

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА
на тему
**«Программное обеспечение прогнозирования и
анализа состояния сети водоснабжения»**

Студент

_____ Желудков Данила Иванович _____
(подпись, дата) (ФИО полностью)

Руководитель

_____ Макарычев П.П. _____
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

_____ Такташкин Д.В. _____
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

_____ Козлов А.Ю. _____
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

_____ Попова Н.А. _____
(подпись) (фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«__» _____ 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Желудков Данила Иванович гр. 19ВП2 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы «Программное обеспечение прогнозирования и анализа состояния сети водоснабжения»

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы Макарычев П. П.

4. Задание на работу (назначение разработки, исходные данные и т.п.) _____

Среда разработки: Windows 11, язык программирования C# 8 на платформе .NET 4.8, платформа для создания графического интерфейса WindowsForms, ORM-система Entity Framework6.0.

Среда выполнения: Microsoft Visual Studio 2019

Разработать программное обеспечение прогнозирования и анализа состояния сети водоснабжения

Основные функции программного обеспечения:

- учёт и анализ автоматов продажи воды;
- учет поставщиков и парка их автомобилей;
- оптимизация маршрута транспортировки воды;
- ведение списка услуг;
- прогнозирование расхода водоматов.

Технология разработки: ООП, MVP

Языки программирования: C# 8.0

5. Перечень подлежащих разработке вопросов _____

1) Анализ предметной области
2) Анализ функциональных требований
3) Планирование разработки и оценка бюджета
4) Проектирование программного обеспечения
5) Реализация программного обеспечения
6) Контроль качества
7) Выполнение пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	12.04.2023 – 16.04.2023	
2	Проектирование программных средств	20%	17.04.2023 – 26.04.2023	
3	Реализация программных средств	30%	27.04.2023 – 11.05.2023	
4	Тестирование программных средств	20%	12.05.2023 – 21.05.2023	
5	Оформление пояснительной записки	20%	22.05.2023 – 01.06.2023	

Дата выдачи задания «22» декабря 2022 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____

Работу к защите допустить
Декан ФВТ

(подпись, дата)

Л.Р. Фионова

Реферат

Пояснительная записка содержит 199 листов, 55 рисунков, 21 таблица, 13 использованных источников, 4 приложения.

АНАЛИЗ, ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА, ТЕСТИРОВАНИЕ, НАСТОЛЬНОЕ ПРИЛОЖЕНИЕ, СЕТЬ ВОДОСНАБЖЕНИЯ, УЧЁТ ВОДОМАТОВ, РЕГРЕССИОННЫЙ АНАЛИЗ, ЗАДАЧА КОММИВОЯЖЁРА,

Объект разработки – программное обеспечение прогнозирования и анализа состояния сети водоснабжения.

Цель работы – изучить предметную область и провести её анализ, выделить процессы, разработать и протестировать программный продукт.

Технология разработки: .NET 4.8, C# 8.0, Entity Framework 6.0, GMapNet, Itinero, OrTools, Microsoft SQL Server 2019.

В результате выполнения работы было спроектировано и создано программное обеспечение, предназначенное для прогнозирования и анализа состояния сети водоснабжения.

					ПГУ 09.03.04 – 8ВР192.08 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	«Программное обеспечение прогнозирования и анализа состояния сети водоснабжения» Пояснительная записка.	Лит.	Лист	Листов
Разраб.		Желудков Д.И.						
Провер.		Макарычев П.П.					4	146
Н. Контр.		Такташкин Д.В.				Группа 19ВП2		
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Разработка и анализ требований для ПО прогнозирования и анализа состояния сети водоснабжения.....	9
1.1 Анализ предметной области и функциональных требований	9
1.1.1 Анализ предметной области	9
1.1.2 Анализ функциональных требований.....	10
1.2 Анализ программного обеспечения аналогичного назначения.....	14
1.2.1 ПО WaterGEMS.....	15
1.2.2 ПО EPANET	16
1.2.3 ПО H2OMAP Water	18
1.2.4 Сравнение аналогов	20
1.3 Регрессионный анализ	21
1.3.1 Модель авторегрессии	22
1.4 Задача коммивояжёра	24
2 Проектирование ПО.....	27
2.1 Проектирование архитектуры ПО	27
2.2 Проектирование базы данных	34
2.3 Выбор программно-аппаратных средств.....	36
2.3.1 Выбор языка программирования	36
2.3.2 Выбор необходимых модулей.....	37
2.3.3 Math.Numerics	38
2.3.4 Google.OrTools	38
2.3.5 GMap.NET	39
2.3.6 Itinero	40
2.4 Проектирование интерфейса	42
2.5 Планирование и расчёт бюджета, разрабатываемого ПО	43
2.5.1 Календарное планирование и список ресурсов.....	43
2.5.2 Расчёт бюджета.....	44

3	Реализация ПО	50
3.1	Кодирование ПО	51
3.2	Руководство для пользователя	57
4	Тестирование ПО	68
4.1	Тестирование пользовательского интерфейса	68
4.1.1	Тестирование по стратегии черного ящика.....	68
4.1.2	Метод граничных условий	71
4.2	Тестирование серверной части	72
4.2.1	Модульное тестирование	72
4.3	Метрики кода	75
4.3.1	Метрики размера программы	76
4.3.2	Метрики Холстеда	80
	Заключение	89
	Список использованных источников.....	91
	Приложения	93
	Приложение А	93
	Приложение Б	183
	Приложение В.....	190
	Приложение Г	198

Введение

Тема «Программное обеспечение прогнозирования и анализа состояния сети водоснабжения» актуальна в свете проблем, с которыми сталкиваются города и регионы по всему миру в сфере водоснабжения. Некоторые из этих проблем включают в себя неэффективное использование ресурсов воды, устаревшую инфраструктуру и неисправности оборудования, риски некачественного водоснабжения и необходимость предотвращения аварийных ситуаций. Поэтому, применение программного обеспечения для прогнозирования и анализа состояния сети водоснабжения позволяет экономить ресурсы, повышать эффективность работы сети и предотвращать аварии. Одним из главных преимуществ программного обеспечения для прогнозирования и анализа состояния сети водоснабжения является возможность быстрого и точного анализа данных, полученных с датчиков и другого оборудования. Это может помочь операторам водоснабжения быстро реагировать на изменения в потреблении воды и других факторах, связанных с работой сети, и принимать своевременные меры для предотвращения возможных аварийных ситуаций.

Цель данной работы – разработать программное обеспечение (ПО), обеспечивающее прогнозирование и анализ состояния сети водоснабжения, тем самым оптимизировав работу системы в целом. Такое ПО позволяет повысить эффективность использования ресурсов воды, предотвратить возможные аварии и некачественное водоснабжение, а также обеспечить быстрый и точный анализ данных, полученных с датчиков и другого оборудования. Операторы водоснабжения, используя такое ПО, могут быстро реагировать на изменения в потреблении воды и других факторах, связанных с работой сети, и принимать своевременные меры для предотвращения возможных аварийных ситуаций.

Для достижения поставленной цели необходимо выполнить ряд задач:

- Изучение источников и анализ источников;
- Анализ предметной области;
- Анализ функциональных требований;

- Анализ программного обеспечения аналогичного назначения;
- Проектирование архитектуры ПО;
- Проектирование базы данных ПО;
- Выбор программно-аппаратных средств разработки;
- Проектирование интерфейса;
- Планирование и расчёт бюджета разрабатываемого ПО;
- Кодирование ПО;
- Разработка пользовательского руководства для ПО;
- Выбор методик тестирования и тестирование пользовательского интерфейса и серверной части ПО.

Объект исследования – автоматизированные информационные системы.

Предмет – автоматизация процессов анализа и прогнозирования сети водоснабжения.

В соответствии с намеченной целью и задачами исследования нами были определены следующие методы:

Теоретические:

- Теоретический анализ технической, экономической и методической литературы по теме исследования;
- Изучение устройства автомата продажи воды «Ключ здоровья»;
- Анализ функционирования автоматов продажи воды.

Эмпирические:

- Разработка ПО;
- Тестирование ПО.

Аналитические:

- Анализ и прогнозирование расхода воды;
- Оптимизация маршрутов доставки воды.

1 Разработка и анализ требований для ПО прогнозирования и анализа состояния сети водоснабжения

1.1 Анализ предметной области и функциональных требований

1.1.1 Анализ предметной области

Основными задачами, рассматриваемыми в данной работе, являются: учёт автоматов для продажи воды, анализ состояния автоматов, учёт поставщиков и парка их автомобилей, прогнозирование расходов воды, учёт транзакций, а также построение эффективного маршрута доставки воды.

Организация «Ключ здоровья» представляет собой городскую сеть автоматов продажи артезианской воды. Каждый автомат имеет датчик, позволяющий контролировать текущее техническое состояние, а также уровень воды в резервуаре. Стоимость литра воды для каждого автомата определяется в индивидуальном порядке. Каждому автомату соответствует адрес (страна, округ, город, район, улица, дом). Также автоматы имеют резервуары для хранения воды различных объёмов, вода в них должна обновляться не реже раза в трое суток. Организации поставщики должны регулярно восполнять запас воды путём ее транспортировки и заправки в резервуар. Маршрут доставки планируется сразу для нескольких точек, на основе кратчайшего пути между ними.

Документы, создаваемые в предметной области – отчёты об учётных операциях над автоматами продажи воды: учёта прибыли от продажи, а также учёта проданной воды.

Действующие лица предметной области:

- Главный специалист финансового отдела (отвечает за учет и анализ операций продажи воды);
- Главный специалист отдела учета автоматов продажи воды (отвечает за учет и анализ технического состояния автоматов продажи воды);

– Главный специалист отдела обслуживания (отвечает за учет и анализ компаний–поставщиков, парка их автомобилей, а также за своевременную и эффективную транспортировку воды).

На рисунке 1 показана диаграмма базы данных, отражающая предметную область.

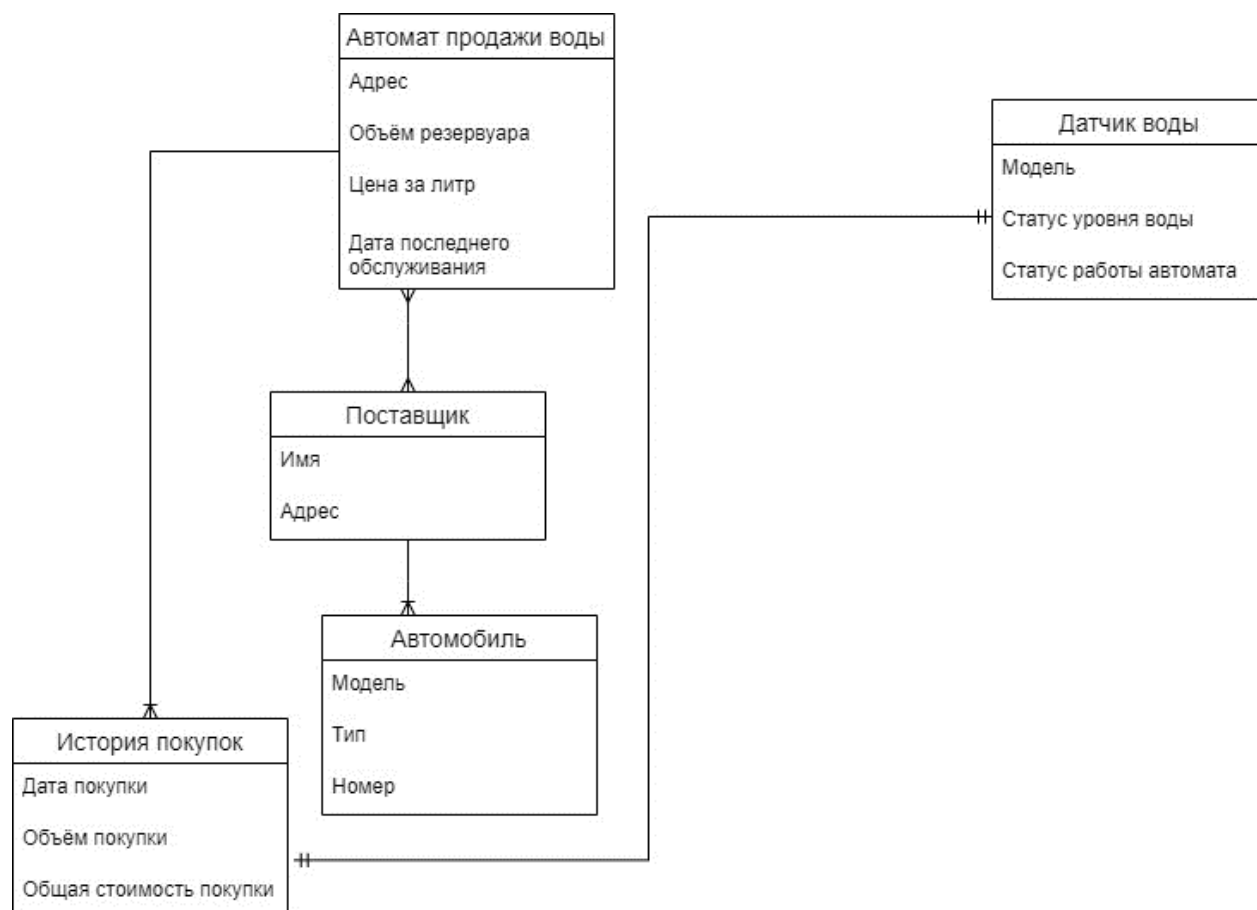


Рисунок 1 – Концептуальная модель базы данных

На рисунке 1 также отображены все основные сущности описываемой предметной области: «Автомат продажи воды», «Поставщик», «История покупок», «Автомобиль», «Датчик воды».

1.1.2 Анализ функциональных требований

Определим основные функциональные требования для описываемой предметной области:

- Подключить новый автомат продажи воды;
- Включить автомат продажи воды;

- Отключить автомат продажи воды;
- Выполнить прогнозирование дохода нового автомата продажи воды;
- Составить эффективный маршрут транспортировки воды;
- Восполнить запасы воды и произвести техническое обслуживание;
- Подключить нового поставщика воды;
- Отключить поставщика воды;
- Добавить в парк новый автомобиль;
- Удаление автомобиля из парка;
- Произвести сравнительный анализ востребованности автоматов продажи воды.

Для отображения выявленных в ходе анализа функциональных требований в данной работе используется диаграмма вариантов использования.

Диаграмма вариантов использования (use case diagram) описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки. Разработка диаграммы вариантов использования преследует цели:

- Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
- Сформулировать общие требования к функциональному поведению проектируемой системы;
- Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями [1].

На Рисунке 2 показана диаграмма зон ответственности и алгоритм деятельности работников, отвечающих за работу телефонной станции учёта и анализа системы водоснабжения.

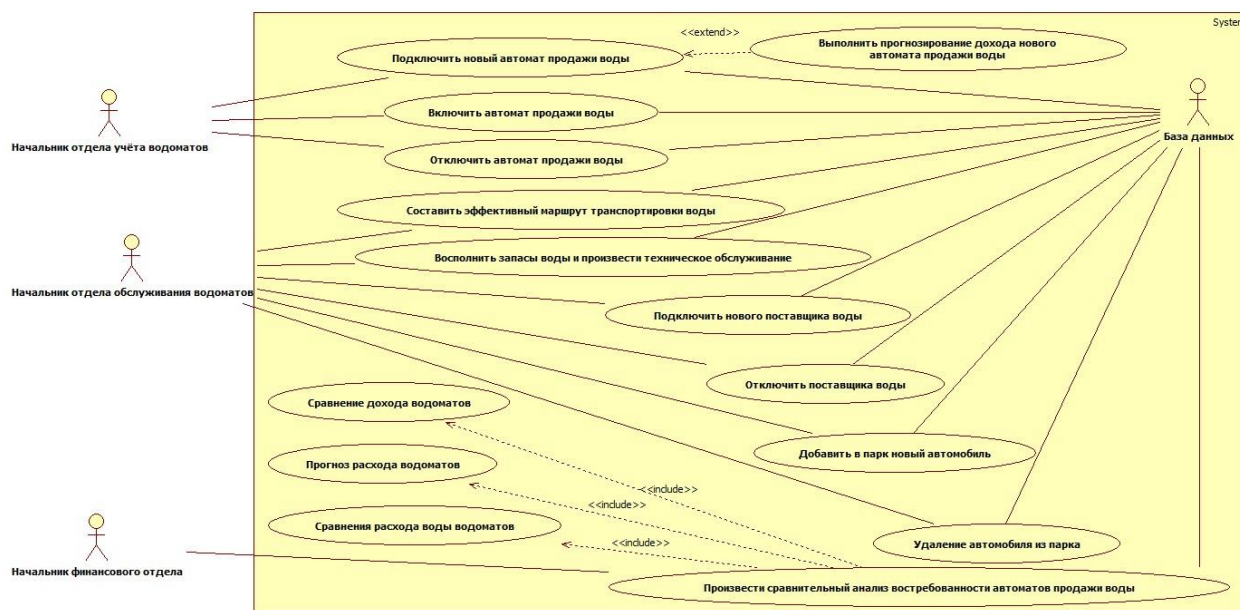


Рисунок 2 – Алгоритм деятельности работников, отвечающих за работу телефонной станции учёта и анализа системы водоснабжения

В Таблицах 1-3 ниже представлен анализ сценария (алгоритма выполнения) конкретных задач, представленных ранее на Рисунке 2.

Таблица 1 – Сценарий задачи «Подключить новый автомат продажи воды»

Наименование: Подключить новый автомат продажи воды
ID: 1
Краткое описание: Система осуществляет добавление нового аппарата продажи воды в базу данных
Действующие лица: Начальник отдела учёта водоматов , База данных
Предусловие: Система загружена, база данных подключена к приложению
Основной поток: <ol style="list-style-type: none"> 1. Прецедент начинается, когда пользователь выбирает опцию «Добавить водомат». 2. Система предоставляет пользователю окно выбора параметров водомата. 3. Пользователь определяет параметры водомата. 4. Система добавляет новый водомат в базу данных. 5. Система отображает таблицу водоматов, включая добавленный водомат.
Постусловие: Содержимое базы данных обновлено

Таблица 2 – Алгоритм выполнения задачи «Восполнить запасы воды и произвести техническое обслуживание»

Наименование: Восполнить запасы воды и произвести техническое обслуживание
ID: 2
Краткое описание: Система осуществляет пополнение баланса воды, а также дадут последнего обслуживания. Содержимое базы данных обновляется.
Действующие лица: Начальник отдела обслуживания водоматов , База данных
Предусловие: Система загружена, база данных подключена к приложению. Выполнено построение маршрута.
Основной поток: 1. Прецедент начинается, когда пользователь выбирает опцию «Обслужить». 2. Система обновляет содержимое резервуара водоматов. 3. Система обновляет дату последнего обслуживания водоматов. 4. Данные сохраняются в базе данных. 5. Система отображает сообщение об успешном обслуживании водомата.
Постусловие: Содержимое базы данных обновлено

Таблица 3 – Алгоритм выполнения задачи «Произвести сравнительный анализ востребованности автоматов продажи воды»

Наименование: Произвести сравнительный анализ востребованности автоматов продажи воды
ID: 3
Краткое описание: Система производит сравнение автоматов продажи воды..
Действующие лица: Начальник финансового отдела , База данных
Предусловие: Система загружена, база данных подключена к приложению. В базе имеются логи пользования водоматом.
Основной поток: 1. Прецедент начинается, когда пользователь выбирает опцию «Анализ». 2. Пользователь выбирает способ отображения анализа. 3. Система загружает исходные данные из БД и проводит анализ. 4. Система отображает результаты анализа..
Постусловие: Произведен сравнительный анализ

Таким образом были сформулированы функциональные требования к программе.

1.2 Анализ программного обеспечения аналогичного назначения

Программы прогнозирования и анализа систем водоснабжения являются актуальными, так как они играют важную роль в поддержании надлежащего функционирования водоснабжающих систем. В современных условиях увеличивается потребление воды, что в свою очередь увеличивает нагрузку на системы водоснабжения. Кроме того, климатические изменения способствуют нарушению стабильности процессов, увеличивая риски возникновения аварийных ситуаций и усложняя планирование производства и распределения воды.

Программы прогнозирования и анализа систем водоснабжения позволяют оперативно реагировать на изменения внешних условий и направлять ограниченные ресурсы на наиболее важные и нагруженные участки. Они также помогают предоставлять более высокое качество воды и снижать потери в системе водоснабжения путем оптимизации режима работы насосных станций, управления водопроводами и учета расхода воды.

Таким образом, программы прогнозирования и анализа систем водоснабжения являются необходимыми инструментами для обеспечения устойчивого функционирования систем водоснабжения и справедливого распределения ограниченных ресурсов в современных условиях.

Среди подобного программного обеспечения стоит выделить следующие:

- WaterGEMS – программа для создания моделей водоснабжения, которая помогает совершенствовать эффективность, сохранность и ценообразование;
- EPANET – бесплатное программное обеспечение, разработанное Агентством по охране окружающей среды США, которое используется для создания моделей и расчета параметров сетей водоснабжения;
- H2OMAP Water - это программа, которая используется для проектирования и оптимизации систем водоснабжения. Она предоставляет возможность анализировать различные параметры, такие как расход воды,

давление и качество воды, и помогает определить наиболее эффективные пути улучшения системы водоснабжения.

1.2.1 ПО WaterGEMS

WaterGEMS – это программный комплекс, разработанный компанией Bentley Systems, ориентированный на моделирование систем водоснабжения. Он представляет собой расширение для платформы MicroStation и включает инструменты для проведения анализа гидравлической эффективности системы водоснабжения и канализации, а также предоставляет функционал для оптимизации эксплуатации водоснабжения.

Основные функции ПО WaterGEMS включают:

1. Моделирование гидравлических систем водоснабжения и канализации;
2. Анализ и оптимизация работы систем водоснабжения и канализации;
3. Поддержка различных систем координат;
4. Интеграция с другими видами Программного обеспечения компании Bentley Systems;
5. Функция отслеживания изменений изменений системы и мониторинга данных;
6. Создание отчетов и документации.

WaterGEMS позволяет моделировать различные типы систем водоснабжения и канализации, включая системы с естественным потоком и системы с использованием насосов. При помощи программы можно анализировать работу системы при различных условиях, таких как распределение тока, расход воды, потери давления и т.д. В случае обнаружения проблем, ПО предложит оптимальный вариант решения проблемы.

WaterGEMS может использоваться на объектах с различными масштабами, от маленьких населенных пунктов до крупных городских канализационных систем. Он также позволяет анализировать системы в различных режимах работы (например, в аварийном режиме). [10]

1. Задачи, которые может выполнять ПО WaterGEMS:
2. Определение оптимальной пропускной способности;
3. Определение минимального размера трубопроводов;
4. Определение уровня напора;
5. Отслеживание утечек воды;
6. Определение максимального расхода в системах пожаротушения;
7. Создание гидравлической модели.

Интерфейс ПО WaterGEMS представлен на Рисунке 3 ниже.

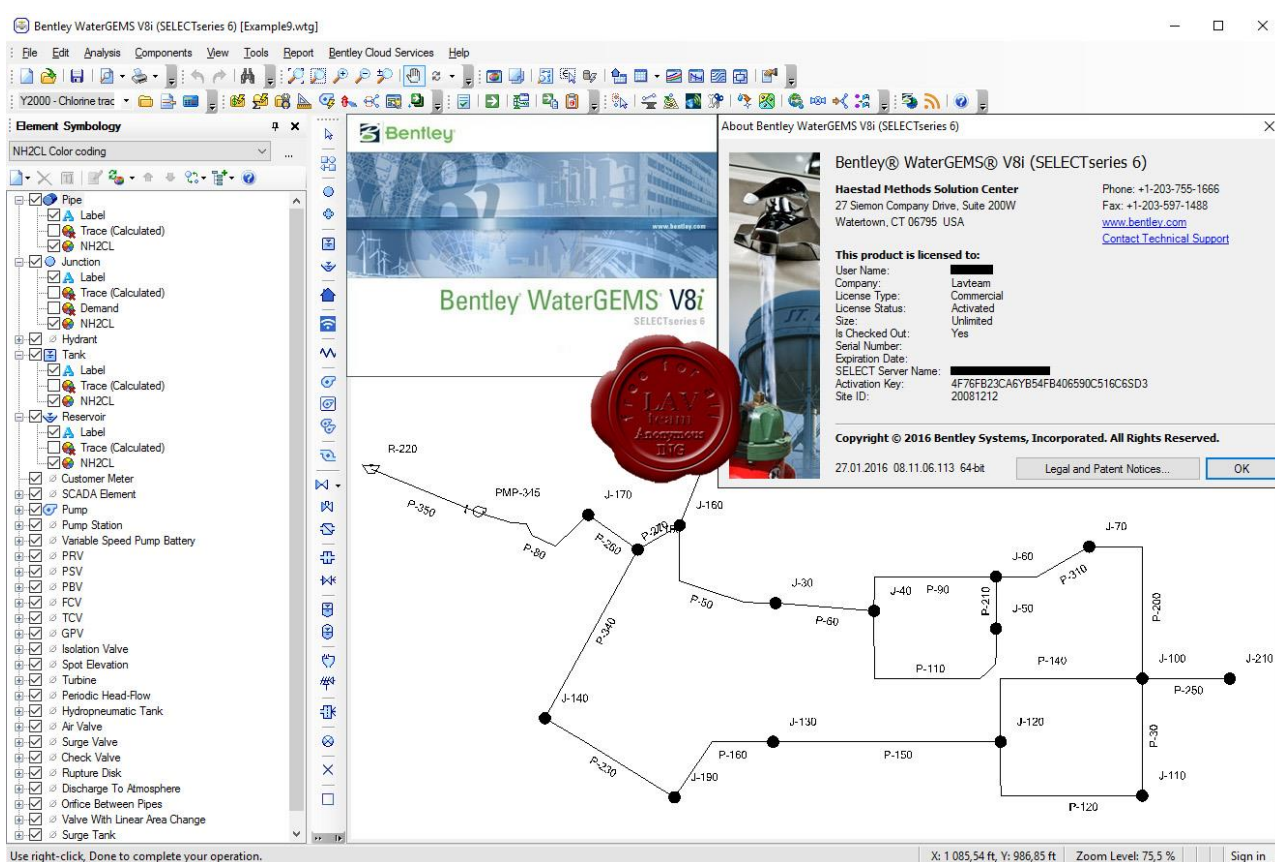


Рисунок 3 – Интерфейс ПО WaterGEMS

В целом, ПО WaterGEMS представляет собой мощный инструмент для проведения анализа работоспособности и оптимизации систем водоснабжения и канализации.

1.2.2 ПО EPANET

EPANET - это свободно распространяемое программное обеспечение,

созданное Агентством по защите окружающей среды США для моделирования систем водоснабжения. Она предоставляется как набор инструментов для анализа гидравлических и водопотребительных характеристик систем водоснабжения и канализации и является одной из наиболее распространенных программ для моделирования систем водоснабжения.

Основные функции ПО EPANET включают:

1. Моделирование гидравлических систем водоснабжения и канализации;
2. Анализ и оптимизация работы систем водоснабжения и канализации;
3. Поддержка различных систем координат;
4. Расчет потребностей в воде и установление цен на водоснабжение;
5. Расчет затрат на эксплуатацию систем водоснабжения;
6. Импорт и экспорт данных.

EPANET предоставляет функционал для моделирования различных типов систем водоснабжения и канализации, включая системы с естественным потоком и системы с использованием насосов. При помощи программы можно анализировать работу системы при различных условиях, таких как распределение тока, расход воды, потери давления и т.д. В случае обнаружения проблем, ПО предложит оптимальный вариант решения проблемы.

EPANET также предоставляет возможность проведения анализа водопотребления, в том числе определения объема водопотребления, используемого в различных зонах, анализ ресурсов и установление цен на воду. Эти функции помогают расчет прогноза будущих потребностей в воде и установление цен на воду в соответствии с данными. [13]

EPANET может использоваться на объектах различных масштабов, от маленьких населенных пунктов до крупных городских систем водоснабжения. Она может использоваться для анализа систем в различных режимах работы (например, в аварийном режиме).

1. Задачи, которые может выполнять ПО EPANET:
2. Определение оптимальной пропускной способности;
3. Определение минимального размера трубопроводов;

4. Определение уровня напора;
5. Отслеживание утечек воды;
6. Определение максимального расхода в системах пожаротушения;
7. Создание гидравлической модели.

Интерфейс ПО EPANET представлен на рисунке 4 ниже.

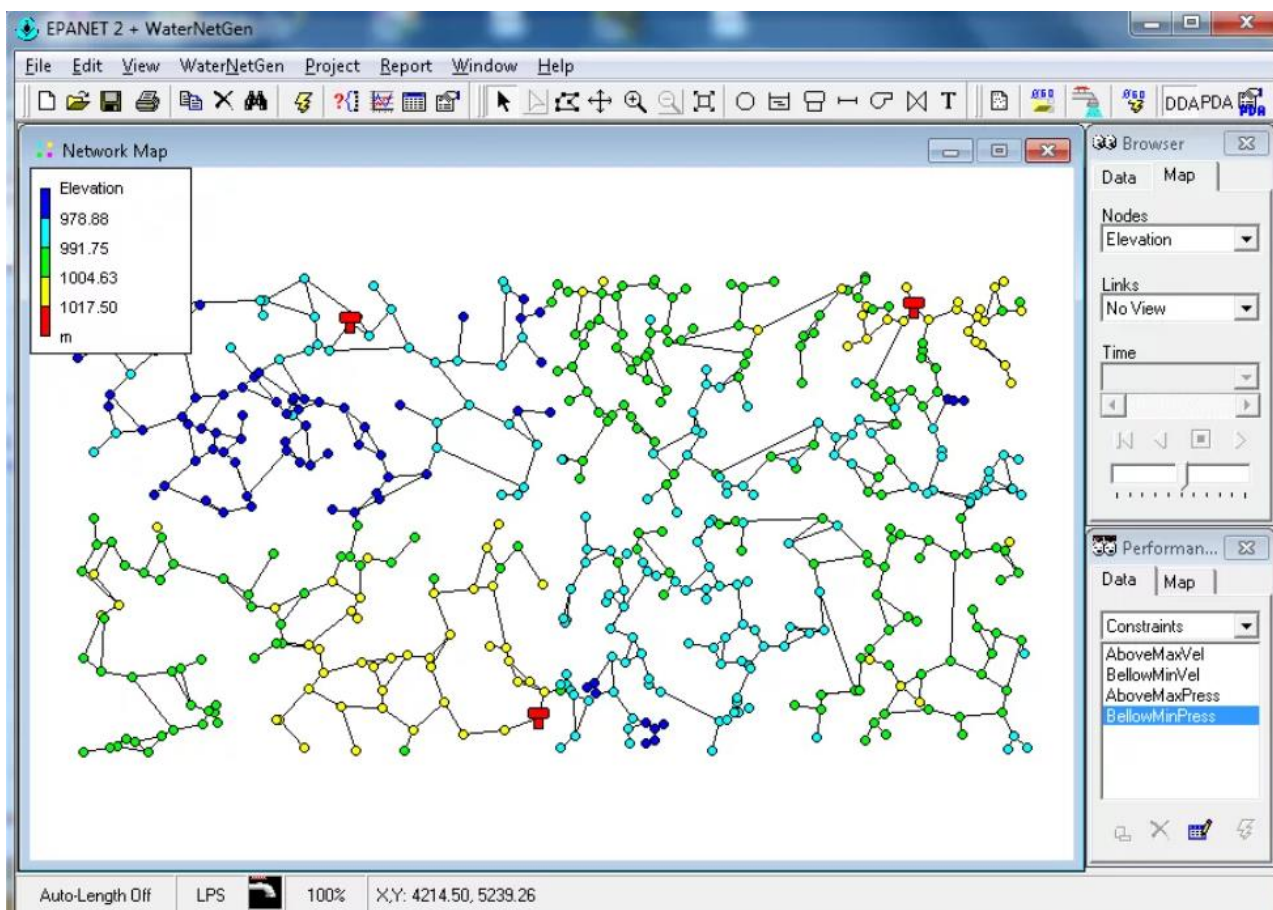


Рисунок 4 – Интерфейс ПО EPANET

ПО EPANET представляет собой мощный инструмент для проведения анализа работоспособности и оптимизации систем водоснабжения и канализации, а также для решения проблем, связанных с водопотреблением и управлением ценами на воду.

1.2.3 ПО H2OMAP Water

H2OMAP Water - это программное обеспечение, разработанное компанией MWH Global с целью моделирования систем водоснабжения и канализации. Его интерфейс предназначен для удобства работы с программой, а также для

упрощения и облегчения процесса создания и редактирования моделей.

Основные функции программы H2OMAP Water включают:

1. Моделирование гидравлических систем водоснабжения и канализации;
2. Анализ и оптимизация работы систем водоснабжения и канализации;
3. Поддержка различных систем координат;
4. Выбор различных вариантов анализа;
5. Создание 3D-моделей водопроводных сетей;
6. Визуализация данных.

H2OMAP Water позволяет моделировать различные типы систем водоснабжения и канализации, включая системы с естественным потоком и системы с использованием насосов. При помощи программы можно анализировать работу системы при различных условиях, таких как распределение тока, расход воды, потери давления и т.д. В случае обнаружения проблем, ПО предложит оптимальный вариант решения проблемы.

H2OMAP Water может использоваться на объектах с различными масштабами, от маленьких населенных пунктов до крупных городских канализационных систем. Он также позволяет анализировать системы в различных режимах работы (например, в аварийном режиме) и создавать 3D-модели систем водоснабжения и канализации для визуализации данных. [11]

Задачи, которые может выполнять ПО H2OMAP Water:

1. Определение оптимальной пропускной способности;
2. Определение минимального размера трубопроводов;
3. Определение уровня напора;
4. Отслеживание утечек воды;
5. Определение максимального расхода в системах пожаротушения;
6. Создание гидравлической модели.

Интерфейс ПО H2OMAP Water приведён на Рисунке 5 ниже.

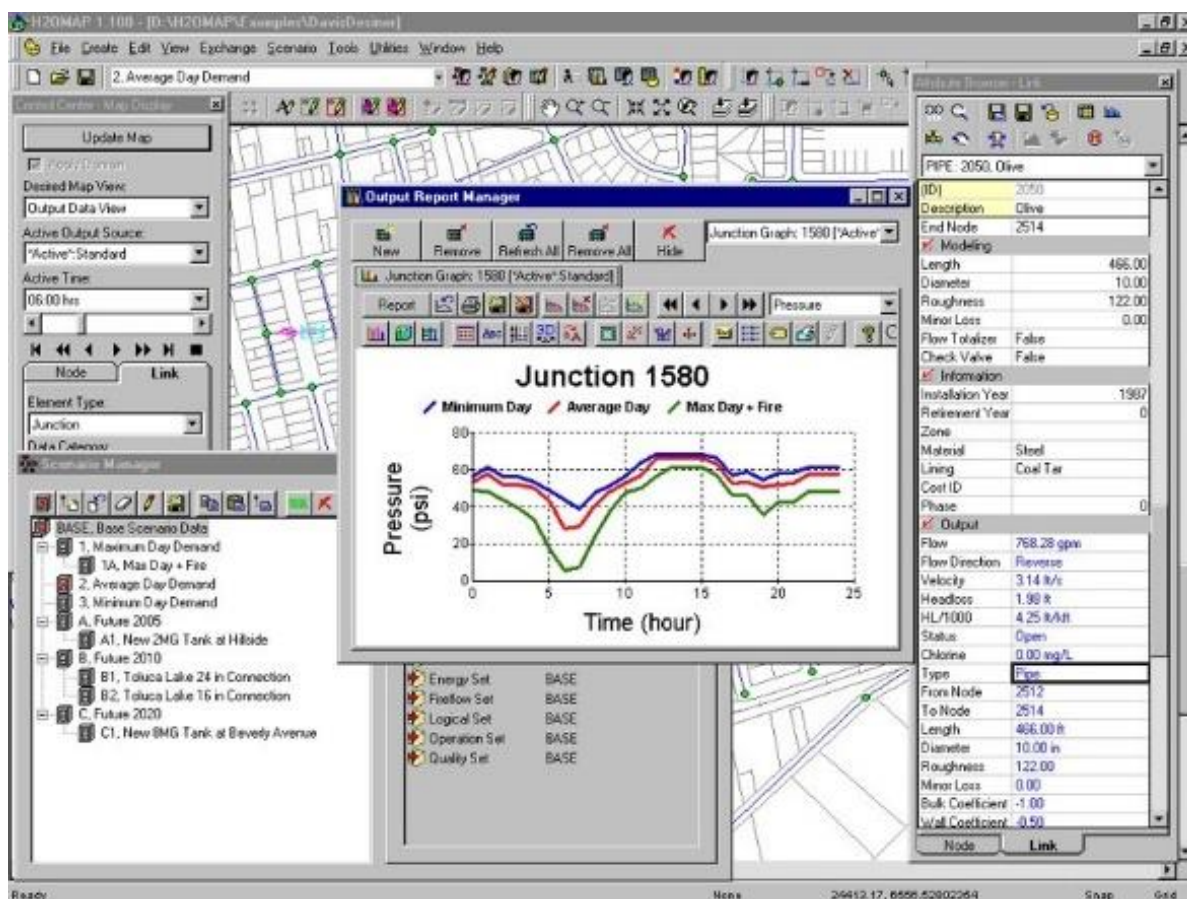


Рисунок 5 – Интерфейс ПО H2OMAP Water

В целом, ПО H2OMAP Water является мощным инструментом для проведения анализа работоспособности и оптимизации систем водоснабжения и канализации, а также для решения проблем, связанных с водопотреблением и управлением ценами на воду.

1.2.4 Сравнение аналогов

После приведения нескольких аналогов с аналогичными или частично схожими функциями, необходимо сделать выводы. Для этого мы создадим таблицу, которая будет содержать основные критерии: построение маршрута водоснабжения, анализ качества воды, возможность настройки пользовательского интерфейса, прогнозирование расхода воды, встроенный инструмент моделирования работы датчика уровня воды.

Таблица 4 – Сравнение аналогов

ПО анализа сети водоснаб жения	Построен ие маршрута водоснаб жения	Анализ качества воды	Возможность настройки пользователь ского интерфейса	Прогнозир ование расхода воды	Встроенный инструмент моделирова ния работы датчика уровня воды
WaterGE MS	+	+	+	-	-
EPANET	+	+	-	-	-
H2OMAP Water	+	+	-	-	-
Разрабаты ваемое ПО	+	-	-	+	+

После проведения анализа, было принято решение разработать собственное ПО, выполняющее функции, которые выполняются системами аналогичного назначения, а также собственные узкоспециализированные задачи. Несмотря на ограниченность временных и денежных ресурсов в сравнении с разработчиками аналогичных систем, мы считаем, что разработка собственного ПО для решения поставленных задач – целесообразна.

1.3 Регрессионный анализ

Для реализации задачи прогнозирования в данной работе будет произведён регрессионный анализ.

В регрессионном анализе рассматривается связь между одной зависимой переменной и несколькими другими независимыми переменными. Эта связь представляется с помощью математической модели. Выбор подходящей модели основывается как на статистических доводах, так и на основе содержательного

смысла моделируемой зависимости.

