**Коммунальное учебное заведение**

**«Днепровский лицей информационных технологий**

**при Днепровском национальном университете**

**имени Олеся Гончара»**

**Выпускная работа**

**на тему:**

**Сбор данных средствами «Telegram» и их визуализация, средствами C#**

**Исполнитель:**

**лицеист 11-Б-II класса**

**Волошин Никита**

**Руководители работы:**

**Лавренюк И.В. \_\_\_\_\_\_\_\_\_\_\_\_\_**

Содержание

[ВВЕДЕНИЕ 2](#__RefHeading___Toc2119_3666729055)

[Тема 3](#__RefHeading___Toc2121_3666729055)

[Цель 3](#__RefHeading___Toc2123_3666729055)

[Задачи 3](#__RefHeading___Toc2125_3666729055)

[РАЗДЕЛ I ТЕОРИЯ 4](#__RefHeading___Toc2127_3666729055)

[Обоснование использования библиотек и платформ 4](#__RefHeading___Toc2129_3666729055)

[C# 4](#__RefHeading___Toc2131_3666729055)

[Windows Forms ( .net Framework 4.8) 4](#__RefHeading___Toc2133_3666729055)

[Telegram 4](#__RefHeading___Toc2135_3666729055)

[Telegram API 4](#__RefHeading___Toc2137_3666729055)

[ASP . net Core Web – API 4](#__RefHeading___Toc2139_3666729055)

[БД (MSSQL) 5](#__RefHeading___Toc2141_3666729055)

[Entity Framework 6 ( для Forms) и пакет Entity Framework Core v8 ( для WebAPI ) 5](#__RefHeading___Toc2143_3666729055)

[Live Chart 5](#__RefHeading___Toc2145_3666729055)

[Алгоритмы 5](#__RefHeading___Toc2147_3666729055)

[БД 5](#__RefHeading___Toc2149_3666729055)

[Telegram бот 6](#__RefHeading___Toc2151_3666729055)

[Клиент для просмотра 7](#__RefHeading___Toc2153_3666729055)

[Раздел II Возможности работы 8](#__RefHeading___Toc2155_3666729055)

[Структурная схема 8](#__RefHeading___Toc2157_3666729055)

[Внешний вид и примеры взаимодействия 8](#__RefHeading___Toc2159_3666729055)

[Применение 11](#__RefHeading___Toc2161_3666729055)

[Раздел ІІІ Выводы и спецификации 13](#__RefHeading___Toc2163_3666729055)

[Системные требования и комплектация 13](#__RefHeading___Toc2165_3666729055)

[Telegram бот 13](#__RefHeading___Toc2167_3666729055)

[Клиент для просмотра 13](#__RefHeading___Toc2169_3666729055)

[Использованные средства 14](#__RefHeading___Toc2171_3666729055)

ВВЕДЕНИЕ

## Тема

Сбор и анализ данных - это основные задачи для информатики, как науки, в этой работе реализован полный цикл работы с информацией от ввода данных пользователями через мессенджер до представления данных в виде наглядных графиков в десктопном приложении.

Сам проект представляет собой систему для анализа показателей работы городского транспорта. С помощью бота в Telegram водитель или контроллер будет указывать данные по передвижению своего транспорта и динамике пассажиров.

Менеджер с помощью приложения на базе Windows Forms сможет быстро анализировать актуальную информацию и своевременно оптимизировать работу транспорта.

## Цель

Создание платформы для сбора и визуализации данных движения и загруженности городского транспорта.

Исследование Telegram, как средства для сбора данных, и последующее создание бота на базе этого мессенджера. Создание механизма передачи собранных данных и приложения для их визуализации.

## Задачи

* Исследование Telegram API для языка C#
* Исследование вариантов передачи данных
* Создание веб-прилоджения на платформе .net Core с архитектурой WebApi, для взаимодействия с Telegram, а через него с пользователем.
* Создание и настройка базы данных на базе MSSQL и реализация ее поддержки в компонентах системы.
* Создание клиента для просмотра и редактирования данных на базе Windows Forms.
* Исследование библиотек для создания инфографик и последующее их примение.
* Объедение всех компонентов в рабочую систему

# РАЗДЕЛ I ТЕОРИЯ

## Обоснование использования библиотек и платформ

### C#

Основным критерием при выборе языка была его гибкость. Язык должен с одинаковой легкостью справляться, как с сетевым прилодением для сбора данных, так и с отрисовкой графиков на ПК. С# был идеальным выбором, Microsoft создала этот язык максимально многогранным, а платформа .net сама позаботится о совместимости со всеми устройствами.

### Windows Forms ( .net Framework 4.8)

Большой арсенал библиотек к этой платформе, обеспечит широкий выбор вариантов для визуализации, также именно эта платформа изучается в нашем учебном заведении, поэтому в случае проблем я с легкостью получу квалифицированную помощь. Последним, решающим фактором стало, то что в отличии от WPF данная платформа не требует знаний XAML, а реализуется полностью на чистом C#.

### Telegram

Удобный и современный мессенджер, который отвечает стандартам безопасности и просто удобен. Это делает эту платформу очень привлекательной, ведь пользователю не нужно даже загружать приложение или использовать сайт – все будет в его любимом мессенджере.

### Telegram API

Хоть эта библиотека и не разработана представителями Telegram именно ее рекомендуют использовать в официальном руководстве по созданию ботов.

### ASP .net Core Web-API

Данный тип проекта используется в официальном руководстве Telegram API, также это обеспечит кроссплатформенность, что чрезвычайно важно для серверного приложения. В заключении, можно указать, что именно .net Core явлется мировым стандартом в сфере веб-приложений на базе языка C#

БД (MSSQL)

База данных в этом проекте жизненно необходима, потому что данные нужно куда-то собирать, а потом с откуда-то брать, это можно было бы реализовать с помощью запросов к боту, однако тогдна на эту часть будет чрезвычайно большая нагрузка, что недопустимо для стабильной системы, не говоря уже о том, что в случае сбоя все данные рискуют быть потеряны. MSSQL же была выбрана для максимального устранения любых проблем с совместимостью.

### Entity Framework 6 ( для Forms) и пакет Entity Framework Core v8 ( для WebAPI )

После окончательного решения по использованию базы данных, пояявился вопрос как с ней взаимодействовать здесь я, не колеблясь, выбрал Entity Framework , эта платформа вытеснила всех конкурентов являясь наиболее быстрой, стабильной и отказоустойчивой.

