Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИУК «Информатика и управление»***

КАФЕДРА ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

О Т Ч Е Т

ПРОИЗВОДСТВЕННАЯ ПРАКТИКА

«Эксплуатационная практика»

Студент гр. ИУК4-42Б _____ (Боков А.А.)
(подпись) (Ф.И.О.)

Руководитель _____ (Ameличева К.А.)
(подпись) (Ф.И.О.)

Оценка руководителя _____ баллов _____
30-50 (дата)

Оценка защиты _____ баллов _____
30-50 (дата)

Оценка практики _____ баллов _____
(оценка по пятибалльной шкале)

Комиссия: _____ (Гагарин Ю.Е.)
(подпись) (Ф.И.О.)

_____ (Пчелинцева Н.И.)
(подпись) (Ф.И.О.)

_____ (Ameличева К.А.)
(подпись) (Ф.И.О.)

Калуга, 2024

Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой **__ИУК4___**
_____(Гагарин Ю.Е.)
«01» июля 2024 г.

З А Д А Н И Е
на ЭКСПЛУАТАЦИОННУЮ ПРАКТИКУ

За время прохождения практики студенту необходимо:

1. Определить предметную область, цель, задачи и основные результаты прохождения практики.
2. Выбирать необходимый математический аппарат для реализации задания, выбирать интегрированную среду разработки программного обеспечения, ознакомиться с технологиями создания графического интерфейса в реализуемом программном обеспечении; подобрать стандартные библиотеки; спроектировать компоненты программного продукта.
3. Разработать тестовое окружение, создать тестовые сценарии.
4. Пройти программу обучения в предприятии, решить необходимые задачи, разработать приложение для работы с API предприятия.
5. Подготовить отчет и защитить результаты практики.

Дата выдачи задания «01» июля 2024 г.

Руководитель практики _____ Амеличева К.А.

Задание получил _____ Боков А.А.

ВВЕДЕНИЕ

В мире с каждым днем экспоненциально повышается количество информации во всех сферах жизнедеятельности человечества. Большие объемы данных требуют специализированного подхода к их обработке, анализу, именно поэтому нельзя недооценивать роль аналитики данных и также систем для отчетности.

Аналитика данных и системы для отчетности – неотъемлемая часть любой крупной компании, поскольку благодаря отчетам можно проанализировать работу как предприятия в целом, так и отдельной определенной его части. Анализируя не обособленные, а обобщенные в отчете данные можно получить ценную информацию о динамике, прогнозировать дальнейшее развитие какой-либо сферы. Именно поэтому была выбрана тема информационных систем для отчетности и аналитики.

Целями учебной практики являются: получение опыта работы на предприятии, а также взаимодействии с коллективом, получение новых знаний и умений, углубленное изучение и развитие уже имеющихся, получение итогового продукта в виде приложения для взаимодействия с API предприятия.

Для достижения поставленной цели решаются следующие задачи:

- Изучение теоретического материала
- Выбор языка программирования, среды разработки, набора библиотек
- Прохождение программы обучения для развития необходимых навыков
- Выбор архитектуры приложения
- Разработка, тестирование и применение приложения

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Исследование предметной области задачи и постановка задачи

В качестве предметной области была выбрана область информационных систем для отчетности и аналитики. Данная предметная область включает в себя разработку и развитие ПО для формирования отчетов, анализа данных. Применение технологий из этой области позволяет прогнозировать динамику, избегать излишних расходов и производить оптимизацию производства, находить критически важные места в определенной сфере, обнаруживать периодические неполадки в работе техники. Программные продукты, нацеленные на решение данной задачи, помогают частично или полностью автоматизировать такие бизнес процессы, как: составление отчетов по определенному периоду, анализ больших объемов данных, прогнозирование на основе предыдущих данных, формирование стратегии развития.

В соответствие с предметной областью поставлена следующая задача: разработать приложение для работы с API предприятия, которое бы позволило пользователю просматривать записи, применяя постраничные переходы, фильтрацию, поиск. Приложение должно иметь интерфейс для взаимодействия с системой пользователя, с помощью которого вся необходимая информация предоставлялась бы пользователю. Взаимодействие с API производится путем HTTP-запросов на сервер.

1.2 Обоснование выбора средства реализации

В связи с тем, что в программе обучения, результатом которой будет описанное ранее приложение, применяется язык программирования C#, были сделаны следующие решения:

- В качестве платформы выбрана платформа .NET 4.8
- В качестве языка программирования выбран C#

- В качестве интегрированной среды разработки выбрана среда Visual Studio 2022 Community Edition

Для упрощения разработки были применены следующие основные библиотеки для ЯП C#:

- FontAwesome.WPF – используется для добавления элемента вращающегося индикатора загрузки и его насройки
- Microsoft.Xaml.Behaviors.Wpf – библиотека для добавления интерактивности элементам управления посредством команд, расширении стандартных возможностей и базы элементов XAML
- Newtonsoft.Json – библиотека для работы с JSON – сериализации и десериализации, обработки, модификации данных в формате JSON, полученных от сервера

В качестве технологии разработки был выбрана система построения клиентских приложения WPF – Windows Presentation Foundation.

1.3 Актуальность решаемой проблемы и возможные области применения данной разработки

Актуальность проблемы анализа данных и создания отчетов нельзя недооценивать. Решение данной проблемы может помочь оптимизировать производство, найти локализированные неполадки в работе какой-либо системы, получить динамику данных за заданный период. Именно поэтому проблема остается актуальной и ее решение – востребованным.

Актуальность же разработанного приложения состоит в практической полезности при анализе данных, удобстве просмотра данных организаций и частных клиентов о текущих услугах. Приложение может использоваться для контролирования текущих подключенных устройств, изменений в их работе, неполадках. Немаловажно, что приложение может использоваться как альтернатива варианту страницы в сети Интернет, так как оно использует

меньше ресурсов компьютеров и также не загружает скрипты и стили извне, что приводит к более эффективной работе на маломощных устройствах и устройствах, имеющих малую скорость передачи информации по сети.
Областями применения приложения могут быть:

- Создание отчетов по объектам
- Просмотр организациями и частными клиентами текущих услуг

1.4 Определение входных и выходных характеристик

Приложение предназначается для графического отображения пользователю данных по обслуживаемым предприятием объектам. Необходимо предусмотреть возможность применения фильтрации по определенным критериям, пагинации записей с возможностью как постраничного переключения, так и ввода необходимой страницы, изменения количества записей на странице, а также поиска определенных записей.
Входными данными и характеристиками являются:

1. Данные по объектам, в том числе полученные с применением фильтров, и при использовании поисковой строки, на выбранной странице и в указанном количестве записей на странице, полученные при помощи HTTP-запросов к серверу
2. Набор значений, которые могут принимать каждый из фильтров, полученные из HTTP-запросов к серверу
3. Набор растровых изображений, используемых в графическом интерфейсе приложения

Выходными характеристиками и данными являются:

1. Отображенные в таблице данные, полученные с применением фильтров, пагинации, поиска
2. Адаптивный графический интерфейс (см. прил. 2), в том числе кнопки использования механизма пагинации

Таким образом, поставлена следующая задача: по итогу разработки должно быть получено приложение, реализующее следующий набор функций и включающее перечисленные далее особенности:

- Графический интерфейс (см. прил. 2) для взаимодействия пользователей с приложением
- Меню с выбором необходимой функции
- Реализация работы с API предприятия по запросу пользователя
- Реализация постраничного просмотра записей
- Возможность использования фильтров для фильтрации записей
- Реализация возможности применения поисковой строки для поиска определенных объектов

1.5 Описание используемого математического аппарата

При проектировании, работе с API, разработке использовались различные материалы для изучения как теоретических аспектов, так и практической реализации систем:

- Программа обучения студентов, проходящих практику в организации АО “ЭЛДИС”
- Теоретические основы использования WPF и создания приложения с помощью этой системы [15]
- Теоретический материал по паттерну программирования MVVM, области его применения, рассмотрение достоинств и недостатков, примеры его реализации [10]
- Документация по используемым библиотекам
- Примеры использования и применения используемых библиотек
- Советы и помощь от руководителя практики
- Учебники и руководства пользования языком C# и платформой .NET [12], [13], [14]

2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

2.1 Общие сведения о программе

Данный программный продукт представляет собой оконное приложение, в котором посредством взаимодействия с графическим интерфейсом пользователь может просматривать записи об обслуживаемых на данный момент времени объектах, таких как муниципальные образования, частные дома, и др. Графический интерфейс (см. прил. 2) включает в себя панели верхнего и вложенного уровней. Панель верхнего уровня содержит кнопки выбора категории. Панель вложенного уровня открывается в том случае, если категория содержит подкатегории, в противном случае происходит загрузка данных, содержащихся в выбранной категории. Имеется индикатор в виде вращающихся стрелочек, отображаемых во время длительных операций получения данных от сервера с помощью взаимодействия с API предприятия. В программе предусмотрена возможность постраничного просмотра записей, перехода на определенную страницу путем ввода номера запрашиваемой страницы или переключении с помощью специальных кнопок, ввода строки для поиска среди объектов, применения различных фильтров для фильтрации по определенным критериям записей. В процессе работы программа взаимодействует посредством HTTP-запросов с API предприятия, получая данные об обслуживаемых объектах, имеющемся наборе значений для каждого из доступных фильтров, параметрах пагинации для текущего запроса.

2.2 Проектирование иерархии классов

Иерархия классов в данном проекте была построена на основе паттерна программирования MVVM. В качестве компонента View выступает XAML-разметка, реализующая интерфейс приложения (см. прил. 2), описывающая

основные привязки данных и устанавливающая основные команды, их параметры для интерактивных элементов интерфейса. Компонент Model представляет класс Object, который представляет собой хранилище данных об одном обслуживаемом объекте, который реализует интерфейс INotifyPropertyChanged для оповещения системы об изменении значения какого-либо из полей. Класс ViewModel является последней компонентой паттерна. Он отвечает за получение и дальнейшее сохранение коллекции полученных от сервера объектов, реализацию механизма пагинации, создании, загрузки и настройки фильтров, содержит основные команды для привязки к интерактивным элементам. Также благодаря этому классу производится рендер некоторых элементов управления, производится первоначальная инициализация данных, фильтров. На рис. 1 представлена UML-диаграмма иерархии классов, реализованная в программном продукте.

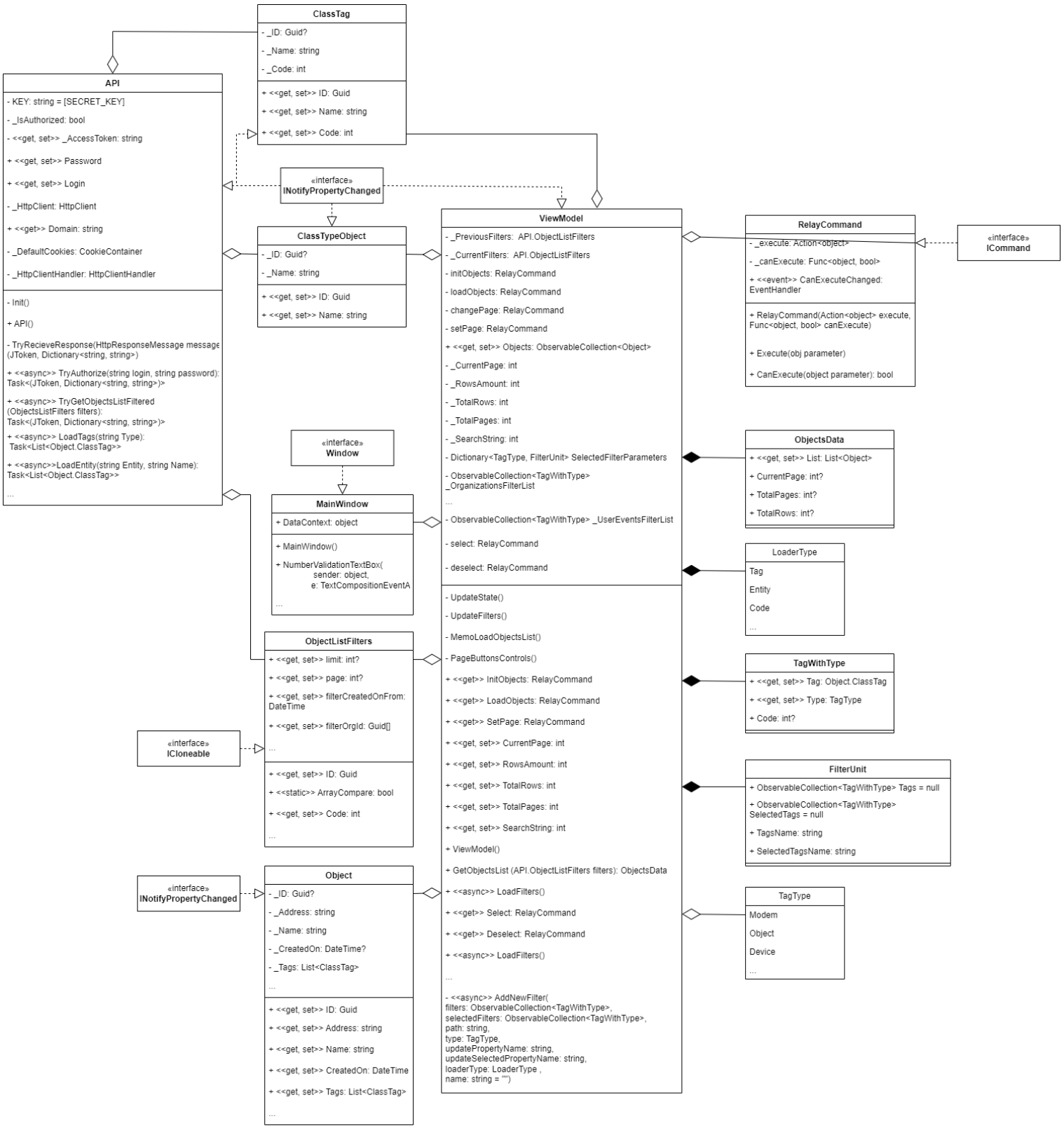

Рис. 1 Иерархия классов приложения

2.3 Описание программных модулей

Программный продукт состоит из нескольких взаимосвязанных модулей, содержащих в себе классы и перечисления:

1. **API.** Данный модуль содержит в себе одноименный класс для взаимодействия с API предприятия, получении данных об объектах и параметрах фильтров в виде JSON токена. Также в этом модуле содержится реализация класса, содержащего поля фильтров – ObjectListFilters.
2. **Object.** Данный модуль содержит в себе класс Object, включающий в себя поля, представляющие собой данные об одном объекте. Также, с помощью установки значений через свойства класса производится оповещение системы об изменении полей. Класс включает вложенные классы для хранения данных об тегах объекта и его типе.
3. **RelayCommand.** В этом модуле содержится реализация интерфейса ICommand в виде класса RelayCommand, который используется для механизма привязки команд и их параметров к интерактивным элементам интерфейса, а также для обработки параметров и вызова функции команды.
4. **ViewModel.** Данный модуль представляет собой класс, который связывает посредством команд, наблюдаемых коллекций и свойств компоненты Model и View. Содержит поля и свойства для использования в механизме привязки в модуле View, функции и набор полей для работы с фильтрами, место сохранения коллекции объектов, команды и их реализации для взаимодействия с View, методы генерации, настройки и рендера некоторых компонентов для View. Класс взаимодействует с классами API, Object, TagWithType и другими.
5. **MainWindow.xaml.cs.** В этом модуле инициализируется объект текущего окна и значением DataContext устанавливается ViewModel. Также в одноименном классе MainWindow содержатся обработчики событий

нажатия на кнопки меню и их реализация. Также в модуле содержится метод расширения встроенной коллекции ObservableCollection и перечисление типа тегов TagType.

6. **MainWindow.xaml**. Данный модуль содержит всю XAML-разметку окна и почти всего интерфейса (см. прил. 2). В нем подключаются необходимые пространства имен для написания разметки, настраиваются параметры окна, указываются привязки, устанавливаются команды для интерактивных элементов и их параметры, описываются стандартные стили элементов, настраиваются иные стили.

2.4 Обработка исключений

В качестве основных обработчиков исключений используется конструкция try-catch, в которую заключены те блоки кода, в которых может возникнуть исключение. Для избегания ошибок, связанных с действиями пользователя, предприняты следующие превентивные меры:

1. Во время загрузки данных из HTTP-запроса, отключается возможность взаимодействия с элементами интерфейса.
2. Поля ввода номера страницы и количества строк, отображаемых на странице, проходят несколько валидаций: валидацию проверки на содержание числового значения и валидацию на ограничение минимального и максимального значений.

