|  |  |  |
| --- | --- | --- |
| **Politechnika Świętokrzyska w Kielcach**  **Wydział Elektroniki, Automatyki i Informatyki** | | |
| Projekt: **Programowanie w Java** | | |
| Grupa:  **2ID14B** | Temat:  **nzOS – Alternatywne oprogramowanie do kontroli chłodzenia wodnego i jego oświetlenia** | Skład grupy:  **Michał Młodawski**  **Konrad Nowakowski** |
| Rok studiów:  **2** |

**nzOS - Alternatywne oprogramowanie do kontroli chłodzenia wodnego i jego oświetlenia**

Opracowanie i sprawozdanie projektu nzOS

Maj 2018

Wersja dokumentu 21052018-002PL

Spis treści

[1. Opis projektu 1](#_Toc514700585)

[2. Wykorzystane technologie 1](#_Toc514700586)

[1. Użyte języki programowania oraz formaty tekstu 1](#_Toc514700587)

[2. Wykorzystane oprogramowanie przy projektowaniu i wdrażaniu projektu 1](#_Toc514700588)

[3. Użyte biblioteki w projekcie 2](#_Toc514700589)

[1. Biblioteki zewnętrzne 2](#_Toc514700590)

[2. Biblioteki wewnętrzne 2](#_Toc514700591)

[4. Funkcjonalność projektu 3](#_Toc514700592)

[5. Obsługa projektu i sposób jego uruchomienia 3](#_Toc514700593)

[1. Korzystanie z projektu 3](#_Toc514700594)

[2. Budowanie projektu 3](#_Toc514700595)

[1. Proces kompilacji aplikacji 3](#_Toc514700596)

[6. Wygląd interfejsu webowego 5](#_Toc514700597)

[1. Strona główna. 5](#_Toc514700598)

[2. Wygląd strony z ustawieniami oświetlenia. 6](#_Toc514700599)

[3. Wygląd strony z ustawieniami chłodzenia. 7](#_Toc514700600)

[4. Wygląd strony z informacjami o projekcie. 8](#_Toc514700601)

[5. Wygląd strony błędu (Animowana) 9](#_Toc514700602)

[6. Powiadomienia o przekroczonej temperaturze 10](#_Toc514700603)

[7. Protokół komunikacyjny dla chłodzenia 11](#_Toc514700604)

[1. Protokół zmiany prędkości wentylatorów 11](#_Toc514700605)

[2. Protokół zmiany prędkości pracy pompy 11](#_Toc514700606)

[3. Protokół zmiany oświetlenia chłodzenia 11](#_Toc514700607)

[4. Protokół zmiany trybów wyświetlania barw 12](#_Toc514700608)

[5. Protokół odpowiedzi chłodzenia 13](#_Toc514700609)

[8. Opis poszczególnych klas i metod 14](#_Toc514700610)

[1. Opis klas 14](#_Toc514700611)

[2. Opis metod 14](#_Toc514700612)

[1. Klasa PreDataBase.java 14](#_Toc514700613)

[2. Klasa FileManagement.java 17](#_Toc514700614)

[3. Klasa ApiManagment.java 20](#_Toc514700615)

[4. Klasa ExceptionApiManagment.java 21](#_Toc514700616)

[5. Klasa ApiMonitoring.java 21](#_Toc514700617)

[6. Klasa Api.java 21](#_Toc514700618)

[7. Klasa CurrentValue.java 22](#_Toc514700619)

[8. Klasa CurrentValueController.java 24](#_Toc514700620)

[9. Klasa NzOsApplication.java 26](#_Toc514700621)

[9. Podsumowanie projektu 27](#_Toc514700622)

# Opis projektu

Projekt nzOS powstał w celu wykorzystania możliwość nowych technologii połączonych razem ze sprzętami peryferyjnymi marki NZXT**®**. Dodatkowym założeniem projektu było napisanie aplikacji webowej, która będzie komunikować się przy pomocy standardu USB (Universal Serial Bus) z chodzeniem wodnym.

# Wykorzystane technologie

W naszym projekcie wykorzystaliśmy najnowocześniejsze technologie dla osiągnięcia jak najlepszych walorów estetycznych aplikacji webowej i szybkości działania chłodzenia wodnego.

## Użyte języki programowania oraz formaty tekstu

1. Apache Maven jako narzędzie do automatyzacji budowy projektu.
2. HTML z wykorzystaniem frameworka Bootstrap, JavaScript z frameworkiem jQuery oraz AngularJS.
3. Java z rozszerzeniem Spring Boot po stronie serwera.
4. JSON jako format tekstowy dla przejrzystego uporządkowania danych w pliku konfiguracyjnym.

## Wykorzystane oprogramowanie przy projektowaniu i wdrażaniu projektu

1. Adobe XD CC w celu szybkiego prototypowanie interfejsu graficznego.
2. Device Monitoring Studio do analizowania ruchu między chłodzeniem a oprogramowaniem w celu inżynierii odwrotnej protokołu komunikacyjnego.
3. IntelliJ IDEA jako główne IDE do programowanie serwera.
4. JavaDoc do prowadzenia dokumentacji kodu.
5. JetBrains WebStorm jako IDE do interfejsu graficznego.

# Użyte biblioteki w projekcie

## Biblioteki zewnętrzne

|  |  |  |  |
| --- | --- | --- | --- |
| **LP.** | **Nazwa biblioteki** | **Licencja** | **Opis** |
| 1. | Apache Tomcat | Apache License 2.0 | Kontener aplikacji webowych. |
| 2. | jSensors | Apache License 2.0 | Biblioteka pozwalająca odczytywanie informacji o CPU. |
| 3. | JSON.simple | Apache License 2.0 | Prosty zestaw narzędzi do obsługi plików JSON. |
| 4. | log4j | Apache License 2.0 | Biblioteka służąca do tworzenia logów podczas działania aplikacji. |
| 5. | Spring Boot | GNU Lesser General Public | Framework Java dodający możliwość tworzenia webowych aplikacji. |
| 6. | Angular.js | MIT | Framework wspomagający tworzenie i rozwój aplikacji internetowych na pojedynczej stronie. |
| 7. | Bootstrap | MIT | Framework CSS dodający przydatne elementy do projektowania stron. |
| 8. | Chart.js | MIT | Biblioteka dodaje możliwość sterowania diodami LED. |
| 9. | Jquery | MIT | Biblioteka programistyczna dla języka JavaScript, ułatwiająca korzystanie z JavaScriptu. |
| 10. | Popper.js | MIT | Biblioteka programistyczna dla języka JavaScript, ułatwiająca korzystanie z modali. |
| 11. | usb4java | MIT | Biblioteka pozwalająca komunikować się z interfejsem USB za pomocą JNA. |
| 12. | Font Awesome | Licencja wewnętrzna:  https://fontawesome.com/  license | Zestaw piktogramów przedstawionych w formacie czcionki. |
| 13. | junit5 | Eclipse Public License 2.0 | Narzędzie służące do tworzenia powtarzalnych testów jednostkowych. |

Tabela 1 Biblioteki zewnętrzne wykorzystane w projekcie.

## Biblioteki wewnętrzne

1. cookie.js – zarządzanie ciasteczkami.
2. fast.js – szybkie odczytywanie plików testowych z poziomu przeglądarki.

