

Mateusz Kumpf

nr albumu: *****

kierunek studiów: Informatyka

specjalność: Systemy komputerowe i oprogramowanie

forma studiów: *stacjonarne*

Opracowanie aplikacji umożliwiającej monitorowanie danych o wydajności i obciążeniu stacji roboczych na centralnym serwerze.

An application that monitors workstations' performance and workload on a central server.

praca dyplomowa inżynierska

napisana pod kierunkiem:

dr hab. inż. Imed El Fray

Katedra Inżynierii Oprogramowania

Data wydania tematu pracy: 11.01.2013

Data złożenia pracy: 31.05.2016

Szczecin, 2016

OŚWIADCZENIE AUTORA PRACY DYPLOMOWEJ

Oświadczam, że praca dyplomowa inżynierska pn.

.....
.....
(temat pracy dyplomowej)

napisana pod kierunkiem:

.....
(tytuł lub stopień naukowy imię i nazwisko opiekuna pracy)

jest w całości moim samodzielnym autorskim opracowaniem sporządzonym przy wykorzystaniu wykazanej w pracy literatury przedmiotu i materiałów źródłowych.

Złożona w dziekanacie **Wydziału Informatyki**
(wydział)

treść mojej pracy dyplomowej w formie elektronicznej jest zgodna z treścią w formie pisemnej.

Oświadczam ponadto, że złożona w dziekanacie praca dyplomowa ani jej fragmenty nie były wcześniej przedmiotem procedur procesu dyplomowania związanych z uzyskaniem tytułu zawodowego w uczelniach wyższych.

.....
podpis dyplomanta

Szczecin, dn.

Abstract

Main purpose of this thesis is creating a library to monitor performance and workload of workstations. Two applications working in client-server architecture will be created to illustrate potential of said library.

First chapter is introduction to this thesis.

In second chapter range of tested environment and terminology will be presented.

Third chapter describes factors influencing general performance of workstations.

Fourth chapter is dedicated to methods of measuring performance in computer systems.

Fifth chapter presents third party solutions for measuring performance of workstations, on the basis of their flaws in their functioning original project will be made.

Sixth chapter is split into two sub-chapters, first elaborates on the project of system to monitor performance and workload, while second explains its implementation.

Seventh chapter is a summary of whole thesis.

Spis treści

1. Wstęp.....	6
2. Środowisko oceny efektywności pracy stacji komputerowej.....	8
2.1. Zakres i ograniczenia testowanego środowiska.....	8
2.2. Terminologia	8
2.2.1. Mtops.....	8
2.2.2. MIPS.....	8
2.2.3. FLOPS	9
2.2.4. Wydajność	9
2.2.5. Obciążenie	9
2.2.6. Sprawność oprogramowania	9
2.2.6. Lokalna baza danych	10
3. Czynniki wpływające na wydajność stacji komputerowej	11
3.1. Obciążenie wprowadzane przez system operacyjny	11
3.2. Zakłócenia	11
3.3 Wpływ komponentów na wydajność.....	12
3.3.1. Procesor	12
3.3.2. Dysk twardy	13
3.3.3. Pamięć RAM	13
3.3.4. Karta grafiki	14
4. Metody pomiaru wydajności systemu	15
4.1. Wskaźnik WEI dla systemów Windows	15
4.2. Wzorce dla oceny wydajności.....	15
4.2.1. Wzorce syntetyczne.....	15
4.2.2. Wzorce organizacji SPEC	16
4.2.3. Wzorce organizacji TCP.....	17
5. Testowanie i analiza programów do pomiaru wydajności	19
5.1. Przegląd i wybór oprogramowania do testów	19
5.2. Monitor wydajności oraz liczniki wydajności.....	27
5.3. Problemy z oceną wydajności	28
6. Aplikacja monitorująca obciążenie i wydajność stacji komputerowych.....	30
6.1. Projekt systemu	30
6.1.1. Cel projektu	30
6.1.2. Specyfikacja wymagań.....	30
6.1.3. Funkcjonalność.....	31
6.1.4. Perspektywa zewnętrzna	31
6.1.5. Perspektywa wewnętrzna	34

6.1.6. Model komunikacji zdalnej	40
6.2. Implementacja	44
6.2.1. Założenia ogólne	44
6.2.2. Biblioteka DLL.....	44
6.2.3. Oprogramowanie ClientApp	47
6.2.4. Oprogramowanie AdminApp	50
6.2.5. Instalacja oprogramowania.....	54
6.2.6. Możliwości dalszej rozbudowy	54
6.3. Testy oprogramowania	55
7. Podsumowanie.....	59
Bibliografia.....	60
Aneks.....	61
Zawartość płyty CD.....	61
Spis rysunków	61

1. Wstęp

Głównym zadaniem komputerów jest wspomaganie pracy człowieka, poprzez odciążenie go od żmudnych obliczeń i zadań które można zautomatyzować przy pomocy maszyny. Początkowo były to proste zadania, takie jak edycja tekstu, magazynowanie informacji w postaci elektronicznej, monitorowanie systemów użyteczności publicznej. W latach 80-tych uważano, że możliwości komputerów daleko wybiegają w przyszłość pod względem architektury i zasobów. Sam Bill Gates mówił wówczas, że nie wyobraża sobie programu który będzie potrzebował aż 640kB, czyli całej ówczesnej pamięci RAM. Czas jednak bardzo szybko zrewidował to twierdzenie i pokazał, że nie tylko wymagania wobec zasobów maszyn drastycznie wzrosły, ale również pierwotna architektura komputera PC stała się potężnym hamulcem w ich rozwoju. Przychodzi tu na myśl tak wychwalana zgodność komputerów PC wstecz, która z czasem nałożyła ogromne ograniczenia w rozwoju sprzętu i co za tym idzie systemów operacyjnych i aplikacji. Aby się przed tym bronić, firmy produkujące oprogramowanie zwiększały wydajność swoich aplikacji opierając się na wybiórczych technologiach producentów procesorów, kart graficznych lub poprzez sprawne manipulowanie dostępnymi zasobami komputerów. „Ciężkie” procesy zastąpiono wieloma lekkimi wątkami. Producenci systemów operacyjnych, ze względu na pojawiające się problemy z ograniczeniami fizycznej pamięci operacyjnej, wprowadzili adresowanie logiczne. W przypadku procesorów gdy technologia powiedziała „stop”, aby przyspieszyć wykonywanie programu w ich konstrukcje wprowadzono mechanizmy prognozowania wyników operacji, co pozwoliło rozłożyć wykonywanie kodu na kilka potoków oraz mechanizm inteligentnego buforowania skracając tym samym czas ładowania powtarzających się instrukcji. Innym również cennym dla wydajności dodatkiem było rozszerzenie listy rozkazów o dodatkowe instrukcje realizujące funkcje multimedialne. W tym samym czasie producenci komponentów komputera również nie byli bierni. Aby sprzedać produkt z oczywistych względów ich plan biznesowy zawierał dwa główne punkty: cena i wskaźnik wydajności w benchmarku. Typowym przykładem takiego działania jest wbudowanie niewielkiej pamięci Flash w mechaniczny dysk twardy (dyski hybrydowe). Przy dużej ilości odczytywania tych samych wartości, dane nawet nie są zapisywane na talerzu tylko pobierane bezpośrednio z pamięci Flash, która przy odczycie jest dużo szybsza od ruchu mechanicznego ramienia głowicy. W przypadku kart graficznych pojawiły się autorskie sposoby modelowania brył i pokrywania ich teksturami. Okazało się jednak, że część aplikacji nie korzysta z tych technologii. Przykładem może być CUDA firmy NVidia. Innym stosowanym sposobem zwiększenia prędkości była kompresja tekstur w locie. Dawało to dobry wynik w testach ale negatywnie rzutowało na jakość grafiki. W konsekwencji użytkownik zwiększał rozdzielczość aby otrzymać satysfakcjonującą go jakość, co przyczyniło się do pogorszenia płynności grafiki i powrót do punktu wyjścia czyli otrzymania takiej samej wydajności co na karcie gorzej wypadającej w testach.

Wszystkie te działania które mają „podrasować” produkty wytwarzane przez firmy tworzące oprogramowanie i produkujące sprzęt w konsekwencji nie gwarantują tak samo wydajnej pracy w

każdym środowisku. Jednak prawie na pewno ich produkty dobrze wypadają w testach syntetycznych. Wspomniane testy wykonują operacje dostosowane do konkretnego komponentu, dlatego też częściej do oceny wydajności stosuje się benchmarki aplikacyjne czyli takie które symulują użycie aplikacji przez użytkownika takich jak edytory tekstu czy programy typu CAD. Oba rozwiązania mogą posłużyć do oszacowania konieczności wymiany podzespołów lecz ich wyniki mogą być w dalszym ciągu niewystarczające. W przypadku testów syntetycznych, jak wcześniej wspomniano, ocena może zostać zawyżona w stosunku do prawdziwej wydajności natomiast testy aplikacyjne ograniczają się tylko do paru zadań nie wyczerpując w pełni puli czynności które w systemie może wykonywać użytkownik. Dlatego też aby nie narażać posiadacza komputera na niepotrzebne koszty należałoby zbadać efektywność jego jednostki podczas wykonywania rzeczywistych operacji na stacji roboczej. Dzięki połączeniu wyników z trzech w ten sposób uzyskanych badań istnieje możliwość dokładnego oszacowania czy komputer wymaga modernizacji czy też nie.

Celem niniejszej pracy będzie opracowanie oprogramowania do analizy czynności które w rzeczywistości wykonywane są na stacji roboczej, a dzięki uzyskanym wynikom istnieje możliwość otrzymania informacji czy dana jednostka jest wydajna dla operacji aktualnie na niej wykonywanych.

Praca została podzielona na siedem rozdziałów, przy czym pierwszym z nich jest niniejszy wstęp. Następne rozdziały natomiast poświęcone są kolejno wprowadzeniem terminologii użytej w pracy, dostarczaniu informacji o czynnikach wpływających na wydajność stacji komputerowej, gdzie każda ze składowych, tj. system operacyjny, procesor, pamięć, dysk twardy, karta graficzna, przedstawione zostaną pod kątem ich wpływu na wydajność. Kolejnym zagadnieniem, któremu został poświęcony rozdział czwarty, są metody określania wydajności, poczynając od tych archaicznych, a kończąc na nowych możliwościach. W rozdziale piątym dokonana została analiza istniejących rozwiązań na rynku oprogramowania do testowania wydajności stacji komputerowych. Przedstawione zostaną ich zalety oraz wady. W niniejszej pracy omówione zostanie także zaprojektowanie oprogramowania do monitorowania wydajności i obciążenia, a także zostanie zaprezentowana implementacja powstałej biblioteki oraz aplikacji. Zwieńczeniem pracy będzie rozdział siódmy w którym to podsumowana zostanie praca pod kątem spełnienia postawionych celów.

2. Środowisko oceny efektywności pracy stacji komputerowej

Istotnym elementem przy tworzeniu projektu oprogramowania jest określenie granic systemu komputerowego będącego przedmiotem testów oraz określenie metod stosowanych do realizacji wyznaczonych przez to oprogramowanie zadań. Ważnym elementem jest również przedstawienie pojęć niezbędnych do zrozumienia tematyki oceny efektywności stacji roboczych.

2.1. Zakres i ograniczenia testowanego środowiska

W zakres badanego systemu komputerowego przyjęto następujące założenia:

- Obiektem badań jest fizyczna maszyna z zainstalowanym jednym fizycznym procesorem
- Systemem operacyjnym zainstalowanym na maszynie jest Microsoft Windows w wersji od 7 do wersji 10
- Do komunikacji z maszyną stosowana jest lokalna sieć komputerowa

2.2. Terminologia

Z dziedziną badań wydajności urządzeń komputerowych związanych jest kilka opisanych poniżej pojęć.

2.2.1. Mtops

Mtops (ang. million theoretical operations per second) jest miarą wydajności, dzięki której zliczamy liczbę operacji komputera w czasie jednej sekundy. Jest to jedynie wartość teoretyczna, związane jest to ze zróżnicowaniem długości operacji, nie każda wykonuje się w identycznym czasie.

Jak wynika z dokumentacji Biura Przemysłu i Bezpieczeństwa Departamentu Handlu Stanów Zjednoczonych [1] miara ta była wykorzystywana w latach 90-tych do oceny czy określona stacja robocza mogła być sprzedawana poza granicami Stanów Zjednoczonych. Miało to na celu uniknięcie wykorzystania mocy obliczeniowej tychże urządzeń do między innymi wytworzenia broni nuklearnej.

2.2.2. MIPS

MIPS (ang. millions of instructions per second) jest miarą wydajności, która wskazuje ile milionów instrukcji zostało wykonanych przez komputer w ciągu jednej sekundy. Wskaźnik ten ma za zadanie zmierzyć szybkość działania systemu komputerowego przy pomocy podstawowych instrukcji arytmetycznych opartych głównie na wartościach zmiennie- i całkowitoliczbowych oraz na instrukcjach logicznych.

Poważną wadą tego rozwiązania jest brak uwzględniania różnic pomiędzy architekturami, które mogą wykonać konkretne zadanie w tym samym czasie, lecz z różną ilością instrukcji, w takim wypadku nie ma możliwości łatwego porównania wydajności obu urządzeń.

2.2.3. FLOPS

FLOPS (ang. FLoating-point Operations Per Second) jest jedną z najpopularniejszych miar wydajności, opiera się ją na liczbie zmiennoprzecinkowych operacji wykonanych w ciągu jednej sekundy. Ze względu na przeważnie dużą ilość tychże operacji częściej możemy spotkać się z określeniem MFLOPS, GFLOPS bądź TFLOPS (megaFLOPS, gigaFLOPS, teraFLOPS). Stosuje się również pojęcie FLOP oznaczające zmiennoprzecinkowe operacje (FLoating-point Operations), jako licznik wykonanych operacji zmiennoprzecinkowych. Natomiast po rozszerzeniu pojęcia o sekundy otrzymujemy szybkość wykonywania tych operacji. Pojęcie to często wykorzystywane jest w tematyce HPC (High-performance computing), czyli superkomputerów. Popularna lista TOP500 [2] składające się z pięciuset najszybszych superkomputerów na świecie, jako miarę szybkości stosuje właśnie FLOPS'y.

2.2.4. Wydajność

Jest to szybkość z jaką pracuje komputer. Parametr ten może być obliczony w sposób teoretyczny np. za pomocą wzoru M_{tops} lub korzystając z testów wydajności które naśladują rzeczywistą pracę użytkownika komputera (np. MIPS). Wydajność można również wyliczyć na podstawie kilku kryteriów, które reprezentują różne punkty odniesienia do funkcjonalności systemu.

Inna definicja określa wydajność jako łączna skuteczność systemu komputerowego. Parametr ten określany jest na podstawie odczytów przepustowości, czasu reakcji i dostępności [3].

2.2.5. Obciążenie

Posiłkując się definicją przedstawioną przez serwis TechTarget [4] obciążenie można określić jako pewną ilość operacji którym komputer został poddany w danym okresie czasu. Zwykle na obciążenie składa się pewna ilość uruchomionych aplikacji oraz pewna liczba użytkowników którzy wchodzi w interakcję z tymi aplikacjami.

2.2.6. Sprawność oprogramowania

Sprawność oprogramowania można określić jako koszt zawłaszczonych zasobów sprzętowych i systemowych potrzebnych do realizacji wyznaczonego zadania.

Każde oprogramowanie może być mniej lub bardziej zoptymalizowane pod względem zapotrzebowania na zasoby systemu. Coraz częściej można się spotkać z oprogramowaniem pisanym na kilka systemów operacyjnych. Przykładem mogą być języki skryptowe takie jak: Java, Perl, PHP, Visual Basic itp. Kody tych programów muszą być uruchamiane w środowisku interpretera który przy

każdym uruchomieniu tłumaczy instrukcje na język zrozumiały dla systemu operacyjnego. Zapotrzebowanie na zasoby systemowe takiej konstrukcji jest znacznie większe od potrzeb programu bezpośrednio wykonywalnego, którego kod jest jednorazowo kompilowany na poziome budowy oprogramowania. Innym przykładem jest dostępna na rynku jakość oprogramowania. Zasada obecnie panująca to: oprogramowanie ma być funkcjonalne, zrobione szybko, najtańszym kosztem oraz realizować co najmniej to co konkurencja. Niestety, optymalizacja schodzi na dalszy plan.

2.2.6. Lokalna baza danych

W dalszej części przedstawianej pracy lokalna baza danych będzie rozumiana jako plik zapisany w strukturze XML posiadający interfejs wejścia/wyjścia.

3. Czynniki wpływające na wydajność stacji komputerowej

Zgodnie z wcześniejszą definicją, wydajność komputera można oszacować na podstawie korelacji pomiędzy złożonością powierzonych mu zadań a czasem jego realizacji. Głównym instrumentem do przeprowadzania tego typu testów jest wyspecjalizowane oprogramowanie, które pracuje w środowisku systemu operacyjnego zainstalowanego na testowanej maszynie. Oznacza to, że oszacowanie wydajności systemu komputerowego wiąże się z ustaleniem relacji pomiędzy wydajnością zainstalowanych w komputerze urządzeń (ich czasu reakcji, przepustowości, ilości wykorzystywanych komponentów) a sprawnością zainstalowanego systemu operacyjnego (jego zapotrzebowanie na zasoby systemowe wymagane do realizacji zleconego zadania) i dodatkowego oprogramowania.

3.1. Obciążenie wprowadzane przez system operacyjny

Tematem pracy jest określenie obciążenia dla maszyn z zainstalowanym systemem Windows. Różna generacja tych systemów zorientowana była na oprogramowanie właściwe dla okresu ich wdrażania na rynek konsumencki. Oznacza to, że starsze systemy na nowszym sprzęcie, w niektórych warunkach pracy mogą osiągnąć dużo mniejszą sprawność niż systemy nowe. Z drugiej strony, obciążenie nakładane na sprzęt dla systemów starszych może być dużo mniejsze. Jako przykład można wskazać rzadziej używany w dzisiejszych czasach system 32-bitowy zainstalowany na komputerze posiadającym 64-bitowy procesor. Prowadzi to do „sztucznego” obciążenia którego skutkiem jest między innymi [5]:

- ograniczenie instrukcji procesora do wykorzystywania 32-bitowych rejestrów przez co funkcje matematyczne są ograniczone pod kątem zakresu oraz precyzji
- zredukowana ilość pamięci operacyjnej do około 4GB
- uruchamiane aplikacje muszą być utworzone pod kątem 32-bitowego systemu operacyjnego, 64-bitowe aplikacje nie zostaną uruchomione poprawnie

3.2. Zakłócenia

Zdalna kontrola obciążenia stacji komputerowej narażona jest na szereg pułapek, często związanych z oprogramowaniem i niepowiązanych z warstwą sprzętową stacji. Podczas odczytu liczników wydajności poszczególnych komponentów oprogramowanie serwera generuje statystykę. Na tej podstawie operator oprogramowania podejmuje decyzję o fizycznej wydolności maszyny. Jednak nie zawsze winę za złe wyniki ponosi starzejący się sprzęt. Powodem problemów z wydolnością maszyny może być załadowane do pamięci złośliwe oprogramowanie lub błędnie napisany program, w efekcie powodują one zawłaszczenie zasobów systemu i generują obciążenie.

3.3 Wpływ komponentów na wydajność

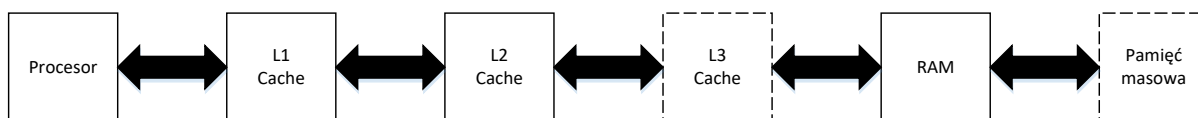
W warstwie sprzętowej stacji komputerowej znajduje się wiele komponentów które mogą wpływać na wydajność komputera. Z tego zbioru można wyszczególnić natomiast cztery których praca zaważa na uzyskanym wyniku wydajności.

3.3.1. Procesor

Procesor jest nieodłączną częścią każdej stacji komputerowej. W dzisiejszych czasach na rynku znajduje się trudna do zliczenia liczba procesorów używanych w rozwiązaniach stacjonarnych, przenośnych bądź serwerowych. Każdy procesor natomiast posiada trzy podstawowe parametry które w dużym stopniu wpływają na jego wydajność [6].

Pierwszym parametrem będzie szybkość taktowania, jest to miara mierzona w hercach, miara ta pod względem teoretycznym wskazuje na ilość wykonywanych instrukcji w ciągu sekundy. W dzisiejszych czasach najczęściej możemy spotkać się z miarą gigaherc, wskazywać ona może na wykonywanie przez procesor miliarda instrukcji w ciągu sekundy, ze względu jednak na to że instrukcje są wykonywane w różnym czasie to wartość ta może odbiegać od rzeczywistej ilości instrukcji.

Drugim parametrem będzie ilość pamięci podręcznej, tak zwany „cache”. Jest to pamięć w której procesor przechowuje dane niezbędne do wykonania operacji. Dane często pobierane są z dysku twardego bądź z pamięci RAM, a następnie trafiają do pamięci procesora która posiada znacznie lepsze parametry co przyczynia się do szybszego dostępu do tychże danych. W dzisiejszych czasach możemy spotkać dwu- lub trzypoziomowe pamięci podręczne (L1, L2, L3), gdzie pamięć oznaczona L1 jest pamięcią najszybszą i zazwyczaj najmniejszą pod względem pojemności (parędziesiąt kilobajtów) natomiast L3 pamięcią najwolniejszą w tym zbiorze natomiast najpojemniejszą (nawet do kilkudziesięciu megabajtów, w przypadku braku poziomu L3 jego rolę przejmuje poziom L2). Poniższy diagram przedstawia przykładową ścieżkę wymiany informacji pomiędzy pamięcią masową (to na niej pierwotnie przechowywane są dane do przetworzenia) oraz procesorem. Linia przerywaną zaznaczono elementy które w dzisiejszych czasach nie zawsze muszą brać udział w wymianie danych.



Rysunek 3.1. Przepływ informacji z pamięci masowej do procesora
Źródło: opracowanie własne

Ostatnim parametrem jest ilość rdzeni. Szybkość taktowania procesora ma swoje ograniczenia, głównym ograniczeniem jest szybkość dzisiejszych tranzystorów. Aby znieść ograniczenia producenci procesorów zdecydowali się wszczepiać w chip procesora większą ilość rdzeni. Rozwiązanie to daje możliwość wykonywania wielu operacji jednocześnie.

3.3.2. Dysk twardy

Dysk twardy jest miejscem w którym można przechowywać wszelkie dane, od zainstalowanych aplikacji po dokumenty, zdjęcia czy filmy. Na wydajność komputera nie ma wpływu pojemność dysku lecz szybkość wykonywania operacji zapisu i odczytu.

Dla tych bardziej popularnych urządzeń, czyli dysków talerzowych (zwane też dyskami magnetycznymi), producenci wskazują najczęściej prędkość z jaką poruszają się znajdujące się w środku talerze (obroty na minutę). Nim wyższa ta wartość tym urządzenia są szybsze. Najpopularniejsze na rynku są urządzenia posiadające prędkości 5400 oraz 7200 obrotów na minutę natomiast te bardziej profesjonalne posiadają nawet 10000 i więcej [6].

Na rynku pojawiły się również szybsze dyski pozbawione mechanicznych elementów zwane dyskami SSD (Solid State Drive). Dla tychże dysków producenci podają maksymalne prędkości odczytu i zapisu (wyrażaną w megabajtach na sekundę) oraz odczyt i zapis losowy. Te ostatnie wyrażane są w jednostce IOPS czyli ilość operacji wejścia/wyjścia w ciągu jednej sekundy. Dla porównania dysków talerzowych z nowocześniejszymi dyskami SSD można wskazać że wynik tego pierwszego przeciętnie plasuje się w granicach ~50-250 IOPS gdzie dyski SSD osiągają wynik nawet 1000 razy większy.

Podsumowując, ogólna wydajność stacji komputerowej zależy od zastosowanego rozwiązania, dysku talerzowego bądź dysku SSD.

