**Wojskowa Akademia Techniczna**

**Wydział Cybernetyki**

**Instytut systemów informatycznych**



**Niezawodność Oprogramowania**

**Sprawozdanie**

Prowadzący: dr hab. inż. Kazimierz Worwa

Wykonał: Michałowski Piotr

Grupa szkoleniowa: I9B2S4

# Spis treści

**Spis treści**

[1. Spis treści 2](#_Toc60921302)

[2. Opis problemu i sformułowanie zadania 3](#_Toc60921303)

[2.1. Sformułowanie zadania laboratoryjnego 3](#_Toc60921304)

[3. Ogólna specyfikacja aplikacji programowej 3](#_Toc60921305)

[3.1. Cel aplikacji 3](#_Toc60921306)

[3.1. Zakres aplikacji 3](#_Toc60921307)

[3.2. Wymagania funkcjonalne 4](#_Toc60921308)

[3.3. Wymagania techniczne 5](#_Toc60921309)

[4. Specyfikacje projektowe 5](#_Toc60921310)

[4.1. Format komunikatów 5](#_Toc60921311)

[4.2. Przetwarzanie komunikatów 6](#_Toc60921312)

[5. Opis implementacji 7](#_Toc60921313)

[5.1. Technologie i narzędzia 7](#_Toc60921314)

[5.2. Opis implementacji aplikacji 7](#_Toc60921315)

[5.3. Opis implementacji klas z pakietu solver 8](#_Toc60921316)

[5.3.1. Solver dla modelu Jelińskiego-Morandy 8](#_Toc60921317)

[5.3.2. Solver dla model Schicka-Wolvertona 8](#_Toc60921318)

[6. Opis wyników testowania 8](#_Toc60921319)

[6.1. Opis testów jednostkowych klas z pakietu *processor* 8](#_Toc60921320)

[6.2. Opis testów jednostkowych klas z pakietu *route* 9](#_Toc60921321)

[6.3. Wyniki testowania podanego zbioru danych 9](#_Toc60921322)

[7. Opis instalacji aplikacji 9](#_Toc60921323)

[7.1. Instalacja 9](#_Toc60921324)

[7.2. Przykład użycia 10](#_Toc60921325)

[7.3. Kod źródłowy 13](#_Toc60921326)

# Opis problemu i sformułowanie zadania

## Sformułowanie zadania laboratoryjnego

Należy zaprojektować i zaimplementować aplikację programową, która dla zadanej dokładności obliczeń i wskazanego zbioru danych zawierającego daną liczbę (240) odstępów czasowych pomiędzy wykryciem kolejnych błędów, umożliwia wyznaczenie wartości estymatorów N, Φ następujących modeli:

* Jelińskiego-Morandy,
* Schicka-Wolvertona.

Wykorzystując wyznaczone wartości parametrów N, Φ dla każdego z ww. modeli aplikacja oblicza wartość oczekiwaną czasu, jaki upłynie do momentu wykrycia kolejnego (241.) błędu.

# Ogólna specyfikacja aplikacji programowej

## Cel aplikacji

Celem aplikacji jest dokonywanie obliczeń wartości parametrów N, Φ estymatorów modeli Jelińskiego-Morandy i Schicka-Wolvertona – czyli modeli niezawodnościowych oprogramowania. Zakładam, że aplikacja programowa jest to wyłącznie moduł systemu przystosowany do rozwiązania określonej dziedziny problemów, dotyczących estymacji parametrów modeli niezawodnościowych oprogramowania.

Do aplikacji należy przesłać dane zawierające zbiór wartości odstępów czasowych pomiędzy kolejnymi wykrytymi błędami. Dane powinny zawierać również parametr określający wartość dokładności obliczeń modeli niezawodnościowych. Dane należy przesłać w ustalonym formacie JSON (co szczegółowo zostało przedstawione w punkcie 4.1.).

