МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тихоокеанский государственный университет»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

«Проектирование и разработка консольной игры “Dungeon”»

Текстовый документ курсовой работы

по дисциплине «Объектно-ориентированное программирование»

КР. 180008865.ТД

Выполнил студент Пшеничный Д.О.

Факультет, группа ФКФН, ПО(аб) – 81

Руководитель работы Федосеев А.А.

Виза: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(доработать, к защите и т.д.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Хабаровск – 2020г.

СОДЕРЖАНИЕ

[1Постановка задачи 3](#_Toc58582576)

[2 Диаграмма классов 4](#_Toc58582577)

[2.1 Описание классов 5](#_Toc58582578)

[3Диаграмма объектов 11](#_Toc58582579)

[4Диаграмма взаимодействия 12](#_Toc58582580)

[5Диаграмма состояний и переходов 13](#_Toc58582581)

[6Диаграмма модулей 14](#_Toc58582582)

[Заключение 15](#_Toc58582583)

[Список использованных источников 16](#_Toc58582584)

[Приложение А 17](#_Toc58582585)

# ПОСТАНОВКА ЗАДАЧИ

Требуется реализовать интерактивное приложение “Дашборд”, ­ обладающее следующими свойствами: Добавление и удаление различных видов виджетов, их обновление, сохранение состояния после закрытия. Для разработки приложения будет использован язык программирования C# в среде разработки Visual Studio 2019.

# ДИАГРАММА КЛАССОВ

На рисунке 1 рассматривается диаграмма классов.

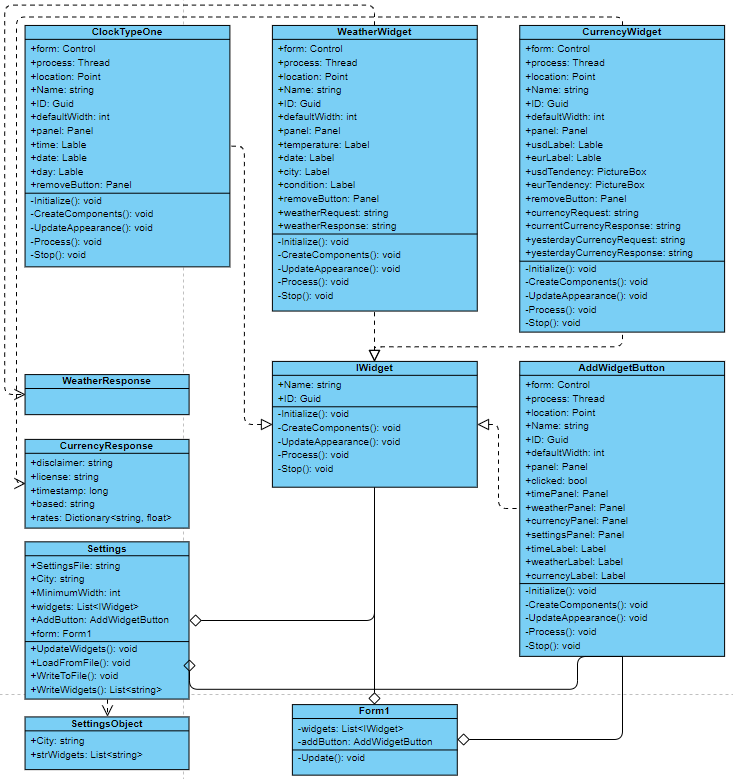
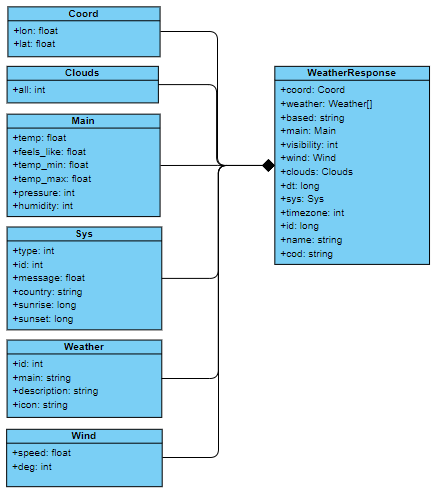
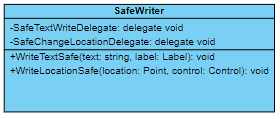
****

