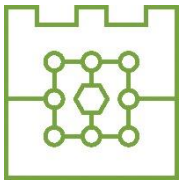
Politechnika Krakowska
im. Tadeusza Kościuszki

Wydział Informatyki i Telekomunikacji



Patryk Paluch

Numer albumu: 129028

Porównanie wydajności przetwarzania multimediów na różnych serwerach wbudowanych w aplikacji Spring Boot

Comparison of the performance of multimedia processing on various embedded servers in Spring Boot application

Praca magisterska
na kierunku Informatyka
specjalność Grafika komputerowa i multimedia

Praca wykonana pod kierunkiem:
dr Radosław Kycia

Kraków, 2023

Streszczenie

Niniejsza praca magisterska zatytułowana „Porównanie wydajności przetwarzania multimediów na różnych serwerach wbudowanych w aplikacji Spring Boot” przedstawia analizę wydajności trzech serwerów wbudowanych: Tomcat, Jetty i Undertow w aplikacji Spring Boot, podczas konwersji pomiędzy różnymi formatami plików audio oraz wideo. Do konwersji wykorzystana została biblioteka Java Audio Video Encoder, która do przetwarzania multimediów wykorzystuje narzędzie FFmpeg.

Pierwsza część pracy zawiera podstawy teoretyczne dla wykorzystywanych narzędzi oraz technologii. Część praktyczna pracy przedstawia sprawdzanie szybkości przetwarzania multimediów w utworzonej aplikacji wraz z analizą uzyskanych wyników oraz zbiorczym porównaniem wydajności poszczególnych serwerów.

Abstract

This master’s thesis entitled „Comparison of the performance of multimedia processing on various embedded servers in Spring Boot application” presents an analysis of the performance of three embedded servers: Tomcat, Jetty and Undertow in a Spring Boot application, during the conversion between various audio and video file formats. The Java Audio Video Encoder library was used for the conversion, which utilizes the FFmpeg tool for multimedia processing.

The first part of the thesis contains the theoretical foundations for the utilized tools and technologies. The practical part of the thesis presents the assessment of multimedia processing speed in the created application, along with an analysis of the obtained results and a comprehensive comparison of the performance of individual servers.

Spis treści

1 Wprowadzenie

W tym rozdziale przedstawione zostały najważniejsze informacje takie jak cel tworzonej pracy, zakres pracy oraz metodyka wykorzystana do rozwiązania poruszanego problemu.

1.1 Cel pracy

Celem niniejszej pracy magisterskiej jest wykonanie testów wydajnościowych konwersji pomiędzy wybranymi formatami audio oraz wideo na trzech różnych serwerów wbudowanych. W celu przeprowadzenia testów, utworzone zostaną trzy oddzielne aplikacje do przetwarzania multimediów, które będą wykorzystywały do swojego działania po jednym z wybranych serwerów wbudowanych.

1.2 Zakres pracy

W zakres pracy wchodzi przetestowanie szybkości konwersji pomiędzy różnymi formatami audio oraz wideo poprzez manualne uruchamianie poszczególnych konwersji oraz zapisanie uzyskanych wyników. W pracy przedstawione zostały wykorzystywane formaty, które korzystają z algorytmów o stratnej oraz bezstratnej kompresji oraz takie algorytmy, które nie wykorzystują kompresji. Po wykonaniu dziesięciu konwersji dla danego formatu, uzyskane wyniki zostaną poddane analizie a następnie zostaną one zestawione z wynikami konwersji pomiędzy innymi formatami.

1.3 Metodyka

Do przetestowania poszczególnych serwerów wbudowanych, utworzone zostały 3 aplikacje Spring Boot, które wykorzystywały inne serwery wbudowane. Aplikacje zostały uruchomione jako osobne kontenery w środowisku Docker, a komunikacji z nimi została przeprowadzona z wykorzystaniem narzędzia Postman. Do przeprowadzania konwersji pomiędzy formatami audio oraz wideo, użyta została ogólnodostępna biblioteka JAVE2.

I. Część teoretyczna

2 Biblioteka Java Audio Video Encoder jako narzędzie do przetwarzania multimediów w aplikacji Spring Boot.

W tym rozdziale przedstawione zostały narzędzia, które zostały wykorzystane podczas tworzenia aplikacji do przetwarzania multimediów. Aplikacja powstała w języku Java z wykorzystaniem frameworka Spring Boot, który pozwala na bardzo łatwe tworzenie aplikacji. Jako główną bibliotekę odpowiadająca za przetwarzania multimediów, wybrana została JAVE2 (Java Audio Video Encoder) [25], która jest opakowaniem dla projektu FFmpeg w Javie.

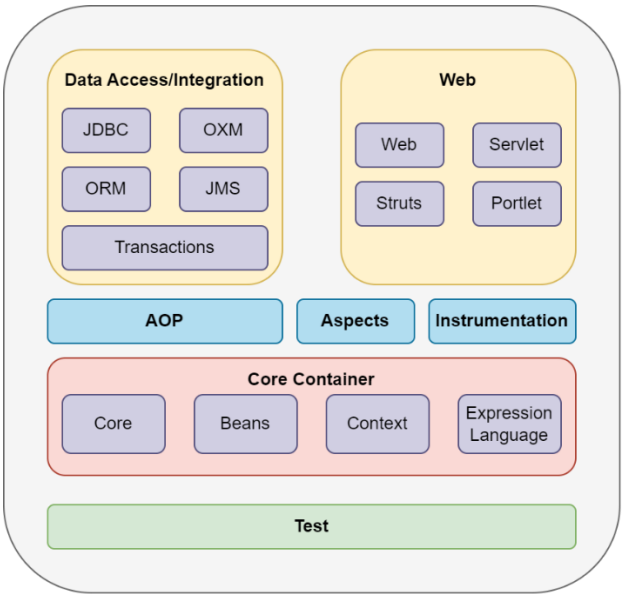
2.1 Spring Framework

Spring Framework to duży i uniwersalny framework, który jest szkieletem do tworzenia aplikacji w języku Java, który wspomaga proces budowania aplikacji w języku Java poprzez dostarczenie kompleksowej infrastruktury. Powstał on jako alternatywa dla Java Enterprise Edition (JEE), zapewniając prostsze podejście do programowania poprzez wstrzykiwanie zależności oraz programowanie aspektowe. Dzięki temu możliwe było uzyskanie funkcjonalności typowych dla Enterprise JavaBeans (EJB) przy użyciu Zwykłych Starych Obiektów Java (POJO - Plain Old Java Object) [46].

Struktura Springa cechuje się modularnością i składa się ona z wielu różnych modułów, a to pozwala na wybieranie tylko tych komponentów, które będą niezbędne podczas tworzenia projektu. Framework ten posiada około 20 modułów widocznych na ***Rysunku 1***, które podzielone zostały częściowo na grupy i należą do nich [43]:

- Grupa kontenera rdzenia, zawierająca moduły rdzenia, ziaren (Beans), kontekstu oraz języka wyrażeń.
    - Moduły rdzenia i ziaren (Beans) dostarczają najważniejsze części frameworka takie jak wstrzykiwanie zależności (Dependency Injection) oraz odwrócenie sterowania (Inversion Of Control).
    - Moduł kontekstu pozwala na dostęp do obiektów w sposób przypominający rejestr JNDI.
    - Moduł języka wyrażeń pozwala na wykonywania zapytań i manipulacji obiektów w czasie rzeczywistym.

- Grupa dostępu do danych i integracji składa się z modułów: JDBC, ORM, OXM, JSM oraz transakcji.
    - Moduł JDBC dostarcza warstwę abstrakcji JDBC, która pozwala wyeliminować ręczne tworzenie połączenia z bazą danych.
    - Moduł ORM dostarcza warstwę integracji takich jak JPA, JDO, Hibernate oraz iBatis, które służą do mapowania obiektowo-relacyjnego.
    - Moduł OXM wspomagający mapowanie Object/XML.
    - Moduł JMS do wysyłania oraz odbierania wiadomości.
    - Moduł transakcji zapewniający programowe oraz deklaratywne zarządzanie transakcjami.
- Grupa modułów Web: Web, Web-Servlet, Web-Struts oraz Web-Portlet.
    - Moduł Web dostarcza podstawowe funkcję integracji zorientowanych webowo.
    - Moduł Web-Servlet zawiera implementację frameworka MVC (Model-View-Controller).
    - Moduł Web-Struts dostarcza klasy pomagające integrację klasycznej warstwy Struts.
    - Moduł Web-Portlet dostarcza implementację MVC do użytku w środowisku portlet.
- Moduł AOP (Aspect-Oriented Programming) dostarcza implementację umożliwiającą m.in. definiowanie przechwytywaczy (interceptors) metod oraz punktów cięcia (pointcuts).
- Moduł aspektów dostarczający integrację z AspectJ.
- Moduł oprzyrządowania zapewniający obsługę oprzyrządowania klas oraz implementację modułu ładującego klasy.
- Moduł testów wspomagający testowanie komponentów za pomocą JUnit lub TestNG.



Rysunek 1. Architektura Spring Framework (źródło: opracowanie własne)

2.2 Aplikacja Spring Boot

Spring Boot jest to lekki framework umożliwiający tworzenie prostych aplikacji na bazie framework Spring. Został on stworzony w kwietniu 2014 roku, aby zredukować obciążenia związane z tworzeniem aplikacji internetowych. Dzięki temu programiści nie muszą poświęcać już dużo czasu na kwestie techniczne, a mogą się bardziej skupić na logice biznesowej aplikacji [37].

Spring Boot ułatwia dostęp do bazy danych w prosty sposób dzięki wykorzystaniu biblioteki Spring Data. Ta biblioteka zapewnia programistom dostęp do wielu źródeł danych, takich jak bazy danych relacyjne, nierelacyjne oraz dane w chmurze, a także zapewnia spójny model programowania poprzez framework Spring. Dzięki użyciu Spring Data JPA, łatwo można korzystać z relacyjnych baz danych, takich jak MySQL, PostgreSQL lub H2. Biblioteka ta eliminuje potrzebę ręcznego pisania kodu, który będzie odpowiadał za komunikację z bazą danych. Java Persistence API (JPA) umożliwia programistom połączenie modelu obiektowego Javy z modelem relacyjnej bazy danych dzięki wykorzystaniu relacyjnego mapowania obiektów (ORM - Object-Relational Mapping) [41].

Klasa główna aplikacji Spring Boot została przedstawiona na ***Rysunku 2,*** który pokazuje także jak tworzone są ziarna (Beans) w aplikacji.

```
@SpringBootApplication
public class MultimediaProcessingApplication {
    @Bean
    public MultimediaConverter multimediaConverter() {
        return new MultimediaConverter();
    }

    public static void main(String[] args) {
        SpringApplication.run(MultimediaProcessingApplication.class, args);
    }
}
```

Rysunek 2. Klasa główna aplikacji Spring Boot wraz z przykładem tworzenia ziarna (Bean) (źródło: opracowanie własne)

2.2.1 Serwery wbudowane

Serwery wbudowane w aplikacji Spring Boot pozwalają na szybkie uruchomienie oraz obsługę aplikacji bez konieczności specjalnego konfigurowania oraz nie wymagają one przygotowanego oddzielne specjalnego serwera aplikacji.

Dzięki wykorzystaniu serwerów wbudowanych, tworzenie oraz uruchamianie aplikacji jest łatwe oraz nie wymaga ono dużej ilości pracy. W zależności jaki serwer wbudowany chcemy wykorzystać, wystarczy odpowiednio zmodyfikować plik konfiguracyjny aplikacji. W większości przypadków zmiana serwera wbudowanego nie powoduje dużych zmian w kodzie jak np. w przypadku zmiany z serwera Tomcat na Jetty. W przypadku zmiany serwera Tomcat na Netty zmienia się już wiele ponieważ Netty jest wykorzystywane do tworzenia reaktywnych i jest nieblokującym serwerem w porównaniu do serwera blokującego Tomcat [22].

Serwery wbudowane dostarczają w aplikacji Spring Boot takie funkcjonalności jak obsługa protokołów HTTP/HTTPS, filtrowanie żądań, obsługa błędów, obsługa sesji oraz wiele innych przydatnych funkcjonalności. Do szybkiego i prostego konfigurowania wybranego serwera wbudowanego, Spring Boot pozwala poprzez wykorzystanie pliku konfiguracyjnego application.properties lub application.yml, w którym możemy zmienić np. port wykorzystywany przez nasz serwer [46].

Aplikacja Spring Boot może wykorzystywać jako serwer aplikacji, różnego rodzaju dostępne serwery takie jak:

- Apache Tomcat – serwer z otwarto źródłową (open source) implementacją m.in. Jakarta Servlet, Jakarta Server Pages [1].
- Jetty – serwer od Eclipse o otwartym kodzie źródłowym (open source) [8].
- Netty – darmowy serwer nieblokujący (NIO – Non-blocking I/O) od Netty Project Community [38]
- OpenLiberty – lekki oraz otwarty serwer aplikacji od IBM, który został następcą serwera WebSphere Liberty [39]
- WebLogic Server – komercyjny serwer od Oracle [40].
- Undertow – serwer WWW od firmy Red Hat z otwartym kodem źródłowym (open source) [44].
- WildFly (dawniej JBoss) – serwer z otwartym źródłem (open source), rozwijany przez Red Hat [48].

W dalszej części pracy przedstawione zostały wybrane serwery wbudowane, w skład których wchodzi Tomcat będący domyślnym serwerem wykorzystywanym przez Spring Boot oraz serwer Jetty i Undertow.

2.2.1.1 Tomcat

Apache Tomcat jest środowiskiem wykonawczym, które jest rozwijane w ramach projektu Apache oraz pozwala ono na uruchamianie aplikacji webowych. Zadaniem serwera Tomcat jest m.in. obsługiwanie oraz przetwarzanie żądań HTTP, w tym zarządzanie sesjami użytkowników oraz wykonywanie kodu aplikacji [31]. Jest on domyślnie wbudowany w zależność spring-boot-start-web widoczną na ***Rysunku 3***.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Rysunek 3. Zależność w Maven zawierająca serwer wbudowany Tomcat (źródło: opracowanie własne)

Tomcat jest oprogramowaniem typu open source, które wykorzystuje od 10 wersji platformę Jakarta EE (Jakarta Enterprise Edition) oraz dostarcza implementację dla Jakarta Servlet, Jakarta Expression Language, Jakarta Server Pages, Jakarta WebSockets,

Jakarta Authentication oraz Jakarta Annotations. Jakarta EE jest kontynuacją platformy Java EE (Java Enterprise Edition), która wcześniej nosiła nazwę J2EE (Java 2 Enterprise Edition). Apache Tomcat był początkowo częścią projektu Apache Jakarta, z którego to stał się oddzielnym projektem o nazwa Apache Tomcat oraz został on udostępniony na licencji Apache License version 2 i jest on rozwijany w partycypacyjnym środowisku [1].

Tomcat ze względu na swoją prostotę konfiguracji, popularność oraz elastyczność jest wykorzystywany w różnych branżach oraz organizacjach jako serwer aplikacji dla wielu różnych aplikacji webowych takich jak sklepy internetowe czy różnego rodzaju portale internetowe [2].

2.2.1.2 Jetty

Serwer Jetty jest serwerem aplikacji webowych napisanych w języku Java i cechuję się tym, że jest on lekki i wydajny [8]. Jest on kontenerem serwletów pozwalającym na uruchamianie na nim różnego rodzaju aplikacji webowych, które powstały w oparciu o takie technologie jak Jakarta Servlets, Jakarta Server Pages (JSP) oraz WebSocket. W starszych wersjach Jetty wykorzystywało Java EE, natomiast od wersji 10 przeniósł się on na Jakarta EE [8]. W celu wykorzystania go w swoim projekcie Spring Boot należy zmienić wykorzystywany serwer z domyślnego Tomcat, poprzez dodanie w pliku konfiguracyjnym „pom.xml" odpowiednich zależności, które zostały przedstawione na ***Rysunku 4***.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

Rysunek 4. Wymagana konfiguracja zależności w Maven do uruchomienia aplikacji na serwerze wbudowanym Jetty (źródło: opracowanie własne)

Jetty jest rozwijany przez Eclipse Foundation jako oprogramowanie typu open source i jest darmowe do użytku komercyjnego. Zapewnia on wsparcie dla obsługi

protokołów HTTP/1.1 oraz HTTP/2, WebSocket, OSGI (Open Service Gateway Initiative), JMX (Java Management Extension), JNDI (Java Naming and Directory Interface), JAAS (Java Authentication and Authorization Service) oraz wielu innych integracji. Serwer Jetty pozwala także na asynchroniczne przetwarzanie żądań, zarządzanie sesjami oraz wiele innych funkcji [8].

