МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

Национальный аэрокосмический университет им. Н.Е. Жуковского  
 “Харьковский авиационный институт”

Факультет программной инженерии и бизнеса

Кафедра инженерии программного обеспечения

**Курсовой проект**

по дисциплине «Операционные системы»

(название дисциплины)

на тему: «Планировщик задач»

Выполнил: студент 3 курса группы №632п

направления подготовки (специальности)

6.050103 «Программная инженерия»\_\_\_\_\_\_\_

(шифр и название направления подготовки (специальности))

\_\_ Карачевцев А.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия и инициалы студента)

Руководитель: старший преподаватель,

Волошин О.Н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, научная степень, фамилия и инициалы)

Национальная шкала: \_\_\_\_\_\_\_\_\_\_

Количество баллов: \_\_\_\_\_

Оценка: ECTS \_\_\_\_\_

Члены комисии: \_\_\_\_\_\_\_\_\_ Волошин О. Н.

(подпись) (фамилия и инициалы)

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

(подпись) (фамилия и инициалы)

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

(подпись) (фамилия и инициалы)

Харьков – 2018

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc514547848)

[1 ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ ПО 4](#_Toc514547849)

[2 НАЗНАЧЕНИЕ РАЗРАБОТКИ И ОБЛАСТЬ ПРИМЕНЕНИЯ 4](#_Toc514547850)

[3 ЦЕЛИ ПРОЕКТА. ЦЕЛИ ПРОДУКТА 5](#_Toc514547851)

[3.1 Цели проекта 5](#_Toc514547852)

[3.2 Цели продукта 5](#_Toc514547853)

[3.3 Ориентировочная стоимость каждого этапа, каждой фазы 5](#_Toc514547854)

[3.4 Цели для каждого вида тестирования. 6](#_Toc514547855)

[3.5 Особенности сопровождения создаваемого программного обеспечения, которые необходимо учитывать при разработке 6](#_Toc514547856)

[3.6 Уровни надёжности 6](#_Toc514547857)

[4 АНАЛИЗ ОСУЩЕСТВИМОСТИ ПРОЕКТА 9](#_Toc514547858)

[4.1 Диаграмма вариантов использования 9](#_Toc514547859)

[4.2 Настройки программного продукта. 11](#_Toc514547860)

[4.3 Определение и анализ рисков 12](#_Toc514547861)

[4.3.3 Расчет уровня рисков и их влияния 13](#_Toc514547862)

[4.3.4 Общий итог 14](#_Toc514547863)

[5 ТРЕБОВАНИЯ ПОЛЬЗОВАТЕЛЯ 15](#_Toc514547864)

[5.1 Мандатные требования 15](#_Toc514547865)

[5.2 Ограничительные требования пользователя 15](#_Toc514547866)

[6 ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ 16](#_Toc514547867)

[6.1 Функциональные требования 16](#_Toc514547868)

[6.2. Нефункциональные требования 16](#_Toc514547869)

[7 МАТРИЦА ТРАССИРОВАНИЯ 17](#_Toc514547870)

[8 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 18](#_Toc514547871)

[8.1 Функциональная модель 18](#_Toc514547872)

[8.2 Модель данных ПО 19](#_Toc514547873)

[8.3 Диаграмма классов 20](#_Toc514547874)

[8.4 Описание алгоритмов 21](#_Toc514547875)

[8.5 Описание программной реализации 24](#_Toc514547876)

[9 ПРОЕКТИРОВАНИЕ И ПРОВЕДЕНИЕ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 26](#_Toc514547877)

[9.1 Системное тестирование 26](#_Toc514547878)

[9.2 Интеграционное тестирование 28](#_Toc514547879)

[9.3 Автономное тестирование 30](#_Toc514547880)

[ВЫВОД 32](#_Toc514547881)

[ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ 33](#_Toc514547882)

[ГЛОССАРИЙ 34](#_Toc514547883)

[ПРИЛОЖЕНИЕ А. ЭКРАННЫЕ ФОРМЫ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ 35](#_Toc514547884)

[ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ИСХОДНОГО КОДА 37](#_Toc514547885)

[ПРИЛОЖЕНИЕ В. ПРЕЗЕНТАЦИЯ 40](#_Toc514547886)

# **ВВЕДЕНИЕ**

**Планировщик задач** — [программа](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), часто называемая сервисом [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), которая запускает другие программы в зависимости от различных критериев, как, например:

* наступление определённого времени;
* переход операционной системы в определённое состояние (бездействие, спящий режим и т. д.);
* поступление запроса от администратора через пользовательский интерфейс или через [инструменты удалённого администрирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B_%D1%83%D0%B4%D0%B0%D0%BB%D1%91%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B0%D0%B4%D0%BC%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F).

Данный вид программ не занимает большое количество оперативной памяти, не влияет на быстроту и работоспособность системы. Более того, работа программы незаметна, пока не наступит время для выполнения задачи. Свернутая программа может быть помещена в системный лоток.

# **ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ ПО**

Документ, на основании которого ведется разработка:

Волошин О.Н., Постернакова В.А., Пудовкина Л.Ф. ОПЕРАЦИОННЫЕ СИСТЕМЫ. Методические рекомендации по выполнению курсового проекта. - Харьков: Национальный аэрокосмический университет им. Н. Е. Жуковского «Харьковский авиационный институт»,2014. - 43с.

# **НАЗНАЧЕНИЕ РАЗРАБОТКИ И ОБЛАСТЬ ПРИМЕНЕНИЯ**

Очень часто возникает необходимость запуска программ в заданное время. Например, утилиты проверки диска, антивирусное ПО, утилиты для резервного копирования и т.д. Специальная программа, под названием «Планировщик задач» поможет пользователю любого ПК организовать свое время при работе. Данное программное обеспечение обладает такими возможностями, как:

* добавление, удаление программ из внутреннего списка;
* возможность задания параметров командной строки для программы;
* указание для каждой программы даты и времени запуска в случае одноразового запуска;
* для повторяющегося запуска необходимо указать периодичность запуска (каждый день в указанное время, каждую неделю в указанный день и указанное время, каждое указанное число месяца и т.д.);
* в указанное время планировщик должен запустить программу

# **ЦЕЛИ ПРОЕКТА. ЦЕЛИ ПРОДУКТА**

## **Цели проекта**

* Разработать ПО «Планировщик задач».
* Уложиться в календарный план.
* Организация документации при работе над проектом.
* Провести тестирование ПО.

