МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ П.М. МАШЕРОВА»

Факультет математики и информационных технологий

Кафедра прикладного и системного программирования

КУРСОВАЯ РАБОТА

# по дисциплине «Операционные системы и системное программирование»

**разработка приложения «Task scheduler»**

Величко Дмитрий Марьянович,

3 курс, группа 34з

Руководитель:

Сергеенко Сергей Владимирович,

**старший преподаватель**

Витебск, 2021

РЕФЕРАТ

Отчет по курсовой работе: с. 18, рис 8, ист. 5, прил. 2.

ПРОГРАММНЫЙ ПРОДУКТ, ДЕКСТОП ПРИЛОЖЕНИЕ, СРУКТУРА, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ПРОЕКТ, ИНТЕРФЕЙС, СРЕДСТВА РАЗРАБОТКИ, ПРОЕКТИРОВАНИЕ, ТЕСТИРОВАНИЕ.

Целью работы является разработка приложения «Task Scheduler».

Объектом и предметом исследования являются технология «.NET».

В первом разделе содержится описание предметной области и описание используемых технологий.

Во втором разделе описывается практическая реализация проекта.

Автор проекта подтверждает, что приведённый в нём расчётно-аналитический материал правильно и объективно отражает состояние исследуемого процесса, а все заимствованные из литературных и других источников теоретические и методологические положения и концепции сопровождаются ссылками на их автора

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc62468939)

[1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc62468940)

[1.1 Язык программирования С# 5](#_Toc62468941)

[1.2 Технология .NET 6](#_Toc62468942)

[2 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ПРОЕКТА 9](#_Toc62468945)

[2.1 Формирование структуры проекта 9](#_Toc62468946)

[2.2 Алгоритмическое обеспечение проекта 9](#_Toc62468947)

[2.3 Реализация программного продукта 10](#_Toc62468948)

[ЗАКЛЮЧЕНИЕ 15](#_Toc62468949)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 16](#_Toc62468950)

[ПРИЛОЖЕНИЕ А 17](#_Toc62468951)

[ПРИЛОЖЕНИЕ Б 20](#_Toc62468952)

# ВВЕДЕНИЕ

Уже несколько десятков лет компьютер занял большую часть нашей повседневной жизни, и многие не представляют своей учёбы, работы без компьютера. В настоящее время трудно назвать те области человеческой деятельности, успехи в которых не были бы связаны с использованием компьютера. Сфера применения компьютера постоянно расширяется, существенно влияя на развитие производительных сил нашего общества. Непрерывно изменяются технико-экономические характеристики компьютера, например, такие, как быстрота действия, ёмкость памяти, надёжность в работе, стоимость, удобства в эксплуатации, габаритные размеры, потребляемая мощность и др. В широком понимании всякий компьютер рассматривается как преобразователь информации.

Целью данного курсового проекта является создание приложения для планирования исполняемых файлов, на языке С# (.NET), с использованием WinForms API.

При разработке приложения, необходимо решить следующие задачи:

* создание новой задачи, которая будет выполняться в запланированное время;
* вывод информации на экран: список всех задач (пользовательские, системные).;
* возможность удаления запланированных задач.

# 1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

## Язык программирования C#

Язык C#, разработанный компанией Майкрософт, один из самых популярных современных языков программирования. Он востребован на рынке разработки в различных странах, C# применяют при работе с программами для ПК, создании сложных веб-сервисов или мобильных приложений. Появившийся как язык для собственных нужд платформы Microsoft .NET, постепенно этот язык стал очень популярным.

В результате язык получился крайне гибкий, мощный и универсальный. На нем пишут практически все, что угодно, от небольших веб-приложений до мощных программных систем, объединяющих в себе веб-структуры, приложения для десктопов и мобильных устройств. Все это стало возможным благодаря удобному Си-подобному синтаксису, строгому структурированию, огромному количеству фреймворков и библиотек (их число достигает нескольких сотен).