Зависимая (объясняемая результирующая, эндогенная) переменная (признак) y - характеризует результат функционирования экономической системы. Ее значения формируются под воздействием других переменных факторов, поэтому результирующая переменная является случайной величины. Объясняющие (независимые, предикторные, экзогенные) переменные

$X = (x_1, x_2, \dots, x_m)$ – переменные (признаки), которые описывают условия функционирования экономической системы и в существенной мере определяют значения объясняемой переменной. Независимые переменные могут быть как случайными, так и детерминированными.

Статистические исследования направлены на оценивание параметров регрессии; проверку гипотез о статистической значимости этих параметров; проверку адекватности построенной модели.

Регрессионный анализ используется с двумя целями. Во-первых, для описания зависимости между переменными и определения причинной связи. Во-вторых, для построения прогнозных значений зависимой переменной. Мерой зависимости является величина коэффициента корреляции. [4]

1.3.1 Модель авторегрессии

Для прогнозирования расхода автомата продажи воды планируется использовать показания за предыдущий период. Данные в таком случае будут представлять собой временной ряд. В таком случае, наиболее подходящей моделью, для решения поставленной задачи, будет авторегрессионная модель.

Модель авторегрессии (AR) - это статистическая модель, которая используется для прогнозирования временных рядов. В модели AR используется линейная регрессия для прогнозирования значения временного ряда на основе его предыдущих значений.

Основная идея модели AR заключается в том, что значение временного ряда в момент времени t зависит от его предыдущих значений. То есть, чем больше времени прошло с момента последнего измерения, тем меньше влияние

этого значения на прогноз.

Модель AR обычно обозначается как $AR(p)$, где p - это количество предыдущих значений временного ряда, используемых для прогнозирования текущего значения. Например, $AR(1)$ означает, что для прогнозирования значения временного ряда в момент времени t используется только его значение в момент времени $t-1$.

Формально, модель AR можно записать как:

$$y(t) = c + \varphi_1 * y(t - 1) + \varphi_2 * y(t - 2) + \dots + \varphi_p * y(t - p) + \varepsilon(t) \quad (1)$$

где $y(t)$ - это значение временного ряда в момент времени t , c - это постоянная, $\varphi_1, \varphi_2, \dots, \varphi_p$ - это параметры регрессии, $\varepsilon(t)$ - это ошибка прогноза.

Таким образом, для прогнозирования значения временного ряда в момент времени t используется его предыдущее значение в момент времени $t-1, t-2, \dots, t-p$. Параметры регрессии настраиваются таким образом, чтобы минимизировать сумму квадратов ошибок прогноза.

Модель AR может быть использована для прогнозирования как стационарных, так и нестационарных временных рядов. Однако, для нестационарных рядов требуется проводить дополнительные преобразования, такие как разность или логарифмирование, чтобы сделать ряд стационарным.[4]

Модель AR имеет ряд ограничений. В частности, она не может учитывать воздействие вне ряда факторов, таких как экономические, социальные или политические события, которые могут оказывать влияние на изменение временного ряда. Кроме того, она не может учитывать изменения тренда в ряде.

В целом, модель AR является мощным инструментом для прогнозирования временных рядов. Однако, ее эффективность зависит от того, насколько хорошо ряд описывается моделью и насколько точно настраиваются параметры регрессии.

Модель множественной регрессии

В ходе разработки также планируется реализовать задачу прогнозирования среднесуточного дохода автомата, зависимость от нескольких независимых критериев. Для этой задачи подойдёт модель множественной регрессии.

Модель множественной регрессии представляет собой статистическую модель, которая помогает прогнозировать зависимую переменную на основе набора независимых переменных. Множественная регрессия используется для анализа и прогнозирования отношений между переменными.

Модель множественной регрессии определяется уравнением:

$$y = b_0 + b_1x_1 + b_2x_2 + \dots b_nx_n + e \quad (2)$$

где y - зависимая переменная, $x_1, x_2, \dots, x_n$ - независимые переменные, $b_0, b_1, b_2, \dots, b_n$ - коэффициенты регрессии, e - остаточная ошибка.

Коэффициенты регрессии $b_0, b_1, b_2, \dots, b_n$ показывают влияние каждой независимой переменной на зависимую переменную. Они вычисляются с помощью метода наименьших квадратов, который минимизирует сумму квадратов остаточных ошибок.[4]

Множественная регрессия учитывает влияние нескольких независимых переменных на зависимую переменную, что позволяет более точно прогнозировать ее значения. Однако при использовании этой модели необходимо учитывать многие факторы, такие как линейность отношений между переменными, нормальность распределения ошибок, отсутствие мультиколлинеарности и т.д.

Множественная регрессия часто используется в экономических и финансовых моделях для прогнозирования различных финансовых показателей. Она также широко применяется в медицинских и социологических исследованиях для анализа факторов, влияющих на здоровье или социальное поведение людей.

1.4 Задача коммивояжёра

Для реализации функционального требования «Составить эффективный маршрут транспортировки воды» необходимо будет решить задачу коммивояжёра по поиску оптимального пути между всеми заданными точками маршрута и возвращением в исходную точку.

Задача коммивояжера – это одна из самых известных и изученных задач в

области комбинаторной оптимизации. Эта задача заключается в поиске кратчайшего пути, проходящего по всем городам из некоторого множества городов, причем каждый город должен быть посещен ровно один раз, а начальный и конечный город должны совпадать.

Формулировка задачи:

Пусть имеется некоторое множество городов и матрица расстояний между ними. Задача состоит в том, чтобы найти такой путь, который решает следующую оптимизационную задачу:

1. Путь должен начинаться и заканчиваться в одном и том же городе;
2. Каждый город должен быть посещен ровно один раз;
3. Общая длина пути должна быть минимальной.

Решение задачи:

Существует множество способов решения задачи коммивояжера. Одним из наиболее распространенных методов является метод ближайшего соседа.

Метод ближайшего соседа заключается в следующем: начиная с любого города, выбирается ближайший к нему город, который еще не был посещен, затем из этого города выбирается ближайший еще не посещенный город, и так далее, пока не будут посещены все города.

Естественно, что этот метод не всегда дает оптимальное решение, однако он является достаточно быстрым и простым в реализации.

Другой метод решения задачи коммивояжера - метод ветвей и границ. Суть этого метода заключается в том, чтобы рекурсивно дробить пространство всех возможных путей на все более мелкие и мелкие фрагменты, пока не будет найдено решение на основе выполнения каких-то ограничений.

Метод ветвей и границ занимает гораздо больше времени, чем метод ближайшего соседа, однако он дает гарантированно оптимальное решение. Кроме того, этот метод предоставляет возможности для уточнения ответа, которых нет в методе ближайшего соседа.

В заключение можно сказать, что задача коммивояжера является не только одной из самых сложных задач в комбинаторной оптимизации, но и одной из

наиболее применимых задач в практике. Задача находит свое применение в таких областях, как логистика, проектирование маршрутов доставки, планирование дорожных работ и многих других.

2 Проектирование ПО

2.1 Проектирование архитектуры ПО

В основу данного приложения была заложена архитектура MVP(Model-View-Presenter), которая позволяет с легкостью масштабировать и изменять структуру без существенных трудностей.

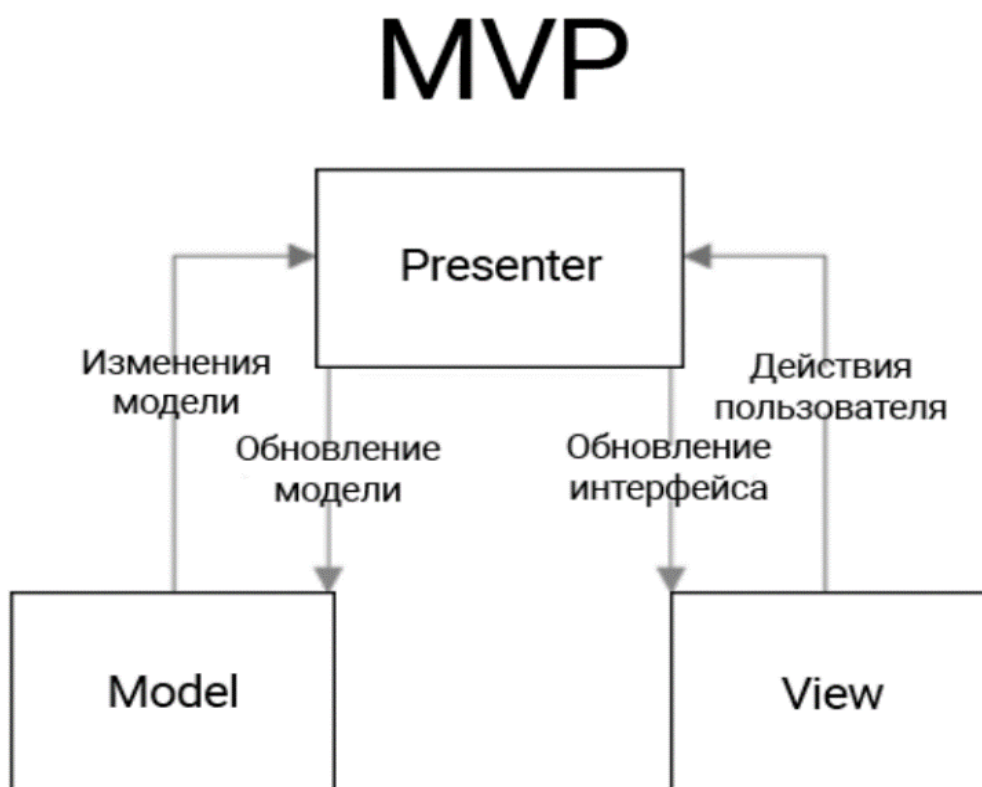


Рисунок 6 – Общая схема архитектуры MVP

MVP (Model-View-Presenter) - это шаблон проектирования, который применяется в разработке программного обеспечения для создания отдельных компонентов приложения, а также для облегчения тестирования и управления его состоянием. Он был разработан как вариант шаблона Model-View-Controller (MVC) и часто используется в мобильной разработке.

MVP разделяет приложение на три основных компонента: модель (Model), представление (View) и презентер (Presenter). Каждый из этих компонентов имеет свою роль и ответственности в процессе разработки приложения:

Модель представляет собой бизнес-логику приложения и данные. Она содержит данные, а также методы для их получения, обновления и удаления.

Представление - это пользовательский интерфейс приложения. Оно предоставляет пользователю доступ к данным и действиям, которые ему доступны, а также отображает изменения, которые он вносит в данные.

Презентер служит связующим звеном между моделью и представлением. Он получает данные от модели и передает их представлению, а также принимает уведомления о действиях пользователя на представлении и передает их модели для обработки.

Основная идея MVP заключается в том, что представление и модель связаны между собой только через презентер. Это позволяет упростить код приложения и сделать его более гибким, а также облегчить его тестирование.

MVP может быть использован в различных областях разработки, но чаще всего он используется в мобильной разработке, где есть много взаимодействия с пользователем и необходимо быстро и гибко реагировать на его действия. Он помогает разработчикам создавать приложения, которые легко масштабировать, обновлять и поддерживать.

Паттерн состоит из трех компонентов: модель (Model), вид (View) и презентатор (Presenter) и позволяет создавать абстракцию для представления. Модель содержит в себе всю бизнес-логику приложения. У представления имеется интерфейс с определенным набором свойств и методов, однако оно не содержит никакой бизнес-логики. Презентатор же является посредником между представлением и моделью. Он получает ссылку на реализацию интерфейса, подписывается на события представления и по запросу изменяет модель. [2]

Для данного приложения создана диаграмма классов. На Рисунках 7-10 и в Приложении Б представлены отдельные фрагменты, содержащие классы и отношения между ними.

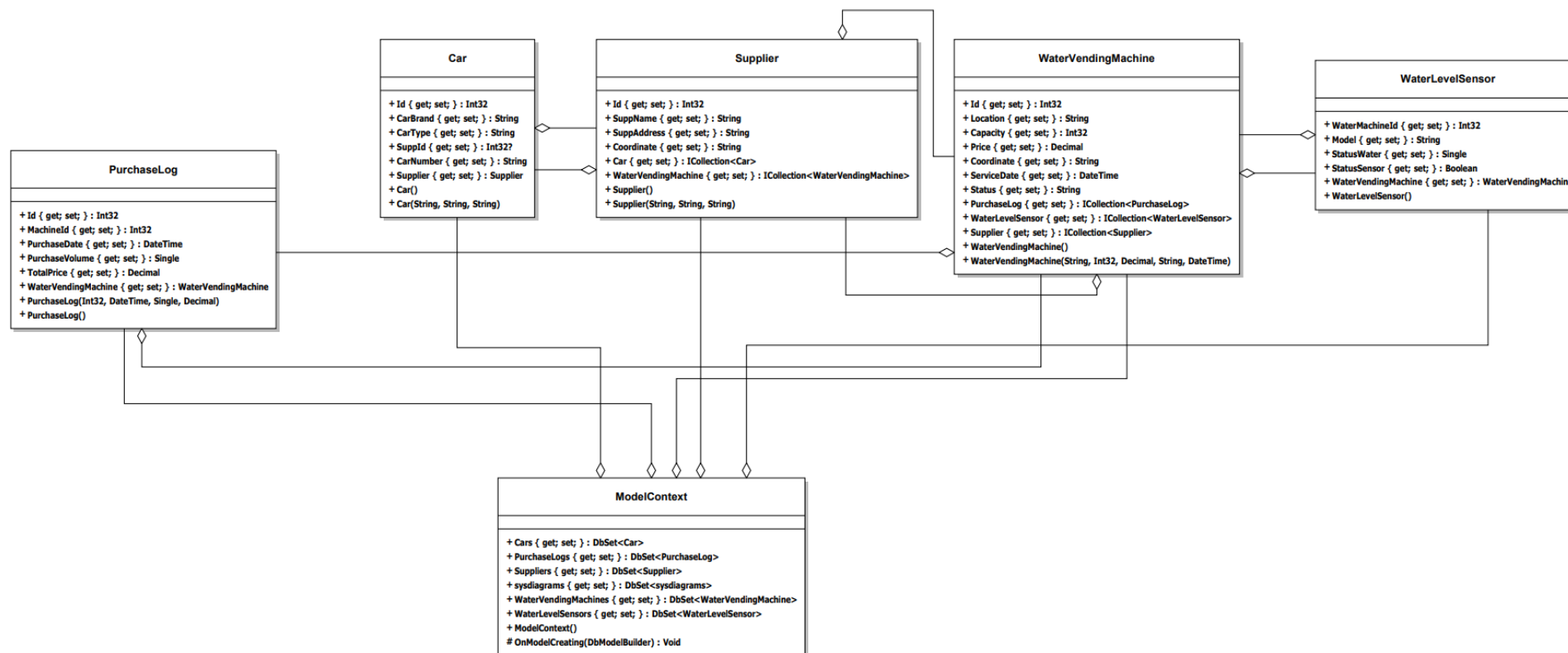


Рисунок 7 – Диаграмма классов для Model

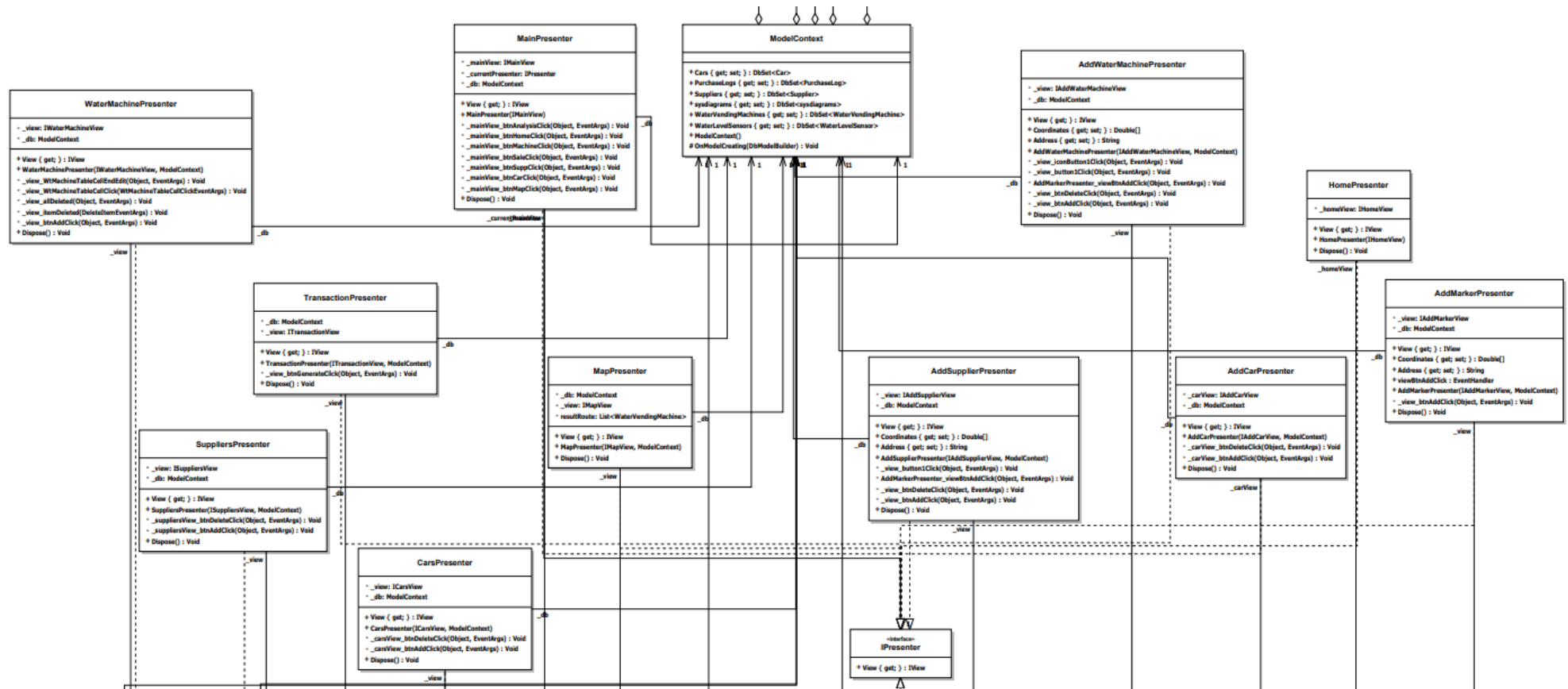


Рисунок 8 – Диаграмма классов для Presenter

Рассмотрим, как работает архитектура MVP в данной программе на примере классов (интерфейсов) AddCarPresenter, IAddCarView и ModelContext.

ModelContext (контекст базы данных) является репозиторием, предоставляющим доступ к моделям предметной области (домена). Репозиторий отделяет прямую работу с базой данных и детали ее реализации от остальной части приложения, используя ORM-систему (Object-Relational Mapping, объектно-реляционное преобразование) из Entity Framework[5] для связывания сущностей базы данных с объектами приложения.

На Рисунке 11 ниже показана схема взаимодействия описанных классов.



Рисунок 11 – Схема взаимодействия классов IAddCarView.cs, AddCarPresenter.cs, ModelContext.cs

Презентатор AddCarPresenter, являющийся посредником между моделью и интерфейсом для добавления нового автомобиля (IAddCarView). Презентатор подписывается на событие добавления автомобиля у представления и при его обработке получает от него введенные данные, создает новый объект автомобиля и добавляет его в базу данных с помощью соответствующего метода, а также на событие, отменяющее добавление нового автомобиля.

Интерфейс IAddCarView определяет свойство для привязки данных, свойства для получения введенных пользовательских данных, а также события подтверждения и отмены добавления автомобиля. При подтверждении добавления автомобиля (в случае пользовательской форме это происходит при нажатии на кнопку) вызывается соответствующее событие, на которое подписан презентатор, что позволяет ему выполнить необходимую логику и организовать взаимодействие между моделью и представлением. Реализация данного

интерфейса – форма AddCarForm.

2.2 Проектирование базы данных

В качестве СУБД выберем MS SQL Server. Разработаем модель БД при помощи MS SQL Server Management Studio. Модель БД изображена на Рисунке 12.

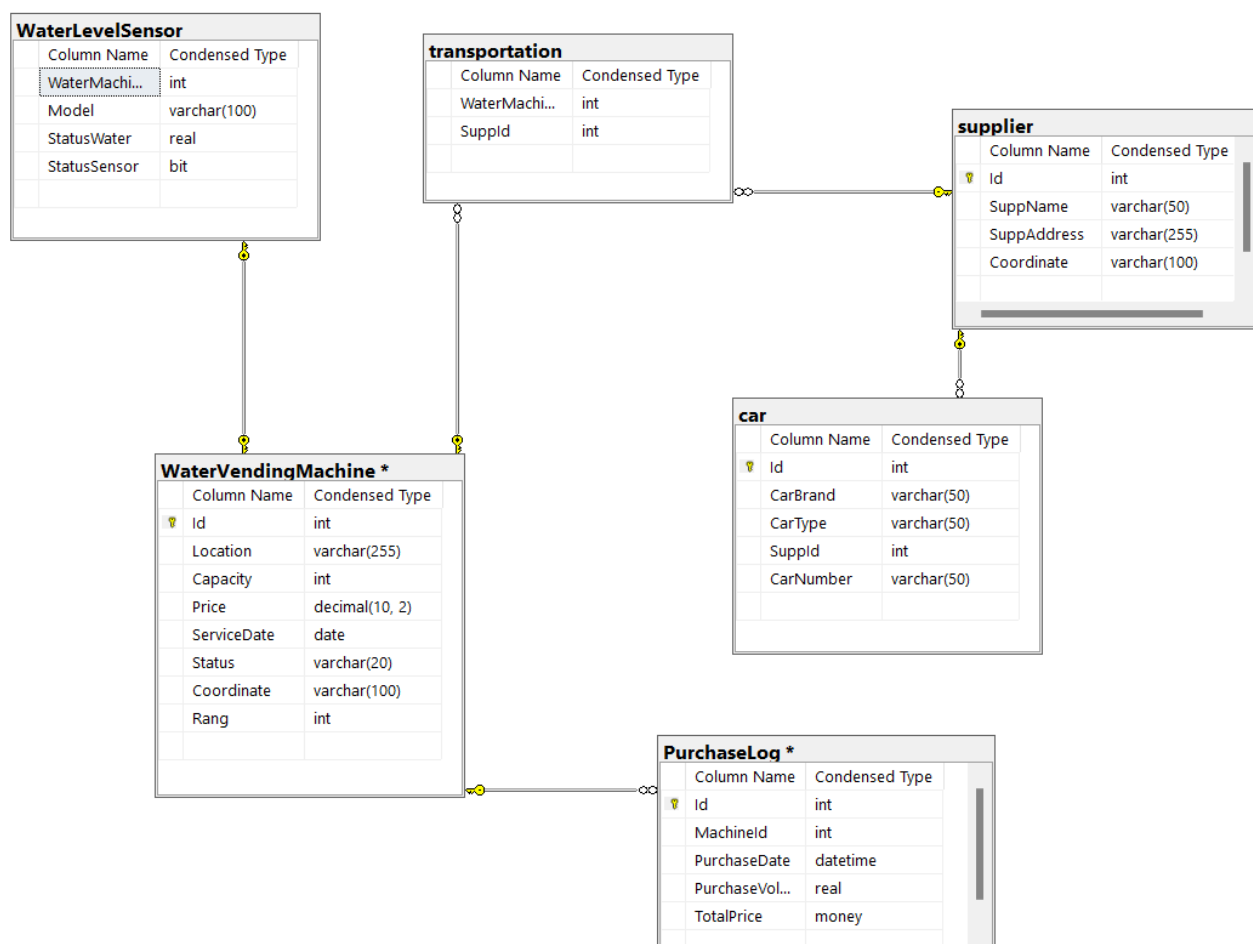


Рисунок 12 – Диаграмма классов БД

В разработанной модели отображены все таблицы БД, первичные и внешние ключи посредством которых осуществляется связь. Так же для каждой из таблиц указаны типы данных

Для хранения всех данных ПО используется SQL Server. База данных состоит из 6 таблиц. Все данные в таблице приведены в нормальную форму.

Таблица WaterVendingMachine содержит основную информацию о водоматах, WaterLevelSensor – информацию о текущем состоянии водомата,

заполненности его резервуара, PurchaseLog – логи транзакций с участием водомата, Supplier – информацию о поставщиках воды, Car – информацию о автомобиле, входящем в парк поставщика. Вспомогательная таблица transportation обеспечивает тип связи «многие-ко-многим» для таблиц «WaterVendingMachine» и «Supplier».

В таблицах 5-10 ниже приведены характеристики таблиц базы данных.

Таблица 5 – Таблица базы данных «WaterVendingMachine»

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	Id	int
2	Адрес	Location	varchar(255)
3	Объём резервуара	Capacity	int
4	Цена за 1 л	Price	decimal(10, 2)
5	Последняя дата обслуживания	ServiceDate	date
6	Статус водомата	Status	varchar(20)
7	Координаты водомата	Coordinate	varchar(100)
8	Промежуточный ранг	Rang	int

Таблица 6 – Таблица базы данных «WaterLevelSensor»

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	WaterMachineId	int
2	Модель	Model	varchar(100)
3	Текущий уровень воды	StatusWater	real
4	Статус работы водомата	StatusSensor	bit

Таблица 7 – Таблица базы данных «PurchaseLog»

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	Id	int
2	Идентификатор водомата	MachineId	int
3	Дата транзакции	PurchaseDate	datetime

4	Объём продажи	PurchaseVolume	real
5	Итоговая стоимость транзакции	TotalPrice	money

Таблица 8 – Таблица «Car» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	Id	int
2	Марка и модель автомобиля	CarBrand	varchar(50)
3	Тип автомобиля	CarType	varchar(50)
4	Идентификатор поставщика	SuppId	int
5	Номер автомобиля	CarNumber	varchar(50)

Таблица 9 – Таблица «Supplier» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор	Id	int
2	Название поставщика	SuppName	varchar(50)
3	Адрес поставщика	SuppAddress	varchar(255)
4	Координаты поставщика	Coordinate	varchar(100)

Таблица 10 – Таблица «Transportation» базы данных

№	Название	Идентификатор	Тип данных, длина
1	Идентификатор водомата	WaterMachineId	int
2	Идентификатор поставщика	SuppId	int

Взаимодействие с приложением будет обеспечиваться посредством Entity Framework, использующий подход «Code-First», автогенерирующий программный код БД.

2.3 Выбор программно-аппаратных средств

2.3.1 Выбор языка программирования

Для разработки данного ПО был выбран объектно-ориентированный язык

программирования – C#. C# является одним из наиболее популярных языков программирования для разработки десктоп-приложений. Он используется для создания широкого спектра разнообразных продуктов: от простых приложений-тулбоксов до сложных систем управления базами данных. C# является объектно-ориентированным языком, с сильной типизацией и сборкой мусора, что обеспечивает высокую эффективность работы. C# также позволяет использовать множество библиотек и компонентов .NET Framework, что делает процесс разработки приложений еще более быстрым и удобным. Кроме того, C# поддерживает многопоточность, что позволяет создавать приложения, которые распараллеливают выполнение задач и улучшают производительность. В целом, C# является высокопроизводительным, мощным и удобным инструментом для разработки десктоп-приложений.

2.3.2 Выбор необходимых модулей

Разработка пользовательского интерфейса будет обеспечиваться платформой WInForms. Windows Forms (WinForms) - это технология разработки графического интерфейса пользователя (GUI) для десктопных приложений с использованием языка программирования C#. Она позволяет разработчикам быстро и легко создавать десктопные приложения с полноценным GUI, используя различные элементы управления, такие как кнопки, поля ввода, таблицы, деревья и т.д. Эти элементы могут быть настроены под различные нужды, включая различную цветовую гамму, шрифты и размеры. Windows Forms также поддерживает многопоточность, что позволяет приложению эффективно выполнять несколько задач одновременно. AWT (Abstract Window Toolkit) и Swing - это аналогичные технологии для разработки пользовательских интерфейсов Java на платформе Windows.

Также в ходе разработки планируется использовать внешние библиотеки: Math.Numerics, Google.OrTools, Gmap.NET, Itinero, Yandex.Geocoder [8]. Для установки вбранных пакетов использовался менеджер пакетов NuGet.

2.3.3 Math.Numerics

Модуль линейной алгебры в Math.Numerics предоставляет набор классов и методов для работы с линейной алгеброй в .NET. Этот модуль включает в себя реализации методов решения линейных уравнений, нахождения определителей и обратных матриц, а также методов нахождения собственных значений и векторов.

В модуле линейной алгебры в Math.Numerics доступны следующие классы:

- `Matrix<T>`: класс, представляющий матрицы произвольной размерности и произвольного типа данных. Этот класс реализует стандартные операции линейной алгебры, такие как умножение матриц, нахождение определителя и обратной матрицы, решение линейных уравнений и т.д.;
- `Vector<T>`: класс, представляющий векторы произвольного размера и произвольного типа данных. Этот класс также реализует стандартные операции линейной алгебры, такие как скалярное произведение, нахождение нормы вектора и т.д.;
- `QrDecomposition<T>`: класс, реализующий QR-разложение матрицы. QR-разложение используется для решения линейных систем уравнений и нахождения собственных значений матрицы;
- `EigenvalueDecomposition<T>`: класс, реализующий разложение на собственные значения и собственные векторы. Это разложение используется для нахождения собственных значений и собственных векторов матрицы.
- Модуль линейной алгебры в Math.Numerics также содержит реализации стандартных алгоритмов линейной алгебры, таких как LU-разложение, метод Гаусса, метод Жордана-Гаусса и т.д.

Данный модуль планируется использовать для работы с матрицами и векторами, в частности для решения уравнения регрессии методом наименьших квадратов.[9]

2.3.4 Google.OrTools

Оптимальное решение задачи коммивояжера может быть найдено с помощью алгоритмов, использующих различные методы поиска: жадные, динамические, метаэвристические и эволюционные.

В Google.Ortools решение задачи коммивояжера осуществляется с использованием решателя TSP (Traveling Salesman Problem), который решает задачу точно или приближенно, в зависимости от параметров задачи. Для решения задачи коммивояжера в Google.Ortools рекомендуется использовать решатель TSP для малых и средних размеров графов, а для больших графов рекомендуется использовать приближенный алгоритм. [12]

2.3.5 GMap.NET

GMap.NET - это бесплатная библиотека для работы с картами в приложениях на платформе .NET. GMap.NET доступен как для Windows Forms, так и для WPF. GMap.WindowsForms - это часть GMap.NET, которая предназначена для работы с картами в Windows Forms.

GMap.NET поддерживает множество слоев карты. В разрабатываемом приложении нужно использовать OpenStreetMap. Каждый слой обеспечивает свои данные, которые могут быть загружены в приложение и отображены на карте. Пользователь может масштабировать карту, перемещать ее и выполнять поиск по ее содержанию.

Интерфейс карты доступен через класс GMapControl, который можно добавить на форму Windows. Этот класс предоставляет множество методов и свойств для работы с картой. Некоторые из наиболее используемых методов включают в себя установку центра карты, добавление маркера, построение маршрута, установку масштаба и поворота карты. [7]

Кроме того, GMap.NET поддерживает множество событий, которые могут быть использованы для отслеживания взаимодействия пользователя с картой. Некоторые из этих событий включают в себя событие щелчка на маркере, событие изменения масштаба карты и событие изменения центра карты.

В данном случае, планируется с помощью библиотеки загрузить карту

города Пензы, на которой будут отображены точки водоматов и поставщиков. Добавления новой точки будет происходить за счёт обработки события двойного клика мыши по карте. Точки будут помечены соответствующими маркерами. Над маркером будет отображён полный адрес точки, полученный при помощи Yandex.Geocoder. Также будет доступна возможность масштабирования карты.

Планируется отображать маршруты транспортировки, при помощи данного map-контроллера.

GMap.NET – это мощная библиотека для работы с картами в приложениях на платформе .NET. Она обеспечивает широкие возможности по работе с различными слоями карты, геокодерами и событиями. GMap.WindowsForms - это важная часть GMap.NET, которая позволяет легко встраивать карты в приложения на Windows Forms.

2.3.6 Itinero

Itinero - это библиотека и инструмент для построения оптимальных маршрутов между точками на карте, основываясь на данных о дорожной сети и транспортных средствах.

Она использует сложные алгоритмы маршрутизации для расчета кратчайшего и быстрого маршрута в зависимости от заданных параметров, таких как расстояние, время, наличие транспортных пересадок и других факторов.

Itinero получает данные из различных источников, таких как OSM (OpenStreetMap), и предоставляет API для интеграции с другими приложениями и средствами. Она также предоставляет инструменты для настройки и управления профилями транспорта, что позволяет оптимизировать маршруты для различных типов транспорта, включая автомобили, велосипеды, пешеходов и т.д.

Itinero нашла применение в различных областях, таких как транспортная логистика, геомаркетинг, маршрутизация доставки и других сферах, где необходимо оптимизировать перемещения по дорогам. [6]

В нашем приложении будет загружен OSM-файл маршрутов для

автомобиля, на основе которого будет построен маршрут между водоматами, исходящий и возвращающийся в точку поставщика. Полученный маршрут будет отображён при помощи GMap.NET.WindowsForms.

2.4 Проектирование интерфейса

Интерфейс ПО должен быть простым и интуитивно понятным, используя знакомые элементы, которые знакомы большинству пользователей. Элементы интерфейса не должны изменять стандартную логику работы. Основные блоки информации должны быть ясно обозначены и разделены по логическим группам. Взаимодействие с элементами должно соответствовать заданным функциям и возможностям приложения.

Для достижения оптимальной удобства использования, необходимо учитывать многозадачность, возможность работы со многими окнами и размещения нескольких элементов интерфейса на экране одновременно. Навигация по интерфейсу должна быть быстрой и легкой для понимания.

Для улучшения визуального представления мы разработаем макет, который послужат основой для реализации интерфейса. Макет интерфейса главного окна представлен на Рисунке 13 ниже.



Рисунок 13 – Макет интерфейса главного окна

Данный интерфейс содержит логотип ПО в верхнем левом углу экрана. Меню с возможностью выбора активной вкладки и её демонстрация в рабочей области приложения.

Согласно изображениям, интерфейс приложения разбивают информацию на отдельные блоки, что позволяет пользователям быстро ориентироваться в ней.

Также спроектируем макет интерфейса вкладки добавления нового водомата в базу данных (Рисунок 14).

Рисунок 14 – Макет интерфейса окна добавления водомата

На данном макете мы можем наблюдать поля ввода параметров водомата, а также кнопку, которая позволяет открыть карту для выбора метки.

Нижняя панель содержит кнопки «Добавить» и «Отменить». Подобная модель будет реализована и в других формах приложения.

2.5 Планирование и расчёт бюджета, разрабатываемого ПО

2.5.1 Календарное планирование и список ресурсов

Планирование является важной частью разработки автоматизированных систем. Оно включает календарное и ресурсное планирование проекта, которые позволяют управлять работами по созданию информационной системы.

Календарное планирование проекта включает определение длительности работ и их взаимосвязей, а диаграмма Ганта графически отображает сроки работ, продолжительность этапов, список требуемых ресурсов и т.д. Составление ресурсной модели проекта также входит в календарное планирование, где ресурсы делятся на трудовые, материальные и невозобновляемые. Ресурсы оказывают влияние на затраты, которые и составляют бюджет проекта. Для получения плана-графика проекта с оптимальным использованием ресурсов и минимальными сроками проводится календарное планирование. Составим план стадий и этапов создания ПО с подробной информацией о наименовании, длительности стадий и этапов разработки, а также о лицах, выполняющих их. План представлен в Приложении Г. Внесём информацию о затраченных ресурсах (Рисунок 15).

Название ресурса	Тип	Краткое название	Стандартная ставка	Ставка сверхурочн	Макс. единиц	Группа
Руководитель проекта	Трудовой	Р	650,00р./час	800,00р./час	100%	Сотрудники
Менеджер проекта	Трудовой	М	400,00р./час	500,00р./час	100%	Сотрудники
Разработчик	Трудовой	Р	450,00р./час	550,00р./час	100%	Сотрудники
Тестирующий	Трудовой	Т	350,00р./час	400,00р./час	100%	Сотрудники
Аналитик	Трудовой	А	400,00р./час	450,00р./час	100%	Сотрудники
Проектировщик	Трудовой	П	350,00р./час	500,00р./час	100%	Сотрудники
Компьютер	Материальный	К	200 000,00р.			Техника
Интернет	Материальный	И	1 000,00р.			
Проектер	Материальный	П	50 000,00р.			Техника
Принтер	Материальный	П	30 000,00р.			Техника
Бумага	Материальный	Б	500,00р.			Материалы

Рисунок 15 – Список ресурсов проекта

После анализа этапов, их продолжительности и взаимосвязей, можно заключить, что для проектирования и разработки программного продукта потребуется 182 часа.

2.5.2 Расчёт бюджета

Проведем расчет стоимости проектирования и разработки проекта. Для расчета затрат необходимо учесть следующие статьи расходов:

- амортизация;

- материалы;
- фонды основной и дополнительной заработной платы;
- социальные выплаты;
- прочие затраты.

Амортизация

Амортизация – это процесс перенесения по частям стоимости изношенной части основных производственных фондов (ОПФ) на стоимость готовой продукции.

Стоимость ОПФ и расчет амортизационных отчислений приведены в Таблице 11 ниже. Обозначения в таблице:

- СПИ: срок полезного использования;
- НА: норма амортизации;
- А: амортизация;
- ТЗР: транспортно-заготовительные расходы.

Таблица 11 – Амортизационные отчисления

Оборудование	Стоимость,руб.	СПИ, лет	НА, %	А, год, руб.
Компьютер с ПО:MS Office; MS Windows 11	200 000	5	20	40 000
Принтер Canon	30 000	5	20	6 000
Проектор Benq	50 000	5	20	10 000
ТЗР (1%)	850			
Итого	280 850			56 000

Для разработки программного продукта планируется использовать программное обеспечение в свободном доступе: Visual Studio 2019, MS SQL Server, GMapNet – контроллер карты, HTTP-геокодер от Yandex.

Амортизационные отчисления рассчитываются по следующей формуле:

$$A = C_{\text{пер}} * \text{НА} \quad (3)$$

где:

A – амортизационные начисления, руб.;

$C_{\text{пер}}$ – первоначальная стоимость основных средств, руб.;

НА – норма амортизации, которая находится по формуле:

$$\text{НА} = 1 / \text{СПИ}$$

где:

СПИ – срок полезного использования основных средств.

Срок полезного использования компьютера = 5 лет

$$\text{НА}_{\text{комп}} = 1 / 5 = 0,2$$

Срок полезного использования принтера = 5 лет

$$\text{НА}_{\text{принт}} = 1 / 5 = 0,2$$

Срок полезного использования проектера = 5 лет

$$\text{НА}_{\text{проект}} = 1 / 5 = 0,2$$

$$A_{\text{комп}} = 200\,000 * 0,2 = 40\,000 \text{ руб.}$$

$$A_{\text{принт}} = 30\,000 * 0,2 = 6\,000 \text{ руб.}$$

$$A_{\text{прjtm}} = 50\,000 * 0,2 = 10\,000 \text{ руб.}$$

$$A_r = 40\,000 + 6\,000 + 10\,000 + 850 = 56\,850 \text{ руб.}$$

$$A_m = 56\,850 / 12 = 4737,5 \text{ руб/мес.}$$

$$A_{\text{итог}} = 2379,2 * 1 = 4737,5 \text{ руб.}$$

Материалы

Работа по созданию программного продукта предполагается вести 182 часа, расчёт ведётся за 1 месяц.

