

Politechnika Świętokrzyska w Kielcach

Wydział Elektroniki, Automatyki i Informatyki

Projekt: **Technologie obiektowe**

Grupa: 1ID21A	Temat: SilentPackage-Lite– Monitorowanie systemu Windows® 10	Skład grupy: Michał Młodawski
Rok studiów: 4		

SilentPackage-Lite – Monitorowanie systemu Windows 10

Opracowanie i sprawozdanie projektu SilentPackage-
Lite

Kwiecień 2020

1. Opis projektu

Projekt SilentPackage-Lite powstał w celu wykorzystania możliwości nowych technologii w celu stworzenia projektu, którego celem było nieustanne monitorowanie specyfikacji systemu Windows.

2. Wykorzystane technologie

W projekcie wykorzystano najnowocześniejsze technologie dla osiągnięcia jak najlepszych walorów estetycznych aplikacji webowej i szybkości działania całego rozwiązania.

1. Użyte języki programowania

1. Java wraz z framework Spring do warstwy serwerowej.
2. C# w wersji .Net Framework do warstwy agregacji danych.
3. HTML z wykorzystaniem frameworka Bootstrap, JavaScript z frameworkiem jQuery oraz AngularJS.
4. JSON jako format tekstowy dla przejrzystego uporządkowania danych.

2. Wykorzystane oprogramowanie przy projektowaniu i wdrażaniu projektu

1. Adobe XD CC w celu szybkiego prototypowanie interfejsu graficznego.
2. IntelliJ IDEA jako główne IDE do programowanie serwera.
3. JavaDoc do prowadzenia dokumentacji kodu.
4. JetBrains WebStorm jako IDE do interfejsu graficznego.
5. Visual Studio z rozszerzeniem ReSharper

3. Użyte biblioteki w projekcie

1. Biblioteki zewnętrzne

LP.	Nazwa biblioteki	Licencja	Opis
1.	Apache Tomcat	Apache License 2.0	Kontener aplikacji webowych.
2.	Spring Boot	GNU Lesser General Public	Framework Java dodający możliwość tworzenia webowych aplikacji.
3.	Angular.js	MIT	Framework wspomagający tworzenie i rozwój aplikacji internetowych na pojedynczej stronie.
4.	Bootstrap	MIT	Framework CSS dodający przydatne elementy do projektowania stron.

5.	Jquery	MIT	Biblioteka programistyczna dla języka JavaScript, ułatwiająca korzystanie z JavaScriptu.
6.	Font Awesome	Licencja wewnętrzna: https://fontawesome.com/ license	Zestaw piktogramów przedstawionych w formie czcionki.
7.	junit5	Eclipse Public License 2.0	Narzędzie służące do tworzenia powtarzalnych testów jednostkowych.

Tabela 1 Biblioteki zewnętrzne wykorzystane w projekcie.

2. Biblioteki wewnętrzne

1. Open Hardware Monitor – Biblioteka do pobierania informacji o urządzeniu. Fork biblioteki, którego celem było unowocześnienie rozwiązania, port do nowszej wersji .NET Framework i zwracania danych w postaci plików JSON.

4. Funkcjonalność projektu

Projekt został stworzony w oparciu o architekturę klient-serwer. Posiada takie funkcjonalności jak:

1. Pobieranie informacji o procesorze, producent i model oraz temperaturze, obciążeniu i taktowaniu każdego z rdzenia procesora.
2. Pobieranie informacji płyty głównej, nazwa producenta, model, wersja BIOS oraz producent BIOS.
3. Pobieranie informacji o dedykowanej karcie graficznej. Umożliwia pobranie takich danych jak: Producent, model, wersja sterownika, gałąź sterownika, temperaturze rdzenia, ilości całkowitej pamięci vRAM i ilości dostępnej pamięci do alokacji vRMA.
4. Pobranie informacji na temat dysków twardych w tym: Nazwy dysku, formatu, lokalizacji, ilości wolnego oraz zajętego miejsca.
5. Pobieranie informacji o zainstalowanej pamięci RAM w tym producencie, modelu, prędkości, numerze banku i ilości wolnego miejsca.
6. Listy procesów z nazwą, identyfikatorem oraz czasem startu procesu.
7. Graficzny interfejs dostępny z poziomu przeglądarki pracujący w koncepcji „jednej strony” obsługujący zapytania AJAX.
8. Dostęp do API w celu rozszerzenia projektu o dodatkowe endpointy.