### Live Chart

Самая красивая библиотека, на мой взгляд, для создания инфографик с богатым пространством для творчества и многообразием функций.

## Алгоритмы

### БД

Создание БД с помощью Entity Framework – чрезвычайно интересный процесс. Во-первых, существует несколько подходов к использованию БД.

* Code-First подход заключающийся в том, что мы создаем таблицу БД с использованием принципов ООП из С# то есть, мы создаем класс, наполнение которого и будут полями таблицы, а набор классов и будет являться своего рода таблицей. Этот подход я применил при разработке базы данных в рамках проекта Web -API.
* Database-First подход состоит в создании ООП модели после подключения к БД, своего рода, реверс-инжиниринг. Этот подход был использован мной при разработке клиента для просмотра данных, потому что база данных должна быть одной для всех проектов, а этот подход автоматически построит модели для успешного подключения.
* Существуют также другие подходы для создания но они представляют собой либо комбинацию предыдущих, либо использование их же но например с интерфейсом как в MS Access. Их я также частично использовал, но не считаю очень важными для упоминания, ведь существенного влияния на проект они не оказали.

Существуют несколько методов загрузки данных из базы данных с помощью Entity Framework .

* Explicit Loading состоит в составление явного запроса на загрузку конкретного свойства или коллекции у заданного объекта.
* Eager Loading заключается в использовании метода .Include , в котором мы с помощью лямбда выражений запрашиваем какое-лиюо свойство или коллекцию для **коллекции объектов** .
* Lazy Loading не очень распространенный подход из-за того, что он заключается в загрузке только того, что мы используем при использовании программы, такой вид загрузки не очень хорошо влияет на скорость работы приложения, ведь по сути он скачивает все данные которые **теоретически** программа может запросит при выполнении.

Все эти методы хороши по своему, но надо понимать, что среди них нет ни одного плохого или хорошего именно это я доказал собственноручно в своей квалификационной работе, где я использую все три вида . Все эти методы имеют разную специализацию, если мы должны загрузить малые объемы данных, то мы используем Explicit loading , если коллекцию данных, то Eager Loading , относительно Lazy Loading все несколько сложнее его всегда можно заменить комбинацией Eager и Explicit Loading , но в десктопных программах где не столь важно использовать минимум оперативной памяти этот подход имеет свою нишу, так как это существенно упрощает разработку.

### Telegram бот

Основная сложность написанием алгоритмов для бота состоит в том, что у него отсутствует оперативная память. Конечно не буквально, но каждое новое сообщение в Telegram API рассматривается как первое и это значительно усложняет создание жизнеспособной системы, где пользователь не сможет случайно воспользоваться программой не так как было задумано, а злоумышленник не сможет удалить или просмотреть конфиденциальные данные.

### Клиент для просмотра

Относительно клиента, основная сложность, заключается в правильном запросе данных для их представления.

Здесь я частично применяли как LINQ так и прямые запросы. LINQ использовался только когда без него были бы значительные падения в быстродействии, или же когда операция описывается LINQ в пару строк, а в циклах все не столь триваильно. В остальном использовались обычные циклы, ведь они гораздо более читабельны, а также, не считая специфических операций, работают ощутимо быстрее.

Следует отметить, что вся статистика показывается только за последние 30 дней, по моему мнению, это оптимальное время для того, чтобы была полная картина анализа, а объемы данных не слишком большие для окна на обычном персональном компьютере, к тому же если использовать большие периоды места на форме недостаточно для комфортного просмотра.

Также были созданы формы для изменения и добавления маршрутов или водителей благодаря тому, что бот *не* имеет оперативной памяти, все изменения, которые делает администратор, отображаются почти моментально.

Раздел II Возможности работы

## Структурная схема

Telegram бот

Клиент для просмотра данных

## Внешний вид и примеры взаимодействия

Telegram бот

Зображення, що містить текст, монітор, комп’ютер, електроніка

Автоматично згенерований описЛюбая неправильная команда или /usage вызывает вспомогательное сообщение

Рисунок 1 Пример /usage

Команда /routes вызывает меню со всеми маршрутами на которых показана начальная и конечная станция. Клик по кнопке выдает Id маршрута и список всех станций

Зображення, що містить текст, монітор, комп’ютер, знімок екрана

Автоматично згенерований опис

Рисунок 2Пример /routes

Команда /start – единственная команда, которая принимает аргументы, а именно Id маршрута нужно начать и стартовое количество пассажиров.

Зображення, що містить текст, монітор, знімок екрана, комп’ютер

Автоматично згенерований опис

Рисунок 3 Пример /start

/continue создает клавиатуру с быстрым вводом чисел для обозначения изменения количества пассажиров.