Aplikacja ma na celu przetworzenie danych wejściowych z wykorzystaniem modeli niezawodnościowych Jelińskiego-Morandy i Schicka-Wolvertona. Generowane są dane wyjściowe w formacie JSON zawierające:

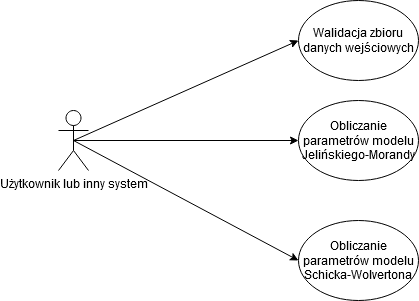
* wprowadzany zbiór wartości odstępów czasowych,
* wartość dokładności obliczeń,
* informacja z nazwą modelu niezawodnościowego oprogramowania,
* wartość parametru N,
* wartość parametru Φ,
* wartość oczekiwaną czasu, jaki upłynie do momentu wykrycia kolejnego błędu.

Omówienie szczegółowe formatu danych wyjściowych zostało dokładniej przedstawione w punkcie 4.1.

## Zakres aplikacji

Aplikacja programowa jest modułem systemu odpowiedzialnym wyłącznie za konkretną technikę rozwiązywania problemów, dotyczących estymacji parametrów modeli Jelińskiego-Morandy i Schicka-Wolvertona.

Z aplikacji programowej korzystają bezpośrednio użytkownicy prowadzący badania statystyczne dotyczące modelów niezawodności oprogramowania. Z aplikacji korzystają również zewnętrzne systemu informatyczne, które komunikują się z aplikacją programową poprzez broker wiadomości. Poniżej przedstawiono diagram przypadków użycia:



Rys : Diagram przypadków użycia aplikacji programowej

## Wymagania funkcjonalne

Niniejsza sekcja zawiera ogólny opis procesów oraz funkcji aplikacji programowej.

|  |  |
| --- | --- |
| **Nazwa funkcjonalności** | Walidacja wprowadzanego zbioru danych. |
| **Rola** | Sprawdza, czy dane są poprawne i można na ich podstawie obliczyć modele niezawodności. |
| **Opis działania** | Pobiera listę wartości odstępów czasu pomiędzy kolejnymi wykrytymi błędami oraz wartość dokładności. Sprawdza czy liczba wartości jest większa bądź równa 100. Sprawdza czy wartość dokładności jest większa od 0.00035 (przy niższej wartości rozwiązania modelu Schicka-Wolvertona są długo liczone). Sprawdza również czy wartości są dodatnie. |
| **Dane wejściowe** | Lista wartości odstępów czasu pomiędzy wykrytymi błędami oraz wartość parametru określającego dokładność obliczeń. |
| **Źródła danych** | Plik JSON przesłany do wejściowej kolejki komunikatów brokera wiadomości. |
| **Dane wyjściowe** | Dane wyjściowe to dane wejściowe przekazane do dalszego przetwarzania. W sytuacji wystąpienia błędu dane wyjściowe dodatkowo zawierają: nazwę błędu oraz komunikat błędu. |
| **Warunki początkowe** | Plik JSON znajduje się w wejściowej kolejce komunikatów. |
| **Warunki końcowe** | Dane wejściowe zostały przekazane do dalszego przetwarzania lub w kolejce zawierającej błędne komunikaty znajduje się plik JSON. |

|  |  |
| --- | --- |
| **Nazwa funkcjonalności** | Obliczanie wartości estymatorów parametrów N, Φ i wartości oczekiwanej czasu dla modelu Jelińskiego-Morandy. |
| **Rola** | Oblicza parametry modelu Jelińskiego-Morandy dla zadanej dokładności. |
| **Opis działania** | Pobiera listę wartości odstępów czasu pomiędzy kolejnymi wykrytymi błędami. Z wykorzystaniem zależności matematycznych wylicza wartości estymatorów N, Φ i wartości oczekiwaną czasu wykrycia kolejnego błędu. |
| **Dane wejściowe** | Lista wartości odstępów czasu pomiędzy wykrytymi błędami oraz wartość parametru określającego dokładność obliczeń. |
| **Źródła danych** | Plik JSON przesłany do wejściowej kolejki komunikatów brokera wiadomości. |
| **Dane wyjściowe** | Lista wartości odstępów czasu pomiędzy wykrytymi błędami, nazwa modelu niezawodności oprogramowania – „Jelińskiego-Morandy”, wartość parametru N, wartość parametru Φ, wartość oczekiwaną czasu, jaki upłynie do momentu wykrycia kolejnego błędu. |
| **Warunki początkowe** | Plik JSON znajduje się w wejściowej kolejce komunikatów. |
| **Warunki końcowe** | Plik JSON znajduje się w wyjściowej kolejce komunikatów. |

