовнішній ключ до таблиці камер, який вказує на належність машини до камери, на якій вона була зафіксована, прив’язаний до поля id таблиці камер;
    - speed - чисельне значення швидкості автівки, яка була зафіксована на камеру;
    - timeofregistrationc - час, коли була зафіксована автівка на камеру.
3. carowners - власники автівок
    - id - цілочисельний автоінкрементний унікальний ідентифікатор;
    - fullname - текстове поле, ПІБ власника(ці).

# 3. Опис програмного забезпечення

## 3.1. Загальна структура програмного забезпечення

Розроблене програмне забезпечення містить такі компоненти:

1. База даних, що зберігає інформацію про камери, автівки та власників автівок у трьох таблицях.
2. Засоби CRUD-функціоналу.
3. Засоби псевдовипадкової генерації даних.
4. Засоби пошуку, фільтрації та валідації.
5. Засоби резервного копіювання з можливістю вибору версії
6. Засоби аналізу даних

## 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 запити до бази даних, їх обробка та пов’язані з цим операції.
2. view
   Даний модуль відповідає за виведення даних у консоль та отримання даних від користувача з консолі. Усі використання функцій input та print знаходяться у цьому модулі.
3. controller
   Цей модуль пов’язує model та view і забезпечує обробку виключних ситуацій. Він отримує інформацію з model та передає її або іншу потрібну інформацію для відображення у view.

## 3.3. Опис основних алгоритмів роботи

При псевдовипадковій генерації даних для кожної із таблиць процес генерації був побудований так, аби генерувалися 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.

Для генерації даних шляхом імпорту відбувається обробка вхідних даних для забезпечення коректного їх занесення у таблиці, оскільки стиль

датасетів, обраних для цієї мети, не повністю відповідає структурі бази даних.

Для аналізу різних даних було використано генерацію графіку та діаграм.

## 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 використано вбудоване резервне копіювання системи керування базами даних PostgreSQL.

Розроблений консольний додаток надає можливість користувачеві керувати резервним копіюванням та відновленням з консолі. Файли резервних копій зберігаються у папці backup, яка знаходиться у корені проєкту. При виконанні відновлення додаток просить ввести назву файлу бекапу і потім база даних відновлюється відповідно до бекапу з вказаної папки.

## 5. Аналіз результатів підвищення швидкодії виконання запитів

З метою підвищення швидкодії запитів для отримання деяких даних було використано індексування поля однієї таблиць. Тип індексування, який був використаний - GIN. Індексування було застосовано до таких полів: поле cameraid таблиці cars.

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.

Запити створення індексів:

1. CREATE INDEX index ON cars USING hash (cameraid).

## 6. Опис результатів аналізу предметної галузі

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

- аналіз швидкостей, з якими їздять в місті; додається графічне представлення у вигляді графіка.
- аналіз кількості машин у власників автомобілів, що підкріплюється діаграмою.
- аналіз кількості машин на дорогах, що підкріплюється діаграмою.

Приклади графіків та діаграми наведені у додатках.

## Висновки

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

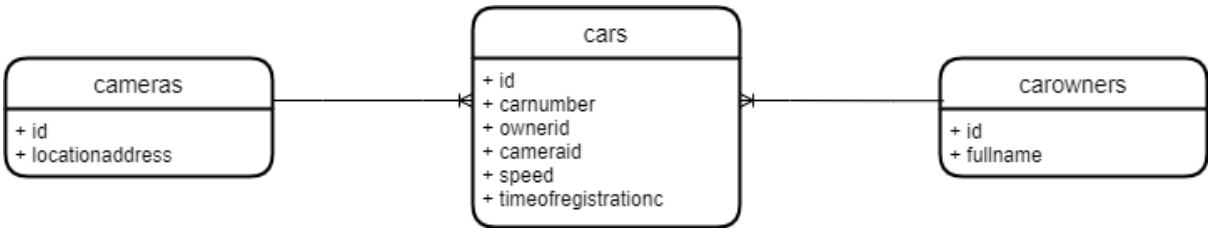
- Було розроблено базу даних, яка відповідає 3-ій нормальній формі та організована максимально зручно та просто
- Резервне копіювання було реалізовано повне, що дає можливість швидкого відновлення
- Була розроблена псевдовипадкова генерація для всіх таблиць, яка генерує реалістичні значення
- Була підвищена швидкодія запитів до бази даних шляхом індексування деяких полів деяких таблиць
- Були розроблені засоби для аналізу даних із бази, які також надають можливість виводити графічне представлення його результату для наочності висновків
- Був розроблений зручний консольний 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 розробки сучасного програмного забезпечення, що взаємодіє з реляційними базами даних, а також здобуто навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проєктної 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.