## **Цели продукта**

В указанное пользователем время запускать программу или команду в командной строке

## **Ориентировочная стоимость каждого этапа, каждой фазы**

Данный показатель может быть абсолютный или относительный.

Рассчитаем стоимость программного продукта «Планировщик задач». Учитывая, что один час оплаты работы составляет 10$, умножим на количество затраченных часов и дней получим стоимость продукта.

Оплата работы за этап "Анализ требований" составляет 4\*22\*10=880$

Оплата работы за этап "Проектирование" составляет 6\*66\*10=3960$

Оплата работы за этап "Тестирование" составляет 6\*7\*10=420$

Затраты на оплату работы за этапы проекту составят 880+3960+420=5260$

Ориентировочная стоимость проекта 5260\*2=10520$, где коэффициент 2 включает затраты на предотвращение и борьбу с рисками и сопутствующие расходы, а также составляет прибыль за проект.

## **Цели для каждого вида тестирования.**

* + 1. Цель автономного тестирования:
* Контроль каждого отдельного программного модуля.
* Сравнивание функций, реализуемые модулем, со спецификациями его функций или интерфейса.
  + 1. Цель интеграционного тестирования:
* Проверка информационных связей модулей
* Нахождение ошибок в интерфейсе между модулями
* Нахождение ошибок в интерфейсах между функциями и модулями
  + 1. Цель проведения системного тестирования клиентской части:
* Тестирование интерфейса «пользователь – ПК»;
* Проверка работы всех режимов и функций.

## **Особенности сопровождения создаваемого программного обеспечения, которые необходимо учитывать при разработке**

Данный программный продукт разрабатывается для операционной системы Windows 8. В будущем предусматривается разработка новых версий для операционной системы Windows 8.1 и 10.

## **Уровни надёжности**

Все уровни надежности должны быть достигнуты на каждом этапе жизненного цикла программного продукта для достижения общей надёжности.

Надежность является внутренним свойством системы, заложенным при ее создании и проявляющимся во времени при функционировании и эксплуатации.

Для данного проекта выбрана модель "модернизация" (рис.3.1). Блок "создание" соответствует фазам "требование заказчика", "требование к ПО", "архитектурное проектирование", "детальное проектирование" и "тестирование".

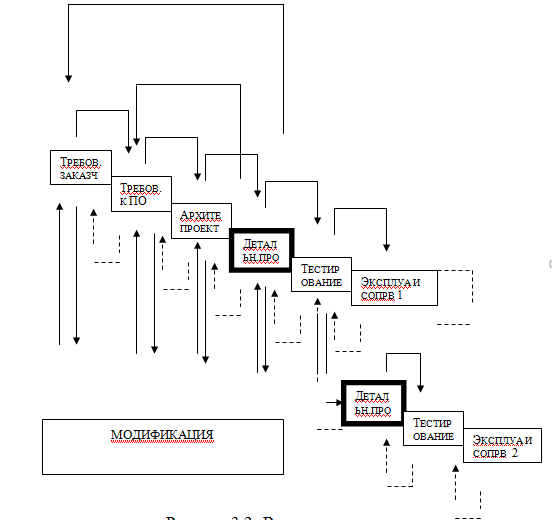


Рисунок 3.1 – Схема модели «Модернизация»

Вариант «модернизация», изображенный на рисунке 3.1, характеризуется плановым осуществлением многократных выпусков ПО. Чтобы получить выпуск системы, необходимо выполнить все фазы жизненного цикла. Каждый выпуск, включает опыт предыдущих выпусков системы. «Модернизация» может быть использована, например, в следующих случаях:

* опыт использования требует уточнить и дополнить требования после фазы «опытная эксплуатация» (показано пунктирной линией);
* некоторые части реализации могут зависеть от применения технологии будущего;
* ожидаются некоторые новые, но еще не определенные требования заказчика/пользователя;
* некоторые требования могут быть значительно более трудными для выполнения, чем другие, и решено не выполнять их, чтобы не задерживать приемлемую поставку.

Пунктирные расширения прямоугольников на рисунке 3.2 показывают, что совмещение фаз будет происходить до момента окончательной приемки каждой новой поставки.

Для варианта «модернизация» учитываются приоритеты заказчика/пользователя в создании проекта с минимальными техническими трудностями или задержками, но с важными частями ПО.

# **АНАЛИЗ ОСУЩЕСТВИМОСТИ ПРОЕКТА**

## **Диаграмма вариантов использования**

Действующие лица:

* пользователь;
* система.

Варианты использования:

* добавление программы в список;
* удаление программы из списка;
* задание параметров командной строки;
* указание времени запуска;
* указание периодичности запуска;
* просмотр информации о задаче.

