

Национална Олимпиада по Информационни Технологии

Областен Кръг

Тема на Проекта

MultiTask

Автор

Александър Ганчев Горанов - 9 клас

e-mail: alex800kill@gmail.com

Телефон: 0888980066

Училище

Природоматематическа гимназия "Акад. Иван Гюзелев" - Габрово

Начален Ръководител:

Име и фамилия: Зорница Дженкова

Длъжност: Старши учител по информатика и ИТ

Месторабота: ПМГ "Акад. Иван Гюзелев", гр. Габрово

Съдържание

Резюме.....	2
Цели и Задачи на проекта.....	2
Технически средства.....	3
Основни етапи в реализирането.....	3
Ниво на сложността на проекта.....	4
Логическо и функционално описание.....	4
Интерфейс.....	5-7
Приложение на програмата.....	8
Заключение.....	8
Част от кода на проекта.....	9-11

Резюме

„**MultiTask**“ е приложение с цел да може да намериш всички данни за компютъра си на едно място, като всичката информация за вашите приложения и файлове. Приложението е направено за хора заинтересувани или любопитни за какво става зад екрана и в кутията на компютъра им, приложението става за всеки който иска да разбере повече за компютъра си.

Цели и Задачи на проекта

„**MultiTask**“ има за цел да събере колкото се може повече информация за компютъра ви на едно място.

Силни страни на "**MultiTask**" са:

- Достъпно за всички хора.
- Интуитивен и лесен за използване потребителски интерфейс(UI).
- Възможност да следиш натоварването на компютъра.

Технически средства

За реализирането на проекта са използвани следните програми и технологии:

1. За създаването на дизайна:
 - Photoshop
 - [Flaticon](#)
2. За организиране на проекта:
 - [Trello](#)
 - [Github](#)
3. Помощни средства при изработването:
 - Официалната документация на Microsoft
 - Не официални документации
 - StackOverFlow
 - [Github](#)
4. За изработване на приложението:
 - Visual Studio 2019/2022
 - Език за програмиране C#

Основни етапи в реализирането

1. Допринасяне на идеята
2. Анализиране на идеята
3. Изработване на основата на приложението:
 - Избиране на дизайн и цветова палитра.
 - Изработване на дизайна.
4. Изработване на функционалността на категориите
5. Прилагане на дизайна по кода
6. Тестване и усъвършенстване

Ниво на сложността на проекта

Основни проблеми при реализация на поставените цели и задачи:

1. Взимането на информация на определени компоненти на компютъра:
 - I. Информация за процесора
 - Основна информация - Производителността на процесора
 - Детайли - Трите групи cache (L1, L2, L3)
 - II. Информация за оперативната памет
 - Основна информация - Използваната и свободната памет
 - Детайли - Типа на паметта
2. Променянето и редактирането на основата функционалност на програмата и продължителната ѝ реализация

Логическо и функционално описание

Основни модули

1. Основно меню
 - Начална страница
 - Оперативна памет
 - Процесор
 - Процеси
2. Линк към GitHub проекта

Интерфейс

Интерфейсът на приложението "MultiTask" е функционален и лесен за използване.

Запознаване с интерфейса:

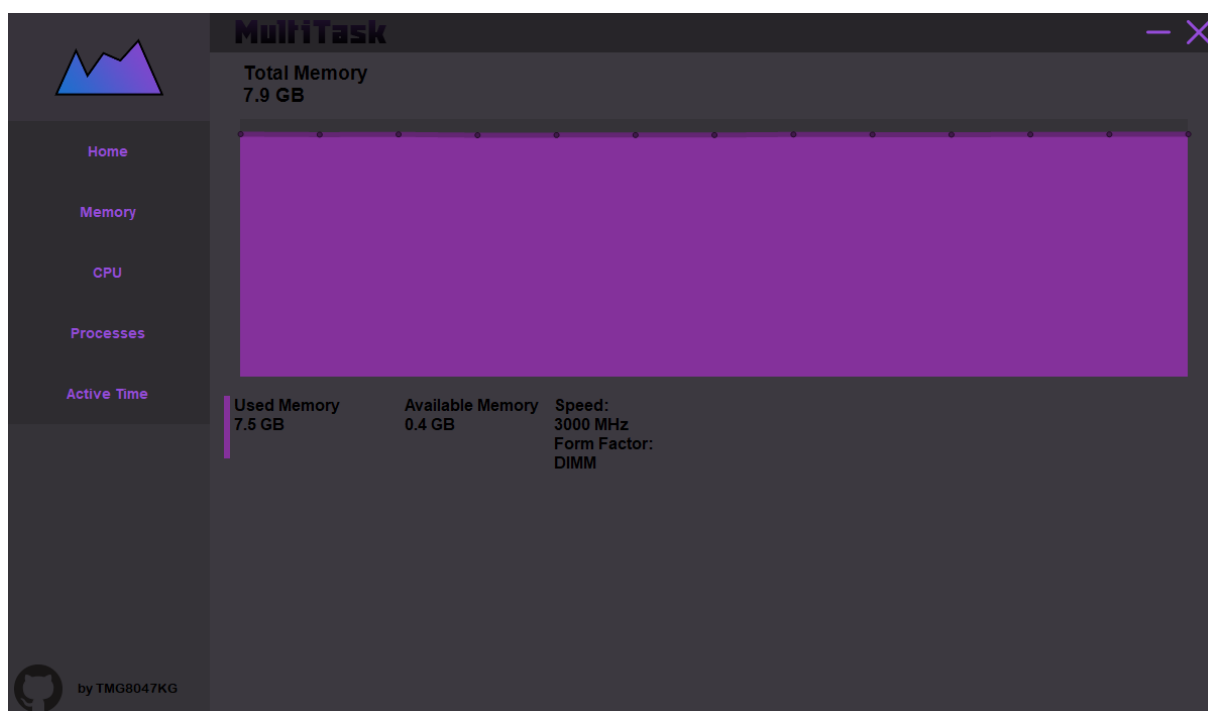
Основното меню



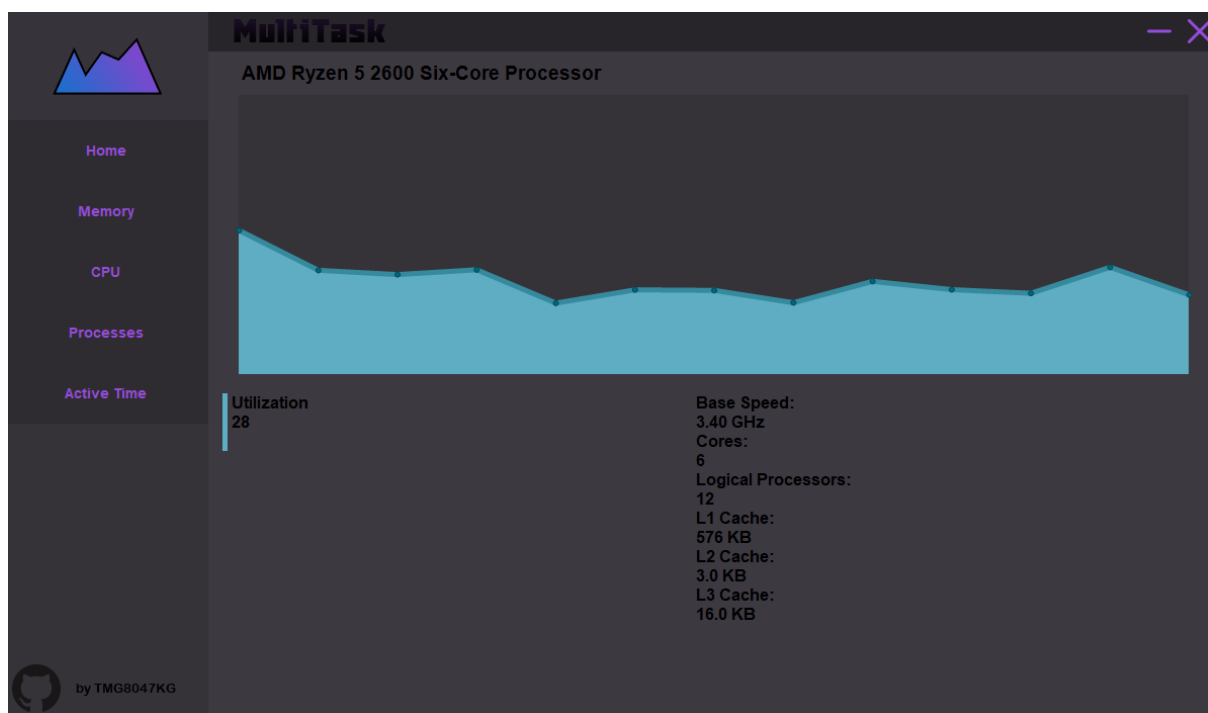
1. Начална страница – В началната страница може да намерите информация за системата ви.



