Politechnika Krakowska

Wydział Inżynierii Elektrycznej i Komputerowej

Dokumentacja

**Programowanie w języku Java**

**Projekt: Aplikacja do nakładania filtrów na zdjęcia**

**Wykonali:**

Paweł Wądolny

Marcin Zieliński

PK WIEiK stacjonarne informatyka rok 3

Spis treści

[1. Cel i zakres projektu: 3](#_Toc504476943)

[2. Charakterystyka użytkowników 3](#_Toc504476944)

[3. Główne funkcje produktu 3](#_Toc504476945)

[4. Wymagania Funkcjonalne 3](#_Toc504476946)

[5. Wymagania Niefunkcjonalne 4](#_Toc504476947)

[6. Architektura i technologie 4](#_Toc504476948)

[7. Połączenie urządzeń 4](#_Toc504476949)

[8. GUI 5](#_Toc504476950)

[8.1 GUI Klient 5](#_Toc504476951)

[8.2 GUI Scheduler Server 5](#_Toc504476952)

[8.3 GUI Processing Server 5](#_Toc504476953)

[9. Klasy i metody 5](#_Toc504476954)

[10. Wyniki z działania systemu 5](#_Toc504476955)

[11. Raporty z szeregowania 5](#_Toc504476956)

[12. Tabele zbiorcze z testami wydajności 5](#_Toc504476957)

[13. Wnioski 5](#_Toc504476958)

1. Cel i zakres projektu:

Celem projektu jest stworzenie aplikacji w języku Java do nakładania filtrów na zdjęcia. Aplikacja będzie działała w następujący sposób: Host (klient) po wybraniu pliku graficznego oraz filtru będzie łączył się z serwerem kolejkującym (Scheduler Server), który sprawdzi dostępność serwerów przetwarzających (Processing Server) oraz ich obciążenie i prześle adres wybranego serwera do klienta, a ten następnie połączy się z serwerem przetwarzającym. Nastąpi przesłanie pliku obrazu do serwera, który po odebraniu obrazu nałoży na niego wybrany filtr, a następnie odeśle zmienioną fotografię do klienta.

Aplikacja musi być stabilna oraz prosta w obsłudze. Aplikacja będzie podzielona na 3 typy działania:

* Scheduler Server – Przechowuje adresy serwerów przetwarzających, komunikuje się z nimi w celu określenia ich obciążenia. Z tym serwerem łączy się klient, aby uzyskać dane serwera, do którego wyśle plik graficzny.
* Processing Server – serwer przetwarzający, jego zadaniem jest udzielenie informacji serwerowi kolejkującemu oraz odebranie, przetworzenie i odesłanie pliku graficznego klienta.
* Host – Aplikacja kliencka, umożliwia wybranie zdjęcia z dysku twardego oraz filtru do nałożenia. Aplikacja łączy się z Schedulerem w celu uzyskania adresu serwera przetwarzającego. Następnie łączy się z tym serwerem, wysyła wybrany plik oraz informację o wybranym filtrze. Po otrzymaniu pliku zwrotnego z serwera zapisuje go na dysku.

Zakres projektu obejmuje stworzenie aplikacji z intuicyjnym GUI. Aplikacja dodatkowo będzie posiadała symulator hostów/klientów w celu zasymulowania obciążenia serwera. Aplikacja ta będzie uruchamiana na jednym komputerze i przy wykorzystaniu wielowątkowości zostanie zasymulowane obciążenie serwerów.

2. Charakterystyka użytkowników

* **Administrator** – jego zadaniem jest połącznie serwerów oraz wpisanie w pliku konfiguracyjnym adresów serwerów (Schedulera i Processing), a następnie uruchomienie serwerów.
* **Użytkownik** – ma możliwość korzystania z aplikacji Host. Po wybraniu danego zdjęcia i filtru otrzyma ze serwera przetworzony plik graficzny.