# Funkcjonalność projektu

Serwerowa część aplikacji nzOS umożliwia za pomocą standardu komunikacyjnego USB, na komunikację z chłodzeniem wodnym. Sterowanie odbywa się za pomocą protokołu dostępnego w [**Tabeli 2**](#_Mikrokod_rozkazu_dla)**,** [**Tebeli 3**](#_Protokół_zmiany_prędkości)oraz[**Tebli 4**](#_Protokół_zmiany_oświetlenia)**.**

# Obsługa projektu i sposób jego uruchomienia

## Korzystanie z projektu

Aby móc korzystać z projektu należy posiadać oprogramowanie Java przynajmniej z **wersji 10 oraz system operacyjny Windows® firmy Microsoft**.

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

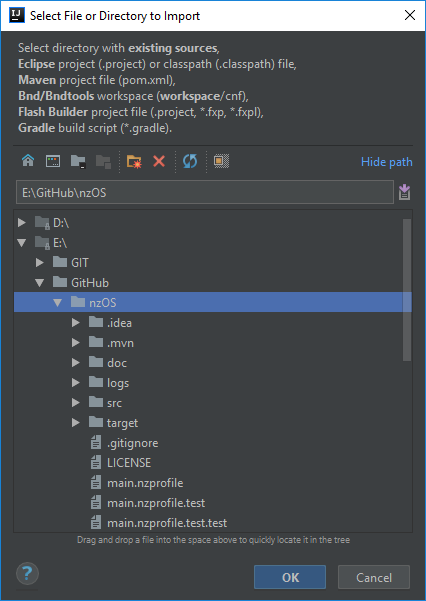
|  |
| --- |
| **Informacja!** Pełne działanie aplikacji wymaga posiadanie chłodzenia marki NZXT® minimum w wersji x62. |

## Budowanie projektu

Aby móc wybudować projekt należy posiadać zintegrowane środowisko programistyczne np. IntelliJ IDEA.

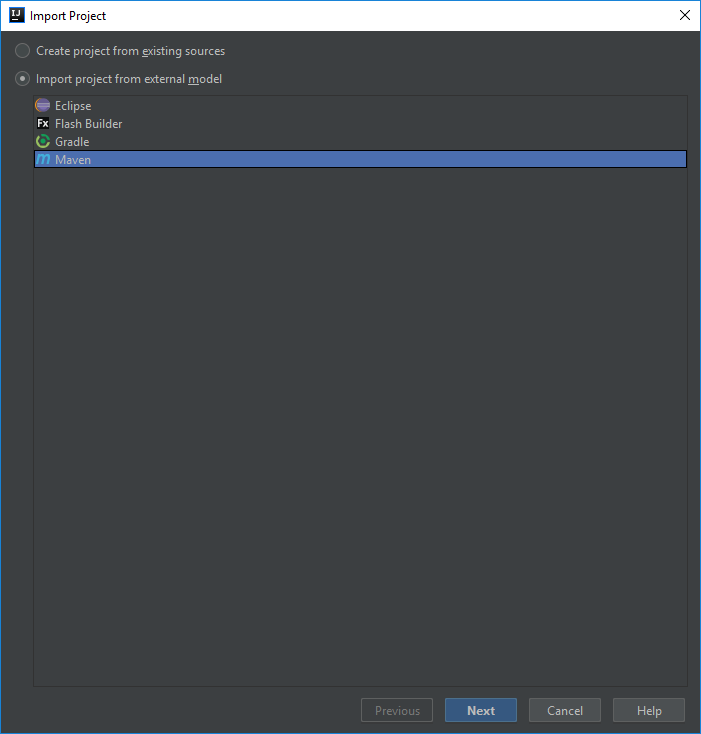
### Proces kompilacji aplikacji

Pierwszym krokiem jest importowanie projektu z istniejących źródeł:

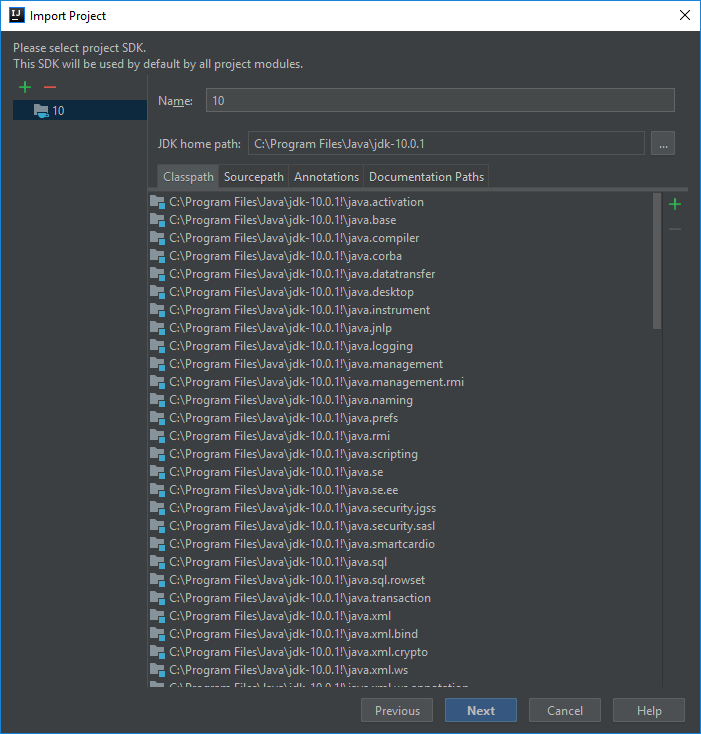


Rysunek 1Poprawne wybranie folderu z istniejących źródeł.

Po skompilowaniu projektu należy użyć narzędzia **Maven.** Służydo zautomatyzowania procesu tworzenia zależności wymaganych do uruchomienia. Tworzy piaskownicę dla środowiska wykonawczego systemu Windows lub drzewo instalacji dla aplikacji pulpitu systemu Windows, które można łatwo dołączyć do pakietu instalacyjnego.



Rysunek 2 Wybranie narzędzia do tworzenia zależności.