Jetty jest używany w wielu projektach jako alternatywa dla Apache Tomcat w kontekście serwera aplikacji webowych. Znajduje on zastosowanie na wielu płaszczyznach takich jak mikroserwisy, aplikacje webowe czy integrację systemów ze względu na jego główne cechy, którymi są m.in. niskie zapotrzebowanie na zasoby systemowe, prostota konfiguracji oraz jego szybkość działania.

2.2.1.3 Undertow

Undertow jest serwerem WWW, który został stworzony przez firmę Red Hat i jest on częścią platformy WildFly, która wcześniej nosiła nazwę JBoss [44]. Serwer jest rozwijany jako projekt open source dzięki czemu jest on w pełni dostępny oraz darmowy dla każdego. Undertow został napisany z użyciem języka Java i udostępnia on interfejsy zarówno blokujące jak i nieblokujące oparte na NIO (New Input/Output) [44].

W celu wykorzystania serwera w aplikacji Spring Boot, należy wykluczyć domyślny serwer Tomcat z głównej zależności Spring Boot odpowiedzialnej za serwer WWW oraz należy dodać oddzielnie zależność zawierającą serwer Undertow. Odpowiednia konfiguracja, którą należy umieścić w pliku „pom.xml” jest widoczna na ***Rysunku 5***.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-undertow</artifactId>
</dependency>
```

Rysunek 5. Wymagana konfiguracja zależności w Maven do uruchomienia aplikacji na serwerze wbudowanym Undertow (źródło: opracowanie własne)

Serwer został zaprojektowany w taki sposób aby był w pełni wbudowany co oznacza, że jest on integralną częścią aplikacji, a nie oddzielnym zewnętrznym serwerem oraz pozwalał na łatwe użytkowanie poprzez wzorzec projektowy Fluent Builder API. Serwer Undertow jest montowany przez aplikację osadzającą i nie wykorzystuje on koncepcji globalnego kontenera, dzięki takiemu rozwiązaniu podczas osadzania, aplikacja może wybrać do złożenia tylko te części, które są jej potrzebne. Serwer jest złożony z trzech głównych elementów, w których skład wchodzi co najmniej jedna instancja robocza XNIO (X non-blocking I/O), co najmniej jeden konektor oraz łańcuch obsługi zapewniający obsługę dla przychodzących żądań [45].

Serwer Undertow charakteryzuje się takimi cechami jak [44]:

- Wsparcie HTTP/2 bez konieczności nadpisywania klasy rozruchowej.
- Obsługa aktualizacji HTTP, która zapewnia multipleksowanie poprzez port HTTP wielu protokołów.
- Pełna obsługa gniazd sieciowych wraz z obsługą JSR-356.
- Obsługa Serwletów 4.0 wraz z osadzonymi serwletami.
- Możliwość osadzania w aplikacji lub uruchamiania oddzielnie.
- Konfigurowanie serwera poprzez wykorzystanie połączonych ze sobą programów obsługi, które pozwalają tworzyć skomplikowane oraz elastyczne łańcuchy obsługi żądań.

2.3 Biblioteka Java Audio Video Encoder

Biblioteka JAVE2 (Java Audio Video Encoder 2) umożliwia konwertowanie plików audio oraz wideo pomiędzy różnymi formatami poprzez udostępnione w niej interfejsy oraz klasy. Do konwersji wykorzystywana jest popularna biblioteka FFmpeg, która zapewnia wsparcie dla wielu różnych formatów plików takich jak MPEG, AVI, MP3 i wielu innych. JAVE2 jest darmowym oprogramowaniem oraz jest udostępniony na licencji GPL3 (GNU General Public License v3.0). Projekt JAVE2 powstał na bazie projektu JAVE (Java Audio Video Encoder) autorstwa Carlo Pellicia, którego projekt przestał być rozwijany i utrzymywany. Pliki wykonywalne zostały zastąpione nowymi aktualnymi wersjami, a sam kod został dostosowany do pracy z aktualnymi plikami binarnymi [25].

Poprzez wykorzystanie biblioteki JAVE2, mamy możliwość wykonywania wielu różnych operacji na wybranych plikach audio oraz wideo, takich jak zmiana formatu, zmiana rozdzielczości czy skalowanie lub kadrowanie. Wszystkie parametry dla pliku wynikowego są możliwe do ustawienia przed rozpoczęciem konwersji poprzez przesłanie do udostępnionych metod, odpowiednio skonfigurowanych obiektów konfiguracyjnych [26].

Biblioteka może być wykorzystywana podczas tworzenia aplikacji multimedialnych takich jak np. serwery do transmisji strumieniowej, edytorów audio oraz wideo lub jako narzędzie do przetwarzania takich plików. W celu wykorzystania biblioteki przez programistę, należy zaimportować ją za pomocą Mavena poprzez dodanie odpowiednich zależności. Odpowiednia konfiguracja dla pliku „pom.xml” została przedstawiona na ***Rysunku 6***.

```
<dependency>
    <groupId>ws.schild</groupId>
    <artifactId>jave-core</artifactId>
    <version>2.7.1</version>
</dependency>
<dependency>
    <groupId>ws.schild</groupId>
    <artifactId>jave-all-deps</artifactId>
    <version>2.7.1</version>
</dependency>
```

Rysunek 6. Wymagane zależności w Maven do poprawnego działania biblioteki JAVE2 na wszystkich systemach operacyjnych (źródło: opracowanie własne)

Biblioteka jest wspierana do działania na wielu systemach operacyjnych takich jak: Windows x32, Windows x64, MacOS x64, MacOS m1, Linux x32, Linux x64, Linux arm32 oraz Linux arm64. Do poprawnego działania biblioteki wymagane jest posiadanie języka Java w wersji 8 lub wyższej [25].

2.3.1 Projekt FFmpeg

Projekt FFmpeg jest popularnym narzędziem służącym do przetwarzania plików audio oraz wideo. Ta platforma zapewnia możliwość kodowania, dekodowania, transkodowania, multipleksowania, demultipleksowania, filtrowania, przysłania strumieniowego oraz odtwarzania praktycznie wszystkiego co zostało stworzone. FFmpeg poprzez wykorzystanie wielu bibliotek pozwala na kodowanie oraz dekodowanie różnych formatów audio oraz wideo, takich jak MP4, MP3, AAC, H.264 oraz wielu innych [9].

W projekcie wykorzystywane są najnowsze oraz najlepsze rozwiązania, z których mogą korzystać twórcy różnych aplikacji oraz sami użytkownicy końcowi [10]. Do osiągnięcia tych celów wykorzystane zostały najlepsze dostępne bezpłatne opcje oprogramowania. Samo narzędzie FFmpeg jest udostępnione na licencji LPGL (GNU Lesser General Public License), które pozwala na używanie go w różnych projektach niekomercyjnych ale także komercyjnych [10].

Sam projekt udostępnia różnego rodzaje bibliotek, które odpowiadają za różne funkcje, z których mogą korzystać bezpośrednio programiści. Takimi bibliotekami są:

- Libavcodec – jest biblioteką zawierająca różnego rodzaju dekodery oraz enkodera dla kodeków audio oraz wideo [11].
- Libavdevice – zawiera ona urządzenia wejścia oraz wyjścia, które służą do przechwytywania oraz renderowania do różnego rodzaju platform oprogramowania wejścia/wyjścia multimediów, w tym VfW, ALSA, Video4Linux oraz Video4Linux2 [12].
- Libavfilter – jest to biblioteka zawierająca filtry mediów [13].
- Libavformat – to biblioteka, w której zawierają się demultipleksery i multipleksery dla różnych formatów kontenerów multimedialnych [14].
- Libavutil – jest to biblioteka zawierająca wiele funkcji upraszczających programowanie. W skład tej biblioteki wchodzą: generatory liczb losowych,

struktury danych, procedury matematyczne oraz narzędzie multimedialne i wiele innych [15].

- Libswersample – to biblioteka, która odpowiada za wykonywanie wysoce zoptymalizowanych operacji resamplingu, konwersji formatu sampli oraz rematrixingu [16].
- Libswscale – biblioteka ta służy do wykonywania wysoce zoptymalizowanych operacji takich jak skalowanie obrazu czy konwersja formatu przestrzeni pikseli lub kolorów [17].

2.4 Wykorzystane narzędzia

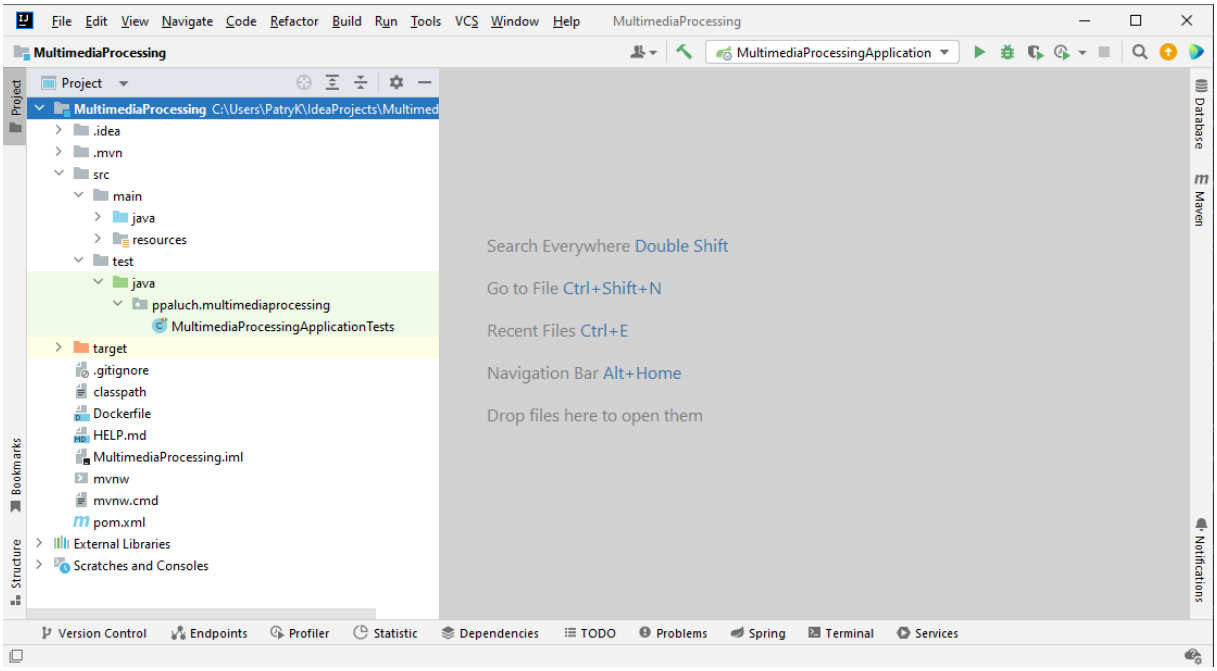
W tym rozdziale przedstawione zostały narzędzia, które były wykorzystane podczas tworzenia aplikacji do konwertowania pomiędzy wybranymi formatami plików. Jako środowisko programistyczne został wykorzystany IntelliJ IDEA, które pozwala w łatwy sposób programować w języku Java ze względu na jego rozbudowane funkcje. Do automatyzacji budowy oprogramowania wykorzystany został Apache Maven. W celu testowania aplikacji oraz zbierania wyników wykorzystany został Postman, który jest narzędziem ułatwiającym komunikację z API. W celu uruchomienia różnych wersji aplikacji w takim samym środowisku oraz z taką samą konfiguracją sprzętową, wykorzystany został Docker, który jest narzędziem służącym do wirtualizacji z poziomu systemu operacyjnego.

2.4.1 IntelliJ IDEA

IntelliJ IDEA to zintegrowane środowisko programistyczne (IDE – Integrated Development Environment), które wspomaga pracę programisty przy tworzeniu aplikacji poprzez różnego rodzaju automatyzację zadań czy sugerowanie odpowiednich działań [23]. Środowisko zapewnia integrację z różnymi popularnymi frameworkami, które są używane przez programistów jako szkielet do budowy aplikacji. Dzięki wykorzystaniu dostępnych narzędzi i integracji z frameworkami takimi jak np. możliwość tworzenia oraz konfigurowania projektów opartych o te framework bezpośrednio z panelu w IDE, zmniejsza się do minimum ryzyko wystąpieniu błędów podczas tworzenia oprogramowania [3].

Przykładowe funkcjonalności, które są oferowane przez to środowisko wyglądają następująco [24]:

- Inteligentne wspieranie różnych języków programowania takich jak Java, Kotlin, Python, JavaScript, TypeScript oraz wiele innych poprzez automatyczną refaktoryzację czy uzupełnianie kodu oraz różnego rodzaju analizy wykrywające błędy czy wyświetlanie sugestii.
- Debugowanie aplikacji pozwalające na krokowe wykonywanie kodu wraz z monitorowaniem zmiennych oraz stanu aplikacji.
- Integracja z wieloma zewnętrznymi narzędziami takimi jak narzędzia wspomagające zarządzanie bazą danych czy systemy odpowiedzialne za raportowanie błędów.
- Wspieranie popularnych bibliotek oraz frameworków takich jak Spring, Angular, Hibernate oraz wiele innych poprzez dostarczanie narzędzi, które ułatwiają ich konfigurację oraz testowanie.

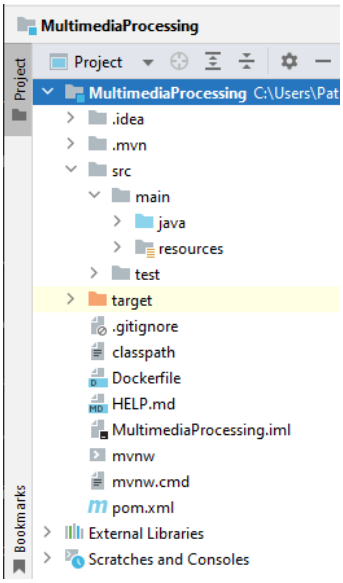


Rysunek 7. Wygląd środowiska IntelliJ IDEA z otwartym projektem (źródło: opracowanie własne)

2.4.2 Maven

Maven jest popularnym narzędziem służącym do zarządzania i automatyzacji projektu tworzonego w języku Java. Zapewnia on domyślne konfiguracje pozwalające na szybsze rozpoczęcie prac przez programistę ponieważ praca potrzebna do konfiguracji aplikacji została ograniczona do minimum [49].

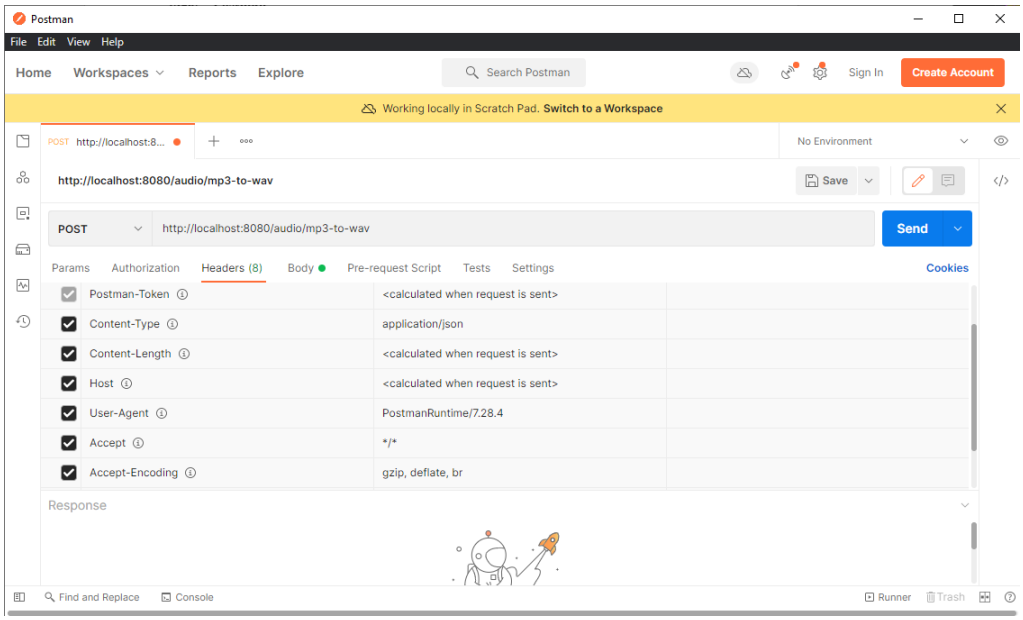
Struktura projektu wykorzystującego narzędzie Maven zawiera w głównym katalogu główny plik konfiguracyjny „pom.xml”, która zawiera wszystkie informację związane z projektem. W głównym pliku konfiguracyjnym znajdują się wszystkie zależności wykorzystywane przez projekt oraz konfiguracje potrzebne do zbudowania aplikacji. Obok pliku “pom.xml” znajdują się wydzielone katalogi widoczne na ***Rysunku 8***, które są odpowiedzialne za przechowywanie innych części projektu. Do przechowywania kodu źródłowego wykorzystywany jest katalog „src/main/java”, do przechowywania wszystkich testów aplikacji takich jak np. testy jednostkowe czy testy integracyjne wykorzystywany jest katalog „src/test/java”. Ostatnim katalog „src/main/resources” i służy on do przechowywania wszystkich potrzebnych zasobów do poprawnego działania aplikacji. W tym katalogu przechowywany jest m.in. plik konfiguracyjny aplikacji Spring Boot “application.properties” [34].



Rysunek 8. Struktura katalogów projektu wykorzystywana przez Maven (źródło: opracowanie własne)

2.4.3 Postman

Postman powstał jako narzędzie do testowania API (Application Programming Interface), który pozwala na wykonywanie różnego rodzaju żądań HTTP na wskazany z interfejsie adres. Dodatkowo pozwala on na wybranie różnych metod HTTP oraz określenie zawartości ciała żądania. Dzięki prostemu interfejsowi, można w łatwy sposób analizować odpowiedzi z aplikacji oraz istnieje możliwość generowania automatycznej dokumentacji [47]. Wygląd interfejsu aplikacji jest widoczny na ***Rysunku 9***.



Rysunek 9. Interfejs programu Postman z przykładowym żądaniem HTTP (źródło: opracowanie własne)

2.4.4 Docker

Docker jest otwartą platformą służącą do uruchamiania, tworzenia oraz dostarczania aplikacji w odizolowanych środowiskach zwanych kontenerem [5]. Izolacja kontenerów pozwala na uruchamianie wielu różnych kontenerów jednocześnie na tym samym hoście. Kontenery są przenośne i lekkie oraz działają one niezależnie od platformy, na której są uruchomione co pozwala na uruchamianie ich na różnych systemach operacyjnych. Każdy kontener może zostać udostępniony w łatwy sposób dzięki czemu można go uruchomić w innym miejscu i będzie on dokładnie taki sam jak w miejscu pierwotnym [7].

Docker do swojego działania wykorzystuje popularną architekturę typu klient-serwer, w której klient Dockera komunikuje się z uruchomionym daemonem, który to służy do budowania, uruchamiania oraz dystrybucji kontenerów. Komunikacja pomiędzy klientem a daemonem jest wykonywana za pomocą REST API, gniazd UNIX lub interfejsu sieciowego [42].

Do budowania obrazów aplikacji wykorzystywany jest plik konfiguracyjny „Dockerfile” przedstawiony na ***Rysunku 10***, w którym zdefiniowane są kolejne kroki budowania aplikacji. Zawiera on informację o obrazie bazowym, na którym jest budowana aplikacja, zależnościach, konfiguracjach sieciowych, portach oraz punktach montowania systemu plików [6].