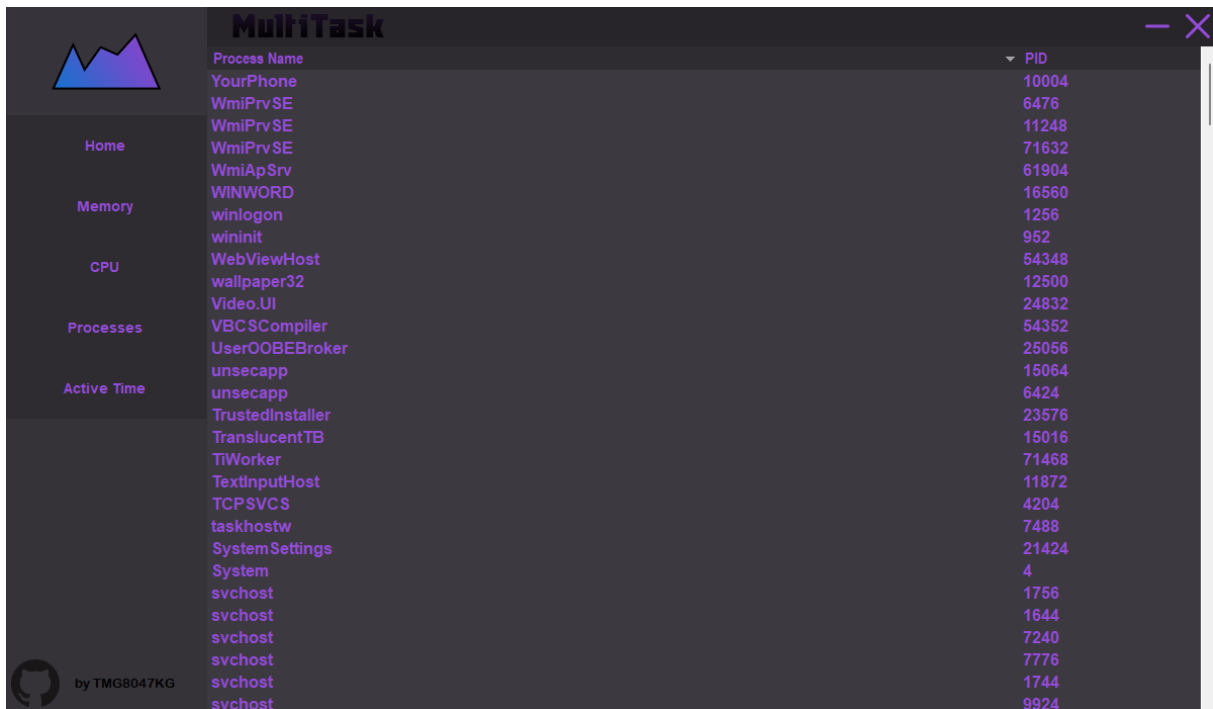
2. Оперативна памет – Тук може да намерите производителността на вашата оперативна памет и информация за нея.



3. Процесор - Тук може да намерите производителността на вашия процесор и информация за него.



4. Процеси – Тук ще намерите всичките приложения които са активни в момента.



	Process Name	PID
	YourPhone	10004
	WmiPrvSE	6476
	WmiPrvSE	11248
Home	WmiPrvSE	71632
	WmiApSrv	61904
Memory	WINWORD	16560
	winlogon	1256
	wininit	952
CPU	WebViewHost	54348
	wallpaper32	12500
	Video.UI	24832
Processes	VBCSCompiler	54352
	UserOOBEBroker	25056
Active Time	unsecapp	15064
	unsecapp	6424
	TrustedInstaller	23576
	TranslucentTB	15016
	TiWorker	71468
	TextInputHost	11872
	TCPSPVCS	4204
	taskhostw	7488
	SystemSettings	21424
	System	4
	svchost	1756
	svchost	1644
	svchost	7240
	svchost	7776
	svchost	1744
	svchost	9924

Приложение на програмата

„**MultiTask**“ е подходящ за хора, които искат да проследят работата на компютъра си, както и да видят информация за системата им и техния hardware. Удобно за проверяване на едно приложение колко натоварва системата.