3. Główne funkcje produktu

1. Serwer kolejkujący (Scheduler Server)
2. Serwer przetwarzający (Processing Server)
3. Aplikacja kliencka (Host)
4. Aplikacja symulująca klientów i testująca obciążenie serwerów (HostSimulator)
5. Przetwarzanie obrazu przy użyciu wybranego filtru

4. Wymagania Funkcjonalne

1. Łączność z serwerem kolejkującym
2. Łączność z serwerem przetwarzającym
3. Przesyłanie przez klienta pliku do serwera
4. Przetwarzanie obrazu na serwerze
5. Odesłanie wyniku do klienta

5. Wymagania Niefunkcjonalne

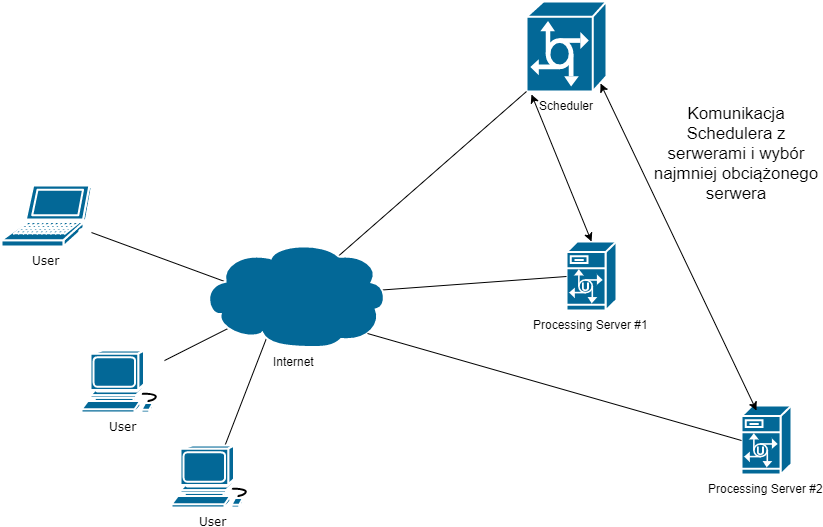
1. Stabilność – aplikacja musi działać niezawodnie
2. Intuicyjność – aplikacja powinna być przyjazna dla użytkownika; niedoświadczony klient nie może mieć problemów z obsługą
3. Łatwość instalacji – system powinien być możliwy do wdrożenia w jeden dzień roboczy

6. Architektura i technologie

1. Łączność pomiędzy serwerami i klientami będzie zrealizowana przy użyciu protokołu TCP/IP
2. GUI Hosta będzie stworzone przy użyciu JavaFX
3. Plik po przetworzeniu będzie kasowany z serwera
4. Aplikacja będzie obsługiwała typy plików graficznych: \*.jpg, \*.jpeg, \*.png, \*.bmp
5. Maksymalny rozmiar pliku będzie ograniczony do około 10-20MB, daje to około   
   10000px x 10000px dla formatów jpg, jpeg. Zabieg ten jest wprowadzony w celu podzielenia mocy obliczeniowej na klientów oraz uwarunkowany wystąpieniem błędu java.lang.OutOfMemoryError przy pracy z bardzo dużymi plikami (np. plik \*.bmp 600MB 20000px x 10000px).
6. Wstępnie dostępne filtry: Sepia, Negatyw, MirrorImage, Black&White, Red Image, Green Image, Blue Image.

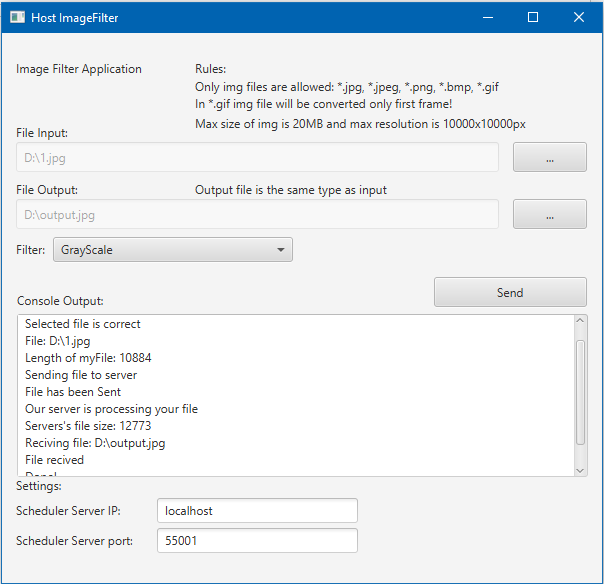
Jeżeli znajdziemy inne modyfikacje obrazu godne uwagi zostaną one zawarte w projekcie.