## Додатки

### А. Графічні матеріали



Структура бази даних

```
        Welcome!

Enter command ('output', 'dataBase', 'statistics', 'exit'): dataBase

Enter command ('insert', 'update', 'delete', 'random', 'backup', 'reservation'): random

Enter entity ('camera', 'car', 'carowner'): camera

How many cameras? 2

Enter command ('output', 'dataBase', 'statistics', 'exit'): exit

        Bye!
```

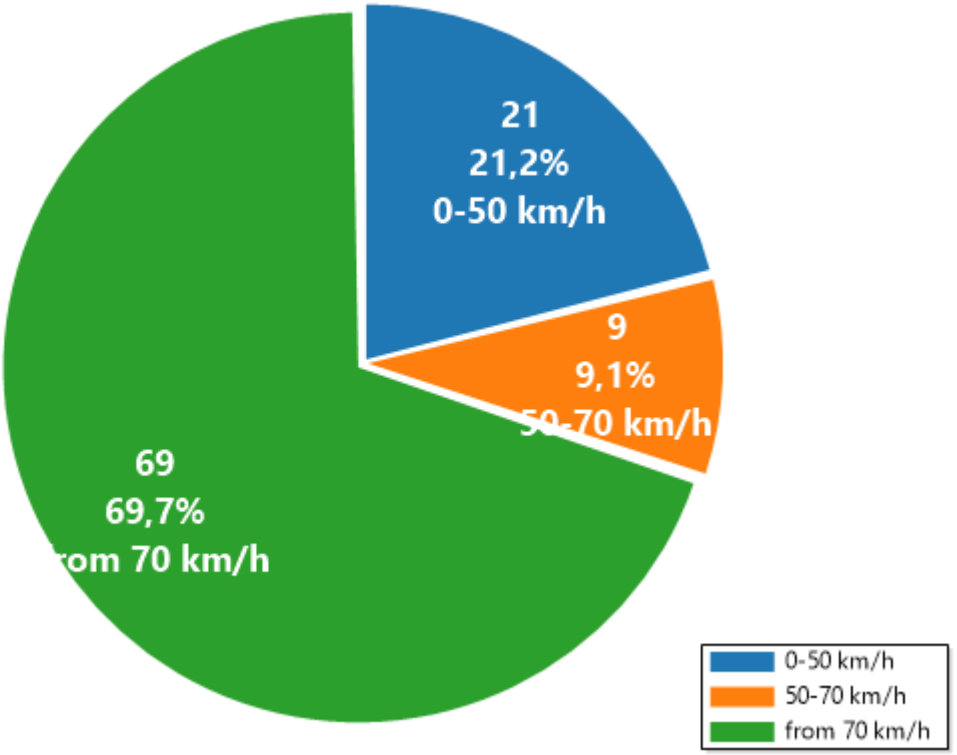
Консольний інтерфейс

```
        Welcome!

Enter command ('output', 'dataBase', 'statistics'

Enter command ('cars', 'index'): index
time without index: 4
time with index: 4
```