На рисунках 4.1 – 4.2 изображен календарный план проекта

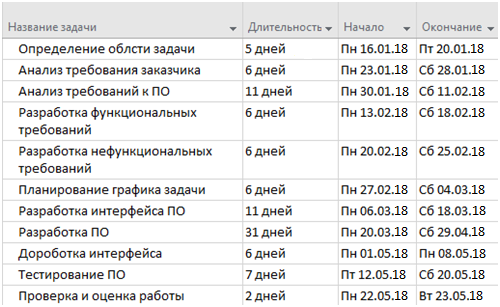


Рисунок 4.1 – План проекта

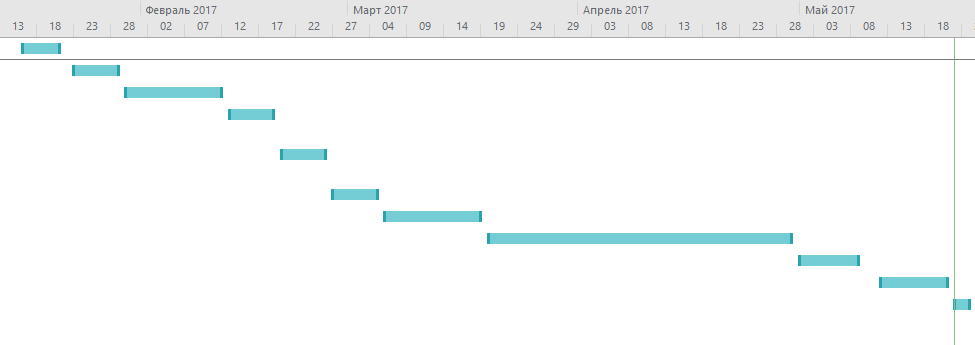


Рисунок 4.2 – Диаграмма Ганта

На рисунке 4.3 изображена USE-CASE диаграмма использования программного продукта

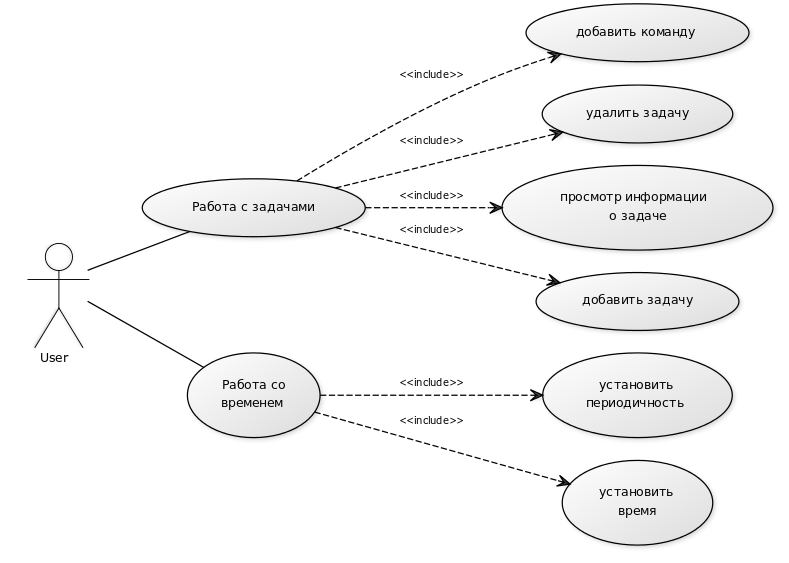


Рисунок 4.3 – USE-CASE диаграмма

## **Настройки программного продукта**.

Сценарий данного требования изображен в таблице 4.1.

Таблица 4.1 – Сценарий настройки планировщика задач

|  |  |
| --- | --- |
| Имя | Settings |
| Название | Изменение настроек планировщика задач |
| Участник | Пользователь |
| Описание | Выбор задачи и установка времени запуска |
| Предусловие | Программа запущена |
| Постусловие | Новая запланированная задача |
| Основной поток | 1. Выбрать задачу:    * программа;    * командная строка. 2. Настройка времени  * выбрать дату; * выбрать время; * выбрать время. * Выбрать периодичность |

## **Определение и анализ рисков**

* + 1. **Виды рисков**
       1. Финансовый риск
* Маленький спрос на программу;
* Нечастое использование.
  + - 1. Технический риск
* Сбой во время работы программы.
  + - 1. Организационные риски
* Ошибки во время планировки задачи
  + - 1. Риск конкуренции
* Имеются другие программы, которые могут иметь лучший интерфейс, лучшее оформление и дизайн;
* Более обновленное программное обеспечение.
  + - 1. Риски спроса
* Понижение интереса к данной программе из-за отсутствия более новых обновлений.
  + 1. **Риски предприятия**

Таблица рисков проекта представлена в таблице 4.1.

Таблица 4.1 – Таблица рисков

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование  риска | Тип | Вероятн-  ость, % | Прио-  ритет | Ущерб,  в грн. | Временные  потери,  в днях |
| Увеличение количества пользователей | PS | 55 | 3 | 1000 | 5 |
| Изменение выходных данных | RE | 60 | 2 | 10000 | 4 |
| Неподготовленный конечный пользователь | PR | 10 | 4 | 2000 | Без затрат |
| Противодействие конечных пользователей внедрению системы | PR | 10 | 1 | 11000 | 10 |

### **Расчет уровня рисков и их влияния**

Итоговая таблица расчет уровня рисков и их влияния по приоритетам представлен в таблице 4.2.

Таблица 4.2 – Итоговая таблица по приоритетам

|  |  |  |  |
| --- | --- | --- | --- |
| Приоритет | Вероятность, % | Средний ущерб, в грн. | Потеря времени, в днях |
| 1 | 0 | 0 | 0 |
| 2 | 35 | 10500 | 3,4 |
| 3 | 55 | 1000 | 1 |
| 4 | 10 | 2000 | 0 |

Итоговая таблица расчет уровня рисков и их влияния по критериям представлен в таблице 4.3.

Таблица 4.3 – Итоговая таблица по критериям

|  |  |  |  |
| --- | --- | --- | --- |
| Тип | Вероятность | Средний ущерб, в грн. | Потеря времени, в днях |
| PR | 10 | 6500 | 1 |
| RE | 60 | 10000 | 2.4 |
| PS | 55 | 1000 | 1.2 |

### **Общий итог**

Общий итог расчета уровня рисков и их влияния представлен в таблице 4.4.

Таблица 4.4 – Итоговая таблица уровня рисков

|  |  |
| --- | --- |
| Ущерб | Потеря времени |
| 17500 | 4.6 |

# **ТРЕБОВАНИЯ ПОЛЬЗОВАТЕЛЯ**

## **Мандатные требования**

* + 1. Функции системы.
       1. Выполнение задач в определенное время.
       2. Выполнение задач разного типа.
       3. Отображение информации о задаче.
       4. Выполнение задач с заданной периодичностью.
    2. Функции пользователя.
       1. Просмотр списка задач.
       2. Добавление новой задачи.
          1. Установка времени выполнения задачи
          2. Установка периодичности выполнения задачи
       3. Удаление существующей задачи
       4. Просмотр информации о задаче

## **Ограничительные требования пользователя**

* + 1. Язык интерфейса – русский
    2. Уровень пользователя – базовые знания работы с компьютером
    3. Работа ПО на операционных системах Windows XP/7/8/10
    4. Отсутствие обновлений ПО
    5. Техническая поддержка в течение первых 6 месяцев
    6. Минималистичный интерфейс

# **ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ**

## **Функциональные требования**

6.1.1. Выполнение задач в указанное время

6.1.2. Выполнение задач разного типа

6.1.3. Отображение информации о задаче

6.1.4. Функции пользователя

* + - 1. Просмотр списка задач
      2. Добавление новой задачи
         1. Установка времени выполнения задачи
         2. Установка периодичности выполнения задачи
      3. Удаление существующей задачи
      4. Просмотр информации о задаче

## **Нефункциональные требования**

* + 1. Язык интерфейса - русский.
    2. Уровень пользователя – базовые знания работы с компьютером
    3. ОС Windows XP 7/8 / 8.1 / 10.
    4. Периферийные устройства: монитор, мышка, клавиатура.
    5. Минимальный объем оперативной памяти 64Мб.
    6. Документация
       1. План управления проектов
       2. Спецификация требований к ПО.
       3. Проектная документация к ПО.
       4. Документация по тестированию.
    7. Понятный интерфейс.
    8. Техническая поддержка на протяжении полугода.
    9. Обновления ПО не требуются.