Долгое время платформа .NET поставлялась с закрытым ядром, что создавало определенные сложности в разработке и снижало популярность C# в профессиональной среде. Но в ноябре 2014 Майкрософт радикально изменила подход и стала выдавать бесплатные лицензии для Visual Studio уже с открытым исходным кодом для всех наборов инструментов.

Инструментарий C# позволяет решать широкий круг задач, язык действительно очень мощный и универсальный. На нем разрабатывают:

* Приложения для WEB.
* Различные игровые программы.
* Приложения платформ Android или iOS.
* Программы для Windows.

## 1.2 Технология .NET

.NET – это платформа от Microsoft, которая позволяет создавать программные приложения. Первый выпуск .NET Framework состоялся в 2002 году. Считается, что .NET Framework была создана как альтернатива платформе Java от компании Sun. Главное отличие состоит в том, что .NET Framework официально рассчитана на работу именно с операционными системами семейства Microsoft Windows. С тех пор она прошла долгий путь от версии 1.0 до 4.8 (18 апреля 2019), и на сегодняшний день, несмотря на появление платформы нового поколения (.NET Core), по-прежнему достаточно популярна: существует множество программных продуктов, библиотек и фреймворков, которые написаны и развиваются под .NET Framework.

В 2016 году в дополнение к .NET Framework выпущена модульная платформа .NET Core, которая совместима с различными операционными системами. Другими словами, она кроссплатформенная. Кроссплатформенность .NET Core открыла большое количество новых сценариев и возможностей её применения. Это сыграло существенную роль в продвижении .NET среди разработчиков и представителей бизнеса.

Многие считают, что язык С# и платформа .NET – это одно и то же. Конечно, это не так. Они, несомненно, развиваются с оглядкой друг на друга, но не имеют строгой взаимозависимости. Например, помимо официально поддерживаемых реализаций .NET, существуют и альтернативные варианты, такие как Mono, .NET Compact Framework, .NET Micro Framework и другие. На всех этих платформах мы можем применять язык C#, но в определённой степени. С другой стороны, с .NET совместим не только C#, но и другие языки: F#, VB.NET и даже C++.

Разработчики, которые знают разные языки, могут собраться вместе и написать общий программный продукт под конкретную .NET-платформу. Элементы этого продукта, написанные на разных языках, смогут коммуницировать между собой без всяких проблем. Кстати, это объясняет, почему комьюнити .NET такое большое и разнообразное: оно объединяет программистов, которые пишут на разных языках.

# 2 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ПРОЕКТА

## 2.1 Формирование структуры проекта

Разработка алгоритма функционирования программного продукта проводилась нисходящим методом, который заключается в том, что разрабатываемый алгоритм разделяется на дочерние алгоритмы, а те, в свою очередь, разделяются на элементарные алгоритмы. Структуру проекта можно представить в виде нескольких блоков. В первом блоке выполняется запуск программы, в ходе которого происходит инициализация всех компонентов формы. Во втором блоке осуществляется просмотр и манипуляция запланированными задачами в ОС Windows.

## 2.2 Алгоритмическое обеспечение проекта

В ходе проектирования программного продукта исследована предметная область, собрана необходимая информация, определены цели и задачи проектирования. Интерфейс программы должен быть простым и удобным для пользователя. При этом программный продукт должен хранить необходимую информацию и выполнять заявленные функции. Для чего необходимо создать программную оболочку для работы с WinAPI.

Вкратце, бизнес логика программного продукта была разработана с помощью паттерна MVC (Model View Controller). MVC – это схема разделения данных приложения, и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо. Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние. Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели. Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений. Структура проекта в виде MVC можно увидеть на диаграмме, которая отображена внизу (Рисунок 2.0).



Рисунок 2.0 – MVC процесс

Таким образом было спроектирован каскад для основной бизнес-логики программного продукта.