В данной статье приведены расчеты оборотных средств, использовавшиеся для разработки программного продукта, они приведены в Таблице 12 ниже.

Таблица 12 – Затраты по статье «Материалы»

Наименование	Количество, шт.	Сумма, руб.
Бумага	2	1000
ТЗР (1%)		10
ИТОГО:		1010

Фонд основной и дополнительной заработной платы

Зарплата – вознаграждение за труд в зависимости от квалификации работника, сложности, количества, качества и условий выполняемой работы.

Трудоемкость всех видов работ, связанных с разработкой рассматриваемого программного продукта представлена в Таблице 13.

Таблица 13 – Расчет основной и дополнительной заработной платы

Исполнитель	Трудоемкость, часов (Т _р)	Ставка, руб./час	ОЗП, руб.	ДЗП, руб.	Соц. выплаты
Руководитель проекта	22,47	650,00	14 604,17	1 460,41	4 819,38
Менеджер проекта	23,88	400,00	9 552,40	955,24	3 152,29
Разработчик	28,25	450,00	12 712,50	1 271,25	4 195,13
Тестировщик	15,75	350,00	5 512,50	551,25	1 819,13
Аналитик	52,02	400,00	20 807,18	2 080,71	6 866,37
Проектировщик	39,58	350,00	13 856,82	1 385,68	4 572,75
Всего:	181,95	2600,00	77 045,57	7 704,56	25 425,05

На разработку и тестирование программного продукта затрачено 376 часов.

Рассмотрим оплату труда на примере расчета заработной платы руководителя проекта.

Оплата за труд рассчитывается следующим образом:

$$\text{ОЗП} = C_{\text{ч}} T_{\text{р}} \quad (4)$$

где:

$T_{\text{р}}$ – трудоемкость.

$C_{\text{ч}}$ – тарифная ставка, которая рассчитывается по формуле:

$$C_{\text{ч}} = (\text{МРОТ} * T_{\text{к}}) / 160 \quad (5)$$

где:

МРОТ – минимальный размер оплаты труда = 12 130 руб.;

$T_{\text{к}}$ – тарифный коэффициент;

160 – количество часов в месяц по ТК РФ.

$$T_{\text{рук}} = 8,57$$

$$C_{\text{рук}} = (12\,130 * 8,57)/160 = 650$$

$$OЗП_{\text{рук}} = 650 * 22,47 = 14\,604,17 \text{ руб.}$$

Мотивация труда оценивается через дополнительную заработную плату, которая рассчитывается по формуле:

$$ДЗП = 10\% OЗП \quad (6)$$

$$ДЗП_{\text{рук}} = 0,1 * 14\,604,17 = 1\,460,41 \text{ руб.}$$

Социальные выплаты

Социальные выплаты состоят из:

- Фонд социального страхования (ФСС) – 5,9%;
- Пенсионный фонд РФ (ПФР) – 22%;
- Фонд обязательного медицинского страхования (ФОМС) – 2,1%.

$$СВ = 30\%(OЗП + ДЗП) \quad (7)$$

$$СВ_{\text{рук}} = 30\% * (14\,604,17 + 1\,460,41) = 4819,38 \text{ руб.}$$

Прочие

В этой статье приведены затраты на доступ в Интернет и Электроэнергия (ЭЭ)

ЭЭ рассчитывается по формуле:

$$ЭЭ = T_c * W * T_{\text{раб}} \quad (8)$$

где:

T_c = стоимость 1 кв/ч = 4,24 руб.;

W – энергомощность компьютера = 450 Вт/ч = 0,45 кВт/ч;

$T_{\text{раб}}$ – время создания, т.к. программный продукт разрабатывался 182 часа.

$$ЭЭ = 4,24 * 0,45 * 182 = 347,26 \text{ руб.}$$

Стоимость услуги доступа в Интернет:

Работа над программным продуктом велась 182 часа, тариф за услугу доступа в Интернет составляет = 1000 руб./мес. Однако, провайдер не предоставляет почасовой доступ в интернет, поэтому расчет ведется за 1 месяц.

$$\text{Интернет} = 1000 * 1 = 1000 \text{ руб.}$$

$$\text{Прочие затраты} = ЭЭ + \text{Интернет} = 347,26 + 1000 = 1347,26 \text{ руб.}$$

Структура затрат представлена в Таблице 14.

Таблица 14 – Структура затрат

Наименование статей	Затраты, руб.	%
Амортизация	4737,5	4,04%
Материалы	1 010	0,86%
Фонд основной и дополнительной ЗП	84 750,13	72,27%
Социальные выплаты	25 425,05	21,68%
Прочие	1 347,26	1,15%
Затраты на проект	117269,94	

Диаграмма структуры затрат представлена на Рисунке 15 ниже.

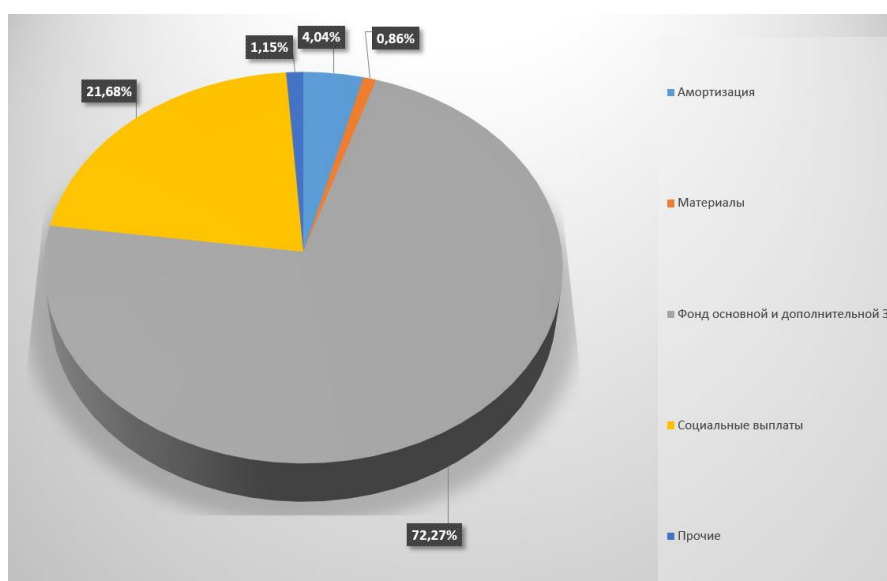


Рисунок 15 – Структура затрат

Проводя анализ структуры расходов можно сделать вывод, что основными статьями расходов является фонд оплаты труда и социальные выплаты. Затраты на проектирование и разработку ПО для анализа и прогнозирования состояния сети водоснабжения составят 117269,94 рублей.

3 Реализация ПО

Разработка велась на языке C# версии 8.0 и платформе .NET 4.8 в интегрированной среде разработки Visual Studio 2019. Разработанное приложение использует библиотеку WinForms для создания графического интерфейса приложения. Архитектура приложения многослойна, реализована с помощью паттерна MVP. Приложение содержит три компонента, которые хранят в себе файлы с реализацией классов.

Код разработанного программного обеспечения приведён в приложении А.

На Рисунке 16 показана укрупнённая диаграмма компонентов для разработанной программы.

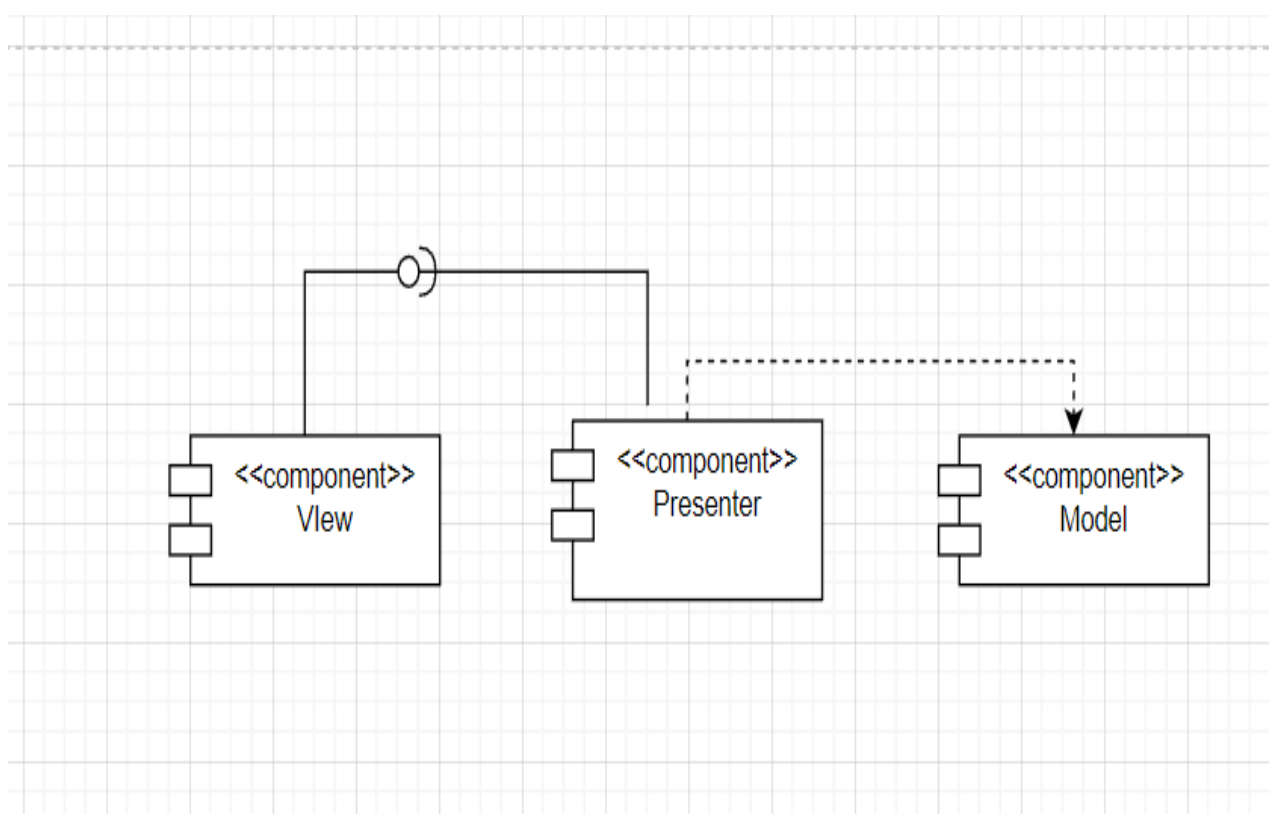


Рисунок 16 – Укрупнённая диаграмма компонентов

В Таблице 15 ниже показаны: название компонентов, изображенных на Рисунке 16, описание их компонентов, а также элементы, содержащиеся внутри них.

Таблица 15 – Назначение элементов укрупнённой диаграммы компонентов

Комонент	Назначение	Содержит элементы
View	Представление (View) содержит классы и файлы, описывающие графический интерфейс программы.	IAddCarView.cs, IAddMarkerView.cs, IAddSupplierView.cs, IAddWaterMachineView.cs, IAnalysisView.cs, ICarsView.cs, IHomeView.cs, IMainView.cs, IMapView.cs, IRegressView.cs, ISuppliersView.cs, ITransactionView.cs, IView.cs, IWaterSensorView.cs, IWaterMachineView.cs, AddCarForm.cs, AddMarkerForm.cs, AddSupplierForm.cs, AddWaterMachineForm.cs, AnalysisForm.cs, ARModelForm.cs, CarsForm.cs, HomeForm.cs, InfoForm.cs, LoadForm.cs, MainForm.cs, MapForm.cs, RegressForm.cs, RouteForm.cs, SupplierForm.cs, TransactionForm.cs, WaterMachineForm.cs, WaterSensorForm.cs
Presenter	Presenter содержит классы, которые связывают представление и модель через механизм привязки данных.	AddCarPresenter.cs, AddMarkerPresenter.cs, AddSupplierPresenter.cs, AddWaterMachinePresenter.cs, CarsPresenter.cs, AnalysisPresenter.cs, HomePresenter.cs, IPresenter.cs, MainPresenter.cs, MapPresenter.cs, SuppliersPresenter.cs, TransactionPresenter.cs, WaterMachinePresenter.cs, WaterSensorPresenter.cs
Model	Модель (Model) содержит классы, описывающие таблицы контекста БД	Car.cs, PurchaseLog.cs, Supplier.cs, WaterLevelSensor.cs, WaterVendingMachine.cs, IDatabaseContext.cs, ModelContext.cs

Также реализованы 3 вспомогательных класса: «TSP» – класс, содержащий в себе модель, позволяющий решить задачу коммивояжера, а также два класса – содержащих в себе регрессионные модели.

3.1 Кодирование ПО

Архитектура MVP позволяет максимально изолировать серверную и клиентскую части приложения, рассмотрим каждый из компонентов ПО по отдельности.

Entity – классы сущностей БД:

- Car.cs – класс, который представляет сущность автомобиля в системе и определяет его основные свойства: марка, модель, номер и поставщик, а также связь с классом Supplier. Он используется в контексте базы данных для создания таблицы Car, в которой будут храниться данные об автомобилях. Класс также имеет конструктор, который позволяет создавать экземпляры объектов автомобилей;

- Supplier.cs – класс, который представляет сущность поставщика (компаний-поставщика) в системе и определяет его свойства: название компании, адрес и координаты. Его используют в контексте базы данных для создания таблицы Supplier, где будут храниться данные о компаниях-поставщиках, а также для установления связей между таблицами Supplier и Car и Supplier и WaterVendingMachine. Класс также имеет конструктор, который позволяет создавать экземпляры объектов поставщиков;

- PurchaseLog.cs – класс, который предназначен для хранения информации о покупках воды в торговых автоматах. Он содержит данные о конкретной покупке, такие как дата, объем, стоимость, а также текущий баланс воды в автомате. Эти данные могут быть использованы для анализа и оптимизации производственных процессов, управления запасами и мониторинга производительности торговых автоматов;

- WaterVendingMachine.cs – класс, который предназначен для хранения информации о торговых автоматах, которые предоставляют доступ к питьевой воде. Он содержит данные об адресе места установки автомата, ёмкости резервуара, цене за литр воды, координатах месторасположения, дате последнего обслуживания и статусе автомата. Также класс содержит информацию о покупках воды (класс PurchaseLog), уровне воды в резервуаре (класс WaterLevelSensor) и поставщиках, обслуживающих автомат (класс Supplier). Эти данные могут быть использованы для управления запасами воды, улучшения производительности автоматов и оптимизации обслуживания;

- WaterLevelSensor.cs – класс, который предназначен для хранения

информации о состоянии датчиков уровня воды в торговых автоматах. Он содержит данные об остатке воды в резервуаре, статусе датчика и модели используемого датчика. Эти данные могут быть использованы для обслуживания автомата, контроля уровня воды в резервуаре, предотвращения простоев автомата из-за нехватки воды и мониторинга состояния оборудования.

Data – класс контекста БД:

- ModelContext.cs – класс, который реализует интерфейс `IDatabaseContext` и предоставляет методы для управления и взаимодействия с базой данных. `IDatabaseContext` определяет контракт для контекста базы данных и требует наличия набора свойств `DbSet` для доступа к соответствующим таблицам базы данных. Это позволяет использовать класс `ModelContext` в качестве контекста базы данных для выполнения операций CRUD (Create, Read, Update, Delete) с объектами сущностей и связей в базе данных.

Presenters – классы презентеров:

- AddCarPresenter.cs – класс, который предназначен для управления добавлением автомобилей в базу данных. Он принимает объекты `IAddCarView` для отображения формы добавления автомобилей и `IDatabaseContext` для работы с базой данных. Класс включает в себя обработчики событий клика на кнопки добавления и удаления, а также использует регулярное выражение для проверки правильности номера автомобиля и проверяет ограничения на ввод данных в других полях формы. После проверки данных, он создает новый объект `Car`, заполняет его данными с формы, связывает его с объектом `Supplier` и сохраняет изменения в базу данных. Класс также реализует `IDisposable` для удаления обработчиков событий при уничтожении объекта;

- AddMarkerPresenter.cs – класс, который предназначен для управления добавлением маркера на карту. Он принимает объекты `IAddMarkerView` для отображения формы добавления маркера и `IDatabaseContext` для работы с базой данных. Класс включает в себя обработчик события клика на кнопку добавления, который сохраняет координаты места установки маркера и связанный с ним объект (торговый автомат или поставщик) в базу данных. Класс также реализует

IDisposable для удаления обработчиков событий при уничтожении объекта;

- AddSupplierPresenter .cs – класс, который отвечает за презентационную логику формы добавления нового поставщика. Он связывает интерфейс и базу данных, обрабатывает действия пользователя и создает новый объект Supplier, который добавляется в базу данных. Класс также управляет отображением формы добавления маркера на карте, который используется для задания координат поставщика;

- AddWaterMachinePresenter.cs – класс, который отвечает за презентационную логику формы добавления нового водомата. Он связывает интерфейс и базу данных, обрабатывает действия пользователя и создает новый объект WaterVendingMachine и связанный с ним объект WaterLevelSensor, которые добавляются в базу данных. Класс также управляет отображением формы добавления маркера на карте, который используется для задания координат водомата. Когда новый водомат добавляется в базу данных, его статус "Status" заполняется на основе состояния датчика уровня воды, а также они связываются между собой;

- AnalysisPresenter .cs – класс, который отвечает за презентационную логику формы анализа продаж воды. Он связывает интерфейс и базу данных, обрабатывает действия пользователя и предоставляет данные о продажах воды и водоматах для анализа. Класс осуществляет запросы к базе данных для получения информации о количестве продаж и доходе в зависимости от даты и выбранного водомата, а также управляет отображением информации на форме. Класс также отслеживает состояние базы данных и обновляет информацию на форме при изменении данных;

- CarsPresenter .cs – класс, который отвечает за презентационную логику формы управления автомобилями. Он связывает интерфейс и базу данных, обрабатывает действия пользователя, добавляет и удаляет объекты Car в базу данных, загружает данные из базы данных на форму и предоставляет пользователю функционал управления данными об автомобилях. Класс реализует интерфейс IPresenter и реализует методы Dispose(), а также определяет

свойство `View`, которое возвращает интерфейс `IView`, связанный с формой;

- `HomePresenter.cs` – класс, который отвечает за презентационную логику главной формы (домашней страницы) приложения. Он не имеет дополнительной функциональности, а просто предоставляет доступ к главной форме. Этот класс реализует интерфейс `IPresenter` и метод `Dispose()`, но не содержит дополнительной логики. Он лишь отображает заставку с информацией о приложении;

- `IPresenter.cs` – интерфейс, который определяет общие свойства и методы для всех презентеров в приложении. `IPresenter` предоставляет свойство `View`, которое возвращает интерфейс `IView`, который соответствует соответствующей форме в приложении. Класс также реализует интерфейс `IDisposable`, который позволяет освободить ресурсы, используемые презентером, при завершении работы. Этот интерфейс предназначен для облегчения механизма управления памятью и удаления объектов в `.NET Framework`;

- `MainPresenter.cs` – класс-презентер, который отвечает за управление взаимодействием между главным окном программы (`IMainView`) и её подокнами. Класс также инициализирует контекст базы данных (`ModelContext`) и управляет созданием и отображением подокон в соответствии с выбранным пользователем функционалом;

- `MapPresenter.cs` – данный класс предназначен для управления окном карты приложения: инициализации окна, загрузки данных из базы данных, отображения меток на карте и закрытия окна;

- `SupplierPresenter.cs` – данный класс предназначен для управления окном управления поставщиками: загрузки списка поставщиков из базы данных, отображения списка, добавления новых поставщиков и удаления существующих. Класс также отвечает за закрытие соединения с базой данных;

- `TransactionPresenter.cs` – данный класс предназначен для управления окном управления транзакциями: генерации тестовых данных, загрузки и отображения списка транзакций из базы данных и удаления обработчиков событий. Класс также используется для расчета объема продаж и их записи в

базу данных;

- `WaterMachinePresenter.cs` – данный класс предназначен для управления окном управления водоматов: загрузки списка машин из базы данных, отображения списка, добавления новых машин, удаления существующих машин и управления датчиком воды в резервуаре водомата. Класс также отвечает за закрытие соединения с базой данных;

- `WaterSensorPresenter.cs` – это презентер, который отвечает за связь между моделью данных и представлением интерфейса пользователя для работы с датчиками уровня воды. Он обрабатывает события, генерируемые пользовательским интерфейсом, и обновляет данные в модели в соответствии с этими событиями. Также он управляет жизненным циклом представления и ресурсами, связанными с его работой.

`Forms` и `Interfaces` – классы форм и их интерфейсов соответственно

`RouteOptimization` – класс оптимизации маршрута:

- `TSP.cs` – класс, который отвечает за оптимизацию маршрута доставки воды с использованием метода "Traveling Salesman Problem" (TSP). Он содержит методы для создания матрицы расстояний между поставщиком и торговыми автоматами, расчета оптимального маршрута по этой матрице и вывода полученного результата в виде списка узлов маршрута. TSP используется для расчета наиболее оптимального маршрута, который позволяет снизить затраты на доставку воды и оптимизировать работу логистических систем в целом.

`MLRegressor` – классы-модели регрессии:

- `AutoRegression` – класс, который отвечает за обучение и применение авторегрессионной модели для прогнозирования временных рядов. Он содержит методы для обучения модели на заданном временном ряде, предсказания значений на основе модели и инициализации модели перед обучением. Класс использует библиотеку `MathNet.Numerics` для работы с математическими вычислениями. Он может быть использован для прогнозирования временных рядов, например, для прогнозирования количества воды, используемой в автоматах за определенный период;

– `MultipleLinearRegressor` – класс, который предназначен для обучения и использования модели множественной линейной регрессии, которая позволяет прогнозировать зависимую переменную на основе нескольких независимых переменных. Он содержит методы для обучения модели на наборе данных, предсказания значений на основе модели и расширения данных, добавляя входной вектор с единицами, что позволяет учитывать свободный член в модели. Класс использует библиотеку `MathNet.Numerics` для работы с математическими вычислениями. Он может быть использован для прогнозирования объема продаж воды на основе нескольких факторов, таких как цена, время и местоположение, что позволит оптимизировать производственные процессы и маркетинговые стратегии.

3.2 Руководство для пользователя

ПО прогнозирования и анализа состояния сети водоснабжения направлено обеспечить автоматизацию работы персонала. Текущая версия ПО – учебная, в будущем возможен переход на коммерческую основу. Для запуска консольной программы достаточно иметь компьютер, работающий на ОС Windows с установленным интерпретатором языка C# версии 8.0 и .NET v4.8, а также наличие библиотек: `Math.Numeric`, `GMap.Net`, `Google.OrTools`, `Itinero`, `Yandex.Geocoder`, `LiveCharts`, `EntityFramework`, `WindowsForms` а также другие стандартные библиотеки из набора .NET. Для корректной работы программы необходимо иметь доступ в Интернет (API-запрос к http-геокодеру), а также доступ к базе данных MS SQL Server.

Для запуска настольного графического приложения необходимо запустить `Diplom.exe`, после запуска модуля, если данные для работы программы не повреждены, на экране должно появиться стартовое окно приложения, как это изображено на рисунке 17.

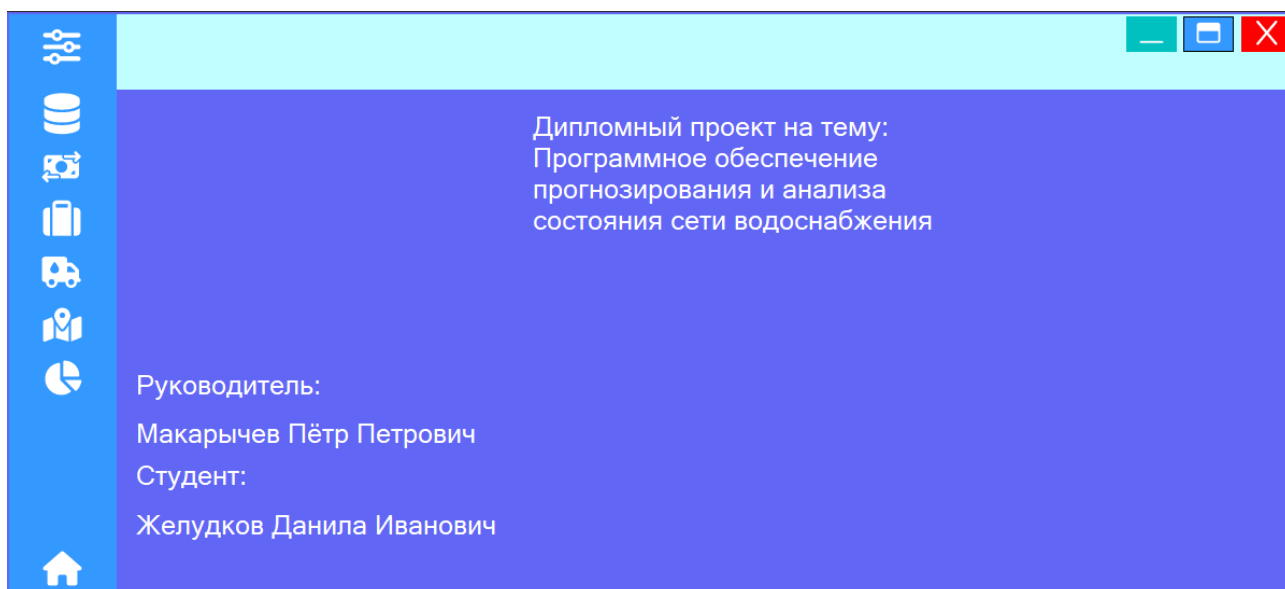


Рисунок 17 – Домашняя страница приложения

Мы можем развернуть пункты меню, чтобы ознакомиться с навигацией приложения (рисунок 18).

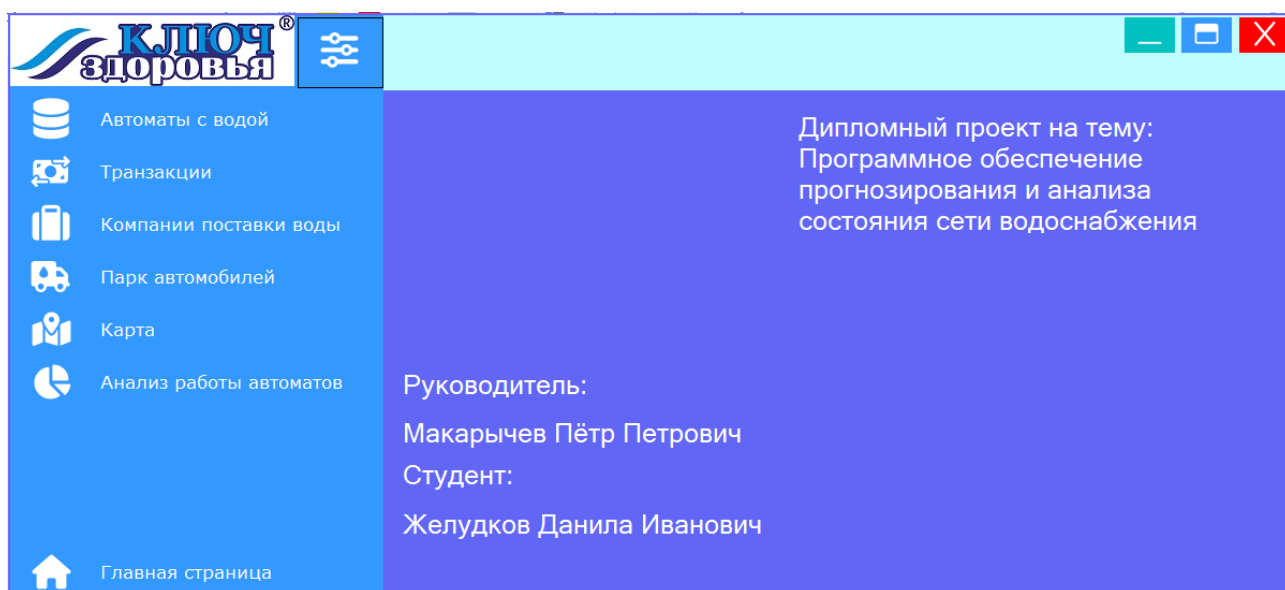


Рисунок 18 – Навигация приложения

Продemonстрируем реализацию предъявленных функциональных требований.

Нажимаем на кнопку «Автоматы с водой», осуществляется переход в раздел, фон рабочей области перерисовывается. Затем кликаем на кнопку «+», интерпретирующую функцию добавления нового водомата. Переходим в форму

добавления (рисунок 19).

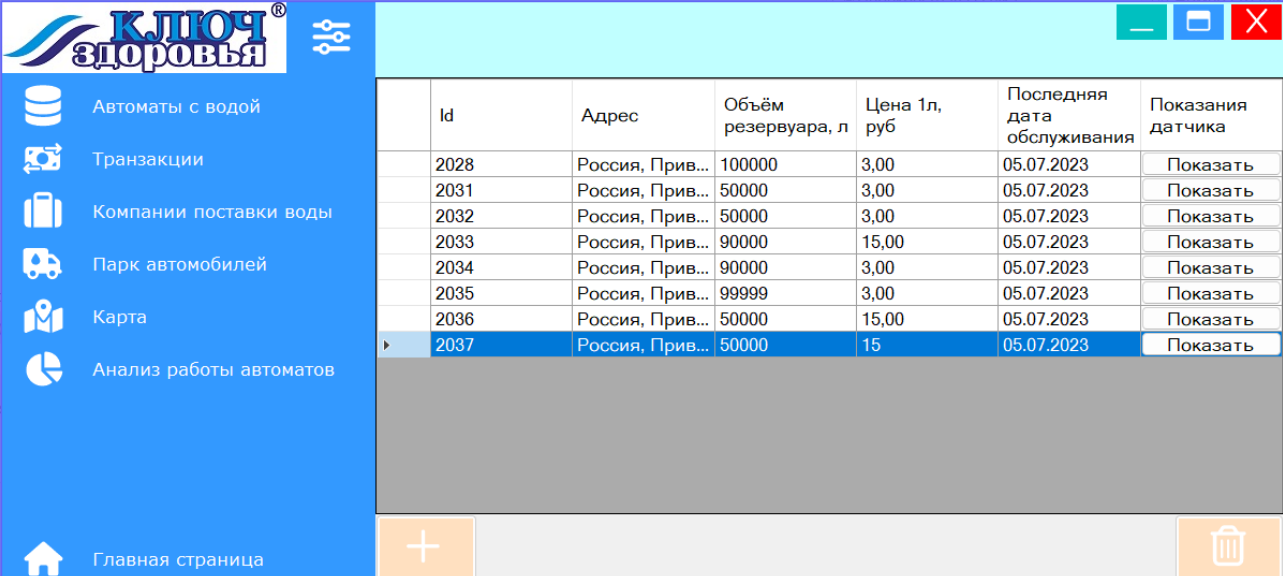
Рисунок 19 – Форма добавления водомата

Заполняем данные, жмём кнопку «Выбрать на карте», двойным кликом отмечаем метку и нажимаем «X». Далее мы можем спрогнозировать расход добавляемого водомата (рисунок 20).

	Id водомата	Расстояние до центра города (X1)	Цена 1 литра воды (X2)	Среднесуточная выручка (Y)
►	2028	2645,58782914031	3	1072,33586666667
	2031	305,293617876983	3	9480,06061333333
	2032	1729,13274674781	3	1634,78388666667
	2033	3568,55548292022	15	991,76626
	2034	1026,75863140837	3	2769,26360666667
	2035	2894,4826177611	3	997,284893333333
	2036	1037,89168200394	15	3369,23272666667
	New	2741,77715802293	15	1285,4428

Рисунок 20 – Форма с прогнозом среднесуточного дохода

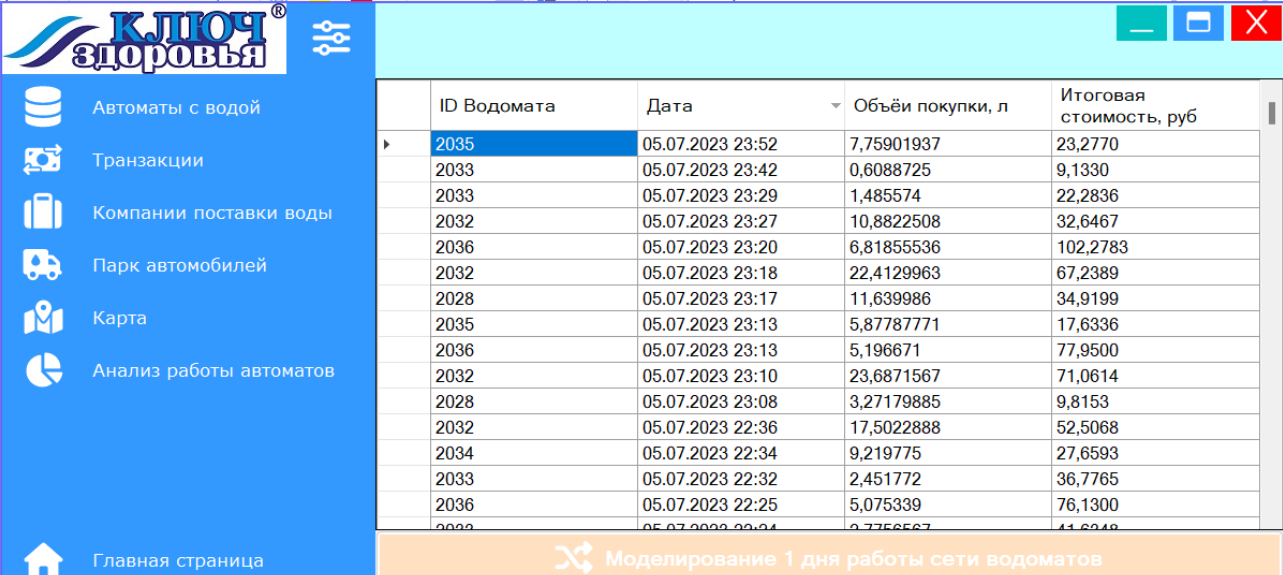
Далее на основе данных мы принимаем решение, добавляя текущий водомат в базу данных. Данные отображаются в таблице (рисунок 21).



Id	Адрес	Объем резервуара, л	Цена 1л, руб	Последняя дата обслуживания	Показания датчика
2028	Россия, Прив...	100000	3,00	05.07.2023	Показать
2031	Россия, Прив...	50000	3,00	05.07.2023	Показать
2032	Россия, Прив...	50000	3,00	05.07.2023	Показать
2033	Россия, Прив...	90000	15,00	05.07.2023	Показать
2034	Россия, Прив...	90000	3,00	05.07.2023	Показать
2035	Россия, Прив...	99999	3,00	05.07.2023	Показать
2036	Россия, Прив...	50000	15,00	05.07.2023	Показать
2037	Россия, Прив...	50000	15	05.07.2023	Показать

Рисунок 21 – Отображение добавленного водомата

После чего мы можем перейти в раздел «Транзакции», нажав, соответствующую кнопку мы смоделировать 1 день работы всей системы. В графе логов будут добавлены данные транзакций всех включенных водоматов (рисунок 22).



ID Водомата	Дата	Объём покупки, л	Итоговая стоимость, руб
2035	05.07.2023 23:52	7,75901937	23,2770
2033	05.07.2023 23:42	0,6088725	9,1330
2033	05.07.2023 23:29	1,485574	22,2836
2032	05.07.2023 23:27	10,8822508	32,6467
2036	05.07.2023 23:20	6,81855536	102,2783
2032	05.07.2023 23:18	22,4129963	67,2389
2028	05.07.2023 23:17	11,639986	34,9199
2035	05.07.2023 23:13	5,87787771	17,6336
2036	05.07.2023 23:13	5,196671	77,9500
2032	05.07.2023 23:10	23,6871567	71,0614
2028	05.07.2023 23:08	3,27179885	9,8153
2032	05.07.2023 22:36	17,5022888	52,5068
2034	05.07.2023 22:34	9,219775	27,6593
2033	05.07.2023 22:32	2,451772	36,7765
2036	05.07.2023 22:25	5,075339	76,1300

Рисунок 22 – Данные транзакций

Далее так же можно включить/отключить водомат, а также проверить

значения датчиков (рисунки 23-24).

	Модель датчика	Текущий уровень воды	Статус
	Bosch-00607138	50000	☒

×

Рисунок 23 – Включение водомата

	Модель датчика	Текущий уровень воды	Статус
	Bosch-00607138	50000	☐

×

Рисунок 24 – Отключение водомата

Аналогично, перейдя в пункт меню «Компании поставки воды» мы можем осуществить добавление поставщиков (рисунки 25-26).

Поставщик

Название компании: New Supplier

Адрес: [Выбрать на карте](#)

Рисунок 25 – Заполнение данных о поставщике

Название поставщика	Адрес поставщика
gt5tg	Россия, Приволжский федеральный округ, Пензенская область, городской округ...
Water 24/7	Россия, Приволжский федеральный округ, Пензенская область, городской округ...
Вода без труда	Россия, Приволжский федеральный округ, Пензенская область, городской округ...
324	Россия, Приволжский федеральный округ, Пензенская область, городской округ...
5555	Россия, Приволжский федеральный округ, Пензенская область, городской округ...
New Supplier	Россия, Приволжский федеральный округ, Пензенская область, городской округ...

Рисунок 26 – Отображение данных нового поставщика

Аналогичная операция добавления осуществляется и для парка автомобилей.

Для работы с картой, мы выбираем пункт меню «Карта», здесь мы выбираем иконку «Настройки», где формируем маршрут, также есть функция автоформирования, где будут выбраны точки, которые необходимо обслужить в первую очередь. А также выбираем поставщика и автомобиль из его автопарка. (рисунок 27).

	Id	Последнее обслуживание	Остаток воды	Статус водомата	Выбрать
	2028	30.06.2023	100000	☒	☒
	2031	30.06.2023	50000	☒	☒
	2032	30.06.2023	50000	☒	☒
	2033	30.06.2023	90000	☒	☒
	2034	30.06.2023	90000	☒	☒
	2035	30.06.2023	99999	☒	☒
▶	2036	30.06.2023	50000	☒	☒

+
↻ Автовыбор
Water 24/7
Nissan GTR n123on58

Рисунок 27 – Выбор маршрута

Далее мы можем нажать на кнопку «Проложить маршрут», после чего на карте будет отображён маршрут (рисунок 28). Пользователь может осуществлять движение по точкам маршрута. А также посмотреть маршрута в текстовом формате, нажав на иконку «Info» (рисунок 29). Также есть возможность сохранения картинки маршрута на ПК, для это нужно кликнуть на иконку «Камеры» и указать путь для выбранной директории. Последней функцией в данном разделе будет обслуживание точек маршрута, нажав её в базе обновятся сведения, точки будут обслужены, водоматы включены, а вода в их резервуарах пополнена (рисунок 30).