|  |  |
| --- | --- |
| **Nazwa funkcjonalności** | Obliczanie wartości estymatorów parametrów N, Φ i wartości oczekiwanej czasu dla modelu Schicka-Wolvertona. |
| **Rola** | Oblicza parametry modelu Schicka-Wolvertona dla zadanej dokładności. |
| **Opis działania** | Pobiera listę wartości odstępów czasu pomiędzy kolejnymi wykrytymi błędami. Z wykorzystaniem zależności matematycznych wylicza wartości estymatorów N, Φ i wartości oczekiwaną czasu wykrycia kolejnego błędu. |
| **Dane wejściowe** | Lista wartości odstępów czasu pomiędzy wykrytymi błędami oraz wartość parametru określającego dokładność obliczeń. |
| **Źródła danych** | Plik JSON przesłany do wejściowej kolejki komunikatów brokera wiadomości. |
| **Dane wyjściowe** | Lista wartości odstępów czasu pomiędzy wykrytymi błędami, nazwa modelu niezawodności oprogramowania - „Schicka-Wolvertona”, wartość parametru N, wartość parametru Φ, wartość oczekiwaną czasu, jaki upłynie do momentu wykrycia kolejnego błędu. |
| **Warunki początkowe** | Plik JSON znajduje się w wejściowej kolejce komunikatów. |
| **Warunki końcowe** | Plik JSON znajduje się w wyjściowej kolejce komunikatów. |

## Wymagania techniczne

Niniejsza sekcja zawiera opis wymagań technicznych aplikacji programowej:

1. Komunikacja z aplikacją programową powinna być realizowana przy użyciu warstwy pośredniej JSON.
2. Komunikacja pomiędzy aplikacją programową, a zewnętrznym systemem lub użytkownikiem odbywa z wykorzystaniem przykładowego brokera wiadomości np. IBM WebSphere MQ, MS MQ, Rabbit MQ (w ramach projektu rozwiązanie to uproszczono).

# Specyfikacje projektowe

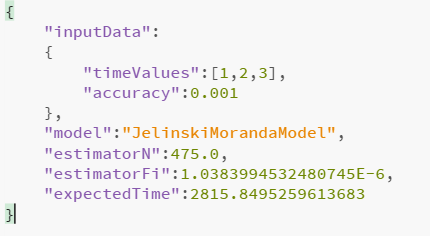
## Format komunikatów

Zgodnie z wymaganiami technicznymi komunikacja z aplikacją programową jest realizowana poprzez wymianę plików w formacie JSON. Rozróżnia się trzy rodzaje komunikatów:

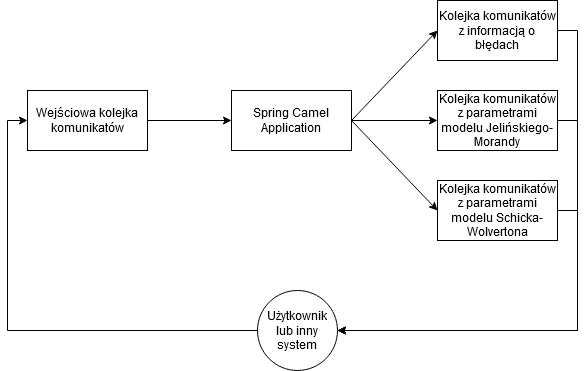
1. Komunikat wejściowy, który powinien mieć format JSON przedstawiony poniżej:

Minimalna ilość elementów tablicy *timeValues* wynosi 100.

Minimalna wartość *accuracy* wynosi 0.00035.