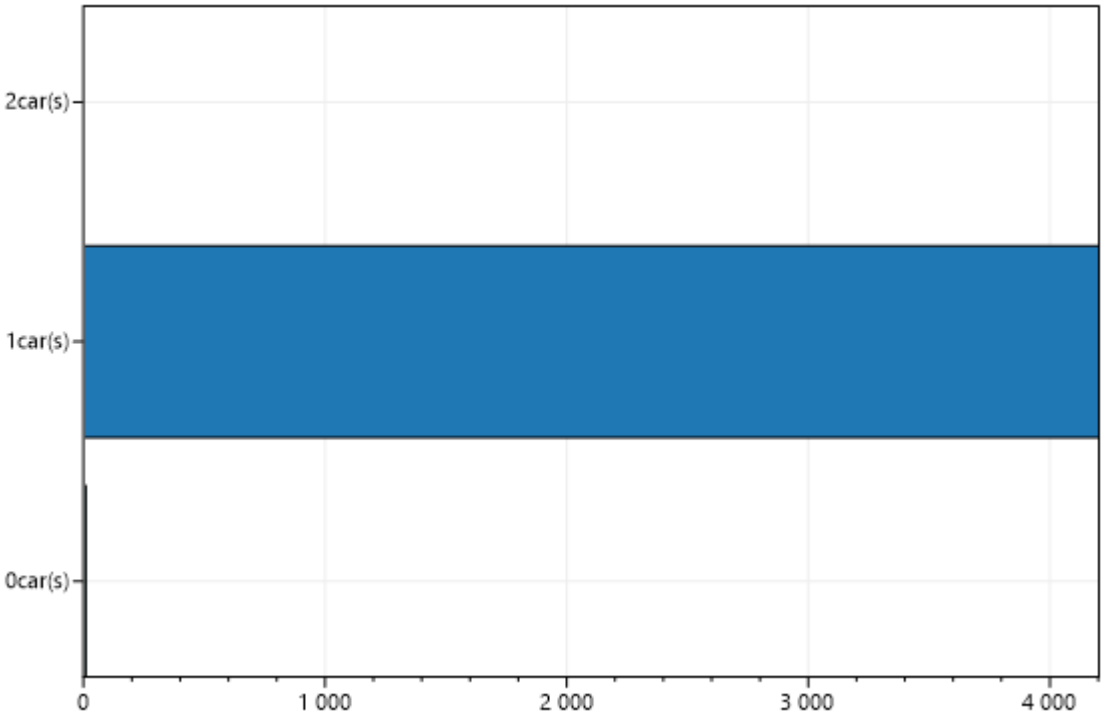
Індексування з малою кількістю даних

```
        Welcome!

Enter command ('output', 'dataBase', 'statistics', 'exit'): output

Enter command ('cars', 'index'): index
time without index: 22
time with index: 8
```

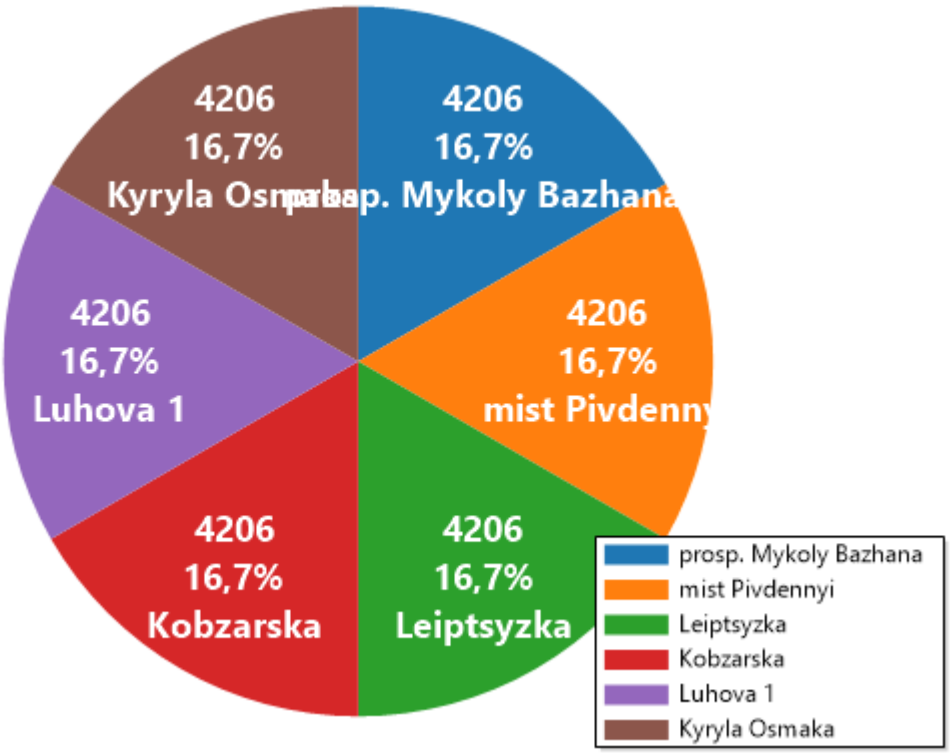
Індексування з великою кількістю даних (понад 3000)