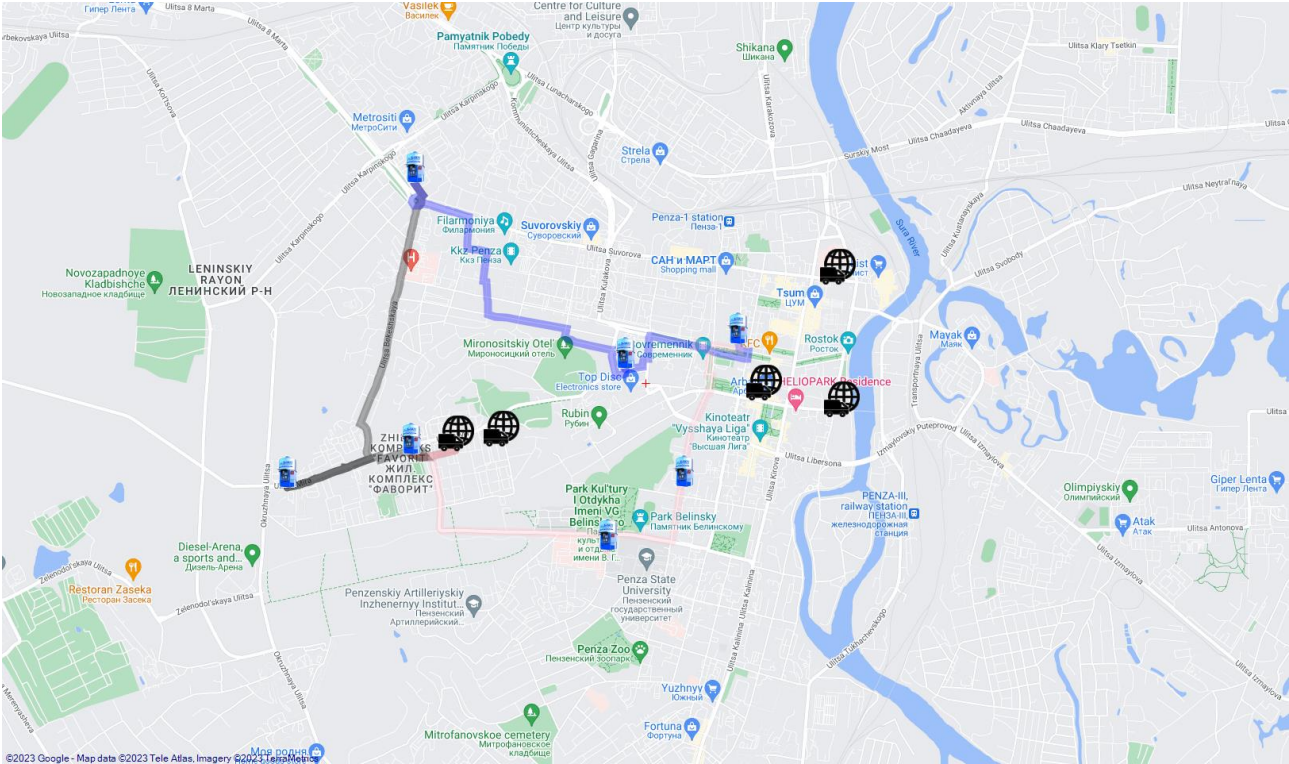


Рисунок 28 – Маршрут движения

Маршрут обслуживания автоматов для продажи воды. Автомобиль - Nissan GTR n123on58
Поставщик Water 24/7 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, Первомайский район, микрорайон Западная Поляна,
---->
Водомат 2028 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица
Мира, 44
Водомат 2028 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица
Мира, 44
---->
Водомат 2033 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица
Мира
Водомат 2033 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица
Мира
---->
Водомат 2035 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица
Карпинского, 38
Водомат 2035 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица
Карпинского, 38
---->
Водомат 2036 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица
Космодемьянской, 1
Водомат 2036 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица
Космодемьянской, 1
---->

Рисунок 29 – Информация о маршруте в текстовом формате

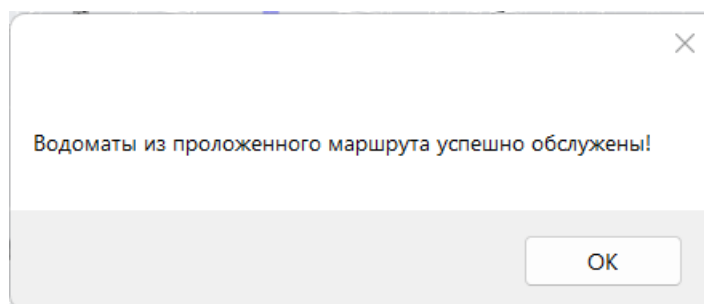


Рисунок 30 – Результат обслуживания водомата

Наконец, последний пункт меню – «Анализ». Здесь мы можем выбрать критерий сравнения и способ отображения сравнения «Таблица» (рисунок 31) или «График» (рисунок 32). А также выбрать подменю «Прогноз», где будет осуществлено прогнозирование расхода водомата, на основе чего можно принять решение, когда осуществлять заправку (рисунки 33 и 34).

ID водомата	Дата	Расход воды
2028	21.06.2023	459,571948
2031	21.06.2023	2803,94807
2032	21.06.2023	691,419497
2033	21.06.2023	82,7019918
2034	21.06.2023	1193,520085
2035	21.06.2023	295,696155
2036	21.06.2023	169,240637
2037	21.06.2023	0

Рисунок 31 – Сравнительный анализ расхода воды. Таблица

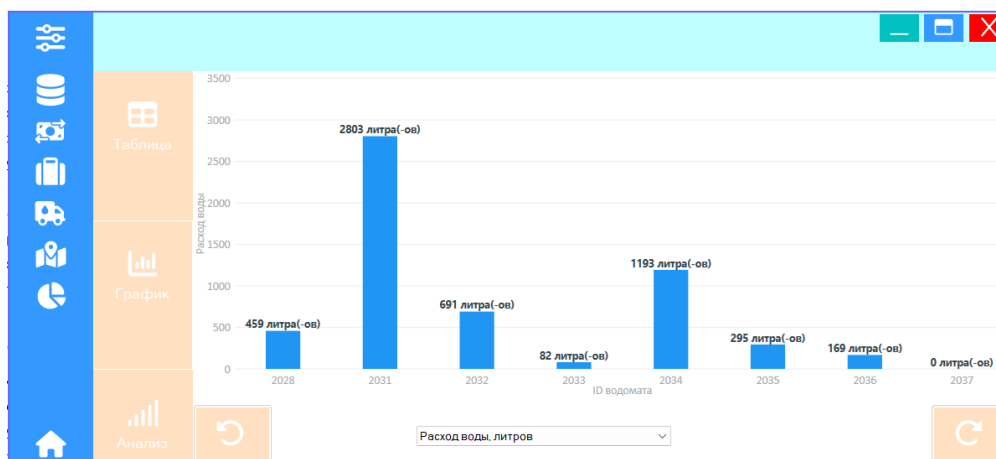


Рисунок 32 – Сравнительный анализ расхода воды. График

	Id водомата	Дата	Расход воды
	2028	26.06.2023	97833,1
	2028	27.06.2023	97514,5
	2028	28.06.2023	97117,7
	2028	29.06.2023	96645,8
	2028	30.06.2023	96385,2
	2028	01.07.2023	96076,1
	2028	02.07.2023	95676,1
	2028	03.07.2023	95205,7
	2028	04.07.2023	94945,5
	2028	05.07.2023	94637,5
	2028	06.07.2023	94268,797839353

√x

2028

X

Рисунок 33 – Таблица анализа

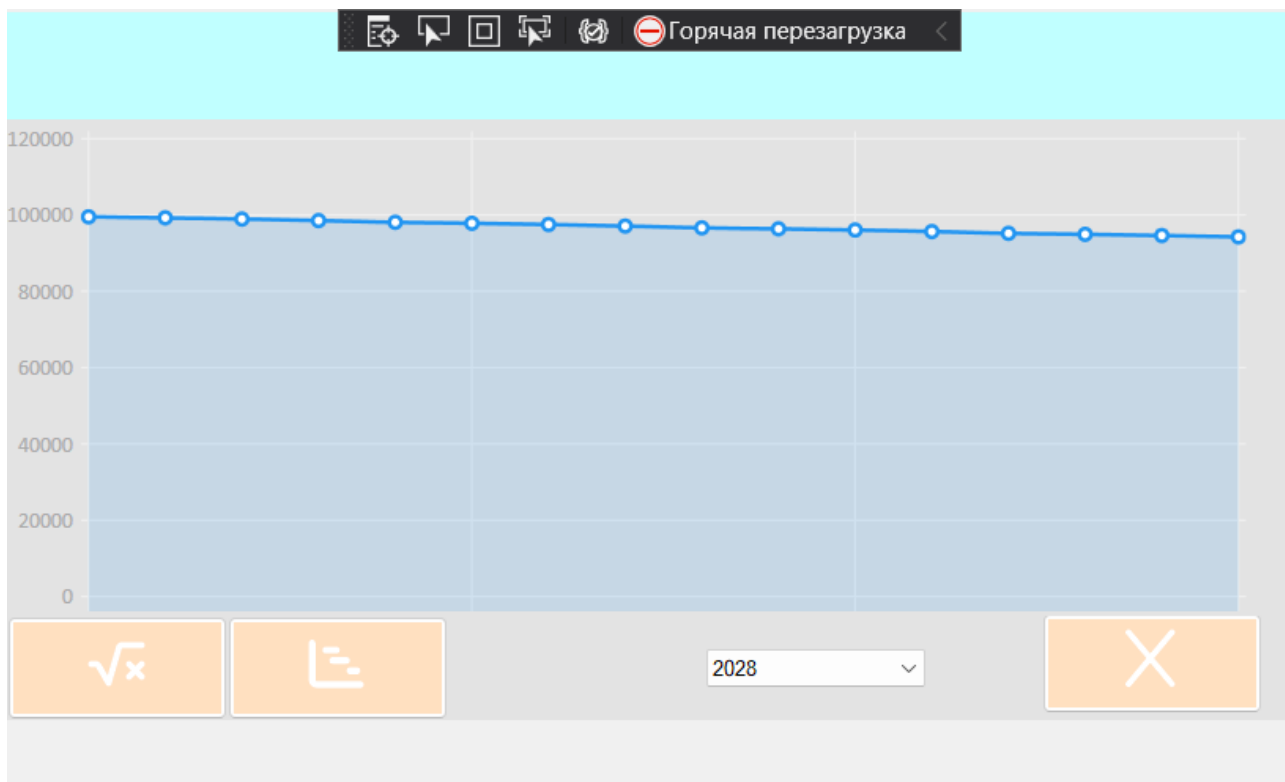


Рисунок 34 – График анализа

Также пользователю доступно удаление строк из базы для этого на любой из вкладок учёта необходимо выделить выбранные строки и нажать на иконку

«Корзины», происходит каскадное удаление, то есть удаляются связанные строки других сущностей (рисунки 35, 36).

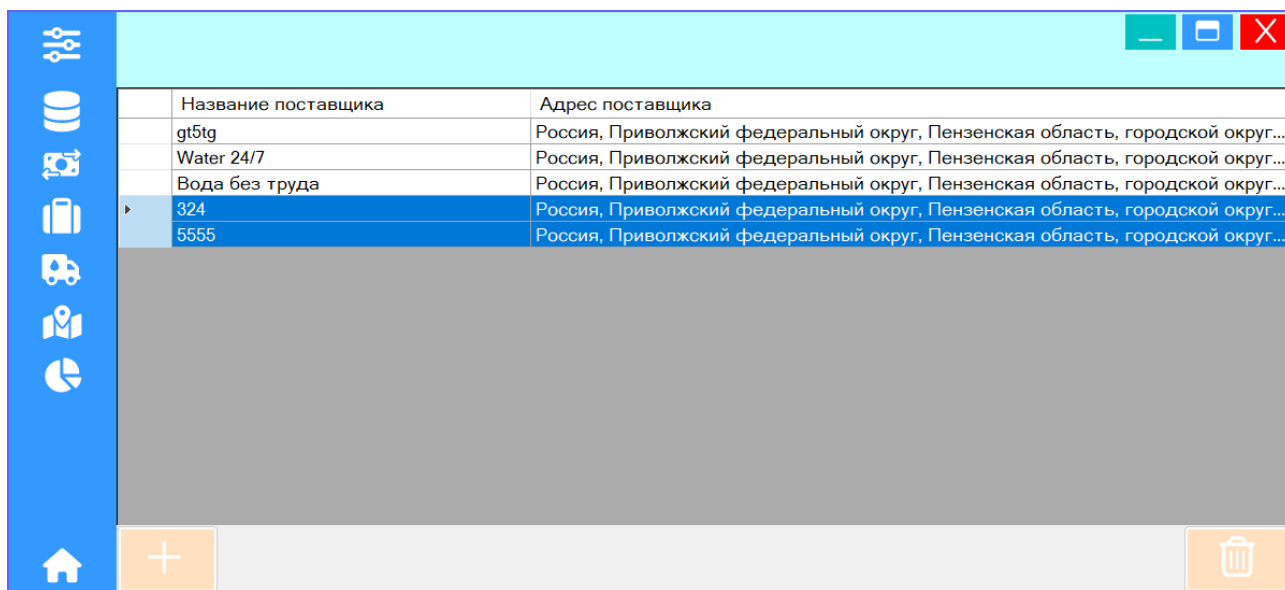


Рисунок 35 – Выделение строк

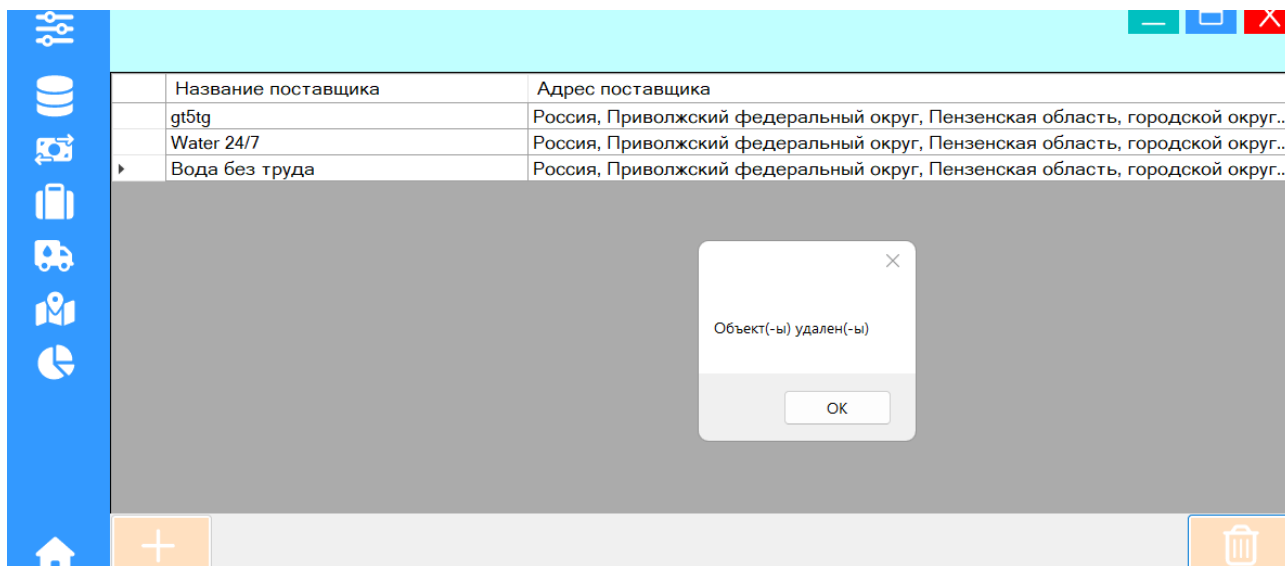


Рисунок 36 – Удаление строк

Таким образом было составлено краткое руководство пользователю для использования ПО.

4 Тестирование ПО

4.1 Тестирование пользовательского интерфейса

4.1.1 Тестирование по стратегии черного ящика

Тестирование программного обеспечения представляет собой один из возможных способов оценки его качества.

Мы планируем использовать метод чёрного ящика на первом этапе тестирования. Этот метод, также известный как функциональное тестирование, позволяет изучить функциональность программного обеспечения без знания внутренней структуры кода. Он может быть использован на всех уровнях тестирования, но на высоких уровнях он применяется чаще всего. Тестирование по стратегии чёрного ящика позволяет изучать поведение системы, независимо от её внутренней структуры, архитектуры и реализации.

Функциональное тестирование (тестирование методом черного ящика, black-box testing). Функции, которые выполняет система, подсистема или компонент, могут быть описаны в таких артефактах процесса разработки, как спецификация требований, сценарии использования системы или функциональная спецификация, либо могут быть недокументированны. Эти функции описывают, «что» данная система делает. Функциональные тесты разрабатываются на основе функций и возможностей системы (описанных в документах или понятных тестировщикам) и их взаимодействия со специфичными системами и могут быть выполнены на всех уровнях тестирования (например, тесты для компонентов могут основываться на спецификациях компонентов) [3].

План тестирования (Таблица 16) составлен на основе выявленных функциональных требований:

1. Подключить новый автомат продажи воды;
2. Включить автомат продажи воды;
3. Отключить автомат продажи воды;
4. Выполнить прогнозирование дохода нового автомата продажи воды;

5. Составить эффективный маршрут транспортировки воды;
6. Восполнить запасы воды и произвести техническое обслуживание;
7. Подключить нового поставщика воды;
8. Добавить в парк новый автомобиль;
9. Отключить поставщика воды;
10. Удаление автомобиля из парка;
11. Произвести сравнительный анализ востребованности автоматов продажи воды.

Примеры функционального тестирования приведены в Таблице 16 ниже.

Таблица 16 — План тестирования

№	Названия тестового случая	Тест	Результат
1	Подключить новый автомат продажи воды	Заполняются данные водомата, после нажатия кнопки «Добавить», новый водомат отображён в таблице.	Данные о новом водомате отображены в таблице
2	Включить автомат продажи воды у	В таблице датчика меняем состояния водомата с «Выкл» на «Вкл»	Статус водомата успешно изменён
3	Отключить автомат продажи воды	В таблице датчика меняем состояния водомата с «Вкл» на «Вцкл»	Статус водомата успешно изменён
4	Выполнить прогнозирование дохода нового автомата продажи воды	Заполняем данные добавления водомата, затем нажимаем на кнопку «Анализа»	Отображена таблица с исходными данными и прогнозом.
5	Составить эффективный маршрут транспортировки воды	Заполняем список точек маршрута, выбираем точку поставщика и автомобиль из парка поставщика. Нажимаем на кнопку «Маршрут»	Маршрут отображён на карте, во вкладке «Информация» маршрут расписан в текстовом формате.

Продолжение Таблицы 16

6	Восполнить запасы воды и произвести техническое обслуживание	После построения маршрута нажимаем кнопку – обслужить, в базе будут обновлены данные и состояние водоматов.	Дата обслуживания и текущий уровень воды в таблице – обновлены.
7	Подключить нового поставщика воды	Заполняются данные поставщика, после нажатия кнопки «Добавить», новый поставщик отображён в таблице.	Данные о новом поставщике отображены в таблице
8	Добавить в парк новый автомобиль	Заполняются данные автомобиля, после нажатия кнопки «Добавить», новый автомобиль отображён в таблице.	Данные о новом автомобиле отображены в таблице
9	Отключить поставщика воды	В таблице «Поставщики» выделяется необходимый, затем нажимается кнопка «Удалить»	Таблица поставщиков обновлена
10	Удаление автомобиля из парка	В таблице «Автомобили» выделяется необходимый, затем нажимается кнопка «Удалить»	Таблица автомобилей обновлена
11	Произвести сравнительный анализ востребованности автоматов продажи воды	Во вкладке «Анализ» выбирается вкладка интересующего анализа, данные отображаются в рабочей области	Результат отображён в рабочей области

Результаты тестов представлены в приложении Г.

4.1.2 Метод граничных условий

Метод граничных условий является техникой тестирования программного обеспечения, которая направлена на проверку корректности поведения программы на границах допустимых значений. Этот метод заключается в выборе тестовых данных, которые лежат на границах допустимых значений входных параметров, и проверке, как программа обрабатывает эти данные. Основная цель этого метода – выявление ошибок, которые могут возникнуть при обработке граничных значений.

Выделенные граничные условия приведены ниже в таблице 17.

Таблица 17 – Граничные условия

Границы	
Физические	Номер автомобиля является не строковым типом данных(Ф1)
Логические	Автомобиль с заданным номером уже имеется в базе (Л1) Номер автомобиля является пустой строкой (Л2)
Технологические	Номер автомобиля содержит символы, которые не входят в состав ASCII таблицы (Т1)
Произвольные	Номер автомобиля меньше 9 символов или больше 14 символов (П1) Номер автомобиля начинается не с буквы (П2)

Для представленных выше граничных значений были составлены соответствующие тест-кейсы (Таблица 18).

Таблица 18 – Тест – кейсы для граничных условий

№	Границы	Входные данные	Выходные данные
1	Ф1	Float: number = 123456789.10	Ошибка: тип данных не является строковым
2	Л1	Add.number(«a123бв101») Add.number(«a123бв101»)	Ошибка: номер уже имеется в базе
3	Л2	«»	Ошибка: Пустая строка

Продолжение Таблицы 18

4	T1	a123бв10®	Ошибка: номер содержит символы не из ASCII таблицы
5	П1	a123бв10101	Ошибка: номер больше 9 символов

4.2 Тестирование серверной части

4.2.1 Модульное тестирование

Модульное тестирование является неотъемлемой частью разработки программного обеспечения. Для его проведения используются специальные инструменты и фреймворки, такие как TestTools.UnitTesting. Данный фреймворк позволяет разработчикам написать модульные тесты для своих программ или библиотек, что позволяет обнаруживать и устранять ошибки на ранних этапах разработки. С помощью модульных тестов можно проверить работу отдельных методов, классов или функциональности программы в целом.

Модульные тесты позволяют убедиться в правильной работе каждого модуля кода и выявить ошибки до того, как они окажутся в готовом продукте. Пространство имен Microsoft.VisualStudio.TestTools.UnitTesting содержит ряд классов и атрибутов, которые позволяют легко и быстро создавать и запускать модульные тесты. Классы позволяют создавать объекты тестирования, атрибуты позволяют указать условия и ограничения для запуска тестов.

При написании тестов необходимо учитывать требования к коду и спецификацию, чтобы гарантировать, что каждый тест будет проверять нужный аспект функциональности. Использование фреймворка позволяет упростить процесс тестирования и повысить эффективность разработки. Таблица 19 содержит план тестирования для сущности Car.

Таблица 19 – План тестирования

№ п/п	Тестовый метод	Описание теста	Входные данные тестируемой функции	Результат теста
1	AddCarPresenterTests	Тест создает объекты тестового интерфейса IAddCarView и тестового контекста базы данных TestDbContext, и затем создает экземпляр презентера AddCarPresenter, передавая ему созданные объекты. Затем тестовый метод вызывает метод btnAddClicked() на тестовом интерфейсе, эмулируя действие пользователя - нажатие кнопки "добавить" на форме добавления автомобиля. После этого тест проверяет, содержит ли тестовый контекст базы данных один добавленный автомобиль.	Номер автомобиля, модель автомобиля, тип автомобиля	Тест пройден (Рисунок 1)
2	AddMarkerPresenterTests	Тест для проверки функциональности класса AddMarkerPresenter. В тесте создаются заглушки представления и базы данных, затем создается объект класса AddMarkerPresenter с этими фиктивными объектами в качестве параметров. В конце теста вызывается метод btnAddClicked на фиктивном представлении, эмулируя действие пользователя по добавлению маркера. Затем проверяется, что координаты и адрес, переданные в фиктивное представление, совпадают с теми, которые были введены в объекте класса AddMarkerPresenter.	Координаты, адрес	Тест пройден (Рисунок 1)

Продолжение Таблицы 19

3	AddSupplierPresenterTests	Тест для проверки функциональности класса AddMarkerPresenter. В тесте создаются заглушки представления и базы данных, затем создается объект класса AddMarkerPresenter с этими фиктивными объектами в качестве параметров. В конце теста вызывается метод btnAddClicked на фиктивном представлении, эмулируя действие пользователя по добавлению маркера. Затем проверяется, что координаты и адрес, переданные в фиктивное представление, совпадают с теми, которые были введены в объекте класса AddMarkerPresenter, а также число объектов в фиктивной базе увеличилось	Координаты, адрес, название поставщика	Тест пройден (Рисунок 1)
---	---------------------------	--	--	--------------------------

Тестирование	Длительн...	Признаки	Сообщен...
✔ UnitTestProject (3)	97 мс		
✔ UnitTestProject (3)	97 мс		
✔ AddCarPresenterTests (1)	17 мс		
✔ AddCarTest	17 мс		
✔ AddMarkerPresenterTests (1)	1 мс		
✔ AddMarkerTest	1 мс		
✔ AddSupplierPresenterTests (1)	79 мс		
✔ AddSupplierTest	79 мс		

Рисунок 37 – Обзор тестов MS Visual Studio

В Листинге 1 приведен код модульного теста «AddCarPresentersTests».

Листинг 1– Код модульных тестов

```
using Diplom.Presenters;
using Diplom.Views.Interfaces;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Text;
using System.Threading.Tasks;

namespace UnitTestProject
{
    [TestClass]
    public class AddMarkerPresenterTests
    {
        [TestMethod]
        public void AddMarkerTest()
        {
            double[] coords = new double[2] { 10, 10 };
            string address = "address";
            IAddMarkerView view = new TestAddMarkerView(coords, address);
            TestDbContext db = new TestDbContext();
            AddMarkerPresenter addMarkerPresenter = new AddMarkerPresenter(view,
db);

            ((TestAddMarkerView)addMarkerPresenter.View).btnAddClicked();

            Assert.AreEqual(coords, addMarkerPresenter.Coordinates);
            Assert.AreEqual(address, addMarkerPresenter.Address);
        }
    }
}

```

Результаты остальных тестов приведены в приложении В.

4.3 Метрики кода

Метрики кода - это количественные индикаторы, которые используются для измерения различных аспектов качества и эффективности программного кода. Они являются инструментами для анализа и оптимизации кода в процессе его разработки, тестирования и поддержки в рамках проекта программирования.

Метрики кода могут быть использованы для следующих целей:

1. Измерение качества кода: метрики кода помогают оценить, насколько код программы соответствует стандартам и критериям качества, таким как чистота кода, структура, надежность и устойчивость к ошибкам;
2. Снижение сложности кода: метрики кода помогают выявлять участки кода, которые могут быть упрощены или оптимизированы, с целью

улучшения читабельности, поддерживаемости, расширяемости и производительности приложения;

3. Оценка производительности: метрики кода помогают определить узкие места в производительности приложения, с целью повышения его скорости работы и эффективности;

4. Снижение затрат на разработку: метрики кода помогают выявить проблемы и ошибки в программе на ранней стадии, что позволяет избежать дорогостоящих процедур тестирования, отладки и исправления ошибок в более поздние этапы разработки;

5. В целом метрики кода помогают значительно сократить время и затраты на разработку, а также улучшить качество и производительность программного кода, что способствует успешной реализации проекта программирования.

Для оценки качества разработанного приложения мы будем использовать следующие группы метрик: метрику размера программы (SLOC), метрики сложности потока управления программы (метрики Холстеда и цикломатическая сложность).

4.3.1 Метрики размера программы

В самом начале используем метрики SLOC по коду, который приведен в Листинге 2.

Листинг 2 – Код для SLOC метрик

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Diplom.Data;
using Diplom.Forms;
using Diplom.Views.Interfaces;

namespace Diplom.Presenters
{
```

```

public class MainPresenter : IPresenter
{
    private IMainView _mainView;
    private IPresenter _currentPresenter;
    private ModelContext _db;

    public MainPresenter(IMainView mainView)
    {
        // Инициализация главного окна и навешивание обработчиков событий
        _mainView = mainView;
        _mainView.btnMapClick += _mainView_btnMapClick;
        _mainView.btnCarClick += _mainView_btnCarClick;
        _mainView.btnSuppClick += _mainView_btnSuppClick;
        _mainView.btnSaleClick += _mainView_btnSaleClick;
        _mainView.btnMachineClick += _mainView_btnMachineClick;
        _mainView.btnHomeClick += _mainView_btnHomeClick;
        _mainView.btnAnalysisClick += _mainView_btnAnalysisClick;

        // Инициализация контекста базы данных
        _db = new ModelContext();

        // Установка главного окна в начальное состояние (на экране будет окно
"Home")
        _currentPresenter = new HomePresenter(new HomeForm());
        _mainView.SetChild(_currentPresenter.View);
    }

    public IView View
    {
        get { return _mainView; }
    }

    private void _mainView_btnAnalysisClick(object sender, EventArgs e)
    {
        // Очистка предыдущего окна
        if (_currentPresenter != null)
        {
            _currentPresenter.Dispose();
        }

        // Открытие окна анализа данных
        _currentPresenter = new AnalysisPresenter(new AnalysisForm(), _db);
    }
}

```

```

        _mainView.SetChild(_currentPresenter.View);
    }

    private void _mainView_btnHomeClick(object sender, EventArgs e)
    {
        // Очистка предыдущего окна
        if (_currentPresenter != null)
        {
            _currentPresenter.Dispose();
        }

        // Открытие окна "Home"
        _currentPresenter = new HomePresenter(new HomeForm());
        _mainView.SetChild(_currentPresenter.View);
    }

    private void _mainView_btnMachineClick(object sender, EventArgs e)
    {
        // Очистка предыдущего окна
        if (_currentPresenter != null)
        {
            _currentPresenter.Dispose();
        }

        // Открытие окна управления водоразбрызгивающими машинами
        _currentPresenter = new WaterMachinePresenter(new WaterMachineForm(),
        _db);

        _mainView.SetChild(_currentPresenter.View);
    }

    private void _mainView_btnSaleClick(object sender, EventArgs e)
    {
        // Очистка предыдущего окна
        if (_currentPresenter != null)
        {
            _currentPresenter.Dispose();
        }

        // Открытие окна управления транзакциями
        _currentPresenter = new TransactionPresenter(new TransactionForm(),
        _db);

        _mainView.SetChild(_currentPresenter.View);
    }

```

```

}

private void _mainView_btnSuppClick(object sender, EventArgs e)
{
    // Очистка предыдущего окна
    if (_currentPresenter != null)
    {
        _currentPresenter.Dispose();
    }

    // Открытие окна управления поставщиками
    _currentPresenter = new SuppliersPresenter(new SupplierForm(), _db);
    _mainView.SetChild(_currentPresenter.View);
}

private void _mainView_btnCarClick(object sender, EventArgs e)
{
    // Очистка предыдущего окна
    if (_currentPresenter != null)
    {
        _currentPresenter.Dispose();
    }

    // Открытие окна управления автомобилями
    _currentPresenter = new CarsPresenter(new CarsForm(), _db);
    _mainView.SetChild(_currentPresenter.View);
}

private void _mainView_btnMapClick(object sender, EventArgs e)
{
    // Очистка предыдущего окна
    if (_currentPresenter != null)
    {
        _currentPresenter.Dispose();
    }

    // Открытие окна карты
    _currentPresenter = new MapPresenter(new MapForm(), _db);
    _mainView.SetChild(_currentPresenter.View);
}

public void Dispose()

```

```

{
    // Удаление обработчиков событий из главного окна
    _mainView.btnMapClick -= _mainView_btnMapClick;
    _mainView.btnCarClick -= _mainView_btnCarClick;
    _mainView.btnSuppClick -= _mainView_btnSuppClick;
    _mainView.btnSaleClick -= _mainView_btnSaleClick;
    _mainView.btnMachineClick -= _mainView_btnMachineClick;
    _mainView.btnHomeClick -= _mainView_btnHomeClick;
    _mainView.btnAnalysisClick -= _mainView_btnAnalysisClick;

    // Закрытие соединения с базой данных
    _db.Dispose();
}
}
}

```

В данном классе имеем:

- 128 физических строк кода;
- 71 логических строк кода;
- 19 строк комментариев.

4.3.2 Метрики Холстеда

Метрики Холстеда - это количественные индикаторы, которые используются для измерения сложности программного кода и количества информации, которую он содержит. Они были предложены Майклом Холстедом в 1977 году и являются одними из самых распространенных метрик кода.

Метрики Холстеда измеряют программный код по четырем категориям:

1. Объем (Volume) — это количество уникальных операторов и операндов в программном коде. Чем больше объем, тем сложнее код;
2. Длина (Length) — это общее количество символов в программном коде. Чем больше длина, тем сложнее код;
3. Сложность (Difficulty) — это мера сложности программного кода, определяемая, как отношение между объемом и длиной кода;

4. Уровень абстракции (Abstraction) — это мера абстракции программного кода, которая измеряет отношение между уникальными операторами и операндами в коде.

Протестируем метрики цикломатической сложности. Для этого возьмём метод, который отвечает за валидацию таксофонной карты и составим граф, который приведен на Рисунке 38. Блок рассматриваемого кода приведен ниже. Блок рассматриваемого кода приведен Листинге 3.

Листинг 3 – Код для метрик Холстеда

```
private void _carView_btnAddClick(object sender, EventArgs e)
{
    if ((new Regex(@"^[a-яё\-\s]{1}[0-9]{3}(?!0{3})[a-яё\-\s]{2}[0-9]{2,3}$")).IsMatch(_carView.Number) == false)
    {
        _carView.ShowDialog("Номер должен соответствовать формату -
a123бв000 ");
        return;
    }
    if (_carView.Model.Length > 100 || _carView.Model.Length == 0)
    {
        _carView.ShowDialog("Количество символов в поле 'Модель
автомобиля' не должно превышать 100 или оставаться пустым");
        return;
    }
    if (_carView.Type.Length > 100 || _carView.Type.Length == 0)
    {
        _carView.ShowDialog("Количество символов в поле 'Тип
автомобиля' не должно превышать 100 или оставаться пустым");
        return;
    }
    foreach (var i in _db.Cars)
    {
        if (i.CarNumber == _carView.Number)
        {
            _carView.ShowDialog("Номер автомобиля должен быть
уникальным");
            return;
        }
    }
}
```

```
// создаём объект
var car = new Car(_carView.Number, _carView.Model, _carView.Type);
// поиск поставщика по имени на форме
Supplier sp = _db.Suppliers.ToList().FirstOrDefault(c => c.SuppName
== _carView.SupplierName);
car.SuppId = sp.Id;
_db.Cars.Add(car);
// сохраняем контекст бд
_db.SaveChanges();
_carView.ShowDialog("Объект успешно добавлен");
_carView.Close();
}
```

Данный код отображает процедуру проверки и добавления нового автомобиля в базу данных.

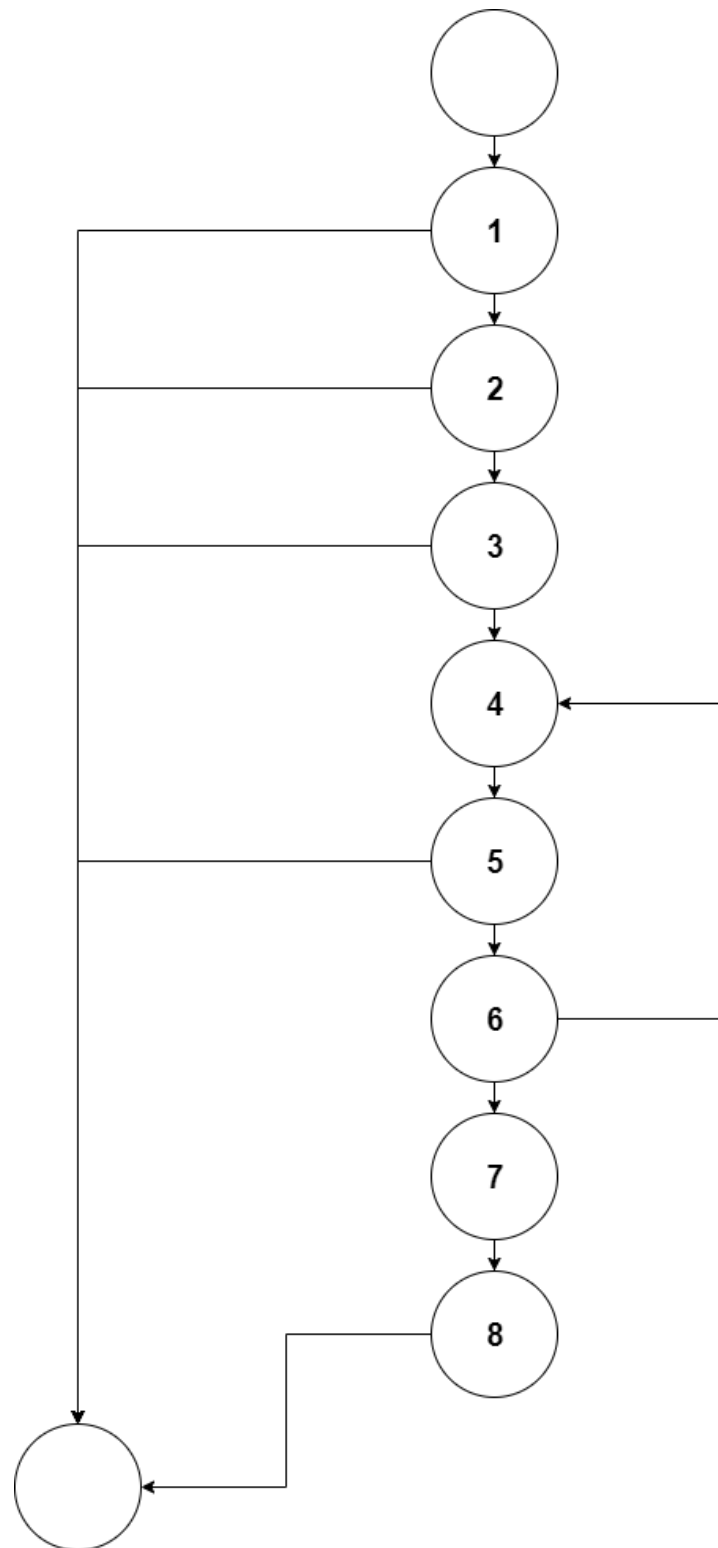


Рисунок 38 – Граф потока и управления

Для вычисления цикломатической сложности $V(G)$ применяется следующая формула:

$$V(G) = e - n + 2p \quad (9)$$

где e - количество дуг, n - количество вершин, p - число компонент

СВЯЗНОСТИ.

Граф содержит 10 вершин, 14 дуг и ноль компонент связности.

$$V(G) = 14 - 10 + 0 = 4$$

Таким образом значение цикломатической сложности равно 4.

Приведем замеры по метрикам Холстеда, метрики будут реализованы на основе метода для моделирования одного дня работы водомата. Ниже приведены в Листинге 4 данные для данной метрики и сам код данного метода.

Листинг 4 – Данные метрики и код

```
private void _view_btnGenerateClick(object sender, EventArgs e)
{
    var _random = new Random();
    var _machines = _db.WaterVendingMachines.ToList().Where(c =>
c.WaterLevelSensor.FirstOrDefault().StatusSensor == true).ToList();
    var _startDate = _db.PurchaseLogs.FirstOrDefault() == null ?
DateTime.Today : _db.PurchaseLogs.ToList().Max(c =>
c.PurchaseDate.Date).AddDays(1);
    var _endDate = _startDate.AddDays(1);

    foreach (WaterVendingMachine m in _machines)
    {
        var rang = m.Rang;
        double sin=0;
        switch (rang)
        {
            case 0:
                sin = Math.PI / 5;
                m.Rang = 1;
                break;
            case 1:
                sin = Math.PI / 4;
                m.Rang = 2;
                break;
            case 2:
                sin = Math.PI / 3;
                m.Rang = 3;
                break;
            case 3:
                sin = Math.PI / 2;
                m.Rang = 0;
        }
    }
}
```