## 2.3 Реализация программного продукта

Процесс создания программного продукта делится на несколько стадий:

* изучение выбранной предметной области;
* обработка полученной информации;
* выбор средств реализации;
* программирование;
* тестирование;
* эксплуатация.

Наиболее целесообразный метод тестирования – проверка работоспособности основных функций программы. Программа должна нормально функционировать при внесении информации и т. п. Процесс создания программного продукта делится на несколько стадий: изучение выбранной предметной области; обработка полученной информации; выбор средств реализации; программирование; тестирование; эксплуатация.

Для запуска приложения, необходимо иметь на компьютере или на удалённом сервере Windows 7 и выше. В папке проекта есть директория «bin» и в ней папка «Release», там вы можете найти исполняемый файл для запуска программного продукта. После запуска мы видим интерфейс программного продукта, изображенная на рисунке 2.1.

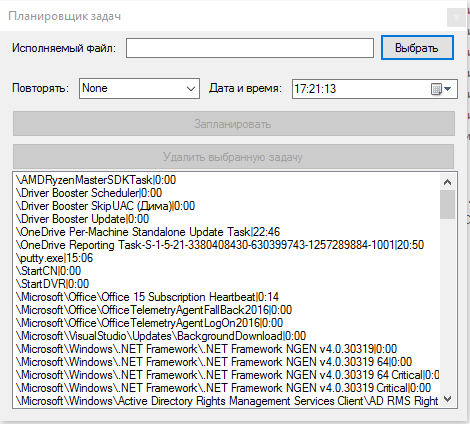


Рисунок 2.1 – Интерфейс приложения

На главной странице сайта представлен перечень всех запланированных задач (пользовательских и системных).

Нажав на кнопку «Выбрать», отобразится окно с выбором любого исполняемого файла (так же файл можно прописать вручную) рисунок 2.2.

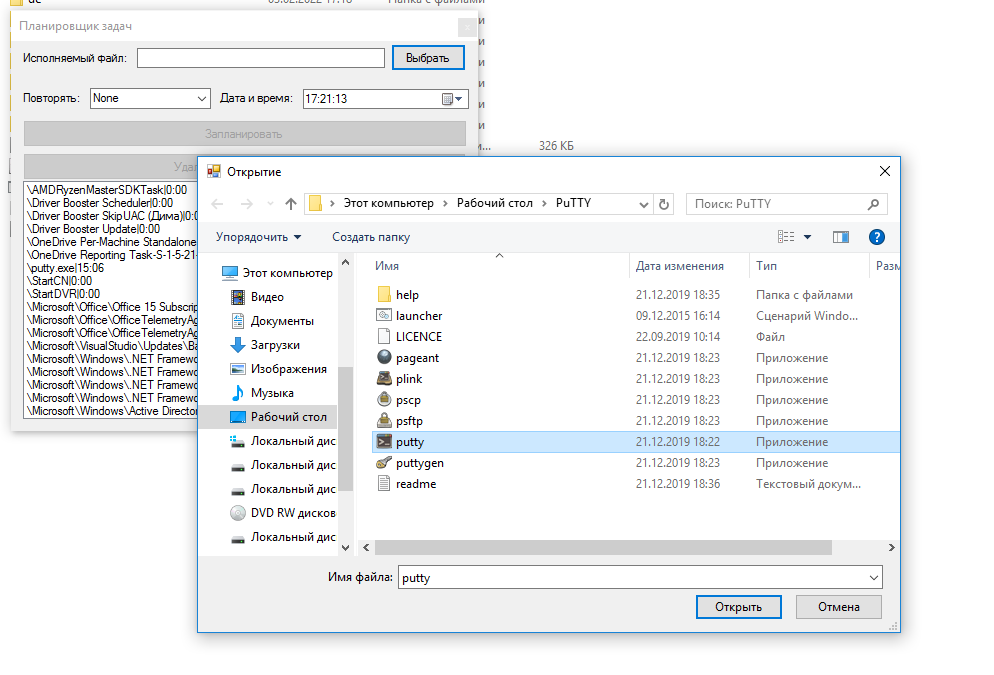


Рисунок 2.2 – Выбор исполняемого файла

После выбора исполняемого файла, видим, что у нас доступна кнопка «Запланировать». (рисунок 2.3) Поэтому мы выбираем время и после нажатия видим окно с успешным планированием задачи. (рисунок 2.4)