```
FROM maven:3.8.5-openjdk-17 as 🔨 builder

RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

COPY pom.xml /usr/src/app

RUN mvn dependency:tree
RUN mvn dependency:resolve
RUN mvn dependency:resolve-plugins
RUN mvn dependency:go-offline
RUN mvn clean package -D spring-boot.repackage.skip=true

COPY src /usr/src/app/src

RUN mvn -T 1C package

FROM openjdk:17-jdk-alpine
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY --from=builder /usr/src/app/target/MultimediaProcessing.jar .
COPY --from=builder /usr/src/app/src/main/resources .
EXPOSE 8080
CMD java $JAVA_OPTS -jar MultimediaProcessing.jar
```

Rysunek 10. Przykładowy plik konfiguracyjny Dockerfile wykorzystywany do tworzenia obrazu dla aplikacji Spring Boot (źródło: opracowanie własne)

2.5 Wykorzystane formaty plików audio

Do przetwarzania formatów audio zostały wybrane 3 różne formaty takie jak: MP3 (MPEG-1/2 Audio Layer-3), WAV (Waveform Audio File Format) oraz OGG (Ogg Vorbis Audio File). Formaty MP3 oraz WAV są często spotykane i z tego powodu zostały wybrane, natomiast format OGG jest rzadziej spotykanym formatem co może powodować konieczność konwersji pomiędzy tym, a innymi formatami plików.

2.5.1 MP3

Format MP3 (MPEG-1/MPEG-2 Audio Layer 3) został opracowany przez Motion Picture Experts Group (MPEG) oraz został on oficjalnie wprowadzony po raz pierwszy w 1993 roku w standardzie MPEG-1 oraz w 1994 w standardzie MPEG-2. Ostatnie patenty, które chroniły format MP3 wygasły kolejno w 2012 roku w Unii Europejskiej oraz w 2017 roku w Stanach Zjednoczonych [28].

Format ten wykorzystuje algorytm o stratnej kompresji dźwięku, który to korzysta ze zmodyfikowanej dyskretnej transformacji cosinusowej (MDCT). W formacie MP3 występują 3 warstwy, które oznaczały różne rodzaje wersji algorytmu wykorzystywanego do kompresji dźwięku. Warstwa 1 była pierwszą wersją, która dostarczała akceptowalną jakość dźwięku wraz ze znaczącym zmniejszeniem rozmiaru pliku. Warstwa 2 powstała

jako ulepszenie Warstwy 1 i zapewniała ona jeszcze większą redukcję rozmiaru pliku w porównaniu do poprzedniej wersji, oferując przy tym jeszcze lepszą jakość dźwięku. Warstwa 3 jest ostatnią wersją, która jest najbardziej zaawansowana z nich wszystkich. Zapewniała ona jeszcze większe zmniejszenie rozmiaru wraz z zachowaniem wysokiej jakości dźwięku [4].

Plik MP3 składa się z ramek, a każda z nich składa się z nagłówka oraz danych. Nagłówek składa się z 32 bitów, które przechowują informację m.in. o: słowie synchronizacji, wersji, warstwie, przepływności, częstotliwości, czy dźwięk jest mono lub stereo, przechowywane są też informację o tym, czy jest to oryginał lub kopia [36].

2.5.2 WAV

Waveform Audio File Format (WAV) został wydany oficjalnie w sierpniu 1991 roku jako standard plików audio. Został on stworzony przez IBM oraz Microsoft w celu umożliwienia przechowywania plików audio na komputerach osobistych. Format ten służy w większości do przechowywania nieskompresowanych plików audio dzięki czemu zachowują one swój oryginalny dźwięk ponieważ nie występuje w nich jakakolwiek kompresja. Z powodu braku kompresji, pliki w tym formacie zajmują więcej przestrzeni dyskowej niż inne formaty takie jak MP3, które wykorzystujące kompresję stratną w celu zmniejszenia wielkości pliku. Istnieje możliwość przechowywania skompresowanego audio w pliku WAV dzięki wykorzystaniu ACM (Audio Compression Manager), w którym mamy do wyboru różnego rodzaju kodeki, które mogą posłużyć do kompresji [30].

Plik WAV posiada na początku nagłówek długości 44 bajtów, który definiuje właściwości pliku i jest on podzbiorem specyfikacji RIFF. Po nagłówku umieszczone są docelowe sekwencje fragmentów danych. Pliki WAV składają się z fragmentu FMT określającego format danych oraz DATA określającego rzeczywiste dane. Wygląd nagłówka przedstawiony jest w ***Tabeli 1*** [21].

Bajty	Opis
1-4	Oznaczenie rodzaju pliku
5-8	Rozmiar pliku
9-12	Typ pliku
13-16	Znacznik fragmentu formatu
17-20	Rozmiar formatu danych
21-22	Typ formatu
23-24	Ilość kanałów
25-28	Częstotliwość próbkowania
29-32	Wielkość strumienia danych
33-34	Ilość bitów na próbkę razy ilość kanałów podzielone przez ilość bitów
35-36	Ilość bitów na próbkę
37-40	Nagłówek oznaczający początek danych
41-44	Wielkość sekcji danych

Tabela 1. Budowa nagłówka pliku WAV

2.5.3 OGG

Format Ogg jest kontenerem multimedialnym, który jest wykorzystywany między innymi jako format plików oraz strumieni kodeków multimedialnych fundacji Xiph, która go stworzyła. Jest on formatem otwartym, wolnym od ograniczeń licencyjnych co oznacza, że każda osoba może korzystać z niego za darmo [52].

Ogg powstał jako prosty kontener strumieniowy, którego podstawowym celem było m.in. kadrowanie, porządkowanie czy przeplatanie. Dzięki swojej budowie może być on używany także w celach przechowywania plików multimedialnych, może służyć jako mechanizm dostarczania strumienia bądź jako element wykorzystywany przy konstruowaniu bardziej złożonego kontenera nieliniowego [52].

Celem kontenera Ogg jest ramkowanie oraz dostarczanie w odpowiedniej kolejności strumieni danych. Blok konstrukcyjny Ogg jest zbudowany ze strumieni elementarnych oraz multipleksowych. Składa się on z 8 pól, których łączna długość wynosi 28 bajtów będących nagłówkiem strony, listy długości pakietów o maksymalnej długości 255 bajtów oraz danych użytkowych, których łączna długość może wynosić nawet 65025 bajtów. Metadane będące częścią Ogg nie są wbudowane w sam kontener i są podzielone na poszczególne przedziały oraz warstwy. Taki sposób ich umiejscowienia pozwala na odizolowanie zbędnych danych strumieniowych od samego kontenera i jego implementacji [51].

Jednym z formatów odpowiedzialnych za kompresję dźwięku w kontenerze Ogg jest Ogg Vorbis, który jest w pełni otwarty, darmowy i wolny od różnego rodzaju patentów. Format wykorzystuje algorytm kompresji stratnej co powoduje, że niektóre informację, które są odpowiedzialne za dźwięk są tracone czego efektem jest mniejszy rozmiar pliku. Pomimo wykorzystania stratnej kompresji, Ogg Vorbis posiada wysoką jakość dźwięku i działa on na zasadzie przepływu bitów o zmiennej szybkości, dzięki czemu wykorzystuje on bardzo efektywnie przepływ danych. Dzięki temu, że powstał on jako niezależny format, może on być także wykorzystywany jako samodzielny format audio, niezależnie od samego kontenera Ogg [20].

Głównym przeznaczeniem Ogg Vorbis jest dźwięk o średniej oraz wysokiej jakości (8kHz – 48kHz). Obsługuje on stały oraz zmienny przepływ bitów, z zakresem bitrate na poziomie od 16kbps do 128 kbps na każdy kanał. Zapewnia on wyższą wydajność niż MPEG-1/2 Audio Layer 3 (MP3) [50].

2.6 Wykorzystane formaty plików wideo

Do przetwarzania formatów wideo zostały wybrane 3 znane formaty: MP4 (MPEG-4), AVI (Audio Video Interleave) oraz MKV (Matroska). Formaty MP4 oraz AVI zostały wybrane na podstawie tego, że są to często spotykane formaty wideo, natomiast format MKV nie jest tak często spotykany dzięki czemu będzie można zobaczyć jak taki format różni się pod względem czasu przetwarzania od pozostałych formatów w przypadku konwersji formatów.

2.6.1 MP4

Format MP4 (MPEG-4 Part 14) jest cyfrowym formatem plików, który jest wykorzystywany do przechowywania m.in. takich danych jak dźwięk czy obraz. Format ten został stworzony przez grupę MPEG (Moving Picture Experts Group), która jest częścią ISO (International Organization of Standarization) oraz ITU (International Telecommunication Union). Do grupy MPEG należy wielu ekspertów, którzy pochodzą z różnych stron świata, a ich celem było tworzenie standardów do przesyłania dźwięku i obrazu. Pierwsza oficjalna wersja (MPEG-4 Part 1) została opublikowana w 2001 roku, a wersja druga (MPEG-4 Part 14) została wydana w 2003 roku [29].

Struktura formatu jest podzielona na dwie sekcje, pierwsza z nich odpowiada za przechowywanie danych takich jak audio oraz wideo, natomiast druga sekcja przechowuje metadane, które to wskazują m.in. na różnego rodzaju flagi dostępu swobodnego czy znaczniki czasu. Struktury będące częścią formatu nazywane są atomami a ich rozmiar minimalny wynosi 8 bajtów, gdzie pierwsze 4 bajty odpowiadają za określenie rozmiaru, a pozostałe 4 bajty służą do określenia typu. Informację o atomach należących do pierwszego poziomu znajdują się w ***Tabeli 2***, natomiast w ***Tabeli 3*** przedstawione zostały atomy z drugiego poziomu [19].

Atom	Opis
ftyp	Typ pliku, opis, używane struktury
pdin	Informacja o progresywnym ładowaniu/pobieraniu pliku
moov	Metadane filmu
moof	Kontener z fragmentami wideo
mfra	Kontener z losowym dostępem do fragmentów wideo
mdat	Kontener danych dla mediów
stts	Tabela próbek do czasu
stsc	Tabela próbek na porcję
stsz	Ramkowanie
meta	Informację o metadanych

Tabela 2. Atomy będące częścią pierwszego poziomu w formacie MP4 (źródło: opracowanie własne)

Atom	Opis
mvhd	Nagłówki z informacjami o wideo
trak	Kontener z indywidualną ścieżką
udta	Informację o użytkowniku i ścieżce
iods	Deskryptor pliku MP4

Tabela 3. Atomy będące częścią drugiego poziomu w formacie MP4 (źródło: opracowanie własne)

2.6.2 AVI

Format AVI (Audio Video Interleave) został opracowany przez firmę Microsoft w 1992 roku jako format pliku multimedialnego wchodzącego w skład technologii Video for Windows. Format ten pozwala na przechowywanie informacji o danych audio oraz wideo jednocześnie. Dzięki wykorzystaniu techniki przeplatania (Interleave) przy przechowywaniu audio oraz wideo, można odtwarzać jednocześnie oba te strumienie [18].

Plik AVI posiada w swojej strukturze różne bloki, które noszą nazwę „chunk” i są one zorganizowane w strukturę typu RIFF (Resource Interchange File Format), która jest

także wykorzystywana przez format WAV. Każdy chunk posiada w sobie metadane opisujące treść jaką zawiera. Do ważnych nagłówków tego formatu zalicza się [35]:

- MAINAVIHEADER – zawiera najważniejsze informację m.in. o liczbie klatek, maksymalnej liczbie bajtów na sekundę, liczbie ścieżek wideo oraz audio.
- AVISTREAMHEADER – występuje przy każdym strumieniu audio oraz wideo. Zawiera on m.in. informację o typie strumienia, liczbie klatek na sekundę w przypadku wideo czy ilości kanałów w przypadku audio.
- BITMAPINFOHEADER – wykorzystywany w przypadku strumieni wideo, zawiera informację o szerokości oraz wysokości klatki czy ilości bitów na piksel.
- WAVEFORMAT – używany w przypadku strumieni audio, zawiera informacje m.in. o liczbie kanałów, częstotliwości próbkowania i ilości bitów na próbkę.

2.6.3 MKV

Format MKV (Matroska Video) powstał w 2002 roku, a jego głównym programistą był Lasse Kärkkäinen, który współpracował z założycielem Matroska, Steve'em Lhomme oraz z pozostałymi osobami będącymi w zespole programistów. Jest to projekt open source, który jest dostępny za darmo do użytku przez każdego. W 2010 roku został on usprawniony oraz stał się bazą dla formatu WebM, który powstał z inicjatywy Google i służy on do zastosowań internetowych [27].

Kontener multimedialny MKV jest podobny do formatów MOV oraz AVI lecz w przeciwieństwie do tych formatów, obsługuje on w tym samym pliku więcej niż jedną ścieżkę audio oraz napisów. MKV jest formatem plików, który jest oparty na EBML (Extensible Binary Meta Language), który to obsługuje wiele różnych formatów kompresji wideo oraz audio. Największą różnicą pomiędzy tym formatem a innymi jest taka, że MKV jest kontenerem, a nie kodekiem. Pliki wykorzystujące ten format mogą zawierać dla wideo H.264 oraz dla dźwięku MP3 (MPEG-1/2 Audio Layer-3) lub ACC (Advanced Audio Coding) ponieważ w jednym pliku różne elementy mogą używać różnych typów kodowania [33].

Matroska wykorzystuje w specyfikacji EBML specjalne wartości m.in. dla kilku parametrów takich jak [32]:

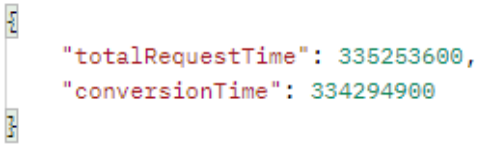
- Nagłówek EBML docType musi posiadać wartość 'matroska'.
- Nagłówek EBMLMaxIDLength musi być ustawiony na wartość 4.
- Nagłówek EBMLMaxSizeLength musi posiadać wartość od 1 do 8.

II. Część badawcza

3 Przetwarzanie multimediów na różnych serwerach wbudowanych w aplikacji Spring Boot.

Do przeprowadzenia testów, utworzona została aplikacja Spring Boot posiadająca oddzielony punkt końcowy (endpoint) do każdego rodzaju konwertowanego formatu pliku. W celu rozpoczęcia konwertowania, użytkownik wysyła żądanie do aplikacji metodą POST, a w odpowiedzi dostaje informację o czasie trwania przetwarzania całego żądania oraz czasie poświęconym bezpośrednio na konwertowaniu formatów. Przyjęcie żądania od klienta skutkowało uruchomieniem przetwarzania z odpowiednio skonfigurowanymi parametrami konwertowania, które były identyczne dla każdego rodzaju serwera wbudowanego.

Aplikacja po przyjęciu żądania rejestrowała czas rozpoczęcia samego żądania oraz przed wysłaniem odpowiedzi zapisywała ile trwał całkowity czas przetwarzania żądania. Pomiar czasu samego konwertowania formatów był zapisywany oddzielne, a następnie wraz z czasem trwania całego żądania, zwracany był do użytkownika w postaci JSON przedstawionej na ***Rysunku 11***.



Rysunek 11. Przykładowy pomiar czasu w formacie JSON zwracany z aplikacji (źródło: opracowanie własne)

3.1 Specyfikacja wymagań

Tworzona aplikacja powinna spełniać wszystkie wymagania funkcjonalne pozwalające ma przetestowanie konwersji pomiędzy wieloma różnymi formatami audio oraz wideo. W skład tych specyfikacji wchodzą takie wymagania jak:

- Uruchamianie konwersji pomiędzy dwoma dostępnymi formatami audio.
- Uruchamianie konwersji pomiędzy dwoma dostępnymi formatami wideo.
- Podgląd uzyskanych czasów konwersji.
- Przechowywanie utworzonych plików podczas konwersji.
- Podgląd informacji o plikach źródłowych oraz utworzonych podczas konwersji.

3.2 Architektura aplikacji

Do utworzenia aplikacji użyta została architektura oparta na wzorcu REST API (Representational State Transfer), która wykorzystuje do tworzenia, wdrażania oraz skalowania systemów rozproszonych, zasady oraz konwencje protokołu HTTP. Głównym celem takiego wzorca jest zapewnienie elastycznego oraz bezstanowego interfejsu programistycznego umożliwiającego łatwą komunikację pomiędzy serwerem a klientem.

Aplikacji została podzielona na 3 główne warstwy:

- Warstwa prezentacji (Presentation Layer) – należą do niej kontrolery REST API w Spring, które obsługują żądania przychodzące. Zdefiniowane w niej są wszystkie punkty końcowe (endpoints) aplikacji. Ta warstwa komunikuje się z warstwą serwisową.
- Warstwa serwisowa (Service Layer) – zawiera ona logikę biznesową aplikacji i odpowiada ona za przetwarzania żądań, które zostały odebrane w warstwie prezentacji. Ta warstwa komunikuje się z warstwą prezentacji oraz warstwą dostępu do danych.
- Warstwa dostępu do danych (Data Access Layer) – odpowiada ona za przechowywanie informacji o plikach źródłowych oraz tych, które powstały podczas przetwarzania multimediów.

3.3 Metodyka pomiaru czasów konwertowania formatów plików

Konwertowanie poszczególnych formatów plików było mierzone od momentu bezpośrednio przed uruchomieniem metody odpowiedzialnej za konwertowanie do momentu jej zakończenia co zostało przedstawione na ***Rysunku 12***. Do końcowego pomiaru nie wliczał się czas potrzebny aplikacji na przyjęcie żądania oraz czas potrzebny na transport żądania do aplikacji i odpowiedzi do klienta. Dzięki takiemu pomiarowi, wszystkie dane pokazują bezpośrednio informację o czasie samego konwertowania multimediów na poszczególnych serwerach wbudowanych.