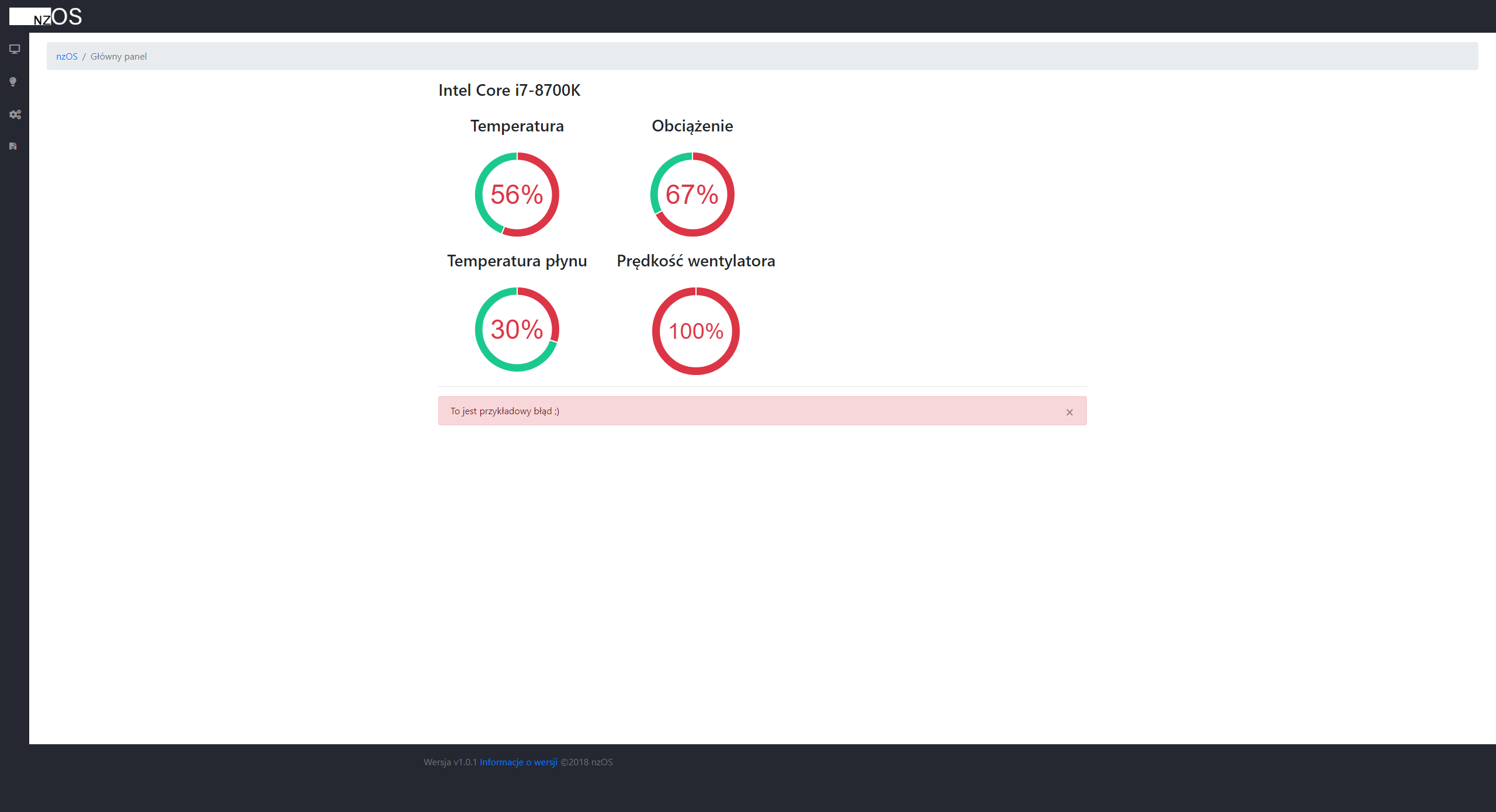


Rysunek 3 Wybór SDK dla projektu

Po tych trzech krokach powinniśmy uzyskać w pełni skompilowaną i gotową do uruchomienia aplikację.