Заклучение

„**MultiTask**“ е ранна разработка на мулти функционално приложение с голямо количество информация и данни. За в бъдеще се планира да се разшири това количество информация да се добавят повече неща като информация за:

- Видео картата на компютъра (ако има такава)
- Интернет връзките и техния upload и download
- Усъвършенстване на дизайна
- Оптимизиране на приложението
- Добавяне на температурата на процесора и видео картата
- Добавяне на повече възможност за интеракция и контрол от потребителя
- Създаване на сайт за приложението

и още други.

Работата и разработването на приложението е като приключение където откриваш и научаваш нови неща. Това приключение все още достигнало своя край.

Част от кода на проекта

Кода показан по долу е от формата за процесора:

```
...
//Дефиниране на стойности
PerformanceCounter cpuCounter;

private List<float> xV = new List<float>();
private List<float> yV = new List<float>();

long L1, L2, L3;
float tick = 0.0f;

public Processor()
{
    InitializeComponent();
}
//Зареждане и активиране на формата
private void Processor_Load(object sender, EventArgs e)
{
    CPU_tick.Start();
    cpuCounter = new PerformanceCounter("Processor", "% Processor Time", "_Total");

    CPU_cache.GetPerCoreCacheSizes(out L1, out L2, out L3);

    lblL1cache.Text = "L1 Cache:\n" + string.Format("{0} KB", (L1/1024)*6);
    lblL2cache.Text = "L2 Cache:\n" + string.Format("{0:0.0} KB", (L2 / Math.Pow(1024, 2) * 6)
+ 0.0);
    lblL3cache.Text = "L3 Cache:\n" + string.Format("{0:0.0} KB", (L3/Math.Pow(1024, 2) * 2)
+ 0.0);

    GetCPUInfo(0);
    GetCPUInfo(1);
    GetCPUInfo(2);
    GetCPUInfo(3);
```

```
}
```

```
//Метода за вземане на времето от кога работи процесора
```

```
public TimeSpan UpTime
```

```
{
```

```
    get
```

```
    {
```

```
        using (var uptime = new PerformanceCounter("System", "System Up Time"))
```

```
        {
```

```
            uptime.NextValue();
```

```
            return TimeSpan.FromSeconds(uptime.NextValue());
```

```
        }
```

```
    }
```

```
}
```

```
private void TimerUpTime_Tick(object sender, EventArgs e)
```

```
{
```

```
    lblUpTime.Text = "Up Time\n" + UpTime.ToString();
```

```
}
```

```
//Функция за вземане на информацията на процесора
```

```
private void GetCPUInfo(int index)
```

```
{
```

```
    try
```

```
    {
```

```
        ManagementObjectSearcher mosuc = new ManagementObjectSearcher("SELECT * FROM Win32_Processor");
```

```
        foreach (ManagementObject obj in mosuc.Get())
```

```
        {
```

```
            switch (index)
```

```
            {
```

```
                case 0: ProcChart.Titles[0].Text = "" + obj["Name"];
```

```
                break;
```

```
                case 1: lblCores.Text = "Cores:\n" + obj["NumberOfCores"];
```

```
                break;
```

```
                case 2: lblLogicalProcs.Text = "Logical Processors:\n" +  
obj["NumberOfLogicalProcessors"];
```

```
                break;
```

```

        case 3: lblBaseSpeed.Text = "Base Speed:\n" + string.Format("{0:0.00} GHz",
Convert.ToInt32(obj["CurrentClockSpeed"])/1000.0);
            break;
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}

```

//Таймер които се активира всяка секунда в която обновява диаграмата

```

private void CPU_tick_Tick(object sender, EventArgs e)
{
    float util = cpuCounter.NextValue();

    xV.Add(tick);
    yV.Add(util);

    if(tick > 12)
    {
        xV.RemoveAt(0);
        yV.RemoveAt(0);
    }

    ProcChart.ChartAreas["chAreaCPU"].AxisX.Minimum = xV[0];
    ProcChart.ChartAreas["chAreaCPU"].AxisX.Maximum = tick;

    ProcChart.Series["CPUchart"].Points.DataBindXY(xV, yV);
    tick += 1;

    lblUtlity.Text = "Utilization\n" + (int)util;
}
}
}

```