Рисунок 1 - Диаграмма классов





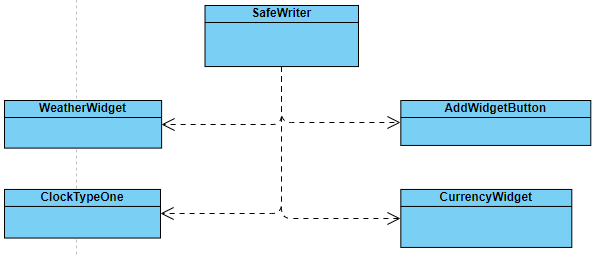


Рисунок 2 - Диаграмма классов продолжение

* 1. Описание классов

2.1.1 IWidget – интерфейс, описывающий общее поведение всех виджетов

Поля:

* string Name – имя виджета
* Guid ID – глобальный ID виджета

Методы:

* public void Initialize() – инициализирует основные компоненты виджета
* public void CreateComponents() – создаёт и инициализирует остальные необходимые для конкретного виджета визуальные компоненты
* public void UpdateAppearance() – отвечает за изменение визуального состояния виджета при изменении размеров окна
* public void Process() – отвечает за изменение контентной составляющей виджета, делает запросы к API. Запускается всегда в отдельном потоке
* public void Stop() – отвечает за остановку работы виджета, и параллельного потока

2.1.2 Описание классов ClockTypeOne, CurrencyWidget, WeatherWidget и AddWidgetButton

Данные классы являются наследниками IWidget, реализуют его поля и методы, и являются конкретными виджетами:

* ClockTypeOne – виджет часов и даты
* WeatherWidget – виджет погоды
* CurrencyWidget – виджет курса валют
* AddWidgetButton – содержит в себе кнопки добавления виджетов, а так же кнопку настроек

Данные классы реализуют метод Process() в зависимости от выполняемых задач.

2.1.3 CurrencyResponce – класс, предназначенный для парсинга JSON ответа от API бесплатного сервиса курса валют

Поля:

* public string disclaimer – описание состояния API
* public string license – описание лицензии использования
* public long timestamp – временная метка полученного курса
* string based – базовая валюта для расчета курса
* Dictionary<string, float> rates – словарь, хранящий курсы валют, с их наименованиями в качестве ключей

2.1.4 Settings – статический класс, предназначенный для хранения, сохранение в файл и загрузку из файла настроек

Поля:

* public string SettingsFile – хранит путь до файла настроек. Обычно это “settings.json” в папке программы
* public string City – город для которого нужно запрашивать информацию о погоде
* public int MinimumWidth – минимальная ширина окна программы
* public List<IWidget> widgets – список всех используемых в данный момент виджетов
* public AddWidgetButton addButton – кнопка добавления виджетов
* public Form1 form – хранит главную форму

Методы:

* public void UpdateWidgets() – принудительно обновляет визуальное состояние всех виджетов
* public void LoadFromFile() – выполняет загрузку настроек из файла
* public void WriteToFile() – выполняет запись настроек в файл
* public List<string> WriteWidgets() – создаёт на основе списка виджетов список их имён

2.1.5 SettingsObject – класс, используемый для парсинга настроек в JSON строку и обратно

Поля:

* public string City – город для которого нужно запрашивать информацию о погоде
* public List<string> strWidgets – список имён используемых виджетов

2.1.6 SafeWriter – статический класс, используемый для безопасного изменения параметров визуальных компонентов извне их потока

Поля:

* private delegate void SafeTextWriteDelegate – делегат, хранящий метод для изменения текста в Lable
* private delegate void SafeChangeLocationDelegate – делегат, хранящий метод для изменения позиции элемента управления

Методы:

* public void WriteTextSafe(string text, Label label) – безопасно изменяет текст в label на text
* public void WriteLocationSafe(Point location, Control control) – безопасно изменяет позицию control на location

2.1.7 Coord – класс, повторяющий структуру координат погодного API

Поля:

* public float lon – широта
* public float lat – долгота

2.1.8 Clouds – класс, повторяющий структуру информации об облаках погодного API

Поля:

* int all – информация о типе облаков

2.1.9 Main – класс, повторяющий структуру основной информации о погоде погодного API

Поля:

* float temp – текущая температура
* float feels\_like – текущая температура по ощущениям
* float temp\_min – минимальная температура за сегодня
* float temp\_max – максимальная температура за сегодня
* int pressure – текущее давление
* int humidity – текущая влажность

2.1.10 Sys – класс, повторяющий системную информацию погодного API

Поля:

* int type – внутренний параметр API
* int id – внутренний параметр API
* float message – внутренний параметр API
* string country – код страны
* long sunrise – время восхода солнца
* lung sunset – время захода солнца

2.1.11 Weather – класс, повторяющий структуру описания погодных условий погодного API

Поля:

* int id – уникальный идентификатор состояния погоды
* string main – краткое описание погодных условий
* string description – более подробное описание погодных условий
* string icon – название иконки состояния погоды

2.1.12 Wind – класс, повторяющий структуру информации о ветре погодного API

Поля:

* float speed – скорость ветра
* int deg – направление ветра

2.1.13 WeatherResponse – класс, повторяющий структуру JSON ответа погодного API, и предназначенный для парсинга ответа в удобную объектную структуру

Поля:

* Coord coord – координаты места, информация о погоде была запрошена
* Weather[] weather – информация о состоянии погоды
* string based – внутренний параметр API
* Main main – основная информация о погодных условиях
* int visibility – информация о видимости
* Wind wind – информация о ветре
* Clouds clouds – информация об облаках
* long dt – время, потраченное на обработку запроса
* Sys sys – внутренние параметры API
* int timezone – часовой пояс
* long id – уникальный идентификатор города
* string name – название города
* string cod внутренний параметр API

# ДИАГРАММА ОБЪЕКТОВ

На рисунке 3 изображен сценарий запуска приложения с инициализацией объектов.

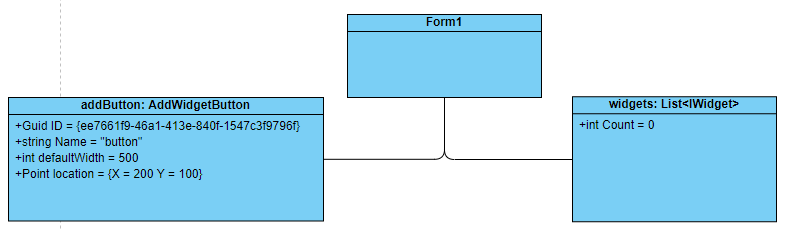


Рисунок 3 - Диаграмма объектов

# ДИАГРАММА ВЗАИМОДЕЙСТВИЯ

На диаграмме отображен сценарий работы приложения по сохранению настроек при закрытии.