5. Obsługa projektu

Aby móc korzystać z projektu należy posiadać oprogramowanie Java przynajmniej z **wersji 11 oraz system operacyjny Windows® 10 firmy Microsoft z zainstalowanym pakietem .NET framework 4.7.2**

Uwaga! Do poprawnego działania wymaganej jest uruchomienie aplikacji z poziomu użytkownika z prawami administratora.

6. Opis architektury systemu

Projekt został wykonany w architekturze klient-serwer. Do komunikacji między modułami wykorzystuje architekturę REST z wykorzystaniem protokołu HTTP. Klient został wykonany w technologii C#, jego zadaniem jest pobieranie telemetry systemu operacyjnego Windows 10 przy wykorzystaniu biblioteki Open Hardware Monitor, która została zmodyfikowana tak aby móc współpracować z nowszą wersją .Net Framework, zostały także wprowadzone zmiany co do agregacji danych i ponownie skompilowana. Następnie telemetria w formacie JSON jest wysyłana za pomocą zapytania HTTP zgodnego z architekturą REST do serwera, który został napisany w języku Java zgodnie z modelem MVC. Jego celem jest odebranie danych i prezentowanie ich w czasie rzeczywistym. Za pomocą interfejsu graficznego dostępnego z poziomu przeglądarki.

7. Wygląd interfejsu webowego

1. Strona główna.

Informacje o specyfikacji

Model płyty głównej: ROG STRIX Z390-F GAMING

Model procesora: Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz

Całkowita ilość pamięci RAM: 32 GB

Model karty graficznej: NVIDIA GeForce GTX 1080 Ti

Informacje o płycie głównej

Producent płyty głównej: ASUSTeK COMPUTER INC.

Model: ROG STRIX Z390-F GAMING

Producent BIOS: American Megatrends Inc.