```

        break;
    default:
        break;
    }

    var n = Math.Sin(sin) * (100*((int)m.Price%5)+1) * (3000 /
(MainForm._distanceToCentre(new double[]
{
double.Parse(m.Coordinate.Split().First()),double.Parse(m.Coordinate.Split().Las
t())

}, new double[] { 53.194049, 45.016735 })))));
_db.SaveChanges();
n += new Random().NextDouble() * (n / 40 + n / 40) - n/40;
var value = n;
List<DateTime> _dates = new List<DateTime>();
while (n>0)
{
    int range = (int)(_endDate - _startDate).TotalSeconds;
    int randomDays = _random.Next(range);

    DateTime randomDate = _startDate.AddSeconds(randomDays);

    var _vol = new
Random((int)DateTime.Now.Ticks).NextDouble()*(value/20 - value/100)+ value/100;
    _vol = _vol>n ? n: _vol;
    if (m.WaterLevelSensor.FirstOrDefault().StatusWater - _vol <
0)

    {

        m.WaterLevelSensor.FirstOrDefault().StatusSensor =
false;

        break;
    }
    _db.WaterLevelSensors.Where(c => c.WaterMachineId ==
m.Id).ToList().FirstOrDefault().StatusWater -= (float)_vol;
    PurchaseLog _log = new PurchaseLog(m.Id, randomDate,
(float)_vol, (decimal)_vol * m.Price);
    _log.WaterVendingMachine = m;
    m.ServiceDate = _log.PurchaseDate;
    _db.PurchaseLogs.Add(_log);
    n -= _vol;
}
}
_db.SaveChanges();

```

```
}
```

Произведём подсчёт количество операторов и операндов, а также количество их вхождений (Таблицы 20 и 21).

Таблица 20 – Операторы

Операторы	Число вхождений
=	25
==	3
.	66
;	40
New	7
If	1
Next()	1
NextDouble()	2
AddDays()	2
Add()	1
Where	2
FirstOrDefault()	5
<	1
[]	1
ToList	4
=>	3
+=	1
-=	2
while	1
double	4
int	5
float	1
decimal	1
Max()	1
Pi	4
?	2
:	7
switch	1
case	4
foreach	1
default	1
breack	6
()	46
{}	7
-	4
+	2

Продолжение Таблицы 20

*	5
/	11
Sin()	1
null	1
True	1
,	6
>	1
Parse()	2
Split()	2
First()	1
Last()	1
_distanceToCentre	1

В результате получим число операторов $n1 = 48$, а также число вхождений операторов $N1 = 298$.

Таблица 21 – Операнды

Операнды	Число вхождений
_random	2
_machines	2
WaterVendingMachines	1
WaterLevelSensor	4
StatusSensor	2
_startDate	4
_endDate	2
_db	6
PurchaseLogs	3
DateTime	5
Today	1
PurchaseDate	2
Date	1
WaterVendingMachine	2
Math	5
MainForm	1
m	13
rang	2
sin	6
n	9
Random	3
value	4
range	2

Продолжение Таблицы 21

randomDays	2
randomDate	2
_vol	9
Price	2
Coordinate	2
TotalSeconds	1
Now	1
Ticks	1
DateTime	5
StatusWater	2
WaterMachineId	1
PurchaseLog	2
_log	4
ServiceDate	1
Rang	5
n2 = 38	N2 = 122

В результате получим число операндов $n2 = 38$, а также число вхождений операторов $N1 = 122$.

Длина программы рассчитывается как:

$$N = N1 + N2 = 298 + 122 = 420$$

Словарь программы рассчитывается как:

$$n = n1 + n2 = 48 + 38 = 86$$

Объем программы рассчитывается как:

$$HV = N * \log_2(n) = 420 * 6,426 = 2\,698,92$$

Потенциальный объем программы:

$$V^* = (2 + n) * \log_2(2 + n) = 4 * \log_2(4) = 8$$

Теоретический уровень программы рассчитывается как:

$$L' = V^* / HV = 8 / 2698,92 = 0,00296$$

Сложность программы рассчитывается как:

$$HD = (n1 / 2) * (N2 / n2) = (48 / 2) * (122 / 38) = 77.05$$

Заключение

В результате проведенного исследования были изучены особенности принципы работы и способы разработки программного обеспечения. Было проведено исследования вариантов использования ПО для прогнозирования и анализа состояния сети водоснабжения.

Была рассмотрена и проанализирована актуальная техническая литература по проектированию и разработке автоматизированных информационных систем.

Проведя анализ существующих аналогов разрабатываемого ПО, было установлено, что такие системы или не отвечают всем функциональным требованиям, или слишком сложны и дороги в использовании.

Для того, чтобы учесть все аспекты работы с приложением, были исследованы варианты использования системы для прогнозирования и анализа сети водоснабжения. Была определена подходящая структура приложения, а также проанализированы средства разработки настольных приложений для Windows.

Было проведено проектирование программных средства, структур хранения данных и пользовательского интерфейса. Разработаны макеты интерфейса клиентской части приложения. Был определен и рассмотрен процесс обработки данных, а также составлена диаграмма компонентов. Была спроектирована структура хранения данных и составлена диаграмма базы данных.

Была спроектирована диаграмма классов для приложения на основе архитектуры MVP и приведены описания, того как выполняются её принципы на основе данного проекта.

После проектирования было реализовано приложение для прогнозирования и анализа состояния сети водоснабжения. Приложение имеет удобный пользовательский интерфейс и позволяет вести учёт и анализ водоматов, поставщиков и парка их автомобилей, а также строить оптимальный маршрут доставки и составлять прогноз расхода автомата на следующий день.

Проект был спланирован с определением сроков выполнения работ и связей между ними. Был разработан план стадий и этапов создания веб-приложения, на основе которого была создана диаграмма Ганта. Изучение этапов и их продолжительности позволило определить, что на проектирование и разработку системы требуется 182 часа. Расчет стоимости проекта был выполнен на основе плана, учитывая расходы на амортизацию, заработную плату, социальные выплаты и другие материальные затраты. Была создана диаграмма структуры расходов, позволившая определить общую стоимость проекта – 117269,94 рублей. В целом можно сказать, что задача была выполнена.

Список использованных источников

1. Леоненков А.В. Самоучитель UML. – 2-е изд., перераб. и доп. – СПб. БХВ-Петербург, 2004. – 432с. : ил. ISBN 5-94157-342-1
2. Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке С#. –Пер. с англ. – СПб.: Символ-Плюс, 2011
3. Пероцкая, В. Н. Основы тестирования программного обеспечения : учеб. пособие / В. Н. Пероцкая, Д. А. Градусов ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2017 – 100 с. ISBN 978-5-9984-0777
4. Максимова Т.Г., Попова И.Н. Эконометрика: учебно-методическое пособие /Т.Г. Максимова, И.Н. Попова. – СПб.: Университет ИТМО, 2018 – 70 с.
5. learn.microsoft.com – EntityFramework [Электронный ресурс]. URL:<https://learn.microsoft.com/ru-ru/ef/> (дата обращения: 20.06.2023).
6. docs.itinero.tech – Itinero [Электронный ресурс]. URL: <http://docs.itinero.tech/docs/index.html> (дата обращения: 20.06.2023).
7. github.com – Map control GMat.Net [Электронный ресурс]. URL: <https://github.com/judero01col/GMap.NET/blob/master/README.md> (дата обращения: 20.06.2023).
8. yandex.ru – Yandex Geocoder [Электронный ресурс]. URL: <https://yandex.ru/dev/maps/geocoder/doc/desc/> (дата обращения: 20.06.2023).
9. numeric.mathdotnet [Электронный ресурс]. URL: <https://numerics.mathdotnet.com/> (дата обращения: 20.06.2023).
10. www.bentley.com – software WaterGEMS [Электронный ресурс]. URL: <https://www.bentley.com/software/openflows-watergems/> (дата обращения: 20.06.2023).
11. www.water-simulation.com – software H2MAP-Water [Электронный ресурс]. URL: <http://www.water-simulation.com/wsp/2005/05/11/h2omap-water/> (дата обращения: 20.06.2023).

12. developers.google.com – OrTools [Электронный ресурс]. URL: <https://developers.google.com/optimization/introduction> (дата обращения: 20.06.2023).
13. www.epa.gov – software EPANET [Электронный ресурс]. URL: <https://www.epa.gov/water-research/epanet> (дата обращения: 20.06.2023).

Приложения

Приложение А **Листинг кода ПО** **(обязательное)**

Интерфейс **IDatabaseContext.cs** :

```
using Diplom.Entity;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Data
{
    public interface IDatabaseContext
    {
        DbSet<Car> Cars { get; set; }
        DbSet<PurchaseLog> PurchaseLogs { get; set; }
        DbSet<Supplier> Suppliers { get; set; }
        DbSet<sysdiagrams> sysdiagrams { get; set; }
        DbSet<WaterVendingMachine> WaterVendingMachines { get; set; }
        DbSet<WaterLevelSensor> WaterLevelSensors { get; set; }
        int SaveChanges();
    }
}
```

Класс **ModelContext.cs** :

```
using System;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;
using System.Linq;
using Diplom.Entity;
namespace Diplom.Data
{
    public partial class ModelContext : DbContext, IDatabaseContext
    {
        public ModelContext(): base("Model")
        {
        }

        public virtual DbSet<Car> Cars { get; set; }
        public virtual DbSet<PurchaseLog> PurchaseLogs { get; set; }
        public virtual DbSet<Supplier> Suppliers { get; set; }
        public virtual DbSet<sysdiagrams> sysdiagrams { get; set; }
        public virtual DbSet<WaterVendingMachine> WaterVendingMachines { get;
set; }
        public virtual DbSet<WaterLevelSensor> WaterLevelSensors { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Car>()
                .Property(e => e.CarBrand)
                .IsUnicode(false);

            modelBuilder.Entity<Car>()
                .Property(e => e.CarType)
                .IsUnicode(false);

            modelBuilder.Entity<Car>()
                .Property(e => e.CarNumber)
                .IsUnicode(false);

            modelBuilder.Entity<PurchaseLog>()
```

```

        .Property(e => e.TotalPrice)
        .HasPrecision(19, 4);

modelBuilder.Entity<Supplier>()
    .Property(e => e.SuppName)
    .IsUnicode(false);

modelBuilder.Entity<Supplier>()
    .Property(e => e.SuppAddress)
    .IsUnicode(false);

modelBuilder.Entity<Supplier>()
    .HasMany(e => e.Car)
    .WithOptional(e => e.Supplier)
    .HasForeignKey(e => e.SuppId);

modelBuilder.Entity<Supplier>()
    .HasMany(e => e.WaterVendingMachine)
    .WithMany(e => e.Supplier)
    .Map(m =>
m.ToTable("Transportation").MapLeftKey("SuppId").MapRightKey("WaterMachineId"));

modelBuilder.Entity<WaterVendingMachine>()
    .Property(e => e.Location)
    .IsUnicode(false);

modelBuilder.Entity<WaterVendingMachine>()
    .Property(e => e.Price)
    .HasPrecision(10, 2);

modelBuilder.Entity<WaterVendingMachine>()
    .Property(e => e.Status)
    .IsUnicode(false);

modelBuilder.Entity<WaterVendingMachine>()
    .HasMany(e => e.PurchaseLog)
    .WithRequired(e => e.WaterVendingMachine)
    .HasForeignKey(e => e.MachineId)
    .WillCascadeOnDelete(false);

modelBuilder.Entity<WaterVendingMachine>()
    .HasMany(e => e.WaterLevelSensor)
    .WithRequired(e => e.WaterVendingMachine)
    .HasForeignKey(e => e.WaterMachineId)
    .WillCascadeOnDelete(false);

modelBuilder.Entity<WaterLevelSensor>()
    .Property(e => e.Model)
    .IsUnicode(false);
    }
}
}

```

Класс Car.cs :

```

namespace Diplom.Entity
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;
    using System.ComponentModel.DataAnnotations.Schema;
    using System.ComponentModel;

```

```

using System.Data.Entity.Spatial;

[Table("Car")]
public partial class Car
{
    public Car() { }
    public Car(string _carNumber, string _carModel, string _carType)
    {
        CarNumber = _carNumber;
        CarBrand = _carModel;
        CarType = _carType;
    }
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Id { get; set; }

    [Required]
    [StringLength(50)]
    public string CarBrand { get; set; }

    [Required]
    [StringLength(50)]
    public string CarType { get; set; }

    public int? SuppId { get; set; }

    [Required]
    [StringLength(50)]
    [Index(IsUnique = true)]
    [RegularExpression(@"^[a-яё\-\s]{1}[0-9]{3}(?!0{3})[a-яё\-\s]{2}[0-9]{2,3}$")]
    public string CarNumber { get; set; }

    public virtual Supplier Supplier { get; set; }
}

```

Класс **PurchaseLog.cs** :

```

namespace Diplom.Entity
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;
    using System.ComponentModel;
    using System.ComponentModel.DataAnnotations.Schema;
    using System.Data.Entity.Spatial;

    [Table("PurchaseLog")]
    public partial class PurchaseLog
    {
        public PurchaseLog(int _machineId, DateTime _purchaseDate, float
        _purchaseVolume, decimal _totalPrice, float _currWaterBalance)
        {
            MachineId = _machineId;
            PurchaseDate = _purchaseDate;
            PurchaseVolume = _purchaseVolume;
            TotalPrice = _totalPrice;
            CurrentWaterBalance = _currWaterBalance;
        }
        public PurchaseLog() { }
        [Key]
    }
}

```

```

        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public int MachineId { get; set; }
        public DateTime PurchaseDate { get; set; }
        public float PurchaseVolume { get; set; }

        [Column(TypeName = "money")]
        public decimal TotalPrice { get; set; }
        public float CurrentWaterBalance { get; set; }
        public virtual WaterVendingMachine WaterVendingMachine { get; set; }
    }
}

```

Класс **Supplier.cs** :

```

namespace Diplom.Entity
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.ComponentModel.DataAnnotations;
    using System.ComponentModel.DataAnnotations.Schema;
    using System.Data.Entity.Spatial;

    [Table("Supplier")]
    public partial class Supplier
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Supplier()
        {
            Car = new HashSet<Car>();
            WaterVendingMachine = new HashSet<WaterVendingMachine>();
        }
        public Supplier(string _suppName, string _suppAddress, string
_coordinate)
        {
            SuppName = _suppName;
            SuppAddress = _suppAddress;
            Coordinate = _coordinate;
        }
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        [Required]
        [StringLength(50)]
        public string SuppName { get; set; }

        [Required]
        [StringLength(255)]
        public string SuppAddress { get; set; }

        [Required]
        [Browsable(false)]
        [StringLength(100)]
        public string Coordinate { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Car> Car { get; set; }
    }
}

```

```

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<WaterVendingMachine> WaterVendingMachine {
get; set; }
    }
}

```

Класс **WaterLevelSensor.cs** :

```

namespace Diplom.Entity
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.ComponentModel.DataAnnotations;
    using System.ComponentModel.DataAnnotations.Schema;
    using System.Data.Entity.Spatial;

    [Table("WaterLevelSensor")]
    public partial class WaterLevelSensor
    {
        [Key]
        [Column(Order = 0)]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int WaterMachineId { get; set; }

        [Key]
        [Column(Order = 1)]
        [DisplayName("Модель датчика")]
        [StringLength(100)]
        public string Model { get; set; }

        [DisplayName("Текущий остаток воды")]
        public float StatusWater { get; set; }

        [DisplayName("Работа датчика")]
        public bool StatusSensor { get; set; }

        public virtual WaterVendingMachine WaterVendingMachine { get; set; }
    }
}

```

Класс **WaterVendingMachine.cs** :

```

namespace Diplom.Entity
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.ComponentModel.DataAnnotations;
    using System.ComponentModel.DataAnnotations.Schema;
    using System.Data.Entity.Spatial;

    [Table("WaterVendingMachine")]
    public partial class WaterVendingMachine
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public WaterVendingMachine()
        {
            PurchaseLog = new HashSet<PurchaseLog>();
            WaterLevelSensor = new HashSet<WaterLevelSensor>();
        }
    }
}

```

```

        Supplier = new HashSet<Supplier>();
    }
    public WaterVendingMachine(string _location, int _capacity, decimal
_price, string _coordinate, DateTime _serviceDate)
    {
        Location = _location;
        Capacity = _capacity;
        Price = _price;
        Coordinate = _coordinate;
        ServiceDate = _serviceDate;
        Rang = 0;
    }
    public int Id { get; set; }

    [Required]
    [StringLength(255)]
    [DisplayName("Адрес")]
    public string Location { get; set; }
    [DisplayName("Ёмкость резервуара")]
    public int Capacity { get; set; }
    [DisplayName("Цена за литр")]
    public decimal Price { get; set; }
    [Required]
    [Browsable(false)]
    [StringLength(100)]
    public string Coordinate { get; set; }

    [Column(TypeName = "date")]
    [DisplayName("Дата последнего обслуживания")]
    public DateTime ServiceDate { get; set; }
    [Browsable(false)]
    public int Rang { get; set; }

    [Required]
    [StringLength(20)]
    [Browsable(false)]
    public string Status { get; set; }

    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<PurchaseLog> PurchaseLog { get; set; }

    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<WaterLevelSensor> WaterLevelSensor { get;
set; }

    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Supplier> Supplier { get; set; }
    }
}

```

Класс **WaterVendingMachine.cs** :

```

namespace Diplom.Entity
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.ComponentModel.DataAnnotations;
    using System.ComponentModel.DataAnnotations.Schema;

```

```

using System.Data.Entity.Spatial;

[Table("WaterVendingMachine")]
public partial class WaterVendingMachine
{
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
    public WaterVendingMachine()
    {
        PurchaseLog = new HashSet<PurchaseLog>();
        WaterLevelSensor = new HashSet<WaterLevelSensor>();
        Supplier = new HashSet<Supplier>();
    }
    public WaterVendingMachine(string _location, int _capacity, decimal
_price, string _coordinate, DateTime _serviceDate)
    {
        Location = _location;
        Capacity = _capacity;
        Price = _price;
        Coordinate = _coordinate;
        ServiceDate = _serviceDate;
        Rang = 0;
    }
    public int Id { get; set; }

    [Required]
    [StringLength(255)]
    [DisplayName("Адрес")]
    public string Location { get; set; }
    [DisplayName("Ёмкость резервуара")]
    public int Capacity { get; set; }
    [DisplayName("Цена за литр")]
    public decimal Price { get; set; }
    [Required]
    [Browsable(false)]
    [StringLength(100)]
    public string Coordinate { get; set; }

    [Column(TypeName = "date")]
    [DisplayName("Дата последнего обслуживания")]
    public DateTime ServiceDate { get; set; }
    [Browsable(false)]
    public int Rang { get; set; }

    [Required]
    [StringLength(20)]
    [Browsable(false)]
    public string Status { get; set; }

    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<PurchaseLog> PurchaseLog { get; set; }

    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<WaterLevelSensor> WaterLevelSensor { get;
set; }

    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Supplier> Supplier { get; set; }
}

```

```
}
```

Класс **AutoRegression.cs** :

```
using MathNet.Numerics.LinearAlgebra;
namespace Diplom.MLRegressor
{
    class AutoRegression
    {
        private double[] _coefficients;
        private int _p;

        public AutoRegression(int order)
        {
            _coefficients = new double[order];
            _p = order;
        }

        public void Fit(double[] data)
        {
            int n = data.Length - _p;

            double[, ] X = new double[n, _p];

            // заполнение матрицы X
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < _p; j++)
                {
                    X[i, j] = data[i + j];
                }
            }

            double[] y = new double[n];

            // заполнение вектора y
            for (int i = 0; i < n; i++)
            {
                y[i] = data[i + _p];
            }

            // решение уравнения  $X * \_coefficients = y$  методом наименьших
квадратов
            var Xmtx = Matrix<double>.Build.DenseOfArray(X);
            var Ymtx = Vector<double>.Build.DenseOfArray(y);
            var coefficients = ((Xmtx.Transpose() * Xmtx).Inverse() *
Xmtx.Transpose() * Ymtx);
            // .Transpose().Row(0);
            _coefficients = coefficients.ToArray();
        }

        public double Predict(double[] data)
        {
            double result = 0.0;
            for (int i = 0; i < _p; i++)
            {
                result += _coefficients[i] * data[data.Length - _p + i];
            }

            return result;
        }
    }
}
```

```
}
```

Класс **MultipleLinearRegression.cs** :

```
using MathNet.Numerics.LinearAlgebra;
using System;
using System.Linq;

namespace Diplom.MLRegressoror
{
    public class MultipleLinearRegressoror
    {
        private double _b;
        private double[] _w;

        public MultipleLinearRegressoror()
        {
            _b = 0;
        }

        public void Fit(double[,] X, double[,] y)
        {
            var input = ExtendInputWithOnes(X);
            var output = Matrix<double>.Build.DenseOfArray(y);

            var coefficients = ((input.Transpose() * input).Inverse() *
input.Transpose() * output)
                .Transpose().Row(0);
            _b = coefficients.ElementAt(0);
            _w = SubArray(coefficients.ToArray(), 1, X.GetLength(1));
        }

        public double Predict(double[,] x)
        {
            var input = Matrix<double>.Build.DenseOfArray(x).Transpose();
            var w = Vector<double>.Build.DenseOfArray(_w);
            return input.Multiply(w).ToArray().Sum() + _b;
        }

        private Matrix<double> ExtendInputWithOnes(double[,] X)
        {
            var ones = Matrix<double>.Build.Dense(X.GetLength(0), 1, 1d);
            var extendedX = ones.Append(Matrix<double>.Build.DenseOfArray(X));

            return extendedX;
        }

        private double[] SubArray(double[] data, int index, int length)
        {
            double[] result = new double[length];
            Array.Copy(data, index, result, 0, length);
            return result;
        }
    }
}
```

Класс **AddCarPresenter.cs** :

```
using Diplom.Data;
using Diplom.Entity;
using Diplom.Views.Interfaces;
using System;
```

```

using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace Diplom.Presenters
{
    public class AddCarPresenter : IPresenter
    {
        private IAddCarView _carView;
        private IDatabaseContext _db;

        public IView View
        {
            get { return _carView; }
        }

        public AddCarPresenter(IAddCarView carView, IDatabaseContext db)
        {
            _carView = carView;
            _db = db;

            _carView.btnAddClick += _carView_btnAddClick;
            _carView.btnDeleteClick += _carView_btnDeleteClick;

            _carView.SuppliersDatasource = _db.Suppliers.ToList();
        }

        private void _carView_btnDeleteClick(object sender, EventArgs e)
        {
            _carView.Close();
        }

        private void _carView_btnAddClick(object sender, EventArgs e)
        {
            if ((new Regex(@"^[a-яё\-\s]{1}[0-9]{3}(?!0{3})[a-яё\-\s]{2}[0-9]{2,3}$")).IsMatch(_carView.Number) == false)
            {
                _carView.ShowDialog("Номер должен соответствовать формату  
- a123бв000 ");
                return;
            }
            if (_carView.Model.Length > 100 || _carView.Model.Length == 0)
            {
                _carView.ShowDialog("Количество символов в поле 'Модель  
автомобиля' не должно превышать 100 или оставаться пустым");
                return;
            }
            if (_carView.Type.Length > 100 || _carView.Type.Length == 0)
            {
                _carView.ShowDialog("Количество символов в поле 'Тип  
автомобиля' не должно превышать 100 или оставаться пустым");
                return;
            }
            foreach (var i in _db.Cars)
            {
                if (i.CarNumber == _carView.Number)
                {

```

```

        _carView.ShowDialog("Номер автомобиля должен быть
уникадъным");
        return;
    }
}
// создаём объект
var car = new Car(_carView.Number, _carView.Model, _carView.Type);
// поиск поставщика по имени на форме
Supplier sp = _db.Suppliers.ToList().FirstOrDefault(c => c.SuppName
== _carView.SupplierName);
car.SuppId = sp.Id;
_db.Cars.Add(car);
// сохраняем контекст бд
_db.SaveChanges();
_carView.ShowDialog("Объект успешно добавлен");
_carView.Close();
}

public void Dispose()
{
    _carView.btnAddClick -= _carView_btnAddClick;
    _carView.btnDeleteClick -= _carView_btnDeleteClick;
}
}
}1

```

Класс AddMarkerPresenter.cs :

```

using Diplom.Data;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Presenters
{
    public class AddMarkerPresenter : IPresenter
    {
        private IAddMarkerView _view;
        private IDatabaseContext _db;
        public IView View
        {
            get { return _view; }
        }

        public event EventHandler viewBtnAddClick;

        public double[] Coordinates { get; set; }
        public string Address { get; set; }

        public AddMarkerPresenter(IAddMarkerView view, IDatabaseContext db)
        {
            _view = view;
            _db = db;

            _view.WaterVendingMachines = _db.WaterVendingMachines.Where(c =>
c.Coordinate != null).ToList();
            _view.Suppliers = _db.Suppliers.Where(c => c.Coordinate !=
null).ToList();

```

```

        _view.btnAddClick += _view_btnAddClick;
    }

    private void _view_btnAddClick(object sender, EventArgs e)
    {
        Coordinates = _view.Coordinates;
        Address = _view.Address;
        viewBtnAddClick?.Invoke(this, null);
    }

    public void Dispose()
    {
        _view.btnAddClick -= _view_btnAddClick;
    }
}
}

```

Класс AddSupplierPresenter.cs :

```

using Diplom.Data;
using Diplom.Entity;
using Diplom.Forms;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Presenters
{
    public class AddSupplierPresenter : IPresenter
    {
        private IAddSupplierView _view;
        private IDatabaseContext _db;
        private IAddMarkerView _markerView;

        public IView View
        {
            get { return _view; }
        }

        public double[] Coordinates { get; set; }
        public string Address { get; set; }

        public AddSupplierPresenter(IAddSupplierView view, IDatabaseContext db,
            IAddMarkerView markerView)
        {
            _view = view;
            _db = db;
            _markerView = markerView;

            _view.btnAddClick += _view_btnAddClick;
            _view.btnDeleteClick += _view_btnDeleteClick;
            _view.button1Click += _view_button1Click;
        }

        private void _view_button1Click(object sender, EventArgs e)
        {
            AddMarkerPresenter addMarkerPresenter = new
            AddMarkerPresenter(_markerView, _db);
        }
    }
}

```

```

        addMarkerPresenter.viewBtnAddClick +=
AddMarkerPresenter_viewBtnAddClick;
        addMarkerPresenter.View.ShowDialog();
    }

    private void AddMarkerPresenter_viewBtnAddClick(object sender, EventArgs
e)
    {
        AddMarkerPresenter addMarkerPresenter = (AddMarkerPresenter)sender;
        addMarkerPresenter.viewBtnAddClick -=
AddMarkerPresenter_viewBtnAddClick;
        Coordinates = addMarkerPresenter.Coordinates;
        Address = addMarkerPresenter.Address;
        addMarkerPresenter.View.Close();
    }

    private void _view_btnDeleteClick(object sender, EventArgs e)
    {
        _view.Close();
    }

    private void _view_btnAddClick(object sender, EventArgs e)
    {
        Supplier supplier = new Supplier(_view.SupplierName, Address,
$"{Coordinates[0]} {Coordinates[1]}");
        _db.Suppliers.Add(supplier);
        _db.SaveChanges();
        _view.Close();
    }

    public void Dispose()
    {
        _view.btnAddClick -= _view_btnAddClick;
        _view.btnDeleteClick -= _view_btnDeleteClick;
        _view.button1Click -= _view_button1Click;
    }
}
}

```

Класс AddWaterMachinePresenter.cs :

```

using Diplom.Data;
using Diplom.Entity;
using Diplom.Forms;
using Diplom.Views.Forms;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;

namespace Diplom.Presenters
{
    public class AddWaterMachinePresenter : IPresenter
    {
        private IAddWaterMachineView _view;
        private ModelContext _db;
        public IView View
        {
            get { return _view; }
        }
    }
}

```

```

public double[] Coordinates { get; set; }
public string Address { get; set; }

public AddWaterMachinePresenter(IAddWaterMachineView view, ModelContext
db)
{
    _view = view;
    _db = db;

    _view.btnAddClick += _view_btnAddClick;
    _view.btnDeleteClick += _view_btnDeleteClick;
    _view.button1Click += _view_button1Click;
    _view.iconButton1Click += _view_iconButton1Click;
}

private void _view_iconButton1Click(object sender, EventArgs e)
{
    if(string.IsNullOrEmpty(_view.Capacity) || Coordinates == null ||
Address == "")
    {
        _view.ShowDialog("Заполните все данные для анализа");
        return;
    }
    try
    {
        List<int> ids = new List<int>();
        List<double[]> inputs = new List<double[]>();
        List<double[]> sum = new List<double[]>();
        var _wm = _db.WaterVendingMachines.Local.Where(d =>
d.PurchaseLog != null && d.PurchaseLog.Count != 0).ToList();
        foreach (WaterVendingMachine m in _wm)
        {
            ids.Add(m.Id);
            var _sum = m.PurchaseLog.Sum(c => c.TotalPrice);
            var _days = m.PurchaseLog.Select(entry =>
entry.PurchaseDate.Date).Distinct().Count();
            var _avg = _sum / _days;
            sum.Add(new double[] { (double)_avg });
            var _dist = MainForm._distanceToCentre(new double[] {
double.Parse(m.Coordinate.Split().First()),double.Parse(m.Coordinate.Split().Las
t())
            }, new double[] { 53.194049, 45.016735 });
            inputs.Add(new double[] { (double)m.Price, _dist });
        }
        IRegressView rf = new RegressForm();
        rf.DataGridViewRows.Clear();
        for (int i = 0; i < ids.Count; ++i)
            rf.DataGridViewRows.Add(ids[i], inputs[i][1], inputs[i][0],
sum[i][0]);
        var x = MainForm.MultRegress(inputs, sum, (double)_view.Price,
MainForm._distanceToCentre(Coordinates, new double[] { 53.194049, 45.016735 }));
        rf.DataGridViewRows.Add("New",
MainForm._distanceToCentre(Coordinates, new double[] { 53.194049, 45.016735 }),
(double)_view.Price, Math.Round(x, 4));
        rf.DataGridViewRows[rf.RowsCount - 1].DefaultCellStyle.BackColor
= Color.Green;
        rf.ShowDialog();
    }
    catch
    {
        _view.ShowDialog("Недостаточно водоматов для
прогнозирования");
    }
}

```

```

        return;
    }
}

private void _view_button1Click(object sender, EventArgs e)
{
    AddMarkerPresenter addMarkerPresenter = new AddMarkerPresenter(new
AddMarkerForm(), _db);
    addMarkerPresenter.viewBtnAddClick +=
AddMarkerPresenter_viewBtnAddClick;
    addMarkerPresenter.View.ShowDialog();
}

private void AddMarkerPresenter_viewBtnAddClick(object sender, EventArgs
e)
{
    AddMarkerPresenter addMarkerPresenter = (AddMarkerPresenter)sender;
    addMarkerPresenter.viewBtnAddClick -=
AddMarkerPresenter_viewBtnAddClick;
    Coordinates = addMarkerPresenter.Coordinates;
    Address = addMarkerPresenter.Address;
    addMarkerPresenter.View.Close();
}

private void _view_btnDeleteClick(object sender, EventArgs e)
{
    _view.Close();
}

private void _view_btnAddClick(object sender, EventArgs e)
{
    if (Address == null || _view.Capacity == null)
    {
        _view.ShowDialog("Заполните все данные");
        return;
    }
    WaterVendingMachine m = new WaterVendingMachine(Address,
int.Parse(_view.Capacity), _view.Price,
    $"{Coordinates[0]} {Coordinates[1]}",
    _db.PurchaseLogs.Count() > 0 ?
    DateTime.Parse(_db.PurchaseLogs.Max(c =>
c.PurchaseDate).ToShortDateString()) :
    DateTime.Now);
    WaterLevelSensor s = new WaterLevelSensor();

    s.Model = "Bosch-00607138";
    s.StatusSensor = true;
    s.StatusWater = m.Capacity;
    s.WaterVendingMachine = m;
    m.Status = s.StatusSensor.ToString();
    m.PurchaseLog = new List<PurchaseLog>();
    _db.WaterLevelSensors.Add(s);
    _db.WaterVendingMachines.Add(m);
    _db.SaveChanges();
    _view.Close();
}

public void Dispose()
{
    _view.btnAddClick -= _view_btnAddClick;
    _view.btnDeleteClick -= _view_btnDeleteClick;
    _view.button1Click -= _view_button1Click;
}

```

```

        _view.iconButton1Click -= _view_iconButton1Click;
    }
}

```

Класс **AnalysisPresenter.cs** :

```

using Diplom.Data;
using Diplom.Views.Interfaces;
using System.Windows.Forms;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Presenters
{
    public class AnalysisPresenter : IPresenter
    {
        public static ModelContext _db;
        private IAnalysisView _view;

        public IView View
        {
            get { return _view; }
        }

        public AnalysisPresenter(IAnalysisView view, ModelContext db)
        {
            _view = view;
            _db = db;

            _view.MinMaxDateRequest += _view_MinMaxDateRequest;
            _view.WaterVendingMachinesRequest +=
            _view_WaterVendingMachinesRequest;
            var w = _db.WaterVendingMachines.ToList();
            _view.WaterVendingMachines = _db.WaterVendingMachines.ToList();
        }

        private void _view_WaterVendingMachinesRequest(AnalysisViewEventArgs e)
        {
            try
            {
                double cur = e.Cur;
                _view.WaterVendingMachines = _db.WaterVendingMachines.ToList();
                _view.Date = _db.PurchaseLogs.Min(c =>
c.PurchaseDate).AddDays(cur).ToShortDateString();
            }
            catch
            {
                MessageBox.Show("Нет водоматов для анализа");
                return;
            }
        }

        private void _view_MinMaxDateRequest(AnalysisViewEventArgs e)
        {
            try
            {
                _view.Max = _db.PurchaseLogs.Local.Max(c => c.PurchaseDate);
                _view.Min = _db.PurchaseLogs.Local.Min(c => c.PurchaseDate);
            }
        }
    }
}

```

```

        catch
        {
            return;
        }
    }

    public void Dispose()
    {
        _view.MinMaxDateRequest -= _view_MinMaxDateRequest;
        _view.WaterVendingMachinesRequest -=
        _view_WaterVendingMachinesRequest;
    }
}

```

Класс CarsPresenter.cs :

```

using Diplom.Data;
using Diplom.Forms;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Presenters
{
    public class CarsPresenter : IPresenter
    {
        private ICarsView _view;
        private ModelContext _db;

        public CarsPresenter(ICarsView view, ModelContext db)
        {
            _view = view;
            _db = db;

            _view.btnAddClick += _carsView_btnAddClick;
            _view.btnDeleteClick += _carsView_btnDeleteClick;

            _db.Cars.Load();
            _view.CarsDatasource = _db.Cars.Local.ToBindingList();
        }

        private void _carsView_btnDeleteClick(object sender, EventArgs e)
        {
            _db.SaveChanges();
            _view.ShowDialog("Объект (-ы) удален (-ы)");
        }

        private void _carsView_btnAddClick(object sender, EventArgs e)
        {
            AddCarPresenter addCarPresenter = new AddCarPresenter(new
            AddCarForm(), _db);
            addCarPresenter.View.ShowDialog();
        }

        public IView View
        {
            get { return _view; }
        }
    }
}

```

```

    }

    public void Dispose()
    {
        _view.btnAddClick -= _carsView_btnAddClick;
        _view.btnDeleteClick -= _carsView_btnDeleteClick;
    }
}

```

Класс **HomePresenter.cs** :

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Diplom.Views.Interfaces;

namespace Diplom.Presenters
{
    public class HomePresenter : IPresenter
    {
        private IHomeView _homeView;

        public HomePresenter(IHomeView homeView)
        {
            _homeView = homeView;
        }

        public IView View { get { return _homeView; } }

        public void Dispose()
        {
        }
    }
}

```

Интерфейс **IPresenter.cs** :

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Diplom.Views.Interfaces;

namespace Diplom.Presenters
{
    public interface IPresenter : IDisposable
    {
        IView View { get; }
    }
}

```

Класс **MainPresenter.cs** :

```

using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Text;
using System.Threading.Tasks;
using Diplom.Data;
using Diplom.Forms;
using Diplom.Views.Interfaces;

namespace Diplom.Presenters
{
    public class MainPresenter : IPresenter
    {
        private IMainView _mainView;
        private IPresenter _currentPresenter;
        private ModelContext _db;

        public MainPresenter(IMainView mainView)
        {
            // Инициализация главного окна и навешивание обработчиков событий
            _mainView = mainView;
            _mainView.btnMapClick += _mainView_btnMapClick;
            _mainView.btnCarClick += _mainView_btnCarClick;
            _mainView.btnSuppClick += _mainView_btnSuppClick;
            _mainView.btnSaleClick += _mainView_btnSaleClick;
            _mainView.btnMachineClick += _mainView_btnMachineClick;
            _mainView.btnHomeClick += _mainView_btnHomeClick;
            _mainView.btnAnalysisClick += _mainView_btnAnalysisClick;

            // Инициализация контекста базы данных
            _db = new ModelContext();

            // Установка главного окна в начальное состояние (на экране будет
окно "Home")
            _currentPresenter = new HomePresenter(new HomeForm());
            _mainView.SetChild(_currentPresenter.View);
        }

        public IView View
        {
            get { return _mainView; }
        }

        private void _mainView_btnAnalysisClick(object sender, EventArgs e)
        {
            // Очистка предыдущего окна
            if (_currentPresenter != null)
            {
                _currentPresenter.Dispose();
            }

            // Открытие окна анализа данных
            _currentPresenter = new AnalysisPresenter(new AnalysisForm(), _db);
            _mainView.SetChild(_currentPresenter.View);
        }

        private void _mainView_btnHomeClick(object sender, EventArgs e)
        {
            // Очистка предыдущего окна
            if (_currentPresenter != null)
            {
                _currentPresenter.Dispose();
            }

            // Открытие окна "Home"

```

```

        _currentPresenter = new HomePresenter(new HomeForm());
        _mainView.SetChild(_currentPresenter.View);
    }

    private void _mainView_btnMachineClick(object sender, EventArgs e)
    {
        // Очистка предыдущего окна
        if (_currentPresenter != null)
        {
            _currentPresenter.Dispose();
        }

        // Открытие окна управления водоразбрызгивающими машинами
        _currentPresenter = new WaterMachinePresenter(new
WaterMachineForm(), _db);
        _mainView.SetChild(_currentPresenter.View);
    }

    private void _mainView_btnSaleClick(object sender, EventArgs e)
    {
        // Очистка предыдущего окна
        if (_currentPresenter != null)
        {
            _currentPresenter.Dispose();
        }

        // Открытие окна управления транзакциями
        _currentPresenter = new TransactionPresenter(new TransactionForm(),
_db);
        _mainView.SetChild(_currentPresenter.View);
    }

    private void _mainView_btnSuppClick(object sender, EventArgs e)
    {
        // Очистка предыдущего окна
        if (_currentPresenter != null)
        {
            _currentPresenter.Dispose();
        }