3.3.3. Pamięć RAM

W pierwszej kolejności warto wskazać że ilość pamięci odgrywa dużą rolę w przypadku wydajności systemu komputerowego. Systemy wykorzystywane w dzisiejszych czasach wymagają co najmniej 1-2 GB pamięci RAM, jeżeli wartość ta jest niższa to istnieje możliwość że system się nie uruchomi. Ze względu na to że użytkownik nie korzysta jedynie z systemu operacyjnego ale również z aplikacji zainstalowanych w tymże systemie to zapotrzebowanie na pamięć rośnie. Jeżeli korzystamy z programów wymagających dużej ilości danych (np. obróbka wideo) można doprowadzić do przepełnienia pamięci RAM i wymuszenie do przechowywania tychże danych na dysku twardym który posiada znacznie większy czas dostępu co powoduje spadek wydajności [7]. Użytkownicy którzy korzystają z komputera jedynie do edycji plików tekstowych czy też korzystają z przeglądarek internetowych nie wymagają dużej ilości pamięci, mimo to można pokusić się o stwierdzenie że nim więcej pamięci tym stacja jest wydajniejsza, ponieważ wymagania użytkownika mogą zmienić się w każdej chwili.

Równie dużą rolę odgrywa tutaj taktowanie pamięci oraz opóźnienia CL (CAS Latency). Nim wyższe taktowanie pamięci i mniejsze opóźnienia tym komponent ten jest szybszy. Różnice widoczne są szczególnie w przypadku wymagających aplikacji jakimi są np. gry komputerowe, przy ich

uruchomieniu przechowywane są duże ilości danych do późniejszego przetworzenia, nim lepszy dostęp do danych tym gra będzie działać płynniej.

3.3.4. Karta grafiki

W dzisiejszych czasach karty graficzne dzielą się na dwie kategorie:

- karty zintegrowane
- karty dedykowane

Podstawową różnicą, poza wydajnością tychże kart (karty zintegrowane z procesorem są stosunkowo mniej wydajne od dedykowanych) jest pamięć z której te karty korzystają. Karta dedykowana, jako osobny układ posiada własną pamięć natomiast karta zintegrowana korzysta z pamięci operacyjnej przez co istnieje możliwość zagarnięcia przez nią zasobów niezbędnych do obsługi aplikacji wymagającej dużej ilości pamięci RAM. Karty zintegrowane sprawdzą się przy pracach biurowych (np. obróbka plików tekstowych) natomiast przy bardziej wymagających aplikacjach, jakimi są np. gry komputerowe czy programy typu CAD lepiej sprawdzają się karty dedykowane.


W przypadku kart dedykowanych liczy się również ilość pamięci wbudowanej, im jej więcej tym większa ilość danych może zostać przetworzona. Istotnym jest również rodzaj pamięci. Te najbardziej popularne to GDDR3 (wolniejsze) oraz GDDR5 (szybsze), w zależności od wybranej technologii wydajność samego urządzenia może wzrosnąć.

Kolejnym dosyć istotnym elementem jest zgodność z bibliotekami graficznymi takimi jak OpenGL czy DirectX. W zależności od wspieranej wersji aplikacje wykorzystujące te biblioteki mogą działać mniej bądź bardziej wydajnie.

4. Metody pomiaru wydajności systemu

4.1. Wskaźnik WEI dla systemów Windows

Od systemu Windows Vista firma Microsoft wbudowała prosty mechanizm nazwany Windows Experience Index (indeks wydajności systemu Windows) i służy do określania wydajności komponentów komputera. Mechanizm składa się ze zbioru testów dzięki którym otrzymujemy wyniki wydajności procesora, karty graficznej, pamięci RAM czy dysku twardego. Najmniejsza wynik określony na podstawie tychże testów staje się „wynikiem podstawowym” który określa wydajność komputera. Nie jest to może sprawiedliwy wynik lecz sam wskaźnik daje nam możliwość wglądu również w wyniki cząstkowe, czyli te osiągnięte przez każdy komponent przez co łatwo możemy określić czy któryś z nich spowalnia pracę całego systemu komputerowego [8]. Poniżej przedstawiono przykładowy wynik uzyskany przy pomocy wskaźnika WEI.

Składnik	Przedmiot klasyfikacji	Wynik cząstkowy	Wynik podstawowy
Procesor:	Obliczenia na sekundę	6,9	 Zależy od najniższego wyniku cząstkowego
Pamięć (RAM):	Operacje pamięci na sekundę	7,4	
Grafika:	Wydajność pulpitu dla interfejsu Windows Aero	5,8	
Grafika w grach:	Wydajność trójwymiarowej grafiki biznesowej i w grach	6,2	
Podstawowy dysk twardy:	Szybkość transferu danych dla dysku	7,9	

*Rysunek 4.1. Przykładowy wynik uzyskany przy pomocy wskaźnika WEI
Źródło: opracowanie własne*

4.2. Wzorce dla oceny wydajności

4.2.1. Wzorce syntetyczne

W latach 70-tych i 80-tych XX wieku inżynierowie zaczęli poszukiwanie programu wzorcowego który umożliwi im pomiar wydajności komputera rzetelnie i sprawiedliwie bez względu na organizację oraz architekturę dowolnego typu systemu. Program ten miał stać się idealnym wskaźnikiem wydajności ze względu na możliwość porównania wielu różnych systemów. Założeniem było aby program ten został napisany w języku C, skompilowany oraz uruchomiony w różnych systemach na których to mierzono czas jego kolejnych wykonywań. Czas który zostałby uzyskany pozwoliłby na pozyskanie pojedynczej wartości wskaźnika wydajności, powiązanego ze wszystkimi testowanymi systemami. Wskaźniki które zostały uzyskane w ten sposób nazywane są „syntetycznymi programami wzorcowymi”, przydomek jest związany z tym że nie reprezentują konkretnego obciążenia

lub zastosowania. Najpopularniejszymi syntetycznymi programami wzorcowymi są Whetstone, Linpack oraz Dhrystone [7].

Program Whetstone opiera się głównie na obliczeniach zmiennoprzecinkowych oraz w około 50% korzysta z funkcji bibliotek matematycznych, w dużej mierze z funkcji trygonometrycznych i wykładniczych. Dzięki zastosowaniu wielu zmiennych globalnych nie ujawnia zalet architektur takich jak RISC w których to wiele rejestrów procesora polepsza obsługę zmiennych lokalnych. Wyniki które uzyskujemy dzięki temu programowi wzorcowemu wyrażone są przy użyciu wskaźników KWIPS (Kilo-Whetstone Instructions Per Second) bądź MWIPS (Mega-Whetstone Instructions Per Second).

Kolejnym programem który dzięki operacjom zmiennoprzecinkowych ocenia wydajność jest Linpack, jest to skrót od LINear algebra PACKage. Sam program jest zbiorem podprocedur nazywanych BLAS (Basic Linear Algebra Subroutines), zbiór ten rozwiązuje równania liniowe oparte na operacjach arytmetycznych o podwójnej precyzji. Dzięki takiemu podejściu nie wykorzystuje funkcji matematycznych w przeciwieństwie do programu Whetstone. Jego głównym zastosowaniem miało być mierzenie wydajności superkomputerów. Nie korzysta on ze zmiennych globalnych, wszystkie operacje przeprowadzane są na zmiennych lokalnych, a tablice przekazywane są do funkcji jako parametry. Największym atutem tego programu jest zastosowanie standardowego wskaźnika FLOPS. Dzięki programowi wzorcowemu Linpack możliwe jest uzyskanie wydajności we FLOPSach w systemach pozbawionych operacji zmiennoprzecinkowych.

W przeciwieństwie do poprzednio opisywanych, program Dhrystone nie skupia się na operacjach na liczbach zmiennoprzecinkowych lecz na przetwarzaniu łańcuchów oraz operacjach na liczbach całkowitych. Jego twórca stwierdził że operacje zmiennoprzecinkowe nie są najważniejsze dla wszystkich użytkowników komputerów. Program ten obciąża procesor lecz nie wykonuje żadnych operacji wejścia-wyjścia lub wywołań systemowych. Jednostką w jakiej obliczany jest wydajność opisujemy jako Dhrystone'y na sekundę czyli ilość możliwych wykonań tegoż programu w ciągu sekundy. W przeciwieństwie do wskaźnika WIPS nie występują określenia takie jak DIPS czy MDIPS (MegaDIPS).

4.2.2. Wzorce organizacji SPEC

Nie da się zaprzeczyć że programy wzorcowe takie jak Whetstone, Linpack czy Dhrystone odegrały bardzo dużą rolę w dziedzinie mierzenia wydajności i porównywania systemów. Ze względu na swoją prostotę programy te posiadały pewne wady, szczególnie porównywanie dwóch różnych systemów różnych producentów, wyniki które otrzymywaliśmy nie dawały stuprocentowej pewności co do przewagi jednego bądź drugiego systemu. Należało stworzyć program bądź programy bardziej złożone które w rzetelny sposób pozwolą na określenie wydajności systemu i porównanie go z innym. I taką właśnie rolę spełniać miały programy wzorcowe organizacji SPEC.

Organizacja SPEC [7], z angielskiego Standard Performance Evaluation Corporation, założona została w 1988 r. przez grupę producentów komputerów we współpracy z grupą Electrical Engineering Times. Obecnie główny trzon organizacji tworzy 50 firm, istnieją również podmioty wspierające ową instytucję, na które składa się 80 organizacji, głównie uczelni. Utworzyła ona benchmarki do wielu elementów systemów komputerowych:

1. CPU
2. Grafika/Stacje robocze
3. ACCEL/MPI/OMP
4. Java Klient/Serwer
5. Serwery poczty
6. Sieciowe systemy plików
7. Zasilanie
8. SIP
9. SOA
10. Wirtualizacja
11. Serwery stron WWW

Ze względu na jedynie płatne wersje powyższe benchmarki nie pojawiają się w testach.

4.2.3. Wzorce organizacji TCP

Wcześniej wspomniana organizacja SPEC w pierwszych latach swojego istnienia skupiała się głównie na mierzeniu wydajności procesora, co przekładało się na skupianie się jedynie na stacjach roboczych. Były i są nadal firmy dla których ważniejszą rolę odgrywają serwery do przetwarzania transakcji, przy tych rozwiązaniach przedsiębiorcę interesuje głównie szybkość przetwarzania dużych ilości jednocześnie wykonywanych operacji. Każda taka operacja wymaga nawiązania połączenia i wykonania żądania wejścia/wyjścia na dysku twardym.

Każdy z producentów posiada swoje aplikacje do pomiarów wydajności systemów które produkują. Aplikacje te pomagają inżynierom zwiększać wydajność tworzonych przez nich sprzętów lecz są bezużyteczne w przypadku uruchomienia ich na sprzęcie konkurencyjnym. Firmą która stworzyła pierwszy program wzorcowy wspomagający projektowanie systemów przemysłowych było IBM Corporation. Na początku lat 80. XX wieku stworzyła program o nazwie TP1 (ang. TP – transaction processing), który finalnie stał się publicznie dostępny. Program ten nie był jednak doskonały ponieważ w symulacjach pomijał kwestie opóźnień sieciowych czy reakcji użytkowników. Wynikiem pomiarów była szczytowa przepustowość systemu przy idealnych warunkach, wynik przydatny dla projektantów systemów natomiast mało użyteczny dla potencjalnych nabywców.

Program wzorcowy który stał się następcą TP1 był DebitCredit. Nad jego stworzeniem pracowano już od 1970 roku natomiast dopiero w 1985 roku pokazano go światu. Pierwotnie pomagał

określać wydajność systemów komputerowych wykonujących transakcje dla Bank of America (największy bank w Stanach Zjednoczonych). Twórcami benchmarku był Jim Grey oraz jego zespół którzy nie tylko opisali całe narzędzie ale również zaproponowali aby uzyskane wyniki były dostarczane wraz z pełną konfiguracją co miało zapewnić rzetelność otrzymanych wyników w porównaniu z innymi tego typu systemami. Program został bardzo dobrze przyjęty przez inżynierów i nabywców, związane było to również z brakiem ciekawej na rynku alternatywy. Producenci natomiast mogli podawać wyniki które miały zapewnić lepszą sprzedaż ich rozwiązania co skłoniło parę firm komputerowych do założenia organizacji która pozwoliłaby na kontrolę i analizę otrzymanych wyników. W ten właśnie sposób powstała organizacja TPC [7] która na dzień dzisiejszy liczy ponad 40 różnych firm z branży oprogramowania oraz sprzętu.

Po powstaniu organizacji w roku 1988 należało stworzyć pierwszy zestaw programów wzorcowych zatwierdzonych przez tą organizację. Taki zestaw powstał na przełomie roku 1989 i 1990 i nazwany został TPC-A. Do dnia dzisiejszego powstało wiele innych zestawów takich jak TPC-B, TPC-H, TPC-R, TPC-W, natomiast najważniejszym z nich jest TPC-C. Jest to zestaw najbardziej złożony który oferuje symulowaniu pięciu rodzajów transakcji wykonywanych w tym samym momencie. Wynik benchmarku TPC-C mierzony jest w transakcjach na minutę (TPM). W skład obsługiwanych transakcji wchodzi:

1. Nowe zamówienie
2. Realizacja zamówienia
3. Zaksięgowanie płatności klienta
4. Pobieranie raportu stanu zamówienia
5. Monitorowanie stanu magazynowego zamówionego produktu

Na głównej stronie organizacji możemy znaleźć wyniki uzyskiwane w konkretnych zestawach programów wzorcowych z wyszczególnieniem między innymi specyfikacji, kosztów nabycia sprzętu czy wykorzystywanej bazy danych.

5. Testowanie i analiza programów do pomiaru wydajności

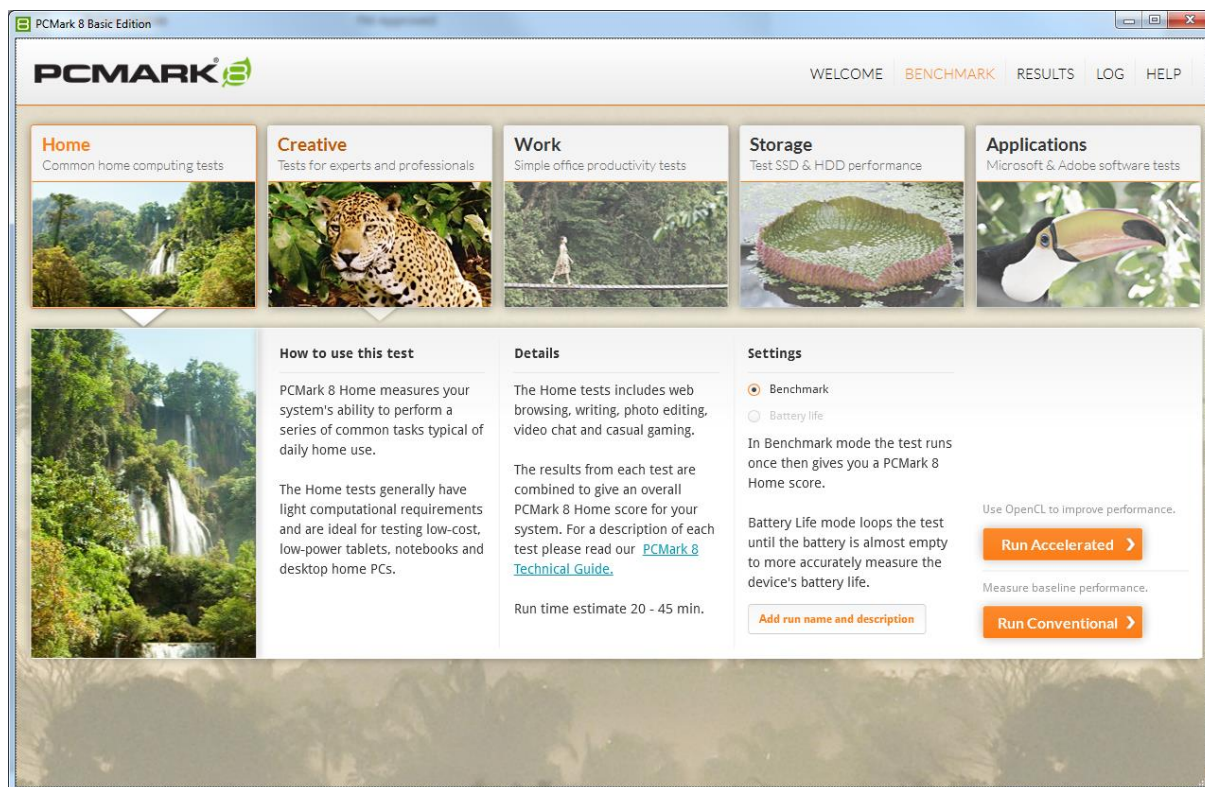
5.1. Przegląd i wybór oprogramowania do testów

W pierwszej kolejności należy przedstawić parametry stacji roboczej na której aplikacje zostały przetestowane. Wskazana stacja była wyposażona w:

- Procesor Intel Core i5-i5-3550 (4 rdzenie oraz 4 wątki)
- Karta graficzna NVIDIA GeForce GTX 660
- 8GB pamięci RAM (kości 2 x 4GB)
- System operacyjny Windows 10 Pro 64-bitowy

Pierwszym programem wybranym do pomiaru wydajności jest PCMark 8 Basic Edition [9]. Aplikacja jest kolejną odsłoną znanej serii benchmarków firmy Futuremark cieszących się dużą popularnością, szczególnie w środowisku recenzentów nowych urządzeń wchodzących na rynek. Wskazana wersja jest dostępna za darmo na stronie producenta, posiada ograniczoną funkcjonalność w stosunku do edycji płatnej (brak dostępnych wszystkich rodzajów testów) lecz na potrzeby testów są to funkcje wystarczające. Aplikacja składa się z trzech rodzajów benchmarków:

- Home test – zawiera testy oparte o najczęściej wykonywane w domu operacje, takie jak:
 - Obsługa przeglądarki (zakupy online, portal typu social media)
 - Edycja dokumentów tekstowych
 - Proste gry 3D
 - Rozmowy wideo z użyciem kamery internetowej
 - Edycja plików graficznych
- Work test – zawiera w sobie testy symulujące pracę biurową, w jego skład wchodzi:
 - Obsługa przeglądarki
 - Edycja dokumentów tekstowych
 - Rozmowy wideo z użyciem kamery internetowej
 - Edycja arkusza kalkulacyjnego
- Creative test – najbardziej rozbudowany szereg testów zawierający:
 - Edycja plików video - skalowanie obrazów (z 4K do 720p czy z 1080p do 720p), zmiana przepływności (bitrate)
 - Edycja plików audio
 - Rozbudowane gry 3D
 - Rozmowy wideo z użyciem kamery internetowej
 - Edycja plików graficznych
 - Obsługa przeglądarki



Rysunek 5.1 Okno główne aplikacji PCMark 8 Basic Edition
Źródło: opracowanie własne

Dla wskazanej wcześniej stacji wykonano po kolei każdy ze zbiorów testów. Wpierw rozpoczęto od Home test. Zakończył się on w ciągu 35 minut i 20 sekund, w tym czasie na ekranie wyświetlały się kolejne okna symulujące działanie użytkownika. Uzyskany wynik zapisywany jest na serwerze firmy i w każdej chwili mamy do niego dostęp, poniższy rysunek prezentuje wynik uzyskany w tymże teście:

PCMARK 8 HOME CONVENTIONAL 3.0



SCORE

3 453

with NVIDIA GeForce GTX 660(1x) and Intel Core i5-3550 Processor

VALID RESULT

Add to compare

Test duration	35 min 20 s	Web Browsing - JunglePin	0.31894 s	Web Browsing - Amazonia	0.12748 s
Writing	5.14805 s	Casual Gaming	82.37 fps	Video Chat playback 1 v2 Conventional	29.99 fps
Video Chat encoding v2 Conventional	74.66667 ms	Advanced Photo Editing1 Conventional	0.43944 s		

Rysunek 5.2 Wynik wykonania Home test
Źródło: opracowanie własne

Jak w większości programów tego typu wynikiem jest liczba symbolizująca wydajność systemu, nim wyższa tym większa wydajność stacji roboczej. Przy tym teście plusem jest możliwość wykonywania innych zadań (np. edycja pliku tekstowego czy obsługa przeglądarki internetowej) ponieważ nie zawłaszcza niewiele zasobów. Spróbowano tego typu działanie wykonać podczas analizy i nie przeszkodziło to w pomyślnym zakończeniu testów. Nie jest to jednak wskazane ponieważ może wpłynąć negatywnie na ostateczny wynik.

Drugim z kolei zbiorem który wykonano był Work test. Wykonane testy zakończyły się w czasie 35 minut i 28 sekund więc o 10 minut dłużej niż zakładał to producent oprogramowania (wskazany przewidywany czas wynosił 15-25 minut). Próba wykonywania innych zadań równocześnie zakończyła się zatrzymaniem się testu i jedyną opcją było jego przerwanie i ponowne uruchomienie. Błąd pojawił się w momencie wykonywania testu związanego z operacjami na arkuszu kalkulacyjnym, przypadkowo prowadzono tekst z klawiatury do jednej z komórek. Dlatego też dla tego zbioru szczególnie niesugerowane jest wykonywanie innych czynności. Poniżej przedstawiono wynik pomiarów wydajności przy pomocy tego zestawu testów:



Rysunek 5.3 Wynik wykonania Work test
Źródło: opracowanie własne

Ostatnim wykonanym zestawem był Creative test. Trwał on dłużej niż pozostałe zbiory, a mianowicie 60 minut i 11 sekund. W tym przypadku jednak nie wszystkie testy zostały zobrazowane w formie okna, tutaj należy wskazać obróbkę plików wideo oraz plików audio przez co użytkownik nie do końca wie jaki jest efekt wykonania testu. Wynik testu został przedstawiony poniżej:

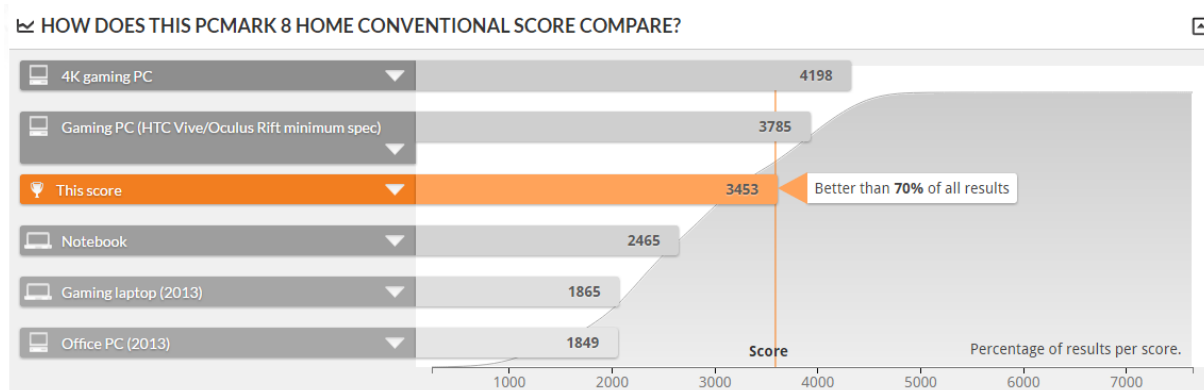


Rysunek 5.4 Wynik wykonania Creative test
Źródło: opracowanie własne

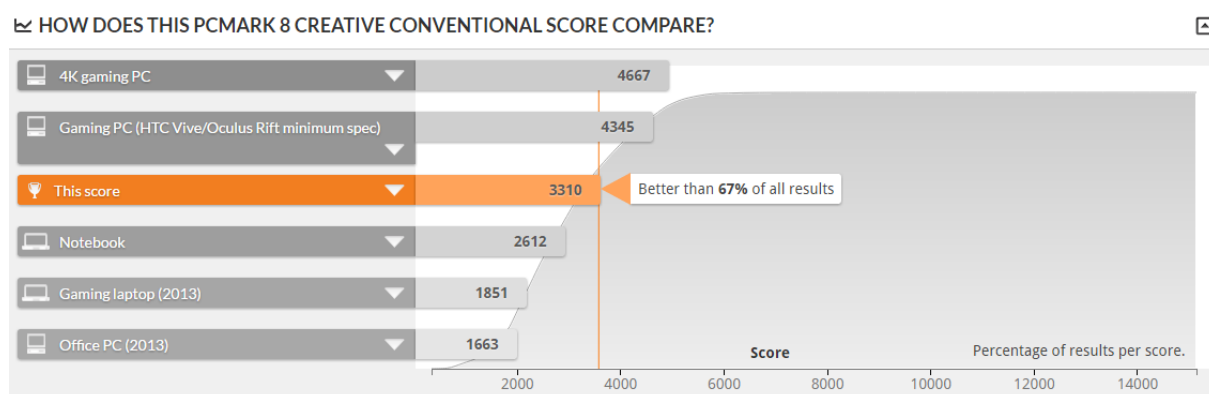
Podczas wykonywania kolejnych testów zasoby stacji roboczej były mocno zagarniane przez co nisko wydajne było wykonywanie własnych czynności (np. obsługa przeglądarki internetowej).