1. Komunikat wyjściowy, stosowany zarówno do danych wyjściowych zawierający estymowane parametry w modelu Jelińskiego-Morandy jak i Schicka-Wolvertona. Jego format został przedstawiony poniżej:
2. Komunikat o błędach, który jest przesyłany w przypadku wystąpienia błędu podczas walidacji lub innego etapu działania aplikacji programowej:

## Przetwarzanie komunikatów

 Aplikację programową zaprojektowano wyłącznie jako moduł systemu odpowiedzialny za obliczanie parametrów modeli Jelińskiego-Morandy i Schicka-Wolvertona. Aplikacja programowa komunikuje się z innymi systemami i użytkownikami za pomocą kolejki komunikatów (brokera wiadomości). Poniżej przedstawiono schemat komunikacji pomiędzy aplikacją programową, a zewnętrznymi systemami:

Rys : Schemat komunikacji pomiędzy aplikacją a MQs

W ramach projektu laboratoryjnego zastąpiono kolejki komunikatów zwykłymi folderami w systemie Windows. Poniżej zaprezentowałem nazwy katalogów odpowiadające danym kolejkom komunikatów:

* Wejściowa kolejka komunikatów – katalog „C:\NO\input”
* Kolejka komunikatów z informacją o błędach – katalog „C:\NO\output\error”
* Kolejka komunikatów z parametrami modelu Jelińskiego-Morandy – katalog „C:\NO\output\jelinski\_miranda\_model”
* Kolejka komunikatów z parametrami modelu Schicka-Wolvertona – katalog „C:\NO\output\schick\_wolverton\_model”

# Opis implementacji

## Technologie i narzędzia

Aplikację programową zaprojektowano z wykorzystaniem języka Java i z pakietem JDK 11. Posłużono się narzędziem *Apache Maven*, które zarządza budową projektu. Wykorzystano również *Spring Framework* do tworzenia szkieletów aplikacji w języku Java. Dodatkowo posłużono się frameworkiem *Apache Camel*, który to jest oprogramowaniem typu open source zorientowanym na przetwarzanie komunikatów z mechanizmem route’ów.

Jako bibliotekę pomocniczą wykorzystano *Project Lombok*. Pomaga ona w redukcji niepotrzebnego, pod względem biznesowym, kodu za pomocą adnotacji. Poprawia to również czytelność kodu. Wszystkie wykorzystane biblioteki i narzędzia znajdują się w pliku *pom*.

## Opis implementacji aplikacji

Aplikacja została zaimplementowana z zastosowaniem technik przetwarzania komunikatów stosowanych razem z frameworkiem Apache Camel. Zostały zaimplementowane cztery route’y, które odpowiadają za przepływ komunikatu:

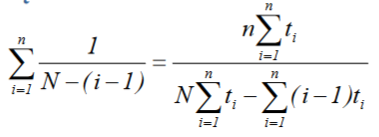
1. BaseRoute – odpowiada za wstępne przetwarzanie oraz walidację komunikatu.
2. JelinskiMorandaRoute – odpowiada za przetwarzanie komunikatu w celu obliczenia parametrów modelu Jelińskiego-Morandy.
3. SchickWolvertonRoute – odpowiada za przetwarzanie komunikatu w celu obliczenia parametrów modelu Schicka-Wolvertona.
4. ErrorHandlerRoute – jest uruchamiany w przypadku wystąpienia błędu. Ma na celu przetwarzanie komunikatu zawierającego informację o błędach.

W routach zawarte są obiekty procesorów. Każdy z zaimplementowanych procesorów jest odpowiedzialny za specyficzne przetwarzanie komunikatu:

1. ValidationProcessor – odpowiada za walidowanie komunikatu, który pobrała aplikacja.
2. JelinskiMorandaProcessor – odpowiada za wykonanie obliczeń parametrów z modelu Jelińskiego-Morandy i utworzenie komunikatu zawierającego estymowane parametry.
3. SchickWolvertonProcessor – odpowiada za wykonanie obliczeń parametrów z modelu Schicka-Wolvertona i utworzenie komunikatu zawierającego estymowane parametry.
4. ErrorHandlerProcessor – odpowiada za utworzenie komunikatu zawierającego informację o błędach.