7. Połączenie urządzeń



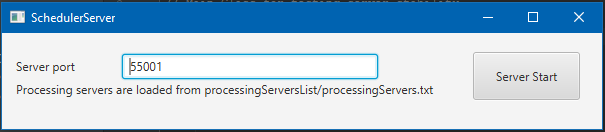
8. GUI

## 8.1 GUI Klient

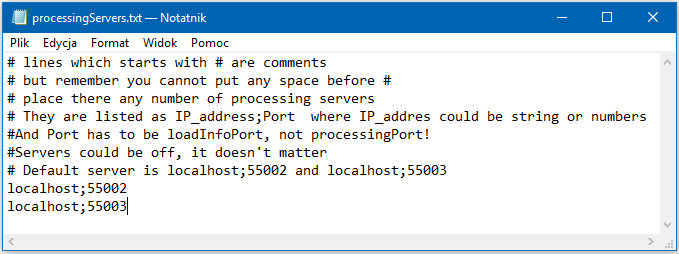


W aplikacji hosta wybieramy ścieżkę wejściową i ścieżkę wyjściową oraz z listy wybrany filtr. Następnie naciskamy Send. Aplikacja skomunikuje się z schedulerem i otrzyma od niego adres IP najmniej obciążonego serwera.

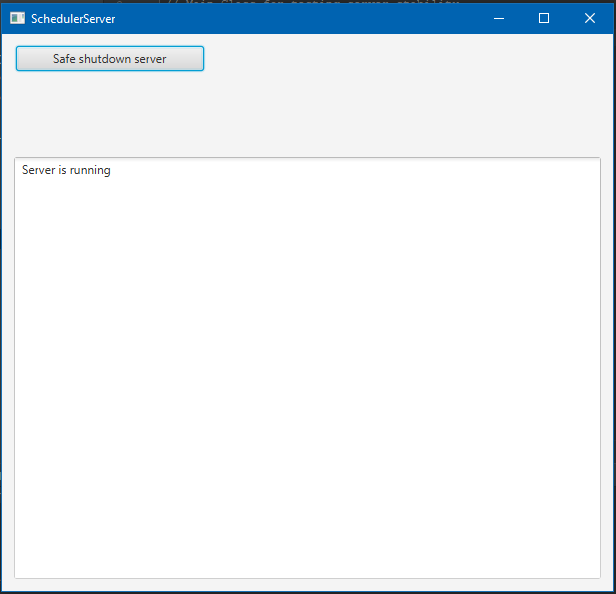
## 8.2 GUI Scheduler Server



Po uruchomieniu aplikacji mamy możliwość zmiany portu, na którym będzie nasłuchiwał serwer lub możemy zostawić domyślnie wpisany port 55001. Po kliknięciu przycisku „server start” z pliku processingServers.txt. Oto przykładowa zawartość pliku:

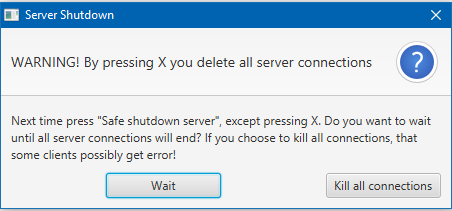


Jeżeli plik z lista serwerów nie istnieje zostanie on automatycznie wygenerowany, aby użytkownik przy następnym uruchomieniu mógł wpisać własne serwery. Po starcie serwera ujrzymy ekran:



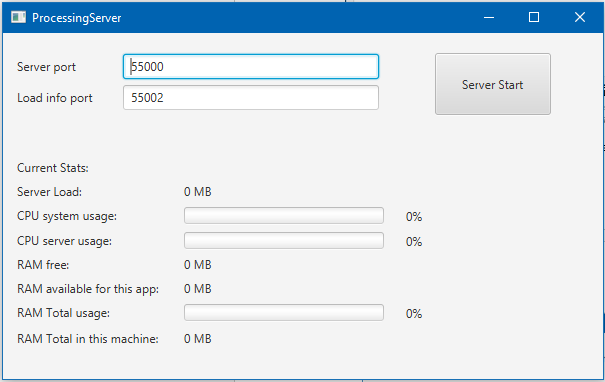
Zawiera on obszar do wypisywania powiadomień ( rozwiązanie przyszłościowe, na razie nic nie jest w nim wyświetlane oprócz informacji, że serwer nasłuchuje). Drugim elementem jest przycisk do bezpiecznego wyłączenia serwera, powoduje on wyłączenie aplikacji, ale aplikacja oczekuje jeszcze na zamknięcie wszystkich wątków.

Jeżeli klikniemy X w prawym górnym roku to wyświetli się nam komunikat:



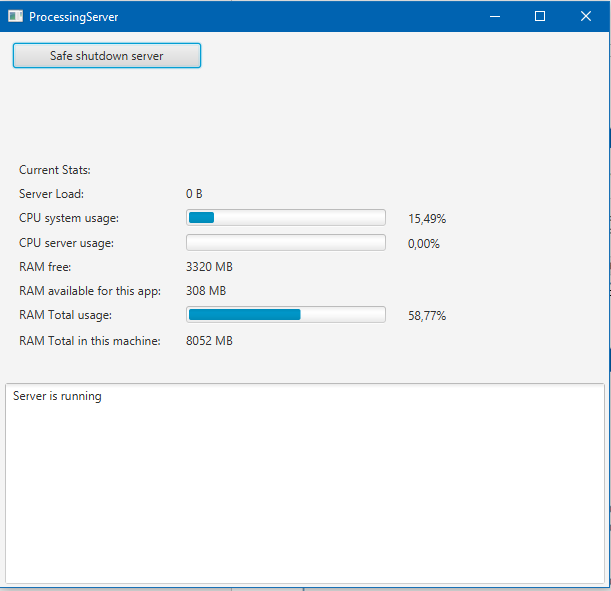
Mamy tutaj możliwość zaczekania na zakończenie wszystkich połączeń lub możemy zakończyć natychmiastowo wszystkie.

## 8.3 GUI Processing Server



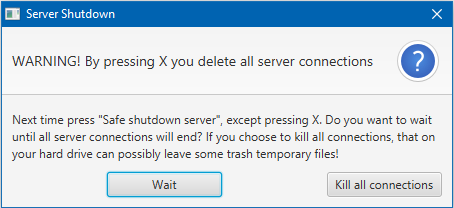
Po uruchomieniu aplikacji mamy możliwość wpisania portów na jakich serwer przetwarza obrazy oraz portu, na którym udziela informacji o obciążeniu.

Po kliknięciu przycisku „Server Start” otrzymujemy następujący ekran:



Na bieżąco (z odświeżaniem co sekundę, aby zbytnio nie obciążyć dodatkowo komputera) są aktualizowane dane o aktualnym obciążeniu serwera. Ponadto aplikacja tak jak w przypadku schedulera zawiera dodatkowy panel do wyświetlania komunikatów, aczkolwiek tak jak w przypadku schedulera jest wyświetlana tylko informacja, że serwer nasłuchuje. Ta opcja jest przygotowana do wykorzystania w przyszłości, aby można było wyświetlać powiadomienia.

Aplikacja tak jak scheduler zawiera przycisk do bezpiecznego wyłączenia serwera (wtedy wszystkie obrazy, które nie zostały przetworzone zostaną do końca przerobione, a następnie apliakcja zostanie zamknięta). Jeżeli naciśniemy X ujrzymy komunikat:



Jeżeli zdecydujemy zakończyć natychmiastowo wszystkie połączenia i w tym momencie będzie miało miejsce przetwarzanie jakiegoś obrazu, wtedy aplikacja zostanie zamknięta, ale na naszym komputerze pozostaną nie usunięte pliki tymczasowe.