Zestaw Creative test wykonuje mocno obciążające i czasochłonne testy, natomiast Work test operuje jedynie na testach zorientowanych na pracę biurową, a mimo tego wyniki mocno od siebie odbiegają co nasunęło pytanie: czy wynik pomiarów jest błędny? Dalsza analiza uzyskanych wyników dodatkowych szczegółów ze strony na której wyniki się znalazły pozwoliła odpowiedzieć na to pytanie. Każdy zbiór testów powinien być traktowany osobno, mimo niewielkich podobieństw wyniki mogą mocno od siebie odbiegać. Udowodniono to dzięki wykresom porównującym stację testową z wynikami

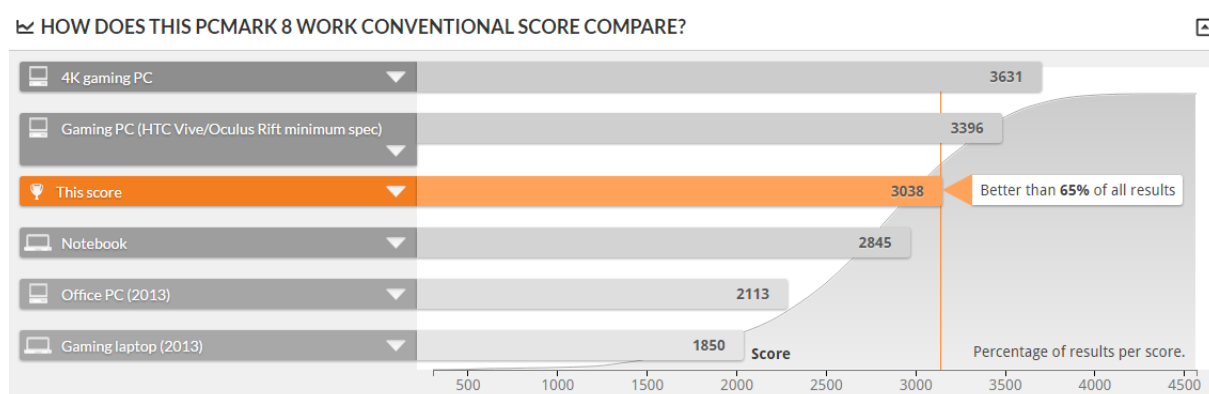
stacji wykonanych przez producenta oprogramowania. Poniżej przedstawione zostaną wykresy dla wszystkich zestawów:



Rysunek 5.5 Wykres porównujący dla Home test
Źródło: opracowanie własne



Rysunek 5.6 Wykres porównujący dla Creative test
Źródło: opracowanie własne

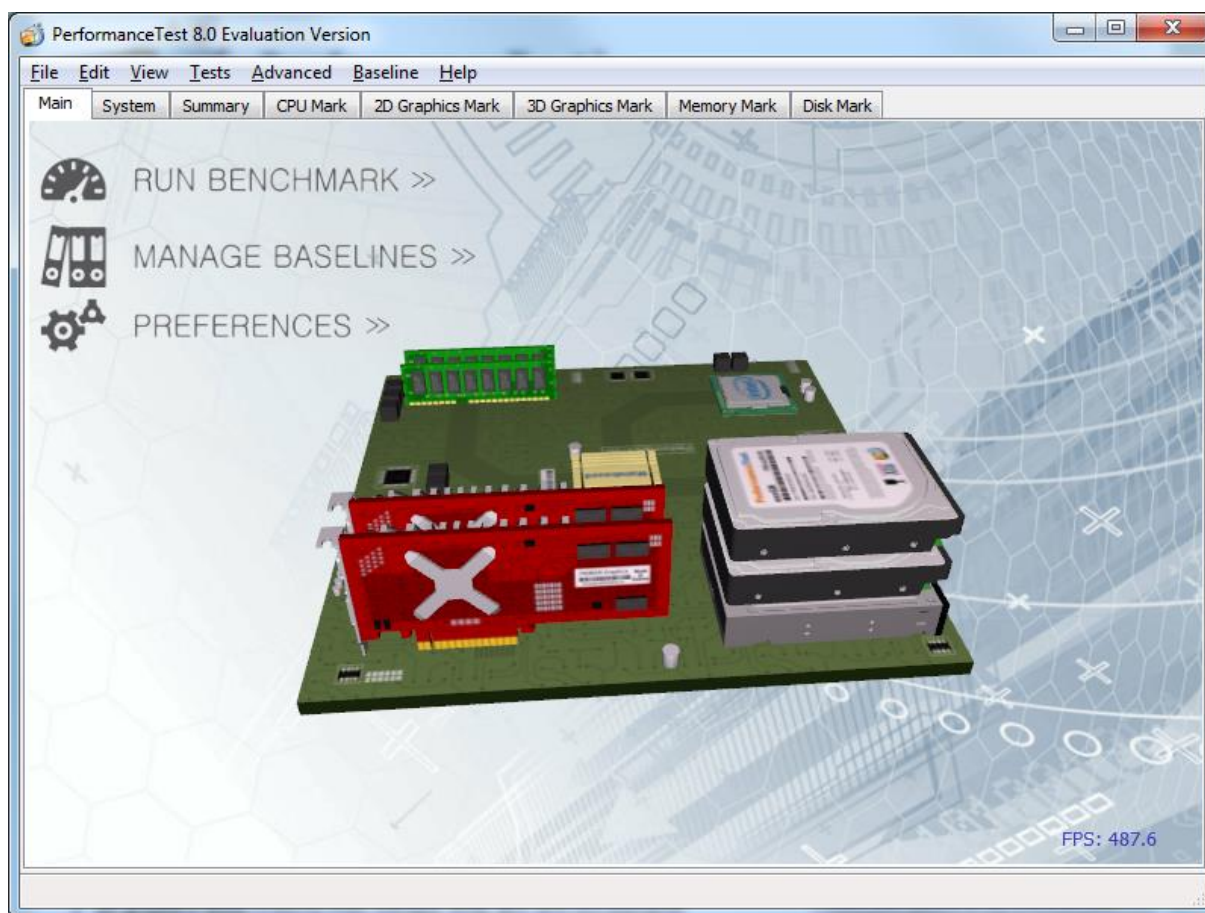


Rysunek 5.7 Wykres porównujący dla Work test
Źródło: opracowanie własne

Wykresy pokazują jednoznacznie że rząd wielkości pomiędzy uzyskanym wynikiem, a najwyższą wynikiem wskazanym przez producenta jest w przybliżeniu taki sam przez co można wysnuć wniosek że do błędu nie doszło. Oznacza to jedynie że każdą ze stacji można oceniać na trzy różne sposoby, a w przypadku porównywania wydajności różnych stacji należy porównać wyniki tych samych zestawów testów.

Kolejną aplikacją którą można znaleźć w jej darmowej wersji jest PerformanceTest 8.0 Evaluation Version [10]. W tym przypadku nie tracimy na wiele na funkcjonalności oprogramowania, jednym z większych ograniczeń jest 30-stodniowy termin obsługi aplikacji.

W przeciwieństwie do poprzedniej aplikacji każdy z testów wchodzący w skład zestawów jest oceniany w formie liczby tak samo jak wynik końcowy co jest treściwsze (w poprzednim programie wyniki cząstkowe były przedstawione w formacie odpowiadającym konkretnemu zadaniu, np. ilości klatek na sekundę).



*Rysunek 5.8 Okno aplikacji PerformanceTest 8.0
Źródło: opracowanie własne*

Poniżej przedstawiona zostanie widoczna na rysunku 5.8 struktura zestawu testów:

- CPU Mark – ocena wydajności procesora, z czego zestaw składa się między innymi z:
 - Operacji matematycznych na liczbach całkowitych, zmiennoprzecinkowych oraz liczbach pierwszych
 - Kompresji danych
 - Szyfrowaniu danych
 - Sortowaniu danych
 - Obsłudze wątków
- 2D Graphics Mark – ocena wydajności dla grafiki 2D, zestaw składa się z następujących testów:

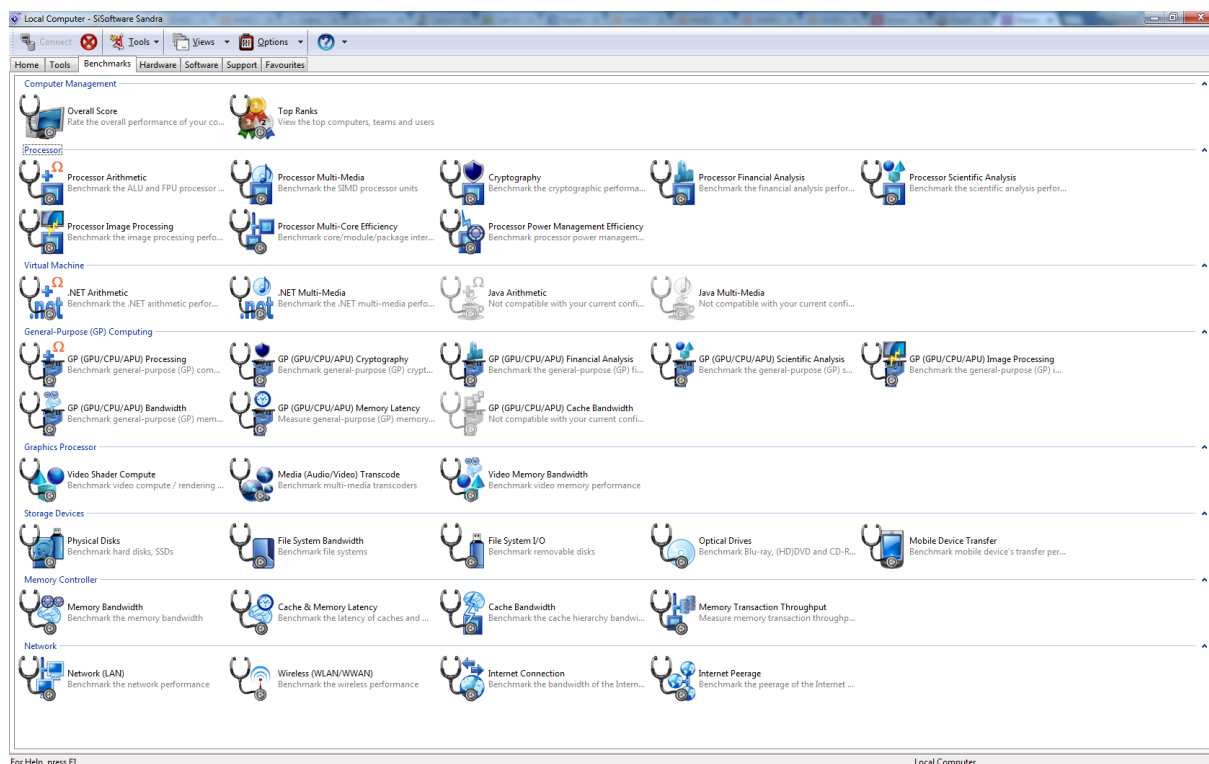
- Operacje na wektorach
 - Operacje na tekście
 - Operacje na oknach
 - Operacje oparte na stosowaniu filtrów dla zdjęciach
 - Rendering zdjęć
 - Obsługa Direct 2D
- 3D Graphics Mark – ocena wydajności dla grafiki 3D, zestaw składa się z następujących testów:
 - Operacje przewidziane dla DirectX począwszy od wersji 9.0 do wersji 11.0
 - Memory Mark – ocena wydajności pamięci, zestaw składa się z następujących testów:
 - Operacje na bazach danych
 - Odczyt oraz zapis do pamięci
 - Dostępność pamięci
 - Opóźnienia w dostępie
 - Obsługa wątków
 - Disk Mark – ocena wydajności dla dysku twardego, zestaw ten składa się z następujących testów:
 - Zapis i odczyt sekwencyjny
 - Losowe wyszukiwanie wraz z zapisem i odczytem



Rysunek 5.9 Podsumowanie wyników dla aplikacji PerformanceTest 8.0
Źródło: opracowanie własne

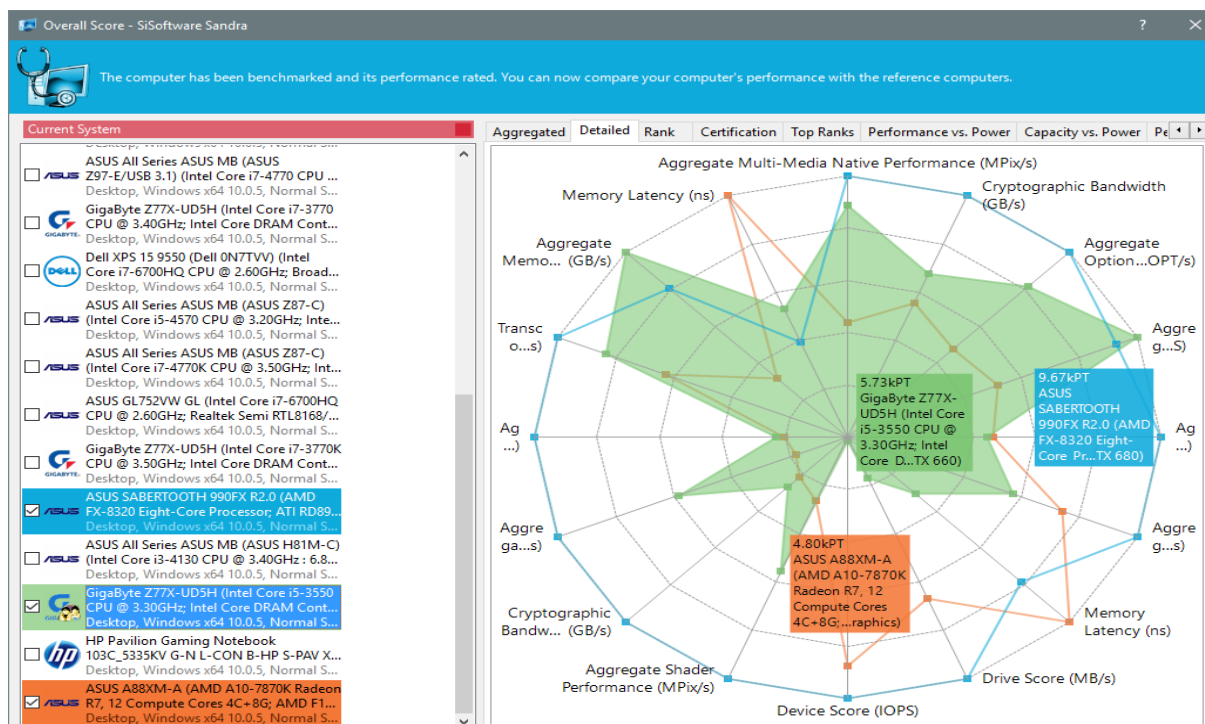
Powyższy rysunek przedstawia zestawienie wyników każdego zestawu wraz z sumarycznym wynikiem. Poza poznaniem wydajności każdego z komponentów istnieje możliwość porównania wyników z innymi modelami dla danej kategorii. Jako kolejny plus należy wskazać szybki czas wykonania wszystkich testów, dla maszyny testowej proces ten trwał około od 3 do 5 minut.

Ostatnią aplikacją silnie rozpowszechnioną jest aplikacja SANDRA od firmy SiSoftware [11], do analizy zostanie wykorzystana jej dostępna wersja oznaczona typem Lite. Sama aplikacja ma szeroki zakres funkcjonalności, nie tylko ocenia wydajność stacji roboczej ale również dostarcza ogólnych informacji o komponentach, nie tylko fizycznych ale również tych systemowych (np. uruchomione procesy, dostępne czcionki, zainstalowane sterowniki, itp.). W naszym przypadku skupimy się jednak na tej pierwszej funkcjonalności.



Rysunek 5.10 Funkcjonalności związane z wydajnością dla aplikacji SANDRA
Źródło: opracowanie własne

Na rysunku 5.10 można zauważyć pierwszą różnicę w stosunku do poprzednio przedstawianych narzędzi. Dodatkowymi elementami pod kątem których można przetestować stację roboczą takimi jak wirtualna maszyna .NET, wydajność karty graficznej dla tych samych zadań co w przypadku procesora bądź wydajność sieci. Dla testowej maszyny proces określania wydajności dla wszystkich możliwych opcji trwał niecałą godzinę. Poniżej przedstawione są wyniki tej operacji:



Rysunek 5.11 Wyniki wydajności dla programu SANDRA
Źródło: opracowanie własne

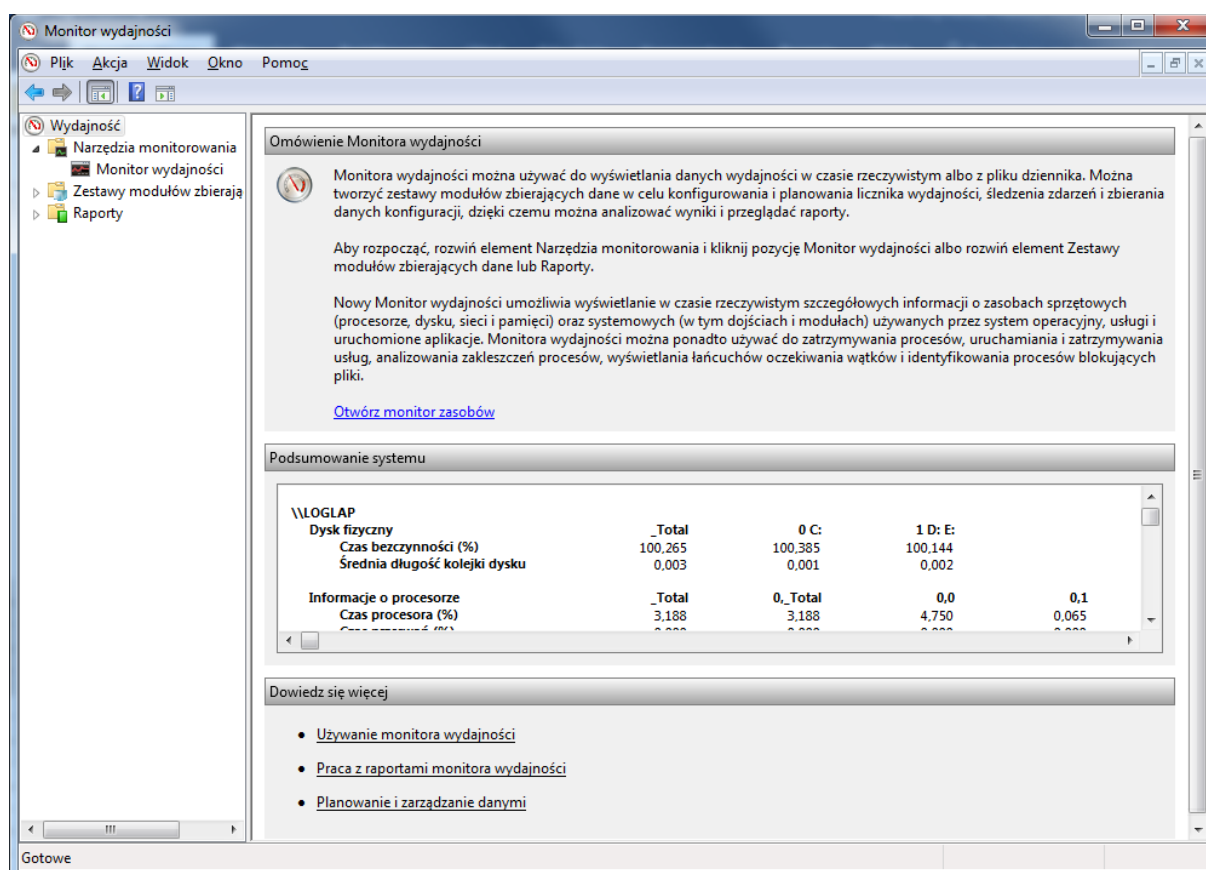
Rysunek 5.11 prezentuje wykres typu radar, zielone pole oznacza wartości uzyskane dla testowej stacji roboczej natomiast punkty w innych kolorach wyniki innych konfiguracji dla celów porównawczych. Istnieje również możliwość dołączenia do wykresu innych stacji przez nas przetestowanych, jest to jednak związane z koniecznością połączenia się do tych maszyn zdalnie przy pomocy tej samej aplikacji, co jednocześnie jest kolejną funkcjonalnością wyróżniającą ją od pozostałych.

Każda z wyżej wymienionych aplikacji posiada swoje własne plusy:

- Aplikacja PCMark 8 Basic Edition posiada intuicyjny interfejs oraz potrafi symulować w pewnym stopniu zachowania użytkownika
- Aplikacja PerformanceTest 8.0 Evaluation Version pozwala rozłożyć ocenę wydajności na najmniejsze składowe np. ocena wydajności procesora w kontekście szybkości wykonywania operacji na liczbach zmiennoprzecinkowych. Dodatkowo można też wskazać krótki czas wykonywania się testów.
- Aplikacja SANDRA posiada mocno rozwinięty interfejs, nie skupia się tylko i wyłącznie na testowaniu wydajności ale również na innych aspektach systemu (np. uruchomione usługi, informacje na temat komponentów). Dostarcza również funkcjonalność zdalnej analizy stacji roboczych, a następnie zestawienie wyników z otrzymanymi z innych urządzeń.

5.2. Monitor wydajności oraz liczniki wydajności

Perfmon czyli tytułowy monitor wydajności w systemach z rodziny Windows jest przystawką programu Microsoft Management Console (MMC). Narzędzie to pozwala na monitorowanie wydajności całej stacji roboczej, w odniesieniu do aplikacji i sprzętu, w czasie rzeczywistym. W tym celu korzysta z liczników wydajności. Zawiera w sobie funkcjonalności narzędzi występujących w systemach starszych od Windows Vista, a mianowicie: dziennik wydajności i alerty, doradca wydajności serwera oraz monitor systemu. Jego struktura nie zmieniała się wraz z kolejnymi wersjami systemu Windows.



Rysunek 5.12 Monitor wydajności – Perfmon
Źródło: opracowanie własne

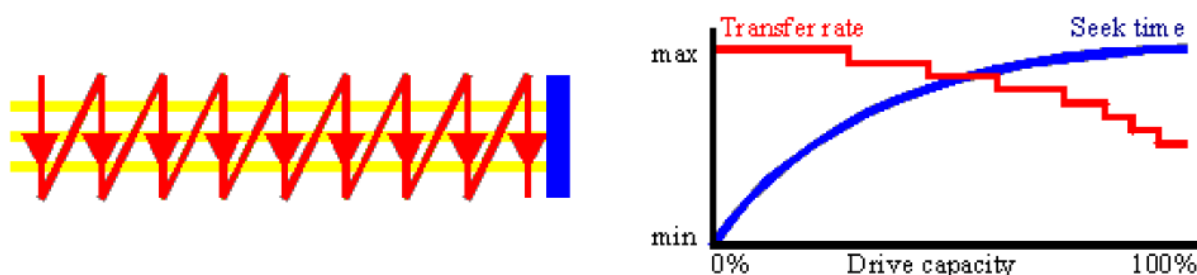
Wspomniane wcześniej narzędzie do określenia wydajności wykorzystuje wbudowane w system liczniki wydajności. W zależności od systemu, zainstalowanych aplikacji i podzespołów ich zestaw może się różnić natomiast istnieją cztery typy które zawsze pojawią się na liście:

- liczniki wydajności procesora
- liczniki wydajności pamięci
- liczniki wydajności dysku twardego (lub dysków)
- liczniki sieci

Każda z grup posiada indywidualne aspekty odpowiadające konkretnemu komponentowi, np. czas procesora, czas dysku czy dostępna pamięć. Szczegółowy opis wskazanych grup można odnaleźć w książce Mark'a T. Edmead'a i Paul'a Hinsberg'a a dokładnie w rozdziale który został udostępniony w serwisie Technet [13].