Wersja BIOS: 2001

Procesor

Producent: Intel(R) Corporation

Model: Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz

Dostępna ilość rdzeni: 8

Dostępna ilość wątków: 16

Temperatury

Taktowanie

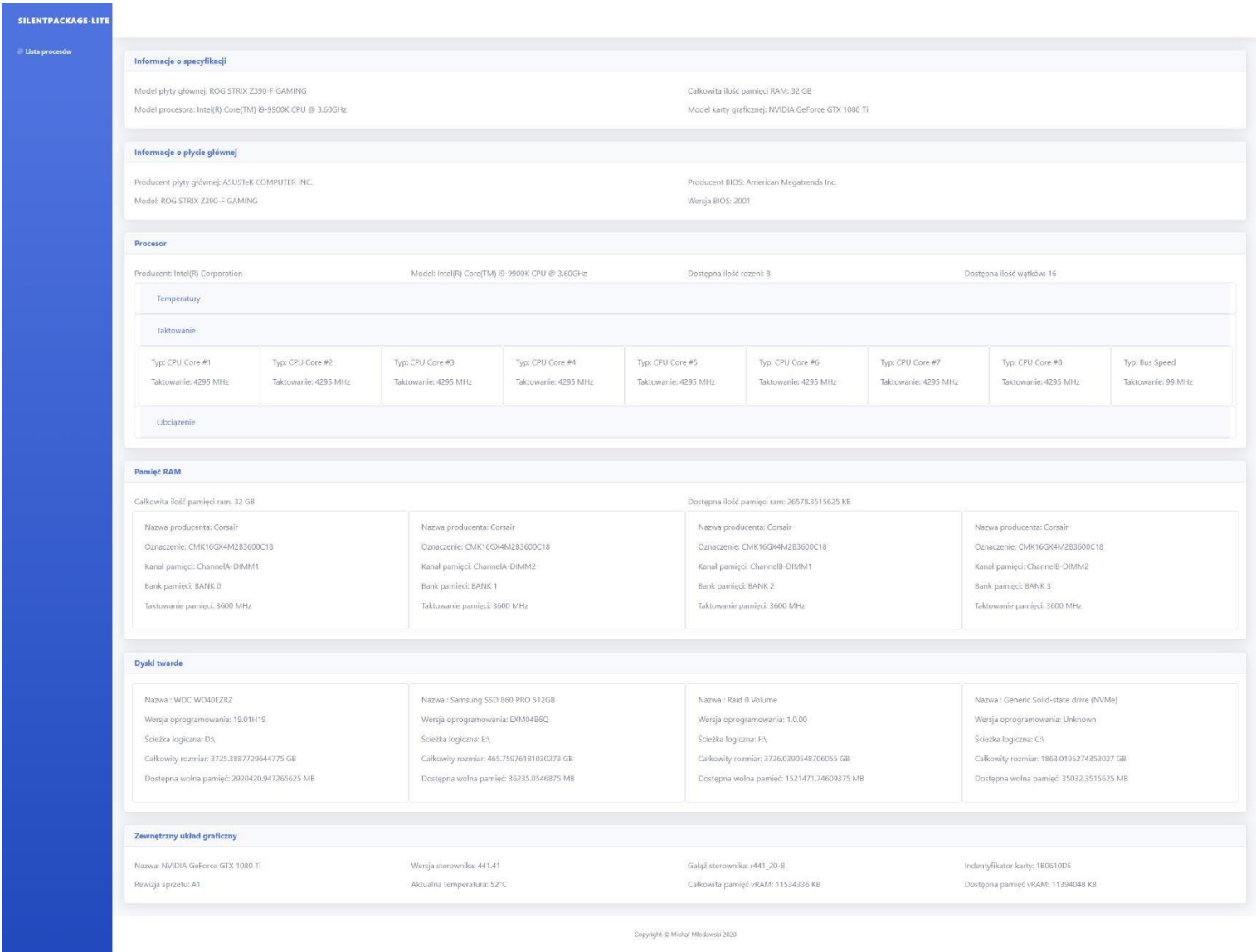
Typ: CPU Core #1	Typ: CPU Core #2	Typ: CPU Core #3	Typ: CPU Core #4	Typ: CPU Core #5	Typ: CPU Core #6	Typ: CPU Core #7	Typ: CPU Core #8	Typ: Bus Speed
Taktowanie: 4295 MHz	Taktowanie: 4295 MHz	Taktowanie: 4295 MHz	Taktowanie: 4295 MHz	Taktowanie: 4295 MHz	Taktowanie: 4295 MHz	Taktowanie: 4295 MHz	Taktowanie: 4295 MHz	Taktowanie: 99 MHz

Obciążenie

Rysunek 1 Częściowy zrzut ekranu z prezentacją funkcjonalności.