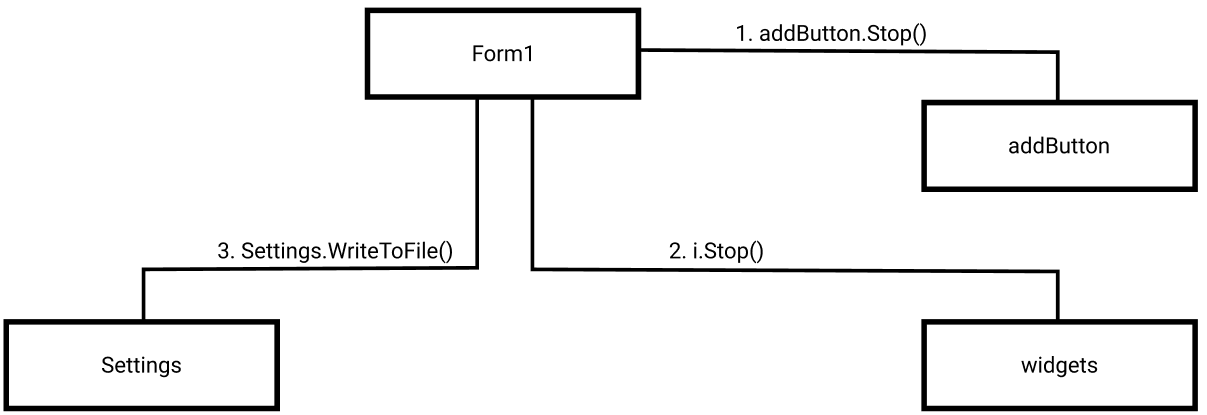


Рисунок 4 - Диаграмма коммуникаций

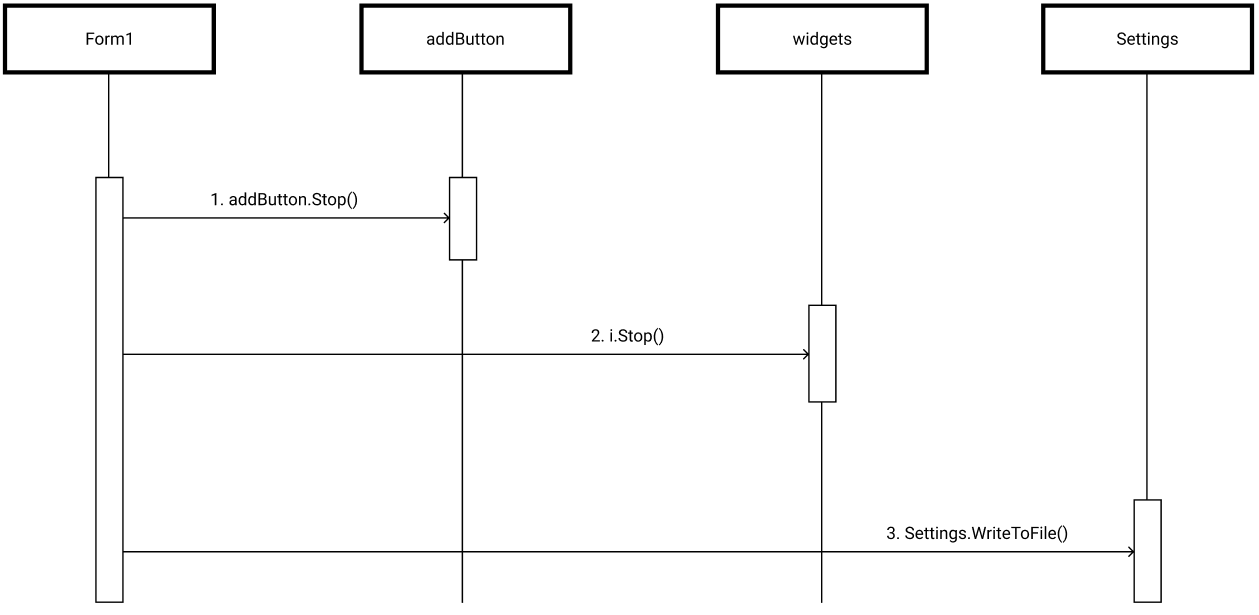
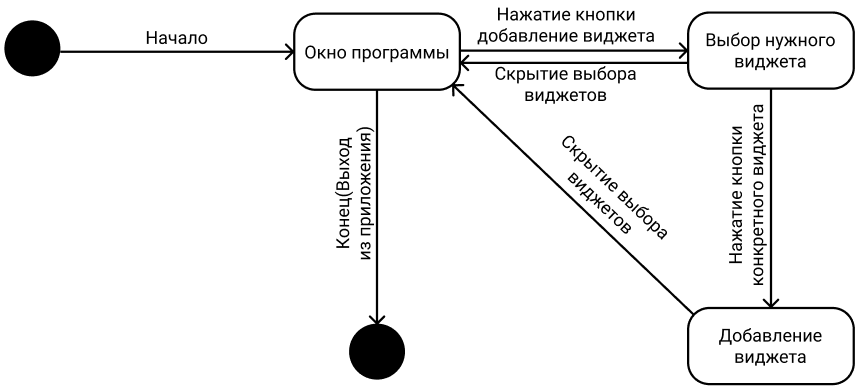


Рисунок 5 - Диаграмма последовательности

# ДИАГРАММА СОСТОЯНИЙ И ПЕРЕХОДОВ

На рисунке 6 изображена диаграмма состояний и переходов. Изображен сценарий добавления и удаления виджета.