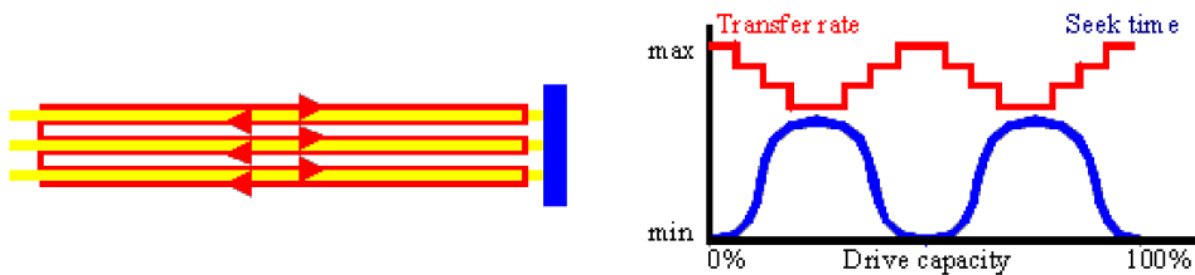
5.3. Problemy z oceną wydajności

Podczas analizy gotowych rozwiązań stwierdzono problemy które można napotkać przy ocenie wydajności. Rzeczywistość pokazuje że otrzymane wyniki w niektórych przypadkach mogą być niewystarczające do prawidłowej oceny wydajności. W pierwszej kolejności należy wskazać że testy wykonywane przez aplikację typu benchmark symulują wykonywane przez użytkownika czynności lecz nie są w stanie wyczerpać puli realizowanych przez niego zadań. Drugi istotny czynnik znajdziemy w pracy Marko Aho oraz Christopher'a Vinckier'a pod tytułem „Computer System Performance Analysis and Benchmarking” [14], jeden z rozdziałów wskazuje możliwość uzyskania przez benchmark wyniku nieoczekiwanego. Jako przykład wybrano dysk twardy, badania wykonane przez autorów wskazują że wydajność może być zależna od zastosowanej metody zapisu danych. W przypadku wykorzystania przez producenta metody pionowego zapisu wraz ze wzrostem zapełnienia dysku rośnie czas wyszukiwania danych oraz maleje szybkość transmisji danych (odczyt oraz zapis).



Rysunek 5.13 Zilustrowanie zapisu pionowego
Źródło: obraz zawarty w dokumencie [12]

Ciekawostką jest jednak wykorzystanie zapisu poziomego, w jego przypadku czas wyszukiwania jest najdłuższy przy zapełnieniu dysku na poziomie 25 i 75 procent, natomiast najkrótszy przy wartościach zbliżonych do 0, 50 oraz 100 procent, tak więc wykresem odzwierciedlającym tą operację będzie sinusoida. Tą samą zależność można zauważyć przy szybkości transmisji danych, z tą różnicą że wszelki wzrost tej wartości jest pozytywnym aspektem w przeciwieństwie do poprzednio prezentowanej operacji.



Rysunek 5.14 Zilustrowanie zapisu poziomego
 Źródło: obraz zawarty w dokumencie [12]

Jednoznacznie pokazuje to że w zależności od zastosowanej metody zapisu oraz czasu w którym zostanie wykonany benchmark istnieje możliwość otrzymania mocno odbiegających od siebie wyników.

6. Aplikacja monitorująca obciążenie i wydajność stacji komputerowych

6.1. Projekt systemu

Sekcja ta zostanie poświęcona wytłumaczeniu działania oprogramowania pod względem teoretycznym. Nie będą poruszane szczegóły, a jedynie zarys budowy aplikacji oraz biblioteki.

6.1.1. Cel projektu

Cel autorskiego projektu można podzielić na trzy elementy z których będzie się on składał. Pierwszym z nich oraz najważniejszym jest utworzenie biblioteki DLL dostarczającej funkcjonalności pozwalające na odczytywanie interesujących nas danych ze stacji roboczej dotyczących jej obciążenia.

Kolejnymi elementami będą aplikacje dzięki którym zostaną zobrazowane możliwości tworzonej biblioteki, w ich skład wchodzi:

- aplikacja zainstalowana na stacjach roboczych dzięki której dane o jej obciążeniu będą magazynowane; w dalszej części pracy aplikacja ta będzie nazywana ClientApp
- aplikacja zainstalowana na stacji administratora, dzięki niej będzie istniała możliwość pobrania danych ze wskazanego klienta, zmianę jego ustawień oraz analizę dotychczas zebranych danych dzięki którym można określić wydajność wskazanej stacji; w dalszej części pracy aplikacja ta będzie nazywana AdminApp

6.1.2. Specyfikacja wymagań

Po przedstawieniu celu projektu należy określić specyfikację wymagań. Poniższe przedstawione są istotne wymogi stawiane aplikacjom ClientApp oraz AdminApp:

1. Pierwszym wymogiem jest aby wyniki pomiarów obciążenia były przedstawione w sposób wiarygodny i odzwierciedlały fizyczny stan obciążenia stacji roboczej.
2. Aplikacja ClientApp musi zostać skonstruowana w taki sposób aby nie zakłócać pracy systemu, a co za tym idzie również wyników odczytanych danych.
3. Dane o obciążeniu magazynowane są w pliku bazodanowym znajdującym się na stacji roboczej, istotnym jest aby katalog wybrany przez użytkownika posiadał uprawnienia odczytu, zapisu oraz modyfikacji znajdujących się w nim plików.
4. Dane zbierane są w czasie rzeczywistym od momentu jej uruchomienia. Jedynym wyjątkiem jest pierwsze uruchomienie przy którym aplikacja jest konfigurowana.
5. Konsola administratora (AdminApp) musi posiadać przyjazny dla użytkownika interfejs aby osoby bez szerokiej wiedzy informatycznej mogły z niego łatwo skorzystać.

6. Aplikacja AdminApp za pośrednictwem protokołu TCP/IP komunikuje się z aplikacją ClientApp zainstalowaną na stacji roboczej przy pomocy utworzonego w ramach biblioteki modelu komunikacji oraz wymienia z nim jedynie istotne dane. Model ten musi zostać stworzony również w taki sposób aby istniała możliwość sprawdzenia czy dane są kompletne oraz czy odpowiadają strukturze którą chcemy otrzymać (np. data, liczba).
7. Aplikacja AdminApp poza odczytem zebranych danych ze stacji roboczej powinna umożliwiać zdalną zmianę parametrów konfiguracji dokonanych podczas pierwszego uruchomienia dla wybranej stacji.

6.1.3. Funkcjonalność

Po wskazaniu najważniejszych wymagań należy określić funkcjonalności które zostaną w późniejszym etapie zaimplementowane. Ze względu na budowę projektu wskazane zostaną funkcjonalności znajdujące się w bibliotece, ponieważ to z nich będą korzystały aplikacje.

Do najważniejszych funkcjonalności wchodzących w skład biblioteki należą:

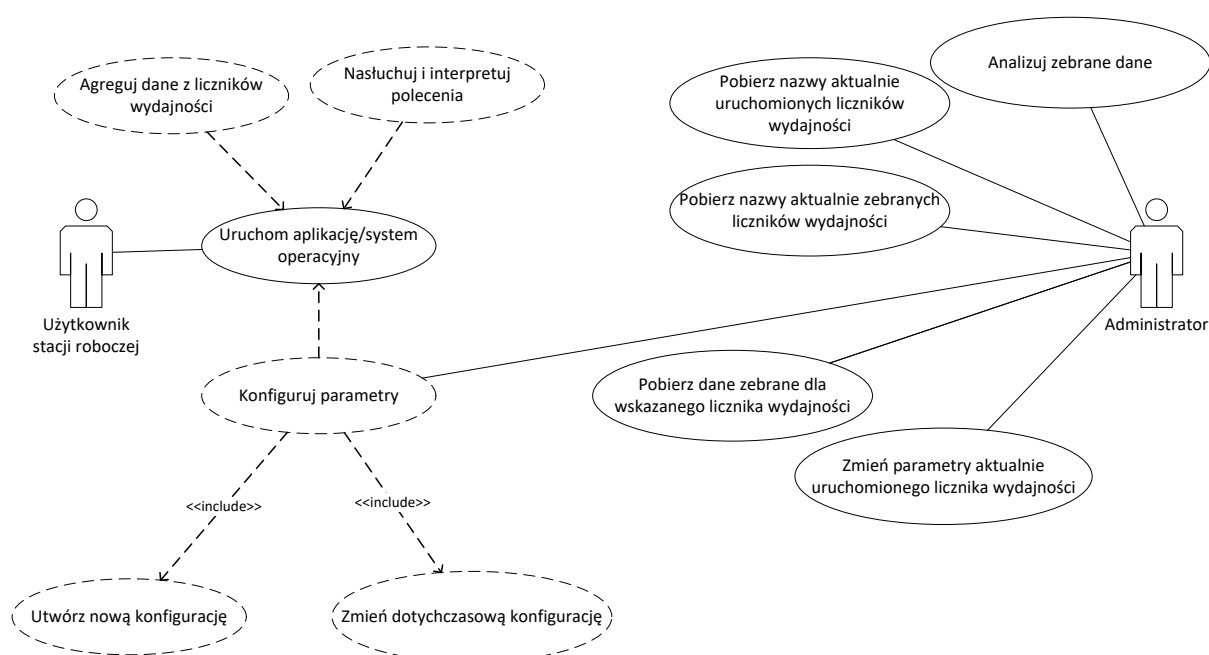
- Odczyt wartości liczników – funkcja ta daje możliwość stałego monitorowania obciążenia konkretnych komponentów przy pomocy liczników wydajności. Otrzymane dane składowane są w pliku tekstowym.
- Dodanie nowych liczników – aby odczyt był możliwy należy w pierwszej kolejności wybrać jakie dane będą zbierane. Dzięki tej funkcji możemy dodać liczników do tejże listy.
- Generowanie komunikatów (poleceń) – głównym celem wskazanej funkcjonalności jest tworzenie komunikatów które następnie zostaną przesłane do aplikacji ClientApp bądź jako odpowiedź zwrotna do aplikacji AdminApp w oparciu o model komunikacji. Model ten zostanie wytłumaczony dalszym podrozdziale.
- Zmiana parametrów licznika – aby spełnić jeden z wymogów projektu należy udostępnić możliwość zmiany parametrów wskazanego licznika.
- Zestawienie wyników – funkcjonalność ta służy do przetworzenia danych zebranych przez aplikację ClientApp z podziałem na dzień pomiaru oraz wybrany licznik.

6.1.4. Perspektywa zewnętrzna

Dzięki perspektywie zewnętrznej istnieje możliwość przedstawienia projektu z widoku użytkownika który będzie korzystał z tworzonych aplikacji. Pozwoli to na zrozumienie jakie możliwości zostaną udostępnione oraz nakreśli granice na etapie implementacji. Do zaprezentowania tejże perspektywy zostanie wykorzystany język UML.

Diagramem który najlepiej opisuje to z jakich funkcji użytkownik może skorzystać jest diagram przypadków użycia. Dla tego typu opisu wykorzystuje się aktorów oraz tytułowe przypadki użycia czyli funkcje które może wykorzystać aktor. W projekcie można wyszczególnić dwóch aktorów:

- Użytkownik stacji roboczej – aktor ten reprezentuje każdego użytkownika pracującego na stacji roboczej z uruchomioną aplikacją ClientApp. Odgrywa bardzo ważną rolę w działaniu aplikacji ponieważ przez wykonywanie różnych czynności powoduje obciążenie stacji, a co za tym idzie możliwość późniejszej analizy zebranych danych.
- Administrator – aktor ten jest osobą obsługującą aplikację AdminApp. Jest osobą która nawiązuje przez swoją aplikację połączenie ze stacjami roboczymi celem np. pobrania danych.



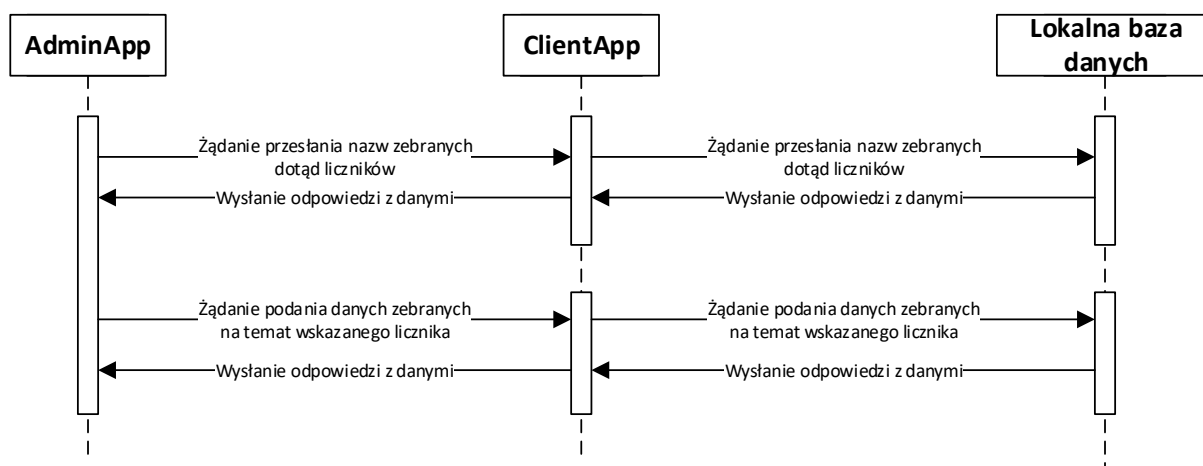
Rysunek 6.1 Diagram przypadków użycia
Źródło: opracowanie własne

Diagram składa się z sześciu głównych przypadków użycia które pokrótce zostaną wyjaśnione poniżej.

- Uruchom aplikację/system operacyjny – jest to czynność wykonywana przez użytkownika stacji roboczej. W zależności od zaleceń administratora aplikację uruchamia użytkownik komputera lub program włączy się wraz ze startem systemu operacyjnego. Pozostałe czynności wykonywane przez aplikację są niezależne od użytkownika i zostały oznaczone w kooperacjach, poniżej zostały one objaśnione:
 - Konfiguruj parametry – dzięki tej czynności istnieje możliwość wprowadzenia konfiguracji dotyczącej agregowania danych. Zawiera ona w sobie możliwość utworzenia nowej konfiguracji (występuje przy pierwszym uruchomieniu

aplikacji) bądź zmianę dotychczasowej konfiguracji. Operacje z tym związane może wykonywać jedynie Administrator.

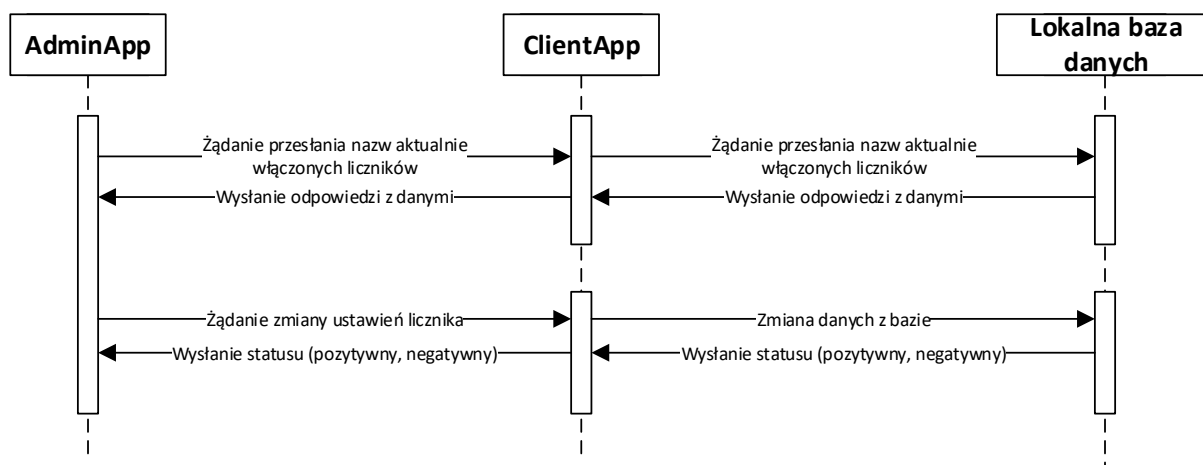
- Agreguj dane z liczników wydajności – dokonuje odczytu danych z liczników wydajności oraz zapisuje je do lokalnej bazy danych.
- Nasłuchuj i interpretuj polecenia – czynność ta zajmuje się obsługą połączeń pomiędzy aplikacją na stacjach roboczych, a konsolą nadzorczą oraz zajmuje się interpretacją tychże poleceń wraz z odesłaniem odpowiedzi.
- Pobierz nazwy aktualnie uruchomionych liczników – funkcjonalność ta jest niezbędna do poprawnej wymiany informacji pomiędzy konsolą administracyjną, a aplikacją kliencką ponieważ znajomość nazw liczników pozwoli na poprawną zmianę jego parametrów.
- Pobierz nazwy aktualnie zebranych liczników wydajności – podobnie jak poprzedni przypadek użycia ten również potrzebny jest do dalszych działań, w tym przypadku do wybrania poprawnego licznika dla którego chcemy odczytać dane.
- Pobierz dane zebranych dla wskazanego licznika wydajności – po wybraniu poprawnego licznika dzięki poprzednio wskazanej funkcjonalności pobierane są dane o jego aktywności z danej stacji roboczej oraz z danego dnia. Dane te niezbędne będą w dalszej analizie. Poniższy diagram ukazuje działanie opisywanego przypadku użycia, gdzie pierwszy człon komunikacji odpowiada za „Pobieranie nazw aktualnie zebranych danych o wskazanym liczniku” natomiast drugi ściśle związany jest z przedstawianym przypadkiem:



Rysunek 6.2 Diagram sekwencji, Pobieranie zebranych danych o wskazanym liczniku
Źródło: opracowanie własne

- Zmień parametry aktualnie uruchomionego licznika wydajności – analogicznie do poprzedniego przypadku tu również wybierany jest licznik dzięki poprzednio pobranym danym, który następnie użyty jest do modyfikacji jego parametrów dla wskazanej stacji roboczej. I tak jak w poprzednim przypadku tu również przedstawiony zostanie diagram,

pierwszy jego człon dotyczy „Pobierania nazw aktualnie uruchomionych liczników” natomiast drugi dotyczy opisywanego przypadku:



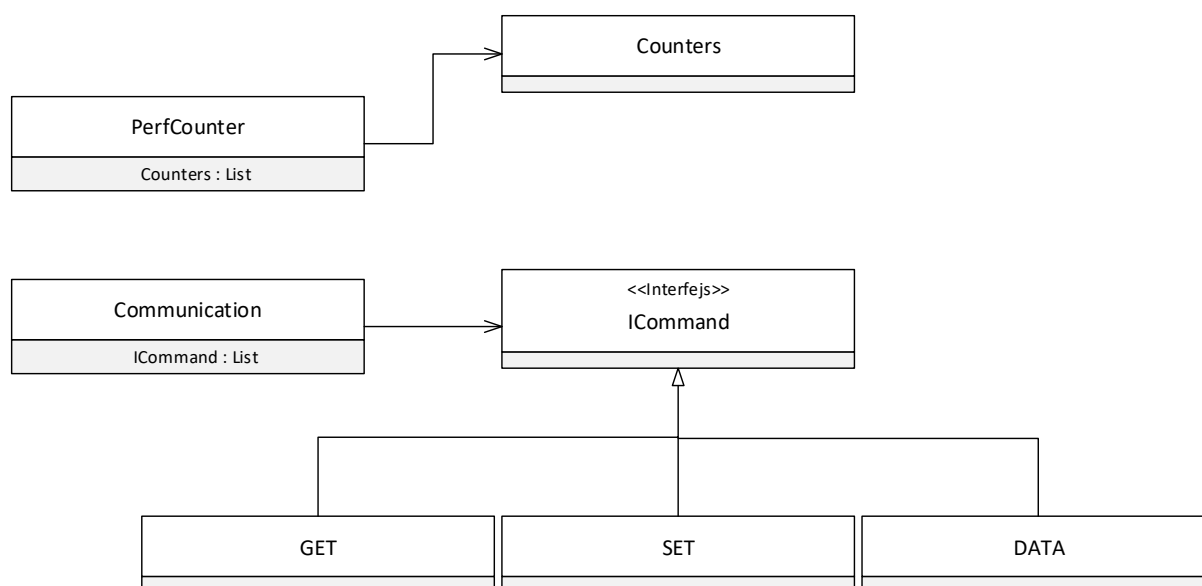
Rysunek 6.3 Diagram sekwencji, Zmiana parametrów aktualnie zbieranego licznika
Źródło: opracowanie własne

- Analiza zebranych danych – po wykonaniu czynności związanych z zebraniem danych istnieje możliwość ich późniejszej analizy. Wynik analizy należy jednak do administratora, aplikacja dostarcza jedynie narzędzie do interpretacji otrzymanych informacji o obciążeniu stacji.

6.1.5. Perspektywa wewnętrzna

Po zapoznaniu się z perspektywą zewnętrzną i spojrzeniu na aplikację oczami użytkownika należy zaznajomić się z perspektywą wewnętrzną. Podrozdział ten poświęcony zostanie na wyjaśnienie projektu od strony technicznej.

Jak już wcześniej wspomniano głównym elementem projektu jest biblioteka DLL. Magazynuje ona funkcjonalności związane z odczytywaniem obciążenia liczników wydajności oraz model komunikacji. Poniżej przedstawiony został diagram klas zawierający najważniejsze elementy wskazanej biblioteki:



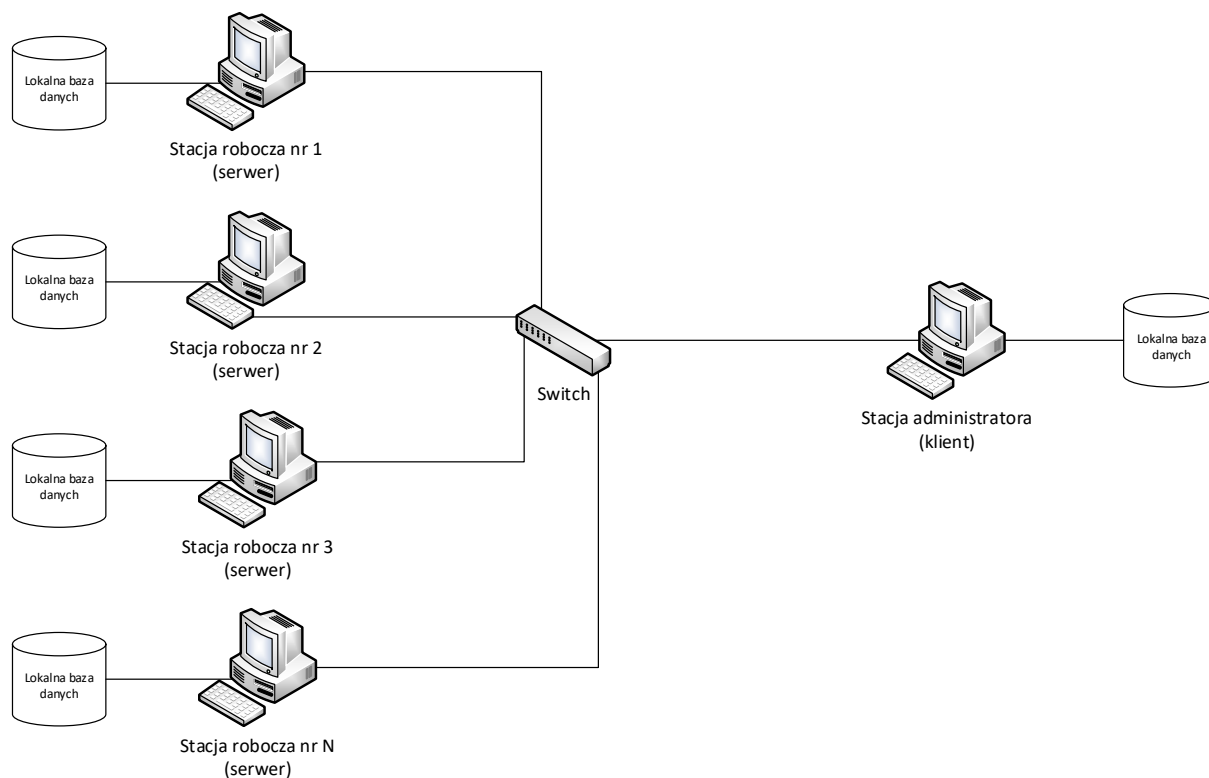
*Rysunek 6.4 Diagram klas
Źródło: opracowanie własne*

Biblioteka posiadać będzie dwie główne klasy:

- PerfCounter – głównym zadaniem klasy jest odczytywanie danych z liczników wydajności i ich zapis danych do bazy danych, zestawianie danych z dni poprzednich czy odczyt danych z bazy.
- Communication – klasa zawierająca model komunikacji. Model ten zostanie dokładnie wytłumaczony w osobnym podrozdziale ze względu na jego obszerność.