        // Открытие окна управления поставщиками
        _currentPresenter = new SuppliersPresenter(new SupplierForm(), _db);
        _mainView.SetChild(_currentPresenter.View);
    }

    private void _mainView_btnCarClick(object sender, EventArgs e)
    {
        // Очистка предыдущего окна
        if (_currentPresenter != null)
        {
            _currentPresenter.Dispose();
        }

        // Открытие окна управления автомобилями
        _currentPresenter = new CarsPresenter(new CarsForm(), _db);
        _mainView.SetChild(_currentPresenter.View);
    }

    private void _mainView_btnMapClick(object sender, EventArgs e)
    {
        // Очистка предыдущего окна
        if (_currentPresenter != null)

```

```

    {
        _currentPresenter.Dispose();
    }

    // Открытие окна карты
    _currentPresenter = new MapPresenter(new MapForm(), _db);
    _mainView.SetChild(_currentPresenter.View);
}

public void Dispose()
{
    // Удаление обработчиков событий из главного окна
    _mainView.btnMapClick -= _mainView_btnMapClick;
    _mainView.btnCarClick -= _mainView_btnCarClick;
    _mainView.btnSuppClick -= _mainView_btnSuppClick;
    _mainView.btnSaleClick -= _mainView_btnSaleClick;
    _mainView.btnMachineClick -= _mainView_btnMachineClick;
    _mainView.btnHomeClick -= _mainView_btnHomeClick;
    _mainView.btnAnalysisClick -= _mainView_btnAnalysisClick;

    // Закрытие соединения с базой данных
    _db.Dispose();
}
}
}

```

Класс MapPresenter.cs :

```

using Diplom.Data;
using Diplom.Entity;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Presenters
{
    public class MapPresenter : IPresenter
    {
        private ModelContext _db;
        private IMapView _view;

        public IView View
        {
            get { return _view; }
        }

        public MapPresenter(IMapView view, ModelContext db)
        {
            _view = view;
            _db = db;

            _view.WaterVendingMachines = _db.WaterVendingMachines.Where(c =>
c.Coordinate != null).ToList();
            _view.Suppliers = _db.Suppliers.Where(c => c.Coordinate !=
null).ToList();
            _view.db = db;
        }
    }
}

```

```

        public void Dispose()
        {
        }
    }
}

```

Класс SuppliersPresenter.cs :

```

using Diplom.Data;
using Diplom.Forms;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Presenters
{
    public class SuppliersPresenter : IPresenter
    {
        private ISuppliersView _view;
        private ModelContext _db;
        public IView View
        {
            get { return _view; }
        }

        public SuppliersPresenter(ISuppliersView view, ModelContext db)
        {
            _view = view;
            _db = db;

            _db.Suppliers.Load();
            _view.SuppliersDatasource = _db.Suppliers.Local.ToBindingList();

            _view.btnAddClick += _suppliersView_btnAddClick;
            _view.btnDeleteClick += _suppliersView_btnDeleteClick;
        }

        private void _suppliersView_btnDeleteClick(object sender, EventArgs e)
        {
            _db.SaveChanges();
            _view.ShowDialog("Объект(-ы) удален(-ы)");
        }

        private void _suppliersView_btnAddClick(object sender, EventArgs e)
        {
            AddSupplierPresenter addSupplierPresenter = new
            AddSupplierPresenter(new AddSupplierForm(), _db, new AddMarkerForm());
            addSupplierPresenter.View.ShowDialog();
        }

        public void Dispose()
        {
            _view.btnAddClick -= _suppliersView_btnAddClick;
            _view.btnDeleteClick -= _suppliersView_btnDeleteClick;
        }
    }
}

```

Класс TransactionPresenter.cs :

```

using Diplom.Data;
using Diplom.Entity;
using Diplom.Views.Interfaces;
using MathNet.Numerics.Distributions;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Diplom.Presenters
{
    public class TransactionPresenter : IPresenter
    {
        private ModelContext _db;
        private ITransactionView _view;

        public IView View
        {
            get { return _view; }
        }

        public TransactionPresenter(ITransactionView view, ModelContext db)
        {
            _view = view;
            _db = db;

            _view.btnGenerateClick += _view_btnGenerateClick;

            _db.PurchaseLogs.Load();
            _view.PurchaseLogDatasource =
            _db.PurchaseLogs.Local.ToBindingList();
        }

        private void _view_btnGenerateClick(object sender, EventArgs e)
        {
            var _random = new Random();
            var _machines = _db.WaterVendingMachines.ToList().Where(c =>
c.WaterLevelSensor.FirstOrDefault().StatusSensor == true).ToList();
            var _startDate = _db.PurchaseLogs.FirstOrDefault() == null ?
DateTime.Today : _db.PurchaseLogs.ToList().Max(c =>
c.PurchaseDate.Date).AddDays(1);
            var _endDate = _startDate.AddDays(1);

            foreach (WaterVendingMachine m in _machines)
            {
                var rang = m.Rang;
                double sin=0;
                switch (rang)
                {
                    case 0:
                        sin = Math.PI / 5;
                        m.Rang = 1;
                        break;
                    case 1:
                        sin = Math.PI / 4;
                        m.Rang = 2;
                        break;
                    case 2:
                        sin = Math.PI / 3;

```

```

        m.Rang = 3;
        break;
    case 3:
        sin = Math.PI / 2;
        m.Rang = 0;
        break;
    default:
        break;
}
var n = Math.Sin(sin) * (100*(((int)m.Price%5)+1) * (3000 /
(MainForm._distanceToCentre(new double[] {
double.Parse(m.Coordinate.Split().First()),double.Parse(m.Coordinate.Split().Las
t()))
    }, new double[] { 53.194049, 45.016735 })))));
n += new Random().NextDouble() * (n / 40 + n / 40) - n/40;
var value = n;
List<DateTime> _dates = new List<DateTime>();
while (n>0)
{
    int range = (int)(_endDate - _startDate).TotalSeconds;
    int randomDays = _random.Next(range);

    DateTime randomDate = _startDate.AddSeconds(randomDays);

    var _vol = new
Random((int)DateTime.Now.Ticks).NextDouble()*(value/20 - value/100)+ value/100;
    _vol = _vol>n ? n : _vol;
    if (m.WaterLevelSensor.FirstOrDefault().StatusWater - _vol <
0)
    {

        m.WaterLevelSensor.FirstOrDefault().StatusSensor =
false;

        break;
    }
    _db.WaterLevelSensors.Where(c => c.WaterMachineId ==
m.Id).ToList().FirstOrDefault().StatusWater -= (float)_vol;
    PurchaseLog _log = new PurchaseLog(m.Id, randomDate,
(float)_vol, (decimal)_vol * m.Price,
m.WaterLevelSensor.FirstOrDefault().StatusWater - (float)_vol);
    _log.WaterVendingMachine = m;
    _log.CurrentWaterBalance =
m.WaterLevelSensor.FirstOrDefault().StatusWater;
    m.ServiceDate = _log.PurchaseDate;
    _db.PurchaseLogs.Add(_log);
    n -= _vol;
}
}
_db.SaveChanges();
}

public void Dispose()
{
    _view.btnGenerateClick -= _view_btnGenerateClick;
}
}
}

```

Класс WaterMachinePresenter.cs :

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Diplom.Data;
using System.Data.Entity;
using Diplom.Views.Interfaces;
using Diplom.Forms;
using Diplom.Entity;
using Diplom.Views.Forms;

namespace Diplom.Presenters
{
    public class WaterMachinePresenter : IPresenter
    {
        private IWaterMachineView _view;
        private ModelContext _db;

        public IView View
        {
            get { return _view; }
        }

        public WaterMachinePresenter(IWaterMachineView view, ModelContext db)
        {
            _view = view;
            _db = db;

            _db.WaterVendingMachines.Load();
            _view.WaterVendingMachinesDatasource =
            _db.WaterVendingMachines.Local.ToBindingList();

            _view.btnAddClick += _view_btnAddClick;
            _view.itemDeleted += _view_itemDeleted;
            _view.allDeleted += _view_allDeleted;
            _view.WtMachineTableCellClick += _view_WtMachineTableCellClick;
            _view.WtMachineTableCellEndEdit += _view_WtMachineTableCellEndEdit;
        }

        private void _view_WtMachineTableCellEndEdit(object sender, EventArgs e)
        {
            _db.SaveChanges();
        }

        private void
        _view_WtMachineTableCellClick(WtMachineTableCellClickEventArgs e)
        {
            var ind = e.Index;
            var wM = _db.WaterVendingMachines.Include(c =>
            c.WaterLevelSensor).ToList();

            List<WaterLevelSensor> sensors = null;
            foreach (var x in wM)
            {
                if (x.Id.ToString() == ind)
                {
                    sensors = x.WaterLevelSensor.ToList();
                    WaterSensorForm.s = sensors.FirstOrDefault();
                    break;
                }
            }
        }
    }
}

```

```

        WaterSensorPresenter waterSensorPresenter = new
WaterSensorPresenter(
            new WaterSensorForm(), _db, sensors);
        waterSensorPresenter.View.ShowDialog();
    }

    private void _view_allDeleted(object sender, EventArgs e)
    {
        _db.SaveChanges();

        _view.ShowDialog("Объект(-ы) удален(-ы)");
    }

    private void _view_itemDeleted(DeleteItemEventArgs e)
    {
        _db.PurchaseLogs.Load();
        var y = _db.WaterVendingMachines
            .FirstOrDefault(c => c.Id.ToString() ==
e.WaterVendingMachineToDelete)
            .PurchaseLog.ToList();
        var z = _db.WaterVendingMachines
            .FirstOrDefault(c => c.Id.ToString() ==
e.WaterVendingMachineToDelete)
            .WaterLevelSensor.ToList();
        foreach (var x in z)
        {
            _db.WaterLevelSensors.Remove(x);
        }
        foreach (var a in y)
        {
            _db.PurchaseLogs.Remove(a);
        }
    }

    private void _view_btnAddClick(object sender, EventArgs e)
    {
        AddWaterMachinePresenter addWaterMachinePresenter = new
AddWaterMachinePresenter(new AddWaterMachineForm(), _db);
        addWaterMachinePresenter.View.ShowDialog();
    }

    public void Dispose()
    {
        _view.btnAddClick -= _view_btnAddClick;
        _view.itemDeleted -= _view_itemDeleted;
        _view.allDeleted -= _view_allDeleted;
        _view.WtMachineTableCellClick -= _view_WtMachineTableCellClick;
        _view.WtMachineTableCellEndEdit -= _view_WtMachineTableCellEndEdit;
    }
}

```

Класс **WaterSensorPresenter.cs** :

```

using Diplom.Data;
using Diplom.Entity;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Diplom.Presenters
{
    public class WaterSensorPresenter : IPresenter
    {
        private ModelContext _db;
        private IWaterSensorView _view;
        public IView View
        {
            get { return _view; }
        }

        public WaterSensorPresenter(IWaterSensorView view, ModelContext db,
            ICollection<WaterLevelSensor> sensors)
        {
            _view = view;
            _db = db;

            _view.WaterLevelSensorsDatasource = sensors;

            _view.btnCloseClick += _view_btnCloseClick;

            private void _view_btnCloseClick(object sender, EventArgs e)
            {
                _db.SaveChanges();
                _view.Close();
            }

            public void Dispose()
            {
                _view.btnCloseClick -= _view_btnCloseClick;
            }
        }
    }
}

```

Интерфейс IAddCarView.cs :

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Diplom.Entity;

namespace Diplom.Views.Interfaces
{
    public interface IAddCarView : IView
    {
        IEnumerable<Supplier> SuppliersDatasource { set; }
        string SupplierName { get; }
        string Number { get; }
        string Model { get; }
        string Type { get; }
        event EventHandler btnAddClick;
        event EventHandler btnDeleteClick;
    }
}

```

Интерфейс IAddMarkerView.cs :

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Diplom.Entity;

namespace Diplom.Views.Interfaces
{
    public interface IAddMarkerView : IView
    {
        double[] Coordinates { get; set; }
        string Address { get; set; }

        List<WaterVendingMachine> WaterVendingMachines { get; set; }
        List<Supplier> Suppliers { get; set; }

        event EventHandler btnAddClick;
    }
}

```

Интерфейс IAddSupplierView.cs :

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Views.Interfaces
{
    public interface IAddSupplierView : IView
    {
        string SupplierName { get; }

        event EventHandler btnAddClick;
        event EventHandler btnDeleteClick;
        event EventHandler button1Click;
    }
}

```

Интерфейс IAddWaterMachineView.cs :

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Views.Interfaces
{
    public interface IAddWaterMachineView : IView
    {
        string Capacity { get; }
        decimal Price { get; }
        event EventHandler btnAddClick;
        event EventHandler btnDeleteClick;
        event EventHandler button1Click;
        event EventHandler iconButton1Click;
    }
}

```

Интерфейс **IAalysisView.cs** :

```
using Diplom.Entity;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Views.Interfaces
{
    public delegate void AnalysisViewEventHandler(AnalysisViewEventArgs e);
    public interface IAnalysisView : IView
    {
        List<WaterVendingMachine> WaterVendingMachines { get; set; }
        string Date { get; set; }
        DateTime Max { get; set; }
        DateTime Min { get; set; }

        event AnalysisViewEventHandler WaterVendingMachinesRequest;
        event AnalysisViewEventHandler MinMaxDateRequest;
    }

    public class AnalysisViewEventArgs
    {
        public AnalysisViewEventArgs(double cur)
        {
            Cur = cur;
        }
        public double Cur { get; set; }
    }
}
```

Интерфейс **ICarsView.cs** :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Diplom.Entity;

namespace Diplom.Views.Interfaces
{
    public interface ICarsView : IView
    {
        IEnumerable<Car> CarsDatasource { set; }

        event EventHandler btnAddClick;
        event EventHandler btnDeleteClick;
    }
}
```

Интерфейс **IHomeView.cs** :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Diplom.Views.Interfaces
{
    public interface IHomeView : IView
    {
    }
}
```

Интерфейс IMainView.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Views.Interfaces
{
    public interface IMainView : IView
    {
        void SetChild(IView child);

        event EventHandler btnMapClick;
        event EventHandler btnCarClick;
        event EventHandler btnSuppClick;
        event EventHandler btnSaleClick;
        event EventHandler btnMachineClick;
        event EventHandler btnHomeClick;
        event EventHandler btnAnalysisClick;
    }
}
```

Интерфейс IMapView.cs :

```
using Diplom.Data;
using Diplom.Entity;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Views.Interfaces
{
    public interface IMapView : IView
    {
        event EventHandler btnRouteClick;
        event EventHandler iconButton4Click;
        event EventHandler iconButton3Click;
        event EventHandler iconButton1Click;
        event EventHandler iconButton5Click;

        List<WaterVendingMachine> WaterVendingMachines { get; set; }
        List<Supplier> Suppliers { get; set; }
        ModelContext db { get; set; }
    }
}
```

Интерфейс IRegressView.cs :

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplom.Views.Interfaces
{
    public interface IRegressView : IView
    {
        DataGridViewRowCollection DataGridViewRows { get; }
        int RowsCount { get; }
        event EventHandler btnCloseClick;
    }
}

```

Интерфейс ISuppliersView.cs :

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Diplom.Entity;

namespace Diplom.Views.Interfaces
{
    public interface ISuppliersView : IView
    {
        IEnumerable<Supplier> SuppliersDatasource { set; }

        event EventHandler btnAddClick;
        event EventHandler btnDeleteClick;
    }
}

```

Интерфейс ITransactionView.cs :

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Diplom.Entity;

namespace Diplom.Views.Interfaces
{
    public interface ITransactionView : IView
    {
        IEnumerable<PurchaseLog> PurchaseLogDatasource { set; }
        event EventHandler btnGenerateClick;
    }
}

```

Интерфейс IView.cs :

```

namespace Diplom.Views.Interfaces
{
    public interface IView
    {

```

```

        void Show();
        void ShowDialog();
        void ShowMessageDialog(string message);
        void Close();
    }
}

```

Интерфейс **IWaterMachineView.cs** :

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Diplom.Entity;

namespace Diplom.Views.Interfaces
{
    public delegate void DeleteItemEventHandler(DeleteItemEventArgs args);
    public delegate void TableCellClickHandler(WtMachineTableCellClickEventArgs args);

    public interface IWaterMachineView : IView
    {
        IEnumerable<WaterVendingMachine> WaterVendingMachinesDatasource { set; }
        event EventHandler btnAddClick;
        event DeleteItemEventHandler itemDeleted;
        event EventHandler allDeleted;
        event TableCellClickHandler WtMachineTableCellClick;
        event EventHandler WtMachineTableCellEndEdit;
    }

    public class DeleteItemEventArgs : EventArgs
    {
        public string WaterVendingMachineToDelete { get; set; }
    }

    public class WtMachineTableCellClickEventArgs : EventArgs
    {
        public string Index { get; set; }
    }
}

```

Интерфейс **IWaterSensorView.cs** :

```

using Diplom.Entity;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Diplom.Views.Interfaces
{
    public interface IWaterSensorView : IView
    {
        IEnumerable<WaterLevelSensor> WaterLevelSensorsDatasource { set; }

        event EventHandler btnCloseClick;
    }
}

```

Класс AddCarForm.cs :

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Data.Entity;
using System.Runtime.InteropServices;
using Diplom.Data;
using Diplom.Views.Interfaces;

namespace Diplom.Forms
{
    public partial class AddCarForm : Form, IAddCarView
    {
        public event EventHandler btnAddClick;
        public event EventHandler btnDeleteClick;

        public AddCarForm()
        {
            InitializeComponent();
        }

        public IEnumerable<Supplier> SuppliersDatasource
        {
            set
            {
                OwnerComboBox.DataSource = value;
                OwnerComboBox.ValueMember = "Id";
                OwnerComboBox.DisplayMember = "SuppName";
            }
        }

        public string Number
        {
            get { return numberTextbox.Text; }
        }

        public string Model
        {
            get { return modelTextbox.Text; }
        }

        public string Type
        {
            get { return typeTextbox.Text; }
        }

        public string SupplierName
        {
            get { return OwnerComboBox.Text; }
        }

        private void AddCarForm_Load(object sender, EventArgs e)
        {}

        private void btnDelete_Click(object sender, EventArgs e)
        {
            btnDeleteClick?.Invoke(sender, e);
        }
    }
}

```

```

private void btnAdd_Click(object sender, EventArgs e)
{
    btnAddClick?.Invoke(sender, e);
}

//Drag Form
[DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
private extern static void ReleaseCapture();
[DllImport("user32.DLL", EntryPoint = "SendMessage")]
private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int
wParam, int lParam);
private void panell1_MouseDown(object sender, MouseEventArgs e)
{
    ReleaseCapture();
    SendMessage(this.Handle, 0x112, 0xf012, 0);
}
protected override void WndProc(ref Message m)
{
    const int WM_NCCALCSIZE = 0x0083;
    if (m.Msg == WM_NCCALCSIZE && m.WParam.ToInt32() == 1)
    {
        return;
    }

    base.WndProc(ref m);
}

void IView.ShowDialog()
{
    ShowDialog();
}

public void ShowMessageDialog(string message)
{
    MessageBox.Show(message);
}
}
}

```

Класс AddMarkerForm.cs :

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Diplom.Data;
using Diplom.Entity;
using Diplom.Views.Interfaces;
using GMap.NET;
using GMap.NET.WindowsForms;
using GMap.NET.WindowsForms.Markers;
using GMap.NET.WindowsForms.ToolTips;
using Yandex.Geocoder;
using Yandex.Geocoder.Enums;

```

```

using Yandex.Geocoder.Extenstions;

namespace Diplom.Forms
{
    public partial class AddMarkerForm : Form, IAddMarkerView
    {
        public double[] Coordinates { get; set; } = new double[2];
        public string Address { get; set; }
        public List<WaterVendingMachine> WaterVendingMachines { get; set; }
        public List<Supplier> Suppliers { get; set; }

        public event EventHandler btnAddClick;

        private GMapOverlay PositionsForUser = new GMapOverlay("ПозицияпоЛКМ");

        public AddMarkerForm()
        {
            InitializeComponent();
        }

        private void AddMarkerForm_Load(object sender, EventArgs e)
        {
        }

        private async Task ReverseGeocode(double x, double y)
        {
            var request = new ReverseGeocoderRequest { Latitude = x, Longitude =
y };
            var client = new GeocoderClient("d606aa9d-f4e7-49dd-b3d0-
2ddc1c5a139e");
            try
            {
                var response = await client.ReverseGeocode(request);
                var firstGeoObject =
response.GeoObjectCollection.FeatureMember.FirstOrDefault();
                var addressComponents =
firstGeoObject.GeoObject.MetadataProperty.GeocoderMetaData.Address.Components;
                Address = "";
                for (int i = 0; i < addressComponents.Count(); i++)
                {
                    if (addressComponents[i] != null)
                        Address += addressComponents[i].Name + ", ";
                }
                Address = Address.Remove(Address.Length - 2);

            }
            catch
            {
                MessageBox.Show("Не удалось подключиться к HTTP - геокодеру");
            }
        }

        private async void btnAdd_Click(object sender, EventArgs e)
        {
            await ReverseGeocode(Coordinates[0], Coordinates[1]);
            btnAddClick?.Invoke(sender, e);
        }

        private void panel3_MouseDown(object sender, MouseEventArgs e)
        {
            ReleaseCapture();
            SendMessage(this.Handle, 0x112, 0xf012, 0);
        }
    }
}

```

```

    }
    //Drag Form
    [DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
    private extern static void ReleaseCapture();
    [DllImport("user32.DLL", EntryPoint = "SendMessage")]
    private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int
wParam, int lParam);
    protected override void WndProc(ref Message m)
    {
        const int WM_NCCALCSIZE = 0x0083;
        if (m.Msg == WM_NCCALCSIZE && m.WParam.ToInt32() == 1)
        {
            return;
        }

        base.WndProc(ref m);
    }

    private void gmap_Load(object sender, EventArgs e)
    {
        // Настройки для компонента GMap
        gmap.Bearing = 0;
        // Перетаскивание правой кнопки мыши
        gmap.CanDragMap = true;
        // Перетаскивание карты левой кнопкой мыши
        gmap.DragButton = MouseButtons.Left;

        gmap.GrayScaleMode = true;

        // Все маркеры будут показаны
        gmap.MarkersEnabled = true;
        // Максимальное приближение
        gmap.MaxZoom = 20;
        // Минимальное приближение
        gmap.MinZoom = 2;
        // Курсор мыши в центр карты
        gmap.MouseWheelZoomType =
GMap.NET.MouseWheelZoomType.MousePositionWithoutCenter;

        // Отключение негативного режима
        gmap.NegativeMode = false;
        // Разрешение полигонов
        gmap.PolygonsEnabled = true;
        // Разрешение маршрутов
        gmap.RoutesEnabled = true;
        // Скрытие внешней сетки карты
        gmap.ShowTileGridLines = false;
        // При загрузке 10-кратное увеличение
        gmap.Zoom = 15;

        // Чья карта используется
        gmap.MapProvider = GMap.NET.MapProviders.GMapProviders.GoogleMap;
        GMap.NET.GMaps.Instance.Mode = GMap.NET.AccessMode.ServerAndCache;
        // Создаём новый список маркеров
        GMapOverlay markersOverlay = new GMapOverlay("markers");

        var _mCoor = WaterVendingMachines;
        var _sCoor = Suppliers;

        gmap.Position = new GMap.NET.PointLatLng(53.194049, 45.016735);
        foreach (WaterVendingMachine m in _mCoor)
        {

```

```

        double _latitude = double.Parse(m.Coordinate.Split().First());
        double _longitude = double.Parse(m.Coordinate.Split().Last());

        GMarkerGoogle marker = new GMarkerGoogle(new
PointLatLng(_latitude, _longitude), Properties.Resources.wtm); //new
Bitmap(@"D:\диплом\CourseWork_App\Diplom\Resources\wtm.png"
        marker.ToolTip = new GMapRoundedToolTip(marker);
        marker.ToolTipText = m.Location;
        markersOverlay.Markers.Add(marker);
        gmap.Overlays.Add(markersOverlay);
    }

    foreach (Supplier m in _sCoor)
    {
        double _latitude = double.Parse(m.Coordinate.Split().First());
        double _longitude = double.Parse(m.Coordinate.Split().Last());
        GMarkerGoogle marker = new GMarkerGoogle(new
PointLatLng(_latitude, _longitude), Properties.Resources.sup); // new
Bitmap(@"D:\диплом\CourseWork_App\Diplom\Resources\sup.png"));
        marker.ToolTip = new GMapRoundedToolTip(marker);
        marker.ToolTipText = m.SuppAddress;
        markersOverlay.Markers.Add(marker);
        gmap.Overlays.Add(markersOverlay);
    }
}

GMarkerGoogle MarkerWithMyPosition;
private void gmap_MouseDoubleClick_1(object sender, MouseEventArgs e)
{
    if (e.Button != MouseButton.Left)
        return;

    PositionsForUser.Markers.Remove(MarkerWithMyPosition);
    gmap.Overlays.Add(PositionsForUser);
    double y = gmap.FromLocalToLatLng(e.X, e.Y).Lng;
    double x = gmap.FromLocalToLatLng(e.X, e.Y).Lat;

    MarkerWithMyPosition = new GMarkerGoogle(new PointLatLng(x, y),
GMarkerGoogleType.blue_pushpin);
    MarkerWithMyPosition.ToolTip = new
GMapRoundedToolTip(MarkerWithMyPosition);
    MarkerWithMyPosition.ToolTipText = "Метка пользователя";
    PositionsForUser.Markers.Add(MarkerWithMyPosition);

    Coordinates[0] = x;
    Coordinates[1] = y;
}

void IView.ShowDialog()
{
    ShowDialog();
}

public void ShowMessageDialog(string message)
{
    MessageBox.Show(message);
}
}
}

```

Класс AddSupplierForm.cs :

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Diplom.Data;
using Diplom.Entity;
using Diplom.Views.Interfaces;

namespace Diplom.Forms
{
    public partial class AddSupplierForm : Form, IAddSupplierView
    {
        public event EventHandler btnAddClick;
        public event EventHandler btnDeleteClick;
        public event EventHandler button1Click;

        public string SupplierName
        {
            get { return nameTextbox.Text; }
        }

        public AddSupplierForm()
        {
            InitializeComponent();
        }
        //Drag Form
        [DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
        private extern static void ReleaseCapture();
        [DllImport("user32.DLL", EntryPoint = "SendMessage")]
        private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int
wParam, int lParam);
        private void panell1_MouseDown(object sender, MouseEventArgs e)
        {
            ReleaseCapture();
            SendMessage(this.Handle, 0x112, 0xf012, 0);
        }
        protected override void WndProc(ref Message m)
        {
            const int WM_NCCALCSIZE = 0x0083;
            if (m.Msg == WM_NCCALCSIZE && m.WParam.ToInt32() == 1)
            {
                return;
            }

            base.WndProc(ref m);
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            btnDeleteClick?.Invoke(sender, e);
        }
    }
}

```

```

private void button1_Click(object sender, EventArgs e)
{
    button1Click?.Invoke(sender, e);
}

private void btnAdd_Click(object sender, EventArgs e)
{
    btnAddClick?.Invoke(sender, e);
}

void IView.ShowDialog()
{
    ShowDialog();
}

public void ShowMessageDialog(string message)
{
    MessageBox.Show(message);
}

private void AddSupplierForm_Load(object sender, EventArgs e)
{
}
}
}

```

Класс **AddWaterMachineForm.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 Diplom.Data;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using Diplom.Views.Interfaces;

namespace Diplom.Forms
{
    public partial class AddWaterMachineForm : Form, IAddWaterMachineView
    {
        public string Capacity
        {
            get { return capacityTextbox.Text; }
        }

        public decimal Price
        {
            get { return numericUpDown1.Value; }
        }

        public AddWaterMachineForm()
        {
            InitializeComponent();
        }

        public event EventHandler btnAddClick;
    }
}

```

```

public event EventHandler btnDeleteClick;
public event EventHandler button1Click;
public event EventHandler iconButton1Click;

private void AddWaterMachineForm_Load(object sender, EventArgs e)
{

}

private void panell1_MouseDown(object sender, MouseEventArgs e)
{
    ReleaseCapture();
    SendMessage(this.Handle, 0x112, 0xf012, 0);
}
//Drag Form
[DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
private extern static void ReleaseCapture();
[DllImport("user32.DLL", EntryPoint = "SendMessage")]
private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int
wParam, int lParam);
protected override void WndProc(ref Message m)
{
    const int WM_NCCALCSIZE = 0x0083;
    if (m.Msg == WM_NCCALCSIZE && m.WParam.ToInt32() == 1)
    {
        return;
    }

    base.WndProc(ref m);
}

private void btnAdd_Click(object sender, EventArgs e)
{
    btnAddClick?.Invoke(sender, e);
}

private void btnDelete_Click(object sender, EventArgs e)
{
    btnDeleteClick?.Invoke(sender, e);
}

private void button1_Click(object sender, EventArgs e)
{
    button1Click?.Invoke(sender, e);
}

private void iconButton1_Click(object sender, EventArgs e)
{
    iconButton1Click?.Invoke(sender, e);
}

void IView.ShowDialog()
{
    ShowDialog();
}

public void ShowMessageDialog(string message)
{
    MessageBox.Show(message);
}

private void panell1_Paint(object sender, PaintEventArgs e)

```

```

    {
    }
}

```

Класс **AnalysisForm.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.Windows.Forms.Integration;
using Diplom.Data;
using Diplom.Entity;
using Diplom.Views.Interfaces;
using LiveCharts;
using LiveCharts.Wpf;

namespace Diplom.Forms
{
    public partial class AnalysisForm : Form, IAnalysisView
    {

        public event AnalysisViewEventHandler WaterVendingMachinesRequest;
        public event AnalysisViewEventHandler MinMaxDateRequest;

        public List<WaterVendingMachine> WaterVendingMachines { get; set; }
        public string Date { get; set; }
        public DateTime Max { get; set; }
        public DateTime Min { get; set; }
        public object ARModel { get; private set; }

        public AnalysisForm()
        {
            InitializeComponent();
            x = new LiveCharts.WinForms.CartesianChart()
            {
                BackColor = Color.White,
                Dock = DockStyle.Fill
            };
            comboBox1.SelectedItem = comboBox1.Items[0];
        }

        private void AnalysisForm_Load(object sender, EventArgs e)
        {
            LoadTableVolume();
        }

        bool fl;
        private double cur = 0;
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (WaterVendingMachines == null)
                return;

            fl = comboBox1.SelectedIndex == 0 ? false : true;

```

```

        if (fl)
        {
            LoadChartPrice();
            LoadTablePrice();
        }
        else
        {
            LoadTableVolume();
            LoadChartVolume();
        }
    }

    private void LoadTablePrice()
    {
        dataGridView1.Rows.Clear();
        dataGridView1.Refresh();
        dataGridView1.Columns[2].HeaderText = "Суточный доход";
        var ids = new List<int>();
        var sum = new List<decimal>();
        var _wm = WaterVendingMachines;
        var _date = Date;
        foreach (var m in _wm)
        {
            ids.Add(m.Id);
            sum.Add(m.PurchaseLog.Where(c => c.PurchaseDate.Day ==
DateTime.Parse(_date).Day).Select(c => c.TotalPrice).Sum());
        }
        for (int i = 0; i < ids.Count; ++i)
            dataGridView1.Rows.Add(ids[i], _date, sum[i]);
    }

    private void LoadTableVolume()
    {
        dataGridView1.Rows.Clear();
        dataGridView1.Refresh();
        dataGridView1.Columns[2].HeaderText = "Расход воды";
        var ids = new List<int>();
        var sum = new List<decimal>();

        AnalysisViewEventArgs args = new AnalysisViewEventArgs(cur);
        WaterVendingMachinesRequest?.Invoke(args);

        var _wm = WaterVendingMachines;
        var _date = Date;
        foreach (var m in _wm)
        {
            ids.Add(m.Id);
            sum.Add(m.PurchaseLog.Where(c => c.PurchaseDate.Day ==
DateTime.Parse(_date).Day).Select(c => (decimal)c.PurchaseVolume).Sum());
        }
        for (int i = 0; i < ids.Count; ++i)
            dataGridView1.Rows.Add(ids[i], _date, sum[i]);
        //cur = int.Parse((db.PurchaseLogs.Max(c => c.PurchaseDate) -
db.PurchaseLogs.Where(c => c.PurchaseDate ==
DateTime.Parse(_date)).First().PurchaseDate).TotalDays.ToString());
    }

    private void btnAdd_Click(object sender, EventArgs e)
    {
        if (cur == 0)
        {
            return;
        }
    }

```

```

    }
    --cur;
    if (fl)
    {
        LoadChartPrice();
        LoadTablePrice();
    }
    else
    {
        LoadTableVolume();
        LoadChartVolume();
    }
}

private void iconButton2_Click(object sender, EventArgs e)
{
    MinMaxDateRequest?.Invoke(null);
    if (cur >= (Max - Min).TotalDays)
    {
        return;
    }
    ++cur;

    if (fl)
    {
        LoadChartPrice();
        LoadTablePrice();
    }
    else
    {
        LoadTableVolume();
        LoadChartVolume();
    }
}

private void iconButton1_Click(object sender, EventArgs e)
{
    panel3.Controls.Clear();
    panel3.Controls.Add(dataGridView1);
    if (fl)
        LoadTablePrice();
    else
        LoadTableVolume();
}

private void LoadChartVolume()
{
    x.AxisX.Clear();
    x.AxisY.Clear();
    x.Series.Clear();
    ColumnSeries col = new ColumnSeries()
    {
        DataLabels = true,
        Values = new ChartValues
        <int>(),
        LabelPoint = point => point.Y.ToString()+" литра(-ов)",
    };
    Axis ax = new Axis() { Separator = new Separator() { Step = 1,
IsEnabled = false },
        Title = "ID водомата"
    };
}

```

```

ax.Labels = new List<string>();

AnalysisViewEventArgs args = new AnalysisViewEventArgs(cur);
WaterVendingMachinesRequest?.Invoke(args);

var _wm = WaterVendingMachines;
var _date = Date;
foreach (var m in _wm)
{
    var z = m.PurchaseLog.Where(c => c.PurchaseDate.Day ==
DateTime.Parse(_date).Day).Sum(c => (decimal)c.PurchaseVolume);
    col.Values.Add((int) z);
    ax.Labels.Add(m.Id.ToString());
}

col.Title = _date.ToString();
x.Series.Add(col);

x.AxisX.Add(ax);
x.AxisY.Add(new Axis
{
    Title = "Расход воды",
    LabelFormatter = v => v.ToString(),
    Separator = new Separator()
});
}
private void LoadChartPrice()
{
    try
    {
        x.AxisX.Clear();
        x.AxisY.Clear();
        x.Series.Clear();
        ColumnSeries col = new ColumnSeries()
        {
            DataLabels = true,
            Values = new ChartValues
            <int>(),
            LabelPoint = point => point.Y.ToString() + " рубля(-ей)",
        };
        Axis ax = new Axis()
        {
            Separator = new Separator() { Step = 1, IsEnabled = false },
            Title = "ID водомата"
        };
        ax.Labels = new List<string>();

        AnalysisViewEventArgs args = new AnalysisViewEventArgs(cur);
        WaterVendingMachinesRequest?.Invoke(args);

        var _wm = WaterVendingMachines;
        var _date = Date;

        foreach (var m in _wm)
        {
            var z = m.PurchaseLog.Where(c => c.PurchaseDate.Day ==
DateTime.Parse(_date).Day).Select(c => c.TotalPrice).Sum();
            col.Values.Add((int)z);
            ax.Labels.Add(m.Id.ToString());
        }
    }
}

```

```

        col.Title = _date.ToString();
        x.Series.Add(col);

        x.AxisX.Add(ax);
        x.AxisY.Add(new Axis
        {
            Title = "Суточный доход",
            LabelFormatter = v => v.ToString(),
            Separator = new Separator()
        });
    }
    catch
    {
        return;
    }
}
LiveCharts.WinForms.CartesianChart x;

private void iconButton3_Click(object sender, EventArgs e)
{
    panel3.Controls.Clear();
    if (fl)
        LoadChartPrice();
    else
        LoadChartVolume();

    panel3.Controls.Add(x);
}

void IView.ShowDialog()
{
    ShowDialog();
}

public void ShowMessageDialog(string message)
{
    MessageBox.Show(message);
}

private void iconButton4_Click(object sender, EventArgs e)
{
    Views.Forms.ARModelForm aR = new Views.Forms.ARModelForm();
    aR.ShowDialog();
}
}
}

```

Класс **ARModelForm.cs** :

```

using Diplom.Data;
using Diplom.MLRegressor;
using Diplom.Presenters;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices;

```

```

using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using LiveCharts.WinForms;
using LiveCharts.Wpf;
using LiveCharts;

namespace Diplom.Views.Forms
{
    public partial class ARModelForm : Form
    {
        public ModelContext db;
        bool flag;
        List<double> data;
        public LiveCharts.WinForms.CartesianChart chart;
        DateTime min;
        DateTime max;
        double prediction;
        List<List<Entity.PurchaseLog>> l;
        public ARModelForm()
        {
            db = AnalysisPresenter._db;
            InitializeComponent();
            var logs = db.WaterVendingMachines.Where(c =>
c.PurchaseLog.Select(entry => entry.PurchaseDate.Day).Distinct().Count() >=
8).ToList();
            comboBox1.DataSource = logs.Select(c => c.Id).ToList();
        }
        private void panell1_MouseDown(object sender, MouseEventArgs e)
        {
            ReleaseCapture();
            SendMessage(this.Handle, 0x112, 0xf012, 0);
        }
        //Drag Form
        [DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
        private extern static void ReleaseCapture();
        [DllImport("user32.DLL", EntryPoint = "SendMessage")]
        private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int
wParam, int lParam);
        protected override void WndProc(ref Message m)
        {
            const int WM_NCCALCSIZE = 0x0083;
            if (m.Msg == WM_NCCALCSIZE && m.WParam.ToInt32() == 1)
            {
                return;
            }

            base.WndProc(ref m);
        }

        private void ARModelForm_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "waterSystemDataSet3.PurchaseLog". При необходимости она может быть перемещена
или удалена.

            this.purchaseLogTableAdapter.Fill(this.waterSystemDataSet3.PurchaseLog);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "waterSystemDataSet3.WaterVendingMachine". При необходимости она может быть
перемещена или удалена.