```
long startTime = System.nanoTime();
try {
    encoder.encode(sourceMultimediaObject, targetFile, encodingAttributes);
} catch (Exception e) {
    e.printStackTrace();
}
long endTime = System.nanoTime();
```

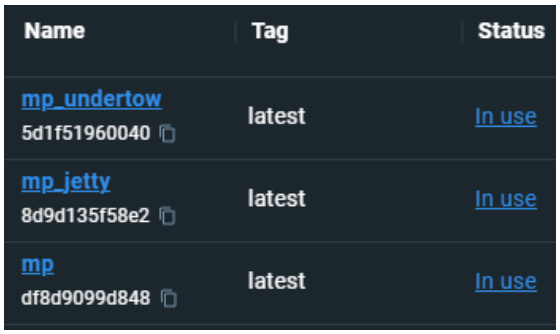
Rysunek 12. Miejsce początkowego i końcowego pomiaru czasu przy konwertowaniu formatów plików (źródło: opracowanie własne)

3.4 Sposób uruchomienia aplikacji

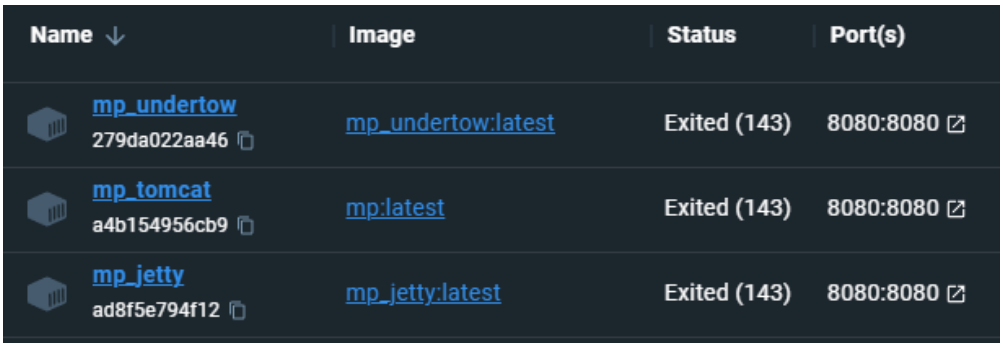
W celu zapewnienia każdej instancji aplikacji takich samych środowisk, w których zostały uruchomione, wszystkie 3 aplikacje wykorzystujące poszczególne serwery wbudowane zostały najpierw przekształcone w obrazy Dockerowe z wykorzystaniem Dockerfile.

Każdy obraz został zbudowany z wykorzystywaniem takiej samej konfiguracji dzięki czemu każdy z nich po uruchomieniu będzie działał w takim samym środowisku i różnić się one będą jedynie wykorzystywanym serwerem wbudowanym.

Następnie utworzone zostały kontenery z wykorzystaniem utworzonych wcześniej obrazów, które są widoczne na ***Rysunku 13*** oraz ***Rysunku 14***. W celu komunikacji z poszczególnym kontenerem wykorzystywany był port 8080, który był przekierowywany wewnątrz niego na ten sam port, który jednocześnie był portem wykorzystywanym przez aplikację do komunikacji. W celu uniknięcia niepotrzebnego obciążenia przez inne kontenery, każdy był uruchamiany pojedynczo co pozwalało na utworzenie jak najbardziej identycznych środowisk testowych, w których wyniki nie powinny być przekłamywane z powodu dostępu do mniejszej ilości zasobów.



Rysunek 13. Lista utworzonych obrazów z poszczególnymi serwerami wbudowanymi (źródło: opracowanie własne)



Rysunek 14. Lista utworzonych kontenerów z poszczególnymi serwerami wbudowanymi (źródło: opracowanie własne)

3.5 Opis testowania aplikacji

Testowanie szybkości konwersji odbywać się będzie poprzez ręczne wysyłanie żądań HTTP do utworzonej aplikacji. Każde żądanie zwraca odpowiedź w postaci JSON, które zostanie zapisane w tabeli uzyskanych wyników dla poszczególnych formatów pomiędzy, którymi odbywała się konwersja. W celu uniknięcia wliczania czasów potrzebnych na inicjalizację potrzebnych zasobów, każda konwersja przed rozpoczęciem zbierania wyników, została uruchomiona co najmniej raz.

Testowanie odbywało się w danym momencie tylko dla jednego sposobu konwersji gdzie każda kolejna próba konwersji rozpoczynała się od razu po zakończeniu poprzedniej tak aby nie dochodziło do sytuacji gdzie aplikacja zostanie obciążona więcej niż jedną konwersją jednocześnie. Dodatkowo po przeprowadzeniu wszystkich wymaganych konwersji pomiędzy danymi formatami, aplikacja była ponownie uruchamiana tak aby wszystkie zasoby, które zostały wykorzystane podczas poprzednich konwersji zostały zwolnione. Dzięki restartowaniu aplikacji, poprzednie konwersje nie wpływa w żaden sposób na przeprowadzanie kolejnych.

3.6 Konwersja formatów plików audio

Do uruchomienia konwersji pomiędzy poszczególnymi formatami audio wymagane było wysłanie żądania metodą POST do jednego z wcześniej przygotowanego punktu końcowego. Jeden z przykładowych punktów końcowych został przedstawiony na ***Rysunku 15***. Każdy z nich rozpoczynał konwersję przygotowanego wcześniej pliku źródłowego audio pomiędzy formatami, które podane były bezpośrednio w adresie żądania. Każdy serwer wykorzystał dokładnie te same pliki audio i taką samą konfigurację przy ich konwersji.