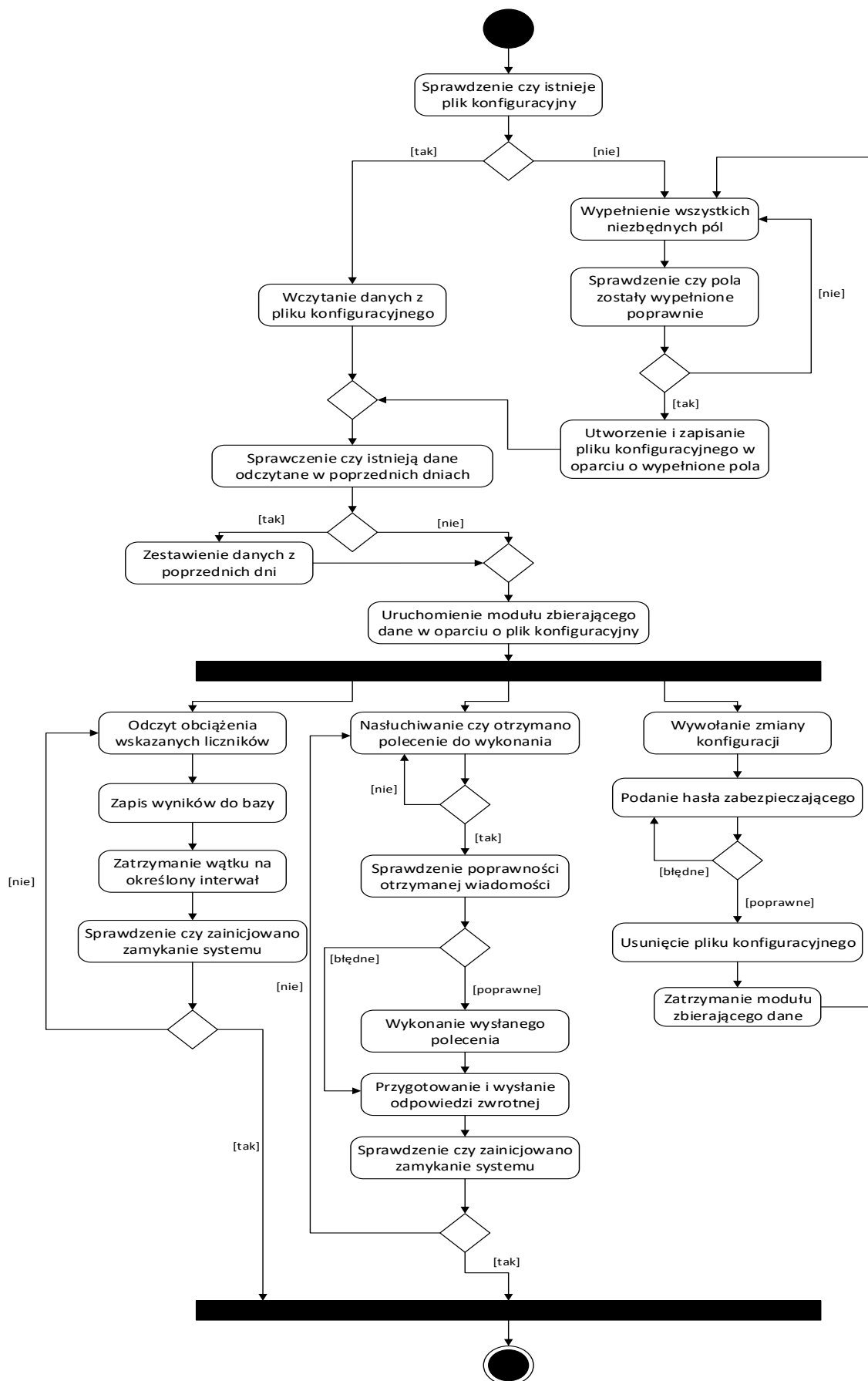
Powyższe klasy posiadają listy obiektów innych klas. Klasa Counters jest klasą zawierającą informacje o liczniku oraz metody dzięki którym istnieje możliwość odczytania z niego danych. Tak więc po utworzeniu w klasie PerfCounter listy obiektów tejże klasy, przechodząc pomiędzy każdym z elementów istnieje możliwość w prosty sposób odczytać dane ze wszystkich liczników które zostały dodane do listy. W klasie Communication zawiera się lista interfejsów ICommand, to dzięki zastosowaniu takiego mechanizmu istnieje możliwość dodania do listy dowolnego polecenia które dziedziczy po tymże interfejsie. Każde polecenie składa się z takich samych funkcji o innym działaniu co na etapie implementacji powinno ułatwić stworzenie systemu komunikacji.

Kolejnymi elementami projektu są aplikacje ClientApp oraz AdminApp. Aplikacje te będą tworzone w architekturze klient-serwer. Dzięki zastosowaniu tego modelu administrator większość czynności będzie mógł wykonać zdalnie bez potrzeby bezpośredniego kontaktu z każdą ze stacji roboczych w celu np. pobrania danych do analizy. Do połączenia obu aplikacji zastosowany będzie protokół TCP/IP gdzie w jednym momencie do ClientApp może się podłączyć tylko jeden program AdminApp. Program zainstalowany na stacjach roboczych w architekturze klient-serwer będzie pełnił rolę serwera ponieważ odbiera połączenie, otrzymuje dane do przetworzenia oraz zwraca wynik, konsola administratora jest natomiast klientem który łączy się do aplikacji ClientApp i przesyła do niego polecenie.



*Rysunek 6.5 Proponowana architektura systemu
Źródło: opracowanie własne*

Powyższy rysunek prezentuje proponowaną architekturę projektowanego systemu. Stacja administratora oraz stacje robocze podłączone są do urządzenia typu switch (bądź paru takich urządzeń), a co za tym idzie znajdują się one w sieci lokalnej co zapewnia mniejsze opóźnienia w przesyłaniu danych. Każda ze stacji posiada odrębną, lokalną bazę danych w której przechowywane są informacje na temat obciążenia. Ze względu na małą ilość danych które będą przesyłane proponowana architektura jest optymalna.



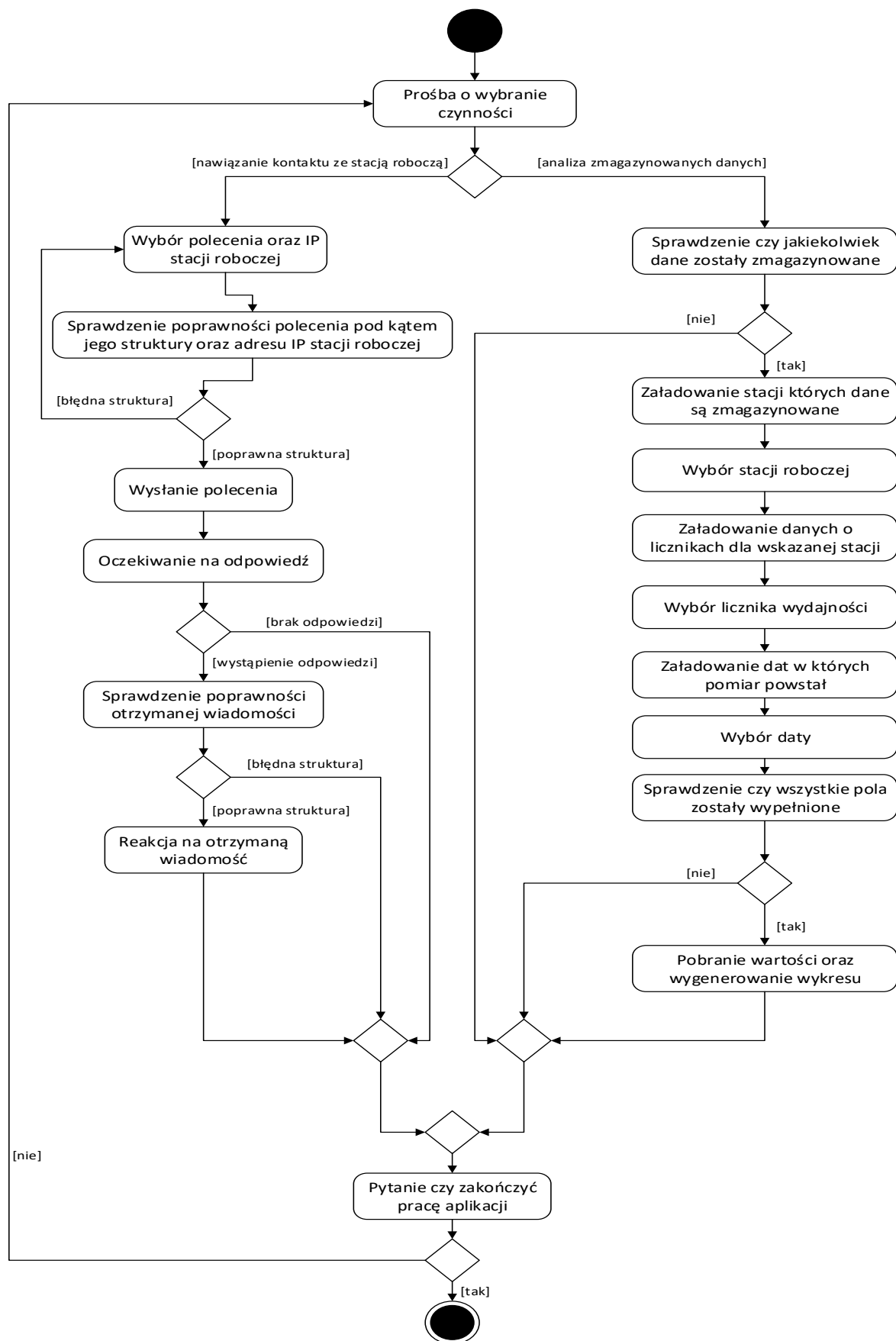
Rysunek 6.6 Diagram aktywności, działanie aplikacji ClientApp
Źródło: opracowanie własne

Powyższy diagram prezentuje dokładniej jak działa aplikacja ClientApp zainstalowana na stacjach roboczych. W pierwszej kolejności sprawdzany jest warunek istnienia pliku konfiguracyjnego. Przy pierwszym uruchomieniu tenże plik nie istnieje co wymusza na administratorze wprowadzenie konfiguracji. Przy próbie zapisu pliku sprawdzany jest warunek czy wszystkie wymagane pola zostały wypełnione, jeżeli tak to tworzony jest plik konfiguracyjny. Jeżeli natomiast aplikacja jest uruchamiana już któryś raz z kolei to odczytywane są jedynie dane z tego pliku. Następnie dochodzi do sprawdzenia czy istnieją zapisy odczytanych obciążeń z innych dni niż ten w którym został program uruchomiony, jeżeli tak to zestawia dane do osobnej bazy danych. W kolejnym kroku uruchamiany jest moduł zbierania danych z liczników wskazanych, czy to przy pierwszej konfiguracji czy pliku który już istniał. Od tego momentu aplikacja obsługuje trzy różne zdarzenia.

Najważniejszym z nich jest odczyt obciążenia ze wskazanych liczników. Następnie dochodzi do zapisania tych danych w bazie oraz zatrzymanie wątku na czas wskazany podczas konfiguracji. W tym momencie należy zaznaczyć że odczyt wykonywany jest w osobno stworzonym dla tego procesu wątku przez co jego zatrzymanie nie zakłóca pracy innych zdarzeń. Pod sam koniec sprawdzany jest warunek dotyczący zamknięcia systemu operacyjnego, jeżeli to wystąpi aplikacja się zamyka, a kolejne wartości nie są już odczytywane.

Drugim zdarzeniem równie ważnym jest nasłuch aplikacji na otrzymanie polecenia. Jeżeli do tego dojdzie nasłuch jest wstrzymywany i rozpoczynany jest proces analizy otrzymanej wiadomości. W pierwszej kolejności sprawdzana jest struktura, jeżeli jest ona niezgodna z modelem komunikacyjnym przygotowywana jest odpowiedź negatywna (ERROR) i wysyłana do klienta. Natomiast jeżeli jest prawidłowa wykonywane są czynności wskazane dla tego polecenia, a następnie wysyłana jest odpowiedź zwrotna w postaci danych bądź komunikatu o pozytywnym zakończeniu operacji (OK). Pod sam koniec sprawdzany jest ponownie warunek o próbie zamknięcia systemu, jeżeli tak to nasłuch nie jest ponownie włączany.

W przypadku trzeciego zdarzenia uzyskujemy możliwość modyfikacji pliku konfiguracyjnego. Aby jednak bez wiedzy administratora parametry nie były zmieniane przez użytkowników stacji roboczej funkcja ta zabezpieczona jest hasłem. Po poprawnym podaniu hasła dotychczasowy plik konfiguracyjny jest usuwany, wyłączany jest również moduł zbierający dane, a następnie dochodzi do ponownej konfiguracji i utworzenia pliku wraz z uruchomieniem modułu z nowymi parametrami.



Rysunek 6.7 Diagram aktywności, działanie aplikacji AdminApp
Źródło: opracowanie własne

Kolejny zaprezentowany diagram aktywności przedstawia działanie aplikacji dla administratora AdminApp. Uruchomienie aplikacji rozpoczyna się od decyzji jaką czynność użytkownik chciałby wykonać. W przypadku nawiązania kontaktu z klientem administrator wybiera polecenie, następnie struktura polecenia jest sprawdzana pod kątem jej poprawności. Jeżeli jest poprawna to aplikacja wysyła komunikat pod wskazany adres IP nawiązując przy tym połączenie z aplikacją ClientApp. Od tego momentu aplikacja nasłuchuje na uzyskanie odpowiedzi, jeżeli ją otrzyma to sprawdzana jest jego struktura. W przypadku poprawności struktury otrzymanej informacji następuje wykonanie czynności zależnych od wysłanego polecenia, może to być np. zapis danych do bazy czy jedynie wyświetlenie komunikatu że zmiana została dokonana poprawnie. Ostatnim pytaniem jakie obsługuje aplikacja jest sprawdzenie czy użytkownik chce zamknąć aplikację, jeżeli nie to ponownie można zdecydować jaką czynność chciałoby się wykonać. W przypadku błędnych struktur bądź braku odpowiedzi aplikacja przechodzi automatycznie do ostatniego pytania o zakończenie pracy aplikacji.

Drugą czynnością jaką administrator może wybrać jest analiza zmagazynowanych danych. Czynność ta może być wykonana jedynie pod warunkiem że jakiekolwiek dane znajdują się w lokalnej bazie (zostały wcześniej pobrane ze stacji i zapisane w bazie). Następnie administrator może przejść do wyboru jakie dane chce przeanalizować, po kolei wybierając stacje roboczą, licznik wydajności oraz datę pomiarów. Zmiana każdego z tych parametrów powoduje zmianę kolejnych, np. wybranie dwóch różnych stacji roboczych zbierających inne dane powoduje załadowanie odpowiadającym im liczników wydajności. Tuż przed końcem sprawdzany jest warunek zaznaczenia wszystkich pól, w przypadku jego spełnienia wyświetlany jest wykres z pobranymi danymi który administrator może przeanalizować. Na koniec ponownie wywoływane jest pytanie o zakończenie pracy aplikacji, analogicznie do poprzednio przedstawianej decyzji.

6.1.6. Model komunikacji zdalnej

W ostatnich podrozdziałach model komunikacji był wskazywany jako jeden z ważniejszych elementów projektowanej biblioteki, w tym rozdziale zostanie ten temat rozszerzony.

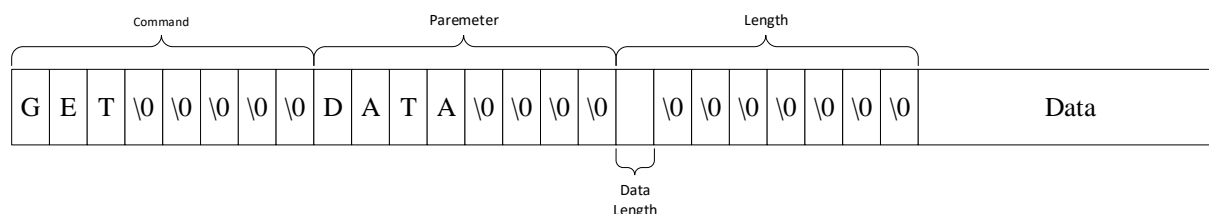
Jego głównym założeniem jest przesyłanie pomiędzy aplikacjami ramki danych o odpowiedniej strukturze. Ramka jest niczym innym jak ciągiem znaków który na etapie analizy dzielony jest na cztery składowe, prezentuje to poniższy rysunek:

Command	Parameter	Length	Data
---------	-----------	--------	------

*Rysunek 6.8 Proponowana ramka danych dla modelu komunikacji
Źródło: opracowanie własne*

Pierwszy człon posiada komendę którą użytkownik chciałby zastosować. Rysunek 6.4 przedstawiający diagram klas przedstawiał komendy które wykorzystywane są tymże modelem, są to GET, SET oraz DATA. Komenda GET pozwala na otrzymanie od serwera którym jest aplikacja

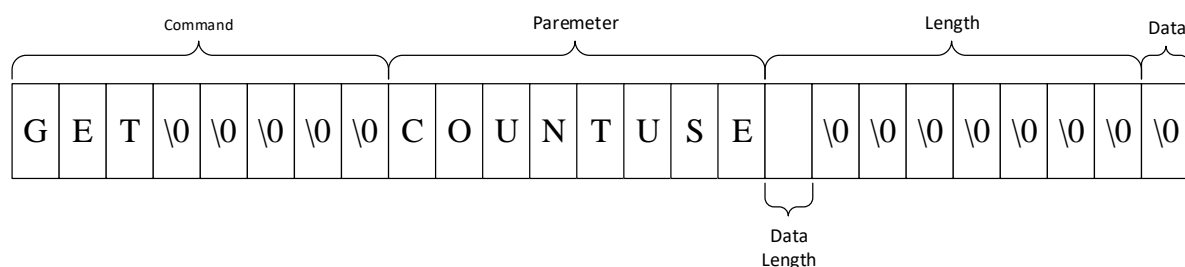
ClientApp dane na temat liczników, jako odpowiedź zwrótną aplikacja kliencka (AdminApp) otrzymuje ramkę z komendą DATA. SET natomiast pozwala na wprowadzenie zmian w parametrach liczników. Kolejny człon to parametr, jest on indywidualny dla każdej komendy. Dzięki niemu wiadomo jaka jest ostateczna potrzeba wykonania polecenia, np. jakie dane mają zostać pobrane. Długość wskazuje na rozmiar danych które zostają przesłane, pozwala to zweryfikować czy wiadomość została wysłana prawidłowo. Na samym końcu występuje blok danych, to jaki jest to rodzaj danych zależy od wybranej komendy oraz parametru, w niektórych przypadkach blok ten jest nieistotny. Poniżej przedstawione zostały wszystkie kombinacje ramek wraz z objaśnieniem działania.



Rysunek 6.9 Komunikat - komenda GET, parametr DATA
Źródło: opracowanie własne

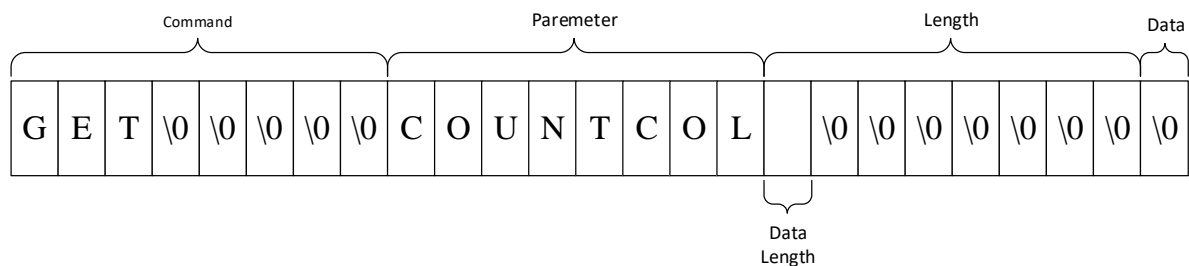
Bloki komendy, parametru oraz długości są podzielone na 8 znaków, te które nie są znaczące zastępowane są znakiem NULL czyli \0. Dzięki temu w łatwy sposób można oddzielić bloki od siebie, a jednocześnie sprawdzić czy struktura wiadomości nie została naruszona. Przed wyjaśnieniem znaczenia powyższego komunikatu warto jeszcze zwrócić uwagę na blok długości. Skrajny od lewej znak w tym bloku oznacza długość bloku danych, w procesie tworzenia tego bloku długość w postaci liczby zamieniana jest na znak odpowiadający jego wartości decymalnej. Przykładem może być tu długość bloku danych równy 33 znaki, wtedy skrajny lewy element bloku zostanie wypełniony znakiem „!” (bez cudzysłowu) [15].

Komunikat widoczny na rysunku 6.9 pozwala na uzyskanie wyników zebranych przy pomocy wskazanego licznika wydajności i podanej daty. Informacje o tym jaki licznik i dzień nas interesuje zostaje zawarty w bloku Data.



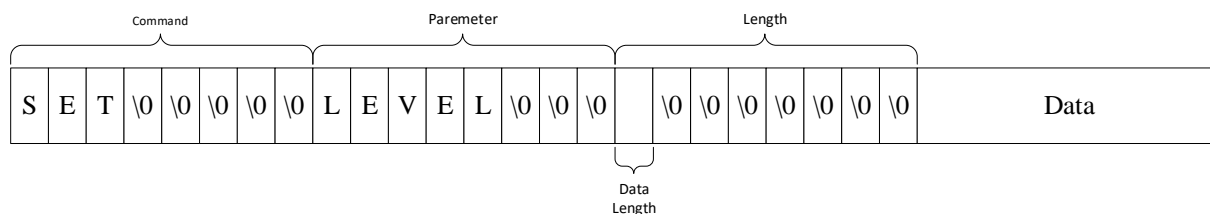
Rysunek 6.10 Komunikat - komenda GET, parametr COUNTUSE
Źródło: opracowanie własne

Powyższy komunikat jest przykładem dla którego bloki Length oraz Data nie biorą udziału w późniejszej interpretacji. Polecenie to pozwala uzyskać informacje o aktualnie zbieranych danych z liczników wydajności (w postaci ich identyfikatorów). Komunikat ten jest o tyle ważny że otrzymane dane są wykorzystywane na etapie użycia komendy SET.



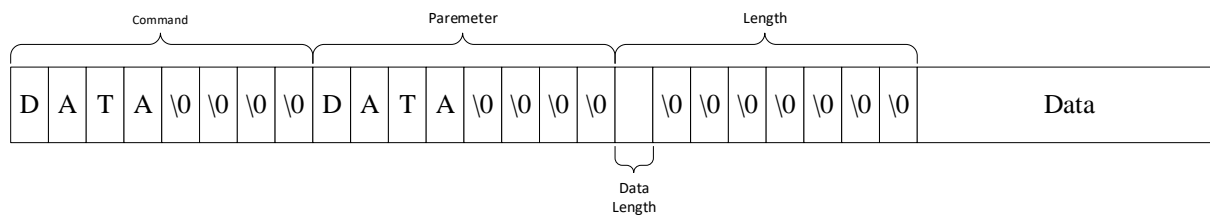
Rysunek 6.11 Komunikat - komenda GET, parametr COUNTCOL
Źródło: opracowanie własne

Tak jak w poprzednim komunikacie tak i w powyższym bloki długości oraz blok danych nie biorą udziału w późniejszej interpretacji. Polecenie to pozwala na uzyskanie informacji o nazwach liczników z których dane zostały zebrane i zestawione w bazie zbiorczej. Wyniki w postaci identyfikatorów wykorzystywane są do wyboru licznika dla komunikatu widocznego na rysunku 6.9.