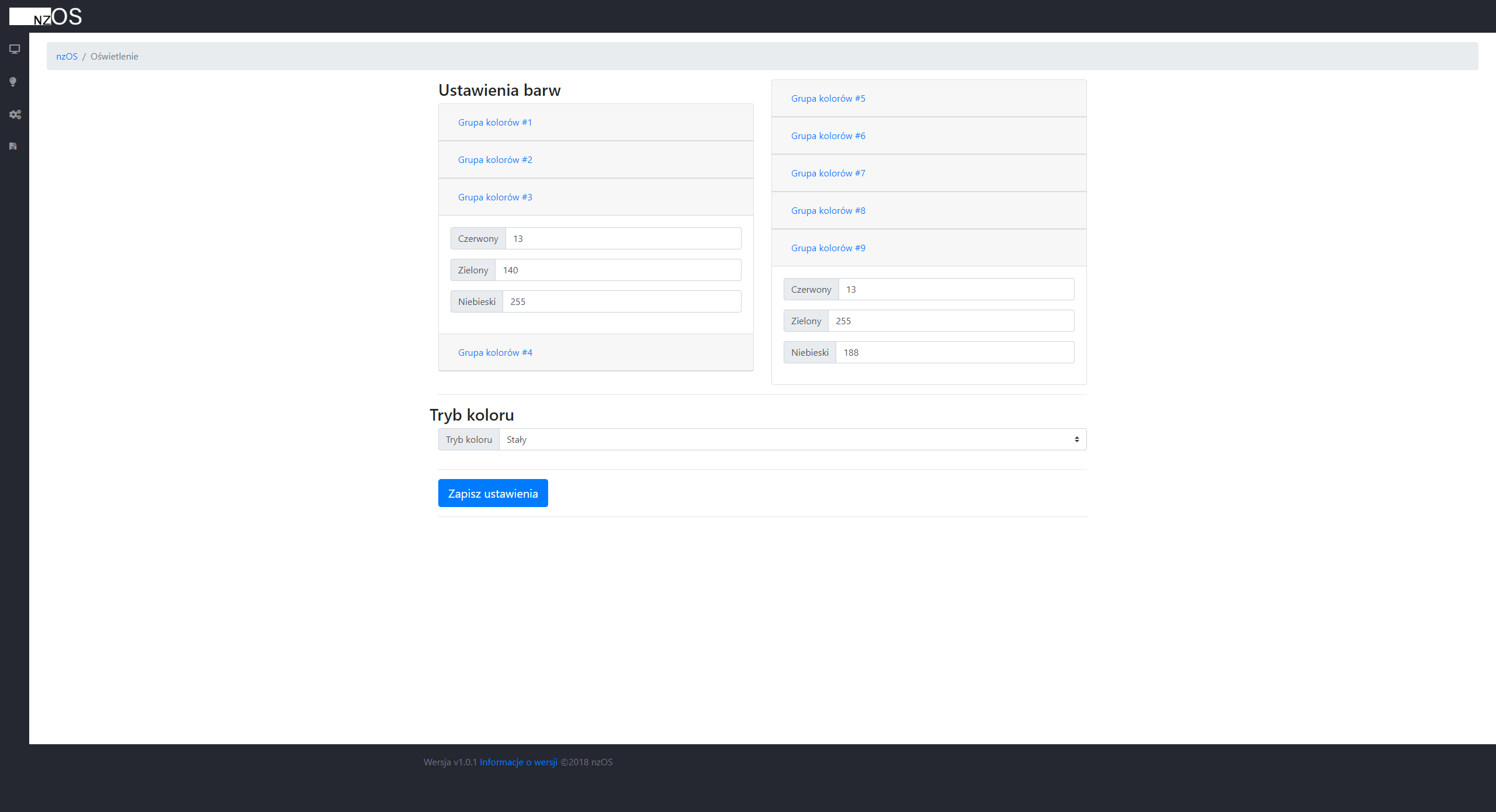
# Wygląd interfejsu webowego

## Strona główna.



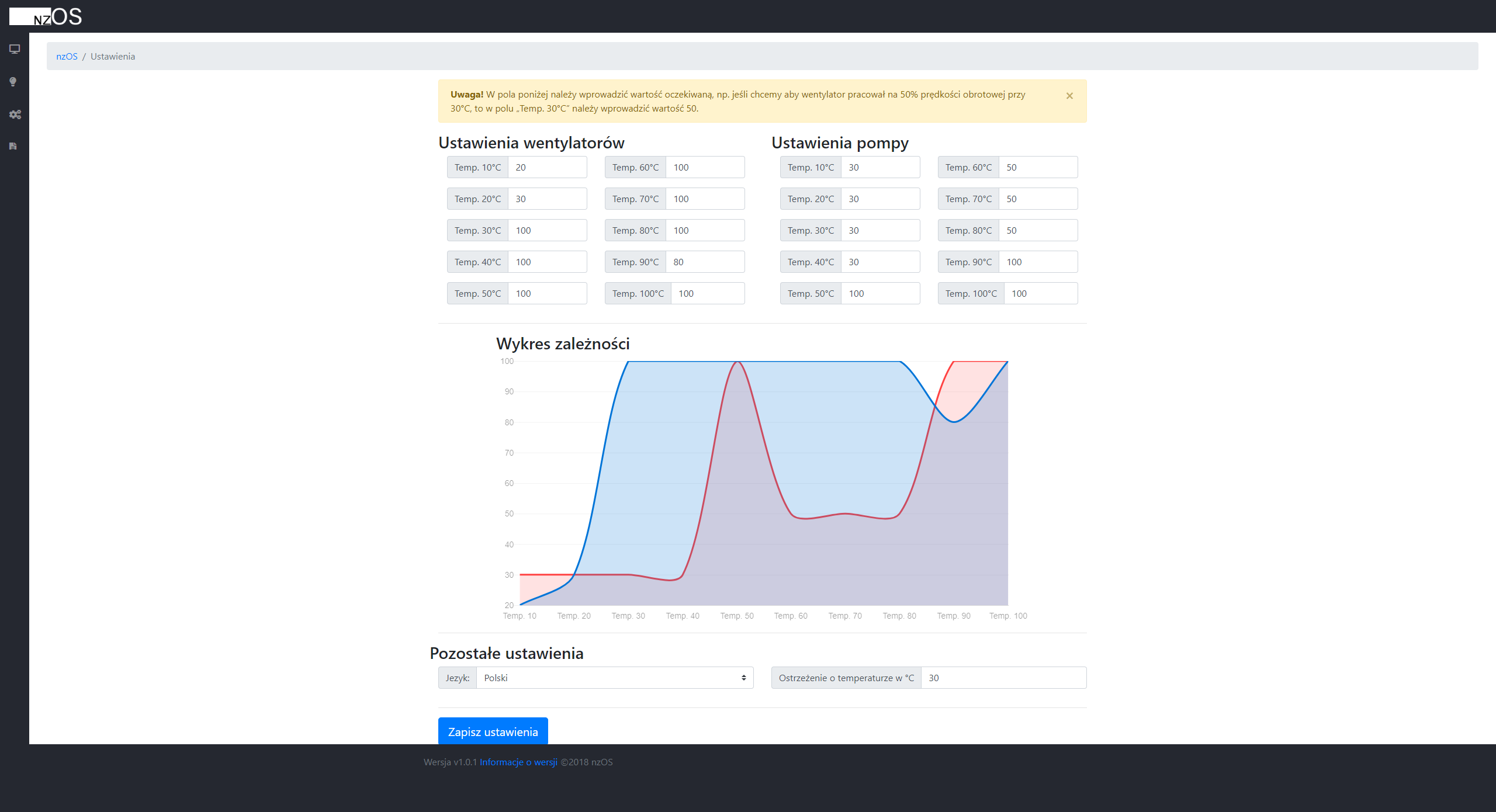
Rysunek 4 Wygląd strony głównej.

## Wygląd strony z ustawieniami oświetlenia.



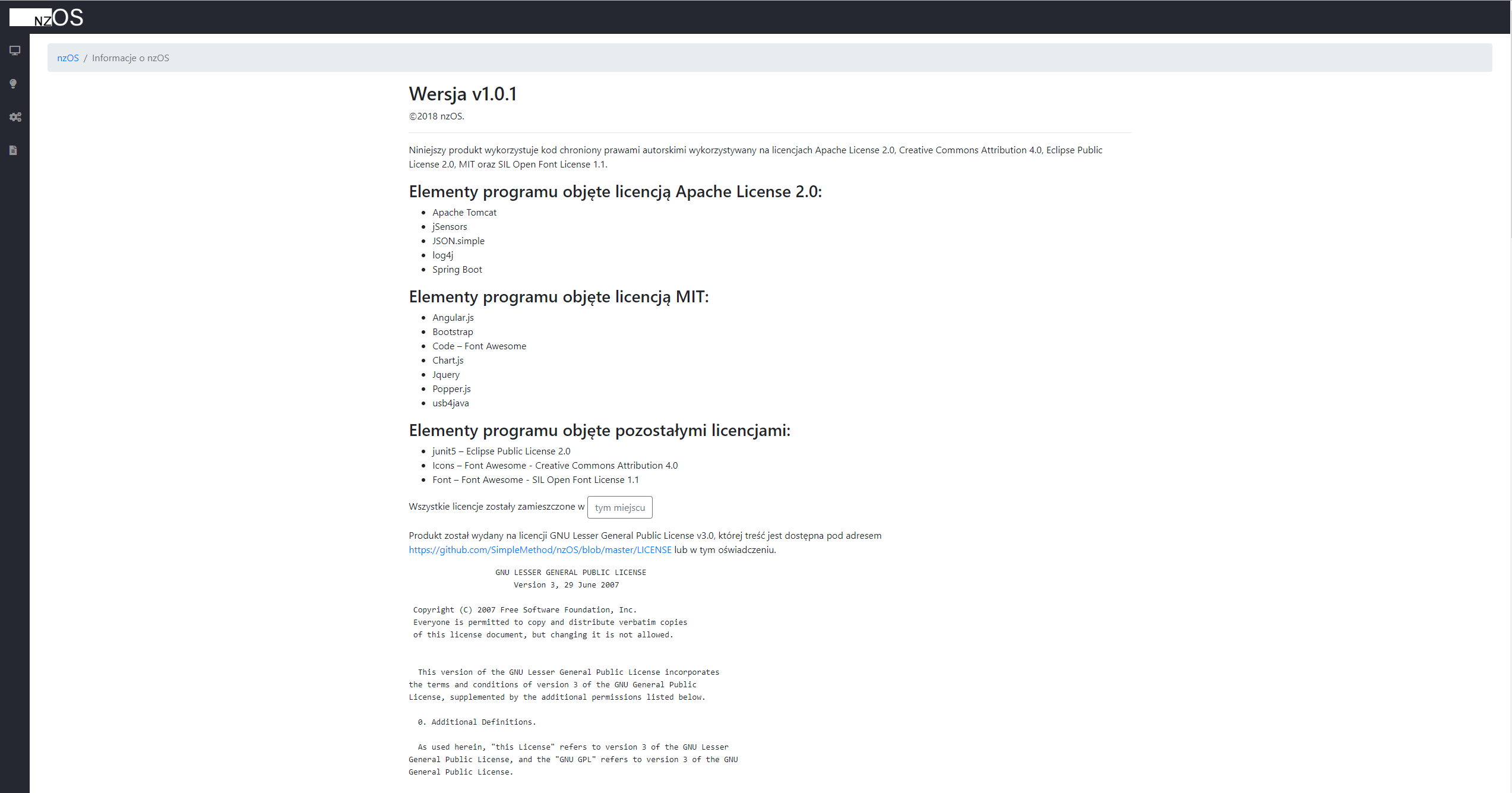
Rysunek 5 Wygląd strony z ustawieniami oświetlenia.

## Wygląd strony z ustawieniami chłodzenia.



Rysunek 6 Wygląd strony z ustawieniami chłodzenia.