# **МАТРИЦА ТРАССИРОВАНИЯ**

Матрица трассирования представлена в таблице 7.1

Таблица 7.1 – Матрица трассирования

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 6.1.1 | 6.1.2 | 6.1.3 | 6.1.4.1 | 6.1.4.2 | 6.1.4.2.1 | 6.1.4.2.2 | 6.1.4.3 | 6.1.4.4 | 6.2.1 | 6.2.2 | 6.2.3 | 6.2.4 | 6.2.5 | 6.2.6 | 6.2.7 | 6.2.8 | 6.2.9 |
| 5.1.1.1 | + |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5.1.1.2 |  | + |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5.1.1.3 |  |  | + |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5.1.1.4 |  |  |  |  |  |  |  |  |  |  |  | + | + |  |  |  |  |  |
| 5.1.2.1 |  |  |  | + |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5.1.2.2 |  |  |  |  | + |  | + |  |  |  |  |  |  |  |  |  |  |  |
| 5.1.2.2.1 |  |  |  |  |  |  |  |  | + |  |  |  |  |  |  |  |  |  |
| 5.1.2.2.2 |  |  |  |  |  | + |  |  |  |  |  |  |  |  |  |  |  |  |
| 5.1.2.3 |  |  |  |  |  |  |  | + |  | + |  |  |  |  |  |  |  |  |
| 5.1.2.4 |  |  |  |  |  |  |  |  |  |  | + |  |  |  |  |  |  |  |
| 5.2.1 |  |  |  |  |  |  |  |  |  |  |  |  |  | + |  |  |  |  |
| 5.2.2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5.2.3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5.2.4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + |
| 5.2.5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + |  | + |  |
| 5.2.6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + |  |  |

# **ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

## **Функциональная модель**

Функциональная модель ПО «Планировщик задач» приведена на рисунках 8.1 – 8.2

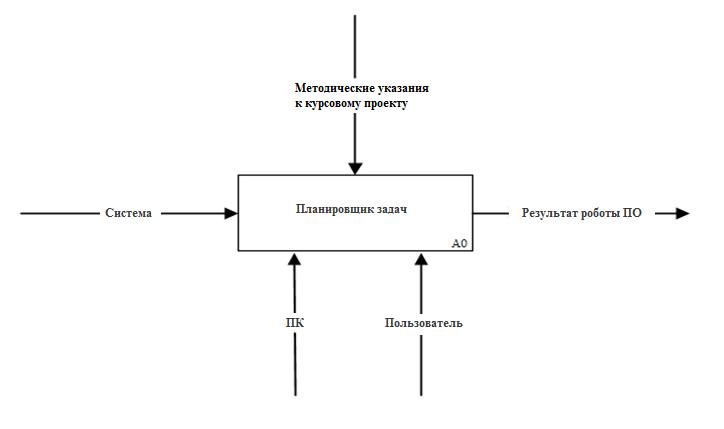


Рисунок 8.1 – Контекстная диаграмма (уровень 1)

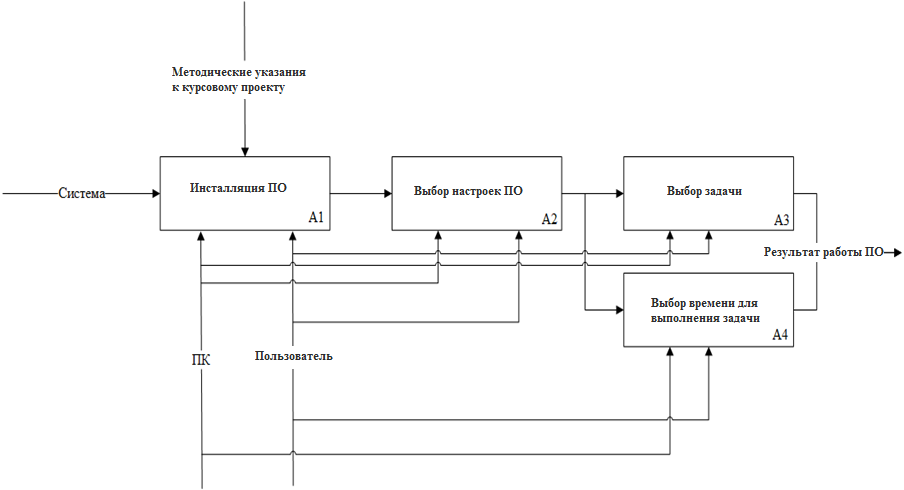


Рисунок 8.2 – Контекстная диаграмма (уровень 2)

## **Модель данных ПО**

Структурная схема приложения «Планировщик задач» изображена на рисунке 8.3

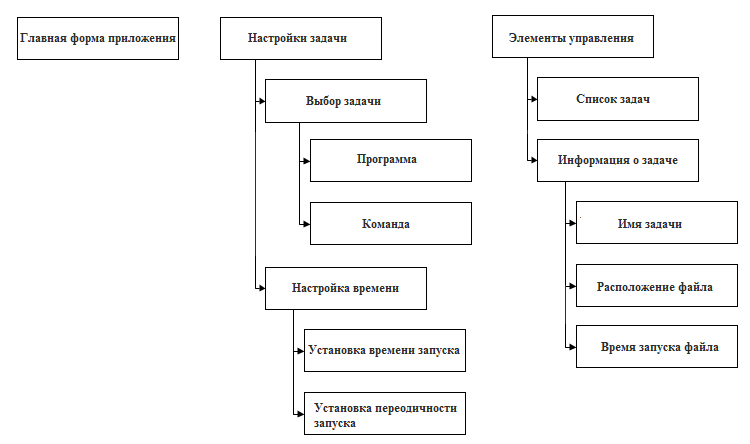


Рисунок 8.3 – Структурная схема интерфейса ПО

## **Диаграмма классов**

После проектирования функциональной модели и модели данных реализован основной класс и методы программного продукта. Диаграмма классов представлена на рисунке 8.4