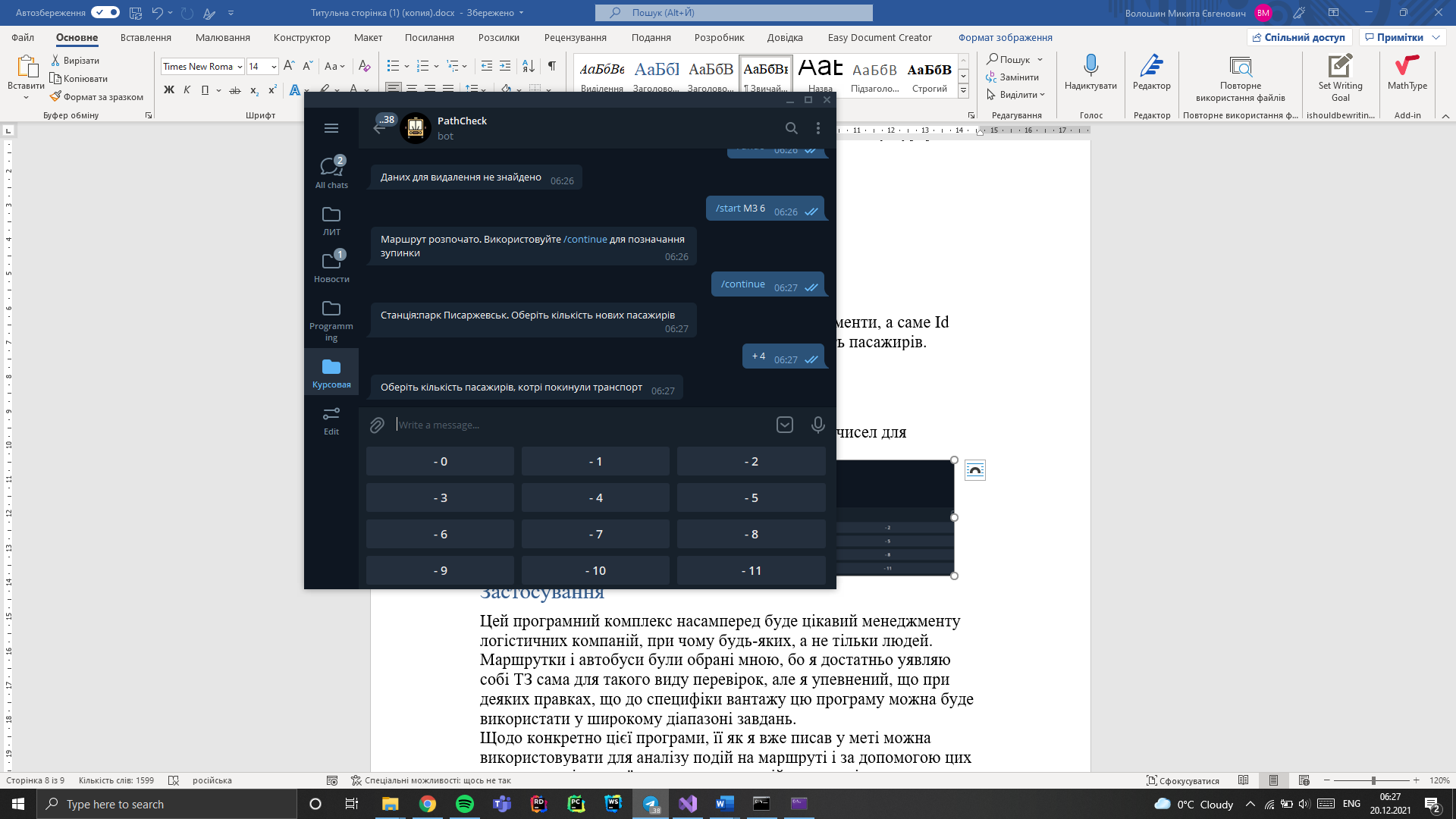


Рисунок 4Пример / continue

Команда /undo удаляет последние введенные данные

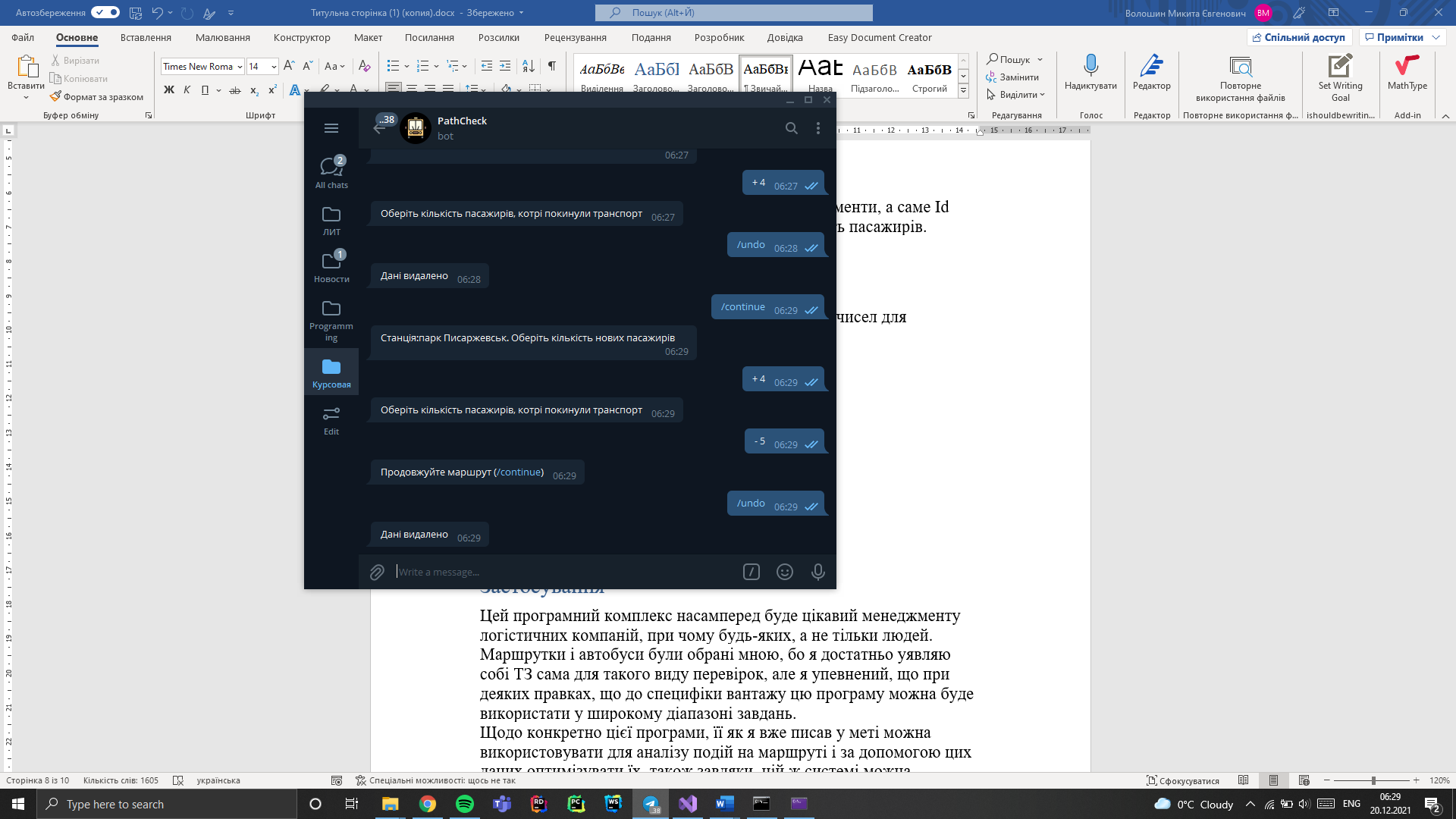


Рисунок 5 Пример /undo

/end удаляет начатый, но незавершенный маршрут и все данные по нему.