## Wygląd strony z informacjami o projekcie.



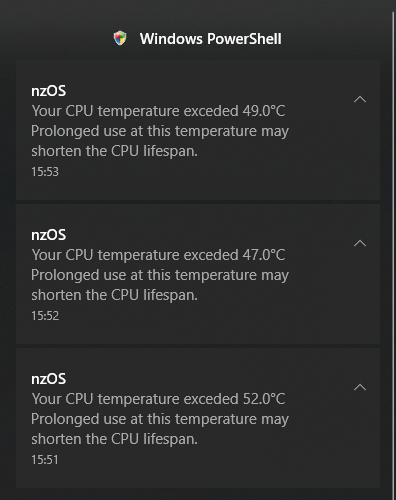
Rysunek 7 Wygląd strony z informacjami o projekcie.

## Wygląd strony błędu (Animowana)



Rysunek 8 Strona błędu, źródło animacji: https://coub.com/view/131ozl

## Powiadomienia o przekroczonej temperaturze



Rysunek 9 Powiadomienia o przekroczonej temperaturze.

# Protokół komunikacyjny dla chłodzenia

## Protokół zmiany prędkości wentylatorów

### 

|  |  |  |  |
| --- | --- | --- | --- |
| **Lp.** | **Numer bajtów** | **Wartość** | **Opis** |
| 1. | 0 | 0x02 | Początkowy bajt inicjujący komunikację. |
| 2. | 1 | 0x4d | Dwa bajty sygnalizujące sterowanie wentylatorami. |
| 3. | 2 | 0x00 |
| 4. | 3 | ---- | Wartość sterowania od 0x14 do 0x64. |

Tabela 2 Opis protokołu dla komunikacji z wentylatorami.

## Protokół zmiany prędkości pracy pompy

|  |  |  |  |
| --- | --- | --- | --- |
| **Lp.** | **Numer bajtów** | **Wartość** | **Opis** |
| 1. | 0 | 0x02 | Początkowy bajt inicjujący komunikację. |
| 2. | 1 | 0x4d | Dwa bajty sygnalizujące sterowanie pompą. |
| 3. | 2 | 0x40 |
| 4. | 3 | ---- | Wartość sterowania od 0x14 do 0x64. |

Tabela 3 Opis protokołu dla komunikacji z wentylatorami.

## Protokół zmiany oświetlenia chłodzenia

|  |  |  |  |
| --- | --- | --- | --- |
| **Lp.** | **Numer bajtów** | **Wartość** | **Opis** |
| 1. | 0 | 0x02 | Początkowy bajt inicjujący komunikację. |
| 2. | 1 | 0x4c | Dwa bajty sygnalizujące sterowanie oświetleniem. |
| 3. | 2 | 0x00 |
| 4. | 3 | ---- | Tryb wyświetlania barw. Więcej szczegółów w [Tabeli 5](#_Protokół_zmiany_trybów). |
| 5. | 4 | ---- |
| 6. | 5-31 | ---- | Kolory zapisane w postaci R, G, B. |

Tabela 4 Opis protokołu dla komunikacji z oświetleniem.

## Protokół zmiany trybów wyświetlania barw

|  |  |  |  |
| --- | --- | --- | --- |
| **Lp.** | **Numer bajtów** | **Wartość** | **Tryb pracy** |
| 1. | 0 | 0x04 | Stały kolor. |
| 2. | 1 | 0x02 |
| 3. | 0 | 0x06 | Oddychanie. |
| 4. | 1 | 0x02 |
| 5. | 0 | 0x07 | Pulsowanie. |
| 6. | 1 | 0x02 |
| 7. | 0 | 0x02 | Randomizacja barw. |
| 8. | 1 | 0x01 |
| 9. | 0 | 0x00 | Stały alternatywny. |
| 10. | 1 | 0x02 |
| 11. | 0 | 0x01 | Stały jedna barwa (Dane z koloru #2) |
| 12. | 1 | 0x02 |
| 13. | 0 | 0x0D | Fala - Najwolniejszy poziom. |
| 14. | 1 | 0x00 |
| 15. | 0 | 0x0D | Fala - Wolny poziom. |
| 16. | 1 | 0x01 |
| 17. | 0 | 0x0D | Fala - Normalny poziom. |
| 18. | 1 | 0x02 |
| 19. | 0 | 0x0D | Fala - Szybki poziom. |
| 20. | 1 | 0x03 |
| 21. | 0 | 0x0D | Fala - Najszybszy poziom. |
| 22. | 1 | 0x04 |
| 23. | 0 | 0x06 | Oddychanie - Najwolniejszy poziom |
| 24. | 1 | 0x00 |
| 25. | 0 | 0x06 | Oddychanie - Wolny poziom |
| 26. | 1 | 0x01 |
| 27. | 0 | 0x06 | Oddychanie - Normalny poziom |
| 28. | 1 | 0x02 |
| 29. | 0 | 0x06 | Oddychanie - Szybki poziom |
| 30. | 1 | 0x03 |
| 31. | 0 | 0x06 | Oddychanie - Najszybszy poziom |
| 32. | 1 | 0x04 |
| 33. | 0 | 0x00 | Stały jedna barwa |
| 34. | 1 | 0x01 |

Tabela 5 Opis protokołu dla trybu pracy.

## Protokół odpowiedzi chłodzenia

|  |  |  |  |
| --- | --- | --- | --- |
| **Lp.** | **Numer bajtów** | **Przykładowe wartość** | **Opis** |
| 1. | 0 | 0x04 | Suma kontrolna odpowiedzi. |
| 2. | 1 | 0x1F | Temperatura płynu chłodniczego. |
| 3. | 2 | 0x08 | ---- |
| 4. | 3 | 0x08 | Pierwsza część odczytu prędkości wentylatorów. |
| 5. | 4 | 0x67 | ---- |
| 6. | 5 | 0x0C | ---- |
| 7. | 6 | 0xB4 | ---- |
| 8. | 7 | 0x00 | ---- |
| 9. | 8 | 0x00 | ---- |
| 10. | 9 | 0x00 | ---- |
| 11. | 10 | 0x4E | Druga część odczytu prędkości wentylatorów. |
| 12. | 11 | 0x03 | ---- |
| 13. | 12 | 0x00 | ---- |
| 14. | 13 | 0x00 | ---- |
| 15. | 14 | 0x05 | Ostatnia część odczytu prędkości wentylatorów. |
| 16. | 15 | 0x1E | Bajty oznaczające koniec transmisji. |
| 17. | 16 | 0x00 |

# Opis poszczególnych klas i metod

## Opis klas

1. PreDataBase.java – Klasa służąca jako kontener pól do przesyłania między klasami.
2. FileManagement.java – Klasa zajmuje się zarządzaniem konfiguracją pliku i obsługą pliku z logami.
3. ApiManagment.java – Klasa ma zadanie obsługiwanie protokołu komunikacyjnego, wątku aplikacji, wyświetlanie komunikatów, sprawdzanie uprawnień administratora i odczyt informacji z procesora.
4. ExceptionApiManagment.java – Klasa służąca jako wyjątek dla klasy ApiManagment.
5. ApiMonitoring.java – Klasa służąca jako starter dla wątku i wstępnej konfiguracji.
6. Api.java – Klasa służąca do wysyłania i odbierania informacji od chłodzenia.
7. CurrentValue.java – Klasa służąca do zapytań REST dla klasy CurrentValueController.
8. CurrentValueController.java – Klasa służąca jako kontroler dla Spring Boot.
9. NzOsApplication.java – Główna klasa obsługująca wywołanie wątku i uruchomienie serwera Tom Cat.