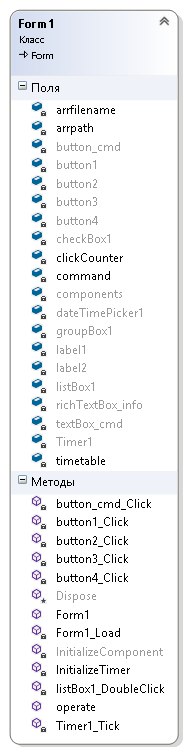


Рисунок 8.4 – Диаграмма классов проекта

## **Описание алгоритмов**

Исходя из постановки задачи и спроектированных методов, были разработаны алгоритмы этих методов. Алгоритмы представлены в виде псевдокода

* + 1. Класс Form1
       1. Метод operate – добавление задачи в очередь

public void operate()

**НАЧАЛО**

OpenFileDialog choofdlog = new OpenFileDialog();

choofdlog.Filter = "All Files (\*.\*)|\*.\*";

choofdlog.FilterIndex = 1;

choofdlog.Multiselect = true;

**ЕСЛИ** (choofdlog.ShowDialog() == DialogResult.OK) **ТО**

arrpath[clickCounter] = choofdlog.FileName; //путь

arrfilename[clickCounter] = choofdlog.SafeFileName; //имя

listBox1.Items.Insert(clickCounter, arrfilename[clickCounter]);

clickCounter++;

**КЕСЛИ**

**КОНЕЦ**

* + - 1. Метод InitializeTimer() – инициализация таймера

private void InitializeTimer()

**НАЧАЛО**

Timer1.Interval = 1000;

Timer1.Tick += new EventHandler(Timer1\_Tick);

Timer1.Enabled = true;

**КОНЕЦ**

* + - 1. Метод Timer1\_Tick – определяет операции, которые проводятся при тике таймера

private void Timer1\_Tick(object Sender, EventArgs e)

**НАЧАЛО**

**ЦИКЛ**(int i = 0; i < listBox1.Items.Count; i++)

**ЕСЛИ** (DateTime.Now.ToString() == timetable[i]) **ТО**

System.Diagnostics.Process.Start(arrpath[i], "/c " + command);

**КЕСЛИ**

**КЦИКЛ**

**КОНЕЦ**

* + - 1. Метод listBox1\_DoubleClick – определяет действие по двойному нажатию на элемент listBox

private void listBox1\_DoubleClick(object sender, MouseEventArgs e)

**НАЧАЛО**

**TRY**

richTextBox\_info.Clear();

richTextBox\_info.AppendText("Имя файла: " + arrfilename[listBox1.SelectedIndex] + "\n");

richTextBox\_info.AppendText("Путь файла: " + arrpath[listBox1.SelectedIndex] + "\n");

richTextBox\_info.AppendText("Время запуска: " + timetable[listBox1.SelectedIndex] );

**TRY**

**CATCH** (Exception)

**CATCH**

**КОНЕЦ**

* + - 1. Метод button\_cmd\_Click – определяет операции, которые выполняются при нажатии на кнопку button\_cmd

private void button\_cmd\_Click(object sender, EventArgs e)

**НАЧАЛО**

command = textBox\_cmd.Text;

arrpath[clickCounter] = "cmd";//путь

arrfilename[clickCounter] = "CMD.exe"; //имя

listBox1.Items.Insert(clickCounter, arrfilename[clickCounter]);

InitializeTimer();

clickCounter++;

**КОНЕЦ**

* + - 1. Метод button\_dlt\_Click – определяет операции, которые выполняются при нажатиии на кнопку button\_dlt

private void button2\_Click(object sender, EventArgs e)

**НАЧАЛО**

arrpath = arrpath.Where(w => w != arrpath[listBox1.SelectedIndex]).ToArray();

arrfilename = arrfilename.Where(w => w != arrfilename[listBox1.SelectedIndex]).ToArray();

timetable = timetable.Where(w => w != timetable[listBox1.SelectedIndex]).ToArray();

clickCounter--;

listBox1.Items.Remove(listBox1.SelectedItem);

**КОНЕЦ**

* + - 1. Метод button\_time\_Click - определяет операции, которые выполняются при нажатиии на кнопку button\_time

private void button4\_Click(object sender, EventArgs e)

**НАЧАЛО**

**TRY**

timetable[listBox1.SelectedIndex] = dateTimePicker1.Value.ToString();

richTextBox\_info.AppendText(dateTimePicker1.Value.ToString());

**TRY**

**CATCH** (Exception)

MessageBox.Show("Выберите задачу, которой хотите задать время!");

**CATCH**

**КОНЕЦ**

* + - 1. Метод button1\_Click - определяет операции, которые выполняются при нажатиии на кнопку button1. Добавляет задачу в очередь

private void button1\_Click(object sender, EventArgs e)

**НАЧАЛО**

operate();

InitializeTimer();

**КОНЕЦ**

## **Описание программной реализации**

Разработано:

* Form1 – форма с интерфейсом ПО
* Form1 – главный класс программы
  + 1. Методы класса Form1
       1. Метод **operate**

Тип доступа - public

Входные параметры - нет

Возвращаемое значение - нет

* + - 1. Метод **button1\_Click**

Тип доступа - private

Входные параметры - нет

Возвращаемое значение - нет

* + - 1. Метод **button\_time\_Click**

Тип доступа - private

Входные параметры - нет

Возвращаемое значение - нет

* + - 1. Метод **button\_dlt\_Click**

Тип доступа - private

Входные параметры - нет

Возвращаемое значение - нет

* + - 1. Метод **button\_cmd\_Click**

Тип доступа - private

Входные параметры – нет

Возвращаемое значение – нет

* + - 1. Метод **listBox1\_DoubleClick**

Тип доступа - private

Входные параметры - нет

Возвращаемое значение - нет

* + - 1. Метод **Timer1\_Tick**

Тип доступа - private

Входные параметры - нет

Возвращаемое значение - нет

* + - 1. Метод **InitializeTimer**

Тип доступа - private

Входные параметры - нет

Возвращаемое значение – нет

* + 1. Поля класса Form1
       1. Поле **clickCounter**

Тип – int

Описание: подсчитывает количество задач

* + - 1. Поле **arrpath**

Тип – string[ ]

Описание: массив, хранящий в себе расположение каждой задачи

* + - 1. Поле **arrfilename**

Тип – string[ ]

Описание: массив, хранящий в себе название каждой задачи

* + - 1. Поле **timetable**

Тип – string[ ]

