PROJEKT

ROZPROSZONE SYSTEMY OPERACYJNE

*FAZA III*

Zespół Mechatroniczni Wojownicy:

Dżumaga Rafał,

Grudzień Ewelina,

Gwiazdowicz Artur,

Jóźwik Mateusz,

Misiowiec Piotr,

Poćwierz Maciej,

Stepek Katarzyna

Politechnika Warszawska

2016-06-09

Spis treści

[**Opis rozwiązania Docker** 2](#_Toc449559798)

[Czym jest kontener 2](#_Toc449559799)

[Czym jest Docker 3](#_Toc449559800)

[Architektura Dockera 3](#_Toc449559801)

[Podstawowe pojęcia 3](#_Toc449559802)

[Docker Toolbox 4](#_Toc449559803)

[Możliwe stany kontenera 6](#_Toc449559804)

[Docker PID Namespace 6](#_Toc449559805)

[Docker Volumes 7](#_Toc449559806)

[Logowanie 7](#_Toc449559807)

[Docker Swarm 8](#_Toc449559808)

[Bezpieczeństwo 9](#_Toc449559809)

[**Raport z testów dockera** 9](#_Toc449559811)

[1. Test prostego przykładu 9](#_Toc449559812)

[2. Test współpracujących kontenerów 10](#_Toc449559813)

[3. Test tworzenia klastra 11](#_Toc449559814)

[**Koncepcja i architektura rozwiązania** 12](#_Toc449559815)

[Role 13](#_Toc449559816)

[Wymaganie niefunkcjonalne **Błąd! Nie zdefiniowano zakładki.**](#_Toc449559817)

[Wymaganie funkcjonalne 13](#_Toc449559818)

[Architektura systemu 14](#_Toc449559819)

[**Harmonogram** 15](#_Toc449559820)

[**Organizacja środowiska programistycznego projektu** 16](#_Toc449559821)

[**Bibliografia:** 16](#_Toc449559822)

FAZA III

# **Dotychczasowa dokumentacja**

We wcześniejszych etapach realizacji projektu, przygotowano

1. Faza-I\_updated.pdf – opisuje kroki podjęte w ramach realizacji   
   fazy I projektu
2. Haronogram.pdf – zaplanowany harmonogram przyjęty na początku realizacji projektu
3. Faza-II-raport.pdf – raport z prac nad projektem zrealizowanych do dnia 27.05.2016
4. scenariuszeTestowe.pdf

# **Szczegółowa koncepcja rozwiązania**

W ramach projektu realizowany był system do zbierania głosów z wyborów parlamentarnych. System otrzymuje paczki danych z głosowania z różnych okręgów i zapisuje je do rozproszonej bazy danych. Dane są następnie cyklicznie przetwarzane, a wyniki przetwarzania zapisywane do kolejnej bazy danych. Wyniki analizy udostępniane są użytkownikowi systemu (administrator danych), które może je oglądać w przeglądarce internetowej. System realizowany jest w środowisku zdalnym. Wszystkie komponenty systemu, oprócz aplikacji generującej dane, umieszczone są w kontenerach Dockera na systemie operacyjnym Ubuntu w chmurze – na serwerach Amazona. Bezpieczeństwo zapewniane jest przez ograniczenie dostępuj do usług i baz danych za pomocą listy akceptowanych adresów IP.

System składa się z następujących komponentów:

## Aplikacja generująca dane (0)

Jes to aplikaja pomocnicza, której zadaniem jest wygenerowanie dużej ilości danych (????? Ile??????) i wysłanie ich za pomocą zapytania http POST w postaci JSON do rozproszonej bazy danych.

## Rozproszona baza danych (1)

Głowna baza danych systemu realizowana jest za pomocą MongoDB. Na bazę składają się minimum 3 węzły. Istnieje możliwość zwiększenia tej liczby przez rekonfigurację systemu.

Na każdym węźle znajdują się aplikacja wystawiająca usługę RESTową, na którą load balancer Amazona przekierowuje zapytanie http z danymi. Dane są następnie zapisywane do bazy za pomocą zatwierdzania dwufazowego. Za pomocą konfiguracji ustawia się też poziom redundancji. Wynosi on minimalnie 1, czyli dane z jednego zatwierdzania umieszczane są zawsze na co najmniej dwóch węzłach.