Pamięć RAM			
Całkowita ilość pamięci ram: 32 GB		Dostępna ilość pamięci ram: 26578.3515625 KB	
Nazwa producenta: Corsair Oznaczenie: CMK16GX4M2B3600C18 Kanał pamięci: ChannelA-DIMM1 Bank pamięci: BANK 0 Taktowanie pamięci: 3600 MHz	Nazwa producenta: Corsair Oznaczenie: CMK16GX4M2B3600C18 Kanał pamięci: ChannelA-DIMM2 Bank pamięci: BANK 1 Taktowanie pamięci: 3600 MHz	Nazwa producenta: Corsair Oznaczenie: CMK16GX4M2B3600C18 Kanał pamięci: ChannelB-DIMM1 Bank pamięci: BANK 2 Taktowanie pamięci: 3600 MHz	Nazwa producenta: Corsair Oznaczenie: CMK16GX4M2B3600C18 Kanał pamięci: ChannelB-DIMM2 Bank pamięci: BANK 3 Taktowanie pamięci: 3600 MHz
Dyski twarde			
Nazwa : WDC WD40EZRZ Wersja oprogramowania: 19.01H19 Ścieżka logiczna: D\ Całkowity rozmiar: 3725.3887729644775 GB Dostępna wolna pamięć: 2920420.947265625 MB	Nazwa : Samsung SSD 860 PRO 512GB Wersja oprogramowania: EXM04B6Q Ścieżka logiczna: E\ Całkowity rozmiar: 465.75976181030273 GB Dostępna wolna pamięć: 36235.0546875 MB	Nazwa : Raid 0 Volume Wersja oprogramowania: 1.0.00 Ścieżka logiczna: F\ Całkowity rozmiar: 3726.0390548706055 GB Dostępna wolna pamięć: 1521471.74609375 MB	Nazwa : Generic Solid-state drive (NVMe) Wersja oprogramowania: Unknown Ścieżka logiczna: C\ Całkowity rozmiar: 1863.0195274353027 GB Dostępna wolna pamięć: 35032.3515625 MB
Zewnętrzny układ graficzny			
Nazwa: NVIDIA GeForce GTX 1080 Ti Rewizja sprzętu: A1	Wersja sterownika: 441.41 Aktualna temperatura: 52°C	Gałąź sterownika: r441_20-8 Całkowita pamięć vRAM: 11534336 KB	Identyfikator karty: 1B0610DE Dostępna pamięć vRAM: 11394048 KB

Rysunek 2 Częściowy zrzut ekranu z prezentacją funkcjonalności.



Rysunek 3 Zrzut ekranu z prezentacją funkcjonalności

2. Lista procesów.



SILENTPACKAGE-LITE		
Lista procesów		
Lista procesów		
PID	Nazwa	Data startu
1620	svchost	27.03.2020 19:12:13
9188	NisSvc	27.03.2020 19:12:15
8136	conhost	27.03.2020 20:39:11
23884	conhost	27.03.2020 20:38:59
908	lsass	27.03.2020 19:12:12
14688	OneDrive	27.03.2020 19:12:31

Rysunek 4 Częściowy zrzut ekranu z prezentacją funkcjonalności.

8. Diagramy prezentujące projekt.

Diagram klas interfejsu webowego.