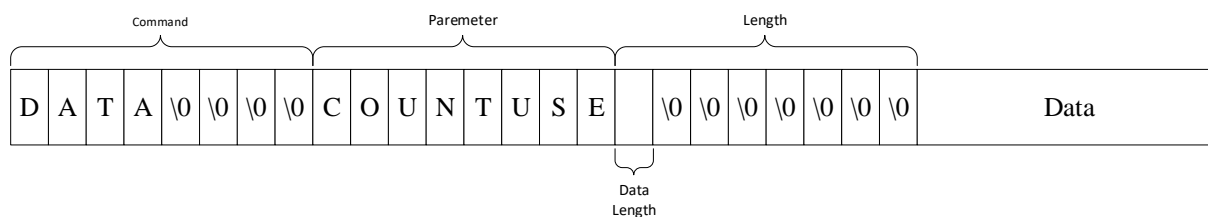
Rysunek 6.12 Komunikat - komenda SET, parametr LEVEL
Źródło: opracowanie własne

Polecenie z komendą SET pozwala na dokonanie zmian w konfiguracji aplikacji ClientApp. Dla tej komendy przewidziano tylko jeden parametr czyli LEVEL. Komunikat daje nam tym samym możliwość modyfikacji skazanego licznika oraz parametrów jego odczytu. Nazwy liczników uzyskujemy dzięki użyciu wcześniej prezentowanego polecenia widocznego na rysunku 6.10.



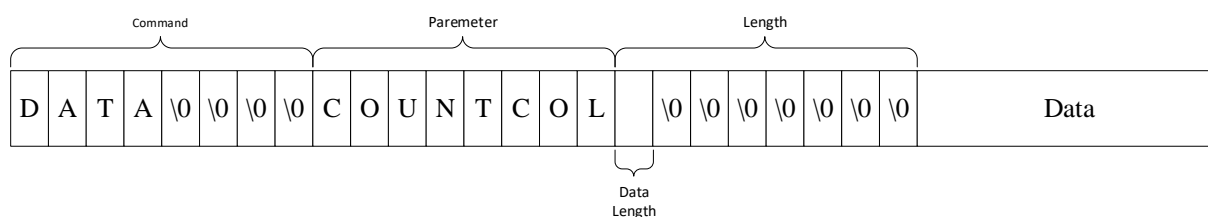
Rysunek 6.13 Komunikat - komenda DATA, parametr DATA
Źródło: opracowanie własne

Poczynając od powyższego rysunku prezentowane komunikaty będą informacjami zwrotnymi z aplikacji ClientApp zawierającymi odpowiedzi na przesłane polecenia w postaci komend GET i SET. Rysunek 6.13 posiadający parametr DATA w bloku danych zawiera informację na temat wyników zebranych dla licznika w danym dniu. Jest on wynikiem przesłania komunikatu z rysunku 6.9.



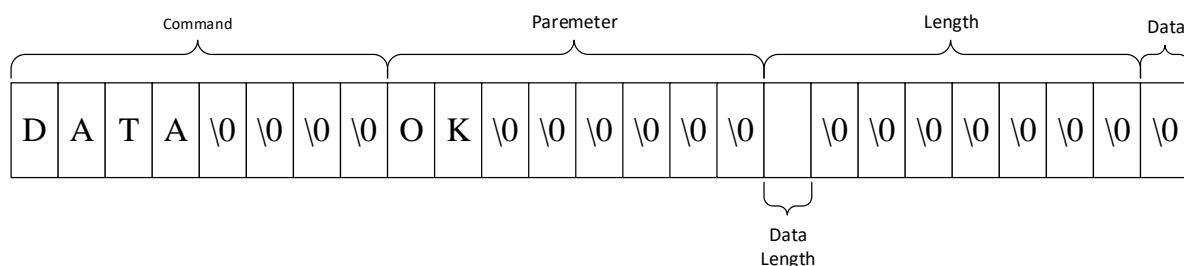
Rysunek 6.14 Komunikat - komenda DATA, parametr COUNTUSE
Źródło: opracowanie własne

Kolejny komunikat z komendą DATA prezentuje odpowiedź w postaci liczników wydajności które aktualnie są zbierane. Polecenie wywołujące widoczne jest na rysunku 6.10.



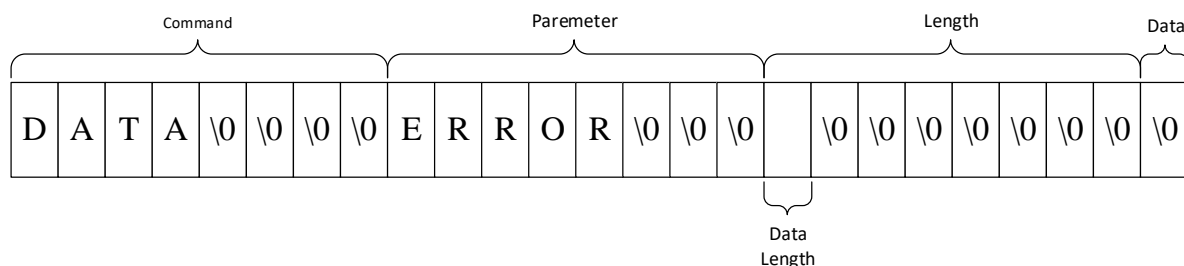
Rysunek 6.15 Komunikat - komenda DATA, parametr COUNTCOL
Źródło: opracowanie własne

Dzięki powyższemu komunikatowi aplikacja AdminApp może otrzymać informację na temat zebranych dotąd wyników pomiarów liczników wydajności. Polecenie wywołujące zaprezentowane jest na rysunku 6.11.



Rysunek 6.16 Komunikat - komenda DATA, parametr OK
Źródło: opracowanie własne

Komunikat na rysunku 6.16 dostarcza informacji o poprawnym wykonaniu przesłanego polecenia, jest też jednym z tych poleceń dla których blok długości i danych jest pomijany. Wykorzystywany jest przy zmianie parametrów liczników (rysunek 6.12).



Rysunek 6.17 Komunikat - komenda DATA, parametr ERROR
Źródło: opracowanie własne

Powyższy jest ostatnim z możliwych do przesłania komunikatem. Sygnalizuje on że polecenie nie zostało pomyślnie przyjęte. Powodem takiego stanu może być:

- Zła składnia przesłanego komunikatu (zachowanie kolejności, nazw komend i parametrów)
- Przesłane dane są błędne bądź podane w złym formacie (może wystąpić np. przy komendzie GET z parametrem DATA gdzie w bloku danych znajdzie się błędna nazwa licznika bądź nieistniejąca w bazie data)
- Nie można było dokonać zmiany parametrów (jedynie dla komendy SET z parametrem LEVEL)

6.2. Implementacja

Rozdział ten zaprezentuje sposób napisania oprogramowania. Zobrazuje w sposób praktyczny jak zbudowane są aplikacje oraz biblioteka oraz jak należy z oprogramowania korzystać.

6.2.1. Założenia ogólne

W podrozdziale 6.1 zaprezentowano projekt systemu do monitorowania obciążenia oraz wydajności stacji roboczych. Przedstawiona dalej implementacja posiadać będzie funkcjonalności których zabrakło w przypadku testowanych w rozdziale piątym aplikacji.

Zgodnie z założeniami projektu aplikacje zostały napisane dla systemów z rodziny Microsoft Windows od wersji 7 do wersji 10. Jako środowisko programistyczne wybrano .NET Framework w wersji 4.5. Posiada on obszerną ilość bibliotek, a dodatkowo systemy wyżej podane wymuszają przy aktualizacji systemu instalacje właśnie tego środowiska przez co można założyć że będzie się ono znajdowało na każdej stacji roboczej. Językiem programowania wybranych przez autora do napisania biblioteki oraz aplikacji został C#, najlepiej poznany przez autora.

Tworzone aplikacje do wymiany informacji stosują protokół TCP/IP oraz będą one wykorzystywały port 20000, ważnym jest aby port ten nie był zablokowany. ClientApp oraz AdminApp przewidziane są do pracy w lokalnej sieci LAN gdzie komputery są dla siebie widoczne.

6.2.2. Biblioteka DLL

W pierwszej kolejności opisana zostanie implementacja biblioteki DLL, czyli elementu, bez którego pozostałe aplikacje nie mogą pracować. Bibliotekę można podzielić na dwa elementy: moduł komunikacyjny oraz agregujący dane, co zostało już zaprezentowane na diagramie klas (rysunek 6.4).

W module komunikacyjnych każda komenda zawarta jest w osobnej klasie dziedziczącej po wspólnym interfejsie. Dzięki zastosowaniu tego mechanizmu w klasie głównej Communication

utworzona jest jedynie lista interfejsów, a do niej przypisane obiekty klas komend. Każda z tych klas posiada następujące funkcje (przedstawione zostaną najważniejsze):

- `GetAllParam()` – zwraca w postaci tablicy nazwy wszystkich parametrów z których komenda może skorzystać.
- `IsAvailable()` – jako daną wejściową otrzymuje nazwę parametru, a następnie zwraca odpowiedź logiczną czy zawiera on się w zbiorze dostępnych dla tej komendy parametrów.
- `PrepareDataCharArray()` – jako dane wejściowe przyjmuje dane przewidziane w bloku parametr oraz danych. Następnie dzieli każdy na osobny znak i uzupełnia tablicę według wytycznych modelu komunikacji tworząc w ten sposób polecenie, a na koniec zwraca powstałą w ten sposób tablicę.
- `PrepareDataString()` – jako dane wejściowe przyjmuje dane przewidziane w bloku parametr oraz danych. Analogicznie do poprzedniej funkcji tworzone jest polecenie według nałożonych ogólnie reguł, ale w przeciwieństwie do niej wynikiem zwracany jest ciąg znaków (string).
- `ValidationAndPrepareData()` – jedna z ważniejszych funkcjonalności, daje możliwość walidacji danych wejściowych takich jak parametr czy dane, a jednocześnie po pozytywnym zakończeniu przygotowuje blok danych do takiej postaci w jakiej powinien zostać wysłany. I tak np. przy wysyłaniu polecenia GET z parametrem DATA w bloku danych powinien znaleźć się identyfikator licznika, po nim znak „:” (bez cudzysłowu), a następnie data. W tym momencie jesteśmy w stanie sprawdzić czy wskazana data faktycznie jest tym co użytkownik podaje, jeżeli nie to funkcja zwróci błąd.

Aby w pełni zobrazować możliwości modelu komunikacji poza powyższymi funkcjami należy również wskazać te które znajdują się we wcześniej wspomnianej klasie nadrzędnej `Communication`. Jej funkcje składają się na (przedstawione zostaną najważniejsze):

- `GetAllCommand()` – funkcja zwraca tablicę nazw dostępnych komend które można użyć przy tworzeniu polecenia.
- `GetAllParam()` – jako dane wejściowe przyjmuje nazwę komendy, a następnie zwraca wszystkie dostępne dla niej parametry.
- `PrepareDataFrame()` – dzięki tej funkcji możliwym jest utworzenie polecenia przesłanego do aplikacji `ClientApp`. Jako dane wejściowe przyjmuje wszystkie składowe ramki danych (komendę, parametr oraz blok danych), informację logiczną jaki rodzaj danych wyjściowych chcielibyśmy uzyskać (polecenie w postaci tablicy znaków czy ciągu znaków), a w osobliwym przypadku może przyjmować również nazwę licznika wydajności (dzięki mechanizmowi przeciążenia funkcji). Wynikiem jest logiczna wartość oznaczająca status

utworzenia polecenia (pozytywny lub negatywny), polecenie natomiast można odczytać w przygotowanych dla tego celu właściwościach.

- `InterpretationOfCommandFrame()` – funkcja daje możliwość zinterpretowania otrzymanej przez `ClientApp` ramki danych. Ramka jest dzielona na bloki, a następnie każdy z nich jest sprawdzany pod kątem poprawności, jeżeli struktura jest poprawna wykonywane są czynności wymagane dla polecenia. Należy również zaznaczyć że jedną z danych wejściowych jest wskaźnik na obiekt klasy `PerfCounters` (w dalszej części pracy implementacja klasy zostanie przedstawiona), zawiera ona funkcje dzięki którym uzyskujemy poprawne dane do odpowiedzi zwrotnej.

- `PrepareDataFrameWithReceivedFrame()` – funkcja ta jest bardzo zbliżona do poprzednio prezentowanej `PrepareDataFrame`, służy ona jednak do utworzenia ramki zwrotnej (z komendą `DATA`). Ramka danych tworzona jest w oparciu o zinterpretowane dane, a w przypadku niepoprawnych danych tworzona jest przedstawiona na rysunku 6.17 ramka błędu. Jako dane wejściowe przyjmowana jest informacja logiczna w jakiej postaci wygenerowana ma być ramka (tablica znaków czy ciąg znaków).

- `InterpretationOfReceived()` – po otrzymaniu odpowiedzi należy ją jeszcze zinterpretować. I tak jak w przypadku poprzedniej funkcji do interpretacji ramki tak i w tym przypadku sprawdzana jest jej struktura pod względem poprawności, jeżeli dane są poprawne dzięki funkcji `GetResultInterpretation()` możemy uzyskać wyniki wydanego wcześniej polecenia.

Poza modulem komunikacyjnych istotnym jest aby nakreślić możliwości modułu agregującego dane czyli klasy `PerfCounters`. Zawiera ona w sobie listę obiektów klasy `Counter` której możliwości zostaną przedstawione w pierwszej kolejności.

Klasa `Counter` posiada tylko jedną funkcję, a mianowicie `GetLevel`, jest to funkcja dzięki której uzyskujemy wartość licznika wydajności. Na etapie inicjalizacji klasy w jego konstruktorze znajdują się informacje o liczniku z którego chcemy skorzystać, nazwie pliku który zostanie utworzony do magazynowania odczytanych danych, ID czyli identyfikator licznika (to on między innymi przesyłany jest w przypadku poleceń wydawanych przez aplikację `AdminApp`) oraz poziomy przy których będzie wykonywany odczyt (ten mechanizm zostanie wyjaśniony szczegółowo w utworzonych aplikacjach).

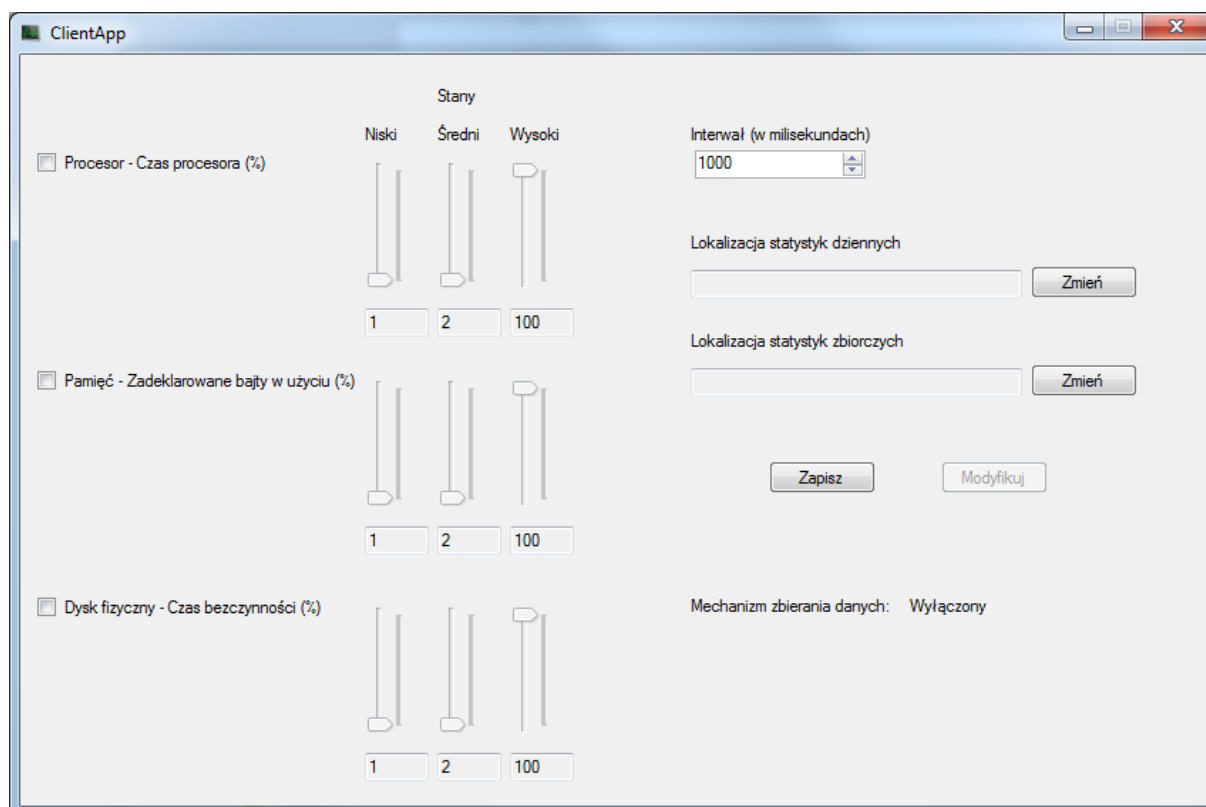
Do najważniejszych funkcjonalności klasy `PerfCounters` należą:

- `AddCounter()` – dzięki tej funkcji dodawane są nowe liczniki wydajności do listy
- `StartCollecting()` – uruchamia wątek który dokonuje odczytu danych z liczników wydajności oraz zapisuje te dane do plików.
- `StopCollecting()` – zatrzymuje działanie wcześniej uruchomionego wątku.

- Summary() – zestawia dane otrzymane z poprzednich dni, a następnie zapisuje je w lokalnej bazie danych z podziałem na daty oraz wartości
- CollectedCounters() – dzięki niej otrzymujemy tablice nazw liczników wydajności które są aktualnie zestawione (odpowiedź dla polecenia z rysunku 6.11).
- CounterInUse() – dzięki tej funkcji otrzymujemy tablicę nazw liczników wydajności które są aktualnie uruchomione i z których czerpane są dane (odpowiedź dla polecenia z rysunku 6.10).
- ChangeLevels() – zmiana poziomów dla wskazanego licznika wydajności.

6.2.3. Oprogramowanie ClientApp

Pierwsza z aplikacji jaka zostanie przedstawiona jest ClientApp czyli aplikacja do agregowania danych z liczników wydajności. Poniżej przedstawione zostanie okno główne aplikacji:

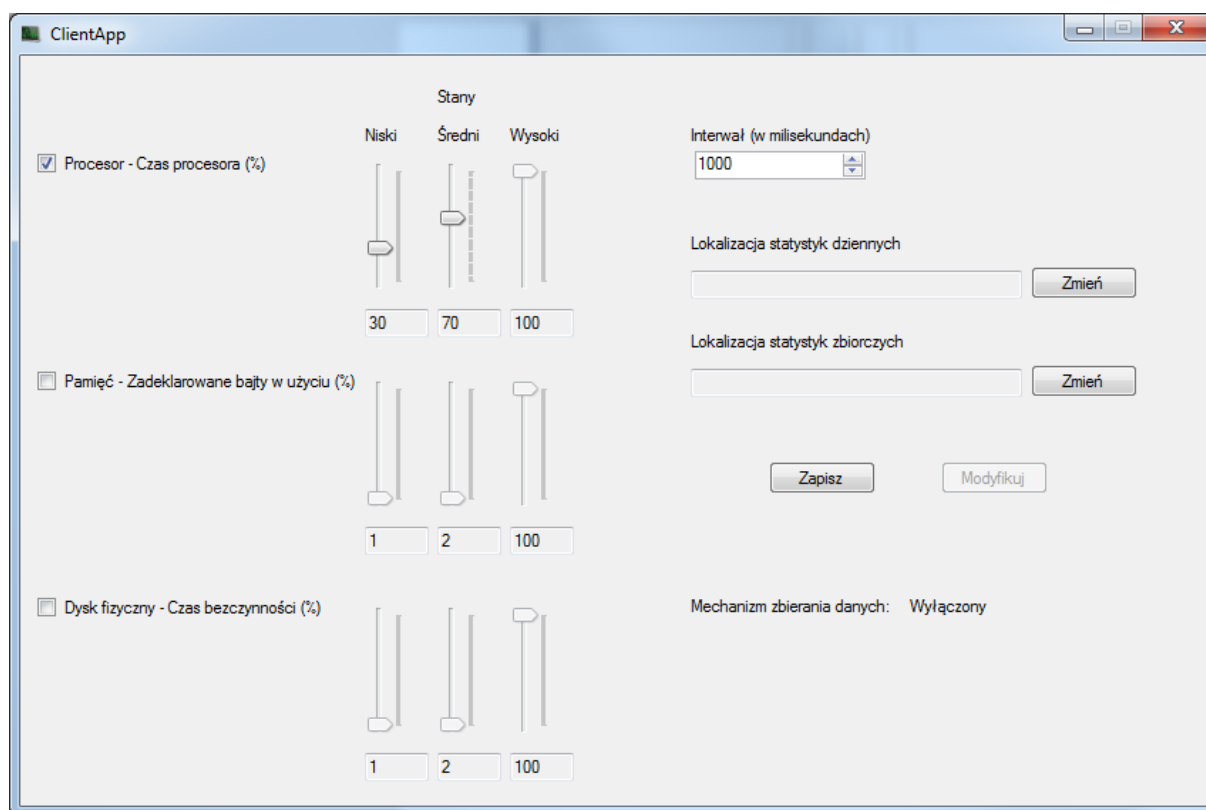


Rysunek 6.18 Ekran główny aplikacji ClientApp
Źródło: opracowanie własne

Po lewej stronie widoczne są liczniki z których można skorzystać. Zostały one wybrane na podstawie artykułu „Mierzmy puls naszego serwera” autorstwa Stevena Choy’a [16]. Autor przedstawia w nim wąskie gardła każdego z komponentów systemu komputerowego oraz liczniki wydajności dzięki którym można mierzyć obciążenie stacji roboczej. I tak oto wybrano czas procesora czyli procent czasu w którym procesorowi zadano wykonanie zadania (jego zajętość), w przypadku pamięci wybrano zadeklarowane bajty w użyciu które odzwierciedlają procentową zajętość pamięci

wirtualnej (pamięć RAM + plik wymiany), a w przypadku dysku zdecydowano się na czas bezczynności czyli procentowy czas w którym dysk nie był obciążony. W przypadku ostatniego licznika ponownie zasugerowano się artykułem, istnieje licznik wydajności odpowiadający czasowi dysku (analogicznie do czasu procesora) lecz w jego przypadku możliwym jest przekroczenie wartości 100%. Problem ten występuje najczęściej przy macierzach dyskowych [17].