## Opis implementacji klas z pakietu solver

### Solver dla modelu Jelińskiego-Morandy

 Zgodnie z podanymi wyrażeniami obliczania parametrów modelu, zostały zaimplementowane odpowiednie metody w klasie *JelinskiMorandaSolver*. Poniżej przedstawiono wykorzystane wyrażenie: wyrażenia:

Lewą stronę powyższego wyrażenia oblicza metoda *solveLeftSideEquation,* natomiast prawą stronę wyrażenia oblicza metoda *solveRightSideEquation.* Wartość parametru N początkowo jest równa liczbie elementów listy zawierającej odstępy czasowe większej o 2. Następnie wartość ta jest stopniowo zwiększana oraz liczony jest błąd obliczeń. Błąd obliczeń jest liczony jako błąd względny pomiędzy lewą stroną równania, a prawą stroną. Metoda licząca błąd względny jest zawarta w klasie nadrzędnej (jest wykorzystywana również w przypadku solvera dla modelu Schicka-Wolvertona). Błąd względny jest liczony jako stosunek różnicy pomiędzie lewą a prawą stroną równania do wartości większej (prawej lub lewej strony równania).

Po znalezieniu wartości parametru N w zadanej dokładności obliczany jest parametr Φ przy użyciu metody *solveFiEquation*. Mając parametry N i Φ obliczany jest czas oczekiwany za pomocą metody *solveExpectedTime*.

### Solver dla model Schicka-Wolvertona

Za pomocą podstawienia, przekształciłem wyrażenia do postaci:

W takiej postaci pozostaje do wyznaczenia jedna niewiadoma. Podobnie jak w poprzednim przypadku rozbiłem wyrażenie na jego lewą i prawą stronę. Następnie obliczałem błąd względny pomiędzy wyliczonymi wartościami i inkrementowałem wartość parametru N do momentu osiągnięcia zadowalającej dokładności.

Po wyznaczeniu parametru N, pozostałe parametry tj. Φ i wartość oczekiwaną czasu obliczyłem przy zaimplementowaniu metod odwzorowujących podane w instrukcji laboratoryjnej wyrażenia.

# Opis wyników testowania

## Opis testów jednostkowych klas z pakietu *processor*

Aplikacja programowa zawiera zestaw testów jednostkowych klas z pakietu *processor*. Testy mają na celu sprawdzenie poprawności działania wszystkich funkcjonalności klas procesora. Poniżej zestawiono listę klas testowych wraz z wymienionymi testowanymi funkcjami:

* ErrorHandlerProcessorTest – sprawdzane jest czy klasa *ErrorHandlerProcessor,* w prawidłowy sposób tworzy obiekt reprezentujący komunikat z informacjami o błędach, które się pojawiły w trakcie działania.
* JelinskiMorandaProcessorTest – sprawdzane jest, czy klasa *JelinskiMorandaProcessor*, w prawidłowy sposób oblicza i dodaje do komunikatu wartości estymowanych parametrów modelu Jelińskiego-Morandy.
* SchickWolvertonProcessorTest – sprawdzane jest, czy klasa *SchickWolvertonProcessor*, w prawidłowy sposób oblicza i dodaje do komunikatu wartości estymowanych parametrów modelu Schicka-Wolvertona.
* ValidationProcessorTest – sprawdzane jest czy klasa ValidationProcessor, w prawidłowy sposób waliduje komunikat. Ma na celu sprawdzenie m. in. jak program zachowa się w przypadku: poprawnego komunikatu, pustej listy wartości odstępów czasowych, ujemnych wartości odstępów czasowych itp.

## Opis testów jednostkowych klas z pakietu *route*

Aplikacja programowa zawiera zestaw testów jednostkowych route’ów. Sprawdzają one poprawność przetwarzania komunikatów przez aplikację od początku do końca.

W ścieżce projektu *resource/samples* znajdują się dwa katalogi *in* oraz *out.* W katalogu *in* znajdują się pliki zawierające przykładowe komunikaty wejściowe do aplikacji. W katalogu *out* znajdują się pliki zawierające poprawne komunikaty wyjściowe.

Zestaw testów sprawdza czy aplikacja (w sensie całościowym) poprawnie przetwarza te komunikaty. Wyodrębniono przypadki komunikatów dla modelu Jelińskiego-Morandy, dla modelu Schicka-Wolvertona, komunikaty niepoprawne oraz oczekiwane komunikaty wyjściowe z informacją o błędach.