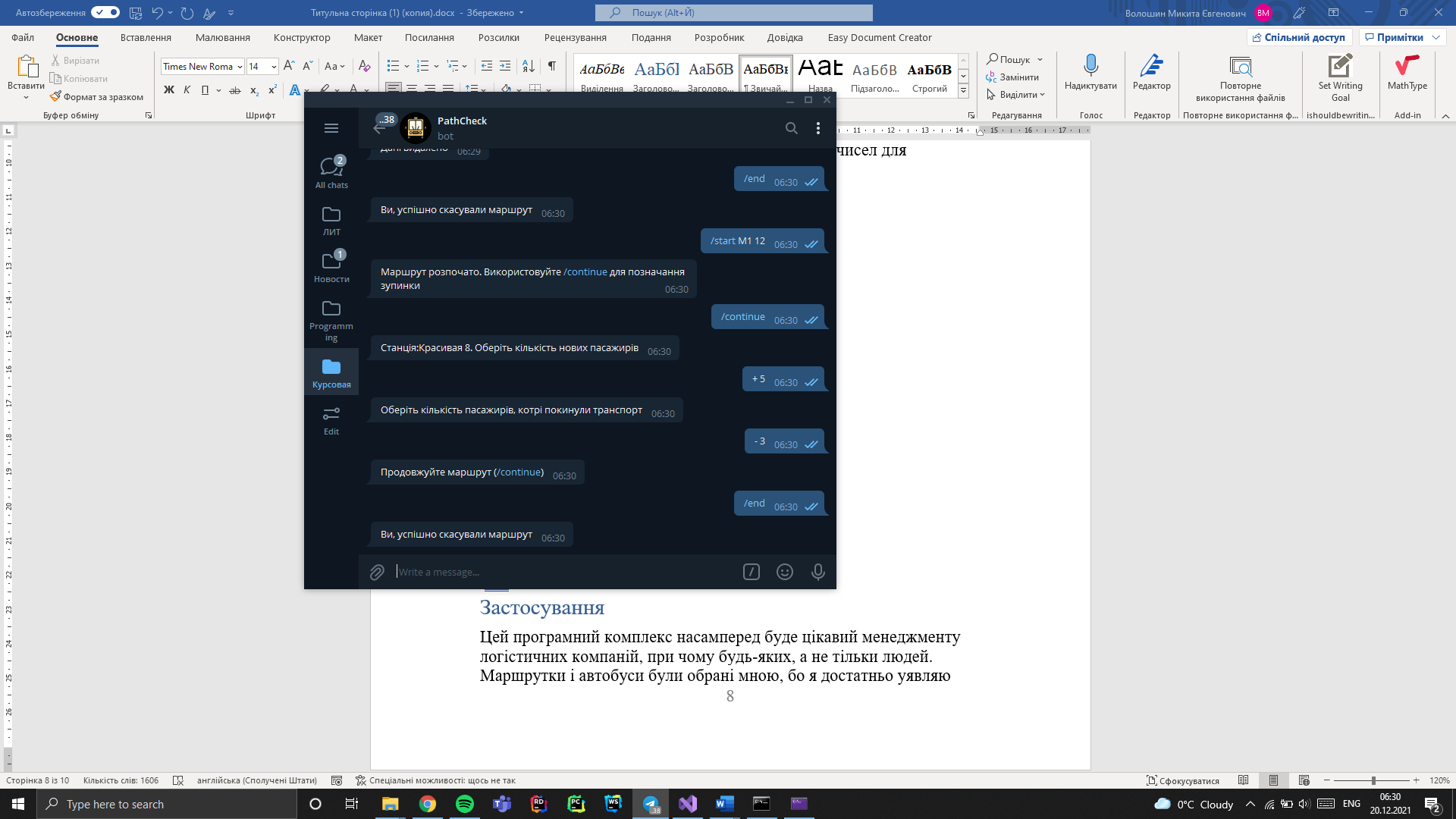


Рисунок 6 Пример / end

Клиент для просмотра данных

После начальной страницы мы попадаем в главное меню. Где мы можем выбрать один из четырех пунктов

* Первый показывает статистику по маршрутам, сначала общие данные о их количестве на диаграмме. После этого можно выбрать тип маршрута и создать график со средними значениями на каждом участке маршрута, в следующем меню можно выбрать детальный анализ выбранного маршрута