9. Sposób działania aplikacji

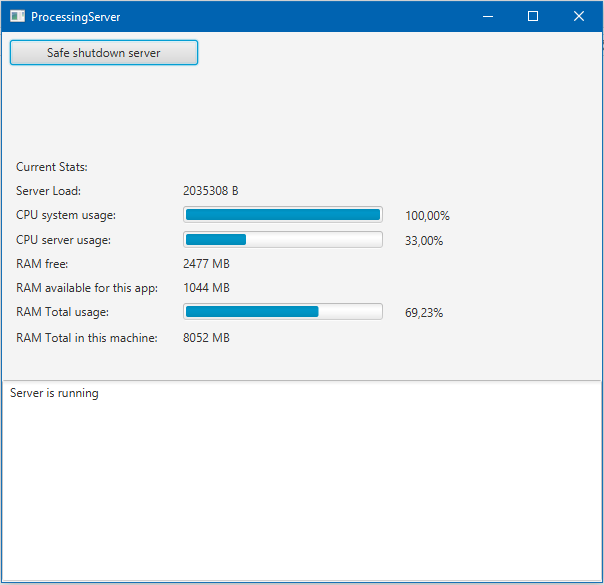
* Przed uruchomieniem schedulera należy wpisać na listę wszystkie serwery przetwarzające, ale należy mieć na uwadze, że jeżeli wpiszemy dużo serwerów, które są wyłączone, może to nieznacznie spowalniać aplikację, gdyż scheduler wybierając najlepszy serwer odpytuje wszystkie serwery z listy, nawet te wyłączone.
* Uruchamiamy wszystkie serwery (min. 1 scheduler i dowolna ilość processing)
* Uruchamiamy aplikację kliencką
* Po wyborze wszystkich danych wejściowych w aplikacji klienckiej jest sprawdzana poprawność pliku
* Jeżeli zostaną spełnione warunki – odpowiedni format pliku oraz rozmiar < 20MB, to klient połączy się z schedulerem, od którego otrzyma adres IP i Port najmniej obciążonego serwera przetwarzającego
* Scheduler pod odebraniu informacji od klienta odpytuje wszystkie serwery przetwarzające o ich aktualne obciążenie i wybiera najmniej obciążony. Obciążenie serwera jest wybierane na podstawie aktualnego zużycia procesora   
  przez system (w %).
* Gdy scheduler odpyta już wszystkie serwery przetwarzające o obciążenie to odsyła do klienta adres IP i Port wybranego serwera lub adres 0.0.0.0 i port 0, gdy żaden serwer nie jest dostępny.
* Po odebraniu informacji przez klienta o serwerze sprawdza on, czy adres jest poprawny, jeżeli wynosi on 0.0.0.0:0 to na ekranie zostanie wyświetlony komunikat, że serwer jest niedostępny.
* Jeżeli adres IP jest poprawny, wtedy Klient łączy się z serwerem processing.
* Klient wysyła informację o obrazie i wybrany typ konwersji a następnie przesyła sam obraz.
* Serwer przetwarzający sprawdza obraz, czy jest poprawny, a następnie go przetwarza i wysyła informację do klienta oraz przetworzony obraz, po czym kasuje pliki tymczasowe z dysku.
* Pliki tymczasowe są przechowywane jako numerKlienta.RozszerzeniePliku [plik otrzymany od klienta] oraz numerKlientaOut.RozszerzeniePliku [przetworzony plik]
* Numer klienta jest nadawany automatycznie przez serwer w kolejności od 0, aż do maksymalnej wartości long. Dlatego należy unikać uruchamiania jednocześnie 2 serwerów przetwarzających na tym samym komputerze z tego samego folderu, gdyż jeżeli to zrobimy to pliki w tym samym folderze mogą zostać nadpisane jeżeli w obu aplikacjach na raz zostanie wywołany np. klient nr.100.
* Jeżeli w trakcie przetwarzania dojdzie do błędu (np. plik będzie niepoprawny) to do klienta zostanie odesłana wiadomość, że z jego plikiem graficznym są problemy.
* Klient otrzymuje przetworzony obraz i zapisuje go na dysku jako ścieżka wyjściowa podana w odpowiednim polu aplikacji.