Приклад діаграми швидкостей




Приклад графіка кількість автівок у власників




Приклад діаграми кількості автівок на конкретних вулицях

## Б. Фрагменти програмного коду

Приклади CRUD

```
public void InsertCar(int id, string carnumber, int ownerid, int cameraid, int speed, DateTime timeofregistrationc)
   {
      if(id == -1)
      {
         id = NextCarId();
      }
      if(!CheckCarNumber(carnumber))
      {
         return;
      }
      connection.Open();
      try
      {
         using (var command = new NpgsqlCommand("INSERT INTO \"cars\" (id, carnumber, ownerid, cameraid, speed,
timeofregistrationc) VALUES (@id, @carnumber, @ownerid, @cameraid, @speed, @timeofregistrationc)", connection))
         {
            command.Parameters.AddWithValue("id", id);
            command.Parameters.AddWithValue("carnumber", carnumber); //TODO check
            command.Parameters.AddWithValue("ownerid", ownerid);
            command.Parameters.AddWithValue("cameraid", cameraid);
            command.Parameters.AddWithValue("speed", speed);
            command.Parameters.AddWithValue("timeofregistrationc", timeofregistrationc); //???????
            command.ExecuteNonQuery();
         }
      }
      catch(NpgsqlException ex)
      {
```

```
            Console.WriteLine($"{ex.Message}");
        }
        connection.Close();
    }
    public void UpdateCar(int id, string carnumber, int ownerid, int cameraid, int speed, DateTime timeofregistrationc)
    {
        if(!CheckCarNumber(carnumber))
        {
            return;
        }
        connection.Open();
        try
        {
            using (var command = new NpgsqlCommand("UPDATE \"cars\" SET carnumber = @carnumber, ownerid = @ownerid, cameraid =
@cameraid, speed = @speed, timeofregistrationc = @timeofregistrationc WHERE id = @id", connection))
            {
                command.Parameters.AddWithValue("id", id);
                command.Parameters.AddWithValue("carnumber", carnumber); //TODO check
                command.Parameters.AddWithValue("ownerid", ownerid);
                command.Parameters.AddWithValue("cameraid", cameraid);
                command.Parameters.AddWithValue("speed", speed);
                command.Parameters.AddWithValue("timeofregistrationc", timeofregistrationc);
                command.ExecuteNonQuery();
            }
        }
        catch(NpgsqlException ex)
        {
            Console.WriteLine($"{ex.Message}");
        }
        connection.Close();
    }
    public void DeleteCar(int id)
    {
        connection.Open();
        try
        {
            int ownerId = -1;
            using (var command = new NpgsqlCommand("SELECT * FROM \"cars\" WHERE id = @id", connection))
            {
                command.Parameters.AddWithValue("id", id);
                var reader = command.ExecuteReader();
                ownerId = (int)reader[2];
                reader.Close();
            }
            using (var command = new NpgsqlCommand("DELETE FROM \"cars\" WHERE id = @id", connection))
            {
                command.Parameters.AddWithValue("id", id);
                command.ExecuteNonQuery();
            }
            List<int> listOfCars = ListOfCar(ownerId);
            if(listOfCars.Count == 1)
            {
                DeleteCarOwner(ownerId);
            }
        }
        catch(NpgsqlException ex)
        {
            Console.WriteLine($"{ex.Message}");
        }
        connection.Close();
    }
```