2.5 Особенности реализации и оптимизация

К особенностям реализации можно отнести:

1. Механизм кеширования фильтров, что позволяет избежать повторной загрузки данных при условии равенства всех полей фильтров.

2. Данные приложения (список объектов, значения полей фильтров) производятся в асинхронном режиме, что позволяет уменьшить время загрузки.
3. Поскольку реализуется MVVM паттерн, объекты, используемые в механизме привязки, при изменении применяют метод оповещения OnPropertyChanged, таким образом система может отслеживать все изменения и корректно обрабатывать их при отображении.

ЗАКЛЮЧЕНИЕ

В заключении хочу подытожить прогресс выполненной работы и пройденной эксплуатационной практики. В результате прохождения практики было создано приложение с графическим интерфейсом, взаимодействующее с API предприятия, пройдена программа обучения для практикантов АО “ЭЛДИС”, получен незаменимый опыт работы на настоящем предприятии, выполнения задач, предельно близких к реальным, командной работе с коллегами. Несмотря на то, что прохождение практики заняло только две недели, в процессе прохождения и по окончанию было получено множество ценных знаний и навыков. Вместе с тем были обнаружены некоторые темы, требующие дополнительного и более тщательного изучения, что позволит продолжить развиваться и прогрессировать в изучении языка, библиотек, технологий, паттернов, механизмов и др. Вся проделанная работа помогла познакомиться с новым паттерном программирования, получить опыт работы с различными библиотеками, глубже изучить сам язык программирования C#.

Список использованных источников

Основная литература

1. Моделирование информационных ресурсов [Электронный ресурс]: учебно-методический/ Составитель Огнев Э.Н. - Кемерово: Кемеровский государственный университет культуры и искусств, 2013. - 36 с.: ил., табл. - URL: http://biblioclub.ru/index.php?page=book&id=274218

2. Коваленко, Ю.В. Информационно-поисковые системы [Электронный ресурс]: учебно-методическое пособие / Ю.В. Коваленко, Т.А. Сергиенко. — Омск: Омская юридическая академия, 2017. — 38 с.— Режим доступа: http://www.iprbookshop.ru/66817.html

3. Маюрникова, Л. А. Основы научных исследований в научно-технической сфере [Электронный ресурс]: учебно-методическое пособие / Л. А. Маюрникова, С. В. Новосёлов. — Кемерово: Кемеровский технологический институт пищевой промышленности, 2009. — 123 с. — Режим доступа: http://www.iprbookshop.ru/14381.html

4. Вайнштейн, М. З. Основы научных исследований [Электронный ресурс] : учебное пособие / М. З. Вайнштейн, В. М. Вайнштейн, О. В. Кононова. — Йошкар-Ола: Марийский государственный технический университет, Поволжский государственный технологический университет, ЭБС АСВ, 2011. — 216 с. — Режим доступа: http://www.iprbookshop.ru/22586.html

5. Мокий, М.С. Методология научных исследований[Текст]: учебник / М.С. Мокий, А.Л. Никифоров, В.С. Мокий. - М.: Юрайт, 2015. - 255 с.

6. Рогов, В.А. Методика и практика технических экспериментов[Текст]: учеб.пособие / В.А. Рогов, А.В. Антонов, Г.Г. Поздняк. – М.: Академия, 2005. – 288 с.

7. Щербаков, А. Интернет-аналитика [Электронный ресурс]: поиск и оценка информации в web-ресурсах: практическое пособие / А. Щербаков. - М.: Книжный мир, 2012. - 78 с. - URL: http://biblioclub.ru/index.php?page=book&id=89693.

8. Моделирование систем [Текст]: учебник для вузов / С.И. Дворецкий, Ю.Л. Муромцев, В.А. Погонин, А.Г. Схиртладзе. – М.: Академия, 2009. – 320 с.

9. Порсев, Е. Г. Организация и планирование экспериментов [Электронный ресурс]: учебное пособие / Е. Г. Порсев.— Новосибирск : Новосибирский государственный технический университет, 2010. — 155 с. — Режим доступа: http://www.iprbookshop.ru/45415.html

Дополнительная литература

10. Нейс, П. The MVVM Pattern in .NET MAUI [Текст]: The definitive guide to essential patterns, best practices, and techniques for cross-platform app development / П. Нейс. - Бирмингем: Packt Publishing, 2023. - 386 с.

11. Тепляков, С.В. Паттерны проектирования на платформе .NET [Электронный ресурс] / С.В. Тепляков. — СПб.: Питер, 2015. — 320 с.

12. Прайс, М.Д. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов [Текст] / М.Д. Прайс ; переводчик М. Сагалович, С.В. Черников ; редактор Н. Гринчик. — СПб.: Питер, 2018. — 640 с.
13. Троелсен, А., Джепикс, Ф. Язык программирования C# 7 и платформы .NET и .NET Core [Текст] / А. Троелсен, Ф. Джепикс. — М.: Litres, 2019. — 1330 с.
14. Албахари, Д., Албахари, Б. C# 7.0. Полное описание языка. Справочник [Текст] / Д. Албахари, Б. Албахари. — М.: Диалектика, 2018. — 1024 с.
15. Натан, А. WPF 4. Подробное руководство [Текст] / А. Натан. — М.: Символ-Плюс, 2020. — 880 с.
16. Кумар, В., Кровчик, Э., Лагари, Н. .NET Сетевое программирование [Текст] / В. Кумар, Э. Кровчик, Н. Лагари ; переводчик В. Стрельцов ; редактор Н.А. Смольянинова. — М.: Лори, 2014. — 400 с.

ПРИЛОЖЕНИЯ

Приложение 1

Листинг кода

Object.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace Task20
{
    public partial class Object : INotifyPropertyChanged
    {
        public class ClassTypeObject : INotifyPropertyChanged
        {
            private Guid? _ID;
            private string _Name;

            public Guid? ID
            {
                get => _ID;
                set
                {
                    if (_ID != value) _ID = value;
                    OnPropertyChanged("ID");
                }
            }

            public string Name
            {
                get => _Name;
                set
                {
                    if (_Name != value) _Name = value;
                    OnPropertyChanged("Name");
                }
            }

            public static bool operator ==(ClassTypeObject lhs, ClassTypeObject
rhs)
            {
                if (lhs is null)
                {
                    if (rhs is null)
                    {
                        return true;
                    }
                    return false;
                }
                return lhs.Equals(rhs);
            }

            public static bool operator !=(ClassTypeObject lhs, ClassTypeObject
rhs) => !(lhs == rhs);


            public override bool Equals(object obj)
            {
                if (obj == null) return false;
                if (!(obj is ClassTypeObject o)) return false;
                return (o._ID == _ID && o._Name == _Name);
```

```
    }

    public bool Equals(ClassTypeObject obj)
    {
        if (obj is null)
        {
            return false;
        }
        if (Object.ReferenceEquals(this, obj))
        {
            return true;
        }
        if (this.GetType() != obj.GetType())
        {
            return false;
        }
        return (_ID == obj._ID) && (_Name == obj._Name);
    }

    public override string ToString()
    {
        return _Name;
    }
    public event PropertyChangedEventHandler PropertyChanged;
    public void OnPropertyChanged([CallerMemberName] string prop = "")
    {
        if (PropertyChanged != null)
            PropertyChanged(this, new PropertyChangedEventArgs(prop));
    }
};
public class ClassTag : INotifyPropertyChanged
{
    private Guid? _ID;
    private string _Name;
    private int _Code;
    public int Code
    {
        get => _Code;
        set
        {
            if (_Code != value) _Code = value;
            OnPropertyChanged("Code");
        }
    }
    public Guid? ID
    {
        get => _ID;
        set
        {
            if (_ID != value) _ID = value;
            OnPropertyChanged("ID");
        }
    }

    public string Name
    {
        get => _Name;
        set
        {
            if (_Name != value) _Name = value;
            OnPropertyChanged("Name");
        }
    }
    public static bool operator ==(ClassTag lhs, ClassTag rhs)
```

```
            {
                if (lhs is null)
                {
                    if (rhs is null)
                    {
                        return true;
                    }
                    return false;
                }
                return lhs.Equals(rhs);
            }

            public static bool operator !=(ClassTag lhs, ClassTag rhs) => !(lhs
== rhs);
            public override bool Equals(object obj)
            {
                if (!(obj is ClassTag o)) return false;
                return (o._ID == _ID && o._Name == _Name);
            }

            public bool Equals(ClassTag obj)
            {
                if (obj is null)
                {
                    return false;
                }
                if (Object.ReferenceEquals(this, obj))
                {
                    return true;
                }
                if (this.GetType() != obj.GetType())
                {
                    return false;
                }
                return (_ID == obj._ID) && (_Name == obj._Name);
            }

            public override string ToString()
            {
                return _Name;
            }
            public event PropertyChangedEventHandler PropertyChanged;
            public void OnPropertyChanged([CallerMemberName] string prop = "")
            {
                if (PropertyChanged != null)
                    PropertyChanged(this, new PropertyChangedEventArgs(prop));
            }
        }
        public event PropertyChangedEventHandler PropertyChanged;
        public void OnPropertyChanged([CallerMemberName] string prop = "")
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(prop));
        }

        private Guid? _ID;
        private bool? _IsWinterMode;
        private bool? _IsModeChanged;
        private string _Address;
        private ClassTypeObject _TypeObject;
        private string _Name;
        private string _Consumer;
        private DateTime? _CreatedOn;
        private double? _Latitude;
```

```
private double? _Longitude;
private string _Description;
private Guid? _DashboardID;
private string _DashboardName;
private string _ObjectIdentifier;
private bool? _HasEvents;
private List<ClassTag> _Tags;

public Guid? ID
{
    get => _ID;
    set
    {
        if (_ID != value) _ID = value;
        OnPropertyChanged("ID");
    }
}

public bool? IsWinterMode
{
    get => _IsWinterMode;
    set
    {
        if (_IsWinterMode != value) _IsWinterMode = value;
        OnPropertyChanged("IsWinterMode");
    }
}

public bool? IsModeChanged
{
    get => _IsModeChanged;
    set
    {
        if (_IsModeChanged != value) _IsModeChanged = value;
        OnPropertyChanged("IsModeChanged");
    }
}

public string Address
{
    get => _Address;
    set
    {
        if (_Address != value) _Address = value;
        OnPropertyChanged("Address");
    }
}

public ClassTypeObject TypeObject
{
    get => _TypeObject;
    set
    {
        if (_TypeObject != value) _TypeObject = value;
        OnPropertyChanged("TypeObject");
    }
}

public string Name
{
    get => _Name;
    set
    {
        if (_Name != value) _Name = value;
```

```
            OnPropertyChanged("Name");
        }
    }

    public string Consumer
    {
        get => _Consumer;
        set
        {
            if (_Consumer != value) _Consumer = value;
            OnPropertyChanged("Consumer");
        }
    }

    public DateTime? CreatedOn
    {
        get => _CreatedOn;
        set
        {
            if (_CreatedOn != value) _CreatedOn = value;
            OnPropertyChanged("CreatedOn");
        }
    }

    public double? Latitude
    {
        get => _Latitude;
        set
        {
            if (_Latitude != value) _Latitude = value;
            OnPropertyChanged("Latitude");
        }
    }

    public double? Longitude
    {
        get => _Longitude;
        set
        {
            if (_Longitude != value) _Longitude = value;
            OnPropertyChanged("Longitude");
        }
    }

    public string Description
    {
        get => _Description;
        set
        {
            if (_Description != value) _Description = value;
            OnPropertyChanged("Description");
        }
    }

    public Guid? DashboardID
    {
        get => _DashboardID;
        set
        {
            if (_DashboardID != value) _DashboardID = value;
            OnPropertyChanged("DashboardID");
        }
    }
```

```
        public string DashboardName
        {
            get => _DashboardName;
            set
            {
                if (_DashboardName != value) _DashboardName = value;
                OnPropertyChanged("DashboardName");
            }
        }

        public string ObjectIdentifier
        {
            get => _ObjectIdentifier;
            set
            {
                if (_ObjectIdentifier != value) _ObjectIdentifier = value;
                OnPropertyChanged("ObjectIdentifier");
            }
        }

        public bool? HasEvents
        {
            get => _HasEvents;
            set
            {
                if (_HasEvents != value) _HasEvents = value;
                OnPropertyChanged("HasEvents");
            }
        }

        public List<ClassTag> Tags
        {
            get => _Tags;
            set
            {
                if (_Tags != value) _Tags = value;
                OnPropertyChanged("Tags");
            }
        }

        public string TagsString
        {
            get => String.Join("\n", _Tags);

        }
    }
}
```

RelayCommand.cs

```
using System;
using System.Windows.Input;

namespace Task20
{
    public class RelayCommand : ICommand
    {
        private readonly Action<object> _execute;
        private readonly Func<object, bool> _canExecute;

        public RelayCommand(Action<object> execute, Func<object, bool>
canExecute = null)
        {
            _execute = execute ?? throw new
ArgumentNullException(nameof(execute));
```

```
            _canExecute = canExecute;
        }

        public bool CanExecute(object parameter)
        {
            return _canExecute == null || _canExecute(parameter);
        }

        public void Execute(object parameter)
        {
            _execute(parameter);
        }