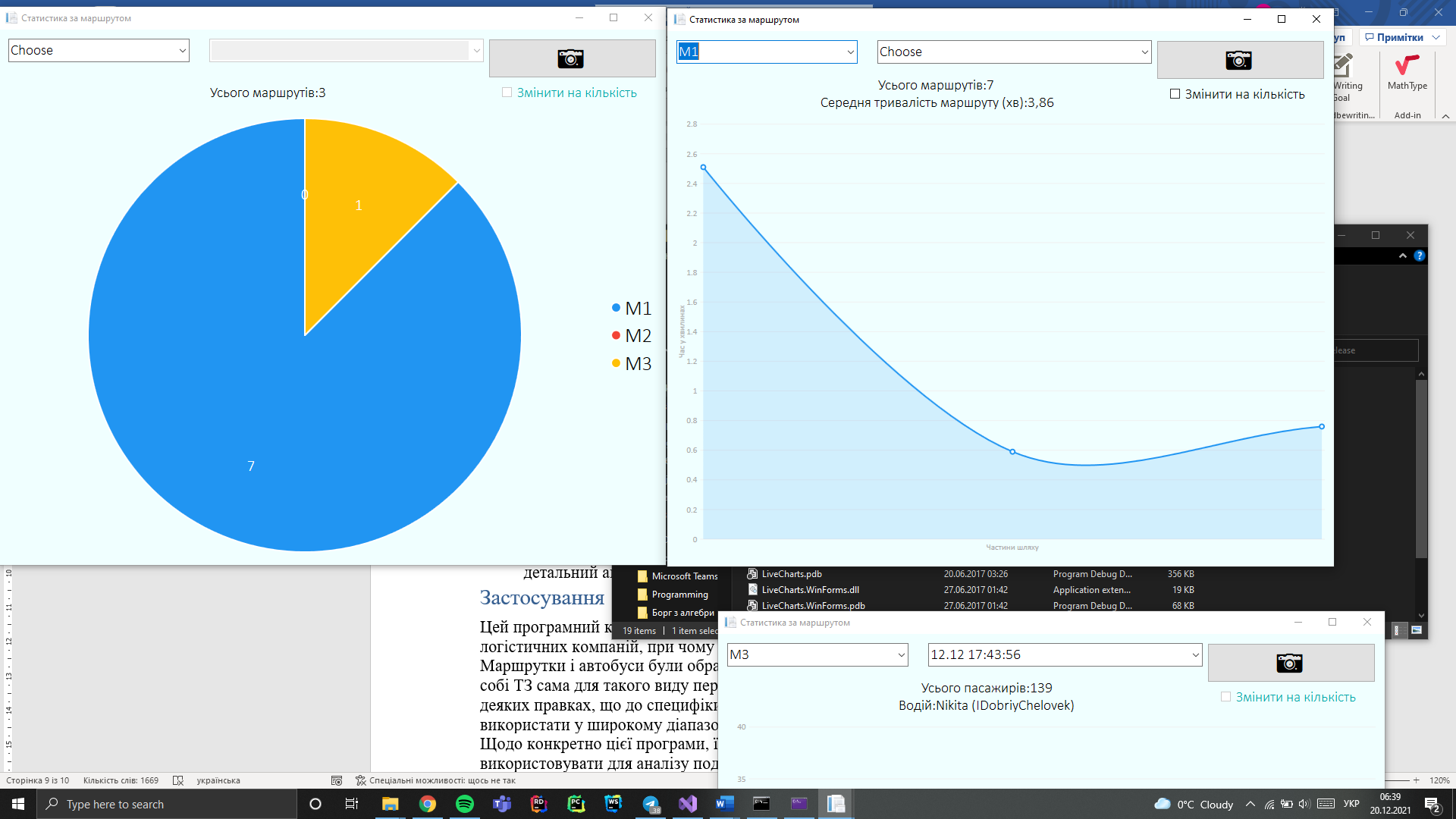


Рисунок 7Пример первых двух инфографик

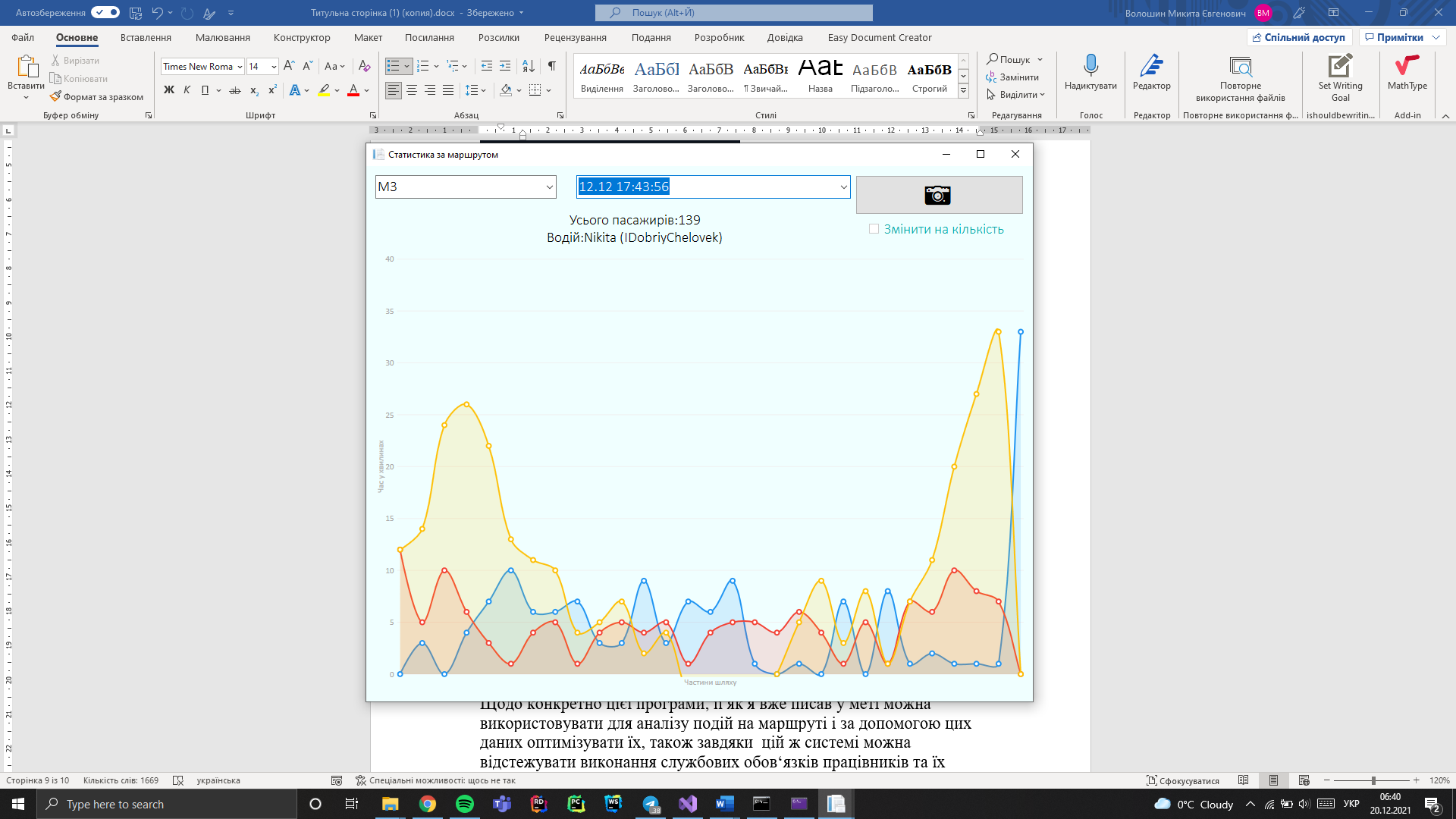


Рисунок 8Пример третьей инфографики

* Во втором пункте после выбора водителя мы увидим диаграмму и график со статистикой его выездов за последние 30 дней

Зображення, що містить стіл

Автоматично згенерований опис

Рисунок 9 Инфорграфики из второго окна

* Третий пункт позволяет создавать и удалять маршруты. Все изменения происходят только по нажатию кнопки «Сохранить».