```

```

this.waterVendingMachineTableAdapter1.Fill(this.waterSystemDataSet3.WaterVending
Machine);
    // TODO: данная строка кода позволяет загрузить данные в таблицу
"waterSystemDataSet.WaterVendingMachine". При необходимости она может быть
перемещена или удалена.

this.waterVendingMachineTableAdapter.Fill(this.waterSystemDataSet.WaterVendingMa
chine);
    LoadTable();
    flag = true;
}

private void btnClose_Click(object sender, EventArgs e)
{
    Close();
}
private void LoadTable()
{
    if (chart != null)
        panel2.Controls.Remove(chart);
    dataGridView1.Rows.Clear();
    DateTime y;
    var x = comboBox1.Text;
    var logs = db.WaterVendingMachines.Where(c => c.Id.ToString() ==
x).FirstOrDefault().PurchaseLog.ToList();
    min = logs.Min(c => c.PurchaseDate.Date);
    max = logs.Max(c => c.PurchaseDate.Date).AddDays(1);
    l = new List<List<Entity.PurchaseLog>>();
    for (DateTime i = min; i < max; i = i.AddDays(1))
    {
        l.Add(logs.Where(c => c.PurchaseDate.Date == i.Date).ToList());
    }
    int j = 0;
    data = new List<double>();
    foreach (var lg in l)
    {
        var sum = Math.Round(lg.Sum(c => c.PurchaseVolume), 1);
        data.Add(sum);
        ++j;
        dataGridView1.Rows.Add(x,
lg.FirstOrDefault().PurchaseDate.ToShortDateString(), sum);
    }
    int order = 4;
    AutoRegression model = new AutoRegression(order);
    model.Fit(data.ToArray());
    double[] testData = new double[] {data[l.Count-4], data[l.Count -
3], data[l.Count - 2], data[l.Count - 1] };
    prediction = model.Predict(testData);
    prediction = prediction < 0 ? 0 : prediction;
    dataGridView1.Rows.Add(x, max.ToShortDateString(), prediction);
    dataGridView1.Rows[dataGridView1.RowCount -
1].DefaultCellStyle.BackColor = Color.Green;
}

void LoadChart()
{
    var m = db.WaterVendingMachines.Where(c => c.Id.ToString() ==
comboBox1.Text).FirstOrDefault();
    var rang = m.Rang;
    double sin = 0;

```

```

chart = new LiveCharts.WinForms.CartesianChart();
chart.Series = new SeriesCollection();
chart.Series.Add(new LineSeries
{
    Title = "Осталось в резервуаре",
    Values = new ChartValues<double>(),
});
//chart.Series.Add(new LineSeries
//{
//    Title = comboBox1.Text,
//    Values = new ChartValues<double>(),
//});
//List<double> ls = new List<double>();
//var rk = rang + 1 == 4 ? 0 : rang + 1;
//switch (rk)
//{
//    case 0:
//        sin = Math.PI / 2;
//        break;
//    case 1:
//        sin = Math.PI / 5;
//        break;
//    case 2:
//        sin = Math.PI / 4;
//        break;
//    case 3:
//        sin = Math.PI / 3;
//        break;
//    default:
//        break;
//}
//var z = Math.Sin(sin) * (100 * (((int)m.Price % 5) + 1) * (3000 /
(MainForm._distanceToCentre(new double[] {
double.Parse(m.Coordinate.Split().First()), double.Parse(m.Coordinate.Split().Last())
//    }, new double[] { 53.194049, 45.016735 })))));
for (int i = 0; i < data.Count; ++i)
{
    var x = l[i].Max(c => c.CurrentWaterBalance);
    var y = l[i].Min(c => c.CurrentWaterBalance);
    chart.Series[0].Values.Add((double)x);
    chart.Series[0].Values.Add((double)y);
    //switch (rang)
    //{
    //    case 0:
    //        sin = Math.PI / 2;
    //        rang = 3;
    //        break;
    //    case 1:
    //        sin = Math.PI / 5;
    //        rang = 0;
    //        break;
    //    case 2:
    //        sin = Math.PI / 4;
    //        rang = 1;
    //        break;
    //    case 3:
    //        sin = Math.PI / 3;
    //        rang = 2;
    //        break;
    //    default:

```

```

        //          break;
        //}
        //var n = Math.Sin(sin) * (100 * (((int)m.Price % 5) + 1) *
(3000 / (MainForm._distanceToCentre(new double[] {
double.Parse(m.Coordinate.Split().First()),double.Parse(m.Coordinate.Split().Las
t()))
        //}}, new double[] { 53.194049, 45.016735 })))));
        //ls.Add(n);
    }
    //ls.Reverse();
    //ls.Add(z);
    //foreach (var el in ls)
    //    chart.Series[1].Values.Add(el);
    chart.Series[0].Values.Add((double)l[l.Count-1].Min(c =>
c.CurrentWaterBalance));
    chart.Series[0].Values.Add((double)l[l.Count - 1].Min(c =>
c.CurrentWaterBalance) - prediction);
    Axis ax = new Axis()
    {
        LabelFormatter = value => value.ToString()
    };
    chart.AxisY.Add(new Axis
    {
        // Title = "Count",
        MinValue = db.WaterVendingMachines.Where(c => c.Id.ToString() ==
comboBox1.Text).FirstOrDefault().Capacity/(-5),
        LabelFormatter = value => value.ToString()
    });
    ax.Labels = new List<string>();
    foreach (var ll in l)
        ax.Labels.Add(ll.Max(c => c.PurchaseDate).ToShortDateString());

    var _max = max.AddDays(1);
    //for (DateTime i = min; i < _max; i = i.AddDays(1))
    //{
    //    ax.Labels.Add(i.ToShortDateString());
    //}
    chart.AxisX.Add(ax);
    chart.Dock = DockStyle.Fill;
}
private void iconButton1_Click(object sender, EventArgs e)
{
    flag = true;
    LoadTable();
    panel2.Controls.Add(dataGridView1);
}

private void iconButton2_Click(object sender, EventArgs e)
{
    flag = false;
    panel2.Controls.Remove(dataGridView1);
    LoadChart();
    panel2.Controls.Add(chart);
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    LoadTable();
    if (flag) panel2.Controls.Add(dataGridView1);
    else
    {

```

```

        LoadChart();
        panel2.Controls.Add(chart);
    }
}
}

```

Класс **CarsForm.cs** :

```

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

using Diplom.Entity;
using Diplom.Views.Interfaces;

namespace Diplom.Forms
{
    public partial class CarsForm : Form, ICarsView
    {
        public event EventHandler btnAddClick;
        public event EventHandler btnDeleteClick;
        public event EventHandler CarsTableCellEndEdit;
        public CarsForm()
        {
            InitializeComponent();
        }

        public IEnumerable<Car> CarsDatasource
        {
            set
            {
                CarsTable.DataSource = value;
            }
        }

        private void CarsForm_Load(object sender, EventArgs e)
        {
            this.supplierTableAdapter.Fill(this.waterSystemDataSet2.Supplier);
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            btnAddClick?.Invoke(sender, e);
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            if (CarsTable.SelectedRows.Count > 0)
            {
                foreach (DataGridViewRow item in this.CarsTable.SelectedRows)
                {
                    CarsTable.Rows.RemoveAt(item.Index);
                }
                //db.SaveChanges();
            }
        }
    }
}

```

```

        btnDeleteClick?.Invoke(sender, e);
        //MessageBox.Show("Объект (-ы) удален (-ы)");
    }
}

void IView.ShowDialog()
{
    throw new NotImplementedException();
}

public void ShowMessageDialog(string message)
{
    MessageBox.Show(message);
}

private void CarsTable_CellEndEdit(object sender,
DataGridViewCellEventArgs e)
{
    CarsTableCellEndEdit?.Invoke(sender, null);
}
}
}

```

Класс **HomeForm.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 Diplom.Views.Interfaces;

namespace Diplom.Forms
{
    public partial class HomeForm : Form, IHomeView
    {
        private int borderSize = 1;
        public HomeForm()
        {
            InitializeComponent();
            this.Padding = new Padding(borderSize); //Border size
            this.BackColor = Color.FromArgb(98, 102, 244); //Border color
        }

        public void ShowMessageDialog(string message)
        {
            throw new NotImplementedException();
        }

        void IView.ShowDialog()
        {
            throw new NotImplementedException();
        }
    }
}

```

Класс **InfoForm.cs** :

```

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

namespace Diplom.Views.Forms
{
    public partial class InfoForm : Form
    {
        public InfoForm()
        {
            InitializeComponent();
        }
        public void AddText(string s, Color c)
        {
            richTextBox2.SelectionStart = richTextBox2.TextLength;
            richTextBox2.SelectionLength = 0;
            richTextBox2.SelectionColor = c;
            richTextBox2.AppendText(s);
            richTextBox2.SelectionColor = richTextBox2.ForeColor;
        }
        private void ChartForm_Load(object sender, EventArgs e)
        {
        }
        //Drag Form
        [DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
        private extern static void ReleaseCapture();
        [DllImport("user32.DLL", EntryPoint = "SendMessage")]
        private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int
wParam, int lParam);
        private void panell1_MouseDown(object sender, MouseEventArgs e)
        {
            ReleaseCapture();
            SendMessage(this.Handle, 0x112, 0xf012, 0);
        }
        protected override void WndProc(ref Message m)
        {
            const int WM_NCCALCSIZE = 0x0083;
            if (m.Msg == WM_NCCALCSIZE && m.WParam.ToInt32() == 1)
            {
                return;
            }

            base.WndProc(ref m);
        }
        private void btnClose_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}

```

Класс LoadForm.cs :

```

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

namespace Diplom.Views.Forms
{
    public partial class LoadForm : Form
    {
        public LoadForm()
        {
            InitializeComponent();
        }
        //Drag Form
        [DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
        private extern static void ReleaseCapture();
        [DllImport("user32.DLL", EntryPoint = "SendMessage")]
        private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int
wParam, int lParam);
        protected override void WndProc(ref Message m)
        {
            const int WM_NCCALCSIZE = 0x0083;
            if (m.Msg == WM_NCCALCSIZE && m.WParam.ToInt32() == 1)
            {
                return;
            }

            base.WndProc(ref m);
        }
        protected override async void OnLoad(EventArgs e)
        {
            base.OnLoad(e);
            await Task.Delay(3000);
            Close();
        }
    }
}

```

Класс MainForm.cs :

```

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.IO;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

```

```

using GMap.NET;
using GMap.NET.MapProviders;
using GMap.NET.WindowsForms;
using GMap.NET.WindowsForms.Markers;
using GMap.NET.WindowsForms.ToolTips;
using Itinero;
using Itinero.Osm.Vehicles;
using Itinero.IO.Osm;
using Itinero.Osm;
using System.Data.Entity;
using Yandex.Geocoder;
using Yandex.Geocoder.Enums;
using Yandex.Geocoder.Extenstions;
using Diplom.Data;
using Diplom.Entity;
using System.Runtime.InteropServices;
using GeoCoordinatePortable;
using Diplom.MLRegressor;
using Diplom.Views.Interfaces;

namespace Diplom
{

    public partial class MainForm : Form, IMainView
    {

        public event EventHandler btnMapClick;
        public event EventHandler btnCarClick;
        public event EventHandler btnSuppClick;
        public event EventHandler btnSaleClick;
        public event EventHandler btnMachineClick;
        public event EventHandler btnHomeClick;
        public event EventHandler btnAnalysisClick;
    }
}

```

```

private int borderSize = 2;

private Size formSize;

private Form activeForm;

public static Form f;

public MainForm()
{
    InitializeComponent();

    CollapseMenu();

    this.Padding = new Padding(borderSize); //Border size

    this.BackColor = Color.FromArgb(98, 102, 244); //Border color
}

//Drag Form

[DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
public extern static void ReleaseCapture();

[DllImport("user32.DLL", EntryPoint = "SendMessage")]
public extern static void SendMessage(System.IntPtr hWnd, int wMsg, int
wParam, int lParam);

private void Form1_Load(object sender, EventArgs e)
{

}

public static double[,] DoubleTo2(double[][] input )
{
    int rows = input.Length;
    int cols = input[0].Length;

    double[,] output = new double[rows, cols];

    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)

```

```

        {
            output[i, j] = input[i][j];
        }
    }

    return output;
}

public static double MultRegress(List<double[]> _x, List<double[]>
_y, double _newX1, double _newX2)
{
    MultipleLinearRegressor lr = new MultipleLinearRegressor();
    lr.Fit(DoubleTo2(_x.ToArray()), DoubleTo2(_y.ToArray()));
    double[,] _X = new double[,] { { _newX1 }, { _newX2 } };
    var _resY = lr.Predict(_X);
    return _resY;
}

public static double _distanceToCentre(double[] coorX, double[] coorY)
{
    GeoCoordinate _point1 = new GeoCoordinate(coorX[0], coorX[1]);
    GeoCoordinate _point2 = new GeoCoordinate(coorY[0], coorY[1]);

    // Расчет расстояния между точками с использованием функции
    GreatCircleDistance()

    double _distance = _point1.GetDistanceTo(_point2);
    return _distance;
}

private void panelTitle_MouseDown(object sender, MouseEventArgs e)
{
    ReleaseCapture();
    SendMessage(this.Handle, 0x112, 0xf012, 0);
}

protected override void WndProc(ref Message m)
{
    const int WM_NCCALCSIZE = 0x0083; // Standar Title Bar - Snap Window
    const int WM_SYSCOMMAND = 0x0112;

```

```

const int SC_MINIMIZE = 0xF020; //Minimize form (Before)

const int SC_RESTORE = 0xF120; //Restore form (Before)

const int WM_NCHITTEST = 0x0084; //Win32, Mouse Input Notification:
Determine what part of the window corresponds to a point, allows to resize the
form.

const int resizeAreaSize = 10;

#region Form Resize

// Resize/WM_NCHITTEST values

const int HTCLIENT = 1; //Represents the client area of the window

const int HTLEFT = 10; //Left border of a window, allows resize
horizontally to the left

const int HTRIGHT = 11; //Right border of a window, allows resize
horizontally to the right

const int HTTOP = 12; //Upper-horizontal border of a window,
allows resize vertically up

const int HTTOPLEFT = 13; //Upper-left corner of a window border,
allows resize diagonally to the left

const int HTTOPRIGHT = 14; //Upper-right corner of a window border,
allows resize diagonally to the right

const int HTBOTTOM = 15; //Lower-horizontal border of a window,
allows resize vertically down

const int HTBOTTOMLEFT = 16; //Lower-left corner of a window border,
allows resize diagonally to the left

const int HTBOTTOMRIGHT = 17; //Lower-right corner of a window
border, allows resize diagonally to the right

///<Doc> More Information: https://docs.microsoft.com/en-
us/windows/win32/inputdev/wm-nchittest </Doc>

if (m.Msg == WM_NCHITTEST)

{ //If the windows m is WM_NCHITTEST

    base.WndProc(ref m);

    if (this.WindowState == FormWindowState.Normal) //Resize the form
if it is in normal state

    {

        if ((int)m.Result == HTCLIENT) //If the result of the m
(mouse pointer) is in the client area of the window

        {

            Point screenPoint = new Point(m.LParam.ToInt32());
//Gets screen point coordinates(X and Y coordinate of the pointer)

            Point clientPoint = this.PointToClient(screenPoint);
//Computes the location of the screen point into client coordinates

```

```

        if (clientPoint.Y <= resizeAreaSize)//If the pointer is
at the top of the form (within the resize area- X coordinate)

        {

            if (clientPoint.X <= resizeAreaSize) //If the
pointer is at the coordinate X=0 or less than the resizing area(X=10) in

            m.Result = (IntPtr)HTTOPLEFT; //Resize
diagonally to the left

            else if (clientPoint.X < (this.Size.Width -
resizeAreaSize))//If the pointer is at the coordinate X=11 or less than the
width of the form(X=Form.Width-resizeArea)

            m.Result = (IntPtr)HTTOP; //Resize vertically up

            else //Resize diagonally to the right

            m.Result = (IntPtr)HTTOPRIGHT;

        }

        else if (clientPoint.Y <= (this.Size.Height -
resizeAreaSize)) //If the pointer is inside the form at the Y
coordinate(discounting the resize area size)

        {

            if (clientPoint.X <= resizeAreaSize)//Resize
horizontally to the left

            m.Result = (IntPtr)HTLEFT;

            else if (clientPoint.X > (this.Width -
resizeAreaSize))//Resize horizontally to the right

            m.Result = (IntPtr)HTRIGHT;

        }

        else

        {

            if (clientPoint.X <= resizeAreaSize)//Resize
diagonally to the left

            m.Result = (IntPtr)HTBOTTOMLEFT;

            else if (clientPoint.X < (this.Size.Width -
resizeAreaSize)) //Resize vertically down

            m.Result = (IntPtr)HTBOTTOM;

            else //Resize diagonally to the right

            m.Result = (IntPtr)HTBOTTOMRIGHT;

        }

    }

}

```

```

        return;
    }
#endregion

//Remove border and keep snap window
if (m.Msg == WM_NCCALCSIZE && m.WParam.ToInt32() == 1)
{
    return;
}

//Keep form size when it is minimized and restored. Since the form
is resized because it takes into account the size of the title bar and borders.

if (m.Msg == WM_SYSCOMMAND)
{
    /// <see cref="https://docs.microsoft.com/en-
us/windows/win32/menurc/wm-syscommand"/>
    /// Quote:
    /// In WM_SYSCOMMAND messages, the four low - order bits of the
wParam parameter
    /// are used internally by the system.To obtain the correct
result when testing
    /// the value of wParam, an application must combine the value
0xFFFF0 with the
    /// wParam value by using the bitwise AND operator.

    int wParam = (m.WParam.ToInt32() & 0xFFFF0);

    if (wParam == SC_MINIMIZE) //Before
        formSize = this.ClientSize;

    if (wParam == SC_RESTORE) // Restored form(Before)
        this.Size = formSize;
}

base.WndProc(ref m);
}

private void AdjustForm()
{
    switch (this.WindowState)
    {
        case FormWindowState.Maximized: //Maximized form (After)
            this.Padding = new Padding(8, 8, 8, 0);
    }
}

```

```

        break;

    case FormWindowState.Normal: //Restored form (After)
        if (this.Padding.Top != borderSize)
            this.Padding = new Padding(borderSize);
        break;
    }
}

private void CollapseMenu()
{
    if (this.panelMenu.Width > 200) //Collapse menu
    {
        panelMenu.Width = 100;
        pictureBox1.Visible = false;
        btnMenu.Dock = DockStyle.Top;

        foreach (Button menuButton in
panelMenu.Controls.OfType<Button>())
        {
            menuButton.Text = "";
            menuButton.ImageAlign = ContentAlignment.MiddleCenter;
            menuButton.Padding = new Padding(0);
        }
    }
    else
    { //Expand menu
        panelMenu.Width = 348;
        pictureBox1.Visible = true;
        btnMenu.Dock = DockStyle.None;

        foreach (Button menuButton in
panelMenu.Controls.OfType<Button>())
        {
            menuButton.Text = "  " + menuButton.Tag.ToString();
            menuButton.ImageAlign = ContentAlignment.MiddleLeft;
            menuButton.Padding = new Padding(10, 0, 0, 0);
        }
    }
}

```

```
    }  
}  
  
private void btnMenu_Click(object sender, EventArgs e)  
{  
    CollapseMenu();  
}  
  
private void btnClose_Click(object sender, EventArgs e)  
{  
    Application.Exit();  
}  
  
private void btnMinimize_Click(object sender, EventArgs e)  
{  
    formSize = this.ClientSize;  
    this.WindowState = FormWindowState.Minimized;  
}  
  
private void btnMaximize_Click(object sender, EventArgs e)  
{  
    if (this.WindowState == FormWindowState.Normal)  
    {  
        formSize = this.ClientSize;  
        this.WindowState = FormWindowState.Maximized;  
    }  
    else  
    {  
        this.WindowState = FormWindowState.Normal;  
        this.Size = formSize;  
    }  
}
```

```
private void Form1_Resize(object sender, EventArgs e)
{
    AdjustForm();
}

public void SetChild(IView childForm)
{
    if (activeForm != null)
        activeForm.Close();
    activeForm = (Form)childForm;
    ((Form)childForm).TopLevel = false;
    ((Form)childForm).FormBorderStyle = FormBorderStyle.None;
    ((Form)childForm).Dock = DockStyle.Fill;
    this.panelDesktop.Controls.Add((Form)childForm);
    this.panelDesktop.Tag = childForm;
    ((Form)childForm).BringToFront();
    childForm.Show();
}

private void btnMap_Click(object sender, EventArgs e)
{
    btnMapClick?.Invoke(sender, e);
}

private void btnCar_Click(object sender, EventArgs e)
{
    btnCarClick?.Invoke(sender, e);
}

private void btnSupp_Click(object sender, EventArgs e)
{
    btnSuppClick?.Invoke(sender, e);
}
```

```

private void btnSale_Click(object sender, EventArgs e)
{
    btnSaleClick?.Invoke(sender, e);
}

private void btnMachine_Click(object sender, EventArgs e)
{
    btnMachineClick?.Invoke(sender, e);
}

private void btnHome_Click(object sender, EventArgs e)
{
    btnHomeClick?.Invoke(sender, e);
}

private void btnAnalysis_Click(object sender, EventArgs e)
{
    btnAnalysisClick?.Invoke(sender, e);
}

void IView.ShowDialog()
{
    throw new NotImplementedException();
}

public void ShowMessageDialog(string message)
{
    throw new NotImplementedException();
}
}
}

```

Класс MapForm.cs :

```

using GMap.NET;
using GMap.NET.MapProviders;
using GMap.NET.WindowsForms;
using GMap.NET.WindowsForms.Markers;
using GMap.NET.WindowsForms.ToolTips;
using Itinero;
using Itinero.Osm.Vehicles;
using Google.OrTools;
using System.Data.Entity;
using System;
using System.Linq;
using System.Threading.Tasks;
using Yandex.Geocoder;
using Yandex.Geocoder.Enums;
using Yandex.Geocoder.Extenstions;
using System.Windows.Forms;
using System.IO;
using System.Collections.Generic;
using Diplom.Data;
using System.Drawing;
using Diplom.Entity;
using Diplom.Views.Forms;
using Diplom.RouteOptimization;
using Google.OrTools.ConstraintSolver;

using Diplom.Views.Interfaces;

namespace Diplom.Forms
{
    public partial class MapForm : Form, IMapView
    {
        public List<WaterVendingMachine> WaterVendingMachines { get; set; }
        public List<Supplier> Suppliers { get; set; }
    }
}

```

```

public event EventHandler btnRouteClick;
public event EventHandler iconButton4Click;
public event EventHandler iconButton3Click;
public event EventHandler iconButton1Click;
public event EventHandler iconButton5Click;

public ModelContext db { get; set; }

Supplier supp;
List<int> indexs;
List<WaterVendingMachine> resultRoute;
GMapOverlay newMarsh = new GMapOverlay("Atob");
static RouterDb routerDb;
static FileStream stream;
List<Color> tmp = new List<Color>();
public float Distance { get; set; }

public MapForm()
{
    InitializeComponent();
    gmap.Overlays.Add(newMarsh);
}

public List<Color> colors = new List<Color>() { Color.Red, Color.Green,
Color.Blue, Color.Black, Color.Violet, Color.Gold,
Color.OrangeRed, Color.LightGreen, Color.Pink, Color.Teal, Color.Brown,
Color.DarkGray, Color.SkyBlue, Color.Gray, Color.Yellow};

private async void LoadRoute()
{
    LoadForm lForm = new LoadForm();
    lForm.ShowDialog();
    await Task.Run(() =>

```

```

        {
            stream = new FileInfo(@"volga-fed-district-
latest.routerdb").Open(FileMode.Open);

            routerDb = RouterDb.Deserialize(stream);

        });
    }

    public void OptimizeRoute(List<WaterVendingMachine> m, Supplier s)
    {
        // Instantiate the data problem.
        TSP data = new TSP();

        data.FillDistance(m, s);

        // Create Routing Index Manager
        RoutingIndexManager manager = new RoutingIndexManager(
            TSP.DistanceMatrix.GetLength(0),
            data.VehicleNumber,
            data.D Depot);

        // Create Routing Model.
        RoutingModel routing = new RoutingModel(manager);

        int transitCallbackIndex = routing.RegisterTransitCallback(
            (long fromIndex, long toIndex) => {
                // Convert from routing variable Index to distance matrix
                NodeIndex.
                var fromNode = manager.IndexToNode(fromIndex);
                var toNode = manager.IndexToNode(toIndex);
                return TSP.DistanceMatrix[fromNode, toNode];
            }
        );

        // Define cost of each arc.
        routing.SetArcCostEvaluatorOfAllVehicles(transitCallbackIndex);
    }

```

```

// Setting first solution heuristic.
RoutingSearchParameters searchParameters =
operations_research_constraint_solver.DefaultRoutingSearchParameters();
searchParameters.FirstSolutionStrategy =
    FirstSolutionStrategy.Types.Value.PathCheapestArc;
searchParameters.LogSearch = true;

// Solve the problem.
Assignment solution = routing.SolveWithParameters(searchParameters);

// Print solution on console.
indexs=data.Route(routing, manager, solution);
resultRoute = new List<WaterVendingMachine>();
foreach(var i in indexs)
{
    resultRoute.Add(RouteForm.WtM.ToList()[i-1]);
}
}

private void DrawRoute(Itinero.Profiles.Profile profile,Router router,
float x1, float y1, float x2, float y2)
{
    var start = router.Resolve(profile, x1, y1);
    var end = router.Resolve(profile, x2, y2);
    RouterPoint[] lst = { start, end };
    // calculate a route.
    var route = router.Calculate(profile, lst);
    Distance += route.TotalDistance;
    List<PointLatLng> put = new List<PointLatLng>();
    foreach (var itm in route.Shape)
    {
        PointLatLng pt = new PointLatLng(Convert.ToDouble(itm.Latitude),
Convert.ToDouble(itm.Longitude));
        put.Add(pt);
    }
}

```

```

        var chars =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";

        GMapRoute rt = new GMapRoute(put, chars[new
Random().Next(chars.Length)].ToString());

        var c = colors[new Random((int)DateTime.Now.Ticks).Next(0,
colors.Count() - 1)];

        tmp.Add(c);

        Pen pen = new Pen(Color.FromArgb(85, c), 10);

        pen.StartCap = System.Drawing.Drawing2D.LineCap.Round;
        pen.EndCap = System.Drawing.Drawing2D.LineCap.Round;
        pen.DashStyle = System.Drawing.Drawing2D.DashStyle.Solid;
        rt.Stroke = pen;

        newMarsh.Routes.Add(rt);
    }

    private void GenerateRoute( )
    {
        OptimizeRoute(RouteForm.WtM.ToList(), supp);

        newMarsh.Routes.Clear();

        var profile = Vehicle.Car.Shortest();
// create a router.

        var router = new Router(routerDb);

        DrawRoute(profile, router,
float.Parse(supp.Coordinate.Split().First()),
float.Parse(supp.Coordinate.Split().Last()),

        float.Parse(resultRoute[0].Coordinate.Split().First()),
float.Parse(resultRoute[0].Coordinate.Split().Last()));

        for (int i = 0; i < resultRoute.Count-1; ++i)
        {
            DrawRoute(profile, router,
float.Parse(resultRoute[i].Coordinate.Split().First()),
float.Parse(resultRoute[i].Coordinate.Split().Last()),

            float.Parse(resultRoute[i+1].Coordinate.Split().First()),
float.Parse(resultRoute[i+1].Coordinate.Split().Last()));

        }

        DrawRoute(profile, router,
float.Parse(resultRoute[resultRoute.Count-1].Coordinate.Split().First()),
float.Parse(resultRoute[resultRoute.Count-1].Coordinate.Split().Last()),

```

```

        float.Parse(supp.Coordinate.Split().First()),
        float.Parse(supp.Coordinate.Split().Last()));

        MessageBox.Show("Длина маршрута - " + Distance.ToString() + "
метра(-ов)");

        FocusMarker(supp);

        iconButton3.Enabled = true;

        iconButton4.Enabled = true;

        iconButton5.Enabled = true;
    }

    private void btnRoute_Click(object sender, EventArgs e)
    {
        tmp.Clear();

        Distance = 0;

        if (RouteForm.WtM == null || RouteForm.WtM.Count() == 0 ||
RouteForm.toch[0] == null || RouteForm.toch[1] == null)
        {
            MessageBox.Show("Задайте точки для проложения маршрута");

            return;
        }

        string name = RouteForm.toch[0];

        supp = db.Suppliers.FirstOrDefault(c => c.SuppName == name );

        GenerateRoute();
    }

    private void btnScreen_Click(object sender, EventArgs e)
    {
        try
        {
            using (SaveFileDialog dialog = new SaveFileDialog())
            {
                dialog.Filter = "PNG (*.png)|*.png";

                dialog.FileName = "GMap.NET image";

                Image image = this.gmap.ToImage();
            }
        }
    }

```

```

        if (image != null)
        {
            using (image)
            {
                if (dialog.ShowDialog() == DialogResult.OK)
                {
                    string fileName = dialog.FileName;

                    if (!fileName.EndsWith(".png",
                        StringComparison.OrdinalIgnoreCase))
                    {
                        fileName += ".png";
                    }

                    image.Save(fileName);

                    MessageBox.Show("Карта успешно сохранена в
директории: "

                                + Environment.NewLine
                                + dialog.FileName, "GMap.NET",
                                MessageBoxButtons.OK,
                                MessageBoxIcon.Asterisk);
                }
            }
        }
    }
}

catch (Exception exception)
{
    MessageBox.Show("Ошибка при сохранении карты: "

                    + Environment.NewLine
                    + exception.Message,
                    "GMap.NET",
                    MessageBoxButtons.OK,

```

```

        MessageBoxIcon.Hand);

    }

}

private void MapForm_Load(object sender, EventArgs e)
{
    if(stream == null)
        LoadRoute();
}

private void panelDesktop_Paint(object sender, PaintEventArgs e)
{
}

private void gmap_Load(object sender, EventArgs e)
{
    // Настройки для компонента GMap
    gmap.Bearing = 0;

    // Перетаскивание правой кнопки мыши
    gmap.CanDragMap = true;

    // Перетаскивание карты левой кнопкой мыши
    gmap.DragButton = MouseButton.Left;

    gmap.GrayScaleMode = true;

    // Все маркеры будут показаны
    gmap.MarkersEnabled = true;

    // Максимальное приближение
    gmap.MaxZoom = 20;

    // Минимальное приближение
    gmap.MinZoom = 2;

    // Курсор мыши в центр карты

```

```

gmap.MouseWheelZoomType =
GMap.NET.MouseWheelZoomType.MousePositionWithoutCenter;

// Отключение нигативного режима
gmap.NegativeMode = false;
// Разрешение полигонов
gmap.PolygonsEnabled = true;
// Разрешение маршрутов
gmap.RoutesEnabled = true;
// Скрытие внешней сетки карты
gmap.ShowTileGridLines = false;
// При загрузке 10-кратное увеличение
gmap.Zoom = 15;

// Чья карта используется
gmap.MapProvider = GMap.NET.MapProviders.GMapProviders.GoogleMap;
GMap.NET.GMaps.Instance.Mode = GMap.NET.AccessMode.ServerAndCache;
// Создаём новый список маркеров
GMapOverlay markersOverlay = new GMapOverlay("markers");

var _mCoor = WaterVendingMachines;
var _sCoor = Suppliers;

gmap.Position = new GMap.NET.PointLatLng(53.194049, 45.016735);
foreach (WaterVendingMachine m in _mCoor)
{
    double _latitude = double.Parse(m.Coordinate.Split().First());
    double _longitude = double.Parse(m.Coordinate.Split().Last());

    GMarkerGoogle marker = new GMarkerGoogle(new
PointLatLng(_latitude, _longitude), Properties.Resources.wtm); //new
Bitmap(@"D:\диплом\CourseWork_App\Diplom\Resources\wtm.png"

    marker.ToolTip = new GMapRoundedToolTip(marker);
    marker.ToolTipText = m.Location;
    markersOverlay.Markers.Add(marker);
}

```

```

        gmap.Overlays.Add(markersOverlay);
    }

    foreach (Supplier m in _sCoor)
    {
        double _latitude = double.Parse(m.Coordinate.Split().First());
        double _longitude = double.Parse(m.Coordinate.Split().Last());

        GMarkerGoogle marker = new GMarkerGoogle(new
PointLatLng(_latitude, _longitude), Properties.Resources.supp); // new
Bitmap(@"D:\диплом\CourseWork_App\Diplom\Resources\supp.png");

        marker.ToolTip = new GMapRoundedToolTip(marker);
        marker.ToolTipText = m.SuppAddress;
        markersOverlay.Markers.Add(marker);
        gmap.Overlays.Add(markersOverlay);
    }
}

private void iconButton2_Click(object sender, EventArgs e)
{
    RouteForm.db = db;

    RouteForm rForm = new RouteForm();
    rForm.ShowDialog();
}

private void FocusMarker( WaterVendingMachine x)
{
    gmap.Position = new
GMap.NET.PointLatLng(double.Parse(x.Coordinate.Split().First()),
double.Parse(x.Coordinate.Split().Last()));
}

private void FocusMarker(Supplier x)
{
    gmap.Position = new
GMap.NET.PointLatLng(double.Parse(x.Coordinate.Split().First()),
double.Parse(x.Coordinate.Split().Last()));
}

```

```

private int _cur;

private void iconButton4_Click(object sender, EventArgs e)
{
    if (_cur == resultRoute.Count+1)
    {
        return;
    }
    ++_cur;
    if (_cur == resultRoute.Count+1)
        FocusMarker(supp);
    else
        FocusMarker(resultRoute[_cur-1]);
}

private void iconButton3_Click(object sender, EventArgs e)
{
    if (_cur == 0)
    {
        return;
    }
    --_cur;
    if (_cur == 0)
        FocusMarker(supp);
    else
        FocusMarker(resultRoute[_cur-1]);
}

private void iconButton1_Click(object sender, EventArgs e)
{
    if (resultRoute == null)
    {
        MessageBox.Show("Выберите точки для построения маршрута");
    }
}

```

```

        return;

    }

    InfoForm infForm = new InfoForm();

    infForm.AddText("Маршрут обслуживания автоматов для продажи воды.
Автомобиль - " + RouteForm.toch[1]+"\\n ",Color.Black);

    string route = "Поставщик "+supp.SuppName + " " + supp.SuppAddress +
" \\n" + " ---> \\n"+" Водомат "+resultRoute[0].Id+" "+resultRoute[0].Location;

    infForm.AddText(route + "\\n", tmp[0]);

    for (int i=0; i<resultRoute.Count-1;++i)

    {

        route = "Водомат " + resultRoute[i].Id + " " +
resultRoute[i].Location + " \\n" + " ---> \\n" + "Водомат " + resultRoute[i+1].Id
+ " " + resultRoute[i+1].Location;

        infForm.AddText(route + "\\n", tmp[i+1]);

    }

    route = "Водомат" + resultRoute[resultRoute.Count-1].Id + " " +
resultRoute[resultRoute.Count - 1].Location + " \\n" + " ---> \\n" + "Поставщик "
+ supp.SuppName + " " + supp.SuppAddress;

    infForm.AddText(route + "\\n", tmp[resultRoute.Count]);

    infForm.ShowDialog();

}

private void iconButton5_Click(object sender, EventArgs e)

{

    foreach (var x in resultRoute)

    {

        db.WaterVendingMachines.FirstOrDefault(c => x.Id ==
c.Id).ServiceDate = db.PurchaseLogs.Max(c => c.PurchaseDate);

        db.WaterVendingMachines.FirstOrDefault(c => x.Id ==
c.Id).WaterLevelSensor.FirstOrDefault().StatusSensor = true;

        db.WaterVendingMachines.FirstOrDefault(c => x.Id ==
c.Id).WaterLevelSensor.FirstOrDefault().StatusWater = x.Capacity;

        db.SaveChanges();

    }

    MessageBox.Show("Водоматы из проложенного маршрута успешно
обслужены!");

}

```

```

void IView.ShowDialog()
{
    ShowDialog();
}

public void ShowMessageDialog(string message)
{
    MessageBox.Show(message);
}
}
}

```

Класс **RegressForm.cs** :

```

using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplom.Views.Forms
{
    public partial class RegressForm : Form, IRegressView
    {
        public DataGridViewRowCollection DataGridViewRows
        {
            get { return dataGridView1.Rows; }
        }

        public int RowsCount
        {
            get { return dataGridView1.Rows.Count; }
        }

        public RegressForm()
        {
            InitializeComponent();
        }

        public event EventHandler btnCloseClick;

        private void RegressForm_Load(object sender, EventArgs e)
        {
        }
    }
}

```

```

private void panell1_MouseDown(object sender, MouseEventArgs e)
{
    ReleaseCapture();
    SendMessage(this.Handle, 0x112, 0xf012, 0);
}
//Drag Form
[DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
private extern static void ReleaseCapture();
[DllImport("user32.DLL", EntryPoint = "SendMessage")]
private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int
wParam, int lParam);
protected override void WndProc(ref Message m)
{
    const int WM_NCCALCSIZE = 0x0083;
    if (m.Msg == WM_NCCALCSIZE && m.WParam.ToInt32() == 1)
    {
        return;
    }

    base.WndProc(ref m);
}

private void btnClose_Click(object sender, EventArgs e)
{
    Close();
}

void IView.ShowDialog()
{
    ShowDialog();
}

public void ShowMessageDialog(string message)
{
    MessageBox.Show(message);
}

private void supplierBindingSource_CurrentChanged(object sender,
EventArgs e)
{
}

private void panelMenu_Paint(object sender, PaintEventArgs e)
{
}

private void panel3_Paint(object sender, PaintEventArgs e)
{
}

private void panel2_Paint(object sender, PaintEventArgs e)
{
}

private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
}

```

```

    }

    private void waterLevelSensorBindingSource_CurrentChanged(object sender,
EventArgs e)
    {

    }

    private void panell1_Paint(object sender, PaintEventArgs e)
    {

    }

}
}

```

Класс RouteForm.cs :

```

using Diplom.Data;
using Diplom.Entity;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplom.Views.Forms
{
    public partial class RouteForm : Form
    {
        public static ModelContext db;
        private List<WaterVendingMachine> machines;
        public static HashSet<WaterVendingMachine> WtM;
        public static string[] toch = new string[2];
        public RouteForm()
        {
            InitializeComponent();

        }

        private void RouteForm_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "waterSystemDataSet2.Car". При необходимости она может быть перемещена или
            удалена.
            this.carTableAdapter.Fill(this.waterSystemDataSet2.Car);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "waterSystemDataSet.Supplier". При необходимости она может быть перемещена или
            удалена.
            this.supplierTableAdapter.Fill(this.waterSystemDataSet.Supplier);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "waterSystemDataSet2.WaterLevelSensor". При необходимости она может быть
            перемещена или удалена.

            this.waterLevelSensorTableAdapter.Fill(this.waterSystemDataSet2.WaterLevelSensor
);

```

// TODO: данная строка кода позволяет загрузить данные в таблицу "waterSystemDataSet2.WaterVendingMachine". При необходимости она может быть перемещена или удалена.

```
this.waterVendingMachineTableAdapter.Fill(this.waterSystemDataSet2.WaterVendingMachine);
```

// TODO: данная строка кода позволяет загрузить данные в таблицу "waterSystemDataSet.WaterVendingMachine". При необходимости она может быть перемещена или удалена.

// TODO: данная строка кода позволяет загрузить данные в таблицу "waterSystemDataSet2.WaterVendingMachine". При необходимости она может быть перемещена или удалена.