        public event EventHandler CanExecuteChanged
        {
            add => CommandManager.RequerySuggested += value;
            remove => CommandManager.RequerySuggested -= value;
        }
    }
}
```

ViewModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using Newtonsoft.Json.Linq;
using System.IO;
using System.Collections.ObjectModel;
using FontAwesome.WPF;

namespace Task20
{
    public partial class ViewModel : INotifyPropertyChanged
    {
        private      API.ObjectListFilters      _PreviousFilters      =      new
API.ObjectListFilters() { page = -1 };
        private      API.ObjectListFilters      _CurrentFilters      =      new
API.ObjectListFilters();

        private void UpdateState()
        {
            UpdateFilters();
            PageButtonsControls();
        }
        private void UpdateFilters()
        {
            Guid[]            GetArrayFromList(ObservableCollection<TagWithType>
collection)
            {
                try
                {
                    return  collection.Select(typedTag  =>  typedTag.Tag.ID  ??
default).ToArray();
                }
                catch (Exception ex) { return null; }
            }
            //_PreviousFilters = (API.ObjectListFilters)_CurrentFilters.Clone();

            _CurrentFilters = new API.ObjectListFilters()
```

```
            {
                limit = RowsAmount,
                page = CurrentPage,
                search = SearchString,
                filterCreatedOnFrom = FilterCreatedOnFrom,
                filterCreatedOnTo = FilterCreatedOnTo,
                filterTagModemId                                                =
GetArrayFromList(ModemTagsFilterListSelected),
                filterTagId = GetArrayFromList(ObjectsTagsFilterListSelected),
                filterTagDeviceId                                               =
GetArrayFromList(DevicesTagsFilterListSelected),
                filterTagTVId = GetArrayFromList(TVTagsFilterListSelected),
                filterOrgId = GetArrayFromList(OrganizationsFilterListSelected),
                filterPayerId = GetArrayFromList(OwnersFilterListSelected),
                filterSourceTVId = GetArrayFromList(SourcesFilterListSelected),
                filterAddressLevelId                                            =
GetArrayFromList(MunicipalitiesFilterListSelected),
                filterInspectorId                                               =
GetArrayFromList(InspectorsFilterListSelected),
                filterInpectionAreaId                                           =
GetArrayFromList(InspectionAreaFilterListSelected),
                filterEventCode = UserEventsFilterListSelected.Select(typedTag =>
typedTag.Code.GetValueOrDefault()).ToArray(),
                filterModeId = OperationModeListSelected.Code,
                filterEventSetId = EventsSettingSelected.Code,
            };
        }
        private void MemoLoadObjectsList()
        {
            if (_PreviousFilters != _CurrentFilters)
            {
                _PreviousFilters                                                =
(API.ObjectListFilters)_CurrentFilters.Clone();
                LoadObjectsListAsync(_CurrentFilters);
            }
        }

        private void PageButtonsControls()
        {
            var                             StackPanel                          =
((StackPanel)Application.Current.MainWindow.FindName("PageButtonsStackPanel"));
            StackPanel.Children.Clear();
            if (_TotalPages <= 5)
            {
                for (int i = 1; i <= _TotalPages; i++)
                    StackPanel.Children.Add(new Button()
                    {
                        Name = "ButtonGoToPage" + i.ToString(),
                        Content = i.ToString(),
                        Command = SetPage,
                        CommandParameter = i.ToString()
                    });
            }
            else
            {
                if (_CurrentPage - 2 <= 1)
                {
                    for (int i = _CurrentPage; i <= _CurrentPage + 2; i++)
                        StackPanel.Children.Add(new Button()
                        {
                            Name = "ButtonGoToPage" + i.ToString(),
                            Content = i.ToString(),
                            Command = SetPage,
                            CommandParameter = i.ToString()
```

```
            });

        StackPanel.Children.Add(new Button()
        {
            Name = "ButtonDots",
            Content = "..."
        });

        StackPanel.Children.Add(new Button()
        {
            Name = "ButtonLast",
            Content = _TotalPages.ToString(),
            Command = SetPage,
            CommandParameter = _TotalPages.ToString()
        });
    }
    else if (_CurrentPage + 2 >= _TotalPages)
    {
        StackPanel.Children.Add(new Button()
        {
            Name = "ButtonFirst",
            Content = "1",
            Command = SetPage,
            CommandParameter = "1"
        });
        StackPanel.Children.Add(new Button()
        {
            Name = "ButtonDots",
            Content = "..."
        });
        for (int i = _CurrentPage - 2; i <= _TotalPages; i++)
            StackPanel.Children.Add(new Button()
            {
                Name = "ButtonGoToPage" + i.ToString(),
                Content = i.ToString(),
                Command = SetPage,
                CommandParameter = i.ToString()
            });

    }
    else
    {
        StackPanel.Children.Add(new Button()
        {
            Name = "ButtonFirst",
            Content = "1",
            Command = SetPage,
            CommandParameter = "1"
        });
        StackPanel.Children.Add(new Button()
        {
            Name = "ButtonDots",
            Content = "..."
        });

        for (int i = _CurrentPage - 2; i <= _CurrentPage + 2; i++)
            StackPanel.Children.Add(new Button()
            {
                Name = "ButtonGoToPage" + i.ToString(),
                Content = i.ToString(),
                Command = SetPage,
                CommandParameter = i.ToString()
            });
```

```
                    StackPanel.Children.Add(new Button()
                    {
                        Name = "ButtonDots",
                        Content = "..."
                    });

                    StackPanel.Children.Add(new Button()
                    {
                        Name = "ButtonLast",
                        Content = _TotalPages.ToString(),
                        Command = SetPage,
                        CommandParameter = _TotalPages.ToString()
                    });

                }

            }
        }

        private RelayCommand initObjects;
        public RelayCommand InitObjects
        {
            get
            {
                return initObjects ??
                  (initObjects = new RelayCommand(obj =>
                  {
                      MemoLoadObjectsList();
                      UpdateState();
                      InitFilters();
                  }));
            }
        }

        private RelayCommand loadObjects;
        public RelayCommand LoadObjects
        {
            get
            {
                return loadObjects ??
                  (loadObjects = new RelayCommand(obj =>
                  {
                      MemoLoadObjectsList();
                      UpdateState();
                  }));
            }
        }

        private RelayCommand changePage;
        public RelayCommand ChangePage
        {
            get
            {
                return changePage ??
                  (changePage = new RelayCommand(obj =>
                  {
                      string str = obj as string;
                      if (!int.TryParse(str, out int delta)) return;
                      if (_CurrentPage + delta <= TotalPages && _CurrentPage +
delta > 0)
                      {

                          _CurrentPage += delta;
                          UpdateState();
```

```
                    MemoLoadObjectsList();
                }
            }));
        }
    }

    private RelayCommand setPage;
    public RelayCommand SetPage
    {
        get
        {
            return setPage ??
              (setPage = new RelayCommand(obj =>
              {
                  string str = obj as string;
                  if (!int.TryParse(str, out int page)) return;
                  if (page <= TotalPages && page > 0)
                  {
                      _CurrentPage = page;
                      UpdateState();
                      MemoLoadObjectsList();
                  }
              }));
        }
    }

    private Object _CurrentObject;
    public ObservableCollection<Object> Objects { get; set; } = new
ObservableCollection<Object>();

    private int _CurrentPage = 1;
    public int CurrentPage
    {
        get => _CurrentPage;
        set
        {
            if (_CurrentPage != value && value > 0 && value < TotalPages)
            {
                _CurrentPage = value;
                UpdateState();
            }
            OnPropertyChanged("CurrentPage");
        }
    }

    private int _RowsAmount = 50;
    public int RowsAmount
    {
        get => _RowsAmount;
        set
        {
            if (_RowsAmount != value && value >= 1 && value <= 500)
            {
                _RowsAmount = value;

                UpdateState();
            }
            OnPropertyChanged("RowsAmount");
        }
    }

    private int _TotalRows = 0;
    public int TotalRows
    {
```

```
            get => _TotalRows;
            set
            {
                if (_TotalRows != value && value >= 0) _TotalRows = value;

                OnPropertyChanged("TotalRows");
            }
        }

        private int _TotalPages = 0;
        public int TotalPages
        {
            get => _TotalPages;
            set
            {
                if (_TotalPages != value && value >= 0) _TotalPages = value;

                OnPropertyChanged("TotalPages");
            }
        }

        private string _SearchString = string.Empty;
        public string SearchString
        {
            get => _SearchString;
            set
            {
                _SearchString = value;
                CurrentPage = 1;
                UpdateState();
                OnPropertyChanged("SearchString");
                OnPropertyChanged("CurrentPage");
            }
        }

        public Object CurrentObject
        {
            get => _CurrentObject;
            set
            {
                if (_CurrentObject != value) _CurrentObject = value;

                OnPropertyChanged("CurrentObject");
            }
        }

        public event PropertyChangedEventHandler PropertyChanged;
        public void OnPropertyChanged([CallerMemberName] string prop = "")
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(prop));
        }

        private string[] LoadPasswordAndLogin()
        {
            return File.ReadAllText("lgnpwd.txt").Split(new char[] { ' ' });
        }
        public ViewModel() { }

        private async void LoadObjectsListAsync(API.ObjectListFilters filters)
        {

((Grid)Application.Current.MainWindow.FindName("ObjectsGrid")).IsEnabled       =
false;
```

```
((ImageAwesome)Application.Current.MainWindow.FindName("LoadingSpinner")).Visibi
lity = Visibility.Visible;
            var data = await GetObjectsList(filters);//await Task.Run(() => await
GetObjectsList(filters));
            Objects.Clear();
            Objects = new ObservableCollection<Object>(data.List);
            CurrentPage = data.CurrentPage ?? 1;
            TotalPages = data.TotalPages ?? 1;
            TotalRows = data.TotalRows ?? 0;

((ImageAwesome)Application.Current.MainWindow.FindName("LoadingSpinner")).Visibi
lity = Visibility.Collapsed;

((Grid)Application.Current.MainWindow.FindName("ObjectsGrid")).IsEnabled = true;
            UpdateState();
            OnPropertyChanged("CurrentPage");
            OnPropertyChanged("TotalPages");
            OnPropertyChanged("TotalRows");
            OnPropertyChanged("Objects");
            OnPropertyChanged("CurrentObject");
        }

        public class ObjectsData
        {
            public List<Object> List { get; set; }
            public int? CurrentPage = 0;
            public int? TotalPages = 0;
            public int? TotalRows = 0;
        }

        async Task<ObjectsData> GetObjectsList(API.ObjectListFilters filters)
        {
            var loginAndPassword = LoadPasswordAndLogin();
            API currentAPI = new API { Login = loginAndPassword[0], Password =
loginAndPassword[1] };
            await                    currentAPI.TryAuthorize(loginAndPassword[0],
loginAndPassword[1]);
            var          (Response,          Cookies)          =          await
currentAPI.TryGetObjectsListFiltered(filters);

            var                           ParsedObjects                           =
Response["response"]["objects"]["list"].Select(obj => new Object()
            {
                ID = (Guid?)obj["id"],
                IsWinterMode = (bool?)obj["isWinterMode"],
                IsModeChanged = (bool?)obj["isModeChanged"],
                Address = (string)obj["address"],
                TypeObject = new Object.ClassTypeObject()
                {
                    ID = (Guid?)obj["typeObject"]["id"],
                    Name = (string)obj["typeObject"]["name"],
                },
                Name = (string)obj["name"],
                Consumer = (string)obj["consumer"],
                CreatedOn                                                       =
DateTimeOffset.FromUnixTimeSeconds((long?)obj["createdOn"] ?? 0).UtcDateTime,
                Latitude = (double?)obj["latitude"],
                Longitude = (double?)obj["longitude"],
                Description = (string)obj["description"],
                DashboardID = (Guid?)obj["dashboardID"],
                DashboardName = (string)obj["dashboardName"],
                ObjectIdentifier = (string)obj["objectIdentifier"],
                HasEvents = (bool?)obj["hasEvents"],
```

```
                Tags = obj["tags"].Select(tag => new Object.ClassTag()
                {
                    ID = (Guid?)tag["id"],
                    Name = (string)tag["name"],
                }).ToList(),
            }).ToList();
            return new ObjectsData()
            {
                List = ParsedObjects,
                CurrentPage                                                    =
Response["response"]["objects"]["pagination"]["currentPage"].Value<int?>() ?? 1,
                TotalPages                                                     =
Response["response"]["objects"]["pagination"]["numberOfPages"].Value<int?>()   ??
1,
                TotalRows                                                      =
Response["response"]["objects"]["pagination"]["numberOfRows"].Value<int?>()    ??
0,
            };

        }
        public class TagWithType
        {
            public Object.ClassTag Tag { get; set; }
            public TagType Type { get; set; } = TagType.Default;

            public int? Code = null;
            public override string ToString()
            {
                return Tag.Name;
            }
        }
        enum LoaderType { Tag, Entity, Inspector, Code, Modes, EventSetting,
Default }

        async void LoadFilters()
        {
            var LoginAndPassword = LoadPasswordAndLogin();
            async    Task<bool>    AddNewFilter(ObservableCollection<TagWithType>
filters,
                ObservableCollection<TagWithType> selectedFilters,
                string path,
                TagType type,
                string updatePropertyName,
                string updateSelectedPropertyName,
                LoaderType loaderType,
                string name = ""
                )
            {
                filters?.Clear();
                selectedFilters?.Clear();
                SelectedFilterParameters[type] =
                    new   FilterUnit   {   Tags   =   filters,   SelectedTags   =
selectedFilters,    TagsName    =    updatePropertyName,    SelectedTagsName    =
updateSelectedPropertyName };
                API currentAPI = new API() { Login = LoginAndPassword[0], Password
= LoginAndPassword[1] };
                await               currentAPI.TryAuthorize(LoginAndPassword[0],
LoginAndPassword[1]);
                switch (loaderType)
                {
                    case LoaderType.Entity:
                        filters.AddRange(new
ObservableCollection<TagWithType>((await             currentAPI.LoadEntity(path,
name)).Select(tag => new TagWithType() { Tag = tag, Type = type })));
```

```
                        break;
                    case LoaderType.Tag:
                        filters.AddRange(new
ObservableCollection<TagWithType>((await   currentAPI.LoadTags(path)).Select(tag
=> new TagWithType() { Tag = tag, Type = type })));
                        break;
                    case LoaderType.Inspector:
                        filters.AddRange(new
ObservableCollection<TagWithType>((await         currentAPI.LoadInspectors(path,
name)).Select(tag => new TagWithType() { Tag = tag, Type = type })));
                        break;
                    case LoaderType.Code:
                        filters.AddRange(new
ObservableCollection<TagWithType>((await         currentAPI.LoadUserEvents(path,
name)).Select(tag => new TagWithType() { Tag = tag, Type = type })));
                        break;