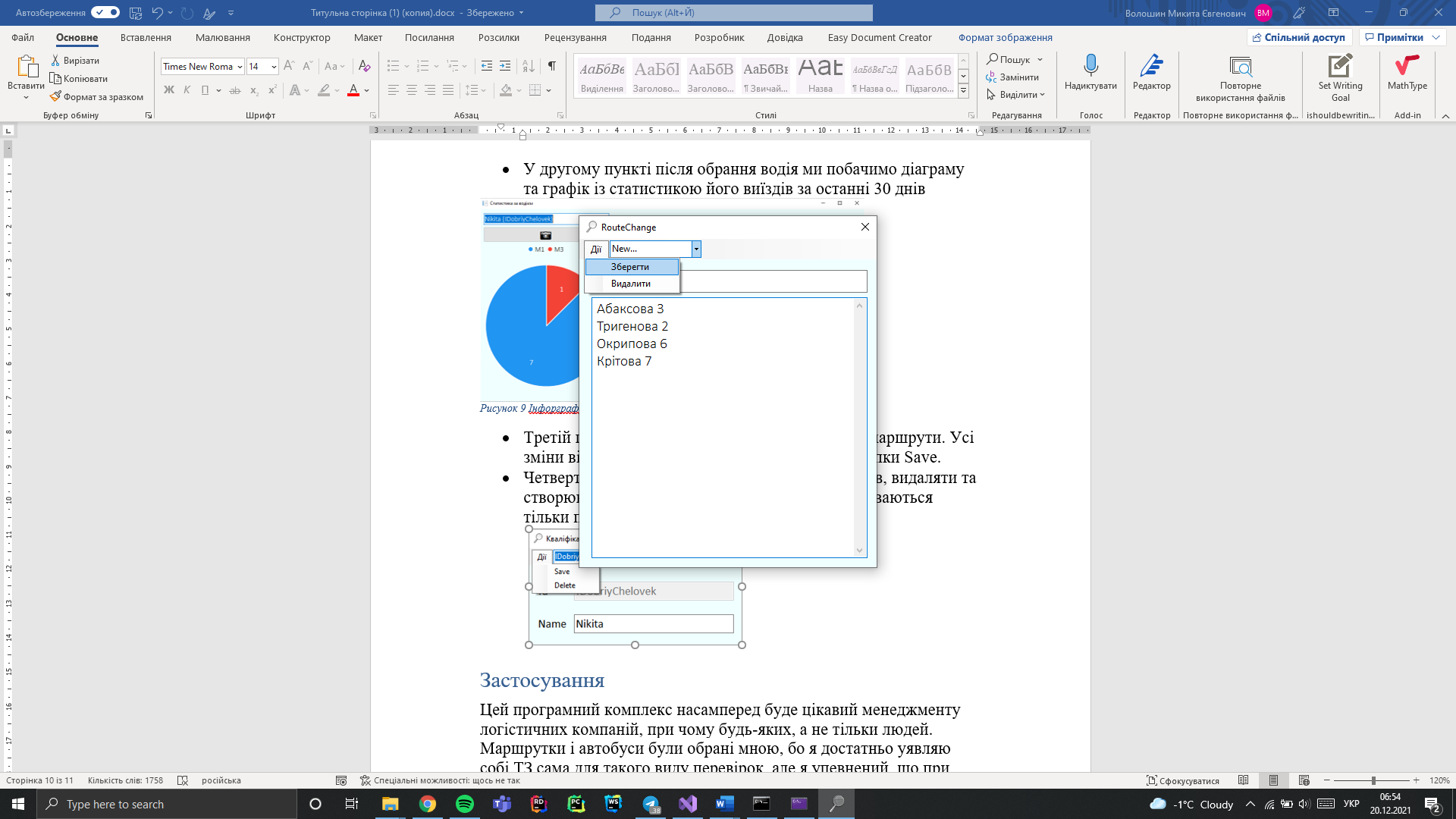


Рисунок 10Редактор маршрутов

* Четвертый позволяет редактировать имя пользователя, удалять и создавать новых пользователей. Все изменения происходят только после нажатия кнопки « Сохранить».