Описание: массив, хранящий время выполнения для каждой задачи

* + - 1. Поле **command**

Тип – string

Описание: предназначена для передачи команды в командную строку, если она вызывается

# **ПРОЕКТИРОВАНИЕ И ПРОВЕДЕНИЕ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

## **Системное тестирование**

Объект тестирования - ПО «Планировщик задач». Тестирование проводится разработчиком программного обеспечения, сроки выполнения тестирования 1 день. ПО тестируется в среде Windows 8

* + 1. **План системного тестирования**

В таблице 9.1 приведен набор тестовых данных для программного продукта "Планировщик задач"

Таблица 9.1 – Набор тестовых данных для системного тестирования.

|  |  |  |  |
| --- | --- | --- | --- |
| № теста | Название теста | Входные данные | Ожидаемый результат |
| 1 | Добавление задачи | Нажатие на кнопку, выбор файла | Новая задача добавляется в очередь на выполнение |
| 2 | Удаление задачи | Нажатие на задачу в списке, нажатие на кнопку «Удалить» | Задача удалена из очереди на выполнение |
| 3 | Присвоение задачи времени для запуска | Нажатие на задачу в списке, выбор времени в поле, нажатие на кнопку «Назначить время» | Задаче назначено время для запуска |

* + 1. **Проведение системного тестирования**

В таблице 9.2 приведен набор тестовых данных для системного тестирования

Таблица 9.2 – Набор тестовых данных для системного тестирования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Название теста | Входные данные | Ожидаемый результат | Успешность |
| 1 | Добавление задачи | Нажатие на кнопку, выбор файла | Новая задача добавляется в очередь на выполнение | + |
| 2 | Удаление задачи | Нажатие на задачу в списке, нажатие на кнопку «Удалить» | Задача удалена из очереди на выполнение | + |
| 3 | Присвоение задачи времени для запуска | Нажатие на задачу в списке, выбор времени в поле, нажатие на кнопку «Назначить время» | Задаче назначено время для запуска | + |

## **Интеграционное тестирование**

Объектом тестирования в данном виде выступает элемент listBox\_task, который выводит список всех задач. Тестирование проводится разработчиком программного обеспечения. Сроки проведения от 1 до 2 дней.

* + 1. **План интеграционного тестирования**

В таблице 9.3 приведен набор тестовых данных для программного продукта "Планировщик задач"

Таблица 9.3 – Набор тестовых данных для интеграционного тестирования.

|  |  |  |  |
| --- | --- | --- | --- |
| № теста | Название теста | Входные данные | Ожидаемый результат |
| 1 | Добавление задачи в список | Нажатие на кнопку, выбор файла | Новая задача появляется в списке |
| 2 | Удаление задачи из списка | Нажатие на задачу в списке, нажатие на кнопку «Удалить» | Задача удалена из списка |
| 3 | Просмотр информации о задаче | Двойное нажатие на задачу в списке | В окно «Информация о задаче» выводится информация о задаче |

* + 1. **Проведение интеграционного тестирования**

В таблице 9.4 приведен набор тестовых данных для интеграционного тестирования

Таблица 9.4 – Набор тестовых данных для интеграционного тестирования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Название теста | Входные данные | Ожидаемый результат | Успешность |
| 1 | Добавление задачи в список | Нажатие на кнопку, выбор файла | Новая задача появляется в списке | + |
| 2 | Удаление задачи из списка | Нажатие на задачу в списке, нажатие на кнопку «Удалить» | Задача удалена из списка | + |
| 3 | Просмотр информации о задаче | Двойное нажатие на задачу в списке | В окно «Информация о задаче» выводится информация о задаче | + |

## **Автономное тестирование**

Объектом тестирования в данном виде выступают все методы всех классов, предоставленных в данном курсовом проекте. Тестирование проводится разработчиком программного обеспечения, сроки проведения от 1 до 3 дней.

* + 1. **Проведение автономного тестирования**

В таблице 9.5 – 9. 11 приведены набор тестовых данных для интеграционного тестирования

Таблица 9.5 - Метод «operate()». Выполнение автономного тестирования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Название теста | Входные данные | Ожидаемый результат | Успешность |
| 1 | Добавление задачи в список | Нажатие на кнопку, выбор файла | Новая задача появляется в списке | + |

Таблица 9.6 - Метод «button1\_Click». Выполнение автономного тестирования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Название теста | Входные данные | Ожидаемый результат | Успешность |
| 1 | Добавление задачи | Нажатие на кнопку, выбор файла | Новая задача появляется в списке и очереди на выполнение, таймер инициализирован | + |

Таблица 9.7 - Метод «InitializeTimer». Выполнение автономного тестирования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Название теста | Входные данные | Ожидаемый результат | Успешность |
| 1 | Инициализация таймера | Нажатие на кнопку, выбор файла, вызов метода | Таймер инициализирован и работает | + |

Таблица 9.8 - Метод «Timer1\_Tick». Выполнение автономного тестирования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Название теста | Входные данные | Ожидаемый результат | Успешность |
| 1 | Проверка работы таймера | Вызов метода в коде | Таймер работает, задачи выполняются | + |

Таблица 9.9 - Метод «listBox1\_DoubleClick». Выполнение автономного тестирования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Название теста | Входные данные | Ожидаемый результат | Успешность |
| 1 | Просмотр информации о задаче | Двойное нажатие по задаче в списке | В окне «Информация о задаче» отображается информация о задаче | + |

Таблица 9.10 - Метод «button\_dlt\_Click». Выполнение автономного тестирования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Название теста | Входные данные | Ожидаемый результат | Успешность |
| 1 | Удаление задачи | Нажатие на задачу в списке, нажатие на кнопку «Удалить» | Задача удалена из списка и очереди | + |

Таблица 9.11 - Метод «button\_time\_Click». Выполнение автономного тестирования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Название теста | Входные данные | Ожидаемый результат | Успешность |
| 1 | Присвоение задачи времени для выполнения | Нажатие на задачу в списке, выбор времени в поле, нажатие на кнопку «Назначить время» | Задаче задано время для выполнения | + |

**Вывод раздела:**

При проведении системного тестирования проверено работоспособность программы в целом и по всем функциональным возможностям. При проведении интеграционного тестирования протестирован отдельный элемент программы. При проведении автономного тестирования протестировано 7 методов.

В результате проведения всех видов тестирования все обнаруженные ошибки исправлены, повторное тестирование новых не выявило

# **ВЫВОД**

В ходе выполнения данного курсового проекта на тему «Планировщик задач» был разработан пакет документов и поставленная задача была реализована на ПК. Пакет документов содержит пояснительную записку и описание программы. Программная реализация курсового проекта выполнена на языке программирования С#.