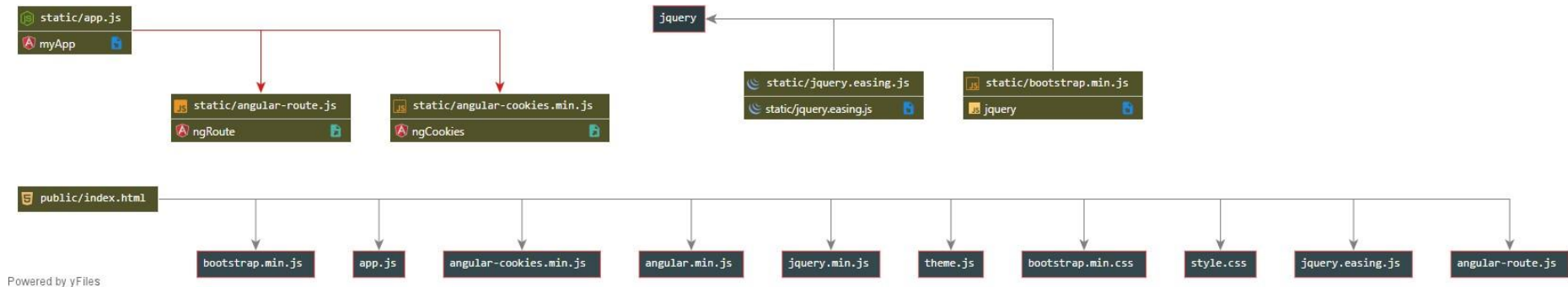
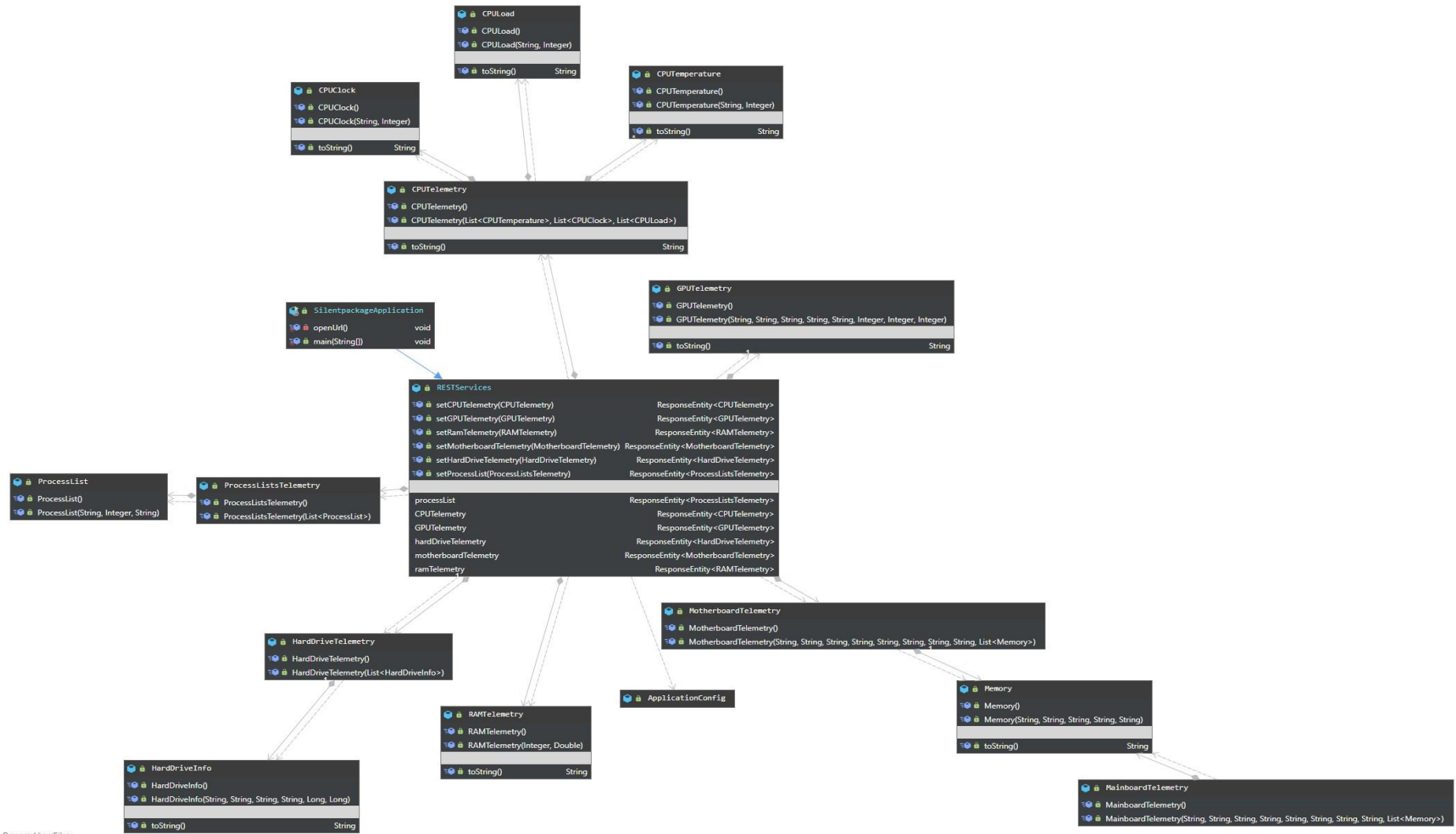
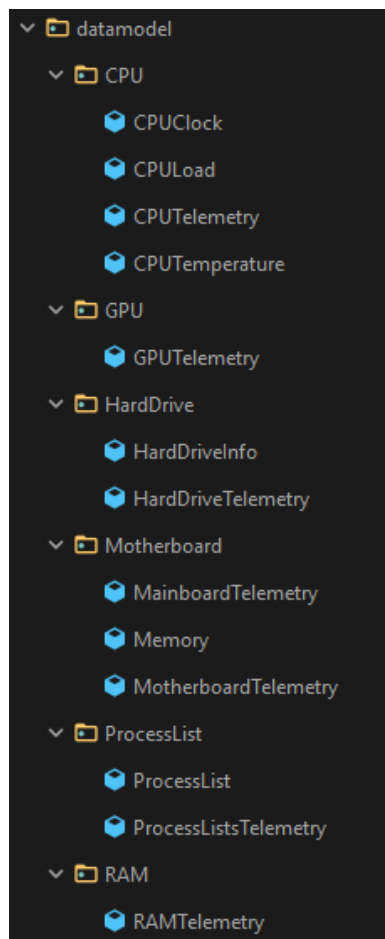