                    case LoaderType.Modes:
                        filters.AddRange(new
ObservableCollection<TagWithType>((await     currentAPI.LoadOperationModes(path,
name)).Select(tag => new TagWithType() { Tag = tag, Type = type, Code = tag.Code
})));
                        filters.Add(new    TagWithType()    {    Tag    =    new
Object.ClassTag() { Name = "Не выбрано" }, Type = TagType.OperationMode, Code =
null });

                        break;
                    case LoaderType.EventSetting:
                        filters.AddRange(new
ObservableCollection<TagWithType>((await     currentAPI.LoadEventsSettings(path,
name)).Select(tag => new TagWithType() { Tag = tag, Type = type, Code = tag.Code
})));
                        filters.Add(new    TagWithType()    {    Tag    =    new
Object.ClassTag() { Name = "Не выбрано" }, Type = TagType.EventsSetUp, Code = null
});

                        break;
                    default: break;
                }
                OnPropertyChanged(updatePropertyName);
                OnPropertyChanged(updateSelectedPropertyName);
                return true;
            }
            try
            {

((Grid)Application.Current.MainWindow.FindName("ObjectsGrid")).IsEnabled       =
false;

((ImageAwesome)Application.Current.MainWindow.FindName("LoadingSpinner")).Visibi
lity = Visibility.Visible;

                var Tasks = new[]
                {
                    AddNewFilter(                        _OrganizationsFilterList,
_OrganizationsFilterListSelected,     "organizations",     TagType.Organization,
nameof(OrganizationsFilterList),        nameof(OrganizationsFilterListSelected),
LoaderType.Entity, "name"),
                    AddNewFilter(                          _ObjectsTagsFilterList,
_ObjectsTagsFilterListSelected,           "objects",           TagType.Object,
nameof(ObjectsTagsFilterList),           nameof(ObjectsTagsFilterListSelected),
LoaderType.Tag),
                    AddNewFilter(                            _ModemTagsFilterList,
_ModemTagsFilterListSelected,             "modems",             TagType.Modem,
```

```
nameof(ModemTagsFilterList),                nameof(ModemTagsFilterListSelected),
LoaderType.Tag),
                    AddNewFilter(                          _DevicesTagsFilterList,
_DevicesTagsFilterListSelected,           "devices",           TagType.Device,
nameof(DevicesTagsFilterList),           nameof(DevicesTagsFilterListSelected),
LoaderType.Tag),
                    AddNewFilter( _TVTagsFilterList,  _TVTagsFilterListSelected,
"tv",  TagType.TV,  nameof(TVTagsFilterList),  nameof(TVTagsFilterListSelected),
LoaderType.Tag),
                    AddNewFilter( _OwnersFilterList,  _OwnersFilterListSelected,
"organizations",           TagType.Owner,            nameof(OwnersFilterList),
nameof(OwnersFilterListSelected), LoaderType.Entity, "name"),
                    AddNewFilter(                                 _SourcesFilterList,
_SourcesFilterListSelected,          "sourceObjects",          TagType.Source,
nameof(SourcesFilterList), nameof(SourcesFilterListSelected), LoaderType.Entity,
"addressObject"),
                    AddNewFilter(                     _MunicipalitiesFilterList,
_MunicipalitiesFilterListSelected,    "addressLevels",    TagType.Municipality,
nameof(MunicipalitiesFilterList),     nameof(MunicipalitiesFilterListSelected),
LoaderType.Entity, "name"),
                    AddNewFilter(                          _InspectorsFilterList,
_InspectorsFilterListSelected,          "users",          TagType.Inspector,
nameof(InspectorsFilterList),            nameof(InspectorsFilterListSelected),
LoaderType.Inspector, "name"),
                    AddNewFilter(                     _InspectionAreaFilterList,
_InspectionAreaFilterListSelected,   "organizations",   TagType.InspectionArea,
nameof(InspectionAreaFilterList),     nameof(InspectionAreaFilterListSelected),
LoaderType.Entity, "name"),
                    AddNewFilter(                          _UserEventsFilterList,
_UserEventsFilterListSelected,      "customEventTypes",      TagType.UserEvent,
nameof(UserEventsFilterList),           nameof(UserEventsFilterListSelected),
LoaderType.Code, "name"),
                    AddNewFilter(                     _OperationModesFilterList,
_OperationModesFilterListSelected,    "modeObjects",    TagType.OperationMode,
nameof(OperationModesFilterList),           nameof(OperationModeListSelected),
LoaderType.Modes, "name"),
                    AddNewFilter(                       _EventsSettingFilterList,
_UserEventsFilterListSelected,        "stringMap",        TagType.EventsSetUp,
nameof(EventsSettingFilterList),                nameof(EventsSettingSelected),
LoaderType.EventSetting, "name"),
                };
                await Task.WhenAll(Tasks);

((ImageAwesome)Application.Current.MainWindow.FindName("LoadingSpinner")).Visibi
lity = Visibility.Collapsed;

((Grid)Application.Current.MainWindow.FindName("ObjectsGrid")).IsEnabled = true;

            }
            catch (Exception ex) { return; }
        }

        Dictionary<TagType,   FilterUnit>   SelectedFilterParameters   =   new
Dictionary<TagType, FilterUnit>();

        class FilterUnit
        {
            public ObservableCollection<TagWithType> Tags = null;
            public ObservableCollection<TagWithType> SelectedTags = null;
            public string TagsName = string.Empty;
            public string SelectedTagsName = string.Empty;
        }

        void InitFilters()
```

```
{
    LoadFilters();
}

private RelayCommand commandRefresh;
public RelayCommand CommandRefresh
{
    get
    {
        return commandRefresh ??
          (commandRefresh = new RelayCommand(obj =>
          {
              InitFilters();
          }));
    }
}

private RelayCommand clearModes;
public RelayCommand ClearModes
{
    get
    {
        return clearModes ??
          (clearModes = new RelayCommand(obj =>
          {
              OperationModeListSelected = new TagWithType() { };
              OnPropertyChanged("OperationModeListSelected");
          }));
    }
}

private RelayCommand clearSettings;
public RelayCommand ClearSettings
{
    get
    {
        return clearSettings ??
          (clearSettings = new RelayCommand(obj =>
          {
              EventsSettingSelected = new TagWithType() { };
              OnPropertyChanged("EventsSettingSelected");
          }));
    }
}

#region FilterFields
private DateTime? _FilterCreatedOnFrom = null;
public DateTime? FilterCreatedOnFrom
{
    get => _FilterCreatedOnFrom;
    set
    {
        _FilterCreatedOnFrom = value;
        UpdateState();
        OnPropertyChanged("FilterCreatedOnFrom");
    }
}

private DateTime? _FilterCreatedOnTo = null;
public DateTime? FilterCreatedOnTo
{
    get => _FilterCreatedOnTo;
    set
    {
```

```
                _FilterCreatedOnTo = value;
                UpdateState();
                OnPropertyChanged("FilterCreatedOnTo");
            }
        }

        private ObservableCollection<TagWithType> _OrganizationsFilterList = new
ObservableCollection<TagWithType>();
        private                                 ObservableCollection<TagWithType>
_OrganizationsFilterListSelected = new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> OrganizationsFilterList { get =>
_OrganizationsFilterList; set => _OrganizationsFilterList = value; }
        public ObservableCollection<TagWithType> OrganizationsFilterListSelected
{       get       =>       _OrganizationsFilterListSelected;       set       =>
_OrganizationsFilterListSelected = value; }

        public  ObservableCollection<TagWithType>  _ObjectsTypeFilterList  =  new
ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> _ObjectsTypeFilterListSelected =
new ObservableCollection<TagWithType>();
        public  ObservableCollection<TagWithType> ObjectsTypeFilterList  {  get  =>
_ObjectsTypeFilterList; set => _ObjectsTypeFilterList = value; }
        public ObservableCollection<TagWithType> ObjectsTypeFilterListSelected {
get => _ObjectsTypeFilterListSelected; set => _ObjectsTypeFilterListSelected =
value; }

        public ObservableCollection<TagWithType> _OperationModesFilterList = new
ObservableCollection<TagWithType>();
        public                                  ObservableCollection<TagWithType>
_OperationModesFilterListSelected = new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> OperationModesFilterList { get
=> _OperationModesFilterList; set => _OperationModesFilterList = value; }
        public                                  ObservableCollection<TagWithType>
OperationModesFilterListSelected { get => _OperationModesFilterListSelected; set
=> _OperationModesFilterListSelected = value; }

        public  ObservableCollection<TagWithType>  _ObjectsTagsFilterList  =  new
ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> _ObjectsTagsFilterListSelected =
new ObservableCollection<TagWithType>();
        public  ObservableCollection<TagWithType> ObjectsTagsFilterList  {  get  =>
_ObjectsTagsFilterList; set => _ObjectsTagsFilterList = value; }
        public ObservableCollection<TagWithType> ObjectsTagsFilterListSelected {
get => _ObjectsTagsFilterListSelected; set => _ObjectsTagsFilterListSelected =
value; }

        public  ObservableCollection<TagWithType>  _ModemTagsFilterList  =  new
ObservableCollection<TagWithType>();
        public  ObservableCollection<TagWithType>  _ModemTagsFilterListSelected  =
new ObservableCollection<TagWithType>();
        public  ObservableCollection<TagWithType>  ModemTagsFilterList  {  get  =>
_ModemTagsFilterList; set => _ModemTagsFilterList = value; }
        public  ObservableCollection<TagWithType>  ModemTagsFilterListSelected  {
get => _ModemTagsFilterListSelected; set => _ModemTagsFilterListSelected = value;
}

        public  ObservableCollection<TagWithType>  _DevicesTagsFilterList  =  new
ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> _DevicesTagsFilterListSelected =
new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> DevicesTagsFilterList { get =>
_DevicesTagsFilterList; set => _DevicesTagsFilterList = value; }
```

```
        public ObservableCollection<TagWithType> DevicesTagsFilterListSelected { get => _DevicesTagsFilterListSelected; set => _DevicesTagsFilterListSelected = value; }

        public ObservableCollection<TagWithType> _TVTagsFilterList = new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> _TVTagsFilterListSelected = new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> TVTagsFilterList { get => _TVTagsFilterList; set => _TVTagsFilterList = value; }
        public ObservableCollection<TagWithType> TVTagsFilterListSelected { get => _TVTagsFilterListSelected; set => _TVTagsFilterListSelected = value; }

        public ObservableCollection<TagWithType> _OwnersFilterList = new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> _OwnersFilterListSelected = new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> OwnersFilterList { get => _OwnersFilterList; set => _OwnersFilterList = value; }
        public ObservableCollection<TagWithType> OwnersFilterListSelected { get => _OwnersFilterListSelected; set => _OwnersFilterListSelected = value; }

        public ObservableCollection<TagWithType> _SourcesFilterList = new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> _SourcesFilterListSelected = new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> SourcesFilterList { get => _SourcesFilterList; set => _SourcesFilterList = value; }
        public ObservableCollection<TagWithType> SourcesFilterListSelected { get => _SourcesFilterListSelected; set => _SourcesFilterListSelected = value; }

        public ObservableCollection<TagWithType> _MunicipalitiesFilterList = new ObservableCollection<TagWithType>();
        public                                  ObservableCollection<TagWithType> _MunicipalitiesFilterListSelected = new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> MunicipalitiesFilterList { get => _MunicipalitiesFilterList; set => _MunicipalitiesFilterList = value; }
        public                                  ObservableCollection<TagWithType> MunicipalitiesFilterListSelected { get => _MunicipalitiesFilterListSelected; set => _MunicipalitiesFilterListSelected = value; }

        public ObservableCollection<TagWithType> _InspectorsFilterList = new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> _InspectorsFilterListSelected = new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> InspectorsFilterList { get => _InspectorsFilterList; set => _InspectorsFilterList = value; }
        public ObservableCollection<TagWithType> InspectorsFilterListSelected { get => _InspectorsFilterListSelected; set => _InspectorsFilterListSelected = value; }

        public ObservableCollection<TagWithType> _InspectionAreaFilterList = new ObservableCollection<TagWithType>();
        public                                  ObservableCollection<TagWithType> _InspectionAreaFilterListSelected = new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> InspectionAreaFilterList { get => _InspectionAreaFilterList; set => _InspectionAreaFilterList = value; }
        public                                  ObservableCollection<TagWithType> InspectionAreaFilterListSelected { get => _InspectionAreaFilterListSelected; set => _InspectionAreaFilterListSelected = value; }

        public ObservableCollection<TagWithType> _UserEventsFilterList = new ObservableCollection<TagWithType>();
```

```
        public ObservableCollection<TagWithType> _UserEventsFilterListSelected =
new ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> UserEventsFilterList { get =>
_UserEventsFilterList; set => _UserEventsFilterList = value; }
        public ObservableCollection<TagWithType> UserEventsFilterListSelected {
get => _UserEventsFilterListSelected; set => _UserEventsFilterListSelected =
value; }


        public TagWithType _OperationModeListSelected = new TagWithType();
        public     TagWithType     OperationModeListSelected     {     get     =>
_OperationModeListSelected; set => _OperationModeListSelected = value; }

        public ObservableCollection<TagWithType> _EventsSettingFilterList = new
ObservableCollection<TagWithType>();
        public ObservableCollection<TagWithType> EventsSettingFilterList { get =>
_EventsSettingFilterList; set => _EventsSettingFilterList = value; }
        public TagWithType _EventsSettingSelected = new TagWithType();
        public TagWithType EventsSettingSelected { get => _EventsSettingSelected;
set => _EventsSettingSelected = value; }
        #endregion

        private RelayCommand select;
        public RelayCommand Select
        {
            get
            {
                return select ??
                  (select = new RelayCommand(obj =>
                  {
                      void AddFilter(TagWithType tag)
                      {
                          SelectedFilterParameters[tag.Type].Tags.Remove(tag);

SelectedFilterParameters[tag.Type].SelectedTags.Insert(0, tag);

OnPropertyChanged(SelectedFilterParameters[tag.Type].TagsName);

OnPropertyChanged(SelectedFilterParameters[tag.Type].SelectedTagsName);
                      }
                      if (obj is TagWithType t)
                      {

                          AddFilter(t);
                          CurrentPage = 1;
                          UpdateFilters();
                          OnPropertyChanged("CurrentPage");
                      }
                  }));
            }
        }

        private RelayCommand deselect;
        public RelayCommand Deselect
        {
            get
            {
                return deselect ??
                  (deselect = new RelayCommand(obj =>
                  {
                      void RemoveFilter(TagWithType tag)
                      {
                          SelectedFilterParameters[tag.Type].Tags.Insert(0,
tag);
```

```
SelectedFilterParameters[tag.Type].SelectedTags.Remove(tag);

OnPropertyChanged(SelectedFilterParameters[tag.Type].TagsName);

OnPropertyChanged(SelectedFilterParameters[tag.Type].SelectedTagsName);
                    }
                    if (obj is TagWithType s)
                    {
                        RemoveFilter(s);
                        CurrentPage = 1;
                        UpdateFilters();
                        OnPropertyChanged("CurrentPage");
                    }
                }));
            }
        }

        private RelayCommand changeMode;
        public RelayCommand ChangeMode
        {
            get
            {
                return changeMode ??
                  (changeMode = new RelayCommand(obj =>
                  {
                      if (obj is TagWithType item)
                      {

                          OperationModeListSelected = item;
                          UpdateFilters();
                          OnPropertyChanged("OperationModeListSelected");
                      }
                  }));
            }
        }

        private RelayCommand changeEventSetting;
        public RelayCommand ChangeEventSetting
        {
            get
            {
                return changeEventSetting ??
                  (changeEventSetting = new RelayCommand(obj =>
                  {
                      if (obj is TagWithType item)
                      {
                          EventsSettingSelected = item;
                          UpdateFilters();
                          OnPropertyChanged("EventsSettingSelected");
                      }
                  }));
            }
        }
    }
}
```