## Індексування

```
class Index
{
    private Model model;
```

```
   public Index(Model model)
   {
      this.model = model;
   }
   public long[] _Index()
   {
      var result = model.TakeCars(1);

      Stopwatch sw = new Stopwatch();
      sw.Start();
      result = model.TakeCars(1);
      sw.Stop();

      long[] list = new long[2];
      list[0] =  sw.ElapsedMilliseconds;

      model.Connection().Open();
      using (var command = new NpgsqlCommand("CREATE INDEX index\n" + "ON \"cars\" USING hash (\"cameraid\");",
model.Connection())){}
      model.Connection().Close();

      sw = new Stopwatch();
      sw.Start();
      result = model.TakeCars(1);
      sw.Stop();

      list[1] =  sw.ElapsedMilliseconds;

      model.Connection().Open();
      using (var command = new NpgsqlCommand("DROP INDEX index", model.Connection())){}
      model.Connection().Close();

      return list;
   }
}
```

## Псевдовипадкова генерація

```
public void RandomCamera()
   {
      Street street = new Street();
      InsertCamera(NextCId(), street.RandomStreet());
   }
   public void RandomCar()
   {
      int carId =  NextCarId();
      int ownerId = RandomCarOwner();
      Random random = new Random();
      int cameraId = random.Next(1, NextCId());
      connection.Open();
      try
      {
         using (var command = new NpgsqlCommand("INSERT INTO \"cars\" (id, carnumber, ownerid, cameraid, speed,
timeofregistrationc) VALUES (@id, @carnumber, @ownerid, @cameraid, floor(random() * (200-10+1) + 10)::int, @timeofregistrationc)",
connection))
         {
            command.Parameters.AddWithValue("id",carId);
            command.Parameters.AddWithValue("carnumber", RandomCarNumber()); //TODO check
            command.Parameters.AddWithValue("ownerid", ownerId);
            command.Parameters.AddWithValue("cameraid", cameraId);
            command.Parameters.AddWithValue("timeofregistrationc", new DateTime( random.Next(2000, 2021), random.Next(1,12),
random.Next(1,30), random.Next(0,24), random.Next(0,60), random.Next(0,60))); //???????
            command.ExecuteNonQuery();
         }
      }
      catch(NpgsqlException ex)
      {
```

```
        Console.WriteLine($"{ex.Message}");
    }
    connection.Close();
  }
  public int RandomCarOwner()
  {
    PIB pib = new PIB();
    int id = NextOId();
    InsertCarOwner(id, pib.RandomPIB());
    return id;
  }
```

## Приклад фільтрації

```
public List<string[]> TakeCars(int cameraId)
  {
    connection.Open();
    List<string[]> listOfCars = new List<string[]>();
    using (var command = new NpgsqlCommand("SELECT * FROM \"cars\" WHERE cameraid = @cameraid", connection))
    {
      var reader = command.ExecuteReader();

      while(reader.Read())
      {
        string[] car = new string[6];
        car[0] = reader[0].ToString(); //id
        car[1] = reader[1].ToString(); //carnumber
        car[2] = reader[2].ToString(); //ownerid
        car[3] = reader[3].ToString(); //cameraid
        car[4] = reader[4].ToString(); //speed
        car[5] = reader[5].ToString(); //timeofregistrationc
        listOfCars.Add(car);
      }
      reader.Close();
    }
    connection.Close();
    return listOfCars;
  }
```

## Резервне копіювання та відновлення

```
class Exute
{
  public void BackUp()
  {
    var process = new Process
    {
      StartInfo = new ProcessStartInfo
      {
        FileName = "C:/Users/Olha/Desktop/TrafficOrganization/code/backup.bat",
        Arguments = $"{DateTime.Now.ToFileTime()}",
        UseShellExecute = false,
        CreateNoWindow = true,
        Verb = "runas"
      }
    };

    process.Start();
    process.WaitForExit();
  }
  public void Restore(string fileName)
  {
    if (!File.Exists(fileName))
    {
```

```
            return;
        }
        if (!fileName.EndsWith(".sql"))
        {
            Console.WriteLine("You should choose sql-file");
            return;
        }

        var process = new Process
        {
            StartInfo = new ProcessStartInfo
            {
                FileName = "C:/Users/Olha/Desktop/TrafficOrganization/code/restore.bat",
                Arguments = "C:/Users/Olha/Desktop/TrafficOrganization/code/backup/" + fileName,
                UseShellExecute = false,
                CreateNoWindow = true,
            }
        };
        process.Start();
        process.WaitForExit();
    }
}
```