## Wyniki testowania podanego zbioru danych

Ta sekcja zawiera przykład wyników testowania dla podanego w instrukcji laboratoryjnej zbioru danych wejściowych. Poniżej zestawiono wartości parametrów w tabeli.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Model Jelińskiego-Morandy | | | Model Schicka-Wolvertona | | |
| Dokładność | N | Φ | Wartość oczekiwana czasu | N | Φ | Wartość oczekiwana czasu |
| 0.1 | 293 | 2.804103768477116E-6 | 6858.0804487179485 | 267 | 3.4267694278700425E-9 | 4198.85447202356 |
| 0.01 | 382 | 1.7874660556472145E-6 | 3967.738888888889 | 294 | 2.5413139648044545E-9 | 3415.0167820744437 |
| 0.001 | 411 | 1.5986132136945414E-6 | 3679.6599019607843 | 300 | 2.414371712766741E-9 | 3320.7170008633525 |
| 0.00035 | 414 | 1.5813297248794638E-6 | 3655.3710019267824 | 300 | 2.414371712766741E-9 | 3320.7170008633525 |

# Opis instalacji aplikacji

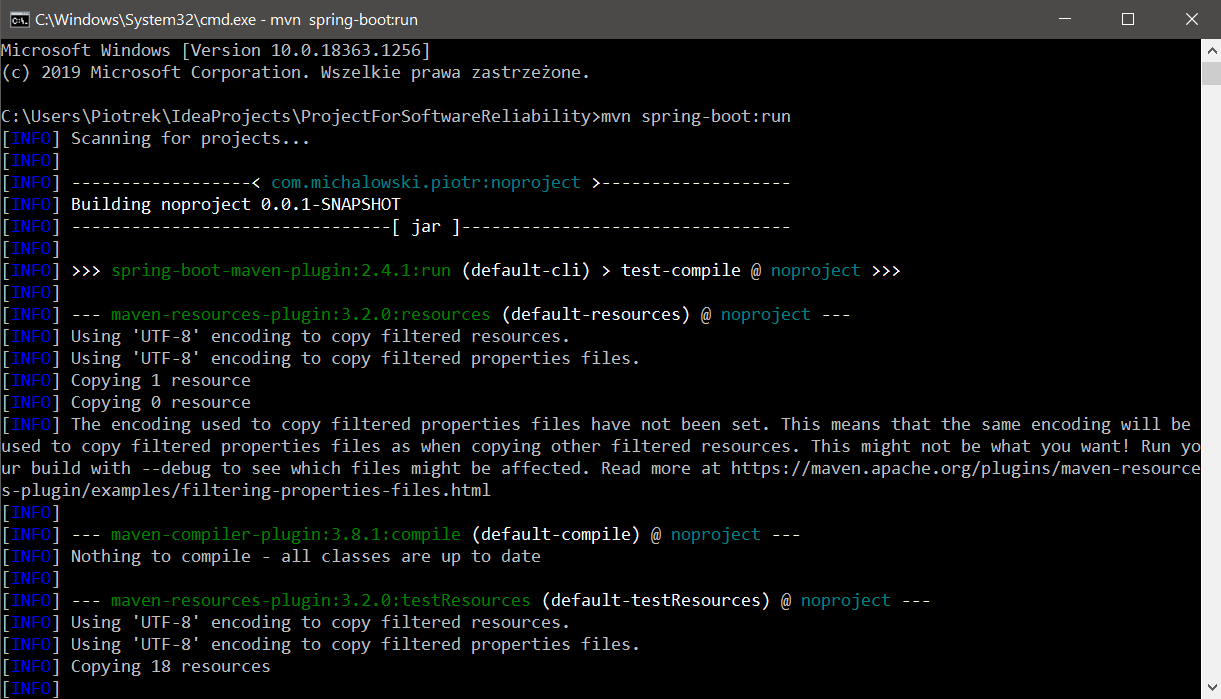
## Instalacja

Aplikacja została zaprojektowana z zastosowaniem SpringBoot. Aby ją uruchomić należy przejść do katalogu z plikami aplikacji. Następnie z użyciem konsoli, należy wprowadzić komendę „mvn spring-boot:run”.

## Przykład użycia

Ta sekcja zawiera przykład zastosowania aplikacji programowej. Celem jest pokazanie w kolejności czynności niezbędnych do uzyskania komunikatów zawierających szukane parametry modeli Jelińskiego-Morandy i Schicka-Wolvertona.