```
@PostMapping("/mp3-to-wav")
public ResponseEntity<TimeResponse> convertMP3toWAV() {
    long startTime = System.nanoTime();
    TimeResponse timeResponse = audioService.convertMP3toWAV();
    long endTime = System.nanoTime();
    timeResponse.setTotalRequestTime(endTime - startTime);
    return ResponseEntity.ok(timeResponse);
}
```

Rysunek 15. Metoda obsługująca żądanie POST dla konwersji pomiędzy formatami MP3 oraz WAV (źródło: opracowanie własne)

Dla konwersji plików audio przygotowane zostały trzy pliki źródłowe po jednym w każdym wykorzystywanym formacie. Wszystkie pliki audio posiadają taką samą długość 4 minuty i 46 sekund co odpowiada mniej więcej czasowi trwania dłuższych piosenek. Parametry plików źródłowych wyglądają następująco:

- Parametry pliku MP3:
  - Przepływność: 128 kb/s
  - Kanały: 2
  - Częstotliwość próbkowania: 48 kHz
  - Rozmiar: 4,37 MB
  - Długość: 4 minuty 46 sekund
- Parametry pliku WAV:
  - Przepływność: 1536 kb/s
  - Kanały: 2
  - Częstotliwość próbkowania: 48 kHz
  - Rozmiar: 52,4 MB
  - Długość: 4 minuty 46 sekund
- Parametry pliku OGG:
  - Przepływność: 128 kb/s
  - Kanały: 2
  - Częstotliwość próbkowania: 48 kHz
  - Rozmiar: 2,74 MB
  - Długość: 4 minuty 46 sekund

3.6.1 Konwersja pomiędzy formatami MP3 oraz WAV

W tym podrozdziale przedstawione zostały wyniki pomiarów konwersji z formatu MP3 do WAV oraz z formatu WAV do MP3 wraz z analizą uzyskanych wyników. Każda konwersja wykorzystywała identyczną konfigurację enkodera i prezentowała ona się następująco:

- Konfiguracja enkodera dla konwersji do formatu MP3:
  - Kodek: libmp3lame
  - Przepływność: 128 kb/s
  - Kanały: 2
  - Częstotliwość próbkowania: 48 kHz
- Konfiguracja enkodera dla konwersji do formatu WAV:
  - Kodek: pcm_s16le
  - Przepływność: 128 kb/s
  - Kanały: 2
  - Częstotliwość próbkowania: 48 kHz

3.6.1.1 Pomiary czasów konwersji

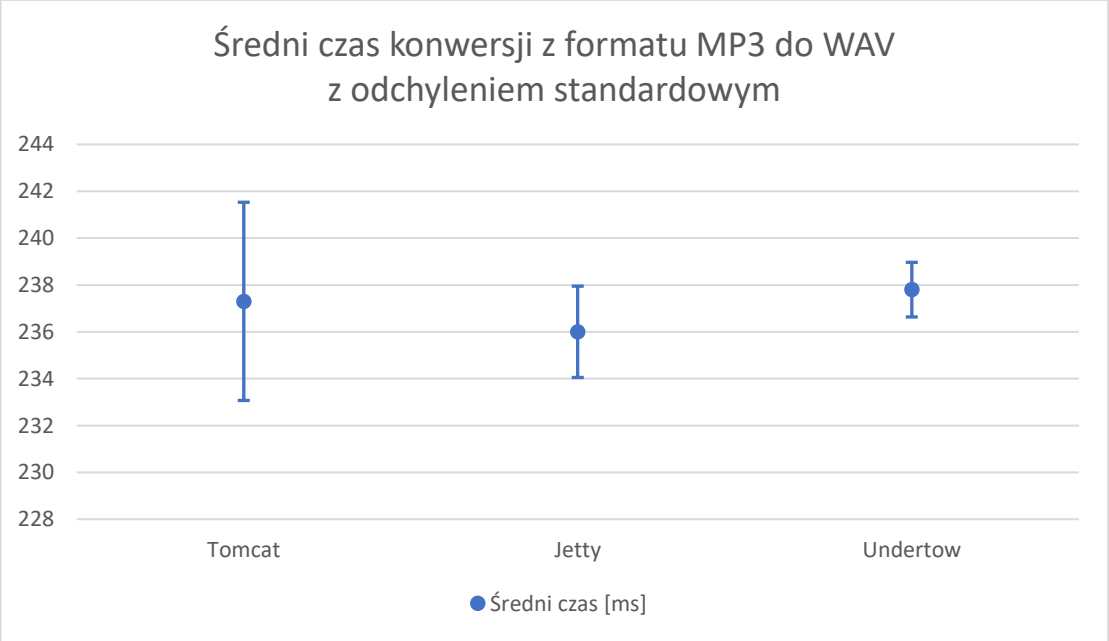
Każda konwersja została uruchomiona 5 razy a następnie z uzyskanych wyników został obliczony średni czas. Każdy pomiar dla konwersji z formatu MP3 do formatu WAV został przedstawiony w ***Tabeli 4***. W ***Tabeli 5*** oraz na ***Rysunku 16*** przedstawiony został średni czas oraz odchylenie standardowe dla poszczególnych serwerów.

Serwer	Pomiar [ms]									
	1	2	3	4	5	6	7	8	9	10
Tomcat	232	242	236	243	238	233	244	234	239	232
Jetty	231	237	237	236	237	234	237	238	236	237
Undertow	237	238	240	238	237	237	236	239	238	238

Tabela 4. Pomiary czasów konwersji z formatu MP3 do WAV

Serwer	Średni czas	Odchylenie standardowe
Tomcat	237,3 ms	4,230 ms
Jetty	236,0 ms	1,950 ms
Undertow	237,8 ms	1,166 ms

Tabela 5. Średni czas wraz z odchyleniem standardowym dla konwersji z formatu MP3 do WAV



Rysunek 16. Wykres przedstawiający średni czas z odchyleniem standardowym dla konwersji z formatu MP3 do WAV na różnych serwerach (źródło: opracowanie własne)

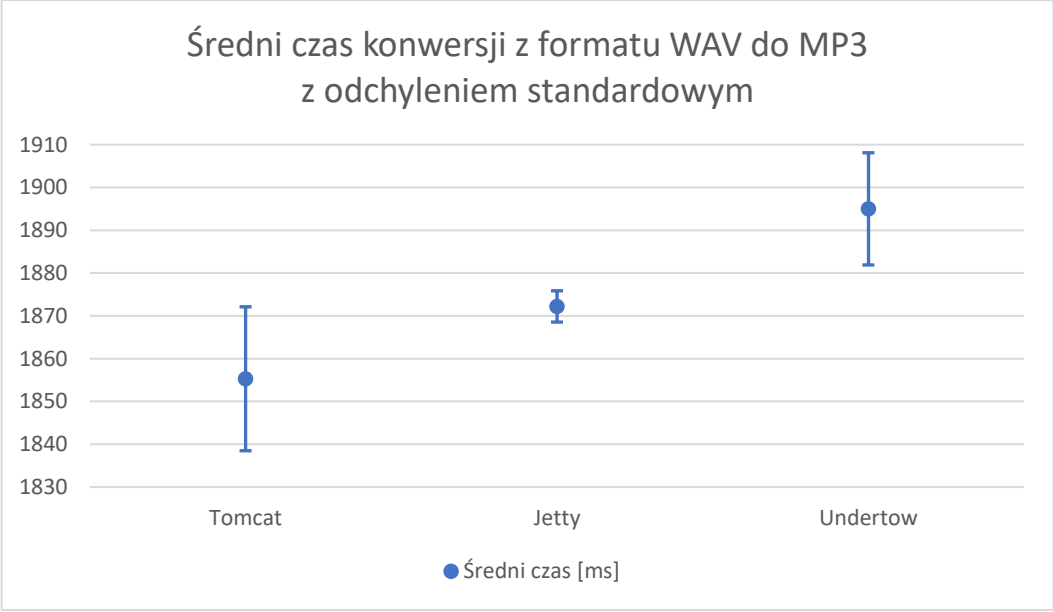
Pomiary czasów dla konwersji z formatu WAV do MP3 zostały wykonane w ten sam sposób jak w przypadku konwersji z MP3 do WAV, a uzyskane wyniki zostały przedstawione w ***Tabeli 6***. W ***Tabeli 7*** oraz na ***Rysunku 17*** przedstawiony został średni czas oraz odchylenie standardowe dla poszczególnych serwerów.

Serwer	Pomiar [ms]									
	1	2	3	4	5	6	7	8	9	10
Tomcat	1826	1860	1858	1867	1870	1851	1874	1879	1848	1820
Jetty	1874	1870	1876	1869	1868	1866	1869	1878	1873	1869
Undertow	1883	1911	1887	1886	1878	1894	1903	1882	1915	1911

Tabela 6. Pomiary czasów konwersji z formatu WAV do MP3

Serwer	Średni czas	Odchylenie standardowe
Tomcat	1855,3 ms	16,82 ms
Jetty	1872,2 ms	3,66 ms
Undertow	1895,0 ms	13,11 ms

Tabela 7. Średni czas wraz z odchyleniem standardowym dla konwersji z formatu WAV do MP3



Rysunek 17. Wykres przedstawiający średni czas z odchyleniem standardowym dla konwersji z formatu WAV do MP3 na różnych serwerach (źródło: opracowanie własne)

3.6.1.2 Analiza uzyskanych wyników

Na podstawie uzyskanych czasów konwersji z formatu MP3 do WAV można zauważyć, że wszystkie serwery wbudowane uzyskały bardzo podobne wyniki co sugeruje, że w przypadku tej konwersji posiadają one bardzo podobną wydajność. Serwer Jetty uzyskał średni czas na poziomie 236 ms, Tomcat uzyskał 237,3 ms, a Undertow 237,8 ms co pokazuje, że te serwery mają bardzo podobną wydajność w tym przypadku.

Najbardziej stabilnie działał serwer Undertow, którego odchylenie standardowe było najmniejsze i wynosiło 1,166 ms. Serwer Jetty posiadał trochę gorszą stabilność, uzyskując odchylenie standardowe 1,95 ms. Najmniej stabilny okazał się serwer Tomcat, którego odchylenie standardowe wyniosło 4,23 ms.

W przypadku konwersji z formatu WAV do MP3, wszystkie serwery uzyskały podobne czasy. Najkrótszy średni czas uzyskał serwer Tomcat i wynosił on 1855,3 ms, na drugim miejscu znalazł się serwer Jetty z czasem 1872,2 ms, a najgorszy okazał się serwer Undertow z wynikiem 1895 ms.

W przypadku stabilności najlepszym serwerem okazał się Jetty, którego odchylenie standardowe wyniosło 3,66 ms. Pozostałe serwery okazały się być dużo mniej stabilne i uzyskały one odchylenie standardowe kolejno dla serwera Undertow 13,11 ms oraz dla serwera Tomcat 16,82 ms.

3.6.2 Konwersja pomiędzy formatami MP3 oraz OGG

W tym podrozdziale przedstawione zostały wyniki pomiarów konwersji z formatu MP3 do OGG oraz z formatu OGG do MP3 wraz z analizą uzyskanych wyników. Każda konwersja wykorzystywała identyczną konfigurację enkodera i prezentowała ona się następująco:

- Konfiguracja enkodera dla konwersji do formatu MP3:
  - Kodek: libmp3lame
  - Przepływność: 128 kb/s
  - Kanały: 2
  - Częstotliwość próbkowania: 48 kHz
- Konfiguracja enkodera dla konwersji do formatu OGG:
  - Kodek: libvorbis
  - Przepływność: 128 kb/s
  - Kanały: 2
  - Częstotliwość próbkowania: 48 kHz

3.6.2.1 Pomiary czasów konwersji

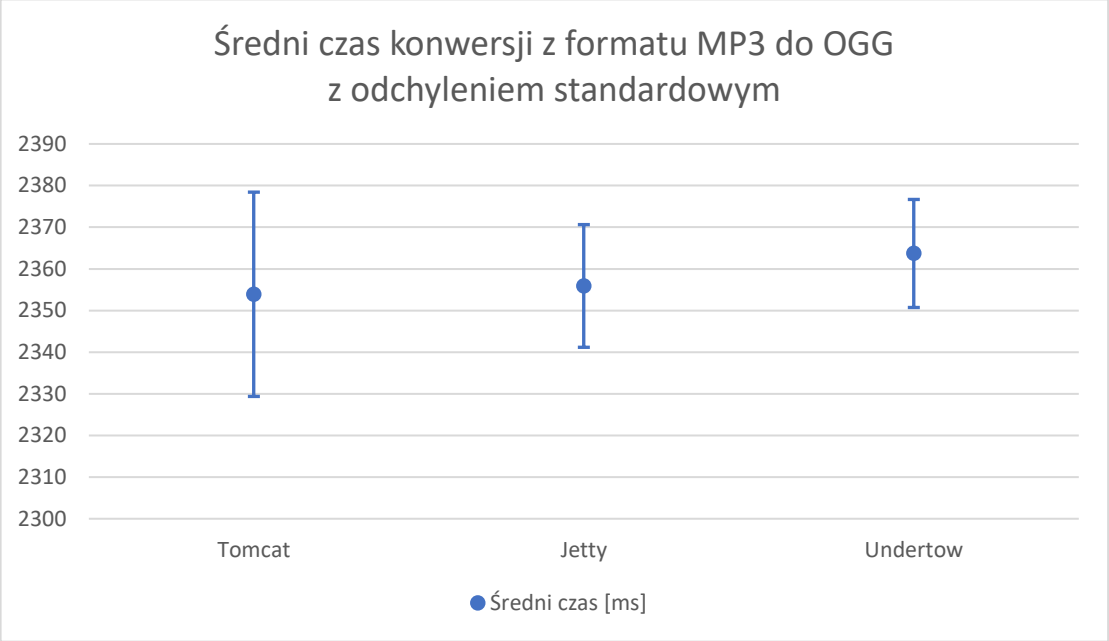
Każda konwersja została uruchomiona 5 razy a następnie z uzyskanych wyników został obliczony średni czas. Każdy pomiar konwersji z formatu MP3 do formatu OGG został przedstawiony w ***Tabeli 8***. W ***Tabeli 9*** oraz na ***Rysunku 18*** przedstawiony został średni czas oraz odchylenie standardowe dla poszczególnych serwerów.

Serwer	Pomiar [ms]									
	1	2	3	4	5	6	7	8	9	10
Tomcat	2375	2346	2335	2344	2381	2374	2347	2310	2391	2336
Jetty	2339	2342	2367	2354	2366	2330	2377	2358	2378	2348
Undertow	2341	2357	2378	2373	2370	2367	2346	2382	2364	2359

Tabela 8. Pomiary czasów konwersji z formatu MP3 do OGG

Serwer	Średni czas	Odchylenie standardowe
Tomcat	2353,9 ms	24,53 ms
Jetty	2355,9 ms	14,73 ms
Undertow	2363,7 ms	12,97 ms

Tabela 9. Średni czas wraz z odchyleniem standardowym dla konwersji z formatu MP3 do OGG



Rysunek 18. Wykres przedstawiający średni czas z odchyleniem standardowym dla konwersji z formatu MP3 do OGG na różnych serwerach (źródło: opracowanie własne)

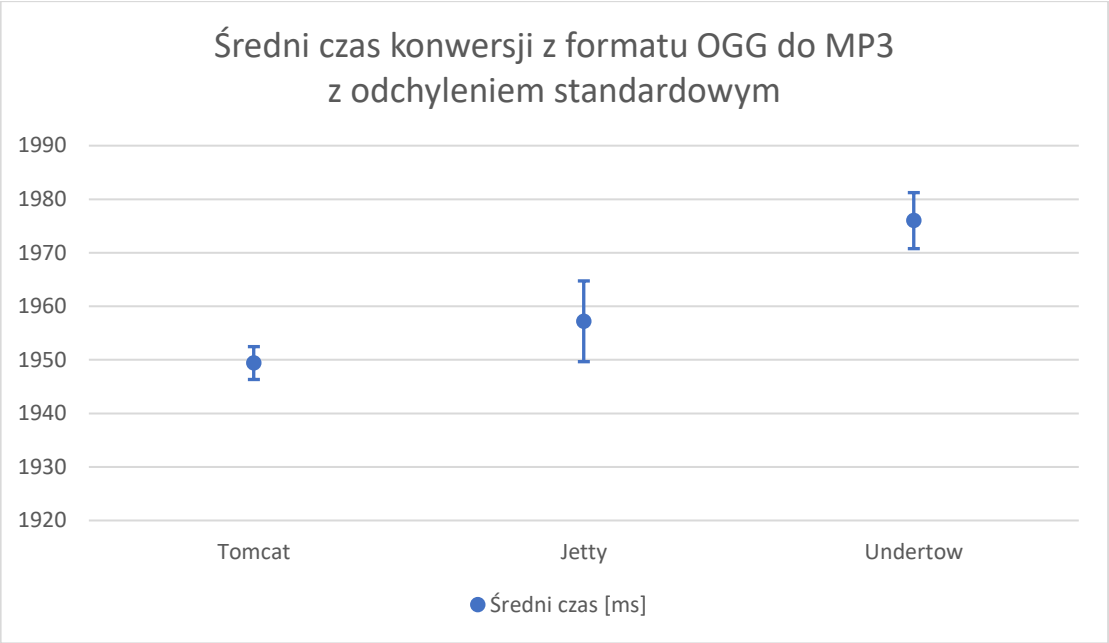