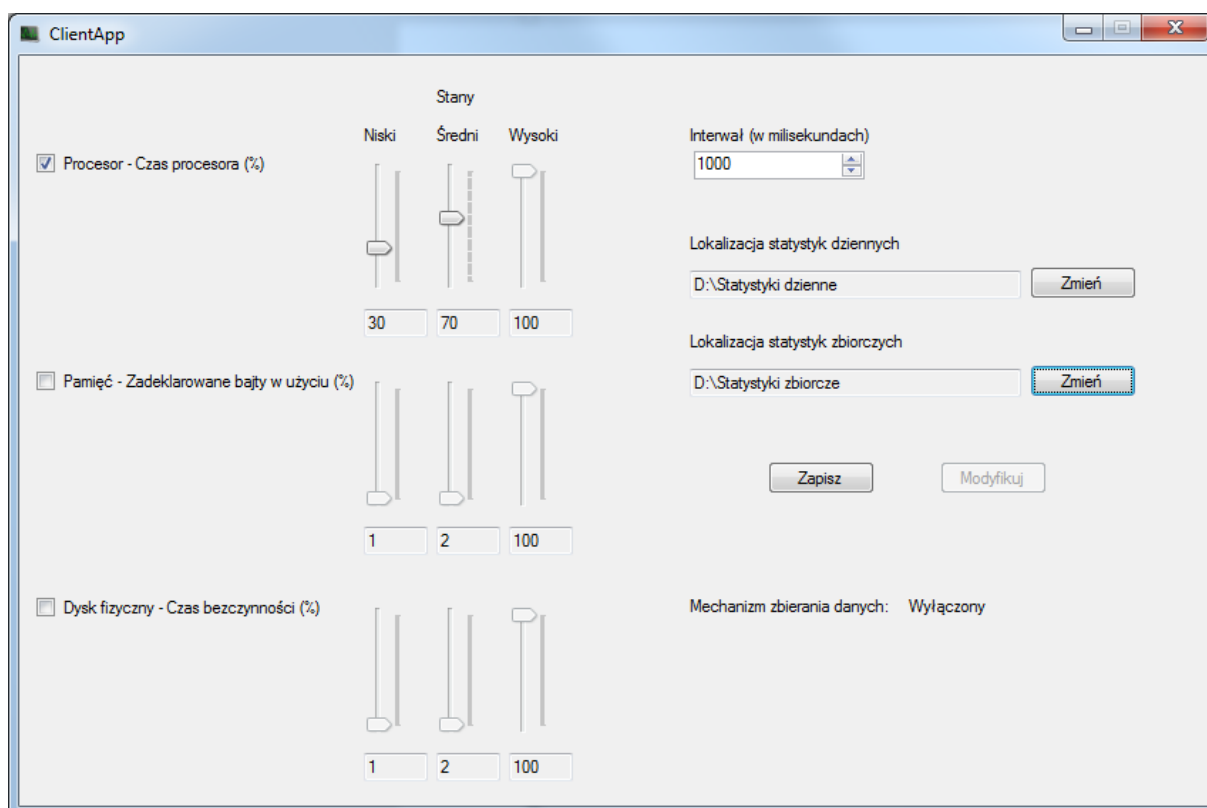
Przy każdym liczniku widoczne są suwaki, zostają odblokowane przy oznaczeniu licznika wydajności. Suwaki te symbolizują poziom odczytu obciążenia. Jeżeli uważamy że niski poziom obciążenia np. procesora jest to stan od 0% do 30% ustawiamy suwak na wartość 30%. Automatycznie suwak drugi nie pozwoli na wybranie wartości mniejszej bądź równej suwakowi pierwszemu. Analogicznie ustawiamy również najwyższą wartość dla stanu średniego. Ostatni ze stanów czyli stan wysoki ustawiony jest od samego początku na wartość 100%. Poniższy rysunek przedstawia przykładową konfigurację:



Rysunek 6.19 Przykładowa konfiguracja licznika Czas procesora (%)
Źródło: opracowanie własne

Następnie możliwe jest ustawienie interwału czyli co jaki czas mają być pobierane dane z liczników wydajności. Najmniejszą wartością jaką można wybrać jest 1000 (1 sekunda) natomiast największą będzie 86400000 (24 godziny). Sugerowane jest jednak pozostawienie wartości 1000 ponieważ wyniki otrzymane z takich pomiarów są łatwiejsze w dalszej analizie.

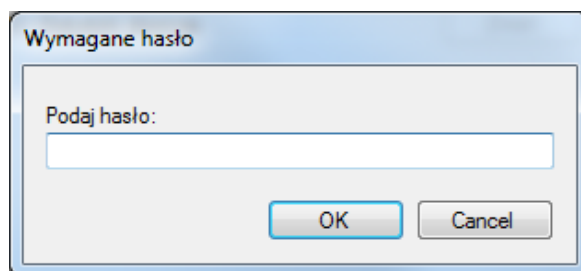
Kolejnym elementem jest wybranie katalogów dla statystyk dziennych oraz statystyk zbiorczych. Ważnym jest aby wybrane katalogi posiadały uprawnienia odczytu, zapisu oraz usuwania plików, uprawnienia te są sprawdzane podczas wybierania katalogu przy pomocy przycisku Zmień.



Rysunek 6.20 Przykładowa konfiguracja aplikacji ClientApp
Źródło: opracowanie własne

Dzięki zastosowaniu konfiguracji z rysunku 6.20 aplikacja po zapisaniu danych wszystkie kontrolki zostaną zablokowane oraz uruchomi mechanizm agregacji danych (wtedy też status zmieniony zostanie na „Włączony”). Oznacza to że co sekundę będzie odczytywana wartości licznika wydajności, oceniana w którym stanie aktualnie się znajduje (pierwszym czyli do 30%, drugim pomiędzy 30% a 70% oraz wysokim czyli 70% do 100%), a następnie jedynie informacja o stanie zostanie zapisana do pliku tekstowego gdzie 1 oznacza stan niski, 2 stan średni natomiast 3 stan wysoki. Plik powstanie w katalogu podanych dla statystyk dziennych, każdy z liczników posiada osobny plik dzienny. Każde ponowne uruchomienie aplikacji powoduje uruchomienie mechanizmu zestawiania danych do lokalnej bazy danych jeżeli dzień został zakończony, w przeciwnym wypadku dane są zapisywane do powstałego już pliku, a sumowanie danych jest pomijany.

Istnieje również możliwość zmodyfikowania konfiguracji. W tym celu natomiast potrzebne jest podanie hasła kontrolnego.



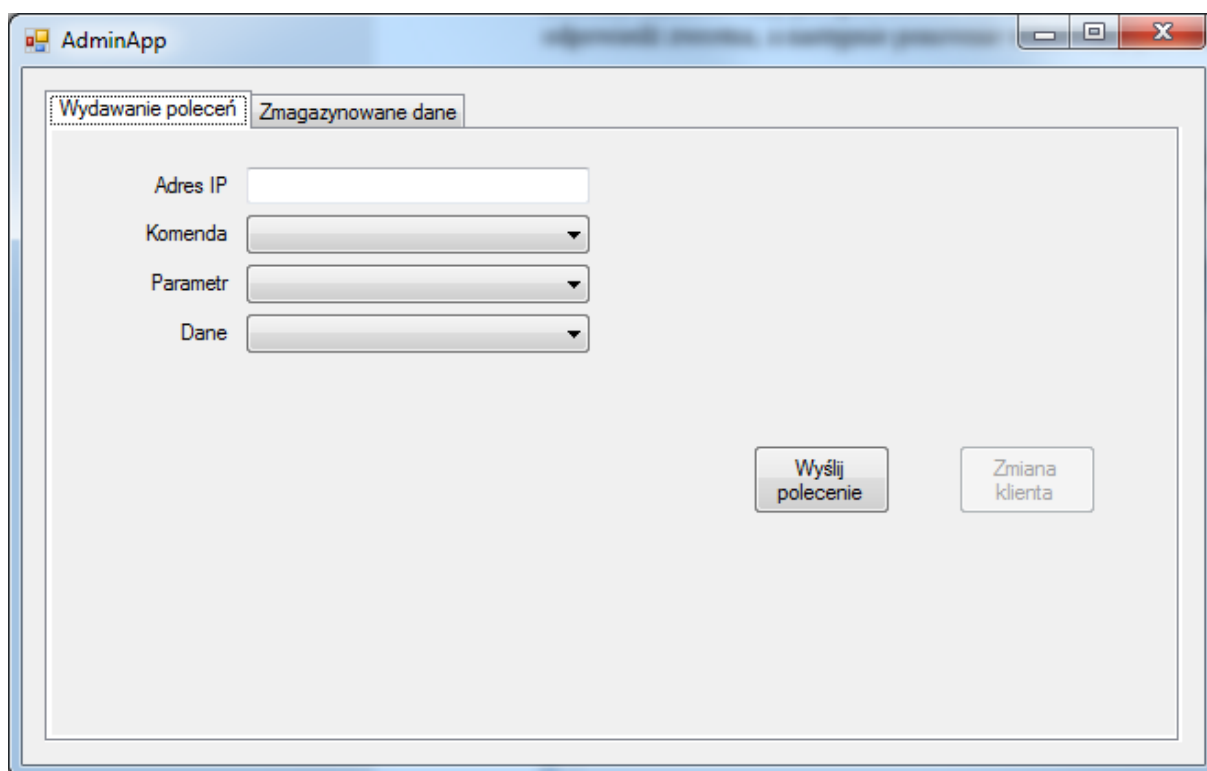
Rysunek 6.21 Okno "Wymagane hasło"
Źródło: opracowanie własne

Aby przejść przez ten proces poprawnie należy w polu do wprowadzania tekstu wpisać ciąg „1234” (bez cudzysłowu) i zaakceptować przyciskiem OK. Uzyskamy wtedy ponowny dostęp do wszystkich kontrolek oraz istnieje możliwość wprowadzenia nowej konfiguracji. Ważnym jest że rozpoczęcie procesu modyfikacji konfiguracji usuwa jej poprzedni stan, niewykonanie zapisu spowoduje przy ponownym uruchomieniu prośbę o wprowadzenie konfiguracji.

W tle natomiast działa mechanizm komunikacji. W tym celu uruchomiony zostaje Listener dzięki któremu nasłuchiwany jest port 20000. Po uzyskaniu polecenia jest on przetwarzany i wysyłana jest odpowiedź zwrotna, a następnie ponownie wznawiany jest nasłuch na powyższym porcie.

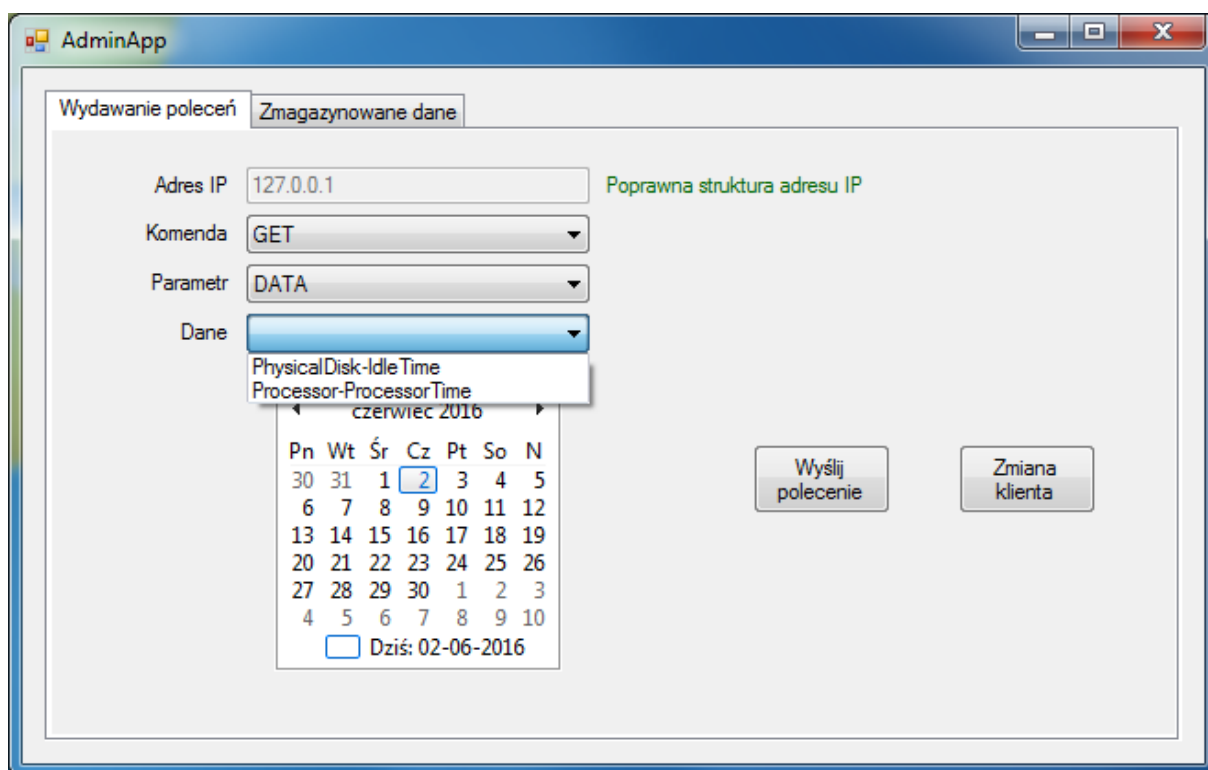
6.2.4. Oprogramowanie AdminApp

Kolejną aplikacją którą należy opisać jest AdminApp. Jest to konsola nadzorcza administratora dzięki której istnieje możliwość zarządzania danymi zbieranymi na stacjach roboczych oraz analizy zebranych danych.



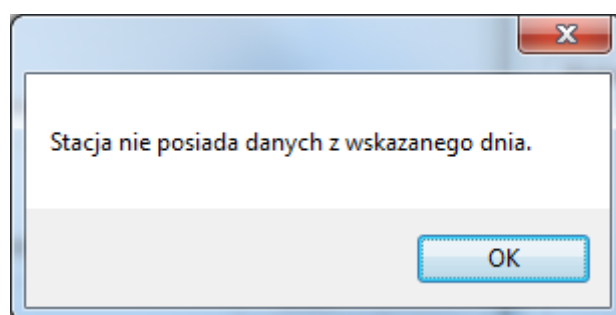
Rysunek 6.22 Ekran główny aplikacji AdminApp
Źródło: opracowanie własne

Pierwsza widoczna zakładka czyli „Wydawanie poleceń” jak sama nazwa wskazuje pozwala nam na utworzenie polecenia oraz wysłanie jej do stacji roboczej pod wskazane IP (w czasie podawania adresu jest on weryfikowany pod względem poprawnej struktury). Dla tej aplikacji dostępne są dwie komendy: GET i SET. Dla parametrów COUNTCOL i COUNTUSE komendy GET okno dialogowe się nie zmienia i sugerowanym jest aby polecenia z tą komendą i tymi parametrami zostały wysłane na samym początku ponieważ przy próbie pobrania danych (GET z parametrem DATA) czy zmianie danych (SET z parametrem LEVEL) niezbędne będą wyniki tychże poleceń.



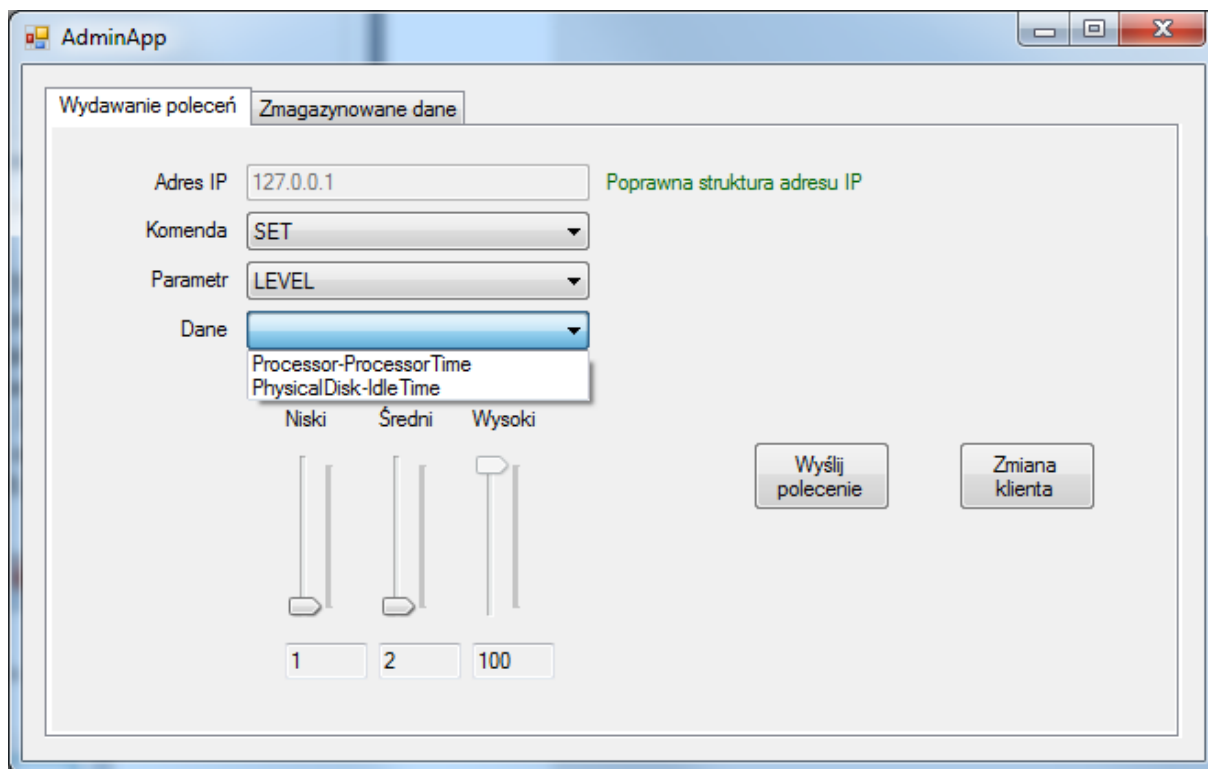
Rysunek 6.23 Wygląd ekranu po wybraniu komendy GET z parametrem DATA
Źródło: opracowanie własne

Dzięki uzyskanym wynikom z polecenia GET z parametrem COUNTCOL administrator uzyskuje dostęp do danych parametru DATA. Przy tej opcji należy wybrać licznik który nas interesuje oraz datę dnia dla którego dane mają zostać pobrane. Jeżeli wskazana data znajduje się w bazie lokalnej wskazanej stacji roboczej to dane zwrotne zapisane zostaną w lokalnej bazie danych na komputerze administratora w katalogu w którym znajduje się konsola nadzorcza. Do tychże danych będzie miał dostęp w zakładce „Zmagazynowane dane”. W przypadku wybrania niepoprawnej daty w kierunku aplikacji AdminApp zostanie przesłana ramka z parametrem ERROR i zostanie wyświetlony poniższy komunikat:



Rysunek 6.24 Ekran źle dobranej daty
Źródło: opracowanie własne

Kolejny rysunek przedstawia widok dla komendy SET z parametrem LEVEL dla poprzednio uzyskanej odpowiedzi polecenia GET z parametrem COUNTUSE.

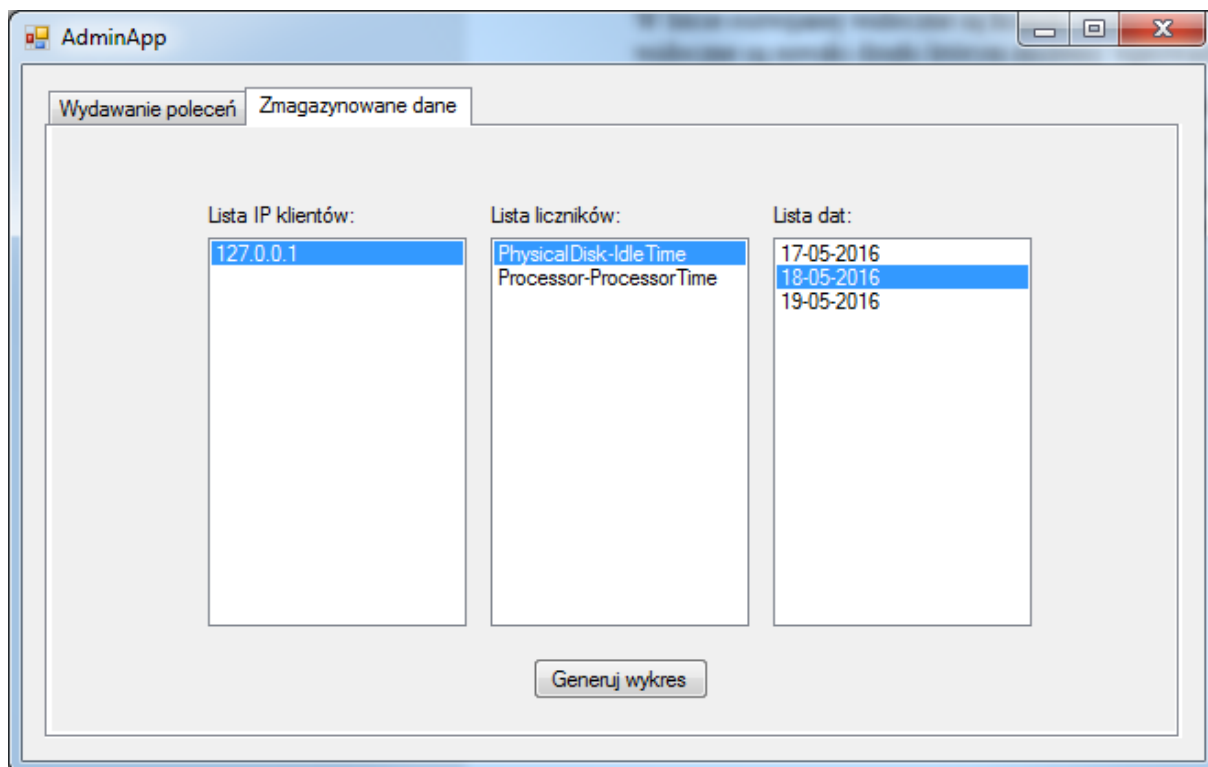


*Rysunek 6.25 Wygląd ekranu po wybraniu komendy SET z parametrem LEVEL
Źródło: opracowanie własne*

W liście rozwijanej widoczne są liczniki wydajności które aktualnie są uruchomione natomiast poniżej widoczne są suwaki dzięki którym możemy wprowadzić interesujące nas zmiany.

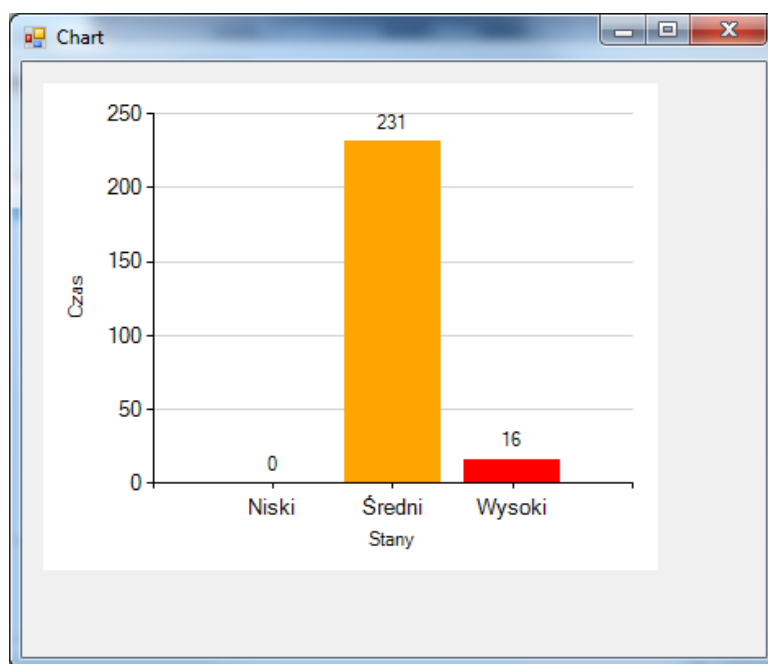
W opisywanej zakładce znajduje się również przycisk „Zmiana klienta”. Powoduje on wyzerowanie wartości dla wszystkich kontrolerek i możliwość wprowadzenia adresu IP nowej stacji roboczej.

Drugą natomiast zakładką udostępnioną w ramach konsoli nadzorczej jest „Zmagazynowane dane”. Znajdują się tam informacje uzyskane dzięki poleceniu GET z parametrem DATA. Poniżej przedstawiony zostaje przykładowy widok zakładki:



Rysunek 6.26 Przykładowy widok zmagazynowanych danych
Źródło: opracowanie własne

Pierwsza od lewej widoczna lista zawiera adresy IP wszystkich klientów dla których dane zostały zebrane. Po wybraniu klienta druga lista wypełnia się nazwami wszystkich liczników zebranych przez wskazanego klienta. Trzecia lista natomiast odpowiada za wybranie daty dla danych które chcemy wyświetlić. Po naciśnięciu przycisku „Generuj wykres” pojawi się wykres słupkowy dla każdego ze stanów, poniżej przykładowy wynik:



Rysunek 6.27 Przykładowy wygenerowany wykres
Źródło: opracowanie własne

I tak uzyskaliśmy wykres dla licznika wydajności związanego z czasem bezczynności dysku. Można zauważyć że w stanie średnim dysk znajdował się w stanie bezczynności przez 231 sekund, a w stanie wysokim przez 16 sekund. Założenie to jest prawdziwie jeżeli interwał był ustawiony na 1000 milisekund. Dodatkowo można wskazać że dysk ze względu na mierzenie bezczynności dla stanu wysokiego tego licznika należy interpretować jako zajętość na niskim poziomie, natomiast niski stan jako zajętość na wysokim poziomie. W przypadku pozostałych interpretacja jest dokładnie taka jak na wykresie.

6.2.5. Instalacja oprogramowania

W tym podrozdziale zostaną przedstawione wskazówki co do instalacji oprogramowania. W pierwszej kolejności zostanie zaprezentowana aplikacja ClientApp.