MainWindow.xaml.cs

```
using System.Collections.Generic;
using System.Windows;
using System.Windows.Input;
using System.Text.RegularExpressions;
using System.Collections.ObjectModel;
//using System.Windows.Forms;
```

```
namespace Task20
{
    public static class ObservableCollectionExtensions
    {
        public static void AddRange<T>(this ObservableCollection<T> collection,
IEnumerable<T> items)
        {
            foreach (var item in items)
            {
                collection.Add(item);
            }
        }
    }
    public enum TagType { Modem = 1, Object, ObjectType, OperationMode, TV,
Device, Owner, Organization, Municipality, Source, Inspector, InspectionArea,
UserEvent, EventsSetUp, Default }
    public partial class MainWindow : Window
    {

        public MainWindow()
        {
            InitializeComponent();
            DataContext = new ViewModel();

        }
        public void AppliancesRadioButtonChecked(object sender, RoutedEventArgs
e)
        {

            AppliancesScrollViewer.Visibility = Visibility.Visible;
        }

        public void AppliancesRadioButtonUnhecked(object sender, RoutedEventArgs
e)
        {
            AppliancesScrollViewer.Visibility = Visibility.Collapsed;
        }

        public void ObjectsRadioButtonChecked(object sender, RoutedEventArgs e)
        {
            ObjectsScrollViewer.Visibility = Visibility.Visible;
        }

        public void ObjectsRadioButtonUnchecked(object sender, RoutedEventArgs
e)
        {
            ObjectsScrollViewer.Visibility = Visibility.Collapsed;
        }

        public void AccountingRadioButtonChecked(object sender, RoutedEventArgs
e)
        {
            AccountingScrollViewer.Visibility = Visibility.Visible;
        }

        public void AccountingRadioButtonUnhecked(object sender, RoutedEventArgs
e)
        {
            AccountingScrollViewer.Visibility = Visibility.Collapsed;
        }

        public void ReportsRadioButtonChecked(object sender, RoutedEventArgs e)
        {
```

```
            ReportsScrollViewer.Visibility = Visibility.Visible;
        }

        public void ReportsRadioButtonUnhecked(object sender, RoutedEventArgs e)
        {
            ReportsScrollViewer.Visibility = Visibility.Collapsed;
        }

        public void JournalsRadioButtonChecked(object sender, RoutedEventArgs e)
        {
            JournalsScrollViewer.Visibility = Visibility.Visible;
        }

        public void JournalsRadioButtonUnhecked(object sender, RoutedEventArgs
e)
        {
            JournalsScrollViewer.Visibility = Visibility.Collapsed;
        }

        public void AccountsRadioButtonChecked(object sender, RoutedEventArgs e)
        {
            AccountsScrollViewer.Visibility = Visibility.Visible;
        }

        public void AccountsRadioButtonUnhecked(object sender, RoutedEventArgs
e)
        {
            AccountsScrollViewer.Visibility = Visibility.Collapsed;

        }

        public void ReferenceRadioButtonChecked(object sender, RoutedEventArgs
e)
        {
            ReferenceScrollViewer.Visibility = Visibility.Visible;
        }

        public void ReferenceRadioButtonUnhecked(object sender, RoutedEventArgs
e)
        {
            ReferenceScrollViewer.Visibility = Visibility.Collapsed;
        }

        private void ObjectsObjectsRadioButtonCheck(object sender,
RoutedEventArgs e)
        {
            ObjectsGrid.Visibility = Visibility.Visible;
        }

        private void ObjectsObjectsRadioButtonUnchecked(object sender,
RoutedEventArgs e)
        {
            ObjectsGrid.Visibility = Visibility.Visible;
        }


        private void NumberValidationTextBox(object sender,
TextCompositionEventArgs e)
        {
            Regex regex = new Regex("[^0-9]+");
            e.Handled = regex.IsMatch(e.Text);
        }

        private void FiltersScrollViewerClose(object sender, RoutedEventArgs e)
```

```
        {
            FiltersScrollViewer.Visibility = Visibility.Collapsed;
        }
        private void FiltersScrollViewerOpen(object sender, RoutedEventArgs e)
        {
            FiltersScrollViewer.Visibility = Visibility.Visible;
        }
    }
}
```