Diagram prezentuje warstwę prezentacji serwera. Warstwa ta powstała z wykorzystaniem języka HTML oraz JavaScript z wykorzystaniem biblioteki AngularJS. Strona pracuje według koncepcji jednej strony bez konieczności odświeżania danych. Proces odpytywania odbywa się poprzez zastosowanie techniki AJAX i wykonywaniu zapytań HTTP zgodnych z architekturą REST dane są dostarczone postaci dokumentu JSON. Główny plikiem jest index.html, który stanowi ciało dla reszty interfejsu graficznego, cała logika interfejsu graficznego zawarta jest w pliku app.js, czyli instancji AngularJS. Co określony czas wykonywana jest metoda w pliku app.js, która wykonuje zapytanie REST do warstwy logiki i pobiera ostatnio odebrane dane.

Diagram klas części serwerowej

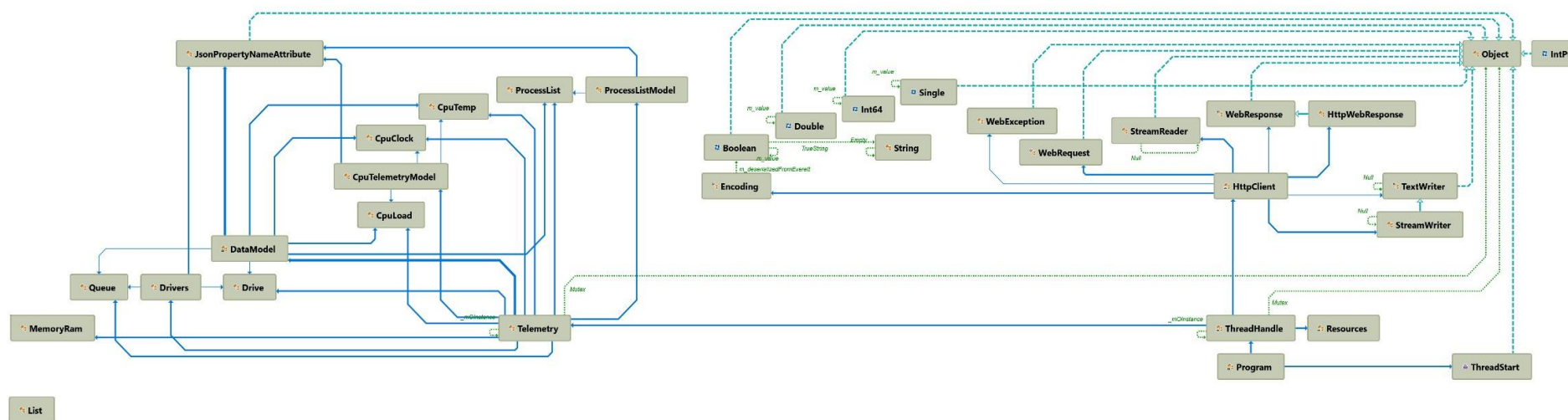


Zgodnie z modelem MVC aplikacja serwera została podzielona na klasy i pakiety. Główną klasą jest RESTservice, która znajduje się w pakiecie webController. Jej zadaniem jest odebranie danych od klienta i utworzenie endpointów dla warstwy prezentacji. Endpointy odpowiedzialne za przyjmowanie danych zostały utworzone zgodnie z architekturą REST, przyjmują one dane typu JSON i połączenia typu POST. W momencie dostarczenia telemetry dane w postaci JSON są podawane walidacji i zapisywane do obiektu. Interfejs graficzny wykorzystuje zapytania REST typu GET, które posiadają własne endpointy. Ich zadaniem jest pobrać obiekt, utworzyć z nich plik JSON i go zwrócić. W pakiecie dataModel znajdują