Пользователь имеет возможность выполнять такие действия как: выбор задачи, задание ей времени для выполнение, удаление задачи и просмотр информации о ней.

Во время выполнения данного курсового проекта получен опыт в разработке функционально законченного программного продукта.

# **ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ**

* 1. Пудовкина, Л.Ф. Анализ требований к программному обеспечению [Текст]: учеб. пособие. / Л.Ф. Пудовкина. - Х. Нац. аерокосм. ун-т им. Н. Е. Жуковского «Харьк. авиация. ин-т », 2011. - 99 с.
  2. Кватрани Т. RationalRose 2000 иUML. Визуальное моделирование: Пер. с англ. - М.: ДМК Пресс, 2001. - 176с.: Ил. (Серия «объектно-ориентированные технологии в программировании»).
  3. Леффингуелл Д., Уидринг Д.Принципы работы с требованиями к программному обеспечению. Унифицированный подход.: Пер. с англ. - М.: Издательский дом «Вильямс», 2002 ...-448с.: Ил. - Парал. Ты т. Англ.

# **ГЛОССАРИЙ**

**Планировщик задач** — [программа](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) ([служба](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B1%D1%8B_Windows)), часто называемая сервисом [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), которая запускает другие программы в зависимости от различных критериев

**Класс Timer** – класс, предоставляющий механизм для выполнения метода в пуле потоков с заданными интервалами. Данный класс не наследуется.

# **ПРИЛОЖЕНИЕ А. ЭКРАННЫЕ ФОРМЫ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ**

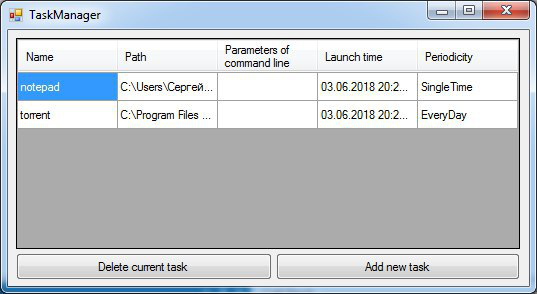


Рисунок А1 – Главная форма приложения

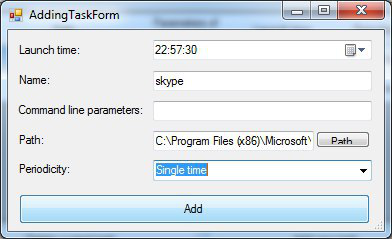


Рисунок А2 – Добавление задачи

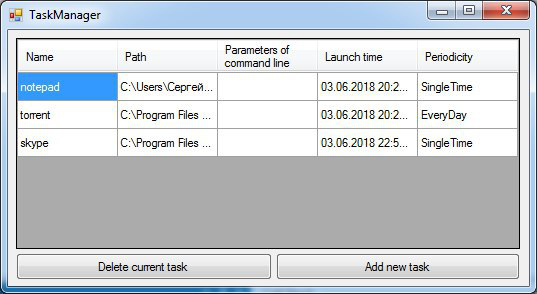


Рисунок А3 – Три задачи в очереди и информация про них

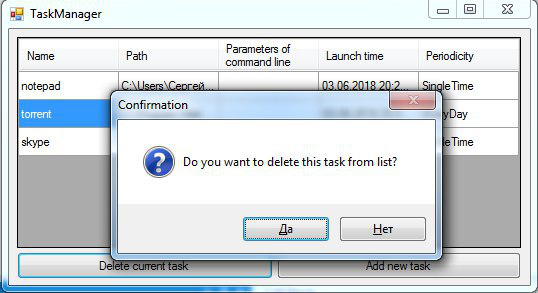


Рисунок А4 – Удаление задачи

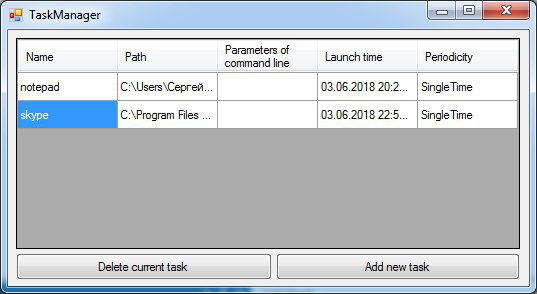


Рисунок А5 – Обновленный список после удаления задачи

# **ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ИСХОДНОГО КОДА**

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MyTaskManager

{

public static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

public static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

**Form1.cs**

using System;

using System.Linq;

using System.Windows.Forms;

namespace MyTaskManager

{

public partial class Form1 : Form

{

public static Tasks ProcessesCollection = new Tasks();

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

TasksSerializer.LoadJson(ref ProcessesCollection);

dataGridViewTasks.InitializeDGV();

dataGridViewTasks.UpdateDGV(ProcessesCollection);

}

private void btnAddTask\_Click(object sender, EventArgs e)

{

using (AddingTaskForm addForm = new AddingTaskForm(dataGridViewTasks))

{

if (addForm.ShowDialog() == DialogResult.OK)

{

}

}

TasksSerializer.SaveJson(ProcessesCollection);

}

private void btnDelete\_Click(object sender, EventArgs e)

{

if (ProcessesCollection.Count == 0)

{

return;

}

if (MessageBox.Show(@"Do you want to delete this task from list?", @"Confirmation",

MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

ProcessesCollection.Remove(ProcessesCollection.