Aplikację można uruchomić za pośrednictwem środowiska np. IntelliJ IDEA. Jednakże w tym sprawozdaniu zawarłem przykład użycia bez środowiska programistycznego. W pierwszej kolejności uruchamiam konsolę w folderze zawierającym pliki aplikacji programowej. Przedstawiono to na poniższym zrzucie ekranu.



Rys : Ekran konsoli przy uruchamianiu aplikacji programowej

W następnej kolejności umieszczam JSON (w formacie danych wejściowych opisanym w punkcie 4.1.) w wejściowej kolejce komunikatów (katalog „C:\NO\input”). Poniżej zaprezentowałem zrzuty ekranu przedstawiające przykładowy plik JSON oraz wejściową kolejkę komunikatów.

Obraz zawierający tekst

Opis wygenerowany automatycznie

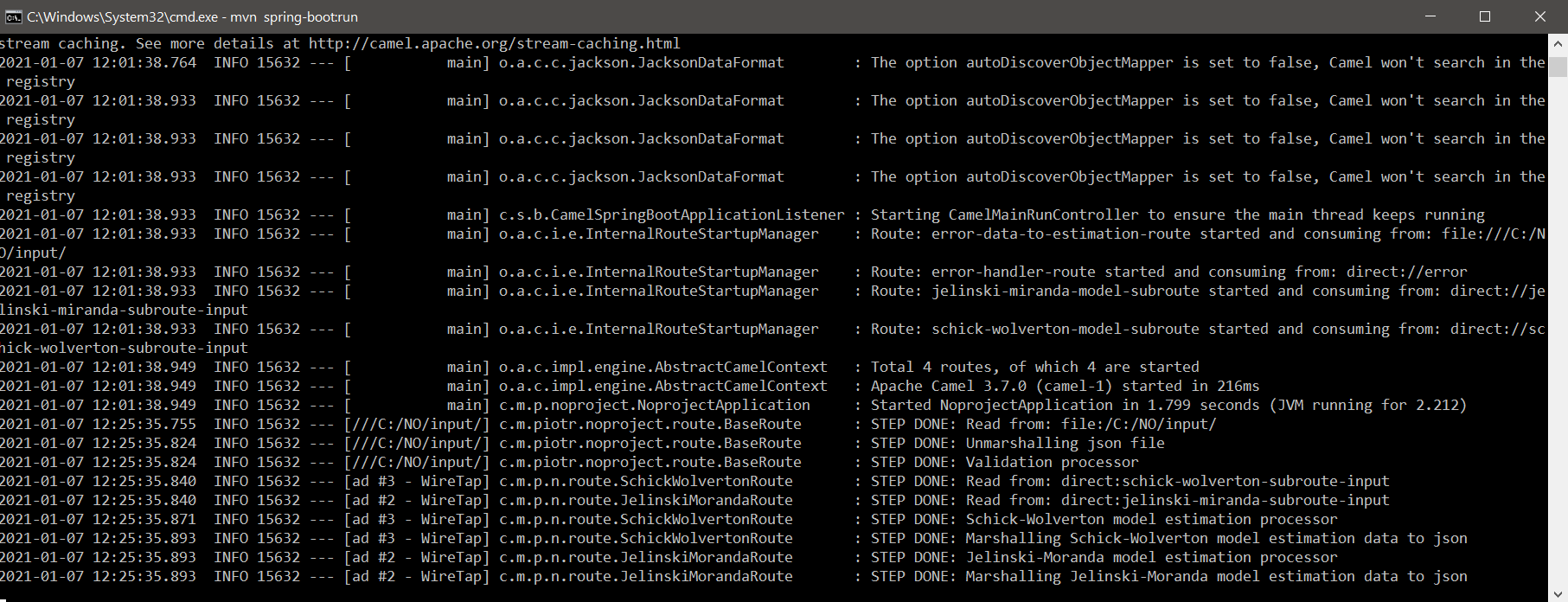
Rys : Zawartość przykładowego pliku danych wejściowych w formacie JSON

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rys : Zawartość wejściowej kolejki komunikatów