się wszystkie klasy, które są podawane serializacji bądź deserializacji w zależności od wykonanego endpointu. Są one dietetyczne jak obiekty zawarte w modelu znajdującym się w kliencie. Każda klasa posiada mechanizm walidacji danych co zapobiega problemów z działaniem aplikacji.



Najbardziej skomplikowane modele takie jak telemetry CPU została podzielona na cztery klasy. Wynika to z tego, że w klasie CPUtelemetry znajdują się trzy listy, aby zapewnić lepszą przejrzystość danych zdecydowałem się aby listy przechowywały obiekty odpowiednich wartości telemetrycznych. Dzięki temu warstwa prezentacji może w szybszy i wydajniejszy sposób prezentować dane.

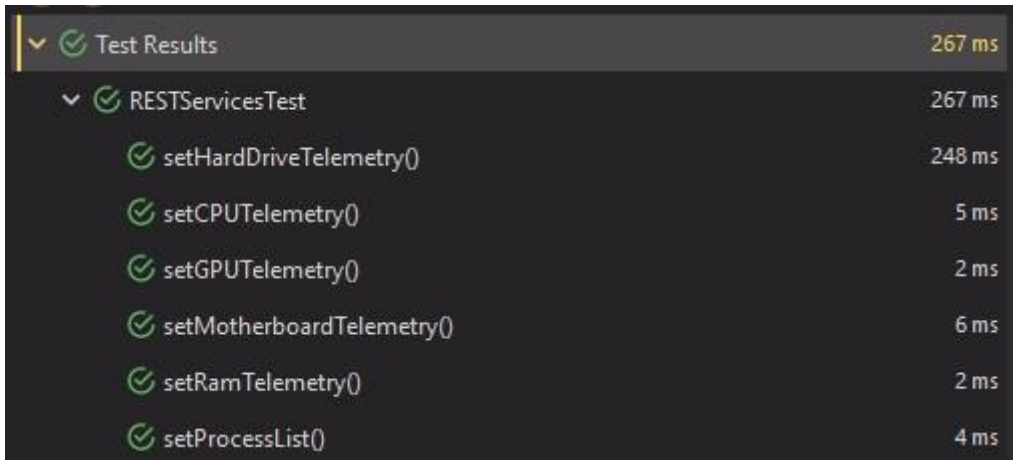
Diagram klas i zależności części projektu odpowiedzialnej za generowanie telemetrii (klient)



Klient podobnie jak serwer został podzielony na dwa główne bloki, pierwszym z nich jest Models, gdzie są przechowywane wszystkie modele danych, które następnie są serializowane do plików JSON. Drugim blokiem jest Controllers, gdzie znajdują się klasy odpowiedzialne za pobieranie telemetrii i wysyłanie ich na serwer. Główną klasą w kliencie jest klasa Telemetry. Jej zadaniem jest pobranie telemetrii systemu Windows 10 odpowiednimi metodami. Następnie przygotowane obiekty są wysyłane do klasy HttpClient, która wykonuje proces wysyłania danych na serwer. Całym procesem gromadzenia i wysyłania telemetrii zarządza klasa ThreadHandle, która uruchamia wątek i wykonuje proces co sekundę.