MainWindow.xaml

```
<Window x:Class="Task20.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Task20"
        xmlns:xf="clr-namespace:XamlFlair;assembly=XamlFlair.WPF"

xmlns:PresentationOptions="http://schemas.microsoft.com/winfx/2006/xaml/presenta
tion/options"
        xmlns:i="http://schemas.microsoft.com/xaml/behaviors"
        xmlns:fa="http://schemas.fontawesome.io/icons/"
        xmlns:li="http://github.com/zeluisping/loadingIndicators/xaml/controls"
        mc:Ignorable="d"
        Title="Меню    ЭЛДИС"    Height="550"    Width="1000"    MinHeight="550"
MinWidth="1000"
        Background="#d0deec"
        xmlns:ui="http://schemas.modernwpf.com/2019"
        ui:WindowHelper.UseModernWindowStyle="True">
    <Window.Resources>
        <Style      x:Key="PlaceHolder"      TargetType="{x:Type      TextBox}"
BasedOn="{StaticResource {x:Type TextBox}}">
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type TextBox}">
                        <Grid>
                            <TextBox Text="{Binding Path=Text,
                                                    RelativeSource={RelativeSource
TemplatedParent},
                                                    Mode=TwoWay,

UpdateSourceTrigger=PropertyChanged}"
                                     x:Name="textSource"
                                     Background="Transparent"
                                     Panel.ZIndex="2" />
                            <TextBox         Text="{TemplateBinding         Tag}"
Background="{TemplateBinding Background}" Panel.ZIndex="1">
                                <TextBox.Style>
                                    <Style TargetType="{x:Type TextBox}">
                                        <Setter            Property="Foreground"
Value="Transparent"/>
                                        <Style.Triggers>
                                            <DataTrigger       Binding="{Binding
Path=Text, Source={x:Reference textSource}}" Value="">
                                                <Setter    Property="Foreground"
Value="LightGray"/>
                                            </DataTrigger>
                                        </Style.Triggers>
                                    </Style>
                                </TextBox.Style>
                            </TextBox>
                        </Grid>
                    </ControlTemplate>
```

```
                </Setter.Value>
            </Setter>
        </Style>

        <Style TargetType="StackPanel">
            <Setter Property="Background" Value="#2a64a6"/>
        </Style>
        <Style TargetType="RadioButton">
            <Setter Property="HorizontalAlignment" Value="Center"/>

            <Setter Property="Background" Value="#2a64a6"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type RadioButton}">
                        <BulletDecorator Cursor="Hand">
                            <BulletDecorator.Bullet>
                                <TextBlock></TextBlock>
                            </BulletDecorator.Bullet>

                            <TextBlock    Margin="3,1,0,0"    Foreground="White"
FontFamily="Calibri" FontSize="16">
                        <ContentPresenter />
                            </TextBlock>
                        </BulletDecorator>


                    </ControlTemplate>
                </Setter.Value>
            </Setter>


        </Style>

        <Style TargetType="TextBlock">
            <Setter Property="HorizontalAlignment" Value="Center"/>
            <Setter Property="TextWrapping" Value="Wrap"/>
            <Setter Property="FontSize" Value="14"/>
        </Style>
    </Window.Resources>
    <Grid HorizontalAlignment="Stretch" VerticalAlignment="Stretch">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition MinWidth="200px" Width="*"/>
        </Grid.ColumnDefinitions>

        <StackPanel Orientation="Horizontal">

            <fa:ImageAwesome      Name="LoadingAnimation"      Panel.ZIndex="100"
Visibility="Collapsed"   Grid.Column="1"  Icon="Refresh"  Spin="True"  Height="48"
Width="48" />
            <ScrollViewer Grid.Column="0" >
                <StackPanel         Margin="10px"         Orientation="Vertical"
Name="MainStackPanel" >
                    <Image>
                        <Image.Source>
                            <FormatConvertedBitmap
Source="Resources/Images/eldispng.png"></FormatConvertedBitmap>
                        </Image.Source>
                    </Image>
                    <Separator/>

                    <RadioButton                                Name="HomeRadioButton"
GroupName="ToolBarButtonsGroup">
                        <BulletDecorator>
```

```
                            <StackPanel                    Orientation="Vertical"
VerticalAlignment="Center">
                                <Image Width="32" Height="32">
                                    <Image.Source>
                                        <FormatConvertedBitmap
Source="Resources/Images/main.png"></FormatConvertedBitmap>
                                    </Image.Source>
                                </Image>
                                <TextBlock            VerticalAlignment="Center"
Foreground="White">Главная</TextBlock>
                            </StackPanel>
                        </BulletDecorator>
                    </RadioButton>
                    <Separator/>

                    <RadioButton                  Name="MnemoshcemesRadioButton"
GroupName="ToolBarButtonsGroup">
                        <BulletDecorator>
                            <StackPanel                    Orientation="Vertical"
VerticalAlignment="Center">
                                <Image Width="32" Height="32">
                                    <Image.Source>
                                        <FormatConvertedBitmap
Source="Resources/Images/mnemoschemes.png"></FormatConvertedBitmap>
                                    </Image.Source>
                                </Image>
                                <TextBlock            VerticalAlignment="Center"
Foreground="White">Мнемо-схемы</TextBlock>
                            </StackPanel>
                        </BulletDecorator>
                    </RadioButton>
                    <Separator/>

                    <RadioButton                       Name="ObjectsRadioButton"
GroupName="ToolBarButtonsGroup"          Unchecked="ObjectsRadioButtonUnchecked"
Checked="ObjectsRadioButtonChecked">
                        <BulletDecorator>
                            <StackPanel                    Orientation="Vertical"
VerticalAlignment="Center">
                                <Image Width="32" Height="32">
                                    <Image.Source>
                                        <FormatConvertedBitmap
Source="Resources/Images/objects.png"></FormatConvertedBitmap>
                                    </Image.Source>
                                </Image>
                                <TextBlock            VerticalAlignment="Center"
Foreground="White">Объекты</TextBlock>
                            </StackPanel>
                        </BulletDecorator>
                    </RadioButton>
                    <Separator/>

                    <RadioButton                     Name="AppliancesRadioButton"
GroupName="ToolBarButtonsGroup"        Unchecked="AppliancesRadioButtonUnhecked"
Checked="AppliancesRadioButtonChecked">
                        <BulletDecorator>
                            <StackPanel                    Orientation="Vertical"
VerticalAlignment="Center">
                                <Image Width="32" Height="32">
                                    <Image.Source>
                                        <FormatConvertedBitmap
Source="Resources/Images/appliances.png"></FormatConvertedBitmap>
                                    </Image.Source>
                                </Image>
```

```
                                <TextBlock            VerticalAlignment="Center"
Foreground="White">Приборы</TextBlock>
                            </StackPanel>
                        </BulletDecorator>
                    </RadioButton>
                    <Separator/>

                    <RadioButton                    Name="AccountingRadioButton"
GroupName="ToolBarButtonsGroup"        Unchecked="AccountingRadioButtonUnhecked"
Checked="AccountingRadioButtonChecked">
                        <BulletDecorator>
                            <StackPanel                    Orientation="Vertical"
VerticalAlignment="Center">
                                <Image Width="32" Height="32">
                                    <Image.Source>
                                        <FormatConvertedBitmap
Source="Resources/Images/accounting.png"></FormatConvertedBitmap>
                                    </Image.Source>
                                </Image>
                                <TextBlock            VerticalAlignment="Center"
Foreground="White">Учет</TextBlock>
                            </StackPanel>
                        </BulletDecorator>
                    </RadioButton>
                    <Separator/>

                    <RadioButton                     Name="ReportsRadioButton"
GroupName="ToolBarButtonsGroup"           Unchecked="ReportsRadioButtonUnhecked"
Checked="ReportsRadioButtonChecked">
                        <BulletDecorator>
                            <StackPanel                    Orientation="Vertical"
VerticalAlignment="Center">
                                <Image Width="32" Height="32">
                                    <Image.Source>
                                        <FormatConvertedBitmap
Source="Resources/Images/reports.png"></FormatConvertedBitmap>
                                    </Image.Source>
                                </Image>
                                <TextBlock            VerticalAlignment="Center"
Foreground="White">Отчеты</TextBlock>
                            </StackPanel>
                        </BulletDecorator>
                    </RadioButton>
                    <Separator/>

                    <RadioButton                    Name="JournalsRadioButton"
GroupName="ToolBarButtonsGroup"          Unchecked="JournalsRadioButtonUnhecked"
Checked="JournalsRadioButtonChecked">
                        <BulletDecorator>
                            <StackPanel                    Orientation="Vertical"
VerticalAlignment="Center">
                                <Image Width="32" Height="32">
                                    <Image.Source>
                                        <FormatConvertedBitmap
Source="Resources/Images/journals.png"></FormatConvertedBitmap>
                                    </Image.Source>
                                </Image>
                                <TextBlock            VerticalAlignment="Center"
Foreground="White">Журналы</TextBlock>
                            </StackPanel>
                        </BulletDecorator>
                    </RadioButton>
                    <Separator/>
```

```
                    <RadioButton                        Name="AccountsRadioButton"
GroupName="ToolBarButtonsGroup"          Unchecked="AccountsRadioButtonUnhecked"
Checked="AccountsRadioButtonChecked">
                        <BulletDecorator>
                            <StackPanel                    Orientation="Vertical"
VerticalAlignment="Center">
                                <Image Width="32" Height="32">
                                    <Image.Source>
                                        <FormatConvertedBitmap
Source="Resources/Images/accounts.png"></FormatConvertedBitmap>
                                    </Image.Source>
                                </Image>
                                <TextBlock            VerticalAlignment="Center"
Foreground="White">Учетные записи</TextBlock>
                            </StackPanel>
                        </BulletDecorator>
                    </RadioButton>

                    <Separator/>

                    <RadioButton                  Name="TechnoschemesRadioButton"
GroupName="ToolBarButtonsGroup">
                        <BulletDecorator>
                            <StackPanel                    Orientation="Vertical"
VerticalAlignment="Center">
                                <Image Width="32" Height="32">
                                    <Image.Source>
                                        <FormatConvertedBitmap
Source="Resources/Images/technoschemes.png"></FormatConvertedBitmap>
                                    </Image.Source>
                                </Image>
                                <TextBlock            VerticalAlignment="Center"
Foreground="White">Техно-схемы</TextBlock>
                            </StackPanel>
                        </BulletDecorator>
                    </RadioButton>
                    <Separator/>

                    <RadioButton                      Name="ReferenceRadioButton"
GroupName="ToolBarButtonsGroup"          Unchecked="ReferenceRadioButtonUnhecked"
Checked="ReferenceRadioButtonChecked">
                        <BulletDecorator>
                            <StackPanel                    Orientation="Vertical"
VerticalAlignment="Center">
                                <Image Width="32" Height="32">
                                    <Image.Source>
                                        <FormatConvertedBitmap
Source="Resources/Images/reference.png"></FormatConvertedBitmap>
                                    </Image.Source>
                                </Image>
                                <TextBlock            VerticalAlignment="Center"
Foreground="White">Справка</TextBlock>
                            </StackPanel>
                        </BulletDecorator>
                    </RadioButton>
                    <Separator/>
                </StackPanel>
            </ScrollViewer>


            <!--ОБЪЕКТЫ-->
            <ScrollViewer Name="ObjectsScrollViewer"  Visibility="Collapsed">
                <StackPanel         Margin="10px"         Orientation="Vertical"
Name="ObjectsStackPanel">
```

```
                    <TextBlock FontSize="20"><Bold>Объекты</Bold></TextBlock>
                    <Separator/>

                    <RadioButton                Name="ObjectsObjectsRadioButton"
GroupName="SubToolBarsGroup"           Command="{Binding           InitObjects}"
Checked="ObjectsObjectsRadioButtonCheck"
Unchecked="ObjectsObjectsRadioButtonUnchecked">Объекты</RadioButton>
                    <RadioButton                  Name="ObjectsFlatsRadioButton"
GroupName="SubToolBarsGroup">Квартиры</RadioButton>

                </StackPanel>
            </ScrollViewer>

            <!--ПРИБОРЫ-->
            <ScrollViewer Name="AppliancesScrollViewer"  Visibility="Collapsed">

                <StackPanel         Margin="10px"         Orientation="Vertical"
HorizontalAlignment="Center" Name="AppliancesStackPanel">
                    <TextBlock FontSize="20"><Bold>Приборы</Bold></TextBlock>
                    <Separator/>

                    <TextBlock Foreground="#bfbcbb">Общий учет</TextBlock>
                    <Separator/>

                    <RadioButton             Name="AppliancesModems1RadioButton"
GroupName="SubToolBarsGroup">Модемы</RadioButton>
                    <RadioButton           Name="AppliancesMetering1RadioButton"
GroupName="SubToolBarsGroup">Приборы учета</RadioButton>
                    <RadioButton           Name="AppliancesSensorsPURadioButton"
GroupName="SubToolBarsGroup">Датчики на ПУ</RadioButton>
                    <RadioButton          Name="AppliancesRegulatorsRadioButton"
GroupName="SubToolBarsGroup">Регуляторы</RadioButton>
                    <RadioButton             Name="AppliancesSensorsRadioButton"
GroupName="SubToolBarsGroup">Датчики</RadioButton>
                    <TextBlock               Foreground="#bfbcbb">Индивидуальный
учет</TextBlock>
                    <Separator/>
                    <RadioButton             Name="AppliancesModems2RadioButton"
GroupName="SubToolBarsGroup">Модемы</RadioButton>
                    <RadioButton           Name="AppliancesMetering2RadioButton"
GroupName="SubToolBarsGroup">Приборы учета</RadioButton>
                </StackPanel>
            </ScrollViewer>


            <!--УЧЕТ-->

            <ScrollViewer Name="AccountingScrollViewer"  Visibility="Collapsed">
                <StackPanel         Margin="10px"         Orientation="Vertical"
Name="AccountingStackPanel" >
                    <TextBlock FontSize="20"><Bold>Учет</Bold></TextBlock>
                    <Separator/>
                    <RadioButton                 Name="AccountingXBCRadioButton"
GroupName="SubToolBarsGroup">ХВС</RadioButton>
                    <RadioButton                 Name="AccountingGBCRadioButton"
GroupName="SubToolBarsGroup">ГВС</RadioButton>
                    <RadioButton                  Name="AccountingTCRadioButton"
GroupName="SubToolBarsGroup">ТС</RadioButton>
                    <RadioButton           Name="AccountingTCInSteamRadioButton"
GroupName="SubToolBarsGroup">ТС в паре</RadioButton>
                    <RadioButton          Name="AccountingElectricityadioButton"
GroupName="SubToolBarsGroup">Эл. энергия</RadioButton>
```

```
                    <RadioButton                  Name="AccountingGasRadioButton"
GroupName="SubToolBarsGroup">Газ</RadioButton>
                    <RadioButton          Name="AccountingWastewaterRadioButton"
GroupName="SubToolBarsGroup">Сточные воды</RadioButton>
                    <RadioButton         Name="AccountingVentilationRadioButton"
GroupName="SubToolBarsGroup">Вентиляция</RadioButton>
                    <RadioButton     Name="AccountingWithoutResourceRadioButton"
GroupName="SubToolBarsGroup">Без ресурса</RadioButton>
                    <RadioButton Name="AccountingAllAccountingPointsRadioButton"
GroupName="SubToolBarsGroup">Все точки учета</RadioButton>
                    <RadioButton          Name="AccountingAllTYFilesRadioButton"
GroupName="SubToolBarsGroup">Файлы на ТУ</RadioButton>
                    <RadioButton  Name="AccountingRegulatorsCircuitsRadioButton"
GroupName="SubToolBarsGroup">Контуры регуляторов</RadioButton>
                    <RadioButton             Name="AccountingArchiveRadioButton"
GroupName="SubToolBarsGroup">Наличие архива в точках учета</RadioButton>
                    <RadioButton     Name="AccountingSourceConsumersRadioButton"
GroupName="SubToolBarsGroup">Источкники/потребители</RadioButton>
                </StackPanel>
            </ScrollViewer>

            <!--ОТЧЕТЫ-->

            <ScrollViewer Name="ReportsScrollViewer"  Visibility="Collapsed">
                <StackPanel         Margin="10px"         Orientation="Vertical"
Name="ReportsStackPanel">
                    <TextBlock FontSize="20"><Bold>Отчеты</Bold></TextBlock>
                    <Separator/>
                    <RadioButton                  Name="ReportsSystemRadioButton"
GroupName="SubToolBarsGroup">Системные</RadioButton>
                    <RadioButton                   Name="ReportsUsersRadioButton"
GroupName="SubToolBarsGroup">Пользовательские</RadioButton>
                    <RadioButton                Name="ReportsHistoryRadioButton"
GroupName="SubToolBarsGroup">История</RadioButton>
                    <RadioButton             Name="ReportsSubscribesRadioButton"
GroupName="SubToolBarsGroup">Подписки</RadioButton>
                </StackPanel>
            </ScrollViewer>

            <!--ЖУРНАЛЫ-->

            <ScrollViewer Name="JournalsScrollViewer"  Visibility="Collapsed">
                <StackPanel         Margin="10px"         Orientation="Vertical"
Name="JournalsStackPanel">
                    <TextBlock FontSize="20"><Bold>Журналы</Bold></TextBlock>
                    <Separator/>
                    <RadioButton            Name="JournalsUserEventsRadioButton"
GroupName="SubToolBarsGroup">Польз. события</RadioButton>
                    <RadioButton                   Name="JournalTagsRadioButton"
GroupName="SubToolBarsGroup">Теги</RadioButton>
                    <RadioButton          Name="JournalMassOperationRadioButton"
GroupName="SubToolBarsGroup">Массовые операции</RadioButton>
                </StackPanel>
            </ScrollViewer>

            <!--УЧЕТНЫЕ ЗАПИСИ-->

            <ScrollViewer Name="AccountsScrollViewer"  Visibility="Collapsed">
                <StackPanel         Margin="10px"         Orientation="Vertical"
Name="AccountsStackPanel">
                    <TextBlock                        FontSize="20"><Bold>Учетные
записи</Bold></TextBlock>
                    <Separator/>
```

```
                    <RadioButton          Name="AccountOrganisationsRadioButton"
GroupName="SubToolBarsGroup">Организации</RadioButton>
                </StackPanel>
            </ScrollViewer>

            <!--СПРАВКА-->

            <ScrollViewer Name="ReferenceScrollViewer"  Visibility="Collapsed">
                <StackPanel         Margin="10px"         Orientation="Vertical"
Name="ReferenceStackPanel">
                    <TextBlock FontSize="20"><Bold>Справка</Bold></TextBlock>
                    <Separator/>
                    <TextBlock Foreground="#bfbcbb">Инструкции</TextBlock>
                    <RadioButton                  Name="ReferenceAUUCRadioButton"
GroupName="SubToolBarsGroup">Работа в АИИС</RadioButton>
                    <RadioButton         Name="ReferenceConnectinoPURadioButton"
GroupName="SubToolBarsGroup">Подключение ПУ</RadioButton>
                    <RadioButton         Name="ReferenceYCPDSettingsRadioButton"
GroupName="SubToolBarsGroup">Настройки УСПД</RadioButton>
                    <TextBlock Foreground="#bfbcbb">Обратная связь</TextBlock>
                    <Separator/>
                    <RadioButton              Name="ReferenceTicketsRadioButton"
GroupName="SubToolBarsGroup">Тикеты (задать вопрос)</RadioButton>
                    <TextBlock     HorizontalAlignment="Center">     <Hyperlink
Foreground="WhiteSmoke"
NavigateUri="support@eldis24.ru">support@eldis24.ru</Hyperlink></TextBlock>
                    <TextBlock> <Bold> 8 800 775-13-93 доб. 2</Bold></TextBlock>

                    <TextBlock               Foreground="#bfbcbb">Информационные
каналы</TextBlock>
                    <Separator/>
                    <TextBlock><Hyperlink                Foreground="WhiteSmoke"
NavigateUri="https://new.eldis24.ru/main/support">Новости</Hyperlink></TextBlock
>
                    <TextBlock><Hyperlink                Foreground="WhiteSmoke"
NavigateUri="https://new.eldis24.ru/main/support">Группа
VK</Hyperlink></TextBlock>
                    <TextBlock><Hyperlink                Foreground="WhiteSmoke"
NavigateUri="https://new.eldis24.ru/main/support">Telegram</Hyperlink></TextBloc
k>
                    <TextBlock><Hyperlink                Foreground="WhiteSmoke"
NavigateUri="https://new.eldis24.ru/main/support">Youtube</Hyperlink></TextBlock
>
                    <TextBlock><Hyperlink                Foreground="WhiteSmoke"
NavigateUri="https://new.eldis24.ru/main/support">Магазин</Hyperlink></TextBlock
>
                </StackPanel>
            </ScrollViewer>


        </StackPanel>
        <fa:ImageAwesome       Name="LoadingSpinner"       Visibility="Collapsed"
Grid.Column="1" Icon="Refresh" Spin="True" SpinDuration="10" Foreground="#2a64a6"
Panel.ZIndex="100" Height="125" Width="125"/>

        <Grid  HorizontalAlignment="Stretch"  Name="ObjectsGrid"  Grid.Column="1"
Visibility="Collapsed">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="Auto"/>
            </Grid.RowDefinitions>
```

```
            <DockPanel              Background="#2a64a6"              Grid.Row="0"
HorizontalAlignment="Stretch">
                <TextBlock   Margin="10   0   10   0"   HorizontalAlignment="Left"
FontSize="20" Foreground="White">Объекты</TextBlock>
                <Image HorizontalAlignment="Center" Margin="10 0 10 0" Width="32"
Height="32">
                    <Image.Source>
                        <FormatConvertedBitmap
Source="Resources/Images/glass.png"/>
                    </Image.Source>
                </Image>

                <TextBox Name="SearchTextBox"
                         MinWidth="350"
                         Text="{Binding        SearchString,        Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"
                         Width="{Binding                                ActualWidth,
ElementName=SearchTextBox}"
                         TextWrapping="Wrap"
                         Style="{StaticResource PlaceHolder}"
                         Tag="Поиск и фильтр"
                         HorizontalAlignment="Center"
                         GotMouseCapture="FiltersScrollViewerOpen">
                    <i:Interaction.Triggers>
                        <i:EventTrigger EventName="GotFocus">