## Створення статистики

```
class Graphic
{
    Model model;
    public Graphic(Model model)
    {
        this.model = model;
    }
    private List<double> Percentages(Int64 allEntities, List<int> listOfEntities)
    {
        List<double> list = new List<double>();
        foreach (var item in listOfEntities)
        {
            double procent = item * 100 / allEntities;
            list.Add(procent);
        }
        return list;
    }
    public string SpeedPercentage()
    {
        var plt = new ScottPlot.Plot(600, 400);

        Int64 allEntities = model.NumberOfCars();
        List<int> listOfEntities = new List<int>(){model.NumberOfCarWithSpeed(0, 50), model.NumberOfCarWithSpeed(51, 70),
model.NumberOfCarWithSpeed(71, 300)};
        double[] values = new double[3];
        int i = 0;
        if(Percentages(allEntities, listOfEntities).Count == 3)
        {
            foreach (var item in Percentages(allEntities, listOfEntities))
            {
                values[i] = item;
                i++;
            }
        }
        string[] labels = { "0-50 km/h", "50-70 km/h", "from 70 km/h"};

        plt.PlotPie(values, labels, showPercentages: true, showValues: true, showLabels: true, explodedChart: true);
        plt.Legend();

        plt.Grid(false);
        plt.Frame(false);
```

```
    string file = "speed_chart.png";
    plt.SaveFig(file);
    return file;
}
private List<Int64[]> listCarsOwners()
{
    Int64[] listNumberOfCars = new Int64[model.TakeCarOwners().Count];
    int k = 0;
    foreach (var item in model.TakeCarOwners())
    {
        int ownerId = int.Parse(item[0]);
        listNumberOfCars[k] = model.TakeNumberOfCarsOwner(ownerId);
        k++;
    }
    List<Int64[]> list = new List<Int64[]>();
    for(int i = 0; i < listNumberOfCars.Length; i++)
    {
        bool exist = false;
        foreach (var item in list)
        {
            if(item[0] == listNumberOfCars[i])
            {
                exist = true;
            }
        }
        if(exist == false)
        {
            Int64[] c = new Int64[2];
            int count = 0;
            for(int j = i; j < listNumberOfCars.Length; j++)
            {
                if(listNumberOfCars[j] == listNumberOfCars[i])
                {
                    count++;
                }
            }
            c[0] = listNumberOfCars[i];
            c[1] = count;
            list.Add(c);
        }
    }
    return list;
}
public string NumberOfCars()
{
    //скільки власників мають певну кількість машин
    var plt = new ScottPlot.Plot(600, 400);

    List<Int64[]> list = listCarsOwners();

    int pointCount = list.Count;
    double[] ys = new double[pointCount];
    int k = 0;
    foreach (var item in list)
    {
        ys[k] = item[1];
        k++;
    }
    double[] xs = new double[pointCount];
    string[] labels = new string[pointCount];
    k = 0;
    foreach (var item in list)
    {
        xs[k] = item[0];
        labels[k] = item[0].ToString() + "car(s)";
        k++;
    }

    plt.PlotBar(xs, ys, horizontal: true);
    plt.YTicks(xs, labels);
```

```
    string file = "numbers_of_car_owners.png";
    plt.SaveFig(file);
    return file;
  }
  public string CarsOnStreet()
  {
    // відсоток машин на вулицях
    int points = model.AllStreets().Count;

    var plt = new ScottPlot.Plot(600, 400);

    double[] values = new double[points];
    string[] labels = new string[points];
    int i = 0;
    foreach (var item in model.AllStreets())
    {
      int streetId = int.Parse(item[0]);
      values[i] = model.TakeCars(streetId).Count;
      labels[i] = item[1];
      i++;
    }

    plt.PlotPie(values, labels , showPercentages: true, showValues: true, showLabels: true);
    plt.Legend();

    plt.Grid(false);
    plt.Frame(false);
    string file = "street_statistic.png";
    plt.SaveFig(file);
    return file;
  }
}
```