## Opis metod

### Klasa PreDataBase.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LP.** | **Metoda:** | **Argument przyjmujący:** | **Argument zwracany:** | **Opis:** |
| 1. | protected String getDEFAULT\_FILENAME() | Brak | Zwraca domyślną nazwę pliku. | Metoda zwracająca domyślną nazwę pliku. |
| 2. | protected static int getTIMEOUT() | Brak | Zwraca czas TIMEOUT dla urządzenia. | Metoda zawracają czas TIMEOUT dla urządzenia. |
| 3. | protected static void setCPUNAME(String CPUNAME) | Nazwę procesora. | Brak | Metoda zapisująca nazwę procesora. |
| 4. | protected static byte getInEndpoint() | Brak | Bajt wejściowy. | Metoda zawracają bajt wejściowy. |
| 5. | protected static byte getINTERFACE() | Brak | Numer interfejsu. | Metoda zwracająca numer interfejsu urządzenia. |
| 6. | protected static byte getOutEndpoint() | Brak | Bajt wyjściowy. | Metoda zawracają bajt wyjściowy. |
| 7. | public static byte[] getColorCustom() | Brak | Paleta kolorów. | Metoda zwraca paletę kolorów. |
| 9. | protected static byte[] getFanData() | Brak | Zwraca informacje o wentylatorze. | Metoda pobierająca informacje o wentylatorze. |
| 10. | public static byte[] getOutDump() | Brak | Dump informacji odebranych od chłodzenia. | Metoda zwraca dump informacji odebranych od chłodzenia. |
| 11. | protected static byte[] getPumpData() | Brak | Informacje o pracy pompy. | Metoda zwraca informacje o pracy pompy. |
| 12. | protected static List<Fan> getFANS() | Brak | Zwraca listę z wentylatorami. | Metoda zwracająca informacje o wentylatorach zamontowanych w komputerze. |
| 13. | protected static List<Load> getLOAD() | Brak | Zwraca listę użyciem rdzeni procesora. | Metoda zwraca obciążenie każdego z rdzeni procesora. |
| 14. | protected static List<Temperature> getTEMPS() | Brak | Zwraca listę z temperaturami procesora. | Metoda zwraca listę z temperaturami procesora. |
| 15. | protected static short getProductId() | Brak | Zwraca identyfikator chłodzenia. | Metoda zwraca identyfikator chłodzenia |
| 16. | protected static short getVendorId() | Brak | Vendor urządzenia. | Metoda zwraca vendor urządzenia. |
| 17. | protected static String getCPUNAME() | Brak | Zwraca nazwę CPU. | Metoda zwracająca nazwę procesora. |
| 18. | protected static String getDefaultFilename() | Brak | Nazwa pliku z konfiguracją. | Metoda pobierająca nazwę pliku z konfiguracjami. |
| 19. | public static void setDefaultFilename(String defaultFilename) | Parametr z nazwą pliku. | Brak | Metoda, która ustawia domyślną nazwę dla pliku z konfiguracją. |
| 20. | public static void setFanData(byte[] fanData) | fanData - zmienna do zapisania rozkazów. | Brak | Metoda do ustawienia prędkości wentylatorów. |
| 21. | protected static void setFANS(List<Fan> FANS) | FANS lista z wentylatorami. | Brak | Metoda do przypisania informacji o wentylatorach. |
| 22. | protected static void setLOAD(List<Load> LOAD) | LOAD lista z obciążeniami rdzeni CPU. | Brak | Metoda do przypisania obciążenia rdzeni CPU. |
| 23. | protected static void setOutDump(byte[] outDump) | outDump informacje odebrane od chłodzenia. | Brak | Metoda przypisania dumpu odebranych informacji. |
| 24. | public static void setPumpData(byte[] pumpData) | pumpData zmienna do zapisu. | Brak | Metoda zapisująca informacje odebrane od pompy. |
| 25. | protected static void setTEMPS(List<Temperature> TEMPS) | TEMPS zmienna do przypisania temp. | Brak | Metoda zapisująca listę z temperaturami rdzeni procesora. |
| 26. | protected static long getTEMP() | Brak | Zwraca temp. procesora. | Metoda zwracająca temperaturę procesora. |
| 27. | protected static long getFAN() | Brak | Zwraca prędkość wentylatorów. | Metoda zwracająca prędkość wentylatorów. |
| 28. | protected static long getLIQUID() | Brak | temperatura płynu chłodniczego. | Metoda zwracająca temperaturę płynu chłodniczego. |
| 29. | protected static void setFAN(long FAN) | FAN prędkość wentylatorów. | Brak | Metoda przypisania prędkości wentylatorów. |
| 30. | protected static void setLIQUID(long LIQUID) | LIQUID zmienna z temperaturą płynu chłodniczego. | Brak | Metoda przypisująca temperaturę płynu chłodniczego. |
| 31. | protected static void setTEMP(long TEMP) | TEMP parametr z temperaturą. | Brak | Metoda zapisująca średnią temperaturę CPU. |