Pomiary czasów dla konwersji z formatu OGG do MP3 zostały wykonane w ten sam sposób jak w przypadku konwersji z MP3 do OGG, a uzyskane wyniki zostały przedstawione w ***Tabeli 10***. W ***Tabeli 11*** oraz na ***Rysunku 19*** przedstawiony został średni czas oraz odchylenie standardowe dla poszczególnych serwerów.

Serwer	Pomiar [ms]									
	1	2	3	4	5	6	7	8	9	10
Tomcat	1952	1944	1952	1952	1951	1948	1950	1945	1947	1953
Jetty	1948	1959	1954	1958	1948	1957	1956	1949	1945	1948
Undertow	1976	1974	1977	1965	1985	1969	1966	1973	1979	1976

Tabela 10. Pomiary czasów konwersji z formatu OGG do MP3

Serwer	Średni czas	Odchylenie standardowe
Tomcat	1949,4 ms	3,07 ms
Jetty	1957,2 ms	7,54 ms
Undertow	1976,0 ms	5,23 ms

Tabela 11. Średni czas wraz z odchyleniem standardowym dla konwersji z formatu OGG do MP3



Rysunek 19. Wykres przedstawiający średni czas z odchyleniem standardowym dla konwersji z formatu OGG do MP3 na różnych serwerach (źródło: opracowanie własne)

3.6.2.2 Analiza uzyskanych wyników

Konwersja z formatu MP3 do OGG okazała się najszybsza na serwerze Tomcat, który uzyskał średni czas przetwarzania na poziomie 2353,9 ms. Serwer Jetty uzyskał bardzo podobny wynik do serwera Tomcat i wynosił on 2355,9 ms. Najwolniejszy okazał się serwer Undertow ze średnim czasem konwersji 2363,7 ms.

Najmniejsze odchylenie standardowe zostało zaobserwowane na serwerze Undertow i wyniosło one 12,97 ms, serwer Jetty uzyskał odchylenie standardowe na poziomie 14,73 ms, a serwer Tomcat o wartości 24,53 ms co oznacza, że najbardziej stabilnie działał serwer Undertow, a najmniej stabilnie Tomcat.

W przypadku konwersji z formatu OGG do MP3 najszybszym serwerem okazał się Tomcat, którego średni czas wynosił 1949,4 ms. Podobny czas uzyskał serwer Jetty, którego średni czas wyniósł 1957,2 ms. Najwolniejszy okazał się serwer Undertow ze średnim czasem wynoszącym 1976 ms.

Najbardziej stabilne czasy konwersji uzyskał serwer Tomcat, którego odchylenie standardowe wynosiło 3,07 ms. Serwer Undertow uzyskał wynik prawie 2 razy gorszy i jego odchylenie wyniosło 5,23 ms. Najmniej stabilnie wypadł serwer Jetty, którego odchylenie standardowe było na poziomie 7,54 ms.

3.6.3 Konwersja pomiędzy formatami OGG oraz WAV

W tym podrozdziale przedstawione zostały wyniki pomiarów konwersji z formatu OGG do WAV oraz z formatu WAV do OGG wraz z analizą uzyskanych wyników. Każda konwersja wykorzystywała identyczną konfigurację enkodera i prezentowała ona się następująco:

- Konfiguracja enkodera dla konwersji do formatu OGG:
  - Kodek: libvorbis
  - Przepływność: 128 kb/s
  - Kanały: 2
  - Częstotliwość próbkowania: 48 kHz
- Konfiguracja enkodera dla konwersji do formatu WAV:
  - Kodek: pcm_s16le
  - Przepływność: 128 kb/s
  - Kanały: 2
  - Częstotliwość próbkowania: 48 kHz

3.6.3.1 Pomiary czasów konwersji

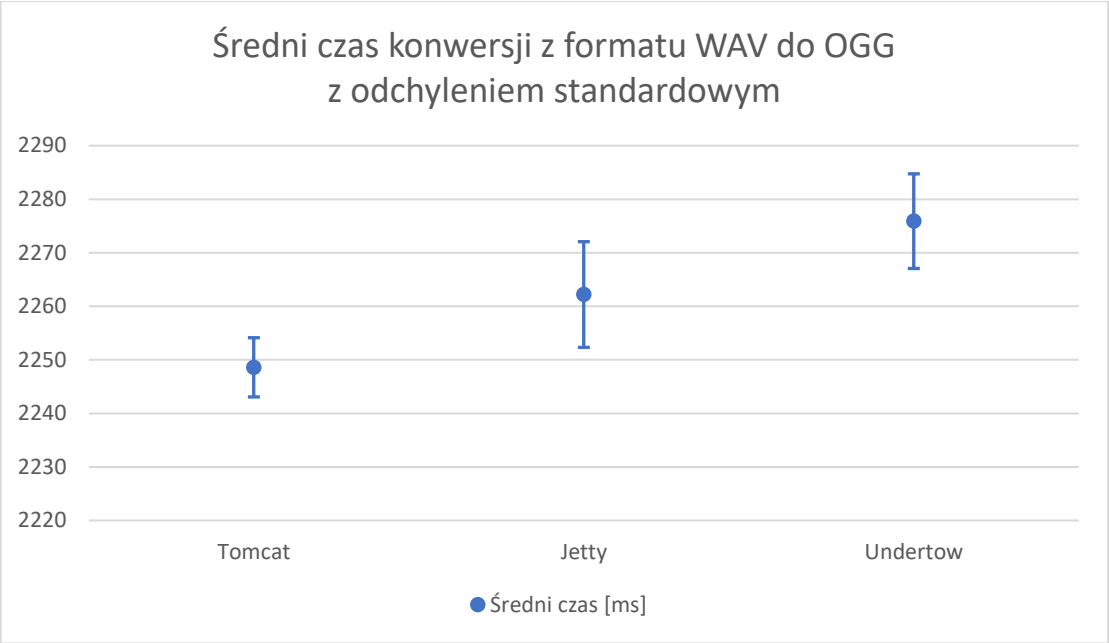
Każda kolejna konwersja została uruchomiona 5 razy, a następnie z uzyskanych wyników został obliczony średni czas tak jak w poprzednich przypadkach konwersji. Każdy pomiar dla konwersji z formatu WAV do formatu OGG został przedstawiony w ***Tabeli 12***. W ***Tabeli 13*** oraz na ***Rysunku 20*** przedstawiony został średni czas oraz odchylenie standardowe dla poszczególnych serwerów.

Serwer	Pomiar [ms]									
	1	2	3	4	5	6	7	8	9	10
Tomcat	2241	2242	2253	2244	2247	2240	2245	2244	2249	2241
Jetty	2251	2277	2246	2249	2245	2265	2249	2258	2263	2259
Undertow	2259	2267	2275	2288	2265	2271	2286	2281	2277	2275

Tabela 12. Pomiary czasów konwersji z formatu WAV do OGG

Serwer	Średni czas	Odchylenie standardowe
Tomcat	2248,6 ms	5,53 ms
Jetty	2262,2 ms	9,87 ms
Undertow	2275,9 ms	8,84 ms

Tabela 13. Średni czas wraz z odchyleniem standardowym dla konwersji z formatu WAV do OGG



Rysunek 20. Wykres przedstawiający średni czas z odchyleniem standardowym dla konwersji z formatu WAV do OGG na różnych serwerach (źródło: opracowanie własne)

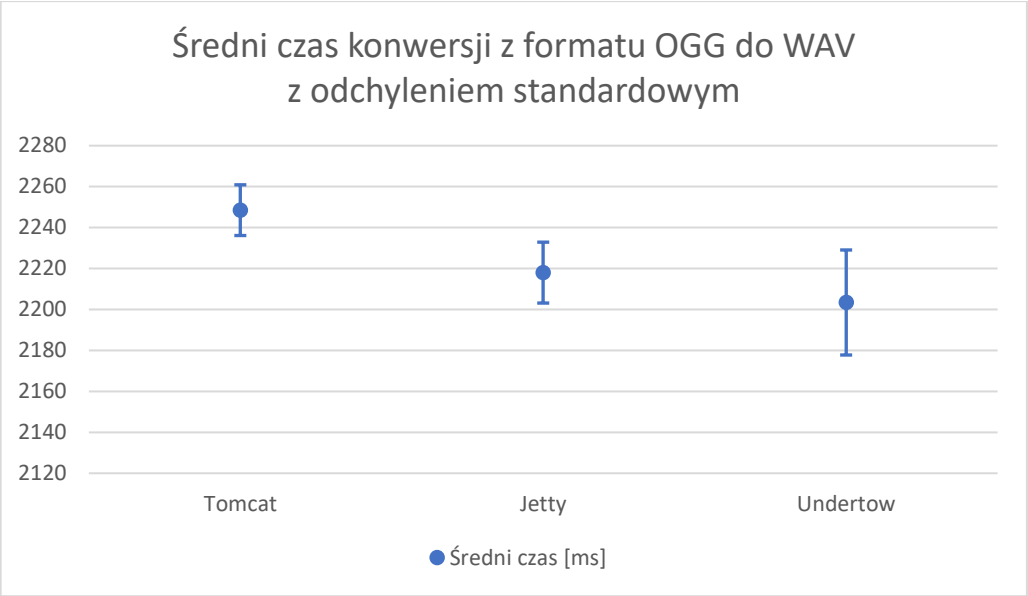
Pomiary czasów dla konwersji z formatu OGG do WAV zostały wykonane w ten sam sposób jak w przypadku konwersji z WAV do OGG oraz przedstawione zostały w ***Tabeli 14***. W ***Tabeli 15*** oraz na ***Rysunku 21*** przedstawiony został średni czas oraz odchylenie standardowe dla poszczególnych serwerów.

Serwer	Pomiar [ms]									
	1	2	3	4	5	6	7	8	9	10
Tomcat	2245	2246	2220	2258	2262	2251	2236	2264	2259	2244
Jetty	2187	2241	2219	2224	2229	2225	2203	2212	2209	2231
Undertow	2161	2180	2225	2257	2204	2201	2189	2182	2214	2221

Tabela 14. Pomiary czasów konwersji z formatu OGG do WAV

Serwer	Średni czas	Odchylenie standardowe
Tomcat	2248,5 ms	12,38 ms
Jetty	2218,0 ms	14,87 ms
Undertow	2203,4 ms	25,65 ms

Tabela 15. Średni czas wraz z odchyleniem standardowym dla konwersji z formatu OGG do WAV



Rysunek 21. Wykres przedstawiający średni czas z odchyleniem standardowym dla konwersji z formatu OGG do WAV na różnych serwerach (źródło: opracowanie własne)

3.6.3.2 Analiza uzyskanych wyników

Najszybszy średni czas konwersji z formatu WAV do OGG uzyskał serwer Tomcat z wynikiem 2248,6 ms. Serwer Jetty był już znacznie wolniejszy niż serwer Tomcat i uzyskał on 2262,2 ms. Najwolniejszy okazał się serwer Undertow, który uzyskał średni czas konwersji na poziomie 2275,9 ms.

Najbardziej stabilne wyniki zostały uzyskane na serwerze Tomcat, którego odchylenie standardowe wyniosło 5,53 ms. Serwer Jetty oraz Undertow były mniej stabilnie i uzyskały wyniki prawie 2 razy gorsze od serwera Tomcat. Odchylenie dla tych serwerów wynosiło kolejno 8,84 ms dla serwera Undertow oraz 9,87 ms dla serwera Jetty.

W przypadku konwersji z formatu OGG do WAV najlepszy średni czas uzyskał serwer Undertow i wynosił on 2203,4 ms. Konwersja na serwerze Jetty okazałą się trochę wolniejsza i wyniosła ona 2218 ms i była o prawie 15 ms dłuższa niż na serwerze Undertow. Najwolniejszy w tym przypadku okazał się serwer Tomcat, który uzyskał średni wynik na poziomie 2248,5 ms.

Najstabilniej wypadł serwer Tomcat, który uzyskał najgorszy średni czas przetwarzania, a jego odchylenie standardowe wyniosło 12,38 ms. Serwer Jetty tak jak w przypadku średniego czasu uzyskał wynik pomiędzy serwerem Tomcat oraz Undertow, a jego odchylenie standardowe było na poziomie 14,87 ms. Najmniej stabilny okazał się serwer Undertow, który uzyskał najlepszy średni czas przetwarzania, a jego odchylenie standardowe było ponad 2 razy większe od serwera Tomcat i wynosiło ono 25,65 ms.

3.7 Konwersja formatów plików wideo

W przypadku konwersji formatów wideo, wymagane było także tak jak w przypadku konwersji audio, skorzystanie z przygotowanych punktów końcowych, które odbierały żądania metodą POST. Przykładowy punkt końcowy przedstawiony został na ***Rysunku 22***. Wysłanie odpowiedniego żądania rozpoczynało konwersję pomiędzy formatami, które były przedstawione w adresie żądania. Tak jak w przypadku konwersji audio, wszystkie konwersje wideo wykorzystywały takie same pliki źródłowe przygotowane wcześniej oraz zostały one umieszczone na serwerze wraz z aplikacją.