                            <i:InvokeCommandAction             Command="{Binding
RefreshFilters}"/>
                            <!--<i:InvokeCommandAction         Command="{Binding
LoadObjects}"/>-->
                        </i:EventTrigger>
                    </i:Interaction.Triggers>
                    <TextBox.InputBindings>
                        <KeyBinding       Command="{Binding       LoadObjects}"
Key="Enter"></KeyBinding>

                    </TextBox.InputBindings>
                </TextBox>
                <Button   HorizontalAlignment="Center"   Margin="10   0   10   0"
Command="{Binding CommandRefresh}">
                    <Image Width="32" Height="32">
                        <Image.Source>
                            <FormatConvertedBitmap
Source="Resources/Images/refresh.png"/>
                        </Image.Source>
                    </Image>
                </Button>
                <TextBlock       HorizontalAlignment="Right"       FontSize="20"
Foreground="White"   Margin="10   0   10   0"   Text="{Binding   Path=TotalRows,
StringFormat=Найдено {0} записей}"></TextBlock>
            </DockPanel>

            <ScrollViewer      Name="FiltersScrollViewer"      Panel.ZIndex="50"
Grid.Row="1"     Visibility="Collapsed"     MouseDown="FiltersScrollViewerClose"
VerticalScrollBarVisibility="Visible">
                <i:Interaction.Triggers>
                    <i:EventTrigger EventName="MouseDown">
                        <i:InvokeCommandAction Command="{Binding LoadObjects}"/>
                    </i:EventTrigger>
                </i:Interaction.Triggers>
                <Grid  Name="FiltersGrid"  Panel.ZIndex="50"  Background="White"
HorizontalAlignment="Center">
                    <Grid.RowDefinitions>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
```

```
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="Auto"/>
                    </Grid.RowDefinitions>
                    <DockPanel Margin="0 10 0 10" Grid.Row="0">
                        <StackPanel  Background="White"  Orientation="Vertical"
HorizontalAlignment="Left">
                            <TextBlock HorizontalAlignment="Center" Grid.Row="0"
Grid.Column="0" Margin="10 0 10 0">Созданы с</TextBlock>
                            <DatePicker             HorizontalAlignment="Center"
Grid.Row="1"  Grid.Column="0"  Margin="10  0  10  0"  SelectedDate="{Binding
FilterCreatedOnFrom,                        UpdateSourceTrigger=PropertyChanged,
Mode=TwoWay}"></DatePicker>
                        </StackPanel>
                        <StackPanel  Background="White"  Orientation="Vertical"
HorizontalAlignment="Right">
                            <TextBlock HorizontalAlignment="Center" Grid.Row="0"
Grid.Column="1" Margin="10 0 10 0">Созданы по</TextBlock>
                            <DatePicker             HorizontalAlignment="Center"
Grid.Row="1"  Grid.Column="1"  Margin="10  0  10  0"  SelectedDate="{Binding
FilterCreatedOnTo,                          UpdateSourceTrigger=PropertyChanged,
Mode=TwoWay}"></DatePicker>
                        </StackPanel>
                    </DockPanel>

                    <DockPanel Margin="0 10 0 10" Grid.Row="1">
                        <TextBlock HorizontalAlignment="Left" FontWeight="Bold"
TextWrapping="Wrap" FontSize="18" Margin="10 0 10 0">Добавить фильтры</TextBlock>
                        <TextBlock HorizontalAlignment="Right" FontWeight="Bold"
TextWrapping="Wrap" FontSize="18" Margin="10 0 10 0">Удалить фильтры</TextBlock>
                    </DockPanel>

                    <!--организации-->
                    <DockPanel    Margin="0    10    0    10"    Grid.Row="2"
HorizontalAlignment="Stretch">
                        <ComboBox               Name="OrganizationsComboBoxAdd"
HorizontalAlignment="Left"    ItemsSource="{Binding    OrganizationsFilterList,
UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
Select}"    CommandParameter="{Binding    ElementName=OrganizationsComboBoxAdd,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                        <TextBlock  FontSize="14"  HorizontalAlignment="Center"
TextWrapping="Wrap" Margin="10 0 10 0">Организации</TextBlock>
                        <ComboBox             Name="OrganizationsComboBoxRemove"
HorizontalAlignment="Right"                                ItemsSource="{Binding
OrganizationsFilterListSelected, UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
```

```
                                    <i:InvokeCommandAction     Command="{Binding
Deselect}"  CommandParameter="{Binding  ElementName=OrganizationsComboBoxRemove,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                    </DockPanel>

                    <!--типы объектов-->
                    <DockPanel     Margin="0     10     0     10"     Grid.Row="3"
HorizontalAlignment="Stretch">
                        <ComboBox                   Name="ObjectsTagsComboBoxAdd"
HorizontalAlignment="Left"     ItemsSource="{Binding     ObjectsTagsFilterList,
UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
Select}"     CommandParameter="{Binding     ElementName=ObjectsTagsComboBoxAdd,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                        <TextBlock   FontSize="14"   HorizontalAlignment="Center"
TextWrapping="Wrap" Margin="10 0 10 0">Теги объектов</TextBlock>
                        <ComboBox               Name="ObjectsTagsComboBoxRemove"
HorizontalAlignment="Right" ItemsSource="{Binding ObjectsTagsFilterListSelected,
UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
Deselect}"   CommandParameter="{Binding   ElementName=ObjectsTagsComboBoxRemove,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                    </DockPanel>
                    <!--модемы-->
                    <DockPanel     Margin="0     10     0     10"     Grid.Row="4"
HorizontalAlignment="Stretch">
                        <ComboBox    Text="asdfsd"    Name="ModemTagsComboBoxAdd"
HorizontalAlignment="Left"      ItemsSource="{Binding      ModemTagsFilterList,
UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
Select}"      CommandParameter="{Binding      ElementName=ModemTagsComboBoxAdd,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                        <TextBlock   FontSize="14"   HorizontalAlignment="Center"
TextWrapping="Wrap" Margin="10 0 10 0">Теги модемов</TextBlock>
                        <StackPanel  Background="White"  Orientation="Horizontal"
HorizontalAlignment="Right">

                            <ComboBox             Name="ModemTagsComboBoxRemove"
ItemsSource="{Binding                                ModemTagsFilterListSelected,
UpdateSourceTrigger=PropertyChanged}">
                                <i:Interaction.Triggers>
                                    <i:EventTrigger
EventName="SelectionChanged">
                                        <i:InvokeCommandAction Command="{Binding
Deselect}"    CommandParameter="{Binding    ElementName=ModemTagsComboBoxRemove,
Path=SelectedItem}" />
```

```
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                        <!--<Button Margin="10 0 10 0" Command="{Binding
ClearFilters, UpdateSourceTrigger=PropertyChanged}" HorizontalAlignment="Right"
                                CommandParameter="{x:Static
local:TagType.Modem}">Очистить</Button>-->

                    </StackPanel>
                </DockPanel>
                <!--приборы устройства учета регуляторы-->
                <DockPanel Margin="0 10 0 10" Grid.Row="5"
HorizontalAlignment="Stretch">
                    <ComboBox Name="DevicesTagsComboBoxAdd"
HorizontalAlignment="Left" ItemsSource="{Binding DevicesTagsFilterList,
UpdateSourceTrigger=PropertyChanged}">
                        <i:Interaction.Triggers>
                            <i:EventTrigger EventName="SelectionChanged">
                                <i:InvokeCommandAction Command="{Binding
Select}" CommandParameter="{Binding ElementName=DevicesTagsComboBoxAdd,
Path=SelectedItem}" />
                            </i:EventTrigger>
                        </i:Interaction.Triggers>
                    </ComboBox>
                    <TextBlock FontSize="14" HorizontalAlignment="Center"
TextWrapping="Wrap" Margin="10 0 10 0">Теги приборов учета и
регуляторов</TextBlock>
                    <ComboBox Name="DevicesTagsComboBoxRemove"
HorizontalAlignment="Right" ItemsSource="{Binding DevicesTagsFilterListSelected,
UpdateSourceTrigger=PropertyChanged}">
                        <i:Interaction.Triggers>
                            <i:EventTrigger EventName="SelectionChanged">
                                <i:InvokeCommandAction Command="{Binding
Deselect}" CommandParameter="{Binding ElementName=DevicesTagsComboBoxRemove,
Path=SelectedItem}" />
                            </i:EventTrigger>
                        </i:Interaction.Triggers>
                    </ComboBox>
                </DockPanel>
                <!--tv теги-->
                <DockPanel Margin="0 10 0 10" Grid.Row="6"
HorizontalAlignment="Stretch">
                    <ComboBox Name="TVTagsComboBoxAdd"
HorizontalAlignment="Left" ItemsSource="{Binding TVTagsFilterList,
UpdateSourceTrigger=PropertyChanged}">
                        <i:Interaction.Triggers>
                            <i:EventTrigger EventName="SelectionChanged">
                                <i:InvokeCommandAction Command="{Binding
Select}" CommandParameter="{Binding ElementName=TVTagsComboBoxAdd,
Path=SelectedItem}" />
                            </i:EventTrigger>
                        </i:Interaction.Triggers>
                    </ComboBox>
                    <TextBlock FontSize="14" HorizontalAlignment="Right"
TextWrapping="Wrap" Margin="10 0 10 0">Теги точек учета и контуров</TextBlock>
                    <ComboBox Name="TVTagsComboBoxRemove"
HorizontalAlignment="Right" ItemsSource="{Binding TVTagsFilterListSelected,
UpdateSourceTrigger=PropertyChanged}">
                        <i:Interaction.Triggers>
                            <i:EventTrigger EventName="SelectionChanged">
                                <i:InvokeCommandAction Command="{Binding
Deselect}" CommandParameter="{Binding ElementName=TVTagsComboBoxRemove,
Path=SelectedItem}" />
                            </i:EventTrigger>
```

```
                        </i:Interaction.Triggers>
                    </ComboBox>
                </DockPanel>
                <!--владельцы-->
                <DockPanel     Margin="0     10     0     10"     Grid.Row="7"
HorizontalAlignment="Stretch">
                    <ComboBox                        Name="OwnersComboBoxAdd"
HorizontalAlignment="Left"        ItemsSource="{Binding        OwnersFilterList,
UpdateSourceTrigger=PropertyChanged}">
                        <i:Interaction.Triggers>
                            <i:EventTrigger EventName="SelectionChanged">
                                <i:InvokeCommandAction     Command="{Binding
Select}"        CommandParameter="{Binding        ElementName=OwnersComboBoxAdd,
Path=SelectedItem}" />
                            </i:EventTrigger>
                        </i:Interaction.Triggers>
                    </ComboBox>
                    <TextBlock   FontSize="14"   HorizontalAlignment="Right"
TextWrapping="Wrap" Margin="10 0 10 0">Владельцы</TextBlock>
                    <ComboBox                    Name="OwnersComboBoxRemove"
HorizontalAlignment="Right"    ItemsSource="{Binding    OwnersFilterListSelected,
UpdateSourceTrigger=PropertyChanged}">
                        <i:Interaction.Triggers>
                            <i:EventTrigger EventName="SelectionChanged">
                                <i:InvokeCommandAction     Command="{Binding
Deselect}"     CommandParameter="{Binding     ElementName=OwnersComboBoxRemove,
Path=SelectedItem}" />
                            </i:EventTrigger>
                        </i:Interaction.Triggers>
                    </ComboBox>
                </DockPanel>
                <!--источники-->
                <DockPanel     Margin="0     10     0     10"     Grid.Row="8"
HorizontalAlignment="Stretch">
                    <ComboBox                        Name="SourceComboBoxAdd"
HorizontalAlignment="Left"        ItemsSource="{Binding        SourcesFilterList,
UpdateSourceTrigger=PropertyChanged}">
                        <i:Interaction.Triggers>
                            <i:EventTrigger EventName="SelectionChanged">
                                <i:InvokeCommandAction     Command="{Binding
Select}"        CommandParameter="{Binding        ElementName=SourceComboBoxAdd,
Path=SelectedItem}" />
                            </i:EventTrigger>
                        </i:Interaction.Triggers>
                    </ComboBox>
                    <TextBlock   FontSize="14"   HorizontalAlignment="Right"
TextWrapping="Wrap" Margin="10 0 10 0">Источники</TextBlock>
                    <ComboBox                    Name="SourceComboBoxRemove"
HorizontalAlignment="Right"   ItemsSource="{Binding   SourcesFilterListSelected,
UpdateSourceTrigger=PropertyChanged}">
                        <i:Interaction.Triggers>
                            <i:EventTrigger EventName="SelectionChanged">
                                <i:InvokeCommandAction     Command="{Binding
Deselect}"     CommandParameter="{Binding     ElementName=SourceComboBoxRemove,
Path=SelectedItem}" />
                            </i:EventTrigger>
                        </i:Interaction.Triggers>
                    </ComboBox>
                </DockPanel>

                <!--муниципальные образования-->
                <DockPanel     Margin="0     10     0     10"     Grid.Row="9"
HorizontalAlignment="Stretch">
```

```
                        <ComboBox               Name="MunicipalitiesComboBoxAdd"
HorizontalAlignment="Left"    ItemsSource="{Binding    MunicipalitiesFilterList,
UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
Select}"    CommandParameter="{Binding    ElementName=MunicipalitiesComboBoxAdd,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                        <TextBlock   FontSize="14"   HorizontalAlignment="Right"
TextWrapping="Wrap" Margin="10 0 10 0">Муниципальные образования</TextBlock>
                        <ComboBox            Name="MunicipalitiesComboBoxRemove"
HorizontalAlignment="Right"                                ItemsSource="{Binding
MunicipalitiesFilterListSelected, UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
Deselect}"  CommandParameter="{Binding  ElementName=MunicipalitiesComboBoxRemove,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                    </DockPanel>

                    <!--инспекторы-->
                    <DockPanel    Margin="0    10    0    10"    Grid.Row="10"
HorizontalAlignment="Stretch">
                        <ComboBox                    Name="InspectorsComboBoxAdd"
HorizontalAlignment="Left"      ItemsSource="{Binding      InspectorsFilterList,
UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
Select}"      CommandParameter="{Binding      ElementName=InspectorsComboBoxAdd,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                        <TextBlock   FontSize="14"   HorizontalAlignment="Right"
TextWrapping="Wrap" Margin="10 0 10 0">Тепловые инспекторы</TextBlock>
                        <ComboBox                Name="InspectorsComboBoxRemove"
HorizontalAlignment="Right"  ItemsSource="{Binding  InspectorsFilterListSelected,
UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
Deselect}"    CommandParameter="{Binding    ElementName=InspectorsComboBoxRemove,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                    </DockPanel>

                    <!--участки тепловой инспекции-->
                    <DockPanel    Margin="0    10    0    10"    Grid.Row="11"
HorizontalAlignment="Stretch">
                        <ComboBox               Name="InspectionAreaComboBoxAdd"
HorizontalAlignment="Left"    ItemsSource="{Binding    InspectionAreaFilterList,
UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
```

```
                                    <i:InvokeCommandAction     Command="{Binding
Select}"    CommandParameter="{Binding    ElementName=InspectionAreaComboBoxAdd,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                        <TextBlock   FontSize="14"   HorizontalAlignment="Right"
TextWrapping="Wrap" Margin="10 0 10 0">Участки тепловой инспекции</TextBlock>
                        <ComboBox            Name="InspectionAreaComboBoxRemove"
HorizontalAlignment="Right"                                ItemsSource="{Binding
InspectionAreaFilterListSelected, UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
Deselect}"  CommandParameter="{Binding  ElementName=InspectionAreaComboBoxRemove,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                    </DockPanel>
                    <!--пользовательские события-->
                    <DockPanel    Margin="0    10    0    10"    Grid.Row="12"
HorizontalAlignment="Stretch">
                        <ComboBox                   Name="UserEventsComboBoxAdd"
HorizontalAlignment="Left"      ItemsSource="{Binding      UserEventsFilterList,
UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
Select}"      CommandParameter="{Binding      ElementName=UserEventsComboBoxAdd,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                        <TextBlock   FontSize="14"   HorizontalAlignment="Right"
TextWrapping="Wrap" Margin="10 0 10 0">Пользовательские события</TextBlock>
                        <ComboBox                Name="UserEventsComboBoxRemove"
HorizontalAlignment="Right"  ItemsSource="{Binding  UserEventsFilterListSelected,
UpdateSourceTrigger=PropertyChanged}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
Deselect}"    CommandParameter="{Binding    ElementName=UserEventsComboBoxRemove,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                    </DockPanel>

                    <!--режим работы-->
                    <StackPanel   Margin="0    10    0    10"    Background="White"
Grid.Row="15" Orientation="Vertical">
                        <TextBlock            HorizontalAlignment="Center">Режим
работы</TextBlock>
                        <ComboBox                   HorizontalAlignment="Center"
Name="OperatingModeComboBox" ItemsSource="{Binding OperationModesFilterList}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
ChangeMode}"    CommandParameter="{Binding    ElementName=OperatingModeComboBox,
Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
```

```
                    </StackPanel>
                    <!--события настроены-->
                    <StackPanel   Margin="0   10   0   10"   Background="White"
Grid.Row="14" Orientation="Vertical">
                        <TextBlock          HorizontalAlignment="Center">События
настроены</TextBlock>
                        <ComboBox                   HorizontalAlignment="Center"
Name="EventSettingsModeComboBox"                            ItemsSource="{Binding
EventsSettingFilterList}">
                            <i:Interaction.Triggers>
                                <i:EventTrigger EventName="SelectionChanged">
                                    <i:InvokeCommandAction     Command="{Binding
ChangeEventSetting}"                                  CommandParameter="{Binding
ElementName=EventSettingsModeComboBox, Path=SelectedItem}" />
                                </i:EventTrigger>
                            </i:Interaction.Triggers>
                        </ComboBox>
                    </StackPanel>
                </Grid>
            </ScrollViewer>

            <ScrollViewer     Grid.Row="1"     Name="ObjectsDataGridScrollViewer"
HorizontalScrollBarVisibility="Auto" VerticalScrollBarVisibility="Auto">

                <DataGrid HeadersVisibility="Column" x:Name="ObjectsDataGrid"
                      Background="#d0deec"  HorizontalContentAlignment="Stretch"
HorizontalAlignment="Stretch"
                      ScrollViewer.HorizontalScrollBarVisibility="Disabled"
                      AutoGenerateColumns="False"
                      ItemsSource="{Binding Objects}" Visibility="Visible"
                      IsReadOnly="True">
                    <DataGrid.Columns>
                        <DataGridTextColumn       Header="Название       объекта"
Binding="{Binding Path=Name}"/>
                        <DataGridTextColumn   Header="Адрес"   Binding="{Binding
Path=Address}"/>
                        <DataGridTextColumn         Header="Тип         объекта"
Binding="{Binding Path=TypeObject}"/>
                        <DataGridTextColumn                 Header="Потребитель"
Binding="{Binding Path=Consumer}"/>
                        <DataGridTextColumn  Header="Описание"  Binding="{Binding
Path=Description}"/>
                        <DataGridTextColumn  Header="События"  Binding="{Binding
Path=HasEvents}"/>
                        <DataGridTextColumn        Header="Дата        создания"
Binding="{Binding Path=CreatedOn}"/>
                        <DataGridTextColumn  Header="Долгота"  Binding="{Binding
Path=Longitude}"/>
                        <DataGridTextColumn   Header="Широта"   Binding="{Binding
Path=Latitude}"/>
                        <DataGridTextColumn         Header="Зимний         режим"
Binding="{Binding Path=IsWinterMode}"/>
                        <DataGridTextColumn       Header="Изменение       режима"
Binding="{Binding Path=IsModeChanged}"/>
                        <DataGridTextColumn               Header="Идентификатор"
Binding="{Binding Path=ID}"/>
                        <DataGridTextColumn    Header="Идентификатор    объекта"
Binding="{Binding Path=ObjectIdentifier}"/>
                        <DataGridTextColumn    Header="Имя    приборной    доски"
Binding="{Binding Path=DashboardName}"/>
                        <DataGridTextColumn    Header="Идентификатор    приборной
доски" Binding="{Binding Path=DashboardID}"/>
                        <DataGridTextColumn    Header="Теги"    Binding="{Binding
Path=TagsString, Mode=OneWay}"/>
```

```
                    </DataGrid.Columns>
                </DataGrid>
            </ScrollViewer>
            <ScrollViewer   Grid.Row="2"   HorizontalScrollBarVisibility="Auto"
VerticalScrollBarVisibility="Auto">
                <StackPanel                    x:Name="ObjectsControlStackPanel"
Orientation="Horizontal">
                    <Button Command="{Binding ChangePage}" CommandParameter="-1"
Margin="10" x:Name="ObjectsBackwardButton">&lt;</Button>
                    <StackPanel                    Name="PageButtonsStackPanel"
Orientation="Horizontal"></StackPanel>
                    <Button Command="{Binding ChangePage}" CommandParameter="1"
Margin="10" x:Name="ObjectsForward">&gt;</Button>
                    <TextBlock   FontSize="20"   Margin="10"   >Перейти   на
страницу:</TextBlock>
                    <TextBox          PreviewTextInput="NumberValidationTextBox"
Margin="5"   Name="ObjectsGoToPageTextBox"   Text="{Binding   Path=CurrentPage,
Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}">
                        <i:Interaction.Triggers>
                            <i:EventTrigger EventName="LostFocus">
                                <i:InvokeCommandAction         Command="{Binding
LoadObjects}"/>
                            </i:EventTrigger>
                        </i:Interaction.Triggers>
                        <TextBox.InputBindings>
                            <KeyBinding     Command="{Binding     LoadObjects}"
Key="Enter"></KeyBinding>
                        </TextBox.InputBindings>
                    </TextBox>
                    <TextBlock FontSize="20" Margin="10" >Строк:</TextBlock>
                    <TextBox          PreviewTextInput="NumberValidationTextBox"
Margin="5"   Name="ObjectsRowsAmountTextBox"   Text="{Binding   Path=RowsAmount,
Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}">
                        <i:Interaction.Triggers>
                            <i:EventTrigger EventName="LostFocus">
                                <i:InvokeCommandAction         Command="{Binding
LoadObjects}"/>
                            </i:EventTrigger>
                        </i:Interaction.Triggers>
                        <TextBox.InputBindings>
                            <KeyBinding     Command="{Binding     LoadObjects}"
Key="Enter"></KeyBinding>
                        </TextBox.InputBindings>
                    </TextBox>
                    <Button Command="{Binding LoadObjects}">
                        <Image Width="32" Height="32">
                            <Image.Source>
                                <FormatConvertedBitmap
Source="Resources/Images/glass.png"></FormatConvertedBitmap>
                            </Image.Source>
                        </Image>
                    </Button>
                </StackPanel>
            </ScrollViewer>
        </Grid>
    </Grid>
</Window>
```