## Aplikacja statystyczna (2)

Aplikacja ta przetwarza cyklicznie dane z rozproszonej bazy danych i wyniki przetwarzania zapisuje lokalnie do bazy PostgreSQL. Aplikacja oraz baza danych działają dwóch węzłach, na wypadek awarii jednego z nich.

## Aplikacja webowa (3)

Zadaniem tej aplikacji jest udostępnianie usługi umożliwiającej administratorowi danych oglądanie w przeglądarce wyników analizy odczytanych z bazy PostreSQL. Aby zapewnić dostępność aplikacji 2 i 3 postanowiono uruchomić aplikację 3 jednocześnie na węźle master i shadow. Aplikacja 2 uruchomiona jest w danej chwili tylko na jednym węźle, ale lokalna baza danych dostępna jest na obu węzłach. Jeśli master działa bez problemu, to aplikacja 2 przesyła regularnie do węzła shadow wyniki analizy tak, by obie bazy zawierały ten sam zestaw danych. W przypadku awarii węzła master, aplikacja 2 uruchamiana jest na węźle shadow i kontynuuje cykliczne przetwarzanie. Gdy awaria mastera zakończy się, aplikacja 2 ponownie działa tylko na masterze. Odpytuje ona węzeł shadow o ilość danych w bazie danych i pobiera od niego brakujące dane, jeśli takie istnieją.

Szczegóły techniczne implementacji systemu znajdują się w załączniku „Dokumentacja techniczna”

# **Organizacja pracy**

Pracę nad projektem uławiał nam portal bitrix24.com, na którym dzieliliśmy się ze sobą informacjami i zrobionymi postępami. Można tam było też przydzielać członkom zespołu zadania do zrealizowania w określonym terminie.  
  
Prace nad realizacją projektu przebiegały następująco:

## [1] pierwsze spotkanie – 2 kwiecień

Ustalono wstępnie temat projektu oraz przydział ról.  
  
Realizowany będzie system do głosowania i analizowania wyników wyborów.  
System będzie składać się z 3 części:  
1. Aplikacja ładująca dane z głosowania z poszczególnych okręgów do rozproszonej bazy danych  
2. Aplikacja analizująca wyniki wyborów   
3. Aplikacja udostępniająca administratorowi wyniki analizy  
  
Założenia techniczne:

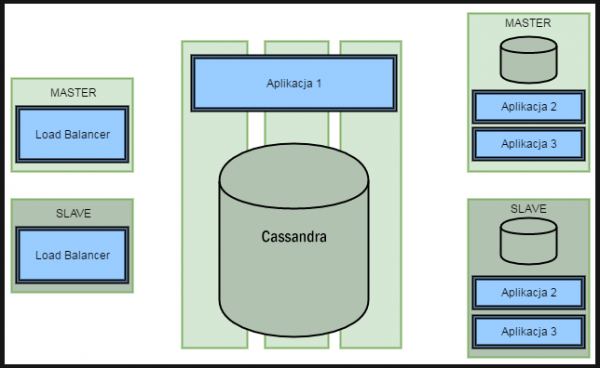
* Dane wejściowe będą generowane dla każdego okręgu wyborczego a następnie wysyłane batchowo
* Wstępnie przyjęto, że dane z głosowania będą zawierały: pesel, głos, okręg, płeć, wiek, adres zameldowania, wykształcenie
* Cześć 1 realizowana będzie za pomocą JAVA + Spring [Linux]
* Cześć 2 realizowana będzie za pomocą JAVA [Windows]
* Cześć 3 realizowana będzie za pomocą C# i JavaScriptu [Windows]

## [2] Drugie spotkanie – 5 kwiecień

Po krótkie konsultacji z prowadzącym, koncepcja uległa zmianie.

Okazało się, że:

* Nie możemy w pełnie skorzystać z mechanizmów synchronizacji/rozpraszania danych oferowanych przez Cassandrę. Powinniśmy sami zaimplementować niektóre mechanizmy np mechanizm elekcji – skorzystamy z bazy MongoDB
* Load balancer może rozdzielać ruch korzystając z Round Robin
* Dokumentacja projektu systemu nie musi być bardzo rozbudowana i bogata w schematy – może być słownie, bez diagramów
* Prowadzący popiera ideę master-slave (a raczej zamiast slave powinno się to nazywać shadow) oraz dedykowanego węzła do load balancera
* Do przechowywania danych z głosowania w sposób redundantny użyta zostanie rozproszona, no-sqlowa baza danych Cassandra
* Do wersjonowania użyty zostanie Git, a repozytorium umieszczone na GitHubie
* Load balancer będzie działał na dwóch węzłach - master i slave; slave będzie działał tylko, gdy master przestranie działać
* Aplikacja 1 i baza danych umieszczone będą na 3 węzłach
* Aplikacja 2 i 3 umieszczone będą na dwóch węzłach - master i slave; slave będzie działał tylko, gdy master przestranie działać
* Aplikacja 2 będzie miała swoją lokalną, scentralizowaną i nieredundantną bazę danych