Tabela 6 Opis metod w klasie PreDataBase.java

### Klasa FileManagement.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LP.** | **Metoda:** | **Argument przyjmujący:** | **Argument zwracany:** | **Opis:** |
| 1. | public T readingFile(String filename, String OBJECT) | filename Nazwa pliku do odczytu. OBJECT Nazwa pojedynczego obiektu do odczytu. | Zwraca wartość zapisaną w obiekcie. | Metoda służąca do czytania pojedynczych obiektów. |
| 2. | public T readingFile(String filename, String OBJECT, String ALTOBJECT) | ALTOBJECT Nazwa obiektu do odczytu. OBJECT Nazwa obiektu do odczytu. filename Nazwa pliku do odczytu. | Zwraca wartość zapisaną w obiekcie. | Metoda służąca do czytania obiektu w obiekcie. |
| 3. | public T readingFile(String filename, String OBJECT, String ALTOBJECT, String ALTALTOBJECT) | filename Nazwa pliku do odczytu. OBJECT Nazwa obiektu do odczytu. ALTOBJECT Nazwa obiektu do odczytu. ALTALTOBJECT Nazwa obiektu do odczytu. | Zwraca wartość zapisaną w obiekcie. | Metoda służąca do czytania obiektu w obiekcie w obiekcie (Obiekto incepcja). |
| 4. | public void writingFile(String filename, String OBJECT, T VALUE) | filename Nazwa pliku do odczytu. OBJECT Nazwa obiektu do zapisu. VALUE Wartość do zapisu. | Brak | Metoda służąca do zapisania obiektu. |
| 5. | public void writingFile(String filename, String OBJECT, String ALTOBJECT, T VALUE) | filename Nazwa pliku do odczytu. OBJECT Nazwa obiektu do zapisu. ALTOBJECT Nazwa obiektu do odczytu. VALUE Wartość do zapisu. | Brak | Metoda służąca do zapisania obiektu w obiekcie. |
| 6. | public void writingFile(String filename, String OBJECT, String ALTOBJECT, String ALTALTOBJECT, T VALUE) | filename Nazwa pliku do odczytu. OBJECT Nazwa obiektu do zapisu. ALTOBJECT Nazwa obiektu do odczytu. ALTALTOBJECT Nazwa obiektu do odczytu. VALUE Wartość do zapisu. | Brak | Metoda służąca do zapisania obiektu w obiekcie w obiekcie. |
| 7. | public void writingFile(String filename, JSONObject JSON\_OBJECT) | filename Nazwa pliku do odczytu. JSON\_OBJECT Wskaźnik na obiekt Json. | Brak | Metoda służąca do zapisu pliku. |
| 9. | public void writingFile(String filename, String TEXT) | filename Nazwa pliku do odczytu. TEXT Tekst do zapisu. | Brak | Metoda służąca do zapisu pliku. |
| 10. | public byte[] colorArray(String filename) | filename Nazwa pliku do odczytu. | zwraca tablicę z kolorami | Metoda służąca do odczytu barw i zapisu jej do tablicy. |
| 11. | public long[] colorLongArray(String filename) | filename Nazwa pliku do odczytu. | Zwraca listę z kolorami. | Metoda zwracająca tablicę z barwami odczytaną z pliku z konfiguracją. |
| 12. | public List<T> arrayToList(String filename, String OBJECT, String ALTOBJECT) | ALTOBJECT Nazwa obiektu do pobrania. OBJECT Nazwa obiektu do pobrania. filename Nazwa pliku do odczytu. | Lista z obiektami. | Metoda służąca do zamiany tablicy na listę. |
| 13. | public List<T> arrayToList(String filename, String OBJECT) | OBJECT Nazwa obiektu do pobrania. filename Nazwa pliku do odczytu. | Lista z obiektami. | Metoda służąca do zamiany tablicy na listę. |
| 14. | public ArrayList<String> showLogFile(String filename) | filename Nazwa pliku do odczytu. | Zwraca tekst w postaci linii z błędami. | Metoda odczytująca plik z logami błędów. |
| 15. | public void forceWritingFile(String filename, String OBJECT, T VALUE) | filename Nazwa pliku do odczytu. OBJECT Obiekt do edycji. VALUE Wartość do edycji. | Brak | Metoda zapisująca docelowy kolor w pliku konfiguracyjnym. |
| 16. | public void forceWritingFile(String filename, String OBJECT, String Type, T VALUE) | filename Nazwa pliku do odczytu. OBJECT Obiekt do edycji. Type Typ koloru do edycji. VALUE Wartość koloru do edycji. | Brak | Metoda zapisująca docelowe kolory w pliku konfiguracyjnym. |

Tabela 7 Opis metod w klasie FileManagement.java

### Klasa ApiManagment.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LP.** | **Metoda:** | **Argument przyjmujący:** | **Argument zwracany:** | **Opis:** |
| 1. | public void changingPumpSpeed(Long VALUE) | VALUE Wartość w zakresie 60-100. | Brak | Metoda służąca zmiany pracy pompy. |
| 2. | public void changingFanSpeed(Long value) | value Wartość w zakresie 25-100. | Brak | Metoda służąca zmiany pracy wentylatorów. |
| 3. | public void changingColor(Integer colorMode) | colorMode Tryb zmiany kolorów. | Brak | Metoda służąca do zmiany braw. |
| 4. | public List getCpuTemps() | Brak | Zwraca listę zawierającą temperatury rdzeni procesora. | Metoda służąca do zwracania temperatury CPU. |
| 5. | public void getCpuInfo() | Brak | Brak | Metoda służąca do zwracania informacji o CPU. |
| 6. | public Double getCpuTemp() | Brak | Zwraca wartość zawierającą uśrednioną temperaturę procesora. | Metoda służąca do zwracania temperatury CPU. |
| 7. | public void theardHelper() | Brak | Brak | Metoda służąca monitorowania pracy nzOS. |
| 9. | public void sendNotification(String title, String subtitle) throws Exception | title Tytuł powiadomienia subtitle Treść powiadomienia Exception Wyjatek przy powiadomieniu | Brak | Metoda służacado wyświetlania powiadomień. |
| 10. | public boolean isAdmin() | Brak | Zwraca prawdę w momencie, gdy aplikacja została uruchomiona z prawami administratora. | Metoda sprawdza, czy aplikacja została uruchomiona z prawami administratora. |

Tabela 8 Opis metod w klasie ApiManagment.java

### Klasa ExceptionApiManagment.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LP.** | **Metoda:** | **Argument przyjmujący:** | **Argument zwracany:** | **Opis:** |
| 1. | public ExceptionApiManagment() | Brak | Brak | Konstruktor klasy. |
| 2. | public ExceptionApiManagment(String message) | message komunikat o błędzie. | Brak | Konstruktor klasy z obsługą tekstu. |
| 3. | public ExceptionApiManagment(String message, Throwable cause) | message komunikat o błędzie. cause wyjątek. | Brak | Konstruktor klasy z obsługą tekstu i zgłaszaniem wyjątku. |
| 4. | public ExceptionApiManagment(Throwable cause) | cause wyjątek. | Brak | Konstruktor klasy z zgłaszaniem wyjątku. |

Tabela 9 Opis metod z klasy ExceptionApiManagment.java

### Klasa ApiMonitoring.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LP.** | **Metoda:** | **Argument przyjmujący:** | **Argument zwracany:** | **Opis:** |
| 1. | public void startTheard() | Brak | Brak | Metoda służąca do startu wątku. |
| 2. | public void run() | Brak | Brak | Metoda do obsługi wątku. |

Tabela 10 Opis metod w klasie ApiMonitoring.java

### Klasa Api.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LP.** | **Metoda:** | **Argument przyjmujący:** | **Argument zwracany:** | **Opis:** |
| 1. | public void writeToDevice(byte[] data) | data Dane do wysłania do urządzenia. | Brak | Metoda służąca do wysłania bajtów. |
| 2. | public ByteBuffer readDataFromDevice(int size) | size Rozmiar ramki do odczytu. | Zwraca odebrane bajty. | Metoda służąca do odbieranie danych. |
| 3. | public long getLiquidTemp() | Brak | Zwraca temp. płynu. | Metoda służąca do zwracania temperatury płynu. |
| 4. | public long getFanSpeed() | Brak | Zwraca wartości obrotów wentylatorów. | Metoda służąca do zwracania wartości obrotów wentylatorów. |