## Приклад деяких методів view

```
class View
{
  public View()
  {
    Console.WriteLine("\tWelcome!");
  }
  public string Command()
  {
    Console.Write("\nEnter command ('output', 'dataBase', 'statistics', 'exit'): ");
    string answer = Console.ReadLine();
    if(answer == "output")
    {
      Console.Write("\nEnter command ('cars', 'index'): ");
      string a = Console.ReadLine();
      if(a == "cars")
      {
        return answer + " " + a;
      }
      else
      {
        return answer + " " + a;
      }
    }
    else if(answer == "dataBase")
    {
      // TODO копіювання та резервування
      Console.Write("\nEnter command ('insert', 'update', 'delete', 'random', 'backup', 'reservation'): ");
      answer = Console.ReadLine();
      if(answer == "insert" || answer == "update" || answer == "delete" || answer == "random")
      {
        Console.Write("\nEnter entity ('camera', 'car', 'carowner'): ");
```

```
string entity = Console.ReadLine();
if(entity == "camera")
{
   if(answer == "insert")
   {
      Console.Write("\nEnter location: ");
      string location = Console.ReadLine();
      return answer + " " + entity + " " + location;
   }
   else if(answer == "update")
   {
      Console.Write("\nEnter location: ");
      string location = Console.ReadLine();
      Console.Write("\nEnter camera id: ");
      string id = Console.ReadLine();
      return answer + " " + entity + " " + location+ " " + id;
   }
   else if(answer == "delete")
   {
      Console.Write("\nEnter camera id: ");
      string id = Console.ReadLine();
      return answer + " " + entity + " " + id;
   }
   else
   {
      Console.Write("\nHow many cameras? ");
      string num = Console.ReadLine();
      return answer + " " + entity + " " + num;;
   }
}
else if(entity == "car")
{
   if(answer == "insert")
   {
      Console.Write("\nEnter 'car number' 'owner id' 'camera id' 'speed': ");
      string a = Console.ReadLine();
      string[] s = a.Split(" ");
      return answer + " " + entity + " " + s[0] + " " + s[1] + " " + s[2] + " " + s[3];
   }
   else if(answer == "update")
   {
      Console.Write("\nEnter 'car number' 'owner id' 'camera id' 'speed': ");
      string a = Console.ReadLine();
      string[] s = a.Split(" ");
      Console.Write("\nEnter car id: ");
      string id = Console.ReadLine();
      return answer + " " + entity + " " + s[0] + " " + s[1] + " " + s[2] + " " + s[3] + " " + id;
   }
   else if(answer == "delete")
   {
      Console.Write("\nEnter car id: ");
      string id = Console.ReadLine();
      return answer + " " + entity + " " + id;
   }
   else
   {
      Console.Write("\nHow many cars? ");
      string num = Console.ReadLine();
      return answer + " " + entity + " " + num;
   }
}
else if(entity == "carowner")
{
   if(answer == "insert")
   {
      Console.Write("\nEnter fullname: ");
      string fullname = Console.ReadLine();
      return answer + " " + entity + " " + fullname;
   }
   else if(answer == "update")
```

```
            {
               Console.Write("\nEnter fullname: ");
               string fullname = Console.ReadLine();
               Console.Write("\nEnter car owner id: ");
               string id = Console.ReadLine();
               return answer + " " + entity + " " + fullname + " " + id;
            }
            else if(answer == "delete")
            {
               Console.Write("\nEnter car owner id: ");
               string id = Console.ReadLine();
               return answer + " " + entity + " " + id;
            }
            else
            {
               Console.Write("\nHow many owners? ");
               string num = Console.ReadLine();
               return answer + " " + entity + " " + num;
            }
         }
         else
         {
            return "";
         }
      }
      else if(answer == "backup")
      {
         return answer;
      }
      else if(answer == "reservation")
      {
         Console.Write("Enter file name: ");
         string a = Console.ReadLine();
         return answer + " " + a;
      }
      else
      {
         return "";
      }
   }
   else if(answer == "statistics")
   {
      Console.Write("\nEnter command ('speed', 'cars-owner', 'streets-cars'): ");
      answer = Console.ReadLine();
      if(answer == "speed" || answer == "cars-owner" || answer == "streets-cars")
      {
         return "graphic " + answer;
      }
      else
      {
         return "";
      }
   }
   else if(answer == "exit")
   {
      return answer;
   }
   else
   {
      return "";
   }
}
…
```