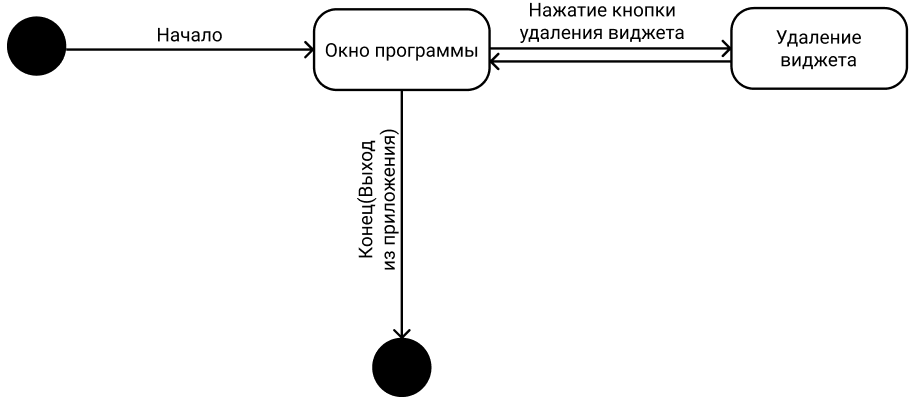


Рисунок 6 - Диаграмма состояний и переходов.

# ДИАГРАММА МОДУЛЕЙ

На рисунке 7 изображена диаграмма модулей. Соединение обозначает подключение пространства имён.

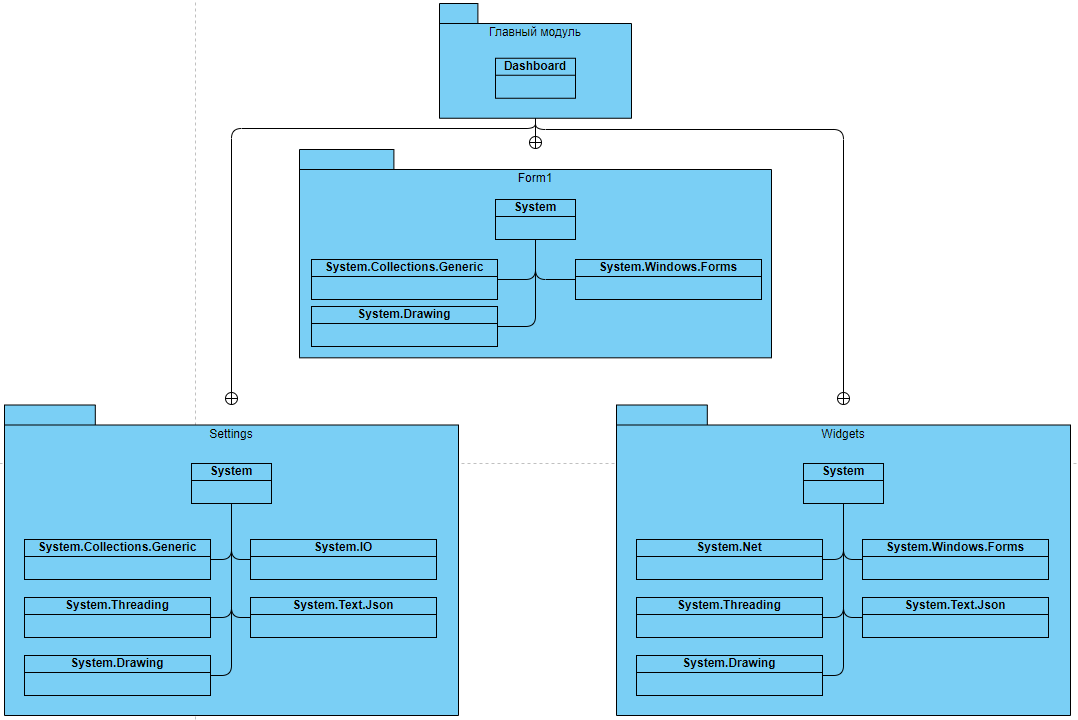


Рисунок 7 - Диаграмма модулей

# ЗАКЛЮЧЕНИЕ

Выполнение курсовой работы завершилось созданием программного продукта, который обладает функциональными возможностями, определенными заданием. Получены навыки использования принципов объектно ориентированного программирования для решения прикладных задач.

В приложении А приведены скриншоты, полученные в результате тестирования программы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Metanit [Электронный ресурс]: онлайн учебник. – Режим доступа: <https://metanit.com/> (дата обращения 12.25.2015)

2. MSDN [Электронный ресурс]: документация. [Microsoft]. – Режим доступа: <https://docs.microsoft.com/> (дата обращения 12.25.2015)

3. Бен Албахари и Джозеф Албахари. C# 7.0 in a Nutshell: The Definitive Reference, 28 октября 2017 г.

4. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.: Паттерны объектно-ориентированного проектирования — Прогресс книга, 2020 г. — 448 с.

# ПРИЛОЖЕНИЕ А

(обязательное)

Функционирование приложения

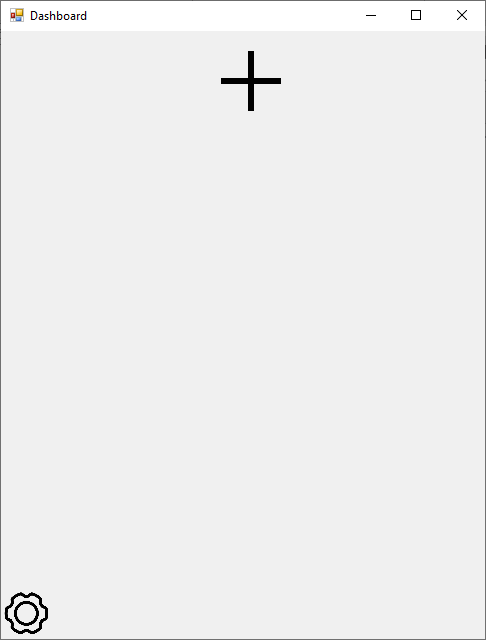


Рисунок А.1 – Основной вид приложения.

Продолжение приложения А

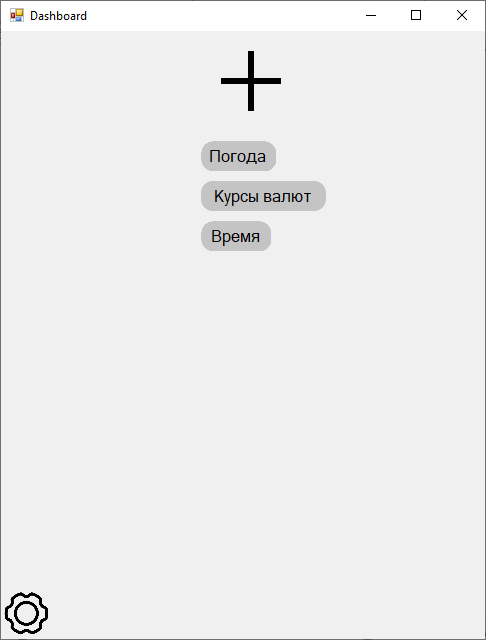


Рисунок А.2 – Добавление виджета.

Продолжение приложения А

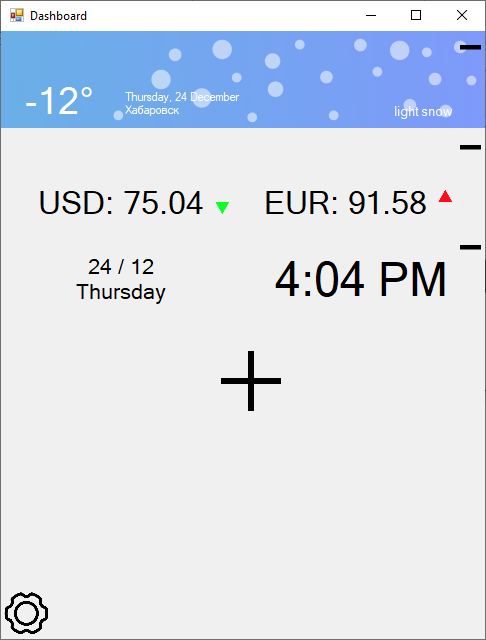


Рисунок А.3 – Демонстрация всех трёх видов виджетов.

Продолжение приложения А

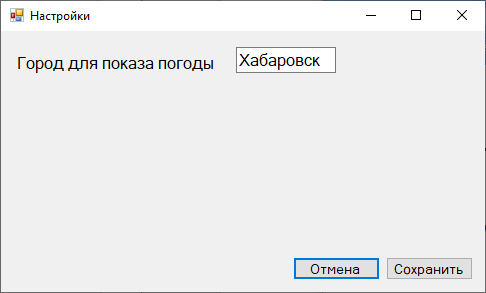


Рисунок А.4 – Окно настроек.