Приложение 2

Скриншоты работы приложения

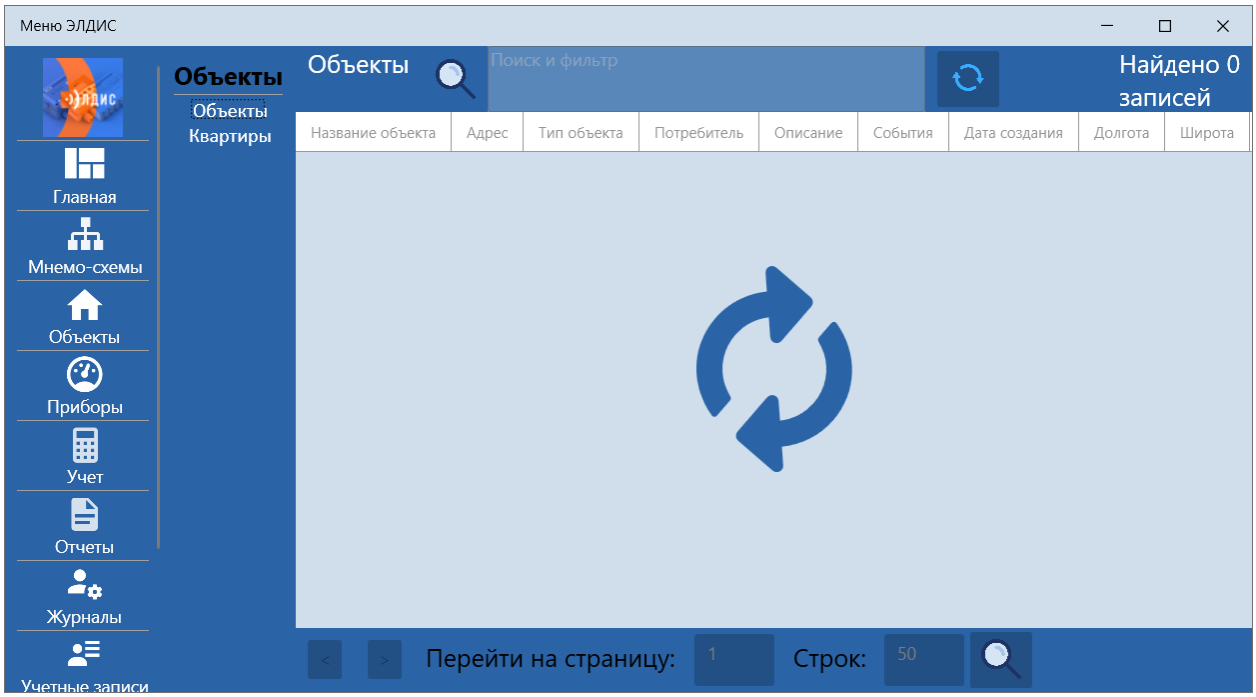

Рис. 1 Первоначальная загрузка данных приложения

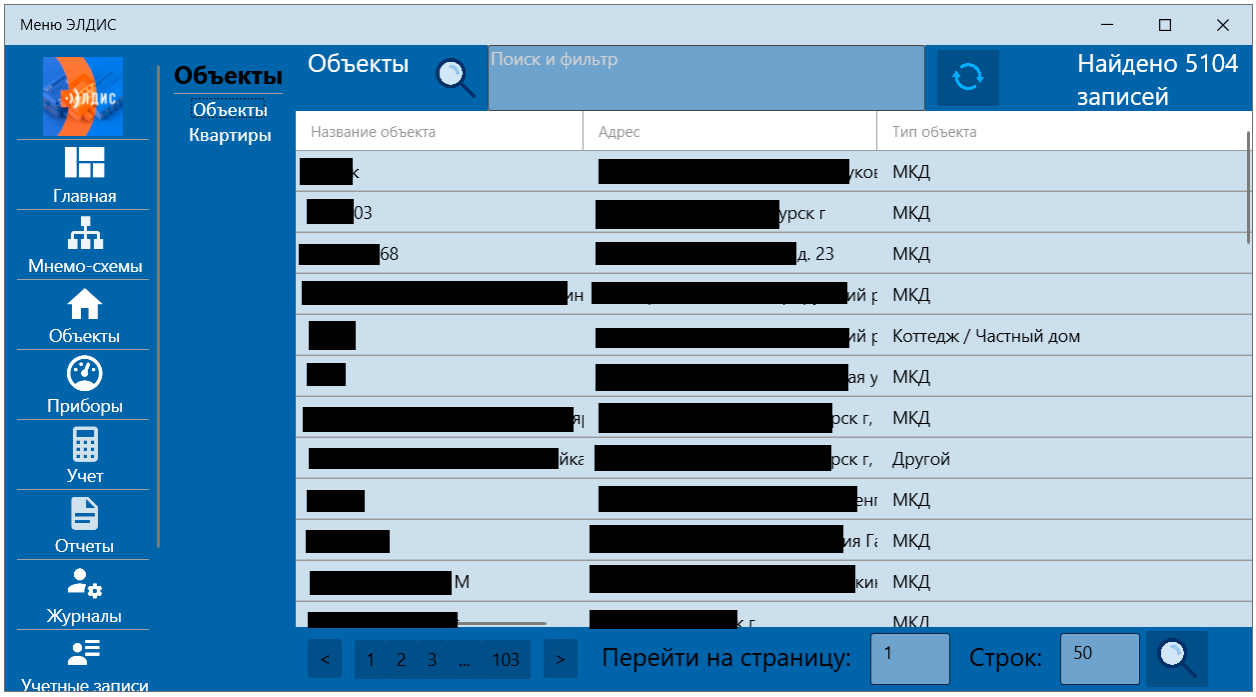

Рис. 2 Отображение данных по объектам

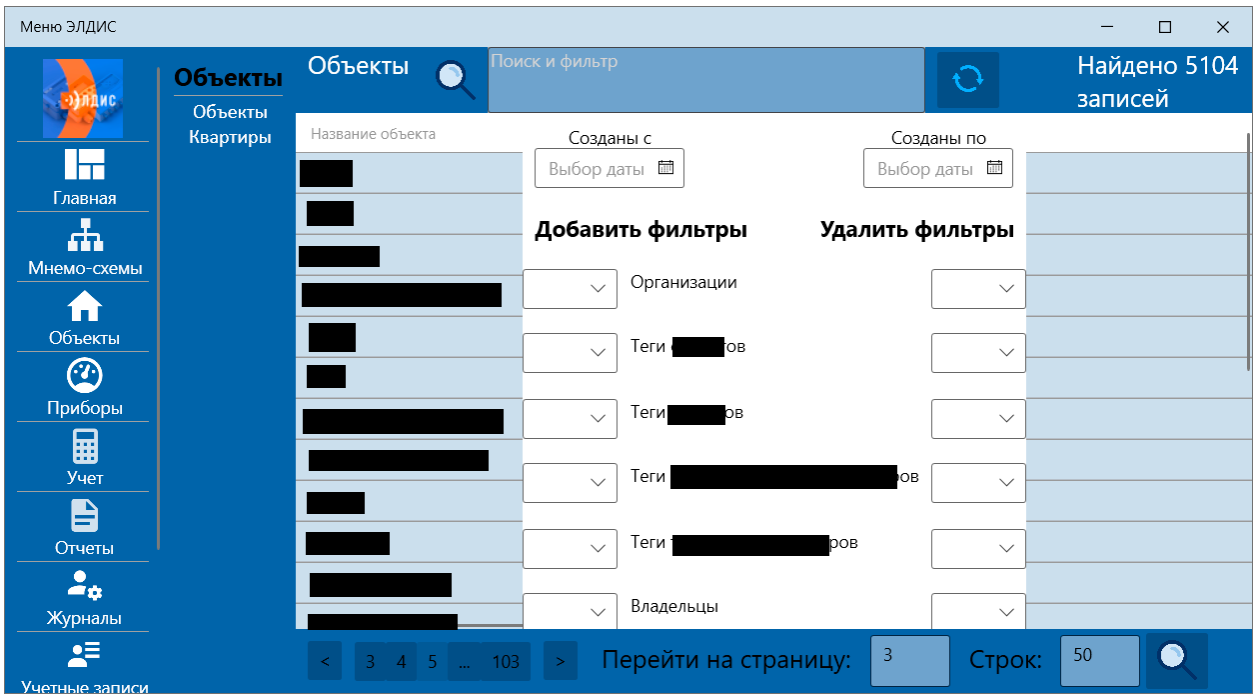

Рис. 3 Меню фильтров

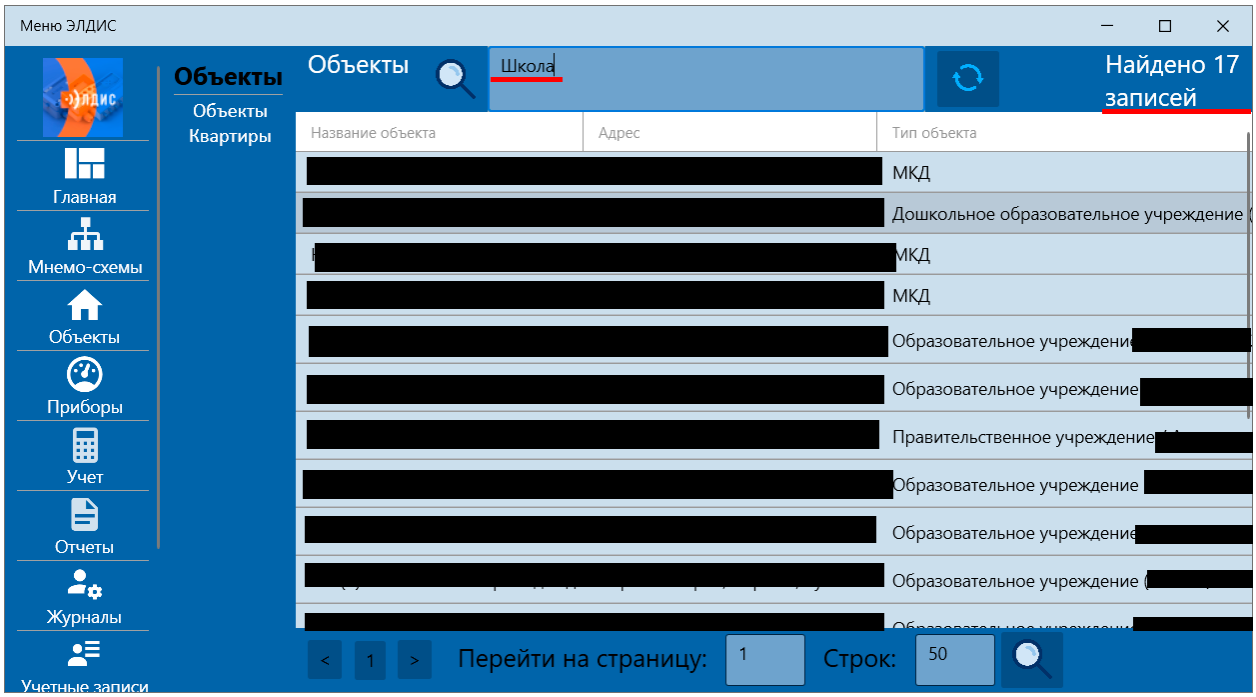

Рис. 4 Результат применения поисковой строки

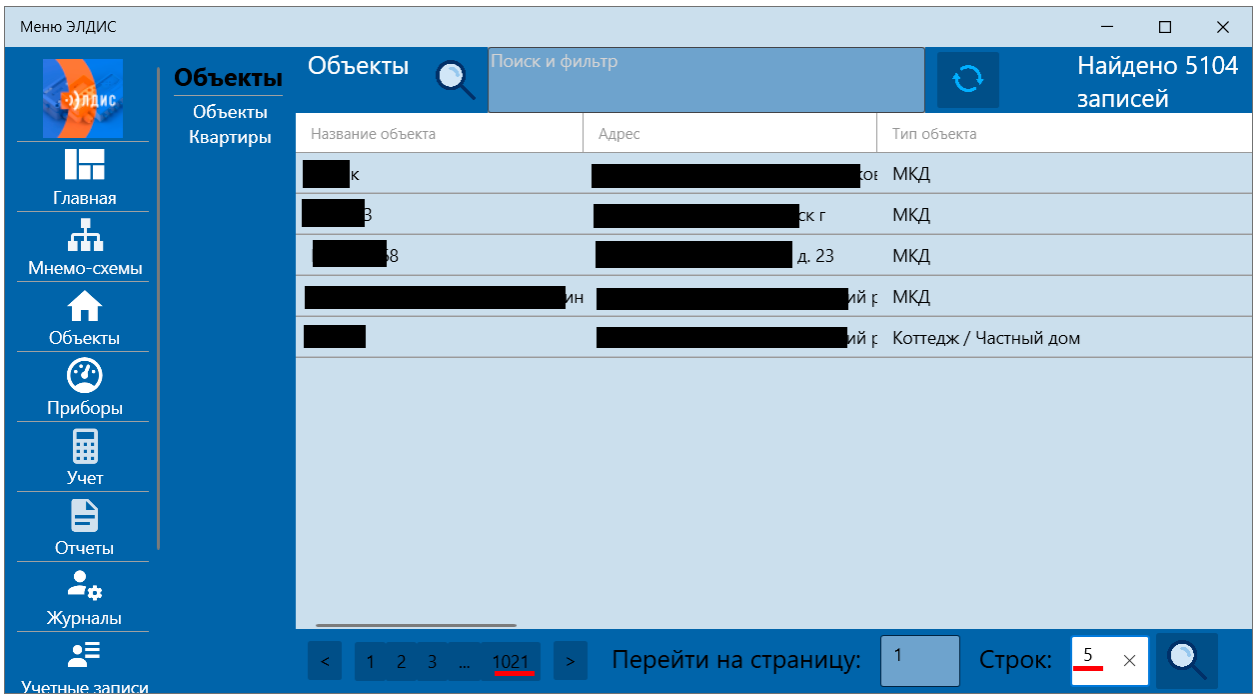

Рис. 5.1 Адаптивность интерфейса приложения в зависимости от выбора количества записей

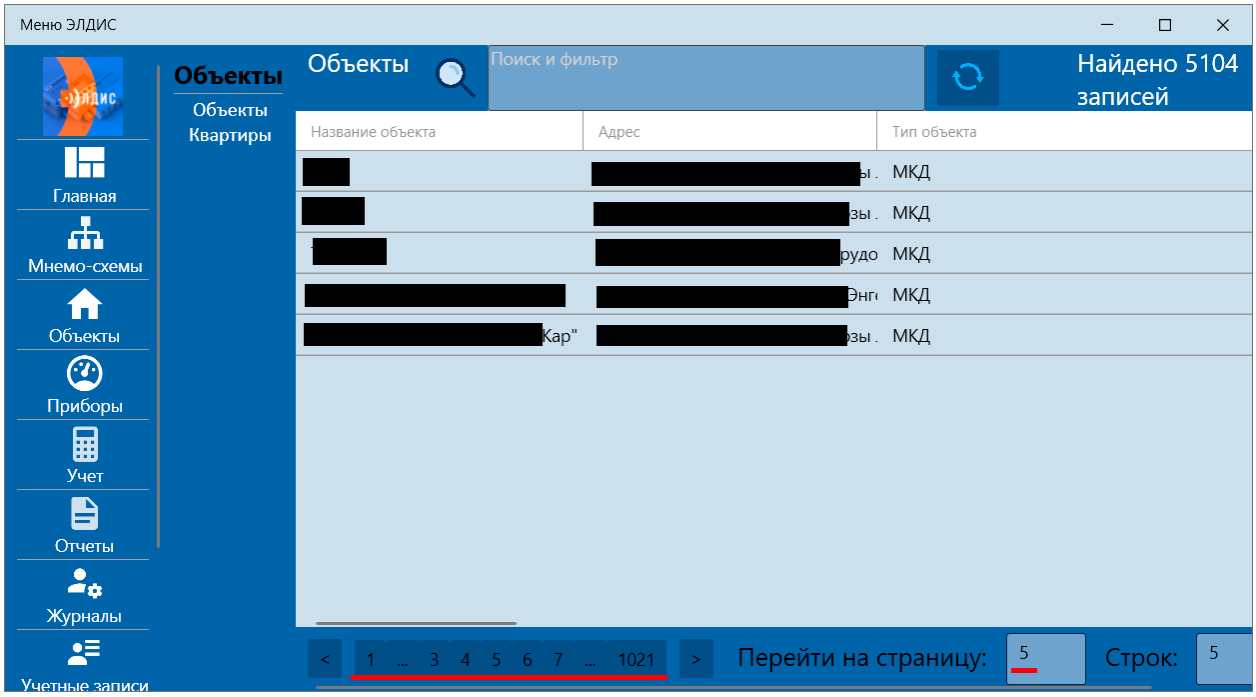

Рис. 5.2 Адаптивность интерфейса приложения в зависимости от постраничного перехода

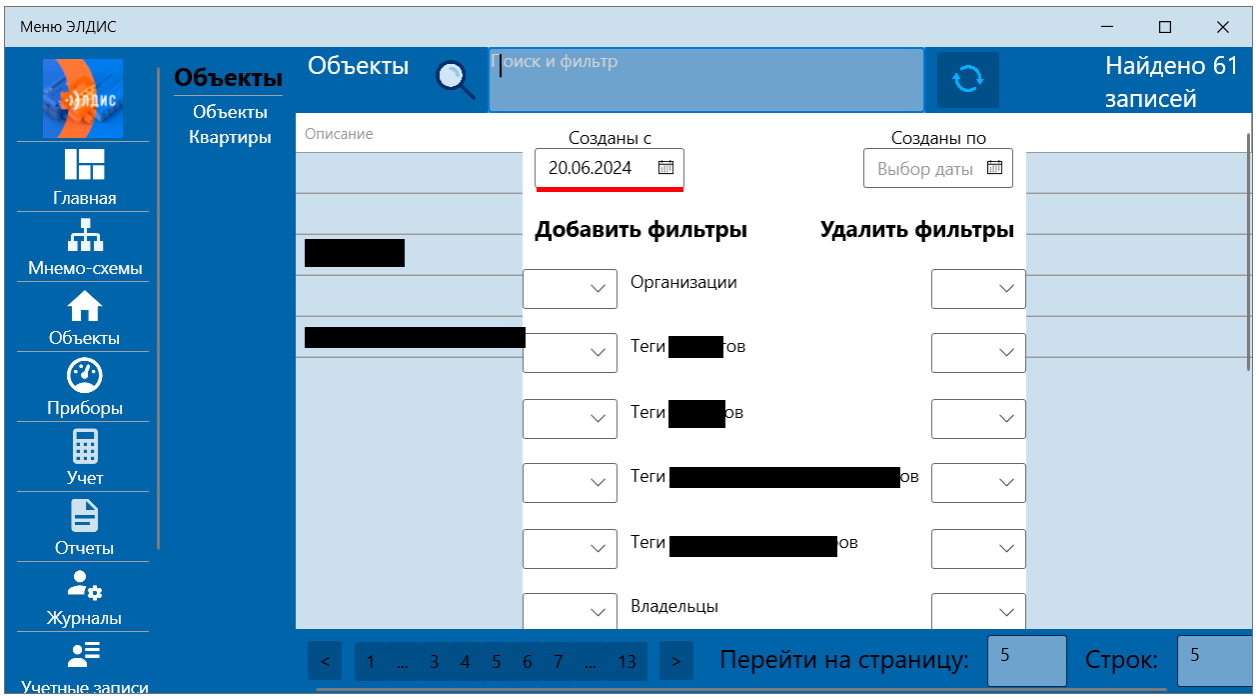

Рис. 6.1 Применение фильтров дат

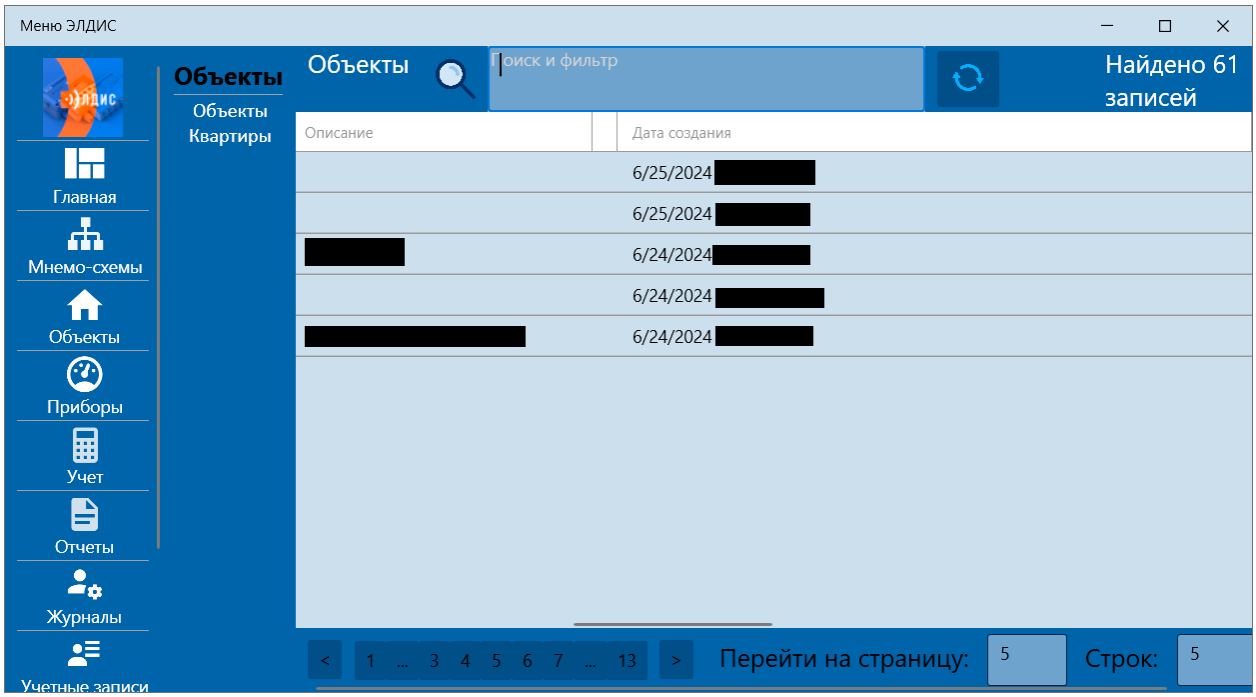

Рис. 6.2 Результат применения фильтров дат