```
@PostMapping("/mp4-to-avi")
public ResponseEntity<TimeResponse> convertMP4toAVI() {
    long startTime = System.nanoTime();
    TimeResponse timeResponse = videoService.convertMP4toAVI();
    long endTime = System.nanoTime();
    timeResponse.setTotalRequestTime(endTime - startTime);
    return ResponseEntity.ok(timeResponse);
}
```

Rysunek 22. Metoda obsługująca żądanie POST dla konwersji pomiędzy formatami MP4 oraz AVI (źródło: opracowanie własne)

Dla konwersji plików wideo przygotowane zostały także trzy pliki źródłowe po jednym na każdy wykorzystywany podczas testów format. Każdy plik posiada długość 4 minut i 46 sekund co odpowiada teledyskowi do dłuższej piosenki. Parametry tych plików wyglądały następująco:

- Parametry pliku AVI:
  - Długość: 4 minuty 46 sekund
  - Rozmiar: 70,5 MB
  - Szerokość klatki: 1920
  - Wysokość klatki: 1080
  - Liczba klatek na sekundę: 25.00
  - Szybkość transmisji bitów: 2065 kb/s
- Parametry pliku MP4:
  - Długość: 4 minuty 46 sekund
  - Rozmiar: 68,0 MB
  - Szerokość klatki: 1920
  - Wysokość klatki: 1080
  - Liczba klatek na sekundę: 25.00
  - Szybkość transmisji bitów: 1994 kb/s
- Parametry pliku MKV:
  - Długość: 4 minuty 46 sekund
  - Rozmiar: 70,3 MB
  - Szerokość klatki: 1920

- Wysokość klatki: 1080
- Liczba klatek na sekundę: 25.00
- Szybkość transmisji bitów: 2061 kb/s

3.7.1 Konwersja pomiędzy formatami MP4 oraz AVI

W tym podrozdziale przedstawione zostały uzyskane czasy konwersji z formatu MP4 do AVI oraz z formatu AVI do MP4 oraz przedstawiona została analiza uzyskanych wyników. Każda konwersja wykorzystywała identyczną konfigurację enkodera i prezentowała ona się następująco:

- Konfiguracja enkodera dla konwersji do formatu MP4:
  - Format: mp4
  - Kodek: libx264
  - Szerokość klatki: 1920
  - Wysokość klatki: 1080
  - Liczba klatek na sekundę: 25.00
- Konfiguracja enkodera dla konwersji do formatu AVI:
  - Format: avi
  - Kodek: libx264
  - Szerokość klatki: 1920
  - Wysokość klatki: 1080
  - Liczba klatek na sekundę: 25.00

3.7.1.1 Pomiary czasów konwersji

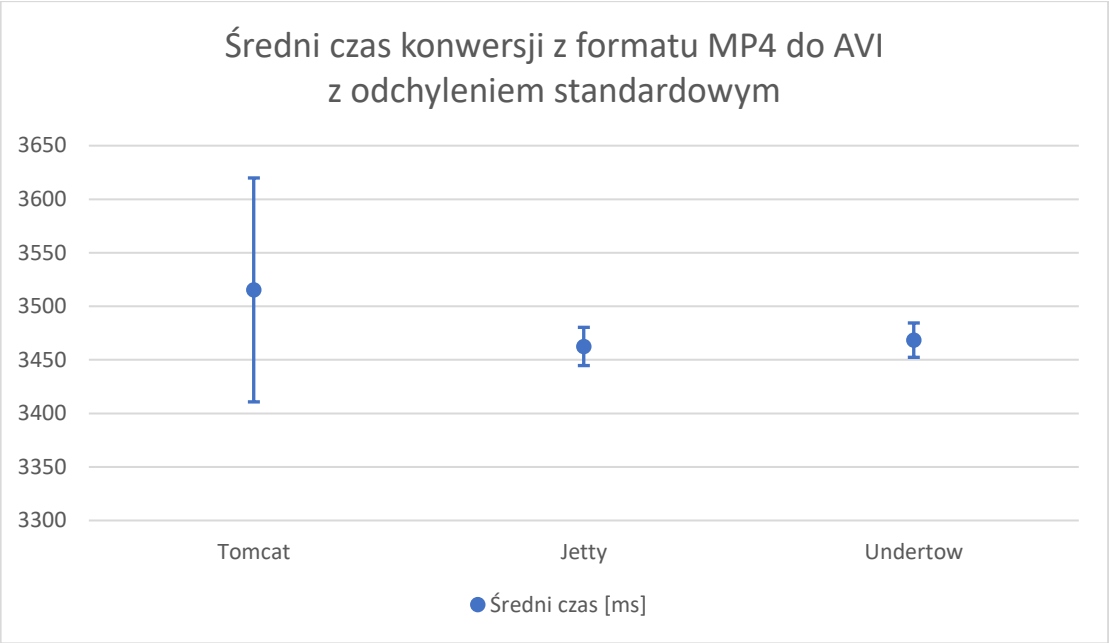
W ***Tabeli 16*** przedstawione zostały pomiary czasów konwersji z formatu MP4 do AVI na poszczególnych serwerach wbudowanych. Przeprowadzonych zostało 5 pomiarów dla każdego serwera wbudowanego oraz na ich podstawie został obliczony średni czas konwersji. W ***Tabeli 17*** oraz na ***Rysunku 23*** przedstawiony został średni czas oraz odchylenie standardowe dla poszczególnych serwerów.

Serwer	Pomiar [ms]									
	1	2	3	4	5	6	7	8	9	10
Tomcat	3325	3643	3526	3623	3445	3494	3571	3592	3510	3534
Jetty	3459	3461	3450	3455	3506	3438	3461	3455	3478	3462
Undertow	3499	3484	3463	3445	3466	3488	3432	3471	3459	3467

Tabela 16. Pomiary czasów konwersji z formatu MP4 do AVI

Serwer	Średni czas	Odchylenie standardowe
Tomcat	3515,3 ms	104,56 ms
Jetty	3462,5 ms	17,84 ms
Undertow	3468,4 ms	16,04 ms

Tabela 17. Średni czas wraz z odchyleniem standardowym dla konwersji z formatu MP4 do AVI



Rysunek 23. Wykres przedstawiający średni czas z odchyleniem standardowym dla konwersji z formatu MP4 do AVI na różnych serwerach (źródło: opracowanie własne)

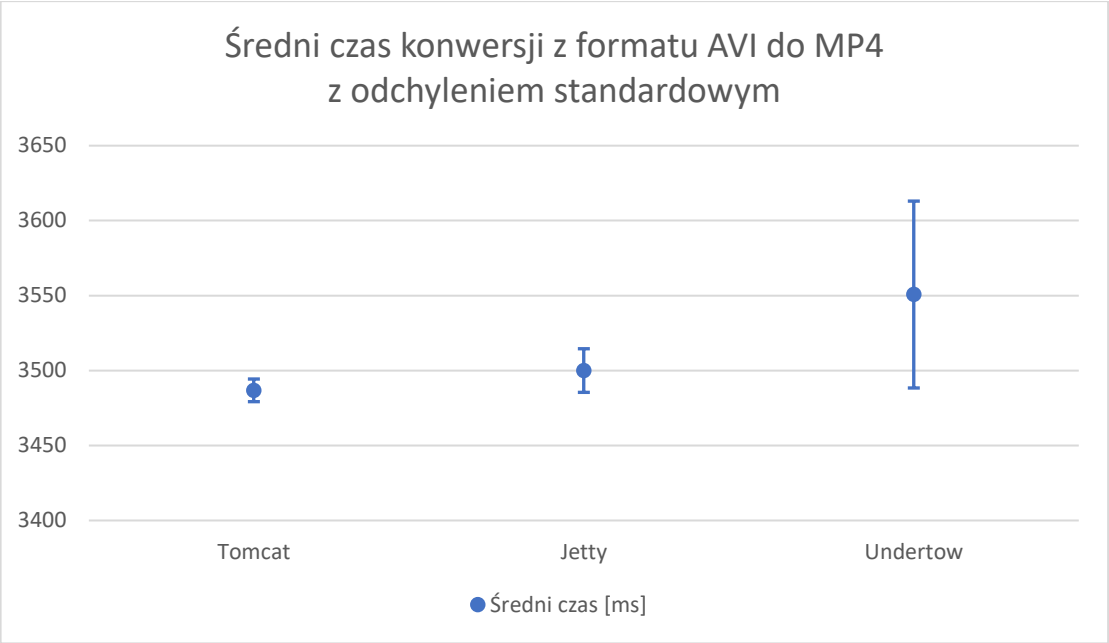
W ***Tabeli 18*** przedstawione zostały pomiary czasów konwersji z formatu AVI do MP4 na poszczególnych serwerach wbudowanych. Dla każdego z serwerów przeprowadzone zostało 5 pomiarów, a na końcu obliczony został średni czas przetwarzania. W ***Tabeli 19*** oraz na ***Rysunku 24*** przedstawiony został średni czas oraz odchylenie standardowe dla poszczególnych serwerów.

Serwer	Pomiar [ms]									
	1	2	3	4	5	6	7	8	9	10
Tomcat	3490	3487	3480	3491	3498	3482	3506	3470	3484	3470
Jetty	3498	3521	3513	3514	3515	3521	3499	3520	3515	3504
Undertow	3499	3663	3558	3501	3504	3604	3686	3555	3558	3579

Tabela 18. Pomiary czasów konwersji z formatu AVI do MP4

Serwer	Średni czas	Odchylenie standardowe
Tomcat	3486,8 ms	7,55 ms
Jetty	3500,0 ms	14,55 ms
Undertow	3550,7 ms	62.32 ms

Tabela 19. Średni czas wraz z odchyleniem standardowym dla konwersji z formatu AVI do MP4



Rysunek 24. Wykres przedstawiający średni czas z odchyleniem standardowym dla konwersji z formatu AVI do MP4 na różnych serwerach (źródło: opracowanie własne)

3.7.1.2 Analiza uzyskanych wyników

Konwersja z formatu MP4 do AVI uzyskała najmniejszy średni czas przetwarzania na serwerze Jetty i wynosił on 3462,5 ms. Nieznacznie wolniejszy okazał się serwer Undertow, który uzyskał średni czas na poziomie 3468,4 ms co pokazuje, że ten serwer oraz Jetty mają tutaj bardzo podobną wydajność. Serwer Tomcat uzyskał najwolniejszy średni czas przetwarzania i wynosił on 3515,3 ms.

Najbardziej stabilnymi serwerami w tym przypadku był Undertow, który uzyskał najmniejsze odchylenie standardowe wynoszące 16,04 ms oraz serwer Jetty z wynikiem 17,84 ms. Serwer Tomcat okazał się być mało stabilny w przypadku tej konwersji i uzyskał bardzo duże odchylenie standardowe na poziomie 104,56 ms.

Podczas konwersji z formatu AVI do MP4, serwer Tomcat uzyskał najlepszy średni czas przetwarzania, który wynosił 3486,8 ms. Serwer Jetty uzyskał wynik gorszy od serwera Tomcat o trochę ponad 13 ms i wynosił on 3500 ms. Najwolniejszy okazał się serwer Undertow uzyskując średni czas na poziomie 3550,7 ms i jest on wolniejszy od serwera Jetty o ponad 50ms.

W tym przypadku najbardziej stabilny okazał się serwer Tomcat, który w poprzedniej konwersji uzyskał najgorsze wyniki pod względem stabilności. Jego odchylenie standardowe wyniosło 7,55 ms. Serwer Jetty okazał się być mniej stabilny, a jego wynik był prawie 2 razy gorszy od serwera Tomcat i wyniósł on 14,55 ms. Serwer

Undertow w tym przypadku wypadł najgorzej, a jego wynik wynosił 62,32 ms, gdzie w przypadku poprzedniej konwersji był najbardziej stabilnym serwerem.

3.7.2 Konwersja pomiędzy formatami MP4 oraz MKV

W tym podrozdziale przedstawione zostały uzyskane czasy konwersji z formatu MP4 do MKV oraz z formatu MKV do MP4 oraz przedstawiona została analiza uzyskanych wyników. Każda konwersja wykorzystywała identyczną konfigurację enkodera i prezentowała ona się następująco:

- Konfiguracja enkodera dla konwersji do formatu MP4:
  - Format: mp4
  - Kodek: libx264
  - Szerokość klatki: 1920
  - Wysokość klatki: 1080
  - Liczba klatek na sekundę: 25.00
- Konfiguracja enkodera dla konwersji do formatu MKV:
  - Format: matroska
  - Kodek: libx264
  - Szerokość klatki: 1920
  - Wysokość klatki: 1080
  - Liczba klatek na sekundę: 25.00

3.7.2.1 Pomiary czasów konwersji

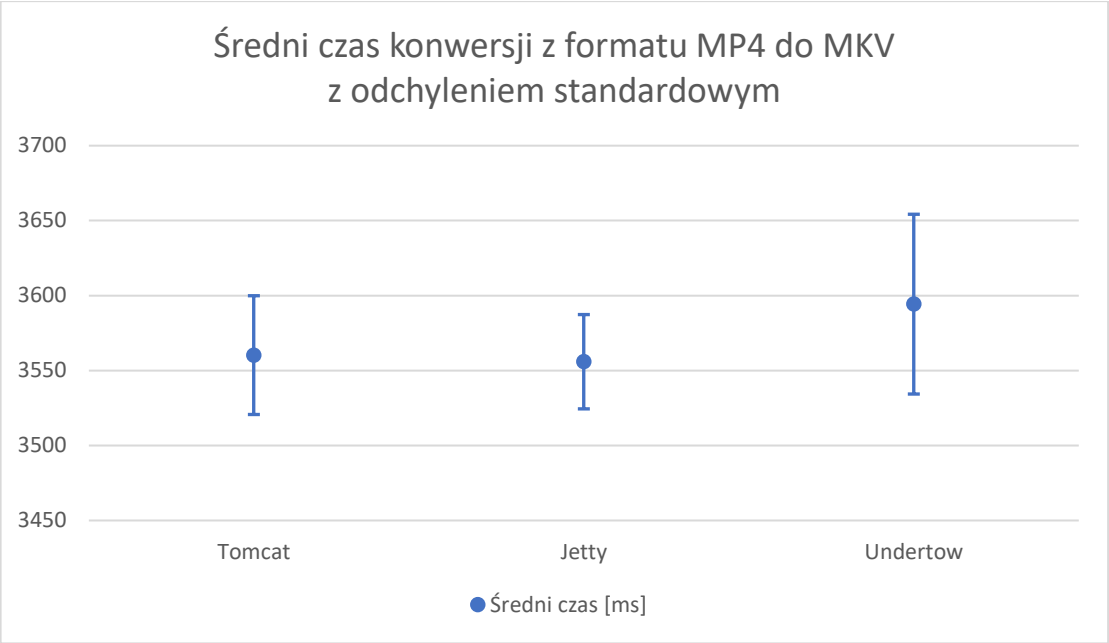
W ***Tabeli 20*** przedstawione zostały pomiary czasów konwersji z formatu MP4 do MKV na poszczególnych serwerach wbudowanych. Przeprowadzonych zostało 5 pomiarów dla każdego serwera wbudowanego oraz na ich podstawie został obliczony średni czas konwersji. W ***Tabeli 21*** oraz na ***Rysunku 25*** przedstawiony został średni czas oraz odchylenie standardowe dla poszczególnych serwerów.

Serwer	Pomiar [ms]									
	1	2	3	4	5	6	7	8	9	10
Tomcat	3534	3651	3508	3561	3549	3532	3606	3545	3534	3553
Jetty	3594	3543	3565	3523	3517	3577	3578	3512	3545	3505
Undertow	3508	3607	3514	3685	3626	3611	3608	3696	3579	3599

Tabela 20. Pomiary czasów konwersji z formatu MP4 do MKV

Serwer	Średni czas	Odchylenie standardowe
Tomcat	3560,3 ms	39,61 ms
Jetty	3555,9 ms	31,45 ms
Undertow	3594,3 ms	59,96 ms

Tabela 21. Średni czas wraz z odchyleniem standardowym dla konwersji z formatu MP4 do MKV



Rysunek 25. Wykres przedstawiający średni czas z odchyleniem standardowym dla konwersji z formatu MP4 do MKV na różnych serwerach (źródło: opracowanie własne)

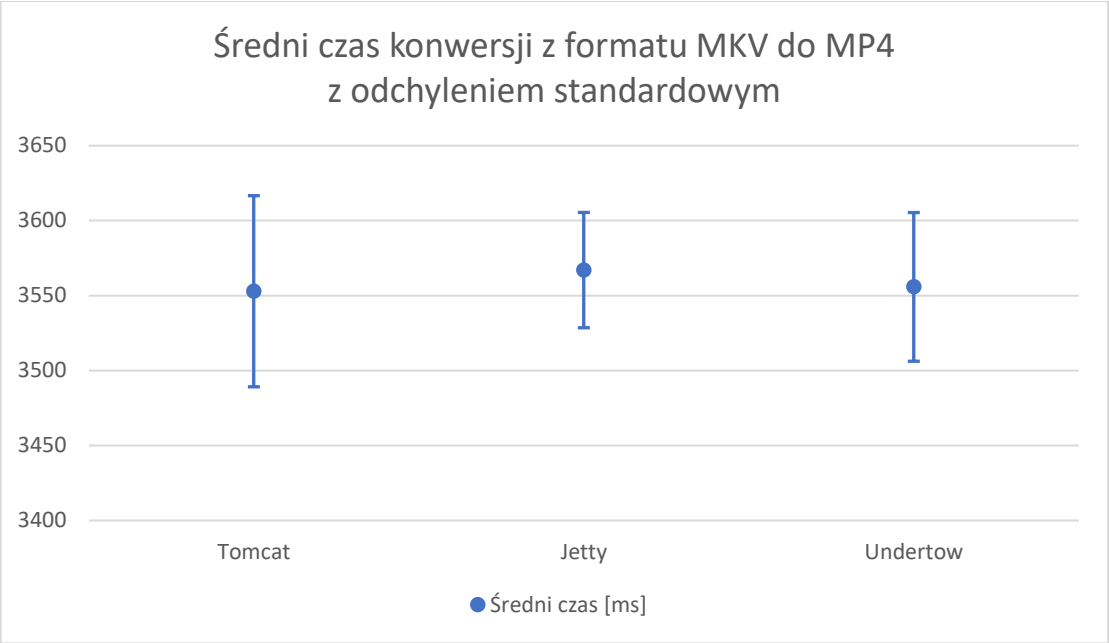
W ***Tabeli 22*** przedstawione zostały pomiary czasów konwersji z formatu MKV do MP4 na poszczególnych serwerach wbudowanych. Dla każdego z serwerów przeprowadzone zostało 5 pomiarów, a na końcu obliczony został średni czas przetwarzania. W ***Tabeli 23*** oraz na ***Rysunku 26*** przedstawiony został średni czas oraz odchylenie standardowe dla poszczególnych serwerów.

Serwer	Pomiar [ms]									
	1	2	3	4	5	6	7	8	9	10
Tomcat	3461	3689	3621	3528	3540	3588	3499	3524	3512	3567
Jetty	3511	3636	3531	3539	3548	3599	3542	3601	3579	3584
Undertow	3512	3526	3507	3509	3657	3524	3581	3549	3602	3591

Tabela 22. Pomiary czasów konwersji z formatu MKV do MP4

Serwer	Średni czas	Odchylenie standardowe
Tomcat	3552,9 ms	63,74 ms
Jetty	3567,0 ms	38,46 ms
Undertow	3555,8 ms	49,59 ms

Tabela 23. Średni czas wraz z odchyleniem standardowym dla konwersji z formatu MKV do MP4



Rysunek 26. Wykres przedstawiający średni czas z odchyleniem standardowym dla konwersji z formatu MKV do MP4 na różnych serwerach (źródło: opracowanie własne)

3.7.2.2 Analiza uzyskanych wyników

Dla konwersji z formatu MP4 do MKV, najlepszy średni czas przetwarzania został uzyskany na serwerze Jetty i wynosił on 3555,9 ms. Nieznacznie wolniejszy okazał się serwer Tomcat, którego wynik wyniósł 3560,3 ms. Serwer Undertow uzyskał średni czas na poziomie 3594,3 ms, a różnicą pomiędzy tym czasem, a wynikiem serwera Tomcat była już ponad 5 razy większa niż w pomiędzy najszybszym serwerem Jetty oraz serwerem Tomcat.

Największą stabilność pomiarów uzyskał serwer Jetty, którego odchylenie standardowe wynosiło 31,45 ms co jest i tak dość wysokim wynikiem pokazującym, że najbardziej stabilny serwer uzyskiwał wyniki z szerokiego zakresu. Serwer Tomcat uzyskał odchylenie standardowe na poziomie 39,61 ms co powoduje, że jest on mniej stabilny. Najgorzej wypadł serwer Undertow, którego odchylenie standardowe wyniosło 59,96 ms co pokazuje, że uzyskiwał on najmniej stabilne pomiary.

W przypadku konwersji z formatu MKV do MP4, najlepiej wypadł serwer Tomcat uzyskując średni czas przetwarzania na poziomie 3552,9 ms. Niewiele wolniejszy okazał się serwer Undertow, który uzyskał średni czas 3555,8 ms. Serwer Jetty był nieznacznie wolniejszy od pozostałych serwerów i uzyskał on średni czas 3567 ms.

Serwer Jetty chociaż wypadł najgorzej pod względem szybkości to uzyskał on najbardziej stabilne wyniki, a jego odchylenie standardowe wynosiło 38,46. Serwer Undertow oraz serwer Tomcat okazały się być mniej stabilne a ich wyniki wyniosły kolejno 49,59 oraz 63,74.

3.7.3 Konwersja pomiędzy formatami AVI oraz MKV

W tym podrozdziale przedstawione zostały uzyskane czasy konwersji z formatu AVI do MKV oraz z formatu MKV do AVI oraz przedstawiona została analiza uzyskanych wyników. Każda konwersja wykorzystywała identyczną konfigurację enkodera i prezentowała ona się następująco:

- Konfiguracja enkodera dla konwersji do formatu AVI:
  - Format: avi
  - Kodek: libx264
  - Szerokość klatki: 1920
  - Wysokość klatki: 1080
  - Liczba klatek na sekundę: 25.00
- Konfiguracja enkodera dla konwersji do formatu MKV:
  - Format: matroska
  - Kodek: libx264
  - Szerokość klatki: 1920
  - Wysokość klatki: 1080
  - Liczba klatek na sekundę: 25.00

3.7.3.1 Pomiary czasów konwersji

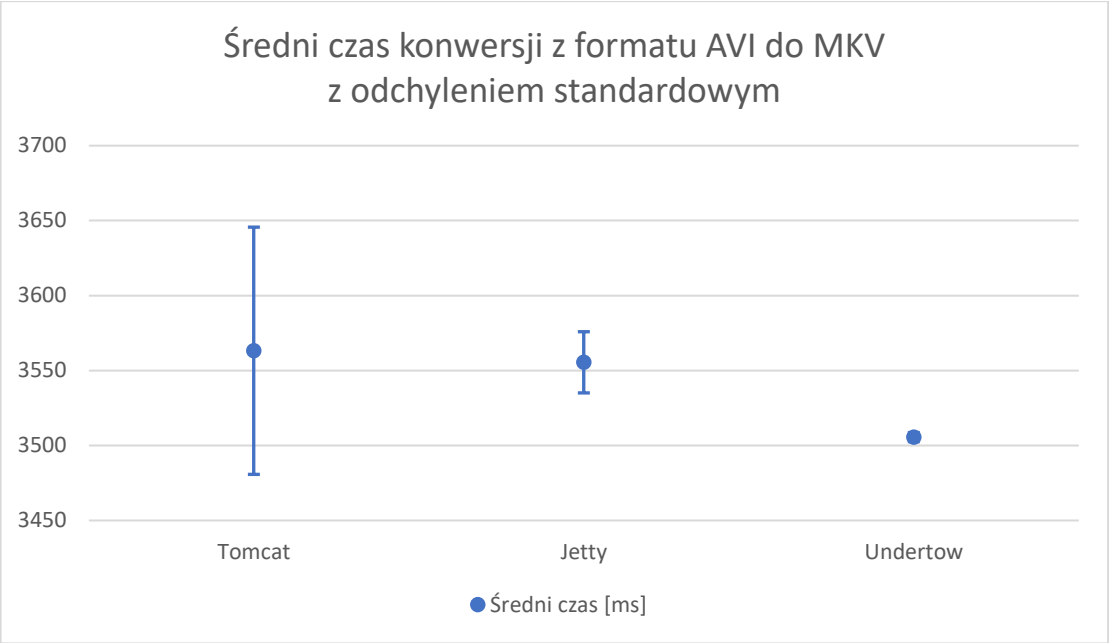
W ***Tabeli 24*** przedstawione zostały pomiary czasów konwersji z formatu AVI do MKV na poszczególnych serwerach wbudowanych. Przeprowadzonych zostało 5 pomiarów dla każdego serwera wbudowanego oraz na ich podstawie został obliczony średni czas konwersji. W ***Tabeli 25*** oraz na ***Rysunku 27*** przedstawiony został średni czas oraz odchylenie standardowe dla poszczególnych serwerów.

Serwer	Pomiar [ms]									
	1	2	3	4	5	6	7	8	9	10
Tomcat	3495	3798	3541	3537	3508	3591	3621	3641	3588	3512
Jetty	3562	3534	3544	3586	3549	3564	3591	3569	3572	3544
Undertow	3511	3502	3504	3504	3504	3509	3505	3508	3508	3501

Tabela 24. Pomiary czasów konwersji z formatu AVI do MKV