9. Testy

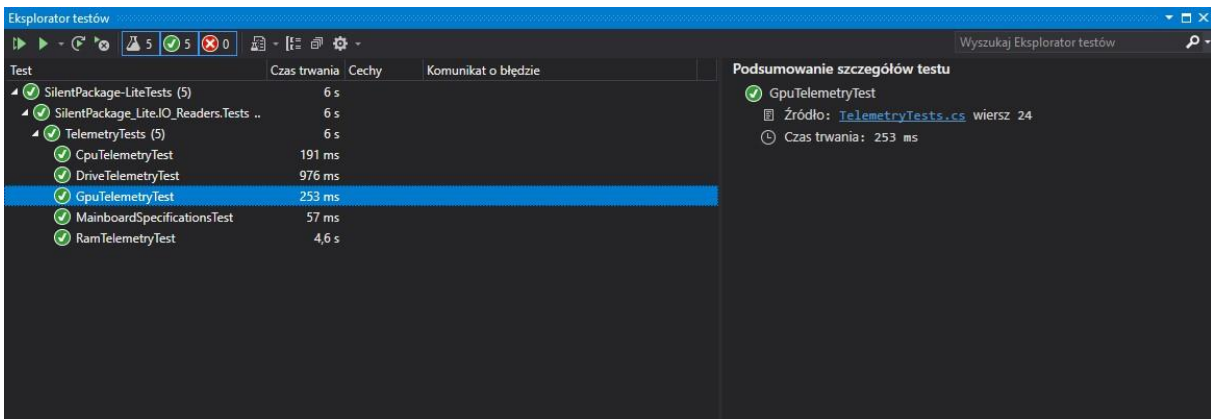
Podczas realizacji projektu przeprowadzono testy jednostkowe oraz funkcjonalne w ramach czarnej skrzynki w celu wyeliminowania potencjalnych problemów.



A screenshot of a 'Test Results' window in a dark-themed IDE. It shows a tree view under 'Test Results' (267 ms) with a sub-entry 'RESTServicesTest' (267 ms). Under 'RESTServicesTest', there are seven methods, each with a green checkmark icon and its execution time: 'setHardDriveTelemetry()' (248 ms), 'setCPUTElemetry()' (5 ms), 'setGPUTElemetry()' (2 ms), 'setMotherboardTelemetry()' (6 ms), 'setRamTelemetry()' (2 ms), and 'setProcessList()' (4 ms).

Test Results	267 ms
RESTServicesTest	267 ms
setHardDriveTelemetry()	248 ms
setCPUTElemetry()	5 ms
setGPUTElemetry()	2 ms
setMotherboardTelemetry()	6 ms
setRamTelemetry()	2 ms
setProcessList()	4 ms

Rysunek 5 Wynik testów jednostkowych dla serwera



A screenshot of the 'Eksplorator testów' (Test Explorer) window. The left pane shows a tree of tests: 'SilentPackage-LiteTests (5)' (6 s), 'SilentPackage_LiteIO_Readers.Tests ..' (6 s), 'TelemetryTests (5)' (6 s), 'CpuTelemetryTest' (191 ms), 'DriveTelemetryTest' (976 ms), 'GpuTelemetryTest' (253 ms, highlighted), 'MainboardSpecificationsTest' (57 ms), and 'RamTelemetryTest' (4,6 s). The right pane shows details for 'GpuTelemetryTest', indicating it passed (green checkmark), source is 'TelemetryTests.cs' line 24, and duration is 253 ms.

Test	Czas trwania	Cechy	Komunikat o błędzie
✓ SilentPackage-LiteTests (5)	6 s		
✓ SilentPackage_LiteIO_Readers.Tests ..	6 s		
✓ TelemetryTests (5)	6 s		
✓ CpuTelemetryTest	191 ms		
✓ DriveTelemetryTest	976 ms		
✓ GpuTelemetryTest	253 ms		
✓ MainboardSpecificationsTest	57 ms		
✓ RamTelemetryTest	4,6 s		

Podsumowanie szczegółów testu

- ✓ GpuTelemetryTest
- 📄 Źródło: [TelemetryTests.cs](#) wiersz 24
- 🕒 Czas trwania: 253 ms

Rysunek 6 Wynik testów jednostkowych dla klienta.

10. Podsumowanie projektu

Podczas realizacji projektu nie napotkano większych. Dzięki wykorzystaniu nowoczesnych technologii, wzorów projektowych takich jak Singleton oraz Fasada stworzono projekt kompaktowy i łatwy do rozwinięcia w przyszłości. Chciałbym podziękować społeczności Stack Overflow za niezliczoną pomoc.