ElementAt(dataGridViewTasks.CurrentCell.RowIndex));

dataGridViewTasks.UpdateDGV(ProcessesCollection);

TasksSerializer.SaveJson(ProcessesCollection);

}

}

}

}

**DGVInitializer.cs**

using System.Linq;

using System.Windows.Forms;

namespace MyTaskManager

{

public static class DGVInitializer

{

public static void InitializeDGV(this DataGridView dgv)

{

dgv.ColumnCount = 5;

dgv.Columns[0].HeaderText = "Name";

dgv.Columns[1].HeaderText = "Path";

dgv.Columns[2].HeaderText = "Parameters of command line";

dgv.Columns[3].HeaderText = "Launch time";

dgv.Columns[4].HeaderText = "Periodicity";

dgv.RowHeadersVisible = false;

dgv.AllowUserToAddRows = false;

dgv.AllowUserToDeleteRows = false;

}

public static void UpdateDGV(this DataGridView dgv, Tasks processesCollection)

{

dgv.RowCount = processesCollection.Count;

for (int i = 0; i < processesCollection.Count; i++)

{

dgv.Rows[i].Cells[0].Value = processesCollection.ElementAt(i).Name;

dgv.Rows[i].Cells[1].Value = processesCollection.ElementAt(i).Path;

dgv.Rows[i].Cells[2].Value = processesCollection.ElementAt(i).ParamsOfCommandLine;

dgv.Rows[i].Cells[3].Value = processesCollection.ElementAt(i).LaunchTime.ToLocalTime();

dgv.Rows[i].Cells[4].Value = processesCollection.ElementAt(i).PeriodicityOfLaunch;

}

}

}

}

**AddingTaskForm.cs**

using System;

using System.IO;

using System.Windows.Forms;

using Exception = System.Exception;

using Task = MyTaskManager.Task;

namespace MyTaskManager

{

public partial class AddingTaskForm : Form

{

private DataGridView \_dgv;

public AddingTaskForm(DataGridView dgv)

{

InitializeComponent();

this.\_dgv = dgv;

}

public Task GetTask()

{

if (string.IsNullOrEmpty(nameTextBox.Text))

{

throw new Exception("Please, enter the name of process.");

}

if (string.IsNullOrEmpty(pathTextBox.Text))

{

throw new Exception("Please, enter the path of your process.");

}

if (DateTimePickerDate.Value.ToUniversalTime() < DateTime.UtcNow)

{

throw new Exception("Please, choose the date of launch later than now.");

}

if (!File.Exists(pathTextBox.Text))

{

throw new Exception("Please, choose the path which exists.");

}

return new Task(nameTextBox.Text, pathTextBox.Text, paramsOfCommandLineTextBox.Text,

DateTimePickerDate.Value.ToUniversalTime(), periodicityOfLaunchComboBox.SelectedIndex);

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

Form1.ProcessesCollection.Add(GetTask());

\_dgv.UpdateDGV(Form1.ProcessesCollection);

this.Close();

}

catch (Exception exc)

{

MessageBox.Show(exc.Message, "Something is wrong", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

private void buttonPath\_Click(object sender, EventArgs e)

{

if (openFileDialogPath.ShowDialog() == DialogResult.OK)

{

pathTextBox.Text = openFileDialogPath.FileName;

}

}

private void AddingTaskForm\_Load(object sender, EventArgs e)

{

DateTimePickerDate.MinDate = DateTime.UtcNow;

DateTimePickerDate.Format = DateTimePickerFormat.Time;

}

}

}

**Task.cs**

using System;

using System.Collections.Generic;

namespace MyTaskManager

{

[Serializable]

public class Tasks : SortedSet<Task>

{

public Tasks()

{

}

}

[Serializable]

public class Task:IComparable

{

public Task()

{

}

public Task(string name, string path, string paramsOfCommandLine,

DateTime launchTime, int periodicityOfLaunch)

{

Name = name;

Path = path;

ParamsOfCommandLine = paramsOfCommandLine;

LaunchTime = launchTime;

if (periodicityOfLaunch == -1)

{

throw new Exception("Please, enter the periodicity of launch");

}

PeriodicityOfLaunch = (Periodicity)periodicityOfLaunch;

}

public string Name { get; set; }

public string Path { get; set; }

public string ParamsOfCommandLine { get; set; }

public DateTime LaunchTime { get; set; }

public Periodicity PeriodicityOfLaunch { get; set; }

public override bool Equals(object obj)

{

if (obj != null && (Name == (obj as Task)?.Name && Path == (obj as Task)?.Path &&

ParamsOfCommandLine == (obj as Task)?.ParamsOfCommandLine &&

LaunchTime == ((Task) obj).LaunchTime))

{

return true;

}

return false;

}

public int CompareTo(object obj)

{

if (!this.Equals(obj))

{

if (LaunchTime.CompareTo((obj as Task).LaunchTime) != 0)

{

return LaunchTime.CompareTo((obj as Task).LaunchTime);

}

if (Name.CompareTo((obj as Task).Name) != 0)

{

return Name.CompareTo((obj as Task).Name);

}

if (ParamsOfCommandLine.CompareTo((obj as Task).ParamsOfCommandLine) != 0)

{

return ParamsOfCommandLine.CompareTo((obj as Task).ParamsOfCommandLine);

}

}

return 0;

}

public override string ToString()

{

return "Name: " + Name + " Planned date: " + LaunchTime + " Parameters: " + ParamsOfCommandLine;

}

}

[Serializable]

public enum Periodicity

{

SingleTime,

EveryDay,

OnceAWeek,

OnceAMonth

}

}

**TaskUpdater.cs**

namespace MyTaskManager.Data

{

public static class TaskUpdater

{

public static bool UpdateOrDelete(Task task)

{

switch ((int)task.PeriodicityOfLaunch)

{

case 0: return false;

case 1: task.LaunchTime = task.LaunchTime.AddDays(1);

return true;

case 2: task.LaunchTime = task.LaunchTime.AddDays(7);

return true;

case 3: task.LaunchTime = task.LaunchTime.AddMonths(1);

return true;

default: return false;

}

}

}

}

**TaskUpdaerTest.cs**

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using MyTaskManager.Data;

namespace MyTaskManager.Tests

{

[TestClass]

public class TaskUpdaterTest

{

[TestMethod]

public void UpdateOrDelete\_TaskWithPeriodicitySingleTime\_returnFalse()

{

Task testingTask = new Task("testTask", "path1", "", DateTime.UtcNow.AddDays(-5), 0);

bool actualResult = TaskUpdater.UpdateOrDelete(testingTask);

bool expected = false;

Assert.AreEqual(expected, actualResult);

}

[TestMethod]

public void UpdateOrDelete\_TaskWithPeriodicityOnceAWeek\_returnTrueAndDateChanged()

{

Task testingTask = new Task("testTask", "path1", "", DateTime.UtcNow.AddDays(-5), 2);

bool actualResult = TaskUpdater.UpdateOrDelete(testingTask);

bool actualTimeResult =

(testingTask.LaunchTime - DateTime.UtcNow.AddDays(-5).AddDays(7) <

DateTime.UtcNow - DateTime.UtcNow.AddSeconds(-2))

? true

: false;

bool expected = true;

Assert.AreEqual(expected, actualResult && actualTimeResult);

}

}

}