Serwer	Średni czas	Odchylenie standardowe
Tomcat	3563,2 ms	82,46 ms
Jetty	3555,5 ms	20,41 ms
Undertow	3505,6 ms	3,01 ms

Tabela 25. Średni czas wraz z odchyleniem standardowym dla konwersji z formatu AVI do MKV



Rysunek 27. Wykres przedstawiający średni czas z odchyleniem standardowym dla konwersji z formatu AVI do MKV na różnych serwerach (źródło: opracowanie własne)

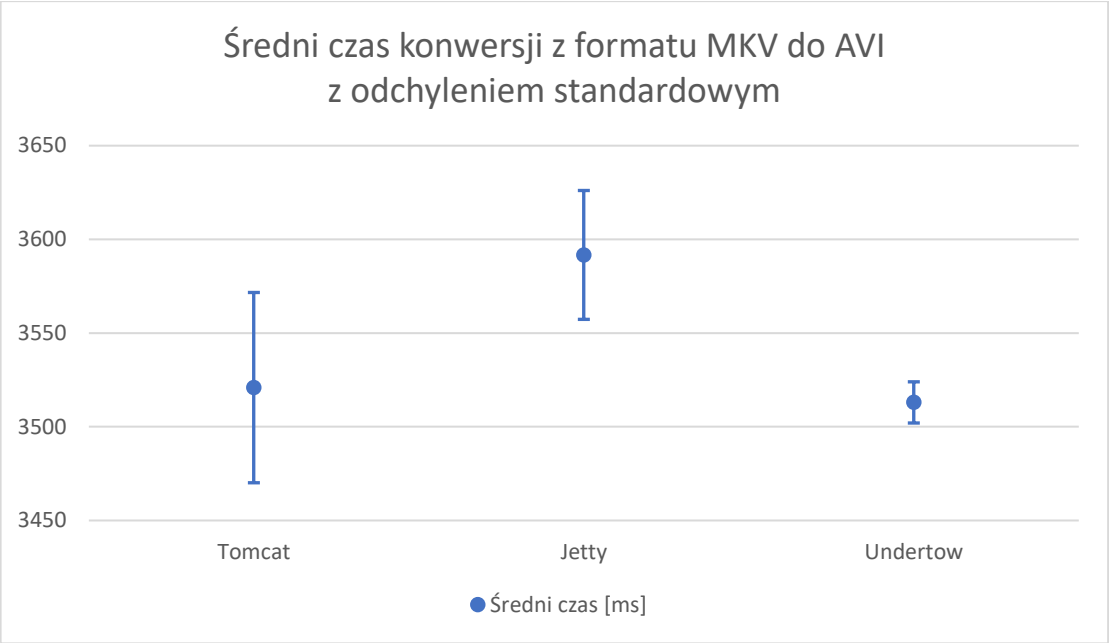
W ***Tabeli 26*** przedstawione zostały pomiary czasów konwersji z formatu MKV do AVI na poszczególnych serwerach wbudowanych. Dla każdego z serwerów przeprowadzone zostało 5 pomiarów, a na końcu obliczony został średni czas przetwarzania. W ***Tabeli 27*** oraz na ***Rysunku 28*** przedstawiony został średni czas oraz odchylenie standardowe dla poszczególnych serwerów.

Serwer	Pomiar [ms]									
	1	2	3	4	5	6	7	8	9	10
Tomcat	3553	3552	3522	3656	3466	3498	3572	3514	3485	3551
Jetty	3544	3676	3578	3541	3628	3601	3584	3597	3559	3609
Undertow	3506	3519	3531	3503	3521	3522	3530	3528	3519	3511

Tabela 26. Pomiary czasów konwersji z formatu MKV do AVI

Serwer	Średni czas	Odchylenie standardowe
Tomcat	3520,9 ms	50,77 ms
Jetty	3591,7 ms	34,39 ms
Undertow	3513,0 ms	10,99 ms

Tabela 27. Średni czas wraz z odchyleniem standardowym dla konwersji z formatu MKV do AVI



Rysunek 28. Wykres przedstawiający średni czas z odchyleniem standardowym dla konwersji z formatu MKV do AVI na różnych serwerach (źródło: opracowanie własne)

3.7.3.2 Analiza uzyskanych wyników

Podczas konwersji a formatu AVI do MKV, serwer Undertow uzyskał najkrótszy średni czas przetwarzania i wynosił on 3505,6 ms, serwer Jetty uzyskał 3555,5 ms, a najwolniejszy okazał się serwer Tomcat z wynikiem 3563,2 ms.

Serwer Undertow uzyskał najbardziej stabilne pomiary z odchyleniem standardowym na poziomie 3,01 ms co pokazuje, że nie odbiegały one bardzo od średniego wyniku. Przeskoki pomiędzy serwerami są bardzo wysokie i w przypadku serwera Jetty uzyskano wynik 20,41 ms, a w przypadku serwera Tomcat 82,46 ms co pokazuje, że radzą sobie one dużo gorzej w przypadku tej konwersji.

W przypadku konwersji z formatu MKV do AVI, średni czas przetwarzania ponownie był najkrótszy na serwerze Undertow i wynosił on 3513 ms, a największy został uzyskany na serwerze Jetty i wynosił 3591,7 ms. Tomcat uzyskał średni czas przetwarzania 3520,9 ms co umieszcza go pomiędzy pozostałymi serwerami.

W tej konwersji najmniejsze odchylenie standardowe zostało odnotowane dla serwera Undertow i wyniosło ono 10,99 ms co pokazuje, że oprócz najbardziej stabilnym pomiarów uzyskał on także najlepszy średni wynik. Serwer Jetty oraz Tomcat okazały się być mniej stabilne i uzyskały one odchylenie standardowe wynoszące kolejno 34,39 ms oraz 50,77 ms.

4 Podsumowanie pracy

W tym rozdziale przedstawione zostało zbiorcze zestawienie wyników uzyskanych podczas wykonywania testów wraz z przedstawieniem wniosków końcowych. Serwery wbudowane zostały przedstawione w zakresie szybkości działania oraz stabilności uzyskanych wyników pomiarów.

4.1 Zbiorcze zestawienie uzyskanych wyników

Wyniki zostały podzielone ze względu na wykorzystywane formaty podczas konwersji. Dla każdego rodzaju konwersji przedstawiony został serwer, który uzyskał najlepszy średni czas konwersji oraz serwer, który cechował się największą stabilnością pomiarów.

Dla konwersji pomiędzy formatami audio, najlepiej wypadł serwer Tomcat, który uzyskał najlepszy średni czas w 4 z 6 konwersji oraz cechował się najlepszą stabilnością również w 3 z 6 przypadków. Serwer Undertow okazał się być mniej wydajny i uzyskał najlepszy wynik w 1 z 6 konwersji dla średniego czasu oraz 2 z 6 w przypadku stabilności. Najgorzej wypadł serwer Jetty, który najlepszy średni czas uzyskał tylko w przy jednej konwersji i w przypadku stabilności też najlepszy był tylko w jednym przypadku. Szczegółowe wyniki zostały przedstawione w ***Tabeli 28***.

Rodzaj konwersji	Najlepszy średni czas	Najlepsza stabilność
Z formatu MP3 do WAV	Jetty	Undertow
Z formatu WAV do MP3	Tomcat	Jetty
Z formatu MP3 do OGG	Tomcat	Undertow
Z formatu OGG do MP3	Tomcat	Tomcat
Z formatu OGG do WAV	Tomcat	Tomcat
Z formatu WAV do OGG	Undertow	Tomcat

Tabela 28. Zestawienie najlepszych serwerów w konwersji formatów audio.

W przypadku konwersji pomiędzy formatami wideo, serwer Jetty uzyskał najlepszy średni czas w 2 z 6 konwersji oraz najlepszą stabilność w 2 z 6 konwersji. Serwer Tomcat okazał się w tym przypadku najgorszy uzyskując najlepszy średni czas w 2 z 6 przypadkach oraz najlepszą stabilność tylko w 1 z 6 konwersji. Najlepszy okazał się serwer Undertow i uzyskał on najlepszy średni czas przetwarzania w 2 z 6 konwersji oraz najlepszą stabilność ponownie w 3 z 6 konwersji. Szczegółowe wyniki zostały przedstawione w ***Tabeli 29***.

Rodzaj konwersji	Najlepszy średni czas	Najlepsza stabilność
Z formatu MP4 do AVI	Jetty	Undertow
Z formatu AVI do MP4	Tomcat	Tomcat
Z formatu MP4 do MKV	Jetty	Jetty
Z formatu MKV do MP4	Tomcat	Jetty
Z formatu AVI do MKV	Undertow	Undertow
Z formatu MKV do AVI	Undertow	Undertow

Tabela 29. Zestawienie najlepszych serwerów w konwersji formatów wideo.

4.2 Wnioski końcowe

Na podstawie uzyskanych wyników nie można jednoznacznie określić, który serwer wbudowany jest najlepszy do przeprowadzania różnego rodzaju konwersji audio oraz wideo. Każdy z testowanych serwerów był dobry w niektórych konwersjach, a w innych okazywało się, że pozostałe radziły sobie lepiej.

Niektóre testy pokazały, że dany serwer osiągał najlepszy średni czas przetwarzania wraz z najlepszą stabilnością uzyskiwanych wyników pomiarów co oznacza, że jest on najlepszym wyborem w dla konkretnej konwersji. Serwer Undertow okazał się być najlepszy w konwersji z MKV do AVI oraz AVI do MKV. W przypadku serwera Tomcat, poradził on sobie najlepiej w konwersjach z formatu AVI do MP4, OGG do WAV oraz OGG do MP3. Serwer Jetty okazał się najlepszy podczas konwersji z formatu MP4 do MKV. W pozostałych przypadkach najlepszy średni czas był uzyskiwany przez inny serwer niż ten, który cechował się najlepszą stabilnością.

Podsumowując wybór odpowiedniego serwera wbudowanego powinien zależeć od tego z jakich formatów będziemy chcieli korzystać podczas konwersji. Każdy serwer ma swoje mocne strony w odniesieniu do konkretnych formatów w przypadku średniego czasu konwersji jak i stabilności.

5 Bibliografia

[1]. Apache Tomcat: https://tomcat.apache.org/, data dostępu: 16.05.2023

[2]. Apache Tomcat - Powered By: https://cwiki.apache.org/confluence/display/TOMCAT/PoweredBy, data dostępu: 16.05.2023

[3]. Assumpção, H. O. (2013). *Getting started with IntelliJ IDEA.* Pack Publishing

[4]. Brandenburg, K. (2001, 03). *MP3 and AAC Explained.* Journal of the Audio Engineering Society.

[5]. Docker: https://www.docker.com/, data dostępu: 21.08.2023

[6]. Docker - Build with Docker: https://docs.docker.com/build/guide/intro/, data dostępu: 21.05.2023

[7]. Docker - Overview: https://docs.docker.com/get-started/overview/, data dostępu: 21.05.2023

[8]. Eclipse - Jetty Project: https://www.eclipse.org/jetty/, data dostępu: 05.06.2023

[9]. FFmpeg - dokumentacja: https://ffmpeg.org/ffmpeg-codecs.html, data dostępu: 14.04.2023

[10]. FFmpeg - informacje: https://ffmpeg.org/about.html, data dostępu: 14.04.2023

[11]. FFmpeg - Libavcodec: https://ffmpeg.org/libavcodec.html, data dostępu: 14.08.2023

[12]. FFmpeg - Libavdevice: https://ffmpeg.org/libavdevice.html, data dostępu: 14.08.2023

[13]. FFmpeg - Libavfilter: https://ffmpeg.org/libavfilter.html, data dostępu: 14.08.2023

[14]. FFmpeg - Libavformat: https://ffmpeg.org/libavformat.html, data dostępu: 14.08.2023

[15]. FFmpeg - Libavutil: https://ffmpeg.org/libavutil.html, data dostępu: 14.08.2023

[16]. FFmpeg - Libswersample: https://ffmpeg.org/libswresample.html, data dostępu: 14.08.2023

[17]. FFmpeg - Libswscale: https://ffmpeg.org/libswscale.html, data dostępu: 14.08.2023

[18]. FileFormat - AVI: https://docs.fileformat.com/video/avi/, data dostępu: 29.05.2023

[19]. FileFormat - MP4: https://docs.fileformat.com/video/mp4/, data dostępu: 25.05.2023

[20]. FileFormat - Ogg Vorbis Audio: https://docs.fileformat.com/audio/ogg/, data dostępu: 22.05.2023

[21]. FileFormat - WAV: https://docs.fileformat.com/audio/wav/, data dostępu: 23.05.2023

[22]. Gutierrez, F. (2016). *Pro Spring Boot.* Apress.

[23]. IntelliJ IDEA: https://www.jetbrains.com/idea/, data dostępu: 21.08.2023

[24]. IntelliJ IDEA - Features: https://www.jetbrains.com/idea/features/, data dostępu: 21.05.2023

[25]. JAVE2 - repozytorium: https://github.com/a-schild/jave2, data dostępu: 14.04.2023

[26]. JAVE2 - Usage: https://github.com/a-schild/jave2/wiki/Usage, data dostępu: 05.06.2023

[27]. Library Of Congress - Matroska Multimedia Container: https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml, data dostępu: 21.04.2023

[28]. Library Of Congress - MP3 (MPEG Layer III Audio Encoding): https://www.loc.gov/preservation/digital/formats/fdd/fdd000012.shtml, data dostępu: 21.05.2023

[29]. Library Of Congress - MPEG-4 File Format: https://www.loc.gov/preservation/digital/formats/fdd/fdd000155.shtml, data dostępu: 25.05.2023

[30]. Library Of Congress - Wave Audio File Format: https://www.loc.gov/preservation/digital/formats/fdd/fdd000001.shtml, data dostępu: 23.05.2023

[31]. Manelli, L., Zambon, G. (2020). *Beginning Jakarta EE Web Development: Using JSP, JSF, MySQL, and Apache Tomcat for Building Java Web Applications.* Apress

[32]. Matroska - Basics: https://www.matroska.org/technical/basics.html, data dostępu: 21.04.2023

[33]. Matroska - Codec Mappings: https://www.matroska.org/technical/codec_specs.html, data dostępu: 21.04.2023

[34]. Maven - Standard Directory Layout: https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html, data dostępu: 21.05.2023

[35]. Microsoft - AVI RIFF File Reference: https://learn.microsoft.com/en-us/windows/win32/directshow/avi-riff-file-reference, data dostępu: 29.05.2023

[36]. MP3 Tech - Frame header: http://www.mp3-tech.org/programmer/frame_header.html, data dostępu: 22.05.2023

[37]. Musib, S. (2021). *Spring Boot in Practice Version 9.* Manning Publications.

[38]. Netty - Home: https://netty.io/index.html, data dostępu: 15.08.2023

[39]. OpenLiberty: https://openliberty.io/, data dostępu: 15.08.2023

[40]. Oracle - WebLogic Server: https://www.oracle.com/pl/java/weblogic/, data dostępu: 15.08.2023

[41]. Reddy, K. Siva Prasad, Upadhyayul, S. (2023). *Beginning Spring Boot 3.* Apress

[42]. Schenker, G., N. (2023). *Docker: Up and Running: Build and deploy containerized web apps with Docker and Kubernetes.* BPB Publications.

[43]. Spring - Intrduction to Spring Framework: https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/overview.html, data dostępu: 07.06.2023

[44]. Undertow - Home: https://undertow.io/, data dostępu: 22.05.2023

[45]. Undertow - Architecture Overview: https://undertow.io/undertow-docs/undertow-docs-2.1.0/index.html#architecture-overview, data dostępu: 06.06.2023

[46]. Walls, C. (2015). *Spring Boot in Action.* Manning.

[47]. Westerveld, D. (2021). *API Testing and Development with Postman.* Packt Publishing

[48]. WildFly: https://www.wildfly.org/, data dostępu: 15.08.2023

[49]. Varanasi, B. (2020). *Introducing Maven - A Build Tool for Today's Java Developers.* Apress

[50]. Xiph - Vorbis: https://xiph.org/vorbis/, data dostępu: 22.05.2023

[51]. Xiph - Ogg Documentation: https://xiph.org/ogg/doc/oggstream.html, data dostępu: 22.05.2023

[52]. Xiph - Ogg: https://xiph.org/ogg/, data dostępu: 22.05.2023

Spis tabel

Spis rysunków