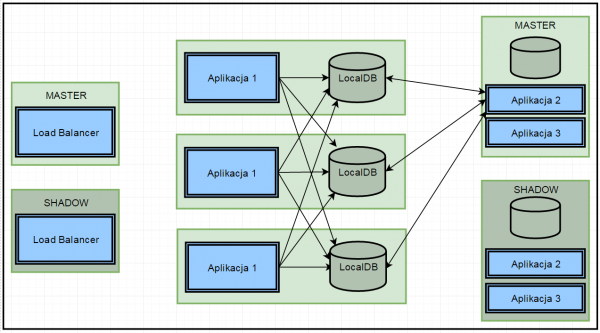
Schemat systemu:  


Rysunek 1

Pytania do konsultacji z prowadzącym:

* Czy można skorzystać z Cassandry?
* Jak realizować load balancer? Round robin?
* Czy aplikacja 2 i 3 może być postawiona na windowsie?

Koncepcja systemu wygląda obecnie następująco:



Rysunek 2

## [3] 5-28 kwiecień

W tym okresie przygotowywaliśmy się do oddania etapu I:

* Lider przygotował harmonogram
* Rozdzieliliśmy między siebie zadania
* Projektanci przygotowali wymagania funkcjonalne i niefunkcjonalne
* Architekci przygotowali koncepcję prototypu (Rys. 3) – omówione to zostało podczas krótkiego *statusu* przeprowadzonego online (za pomocą Skype’a)   
  19 kwietnia
* Spec od Dockera przygotował jego opracowanie i przeprowadził proste testy.
* Spec od repozytorium stworzył repozytorium na portalu github.com pod adresem <https://github.com/veiar/rso_voting>
* Architekcie wstępnie wybrali algorytm zatwierdzania dwufazowego, jako algorytm odpowiedzialny za utrzymanie spójności danych w rozproszonej bazie danych



## [4] Oddanie I etapu – 28 kwiecień

Rysunek 3

[5] 28 kwiecień – 19 maj

Podjęte zostały następujące kroki:

* Spec od danych stworzył prototyp aplikacji generującej dane
* Architekci przygotowali prototyp aplikacji wystawiającej usługę przyjmującą wygenerowane dane oraz dokonującej zatwierdzanie dwufazowe
* Spec od Dockera zaczął migrację prototypów do kontenerów Dockerowych
* Artur podjął się dodatkowego zadania i przygotował uproszczoną, lokalną namiastkę serwera DNS na potrzeby tłumaczenie domeny naszego systemu na adresy IP adresów dwóch load balancerów.
* Spece od aplikacji webowej przegotowali prototyp usługi

[6] Trzecie spotkanie – 19 maj

W czasie spotkanie omówiono i przetestowano to, co już udało się przygotować. Testy odbyły się w sieci lokalnej.

Dodatkowo:

* Architekci opracowali ostateczną koncepcję przechowywania danych przez aplikację analityczną.
* Generator danych został dopracowany
* Wspólnie przedyskutowano możliwe scenariusze testowe oraz kwestie związane z konfiguracją
* Podjęto decyzję o próbie przeniesienia systemu do środowiska   
  zdalnego (Amazon)
* Zrezygnowano z idei serwera DNS. Zdecydowano się skorzystać zamiast tego z pliku /etc/hosts na komputerze użytkownika systemu.