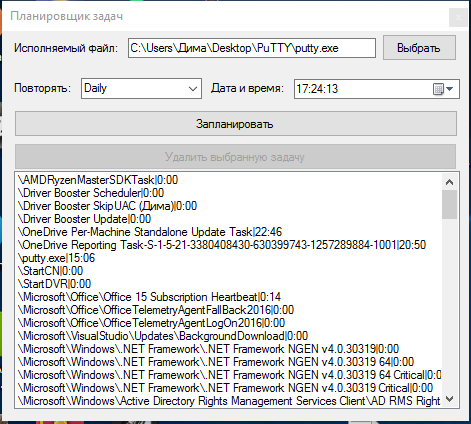


Рисунок 2.3 – Разблокирована кнопка «Запланировать»

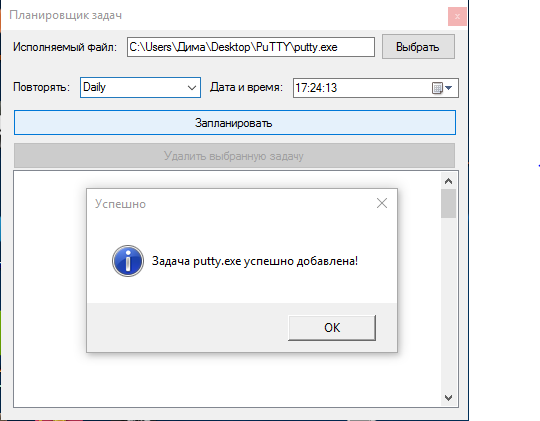


Рисунок 2.4 – Успешно запланированная задача

Для проверки, ждем время 17:24:13 и видим, что exe файл открылка успешно. (рисунок 2.5).

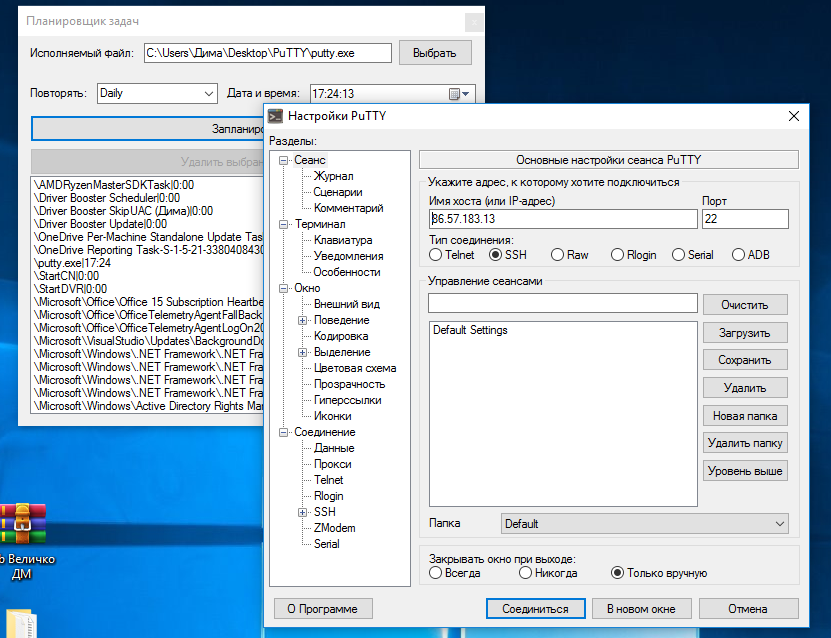


Рисунок 2.5 – Успешно открытый исполняемый файл

Так же есть возможность удалять запланированные задачи, это поможет нам, если мы не так сконфигурировали нашу новую задачу. После выбора элемента в списке с задачами у нас появляется активная кнопка «Удалить выбранную задачу». (рисунок 2.6)