10. Klasy i metody

11. Wyniki z działania systemu

12. Testy wydajności

1. Serwery zostały przetestowane za pomocą klasy testowej. Przykładowy screen pokazujący aktualne obciążenie serwera:



Objaśnienie obciążenia:

* Server Load – aktualna wielkość (w Bajtach) przetwarzanych zdjęć.
* CPU system usage – aktualny % zużycia procesora na danej maszynie
* CPU server usage – aktualny % zużycia procesora przez serwer, jeżeli wynosi np. 33% to tyle % zużywa aktualnie nasz program i wszystkie jego wątki.
* RAM free- ilość dostępnego RAMu w danej maszynie
* RAM availible for this app – wielkość RAM jaką Java Virtual Machine zarezerwowała dla naszego programu (ile jeszcze RAMu nasz serwer może dostać od JVM)
* RAM total usage – ilość RAMu jaka jest aktualnie używana przez dany komputer.
* RAM Total – ilość RAMu jaka jest zamontowana w danej maszynie.

Serwer został poddany 3 krotnej próbie obciążenia 1000 wątkami z plikiem w wielkości 11KB (format jpg, rozdzielczość: 198x254 piksele), z których 2 powiodły się sukcesem i od serwera zostały odebrane wszystkie 1000plików. W 1 z 3 prób serwer zwrócił tylko 908 plików. Po przebadaniu sytuacji testy wykazały zwracanie błędu:

IO: Connection refused: connect

Server is unavailable

Spowodowane jest to prawdopodobnie uruchamianiem wszystkich aplikacji na 1 komputerze, przez co próba nawiązania połączenia gubi się po drodze.

1. Serwer został poddany próbie obciążenia 10 watkami z plikiem o wielkości 2212KB (plik jpg rozdzielczośc 10000x10000pikseli)

Niestety zaalokowanie pamięci dla jednego obrazu to: około 100MB w jednym bloku + dodatkowe 100MB na plik wyjściowy (lub jeżeli jest to mirror image to 200MB). Jak pokazały testy aplikacja na 10 wysłanych plików zwróciła jedynie 4, reszta otrzymała błąd:

Out of memory: Java heap space

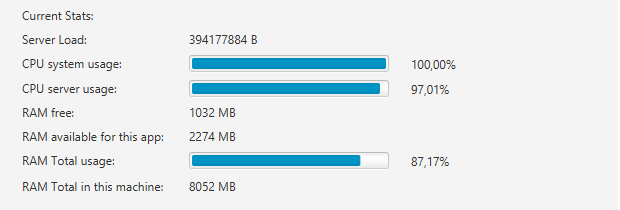
Z powodu wymienionego wyżej. W celu zapobiegnięcia tej sytuacji należało by zrobić kolejkę oczekującą na przydział pamięci zamiast wysyłania od razu błędu do klienta.

1. Serwer został obciążony plikiem png o rozdzielczości 1920x1080 i wielkości 3MB.

Po obciążeniu 200 wątkami, zostały zwrócone 187 obrazy.

W 13 przypadkach wystąpił błąd:

javax.imageio.IIOException: Error reading PNG image data



Jak widać na powyższym obrazie serwer został obciążony, zasoby procesora w pewnym momencie zostały w 100% przydzielone aplikacji, co spowodowało chwilowe zawieszenie muzyki odtwarzanej z przeglądarki internetowej w tle a także całego interfejsu graficznego. Nie udało się nam uchwycić tej chwili, najwyższą wartość jaką udało się uzyskać na screenie wynosi 97.01%.

13. Wnioski

Program działa poprawnie, aczkolwiek należało by w niej poprawić sposób przydzielania pamięcią dla danych obrazów, gdyż w przypadku przepełnienia RAMu aplikacja nie zwróci poprawnych wyników. Pozostałe aspekty działają poprawnie. Serwery łączą się ze sobą oraz z klientem, obrazy są przesyłane.