Tabela 11 Opis metod w klasie Api.java

### Klasa CurrentValue.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LP.** | **Metoda:** | **Argument przyjmujący:** | **Argument zwracany:** | **Opis:** |
| 1. | public CurrentValue() | Brak | Brak | Deklaracja klasy. |
| 2. | List<Long> GetPumpSettings() | Brak | zwraca ustawienia pompy. | Metoda zwracająca ustawienia pompy. |
| 3. | List<Long> GetFanSettings() | Brak | zwraca ustawienia wentylatorów. | Metoda zwracająca ustawienia wentylatorów. |
| 4. | List<Temperature> GetCVTEMPS() | Brak | lista z temp. CPU. | Metoda zwraca listę z temp. CPU. |
| 5. | List<Load> GetCVLOAD() | Brak | Listę z obciążeniem CPU. | Metoda zwracająca obciążenie CPU. |
| 6. | List<Fan> GetCVFAN() | Brak | Listę z prędkością wentylatorów. | Metoda zwracająca prędkość wentylatorów. |
| 7. | String getCVCPUNAME() | Brak | nazwę procesora. | Metoda zwracająca nazwę procesora. |
| 9. | public byte[] GetColourPalette() | Brak | paletę kolorów. | Metoda zwracająca paletę kolorów. |
| 10. | Long GetCurrentColorMode() | Brak | aktualny tryb pracy oświetlenia. | Metoda zwracająca aktualny tryb pracy oświetlenia. |
| 11. | Long GetCurrentTemperature() | Brak | temperatura procesora. | Metoda zwracająca temperaturę procesora. |
| 12. | Long GetCurrentFanSpeed() | Brak | szybkość obrotów wentylatorów. | Metoda zwracająca szybkość obrotów wentylatorów. |
| 13. | Long GetCurrentLiquidTemp() | Brak | temp. płynu chłodniczego. | Metoda zwracająca temp. płynu chłodniczego. |
| 14. | Long GetCurrentSetFanSpeed( | Brak | prędkość wentylatora dla aktualnej temp. | Metoda zwracając prędkość wentylatora dla aktualnej temp. |
| 15. | Long GetCurrentSetPumpSpeed() | Brak | prędkość pompy dla aktualnej temp. | Metoda zwracając prędkość pompy dla aktualnej temp. |
| 16. | Long GetCurrentProtocolVer() | Brak | wersję protokołu. | Metoda zwracająca wersję protokołu. |
| 17. | Long GetCurrentSafeCode() | Brak | identyfikator produktu. | Metoda zwraca identyfikator produktu. |
| 18. | Long GetWarningTemperature() | Brak | temperaturę graniczną dla CPU. | Metoda zwracająca temperaturę graniczną dla CPU. |

Tabela 12 Opis metod w klasie CurrentValue.java

### Klasa CurrentValueController.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LP.** | **Metoda:** | **Argument przyjmujący:** | **Argument zwracany:** | **Opis:** |
| 1. | public String index() | Brak | zwraca plik index.html | Metoda do obsługi statycznych plików. |
| 2. | public void executionColor() | Brak | Brak | Metoda wymuszająca zmianę koloru. |
| 3. | public void updateColorValue(@PathVariable Long range, @PathVariable String range2, @PathVariable Long value) | value wartość koloru. range2 rodzaj koloru. range numer koloru. | Brak | Metoda aktualizująca poszczególny kolor z osobna. |
| 4. | public void updateForceColorValue( @PathVariable Long value) | value dany kolor. | Brak | Metoda aktualizująca wszystkie kolory jako jeden. |
| 5. | public void updateForceColorValueRGB( @PathVariable Long value, @PathVariable String type) | value wartość koloru. type typ koloru. | Brak | Metoda aktualizująca dany kolor. |
| 6. | public void updateColorMode(@PathVariable Long value) | value tryb pracy. | Brak | Metoda aktualizująca tryb pracy oświetlenia. |
| 7. | public void updateTemperatureWarning(@PathVariable Long value) | value temp. graniczną dla CPU. | Brak | Metoda aktualizująca temp. graniczną dla CPU. |
| 9. | public void updateLanguage(@PathVariable Long value) | value język aplikacji. | Brak | Metoda aktualizująca język aplikacji. |
| 10. | public List<Long> showLanguage() | Brak | zwraca język. | Metoda zwracająca ustawiony język. |
| 11. | public void updatePumpSpeed(@PathVariable Long range, @PathVariable Long value) | value wartość. range próg temp. | Brak | Metoda aktualizacje prędkość pompy dla danego progu temp. |
| 12. | public void updateFanSpeed(@PathVariable Long range, @PathVariable Long value) | value wartość. range próg temp. | Brak | Metoda aktualizacje prędkość wentylatorów dla danego progu temp. |
| 13. | public ArrayList<Long> getWarningTemperature() | Brak | temp. graniczną dla CPU. | Metoda zwracająca temp. graniczną dla CPU. |
| 14. | public ArrayList<String> getErrorShow() | Brak | listę błędów. | Metoda zwraca listę błędów. |
| 15. | public ArrayList<String> getLogsShow(@PathVariable String value) | value nazwa pliku do odczytu. | listę z błędami. | Metoda zwraca logi błędów. |
| 16. | public List<String> getCpuNameShow() | Brak | nazwę procesora. | Metoda zwraca nazwę procesora. |
| 17. | public List<Long> getTempShow() | Brak | temp. procesora. | Metoda zwracająca temp. procesora. |
| 18. | public List<Double> getTempsShow() | Brak | temp. rdzeni procesora. | Metoda zwracająca temp. rdzeni procesora. |
| 19. | public List<Double> getLoadsshow() | Brak | listę z obciążeniem CPU. | Metoda zwraca listę z obciążeniem CPU. |
| 20. | public List<Double> getFanshow() | Brak | listę z wentylatorami. | Metoda zwraca listę z wentylatorami. |
| 21. | public List<Long> getFanSpeed() | Brak | prędkość wentylatorów. | Metoda zwraca prędkość wentylatorów. |
| 22. | public List<Long> getLiquidTemp() | Brak | temp. płynu chłodniczego. | Metoda zwraca temp. płynu chłodniczego. |
| 23. | public List<Long> getCriticalVariables() | Brak | najważniejsze informacje o pracy systemu. | Metoda zwraca najważniejsze informacje o pracy systemu. |
| 24. | public List<Long> getFanSettings() | Brak | ustawienia wentylatorów. | Metoda zwraca ustawienia wentylatorów. |
| 25. | public List<Long> getPumpSettings() | Brak | ustawienia pompy. | Metoda zwraca ustawienia pompy. |
| 26. | public List<Long> getVariables() | Brak | aktualną prędkość wentylatorów, pompy, wersję protokołu i identyfikator sprzętu. | Metoda zwraca aktualną prędkość wentylatorów, pompy, wersję protokołu i identyfikator sprzętu. |
| 27. | public List<Long> getColorMode() | Brak | tryb pracy oświetlenia. | Metoda zwraca tryb pracy oświetlenia. |
| 28. | public List<ArrayList> getColor() | Brak | tablicę z kolorami. | Metoda zwraca tablicę z kolorami. |

Tabela 13 Opis metod w klasie CurrentValueController.java

### Klasa NzOsApplication.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LP.** | **Metoda:** | **Argument przyjmujący:** | **Argument zwracany:** | **Opis:** |
| 1. | private static void openUrl() throws Exception | Exception Wyjątek podczas otwierania przeglądarki. | Brak | Metoda otwieracją przeglądarkę. |
| 2. | public static void main(String[] args) | args argumenty wywołania. | Brak | Główna metoda. |

Tabela 14 Opis metod w klasie NzOsApplication.java

# Podsumowanie projektu

Podczas prac nad projektem natknęliśmy się na kilka problemów, takich jak trudność połączeniem się z poziomu Java z interfejsem USB, czy przeprowadzenie testów jednostkowych. Pracowaliśmy razem nad większością elementów projektu. Michał Mołdawski przygotował większość części serwerowej oprogramowanie oraz dokonał inżynierii odwrotnej protokołu. Konrad Nowakowski opracował większość oprogramowania po stronie klienta, pliki konfiguracyjne i dokonał tłumaczenia na język angielski, przygotował dokumentację w JavaDoc i wspomógł pracę przy opracowaniu. Wspólnie opracowaliśmy kod w JavaScript/HTML/CSS oraz dokumentację projektu. Chciałbym podziękować społeczności Stack Overflow.