```
this.waterVendingMachineTableAdapter.Fill(this.waterSystemDataSet2.WaterVendingMachine);
```

```

        WtM = new HashSet<WaterVendingMachine>();
        db.Suppliers.Load();
        db.WaterVendingMachines.Load();
        db.WaterLevelSensors.Load();
        var x = db.Suppliers.Local.Where(c => c.SuppName ==
comboBox1.Text).FirstOrDefault().Car.ToList();
        foreach (var y in x)
            comboBox2.Items.Add(y.CarBrand.ToString() + " " +
y.CarNumber.ToString());
        comboBox2.SelectedIndex = 0;
        toch[0] = comboBox1.Text;
        toch[1] = comboBox2.Text;
        machines = db.WaterVendingMachines.Local.ToList();
        Fill_Table();
    }
    public void Fill_Table()
    {
        foreach (var m in machines)
        {
            RouteTable.Rows.Add(m.Id, m.ServiceDate.ToShortDateString(),
m.WaterLevelSensor.FirstOrDefault().StatusWater,
            m.WaterLevelSensor.FirstOrDefault().StatusSensor, false);
        }
    }
    public static void RouteAdd()
    {
        var _machines = db.WaterVendingMachines.Local.ToList();
        var _date = _machines.Where(c =>
Math.Abs(c.ServiceDate.Subtract(db.PurchaseLogs.Max(d =>
d.PurchaseDate)).TotalDays) > 3).ToList();
        List<WaterVendingMachine> _badM = new List<WaterVendingMachine>();
        foreach (var m in _machines)
        {
            var _sum = m.Capacity -
m.WaterLevelSensor.FirstOrDefault().StatusWater;
            var _days = m.PurchaseLog.Select(entry =>
entry.PurchaseDate.Date).Distinct().Count();
            var _avg = _sum / _days;
            if ((100 * m.WaterLevelSensor.FirstOrDefault().StatusWater) /
m.Capacity < 10 || m.WaterLevelSensor.FirstOrDefault().StatusWater < 2 * _avg)
                _badM.Add(m);
        }
        WtM = WtM == null ? _date.Union(_badM).ToHashSet() :
WtM.Union(_badM).Union(_date).ToHashSet();
    }

```

```

//Drag Form
[DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
private extern static void ReleaseCapture();
[DllImport("user32.DLL", EntryPoint = "SendMessage")]
private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int
wParam, int lParam);
protected override void WndProc(ref Message m)
{
    const int WM_NCCALCSIZE = 0x0083;
    if (m.Msg == WM_NCCALCSIZE && m.WParam.ToInt32() == 1)
    {
        return;
    }

    base.WndProc(ref m);
}
private void panell1_MouseDown(object sender, MouseEventArgs e)
{
    ReleaseCapture();
    SendMessage(this.Handle, 0x112, 0xf012, 0);
}

private void btnAdd_Click(object sender, EventArgs e)
{
    if (comboBox2.Items.Count == 0)
    {
        MessageBox.Show("Выберите компанию с машинами в автопарке");
        return;
    }
    Close();
}

private void iconButton1_Click(object sender, EventArgs e)
{
    RouteAdd();
    foreach (var m in WtM) {
        for (int i = 0; i < RouteTable.RowCount; i++) {
            if (RouteTable.Rows[i].Cells[0].Value.ToString() ==
m.Id.ToString())
                RouteTable.Rows[i].Cells[4].Value = true;
        }
    }
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    comboBox2.Items.Clear();
    var x = db.Suppliers.Local.Where(c => c.SuppName ==
comboBox1.Text.ToString()).FirstOrDefault().Car.ToList();
    foreach (var y in x)
        comboBox2.Items.Add(y.CarBrand.ToString() + "
"+y.CarNumber.ToString());
    if (comboBox2.Items.Count != 0)
    {
        comboBox2.SelectedIndex = 0;
        toch[0] = comboBox1.Text;
        toch[1] = comboBox2.Text;
    }
}

```

```

private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
{
    toch[1] = comboBox2.Text;
}

private void RouteTable_CellValueChanged(object sender,
DataGridViewCellEventArgs e)
{
    try
    {
        if (e.ColumnIndex == 4)
        {
            var ind =
RouteTable.Rows[e.RowIndex].Cells[0].Value.ToString();
            foreach (var x in machines)
            {
                if (x.Id.ToString() == ind)
                {
                    if
(RouteTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString() == "Выбран" ||
RouteTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString() == "True")
                        WtM.Add(x);
                    else
                        WtM.Remove(x);
                }
            }
        }
    }
    catch
    {
        return;
    }
}

private void panell1_Paint(object sender, PaintEventArgs e)
{
}
}

```

Класс **SupplierForm.cs** :

```

using Diplom.Data;
using Diplom.Entity;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplom.Forms
{
    public partial class SupplierForm : Form, ISuppliersView
    {

```

```

public event EventHandler btnAddClick;
public event EventHandler btnDeleteClick;

public SupplierForm()
{
    InitializeComponent();
}

public IEnumerable<Supplier> SuppliersDatasource
{
    set { SuppTable.DataSource = value; }
}

private void SupplierForm_Load(object sender, EventArgs e)
{
}

private void btnDelete_Click(object sender, EventArgs e)
{
    if (SuppTable.SelectedRows.Count > 0)
    {
        foreach (DataGridViewRow item in this.SuppTable.SelectedRows)
        {
            SuppTable.Rows.RemoveAt(item.Index);
        }

        btnDeleteClick?.Invoke(sender, e);
    }
}

private void btnAdd_Click(object sender, EventArgs e)
{
    btnAddClick?.Invoke(sender, e);
    SuppTable.Refresh();
}

void IView.ShowDialog()
{
    throw new NotImplementedException();
}

public void ShowMessageDialog(string message)
{
    MessageBox.Show(message);
}
}
}

```

Класс TransactionForm.cs :

```

using Diplom.Data;
using Diplom.Entity;
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 MathNet.Numerics.Distributions;
using System.Data.Entity;
using Diplom.Views.Interfaces;

namespace Diplom.Forms
{
    public partial class TransactionForm : Form, ITransactionView
    {
        public static ModelContext db;

        public event EventHandler btnGenerateClick;

        public IEnumerable<PurchaseLog> PurchaseLogDatasource
        {
            set { dataGridView1.DataSource = value; }
        }

        public TransactionForm()
        {
            InitializeComponent();
        }

        private void TransactionForm_Load(object sender, EventArgs e)
        {
        }

        private void btnGenerate_Click(object sender, EventArgs e)
        {
            btnGenerateClick?.Invoke(sender, e);
        }

        void IView.ShowDialog()
        {
            throw new NotImplementedException();
        }

        public void ShowMessageDialog(string message)
        {
            throw new NotImplementedException();
        }
    }
}

```

Класс **WaterMachineForm.cs** :

```

using Diplom.Data;
using Diplom.Entity;
using Diplom.Views.Forms;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace Diplom.Forms
{
    public partial class WaterMachineForm : Form, IWaterMachineView
    {
        //public ModelContext db = new ModelContext();

        public event EventHandler btnAddClick;
        public event DeleteItemEventHandler itemDeleted;
        public event EventHandler allDeleted;
        public event TableCellClickHandler WtMachineTableCellClick;
        public event EventHandler WtMachineTableCellEndEdit;

        public IEnumerable<WaterVendingMachine> WaterVendingMachinesDatasource
        {
            set { WtMachineTable.DataSource = value; }
        }

        public WaterMachineForm()
        {
            InitializeComponent();
            DataGridViewButtonColumn btn = new DataGridViewButtonColumn();
            btn.DataPropertyName = "WaterSensor";
            btn.HeaderText = "Показания датчика";
            btn.Text = "Показать";
            btn.Name = "btn";
            btn.UseColumnTextForButtonValue = true;
            WtMachineTable.Columns.Add(btn);
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            btnAddClick?.Invoke(sender, e);
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            if (WtMachineTable.SelectedRows.Count > 0)
            {
                foreach (DataGridViewRow item in
this.WtMachineTable.SelectedRows)
                {
                    var x =
WtMachineTable.Rows[item.Index].Cells[0].Value.ToString();
                    DeleteItemEventArgs args = new DeleteItemEventArgs();
                    args.WaterVendingMachineToDelete = x;
                    itemDeleted?.Invoke(args);
                    WtMachineTable.Rows.RemoveAt(item.Index);
                }

                allDeleted?.Invoke(sender, e);
            }
        }

        private void panelDesktop_Paint(object sender, PaintEventArgs e)
        {
        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)

```

```

    {

    }

    private void waterSystemDataSetBindingSource_CurrentChanged(object
sender, EventArgs e)
    {

    }

    private void panelMenu_Paint(object sender, PaintEventArgs e)
    {

    }

    private void WaterMachineForm_Load(object sender, EventArgs e)
    {

    }

    private void WtMachineTable_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        try
        {
            if (e.ColumnIndex == 5)
            {
                WtMachineTableCellClickEventArgs args = new
WtMachineTableCellClickEventArgs();
                args.Index =
WtMachineTable.Rows[e.RowIndex].Cells[0].Value.ToString();
                WtMachineTableCellClick?.Invoke(args);
            }
        }
        catch
        {
            return;
        }
    }

    private void waterVendingMachineBindingSource_CurrentChanged(object
sender, EventArgs e)
    {

    }

    private void fKwaterlevWater267ABA7ABindingSource_CurrentChanged(object
sender, EventArgs e)
    {

    }

    private void fKwaterlevWater267ABA7ABindingSource1_CurrentChanged(object
sender, EventArgs e)
    {

    }

    private void WtMachineTable_CellEndEdit(object sender,
DataGridViewCellEventArgs e)
    {

```

```

        WtMachineTableCellEndEdit?.Invoke(sender, null);
    }

    void IView.ShowDialog()
    {
        throw new NotImplementedException();
    }

    public void ShowMessageDialog(string message)
    {
        MessageBox.Show(message);
    }
}
}

```

Класс **WaterSensorForm.cs** :

```

using Diplom.Data;
using Diplom.Entity;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diplom.Views.Forms
{
    public partial class WaterSensorForm : Form, IWaterSensorView
    {
        public static WaterLevelSensor s;

        public event EventHandler btnCloseClick;
        public IEnumerable<WaterLevelSensor> WaterLevelSensorsDatasource
        {
            set
            {
                dataGridView1.DataSource = value;
                dataGridView1.Refresh();
            }
        }

        public WaterSensorForm()
        {
            InitializeComponent();
        }

        private void panell1_MouseDown(object sender, MouseEventArgs e)
        {
            ReleaseCapture();
            SendMessage(this.Handle, 0x112, 0xf012, 0);
        }

        //Drag Form
        [DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
        private extern static void ReleaseCapture();
        [DllImport("user32.DLL", EntryPoint = "SendMessage")]

```

```

private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int
wParam, int lParam);
protected override void WndProc(ref Message m)
{
    const int WM_NCCALCSIZE = 0x0083;
    if (m.Msg == WM_NCCALCSIZE && m.WParam.ToInt32() == 1)
    {
        return;
    }

    base.WndProc(ref m);
}

private void btnClose_Click(object sender, EventArgs e)
{
    s.StatusSensor =
Convert.ToBoolean(dataGridView1.Rows[0].Cells[2].Value);
    btnCloseClick?.Invoke(sender, e);
}

private void WaterSensorForm_Load(object sender, EventArgs e)
{
}

private void panell1_Paint(object sender, PaintEventArgs e)
{
}

private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
}

private void supplierBindingSource_CurrentChanged(object sender,
EventArgs e)
{
}

private void panelMenu_Paint(object sender, PaintEventArgs e)
{
}

private void panel3_Paint(object sender, PaintEventArgs e)
{
}

private void panel2_Paint(object sender, PaintEventArgs e)
{
}

private void waterLevelSensorBindingSource_CurrentChanged(object sender,
EventArgs e)
{
}

```

```

        void IView.ShowDialog()
        {
            ShowDialog();
        }

        public void ShowMessageDialog(string message)
        {
            MessageBox.Show(message);
        }
    }
}

```

Класс **Program.cs** :

```

using Diplom.Presenters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Diplom
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

            MainPresenter mainPresenter = new MainPresenter(new MainForm());

            Application.Run((Form)mainPresenter.View);
        }
    }
}

```

Класс **TSP.cs** :

```

using System.Collections.Generic;
using System.Linq;
using Diplom.Entity;
using Google.OrTools.ConstraintSolver;
using System.Windows.Forms;
namespace Diplom.RouteOptimization
{
    public class TSP
    {
        public static long[,] DistanceMatrix;
        public int VehicleNumber = 1;
        public int Depot = 0;
        List<int> lst = new List<int>();
        /// <summary>
        /// Print the solution.
        /// </summary>
        ///
        public void FillDistance(List<WaterVendingMachine> m, Supplier s)
        {

```

```

        DistanceMatrix = new long[m.Count+1,m.Count+1];

        for (int k = 1; k < m.Count+1; k++)
        {
            DistanceMatrix[0, k] = (long)MainForm._distanceToCentre(new double[] {
double.Parse(s.Coordinate.Split().First()), double.Parse(s.Coordinate.Split().Last())
            }, new double[] { double.Parse(m[k-1].Coordinate.Split().First()),
double.Parse(m[k-1].Coordinate.Split().Last()) });
            DistanceMatrix[k, 0] = DistanceMatrix[0, k];
        }
        for (int i=1; i<m.Count+1; i++)
        {
            for (int j = 1; j < m.Count+1; j++)
            {
                DistanceMatrix[i, j] = (long)MainForm._distanceToCentre(new double[] {
double.Parse(m[i-1].Coordinate.Split().First()), double.Parse(m[i-
1].Coordinate.Split().Last())
                }, new double[] { double.Parse(m[j-1].Coordinate.Split().First()),
double.Parse(m[j-1].Coordinate.Split().Last()) });
            }
        }
        var x = DistanceMatrix[m.Count(),m.Count()];
    }
    public List<int> Route(
        in RoutingModel routing,
        in RoutingIndexManager manager,
        in Assignment solution)
    {
        long routeDistance = 0;
        var index = routing.Start(0);
        while (routing.IsEnd(index) == false)
        {
            var previousIndex = index;
            index = solution.Value(routing.NextVar(index));
            lst.Add(manager.IndexToNode((int)index));
            routeDistance += routing.GetArcCostForVehicle(previousIndex, index, 0);
        }
        lst.Remove(lst[lst.Count - 1]);
        return lst;
    }
}

```

Приложение Б
Листинг кода модульных тестов
(обязательное)

Класс TestDbContext.cs :

```

using Diplom.Data;
using Diplom.Entity;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace UnitTestProject
{
    public class TestDbContext : IDatabaseContext
    {
        public TestDbContext()
        {
            Cars = new TestDbSet<Car>();
            PurchaseLogs = new TestDbSet<PurchaseLog>();
            Suppliers = new TestDbSet<Supplier>()
            {
                new Supplier("supplier 1", "address", "0 0")
            };
            WaterVendingMachines = new TestDbSet<WaterVendingMachine>();
            WaterLevelSensors = new TestDbSet<WaterLevelSensor>();
        }

        public DbSet<Car> Cars { get; set; }
        public DbSet<PurchaseLog> PurchaseLogs { get; set; }
        public DbSet<Supplier> Suppliers { get; set; }
        public DbSet<sysdiagrams> sysdiagrams { get => throw new
NotImplementedException(); set => throw new NotImplementedException(); }
        public DbSet<WaterVendingMachine> WaterVendingMachines { get; set; }
        public DbSet<WaterLevelSensor> WaterLevelSensors { get; set; }

        public int SaveChanges()
        {
            return 0;
        }
    }
}

```

Класс TestDbSet.cs :

```

using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace UnitTestProject
{
    public class TestDbSet<T> : DbSet<T>, IQueryable, IEnumerable<T>
        where T : class
    {
        ObservableCollection<T> _data;
        IQueryable _query;

        public TestDbSet()
        {
            _data = new ObservableCollection<T>();
            _query = _data.AsQueryable();
        }

        public override T Add(T item)
        {
            _data.Add(item);
            return item;
        }

        public override T Remove(T item)
        {
            _data.Remove(item);
            return item;
        }

        public override T Attach(T item)
        {
            _data.Add(item);
            return item;
        }

        public override T Create()
        {
            return Activator.CreateInstance<T>();
        }

        public override TDerivedEntity Create<TDerivedEntity>()
        {
            return Activator.CreateInstance<TDerivedEntity>();
        }

        public override ObservableCollection<T> Local
        {
            get { return new ObservableCollection<T>(_data); }
        }

        Type IQueryable.ElementType
        {
            get { return _query.ElementType; }
        }

        System.Linq.Expressions.Expression IQueryable.Expression
        {
            get { return _query.Expression; }
        }

        IQueryProvider IQueryable.Provider
        {
            get { return _query.Provider; }
        }
    }
}

```

```

        System.Collections.IEnumerator
System.Collections.IEnumerable.GetEnumerator()
    {
        return _data.GetEnumerator();
    }

    IEnumerator<T> IEnumerable<T>.GetEnumerator()
    {
        return _data.GetEnumerator();
    }
}

```

Класс **TestAddCarView.cs** :

```

using Diplom.Entity;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace UnitTestProject
{
    public class TestAddCarView : IAddCarView
    {
        private IEnumerable<Supplier> _suppliers;

        public IEnumerable<Supplier> SuppliersDatasource
        {
            set { _suppliers = value; }
        }

        public TestAddCarView(string number, string model, string type)
        {
            Number = number;
            Model = model;
            Type = type;
        }

        public string SupplierName
        {
            get { return _suppliers.FirstOrDefault().SuppName; }
        }

        public string Number { get; set; }

        public string Model { get; set; }

        public string Type { get; set; }

        public event EventHandler btnAddClick;
        public event EventHandler btnDeleteClick;

        public void Close() { }

        public void Show() { }

        public void ShowDialog() { }

        public void ShowMessageDialog(string message) { }
    }
}

```

```

        public void btnAddClicked()
        {
            btnAddClick?.Invoke(this, null);
        }
    }
}

```

Класс **TestAddMarkerView.cs** :

```

using Diplom.Entity;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace UnitTestProject
{
    public class TestAddMarkerView : IAddMarkerView
    {
        public double[] Coordinates { get; set; }
        public string Address { get; set; }
        public List<WaterVendingMachine> WaterVendingMachines { get; set; }
        public List<Supplier> Suppliers { get; set; }

        public TestAddMarkerView(double[] coordinates, string address)
        {
            Coordinates = coordinates;
            Address = address;
        }

        public event EventHandler btnAddClick;

        public void btnAddClicked()
        {
            btnAddClick?.Invoke(this, null);
        }

        public void Close() { }

        public void Show() { }

        public void ShowDialog() { }

        public void ShowMessageDialog(string message) { }
    }
}

```

Класс **TestAddSupplierView.cs** :

```

using Diplom.Entity;
using Diplom.Views.Interfaces;
using System;
using System.Collections.Generic;

namespace UnitTestProject
{
    public class TestAddSupplierView : IAddSupplierView
    {
        public TestAddSupplierView(string supplierName)
        {

```

```

        SupplierName = supplierName;
    }

    public string SupplierName { get; set; }

    public event EventHandler btnAddClick;
    public event EventHandler btnDeleteClick;
    public event EventHandler button1Click;

    public void button1Clicked()
    {
        button1Click?.Invoke(this, null);
    }

    public void btnAddClicked()
    {
        btnAddClick?.Invoke(this, null);
    }

    public void Close() { }

    public void Show() { }

    public void ShowDialog() { }

    public void ShowMessageDialog(string message) { }
}

```

Класс AddCarPresenterTests.cs :

```

using Diplom.Presenters;
using Diplom.Views.Interfaces;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;

namespace UnitTestProject
{
    [TestClass]
    public class AddCarPresenterTests
    {
        [TestMethod]
        public void AddCarTest()
        {
            IAddCarView view = new TestAddCarView("a111aa58", "model1",
"typel");
            TestDbContext db = new TestDbContext();
            AddCarPresenter addCarPresenter = new AddCarPresenter(view, db);

            ((TestAddCarView)addCarPresenter.View).btnAddClicked();

            Assert.AreEqual(1, db.Cars.Local.Count);
        }
    }
}

```

Класс AddMarkerPresenterTests.cs :

```

using Diplom.Presenters;
using Diplom.Views.Interfaces;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;

```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace UnitTestProject
{
    [TestClass]
    public class AddMarkerPresenterTests
    {
        [TestMethod]
        public void AddMarkerTest()
        {
            double[] coords = new double[2] { 10, 10 };
            string address = "address";
            IAddMarkerView view = new TestAddMarkerView(coords, address);
            TestDbContext db = new TestDbContext();
            AddMarkerPresenter addMarkerPresenter = new AddMarkerPresenter(view,
db);

            ((TestAddMarkerView)addMarkerPresenter.View).btnAddClicked();

            Assert.AreEqual(coords, addMarkerPresenter.Coordinates);
            Assert.AreEqual(address, addMarkerPresenter.Address);
        }
    }
}

```

Класс AddMarkerPresenterTests.cs :

```

using Diplom.Presenters;
using Diplom.Views.Interfaces;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace UnitTestProject
{
    [TestClass]
    public class AddSupplierPresenterTests
    {
        [TestMethod]
        public void AddSupplierTest()
        {
            double[] coords = new double[2] { 10, 10 };
            string address = "address";
            IAddMarkerView markerView = new TestAddMarkerView(coords, address);
            TestDbContext db = new TestDbContext();
            IAddSupplierView view = new TestAddSupplierView("supplier 1");

            AddSupplierPresenter addSupplierPresenter = new
AddSupplierPresenter(view, db, markerView);

            ((TestAddSupplierView)addSupplierPresenter.View).button1Clicked();

            ((TestAddMarkerView)markerView).btnAddClicked();

            Assert.AreEqual(coords, addSupplierPresenter.Coordinates);
            Assert.AreEqual(address, addSupplierPresenter.Address);

            ((TestAddSupplierView)addSupplierPresenter.View).btnAddClicked();

            Assert.AreEqual(2, db.Suppliers.Local.Count);
        }
    }
}

```

}

Приложение В
Результаты функционального тестирования
(обязательное)

Автомат с водой

Локация Выбрать на карте

Объем резервуара, л

Цена за 1л

+

↻

⊘

Рисунок В.1 – Добавления водомата. Часть 1

	Id	Адрес	Объем резервуара, л	Цена 1л, руб	Последняя дата обслуживания	Показания датчика
	2028	Россия, Приволж...	100000	3,00	30.06.2023	Показать
	2031	Россия, Приволж...	50000	3,00	30.06.2023	Показать
	2032	Россия, Приволж...	50000	3,00	30.06.2023	Показать
	2033	Россия, Приволж...	90000	15,00	30.06.2023	Показать
	2034	Россия, Приволж...	90000	3,00	30.06.2023	Показать
	2035	Россия, Приволж...	99999	3,00	30.06.2023	Показать
▶	2036	Россия, Приволж...	50000	15	30.06.2023	Показать

+

⊘

Рисунок В.2 – Добавления водомата. Часть 2

	Модель датчика	Текущий уровень воды	Статус
...	Bosch-00607138	50000	☒

×

Рисунок В.3 – Включение водомата

	Модель датчика	Текущий уровень воды	Статус
...	Bosch-00607138	50000	☐

×

Рисунок В.4 – Выключение водомата

	Id водомата	Расстояние до центра города (X1)	Цена 1 литра воды (X2)	Среднесуточная выручка (Y)
▶	2028	2645,58782914031	3	1072,60515
	2031	305,293617876983	3	7816,74091428571
	2032	1729,13274674781	3	1645,8486
	2033	3568,55548292022	15	992,348416666667
	2034	1026,75863140837	3	2766,82218333333
	2035	2894,4826177611	3	730,4974
	New	307,793728759239	15	8673,4419

×

Рисунок В.5 – Прогнозирование дохода водомата

	Id	Последнее обслуживание	Остаток воды	Статус водомата	Выбрать
	2028	30.06.2023	100000	☒	☒
	2031	30.06.2023	50000	☒	☒
	2032	30.06.2023	50000	☒	☒
	2033	30.06.2023	90000	☒	☒
	2034	30.06.2023	90000	☒	☒
	2035	30.06.2023	99999	☒	☒
▶	2036	30.06.2023	50000	☒	☒

+

Автовыбор

Water 24/7

Nissan GTR п123оп58

Рисунок В.6 – Выбор точек для составления маршрута

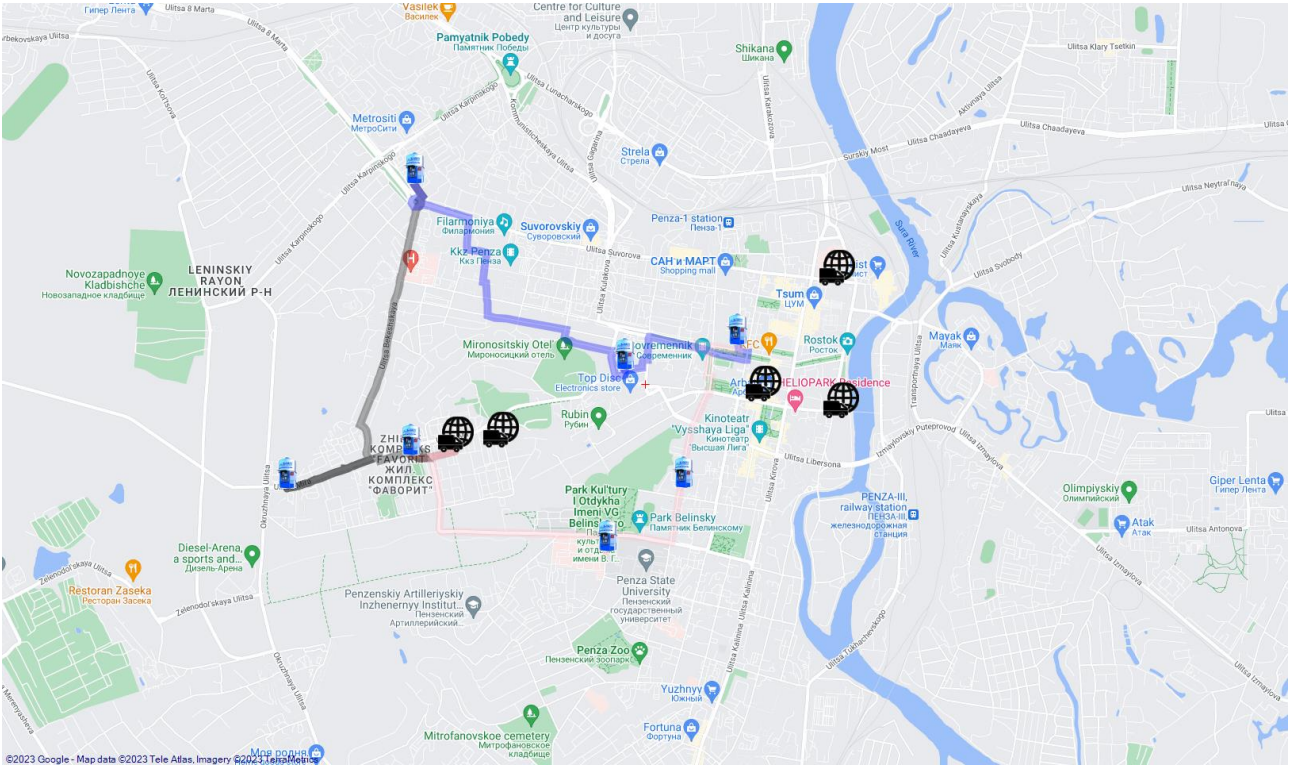


Рисунок В.7 – Маршрут движения

Маршрут обслуживания автоматов для продажи воды. Автомобиль - Nissan GTR n123on58
Поставщик Water 24/7 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, Первомайский район, микрорайон Западная Поляна,
--->
Водомат 2028 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица Мира, 44
Водомат 2028 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица Мира, 44
--->
Водомат 2033 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица Мира
Водомат 2033 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица Мира
--->
Водомат 2035 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица Карпинского, 38
Водомат 2035 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица Карпинского, 38
--->
Водомат 2036 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица Космодемьянской, 1
Водомат 2036 Россия, Приволжский федеральный округ, Пензенская область, городской округ Пенза, Пенза, улица Космодемьянской, 1
--->

Рисунок В.8 – Маршрут движения в текстовом формате

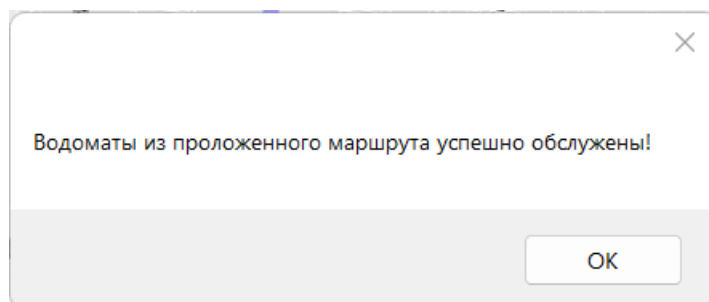


Рисунок В.9 – Окно о результате обслуживания

Поставщик

Название компании: New Supplier

Адрес: [Выбрать на карте](#)

+

-

Рисунок В.10 – Добавление поставщика. Часть 1

Название поставщика	Адрес поставщика
gt5tg	Россия, Приволжский федеральный округ, Пензенская область, городской округ...
Water 24/7	Россия, Приволжский федеральный округ, Пензенская область, городской округ...
Вода без труда	Россия, Приволжский федеральный округ, Пензенская область, городской округ...
324	Россия, Приволжский федеральный округ, Пензенская область, городской округ...
5555	Россия, Приволжский федеральный округ, Пензенская область, городской округ...
New Supplier	Россия, Приволжский федеральный округ, Пензенская область, городской округ...

+

-

Рисунок В.11 – Добавление поставщика. Часть 2

Автомобиль

Марка машины: Рено Сандеро

Номер машины: x255cp58

Тип машины: Легковая

Владелец машины: New Supplier

Рисунок В.12 – Добавление автомобиля. Часть 1

	Id	Модель автомобиля	Тип автомобиля	Номер автомобиля	Компания
	2	Mitsubishi Lancer	Легковая	x777рю58	qt5tg
	3	Nissan GTR	Грузовая	n123on58	Water 24/7
	4	Рено Логан	Легковая	a228op58	qt5tg
	7	Рено Логан	Грузовая	a1236в777	Вода без труда
	10	Рено Сандеро	Легковая	x255cp58	New Supplier

Рисунок В.13 – Добавление автомобиля. Часть 2

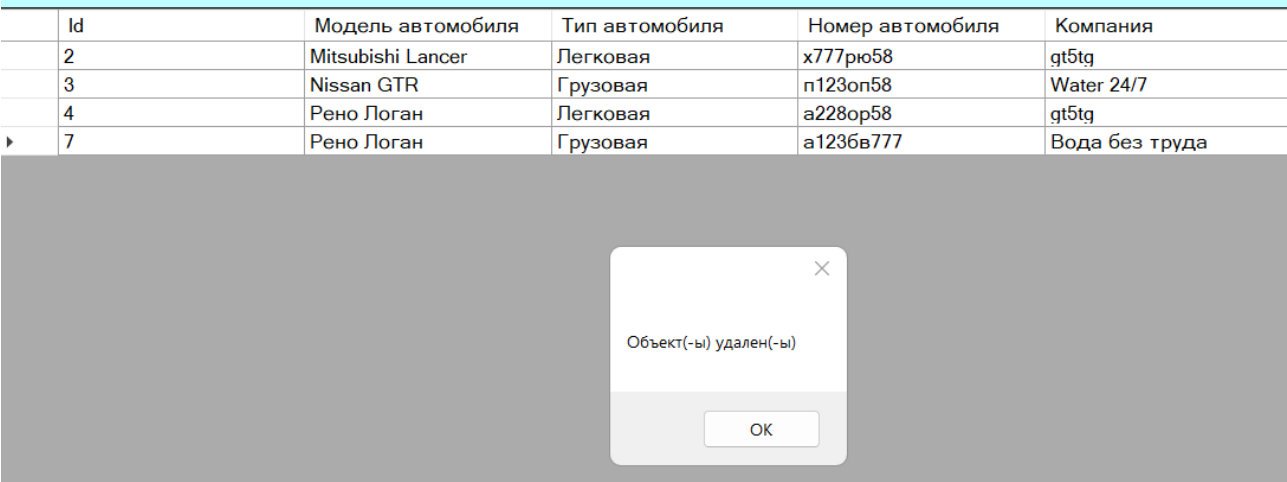


Рисунок В.14 – Удаление автомобиля воды

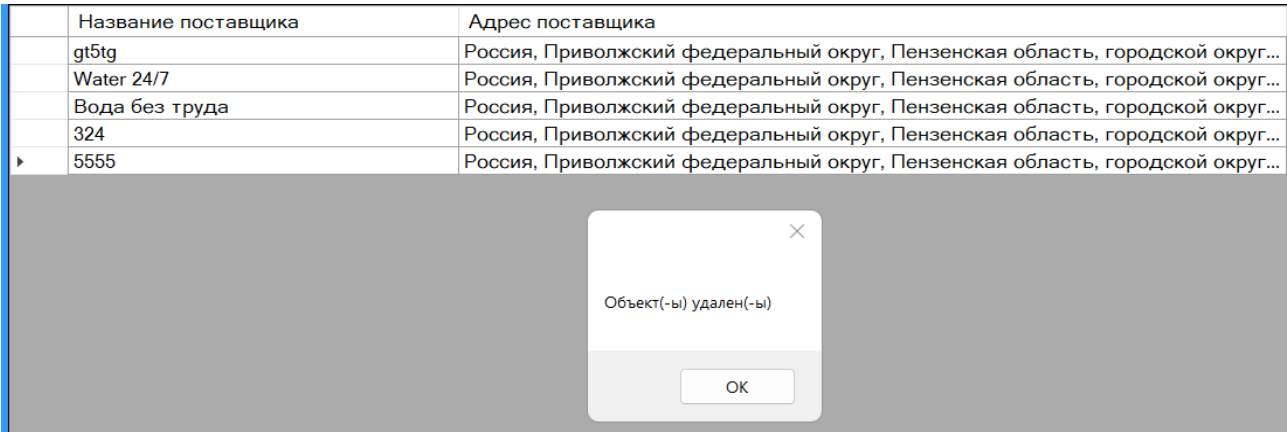


Рисунок В.15 – Удаление поставщика воды

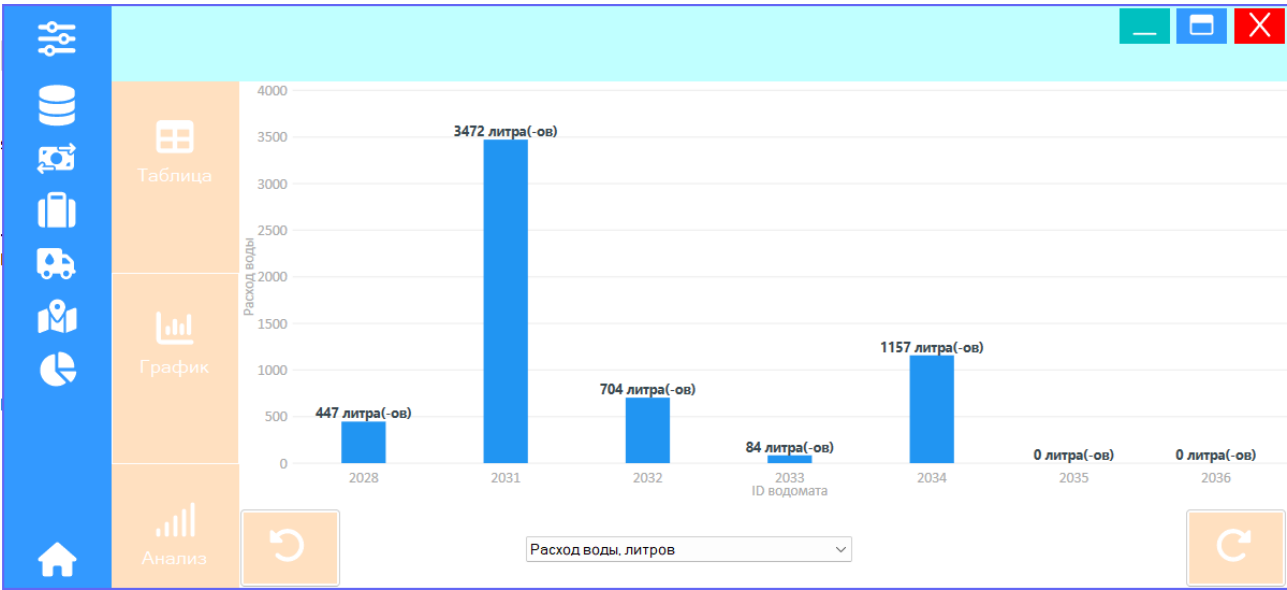


Рисунок В.16 – Сравнительный анализ водоматов

Приложение Г
Диаграмма Ганта
(обязательное)

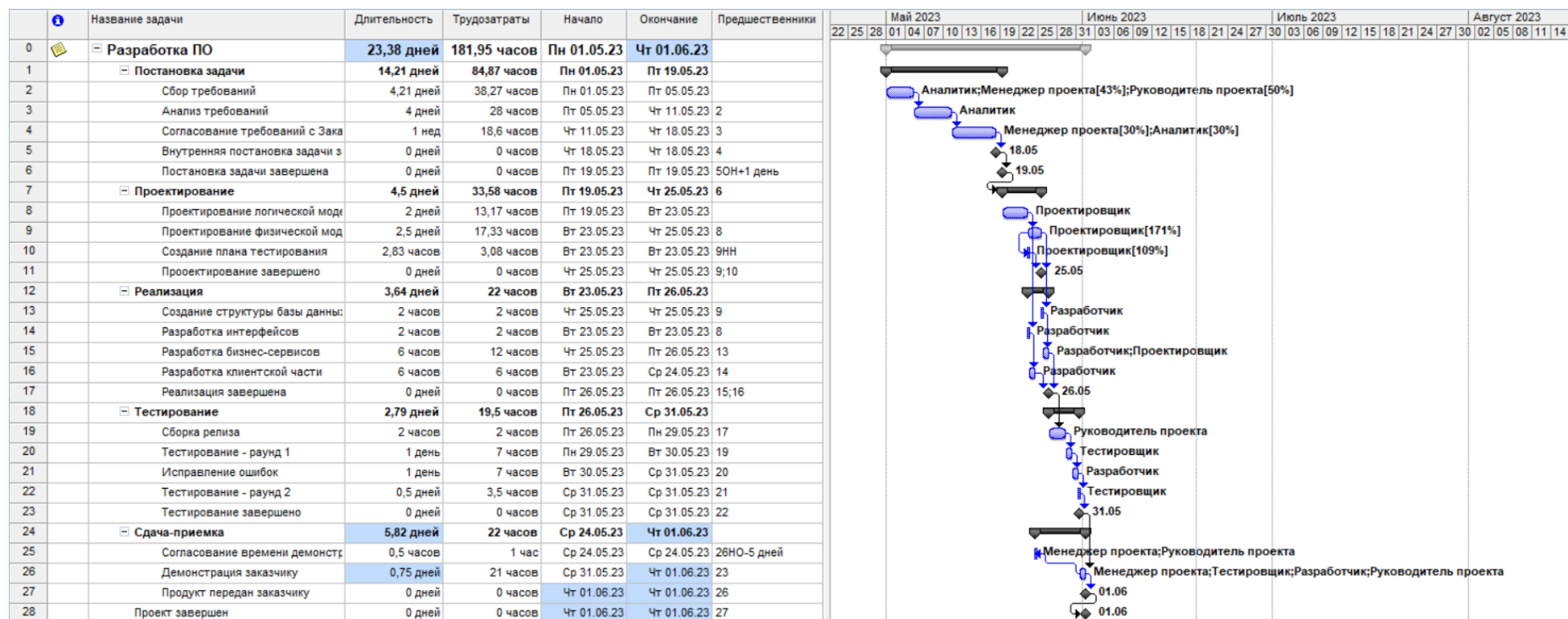


Рисунок Г.1 – Диаграмма Ганта