Aplikacja pobiera wiadomość oraz ją przetwarza. Wiadomość wyjściowa może trafić do różnych kolejek komunikatów w zależności czy w trakcie przetwarzania pojawił się błąd. Jeśli pojawił się błąd, to wiadomość trafiła do kolejki błędów (katalog „C:\NO\output\error”). W tym przypadku użycia, wiadomość została poprawnie przetworzona. Pojawiły się więc komunikaty wyjściowe w dwóch MQ (w przypadku modelu Jelińskiego-Morandy jest to katalog „C:\NO\output\jelinski\_miranda\_model”, natomiast w przypadku modelu Schicka-Wolvertona jest to katalog „C:\NO\output\schick\_wolverton\_model”). Poniżej zaprezentowałem zrzuty ekranu przedstawiające logi aplikacji oraz wyjściowe kolejki wiadomości.

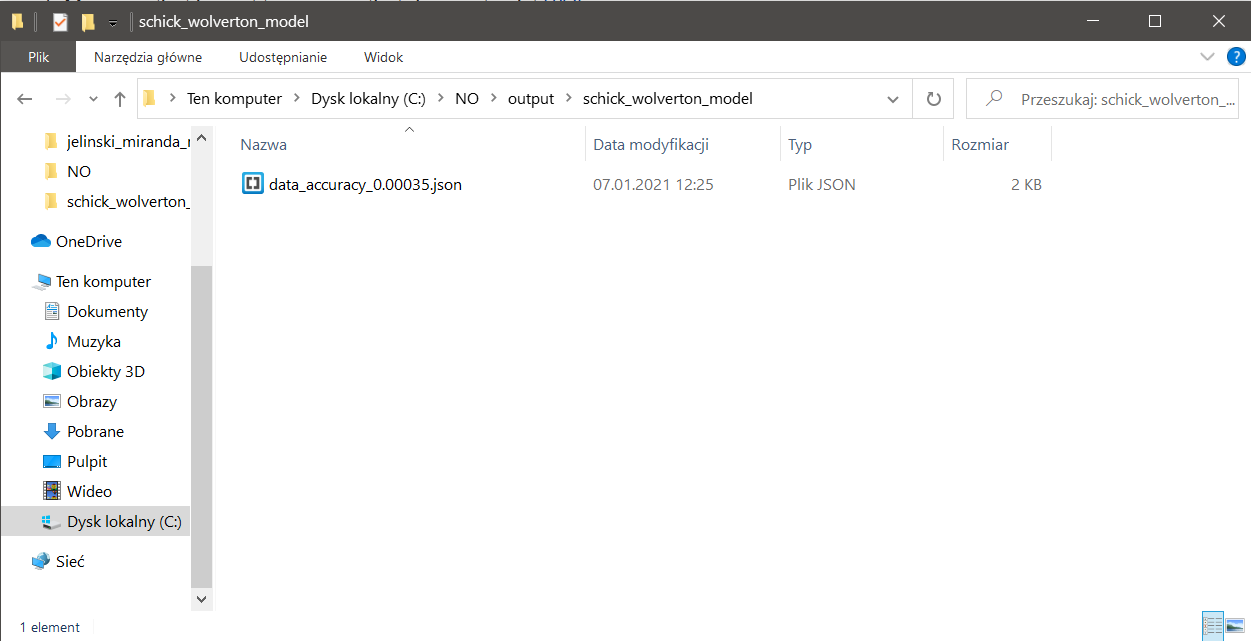


Rys : Logi aplikacji programowej po dokonaniu poprawnego przetworzenia wiadomości

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rys : Zawartość wyjściowej kolejki komunikatów z parametrami modelu Jelińskiego-Morandy



Rys : Zawartość wyjściowej kolejki komunikatów z parametrami modelu Schicka-Wolvertona

Obliczone parametry modelu Jelińskiego-Morandy oraz Schicka-Wolvertona zawarte są w plikach JSON. Poniżej zaprezentowałem zrzuty ekranu wspomnianych plików.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rys : Zawartość pliku JSON zawierającego obliczone parametry modelu Jelińskiego-Morandy

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rys : Zawartość pliku JSON zawierającego obliczone parametry modelu Schicka-Wolvertona

## Kod źródłowy

Github: https://github.com/PiotrMichalowski96/ProjectForSoftwareReliability