[7] 19 – 27 maj

Tester przygotował scenariusze testowe

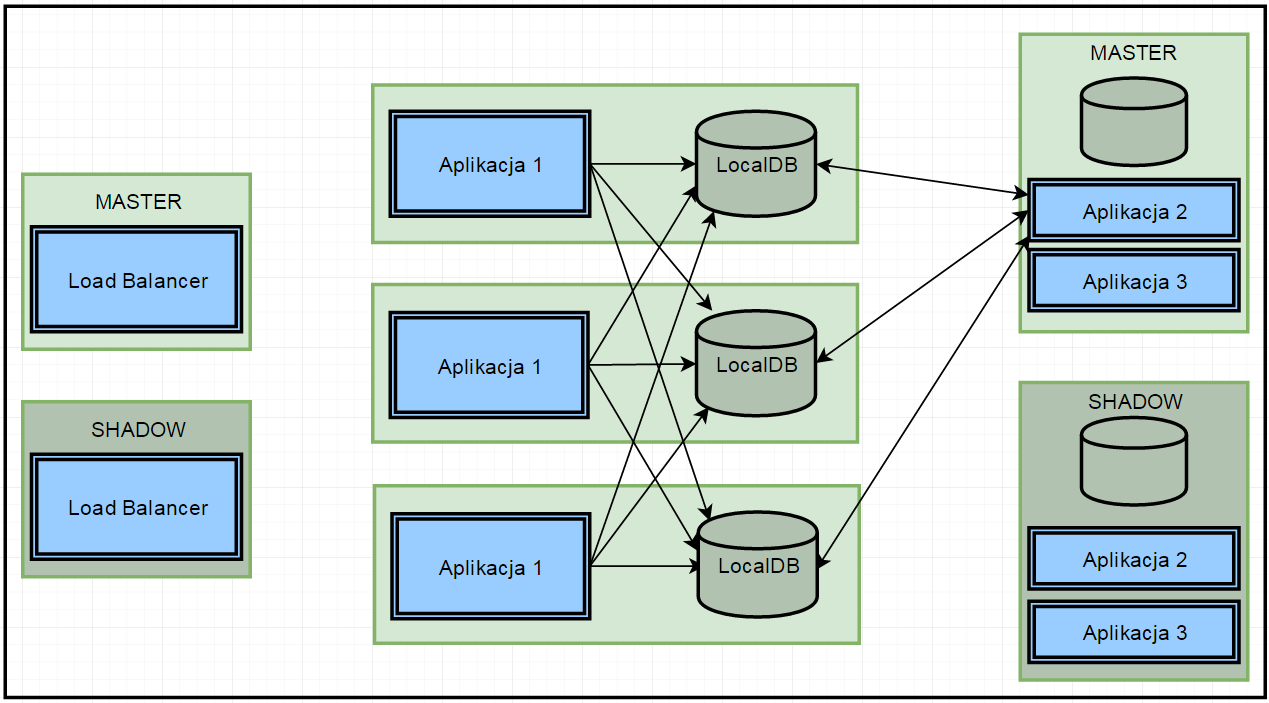
Spec od aplikacji analitycznej utworzył właściwą schemę w PostgreSQL i razem ze specem od generowania danych przygotowali słowniki

Załadowano dane do baz MongoDB oraz PostgreSQL

Spec od aplikacji analitycznej zaimplementował funkcjonalność przetwarzania i agregowania danych z pojedynczego węzła rozproszonej bazy i zapisywania jej do PostgreSQL

Spece od aplikacji webowej zaimplementowali w wersji bazowej wyświetlanie przetworzonych danych.