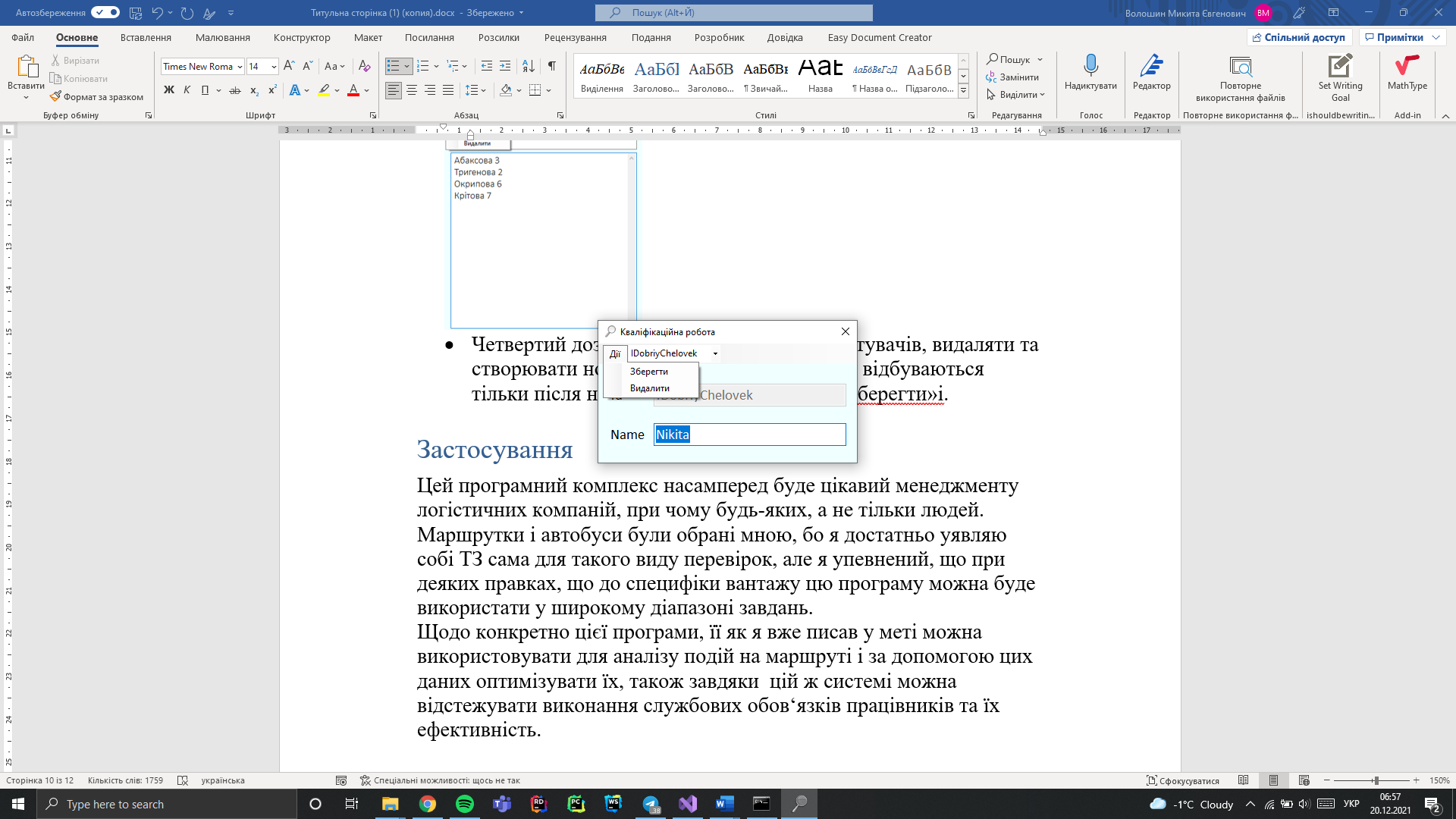


Рисунок 11Редактор водителей

## Применение

Этот программный комплекс в первую очередь будет интересен менеджменту логистических компаний, причем любых, а не только специализирующихся на перевозке людей. Муниципальный транспорт был выбран мной, потому что я достаточно представляю себе их устройство, однако я уверен, что при небольших правках, эту программу можно будет использовать в гораздо более широком диапазоне задач .

Относительно конкретно этой системы, ее как указано выше, ее можно использовать для анализа событий на маршруте и с помощью этих данных оптимизировать их, также благодаря этой системе можно отслеживать выполнение служебных обязанностей работников и их эффективность.

Раздел ІІІ Выводы и спецификации

## Системные требования и комплектация

### Telegram бот

Во-первых, следует отметить, что запустить бота можно двумя путями. Первый на localhost для этого нужна программа хостер. В качестве примера я использовал ngrok, также я написал использовать конфигурационный файл ngrok.bat, что автоматически запустит сервер со всеми необходимыми параметрами *.* После этого в файле appsettings нужно изменить существующий адрес на отображающийся в программе.

Второй же предполагает публикацию бота на сервере, но перед этим следует проверить, что сайт поддерживает https , а хостинг поддерживает .net 6, после этого нужно указать доменное имя в файле appsettings и опубликовать бота с использованием встроенного инструментария visual studio .

* Подключение к интернету
* Свободное пространство 200 MB

**Комплектация** :

* Exe файл с сопутствующими DLL
* Appsettings.json
* ngrok.exe
* ngrok.bat
* web.config
* TgBot.pdb
* TgBot.XML

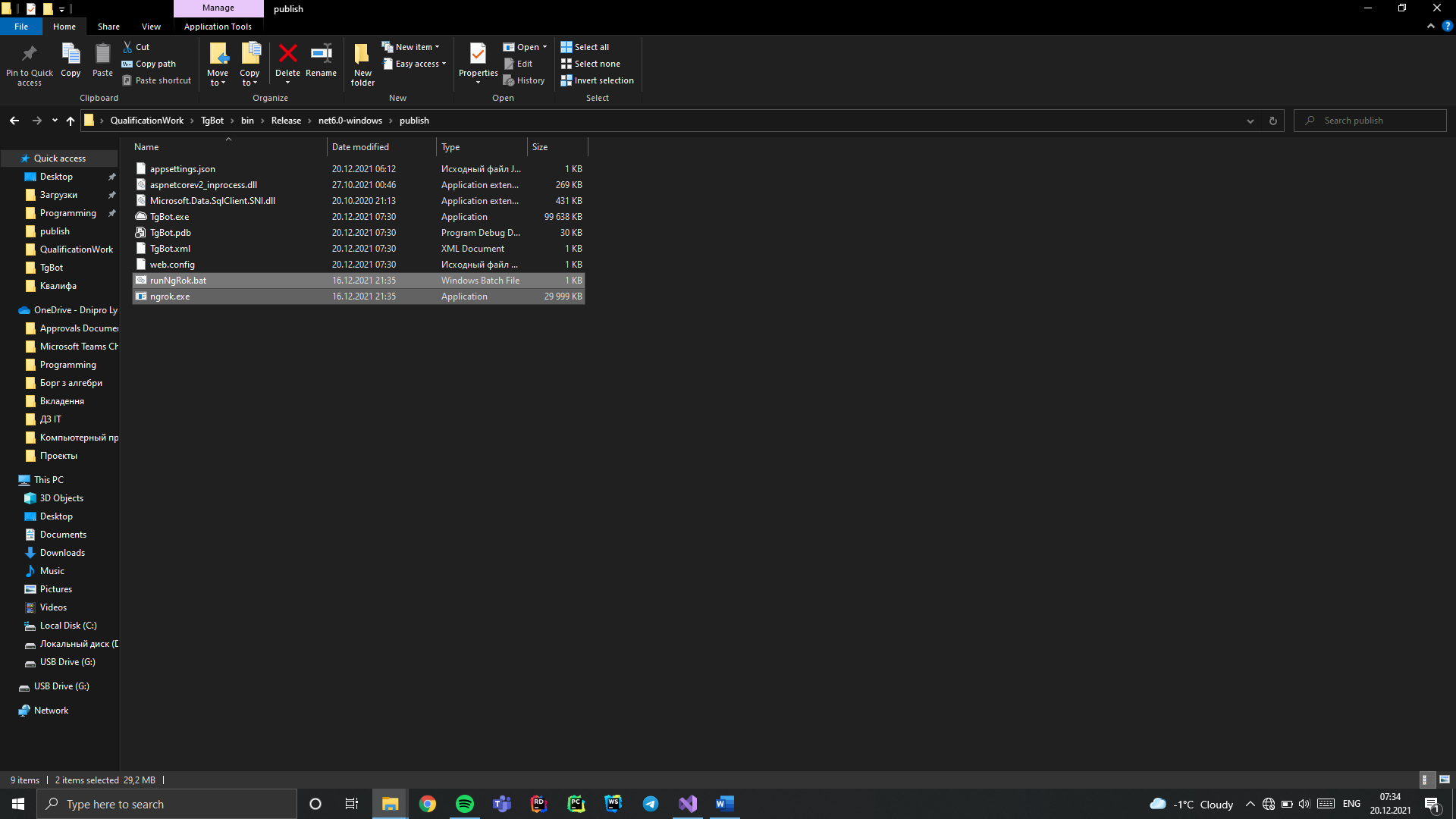


Рисунок 12 Полный список

### Клиент для просмотра

Свободное пространство: 200MB

**Комплектация:**

* Setup.exe
* Папка Application Files / DataClient \_1\_2\_1\_3 с необходимыми для инсталляции файлами