Do jej działania poza plikiem o rozszerzeniu exe wymagane są dwie biblioteki: PerfLibrary.dll oraz HelperLibrary.dll. Jeżeli biblioteki nie znajdą się w tym samym katalogu to aplikacja nie zostanie uruchomiona. Kolejnym wymogiem jest umiejscowienie aplikacji w katalogu który pozwala na odczyt, zapis oraz kasowanie plików, związane jest to z tworzeniem się pliku konfiguracyjnego w katalogu w którym aktualnie aplikacja się znajduje. Jeżeli katalog wymaga uprawnień administratora, np. „C:\Program Files\”, to aplikacja powinna być uruchamiana z uprawnieniami administratora. Sugeruje się również umiejscowienie skrótu do aplikacji w katalogu Autostart tak aby dane były zbierane od startu systemu.

Aplikacja AdminApp posiada podobne wymagania co poprzednio prezentowana aplikacja. Wymagane są te same biblioteki do jej działania oraz katalog w którym się znajduje musi posiada uprawnienia do zapisu, odczytu i kasowanie plików ponieważ lokalna baza danych, czyli plik o strukturze XML, znajduje się w tym samym katalogu co konsola nadzorcza.

6.2.6. Możliwości dalszej rozbudowy

Autorskie oprogramowanie w postaci ClientApp czy AdminApp ukazuje pewien sposób wykorzystania utworzonej biblioteki. Jest jednak możliwość rozbudowania tychże aplikacji o dodatkowe funkcje.

I tak na przykład dla aplikacji ClientApp istnieje możliwość zmodyfikowania interfejsu do postaci w której to administrator sam wybiera z listy licznik który chce zastosować, a wbudowany jest on w system operacyjny danej stacji roboczej. Możliwe jest to dzięki trzem funkcjom które zostały utworzone w bibliotece w czasie gdy cele jakie miały spełniać aplikacje nie zostały do końca określone. Te funkcje to:

- GetCategory() – zwraca w postaci tablicy nazwy wszystkich dostępnych kategorii liczników.

- `GetInstance()` – zwraca w postaci tablicy nazwy wszystkich instancji jakie istnieją dla wskazanej, jako parametr, kategorii. Jeżeli kategoria nie posiada instancji funkcja zwróci wartość `NULL`.
- `GetCounters()` – zwraca w postaci tablicy wszystkie nazwy liczników dostępnych dla wskazanej kategorii oraz licznika, a w przypadku braku licznika to tylko kategorii.

Kolejną funkcjonalnością którą można dołączyć jest większa ilość stanów. Klasa `Counter` dopuszcza zastosowanie większej ilości stanów lecz pewnym ograniczeniem jest to że do każdego licznika musi być przypisana taka sama liczba stanów. Dostarcza ona również możliwość podania maksymalnej wartości dla liczników które nie dostarczają wartości w postaci procentowej dzięki czemu na późniejszym etapie wartość ta przeliczana jest na procentowy udział.

Istnieje również możliwość sprawdzania obciążania kart graficznych, jest to jednak związane z koniecznością instalacji oprogramowania dostarczonego przez producenta, dlatego też taka funkcjonalność nie została zaimplementowana w oprogramowaniu. Szczegóły można odnaleźć w dokumentacji firmy `NVidia` [18] i `AMD` [19].

W przypadku zmian w aplikacji `ClientApp` należy dokonać analogicznych zmian w `AdminApp` ponieważ np. generowany wykres dostosowany jest jedynie do obsługi trzech stanów, nie jest on dynamiczny.

6.3. Testy oprogramowania

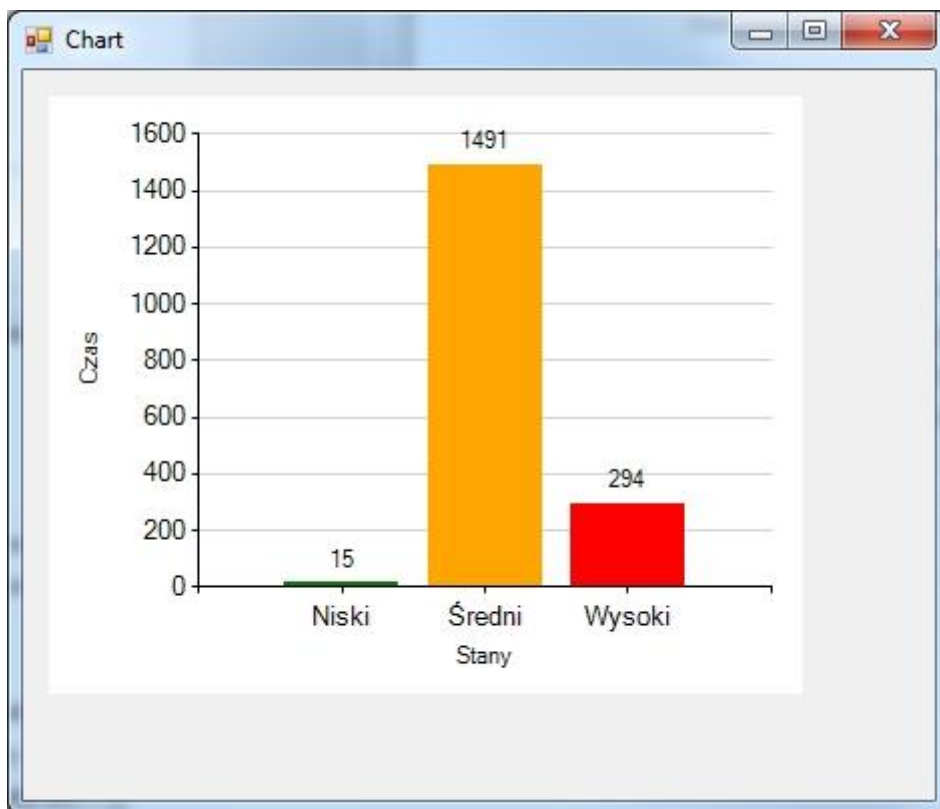
Do testów wybrano trzy stacje robocze różniące się pod względem specyfikacji. Na każdej z nich zostało zainstalowane utworzone oprogramowanie w postaci `ClientApp` oraz `AdminApp`. W pierwszej kolejności skonfigurowano aplikację agregującą dane, skupiono się jedynie na obciążeniu procesora gdzie stan niski oznaczono do 45%, stan średni do 75% natomiast wysoki pozostał na poziomie do 100%. Dla łatwiejszej interpretacji otrzymanych wyników interwał ustawiony został na 1 sekundę.

Testowane stacje były wyposażone w następujące procesory:

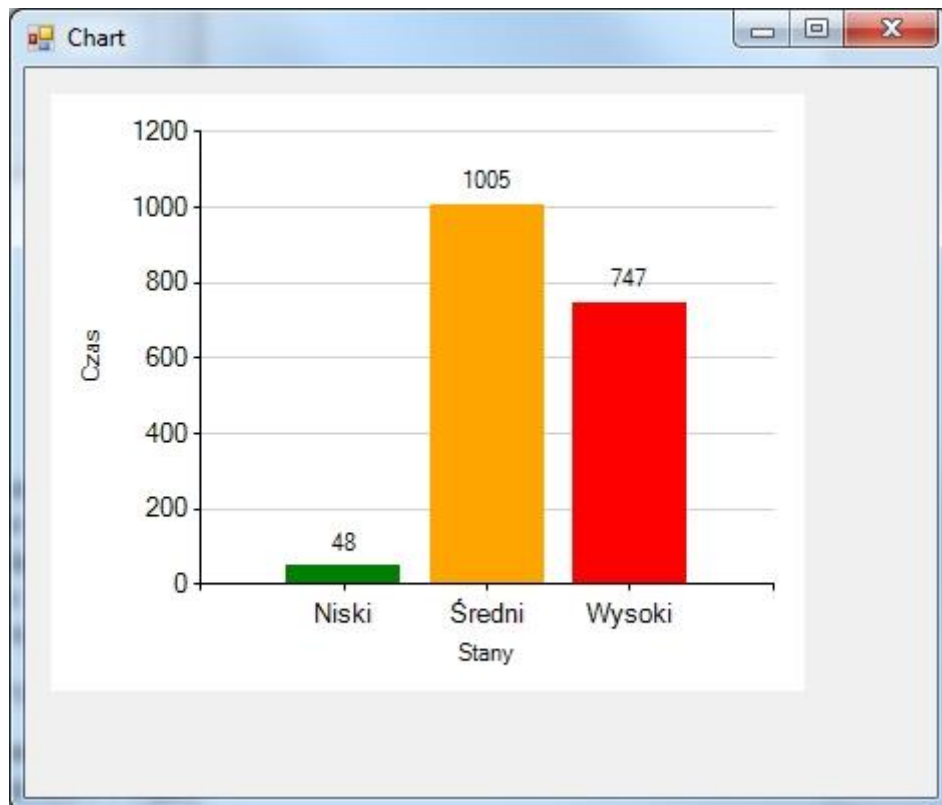
- Maszyna nr 1 – Intel Core i5-3350 (4 rdzenie i 4 wątki)
- Maszyna nr 2 – Intel Core i5-2430M (2 rdzenie i 4 wątki)
- Maszyna nr 3 – Intel Core 2 Duo T8300 (2 rdzenie i 2 wątki)

Na każdej z maszyn zostało zainstalowane oprogramowanie `MATLAB` w wersji `R2016a`. W toku studiów w tymże programie została utworzona symulacja dotycząca lotu samolotu gdzie jako dane wejściowe przyjmowane są między innymi: prędkość samolotu względem powietrza, wysokość na jaką ma się wznieść, kąt natarcia i wiele innych. Symulacja natomiast na bieżąco oblicza jaką pozycję

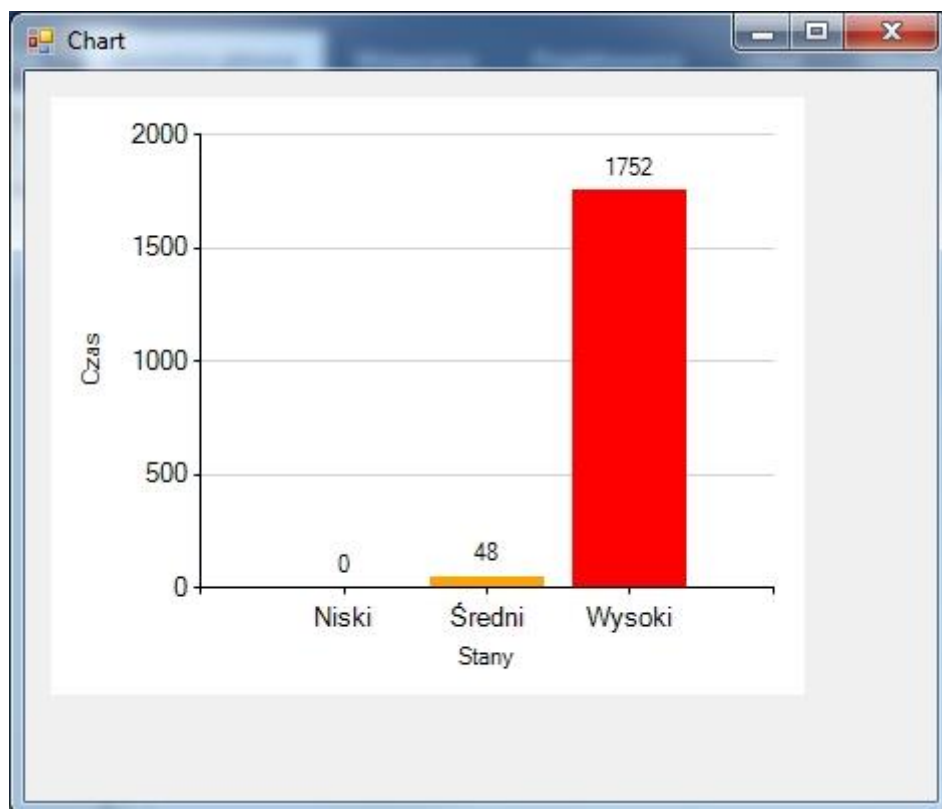
posiada samolot (jak długą trasę przeleciał) na co składa się wiele operacji matematycznych. Zostanie ona wykorzystana w trakcie testów do obciążania maszyny. Symulacja była na każdej stacji roboczej uruchomiona przez 30 minut, w jej trakcie aplikacja ClientApp odczytywała co sekundę jaki jest stan obciążenia procesora. Na sam koniec dzięki aplikacji AdminApp pobrano otrzymane wyniki (aplikacja została lokalnie zainstalowana, w miejscu adresu podawano jedynie pętlę zwrotną „127.0.0.1” aby połączyć się z aplikacją ClientApp).



*Rysunek 6.28 Wynik obciążenia dla maszyny nr 1
Źródło: opracowanie własne*



Rysunek 6.29 Wynik obciążenia dla maszyny nr 2
Źródło: opracowanie własne



Rysunek 6.30 Wynik obciążenia dla maszyny nr 3
Źródło: opracowanie własne

Powyższe rysunki prezentują otrzymane wyniki pomiaru obciążenia dla wszystkich trzech maszyn. Już na pierwszy rzut oka widoczna jest przepaść pomiędzy maszyną nr 3 a pozostałymi stacjami roboczymi. Na rysunku 6.30 widoczna jest znaczna przewaga stanu wysokiego nad pozostałymi co sugeruje że stacja posiada niską wydajność dla zadanej czynności. W tym przypadku można zaproponować wymianę procesora na szybszy.

Na rysunku 6.29 widoczna jest stosunkowo niewielka różnica w wyniku dla stanu średniego i wysokiego, oznaczać to może przeciętną wydajność stacji roboczej. Można tutaj zasugerować wymianę podzespołu jakim jest procesor lecz nie jest to wymagane. Zależy to natomiast od wymagań użytkownika, w przypadku gdy poza przedstawioną symulacją system komputerowy miałby wykonywać również inne czynności to zapewne wartość dla stanu wysokiego by wzrosła. Gdyby jednak symulacja byłaby jedyną wykonywaną operacją to można przystać że procesor jest wystarczający dla tychże potrzeb.

Rysunek 6.28 prezentuje najszybszą maszynę ze wszystkich, można to zauważyć po najniższym wyniku stanu wysokiego. Jednak tak jak w poprzednim wypadku ewentualna wymiana komponentu jakim jest procesor zależne jest od wymagań użytkownika oraz interpretacji samego wykresu.

7. Podsumowanie

Efektem pracy było napisanie oprogramowania uzupełniającego funkcjonalność aplikacji firm trzecich, kilka przykładów zostało zaprezentowanych w rozdziale piątym. Celem natomiast było stworzenie aplikacji klient-serwer dzięki której na bieżąco dane o aktywności użytkownika na stacji roboczej były rejestrowane, a następnie mogły być przetworzone przez administratora bez konieczności fizycznego logowania się na te same stacje (obsługa zdalna). Warto jednak wspomnieć że pierwotne założenia projektu były odmienne od wskazanych w pracy. Dopiero wstępna analiza gotowych rozwiązań wykazała problemy w określaniu wydajności komputera, a co za tym idzie, zainspirowała do stworzenia autorskiej funkcjonalności rozszerzającej możliwości testowanych aplikacji.

Aby jednak połączyć zagadnienia wskazane w temacie pracy należało zauważyć istotną zależność pomiędzy wydajnością, a obciążeniem. Wydajność, poza definicją wskazaną w rozdziale drugim, może być rozumiana jako efektywne wykonywanie operacji przy zachowaniu niskiego stanu obciążenia. Dzięki takiemu rozumowaniu wydajność konkretnej stacji roboczej można wyznaczyć na podstawie pomiarów jej obciążenia w danym okresie czasu. W celu zbadania obciążenia wykorzystano liczniki wydajności, ich zaletą jest to że są wbudowane w system operacyjny, a co za tym idzie korzystanie z nich (jak na przykład odczyt danych) nie wpływa w widoczny sposób na pracę systemu. Dla autorskiego oprogramowania wybrano liczniki związane z czasem procesora, czasem bezczynności dysku oraz rozmiarem zarezerwowanej pamięci wirtualnej. Finalnym tworem oprogramowania stała się biblioteka zawierająca protokół komunikacji oraz moduł do agregacji odczytywanych danych z liczników wydajności, aplikacja ClientApp wykorzystująca funkcjonalności biblioteki w praktyce oraz program AdminApp jako konsola nadzorcza do pobierania danych, zmiany parametrów konfiguracji oraz do analizy otrzymanych wyników pomiarów.

Testy napisanej aplikacji wykazują zróżnicowane wyniki w zależności od zamontowanego komponentu w stacji roboczej. To co można było podczas nich zauważyć to fakt konieczności „ręcznej” analizy otrzymanych danych. W odczuciu autora jest to pozytywny aspekt ponieważ nie narzuca sposobu interpretowania wyników w porównaniu do aplikacji firm trzecich gdzie każdy komponent jest oceniany według nieznanego dla użytkownika wzoru.

Bibliografia

- [1] Dokument dotyczący sprzedaży komputerów w latach 90' poza granice Stanów Zjednoczonych, <http://www.bis.doc.gov/index.php/component/content/article/16-policy-guidance/product-guidance/270-fact-sheet>, [[:@]] kwiecień 2016
- [2] Lista pięciuset najszybszych superkomputerów, <http://www.top500.org/>, [[:@]] kwiecień 2016
- [3] Definicja wydajności według serwisu TechTarget, <http://whatis.techtarget.com/definition/performance>, [[:@]] kwiecień 2016
- [4] Definicja obciążenia według serwisu TechTarget, <http://searchdatacenter.techtarget.com/definition/workload>, [[:@]] kwiecień 2016
- [5] Obciążenie wprowadzane przez system operacyjny, <http://www.makeuseof.com/tag/windows-performance-affected-hardware-software/>, [[:@]] maj 2016
- [6] Piotr Metzger, *Anatomia PC*, Wydanie XI, Helion 2007, ISBN: 978-83-246-1119-5
- [7] Linda Null, Julia Lobur, *Struktura organizacyjna i architektura systemów komputerowych*, Helion 2004, ISBN: 83-7361-430-3
- [8] Definicja wskaźnika WEI, <http://windows.microsoft.com/pl-pl/windows/what-is-windows-experience-index#What-is-windows-experience-index=windows-7>, [[:@]] kwiecień 2016
- [9] PCMark8, <https://www.futuremark.com/benchmarks/pcmark8>, [[:@]] maj 2016
- [10] PerformanceTest 8 http://www.passmark.com/download/pt_download.htm, [[:@]] maj 2016
- [11] Sandra Lite, <http://www.sisoftware.co.uk/download-lite/>, [[:@]] maj 2016
- [12] Opis monitora wydajności, <http://windows.microsoft.com/pl-pl/windows/what-is-windows-experience-index#What-is-windows-experience-index=windows-7>, [[:@]] maj 2016
- [13] Mark T. Edmead, Paul Hinsberg, *Windows NT Performance: Monitoring, Benchmarking, and Tuning, Appendix A*, <https://technet.microsoft.com/pl-pl/library/cc768048.aspx>, [[:@]] maj 2016
- [14] Marko Aho, Christopher Vinckier, *Computer System Performance Analysis and Benchmarking*, http://www.cs.inf.ethz.ch/37-235/studentprojects/vinckier_aho.pdf, [[:@]] maj 2016
- [15] Tablica ASCII, <http://www.asciitable.com/>, [[:@]] maj 2016
- [16] Artykuł dotyczący wyboru liczników wydajności, <https://technet.microsoft.com/pl-pl/library/mierzymy-puls-naszego-serwera.aspx#4>, [[:@]] maj 2016
- [17] Powód przekroczenia wartości 100% dla licznika wydajności „Czas dysku (%)”, <https://support.microsoft.com/pl-pl/kb/310067>, [[:@]] maj 2016
- [18] Dokumentacja instalacji liczników wydajności dla kart z firmy NVidia, <https://developer.nvidia.com/nvidia-perfkit>, [[:@]] maj 2016
- [19] Dokumentacja instalacji liczników wydajności dla kart z firmy AMD, <http://developer.amd.com/tools-and-sdks/graphics-development/gpuperfapi/>, [[:@]] maj 2016

Aneks

Zawartość płyty CD

Na płycie CD w katalogu „Dokumentacja” znajduje się cyfrowa wersja niniejszego dokumentu w formacie PDF. W katalogu „Implementacja” można odnaleźć projekt utworzony w programie Microsoft Visual Studio 2012, w katalogu „Pliki wykonywalne” znalazły się skompilowane aplikacje oraz biblioteki. Ostatnim katalogiem jest „Testy”, jego zawartością jest model oraz funkcja niezbędna do uruchomienia symulacji wykonywanej w rozdziale 6.3 (pliki obsługiwane przez oprogramowanie MATLAB).

Spis rysunków

Rysunek 3.1. Przepływ informacji z pamięci masowej do procesora	12
Rysunek 4.1. Przykładowy wynik uzyskany przy pomocy wskaźnika WEI.....	15
Rysunek 5.1 Okno główne aplikacji PCMark 8 Basic Edition.....	20
Rysunek 5.2 Wynik wykonania Home test	20
Rysunek 5.3 Wynik wykonania Work test.....	21
Rysunek 5.4 Wynik wykonania Creative test.....	21
Rysunek 5.5 Wykres porównujący dla Home test.....	22
Rysunek 5.6 Wykres porównujący dla Creative test.....	22
Rysunek 5.7 Wykres porównujący dla Work test	22
Rysunek 5.8 Okno aplikacji PerformanceTest 8.0	23
Rysunek 5.9 Podsumowanie wyników dla aplikacji PerformanceTest 8.0	24
Rysunek 5.10 Funkcjonalności związane z wydajnością dla aplikacji SANDRA	25
Rysunek 5.11 Wyniki wydajności dla programu SANDRA	26
Rysunek 5.12 Monitor wydajności – Perfmon	27
Rysunek 5.13 Zilustrowanie zapisu pionowego Źródło: obraz zawarty w dokumencie [12].....	28
Rysunek 5.14 Zilustrowanie zapisu poziomego Źródło: obraz zawarty w dokumencie [12].....	29
Rysunek 6.1 Diagram przypadków użycia.....	32
Rysunek 6.2 Diagram sekwencji, Pobieranie zebranych danych o wskazanym liczniku.....	33
Rysunek 6.3 Diagram sekwencji, Zmiana parametrów aktualnie zbieranego licznika	34
Rysunek 6.4 Diagram klas.....	35
Rysunek 6.5 Proponowana architektura systemu	36
Rysunek 6.6 Diagram aktywności, działanie aplikacji ClientApp	37
Rysunek 6.7 Diagram aktywności, działanie aplikacji AdminApp.....	39
Rysunek 6.8 Proponowana ramka danych dla modelu komunikacji	40
Rysunek 6.9 Komunikat - komenda GET, parametr DATA	41
Rysunek 6.10 Komunikat - komenda GET, parametr COUNTUSE	41
Rysunek 6.11 Komunikat - komenda GET, parametr COUNTCOL	42
Rysunek 6.12 Komunikat - komenda SET, parametr LEVEL	42
Rysunek 6.13 Komunikat - komenda DATA, parametr DATA.....	42
Rysunek 6.14 Komunikat - komenda DATA, parametr COUNTUSE.....	43
Rysunek 6.15 Komunikat - komenda DATA, parametr COUNTCOL	43
Rysunek 6.16 Komunikat - komenda DATA, parametr OK	43
Rysunek 6.17 Komunikat - komenda DATA, parametr ERROR.....	43

Rysunek 6.18 Ekran główny aplikacji ClientApp	47
Rysunek 6.19 Przykładowa konfiguracja licznika Czas procesora (%)	48
Rysunek 6.20 Przykładowa konfiguracja aplikacji ClientApp.....	49
Rysunek 6.21 Okno "Wymagane hasło"	49
Rysunek 6.22 Ekran główny aplikacji AdminApp.....	50
Rysunek 6.23 Wygląd ekranu po wybraniu komendy GET z parametrem DATA	51
Rysunek 6.24 Ekran źle dobranej daty	51
Rysunek 6.25 Wygląd ekranu po wybraniu komendy SET z parametrem LEVEL	52
Rysunek 6.26 Przykładowy widok zmagazynowanych danych	53
Rysunek 6.27 Przykładowy wygenerowany wykres	53
Rysunek 6.28 Wynik obciążenia dla maszyny nr 1	56
Rysunek 6.29 Wynik obciążenia dla maszyny nr 2.....	57
Rysunek 6.30 Wynik obciążenia dla maszyny nr 3.....	57