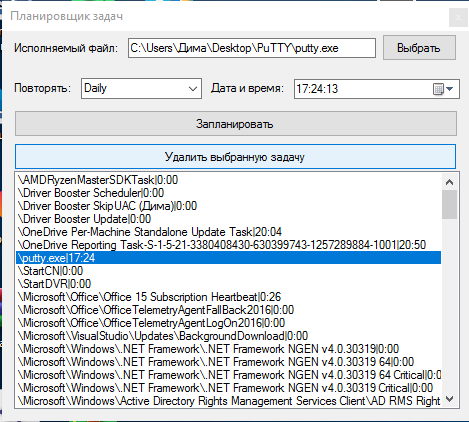


Рисунок 2.6 – Активная кнопка «Удалить выбранную задачу»

Нажмем на эту кнопку, и мы увидим окно с успешным удалением (рисунок 2.7)

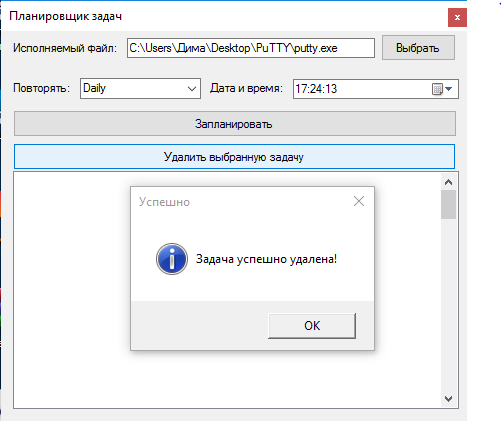


Рисунок 2.7 – Сообщение об успешном удалении задачи

После нажатия на кнопку «OK» у нас появиться список обновленных задач, где нету нашей старой задачи. (рисунок 2.8)

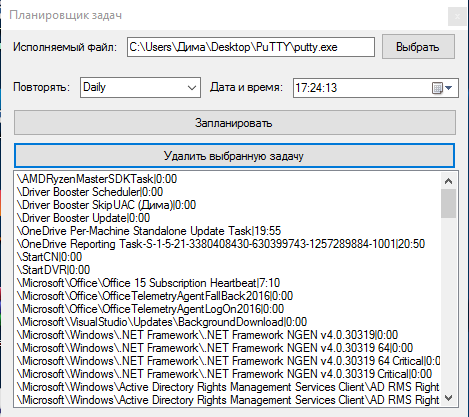


Рисунок 2.8 – Результат после удаления задачи

# 

# ЗАКЛЮЧЕНИЕ

В ходе курсовой работы была разработан и реализовано приложение для планирования исполняемых файлов, на языке С# (.NET), с использованием WinForms API. Приложение было реализовано на технологиях: C#, .NET. Программа предоставляет возможность ввод, удаление, хранения и просмотр информации. Реализованное приложение будет полезна людям, которые следят за своим компьютером и хотят быстро в одну кнопку удалять задачи, которые нагружают компьютер. По окончанию разработки программного продукта были достигнуты следующие цели:

* создание новой задачи, которая будет выполняться в запланированное время;
* вывод информации на экран: список всех задач (пользовательские, системные).;
* возможность удаления запланированных задач.

Разработка программы позволила мне укрепить знания в технологии .NET, а именно в Windows Forms и WinAPI для C#.