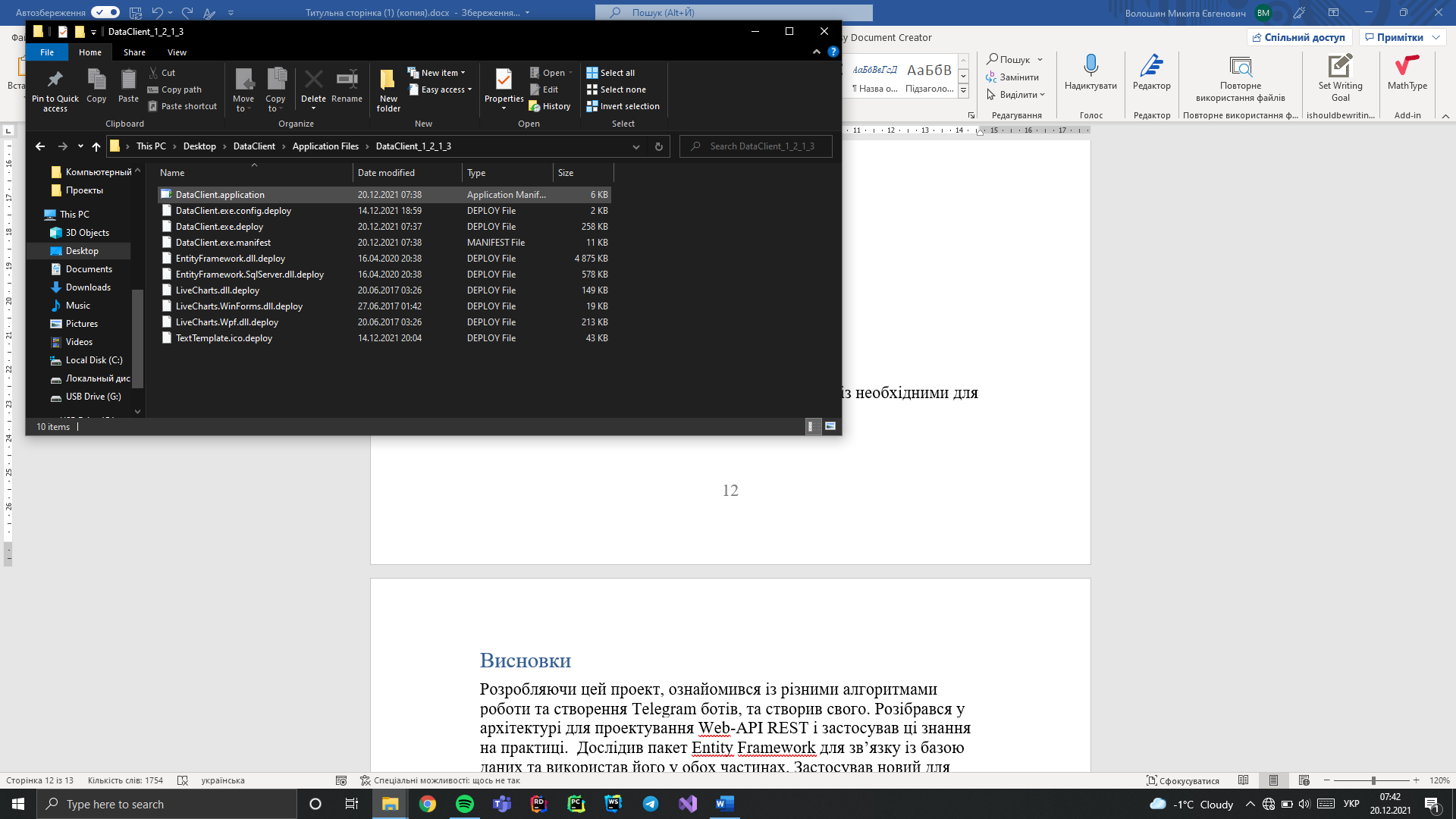


Рисунок 13Необходимые для установки клиенту данные файлы

## Использованные средства

* Visual Studio 2021
* Rider
* Access
* Azure

Выводы

Разрабатывая этот проект, ознакомился с разными алгоритмами работы и создания Telegram ботов и смог разработать своего. Разобрался в архитектуре для проектирования Web -API REST и применил эти знания на практике. Исследовал пакет Entity Framework для связи с базой данных и использовал его в обеих частях. Применил новый библиотеку для создания интерактивных графиков в десктопном приложении.

Источники

1. <https://docs.microsoft.com/ru-ru/dotnet/csharp/>
2. <https://www.youtube.com/watch?v=OTKJfGbgfxM&ab_channel=AtlassDev>
3. [https://www.youtube.com/watch?v=yubASf8mNZw&ab\_channel=Skillbox%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC% D0%B8%D1%80%D0%B0%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5](https://www.youtube.com/watch?v=yubASf8mNZw&ab_channel=SkillboxПрограммирование)
4. <https://habr.com/ru/company/alfa/blog/343846/>
5. [https://ru.stackoverflow.com/questions/784321/telegram-bot-c-%D1%85%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3](https://ru.stackoverflow.com/questions/784321/telegram-bot-c-хостинг)
6. <https://metanit.com/sharp/entityframeworkcore/>
7. <https://metanit.com/sharp/entityframework/>
8. <https://lvcharts.net/App/examples/v1/WinForms/start>
9. <https://github.com/TelegramBots/Telegram.Bot>
10. <https://core.telegram.org/bots>
11. <https://telegrambots.github.io/book/>
12. <https://github.com/TelegramBots/telegram.bot.examples>