[8] Wysłanie raportu z dotychczasowych postępów prac – 27 maj



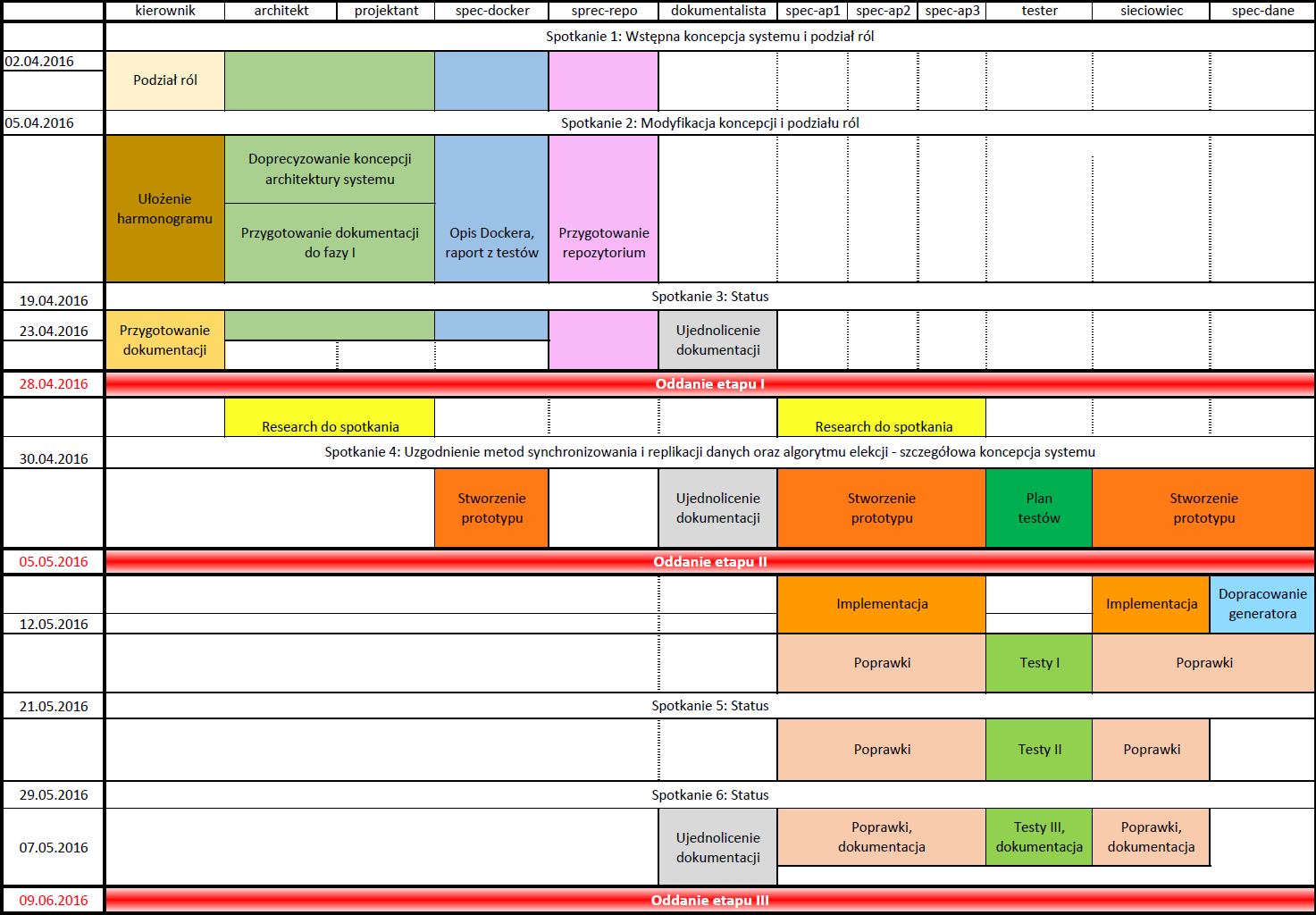
Wstępna koncepcja architektury systemu

# **Harmonogram**

Podział ról w zespole:

1. Kierownik – Katarzyna Stepek
2. Architekt – Artur Gwiazdowski, Maciej Poćwierz
3. Projektant – Ewelina Grudzień, Katarzyna Stepek
4. Spec od repozytorium – Artur Gwiazdowski
5. Dokumentalista – Katarzyna Stepek
6. Tester – Maciej Poćwierz
7. Handlowiec – Piotr Misiowiec
8. Spec od dockera – Mateusz Jóźwik
9. Programista aplikacji 1  - Artur Gwiazdowicz
10. Programista aplikacji 2  - Piotr Misiowiec
11. Programista aplikacji 3 – Rafał Dżumaga, Ewelina Grudzień
12. Spec od Load balancera i https – Mateusz Jóźwik
13. Generowanie danych wejściowych – Ewelina Grudzień

Z powodu dużego rozmiaru, harmonogram został umieszczony w lepszej wersji   
w dodatkowym załączniku (plik harmonogram.pdf)



# **Organizacja środowiska programistycznego projektu**

Na potrzeby projektu stworzono repozytorium i umieszczone je na portalu github.com pod adresem:

https://github.com/veiar/rso\_voting

Dla bezpieczeństwa, specjalista od repozytorium będzie przechowywał aktualną kopię zapasową projektu.

# **Bibliografia:**

1. Using Docker. Adrian Mouat. O’Reilly 2016.
2. Docker in Action. Jeff Nickoloff. Manning 2016.
3. <https://docs.docker.com/>