Таким образом, цель курсового проектирования была успешно достигнута.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. «MVC wiki» [Электронный ресурс] – Режим доступа - <https://ru.wikipedia.org/wiki/Model-View-Controller> – Дата доступа: 02.02.2022.
2. «C# документация» [Электронный ресурс] – Режим доступа - <https://docs.microsoft.com/en-us/dotnet/csharp/> – Дата доступа: 02.02.2022.
3. «C# wiki» [Электронный ресурс] – Режим доступа - <https://ru.wikipedia.org/wiki/C_Sharp> – Дата доступа: 02.02.2022.
4. «Windows Forms документация» [Электронный ресурс] – Режим доступа - https://docs.microsoft.com/en-us/dotnet/desktop/winforms/ – Дата доступа: 02.02.2022.
5. «Документация WinAPI» [Электронный ресурс] – Режим доступа: <https://docs.microsoft.com/en-us/dotnet/api/microsoft.win32> – Дата доступа: 02.02.2022.

# ПРИЛОЖЕНИЕ А

ФРАГМЕНТ КОДА ОТВЕЧАЮЩИЙ ЗА СОЗДАНИЕ НОВОЙ ЗАДАЧИ

public void submitBtnHandler()

{

try

{

String path = this.pathBox.Text;

DateTime dateTime = this.dateTimePicker.Value;

QuickTriggerType repeatType = QuickTriggerType.Idle;

try

{

repeatType = (QuickTriggerType)Enum.Parse(typeof(QuickTriggerType), this.comboBox.Text);

}

catch { }

Trigger trigger = getTriggerByType(repeatType, dateTime);

TaskDefinition td = TaskService.Instance.NewTask();

td.RegistrationInfo.Description = "Proccess";

td.Principal.LogonType = TaskLogonType.InteractiveToken;

td.Triggers.Add(trigger);

td.Actions.Add(new ExecAction(path));

TaskService.Instance.RootFolder.RegisterTaskDefinition(path.Split('\\').Last(), td);

fetchAllTasks();

MessageBox.Show("Задача " + path.Split('\\').Last() + " успешно добавлена!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information, MessageBoxDefaultButton.Button1, MessageBoxOptions.DefaultDesktopOnly);

}

catch (Exception e)

{

MessageBox.Show(e.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error, MessageBoxDefaultButton.Button1, MessageBoxOptions.DefaultDesktopOnly);

}

}

ФРАГМЕНТ КОДА, ОТВЕЧАЮЩИЙ ЗА ОБНОВЛЕНИЯ СПИСКА ЗАДАЧ

public List<WinTask> fetchTasks()

{

RunningTaskCollection collection = TaskService.Instance.GetRunningTasks(true);

List<WinTask> winTasks = new List<WinTask>();

foreach (Microsoft.Win32.TaskScheduler.Task rt in TaskService.Instance.AllTasks)

{

if (rt != null)

{

winTasks.Add(new WinTask(rt.Path, rt.NextRunTime));

}

}

this.tasks = winTasks;

return this.tasks;

}

ФРАГМЕНТ КОДА, ОТВЕЧАЮЩИЙ ЗА УДАЛЕНИЕ ВЫБРАННОЙ ЗАДАЧИ

public void removeTaskHandler()

{

try

{

if (taskList.SelectedIndex != -1)

{

String taskName = taskList.Items[taskList.SelectedIndex].ToString().Split('|').First();

TaskService.Instance.RootFolder.DeleteTask(taskName);

fetchAllTasks();

MessageBox.Show("Задача успешно удалена!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information, MessageBoxDefaultButton.Button1, MessageBoxOptions.DefaultDesktopOnly);

}

}

catch (Exception e)

{

MessageBox.Show(e.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error, MessageBoxDefaultButton.Button1, MessageBoxOptions.DefaultDesktopOnly);

}

}

# ПРИЛОЖЕНИЕ Б

Содержание электронного носителя

На электронном носителе расположены следующие файлы и директории:

– Файл «Величко Д.М. – 3 курс – Task Scheduler.docx» представляет собой текстовый документ с курсовой работой.

– Директория «Проект» содержит готовый проект курсовой работы.