Politechnika Świętokrzyska

Wydział Elektrotechniki, Automatyki i Informatyki

Skład zespołu:

Kaja Wieczorek

Marcin Kroczak

gr. 1ID21B

Paxos wybór lidera

Projekt zespołowy z przedmiotu: Systemy odporne na błędy

na studiach stacjonarnych 2 stopnia

o kierunku Informatyka

Kielce, 25.01.2022

1. Użyte technologie

1.1 Technologie po stronie back-end

Przy implementacji logiki aplikacji posłużono się językiem C# wraz z frameworkiem .NET. Komunikację oparto na protokole TCP/IP i zrealizowano ją zgodnie z architekturą REST API. Celem zapewnienia komunikacji dwukierunkowej w czasie rzeczywistym pomiędzy elementami systemu dodatkowo wykorzystano technologię web socket. Wybrano do tego bibliotekę SignalR, która umożliwia asynchroniczne wysyłanie komunikatów z serwera do klienta.

1.2 Technologie po stronie front-end

Warstwę prezentacji opracowano przy użyciu technologii HTML, CSS, JavaScript i frameworka Angular. Framework zapewnił szkielet aplikacji i funkcje związane z komunikacją między elementami systemu. W arkuszu HTML umieszczono kontrolki i wyświetlono bieżący stan aplikacji.

2. Opis algorytmu

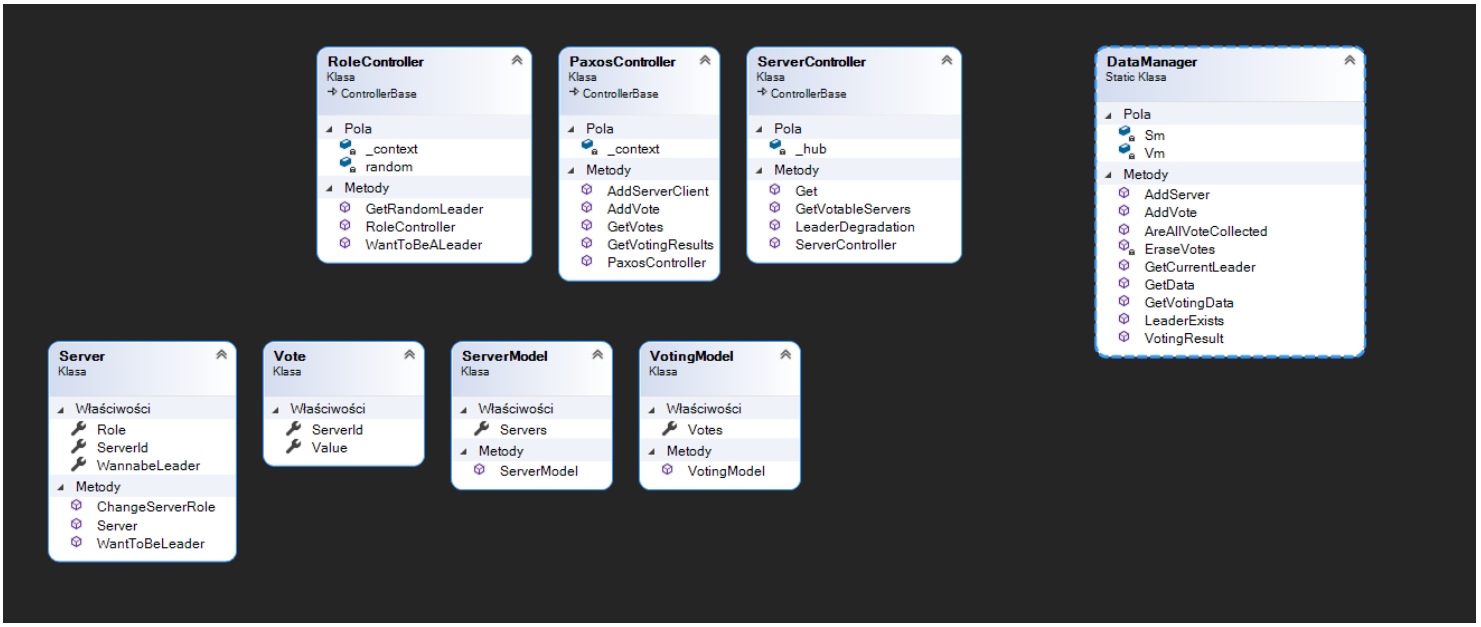
Paxos jest rodziną protokołów umożliwiających konsensus rozdzielczości w sieci omylnych węzłów. Algorytm znajduje zastosowanie w sieciach rozproszonych. Konsensus odnosi się tutaj do faktu, że różne węzły zgadzają się co do wyniku, a operacja jest trudna, gdy węzły lub ich środki komunikacji zawodzą. Jako kworum przyjęliśmy cały zbiór akceptorów. W algorytmie wyróżniamy rolę dla aktorów. Są to:

- Klient - Klient wysyła żądania do systemów rozproszonych i czeka na odpowiedź . Na przykład żądanie zapisu do rozproszonego systemu plików.
- Akceptant - Akceptory służą jako pamięć odporna na uszkodzenia. Akceptanci są pogrupowani w grupy zwane Kworami. Każda wiadomość wysłana do Akceptanta musi zostać wysłana do całego Kworum. Wiadomość otrzymana przez akceptanta, która nie została odebrana przez wszystkich innych akceptantów kworum, jest ignorowana.
- Proposer - Propozycja popycha żądanie klienta, ma na celu przekonanie Akceptujących do porozumienia i działa jako koordynator, aby iść naprzód, gdy pojawi się konflikt.
- Uczniowie - Uczniowie są wykorzystywani do replikacji. Po zaakceptowaniu żądania od Klienta przez Akceptantów, Uczeń może podjąć działanie (np. Wykonać żądanie i wysłać odpowiedź do klienta). Aby zwiększyć dostępność, możemy dodać uczniów.
- Lider - Paxos potrzebuje innego Sugerującego (zwanego przywódcą), aby iść naprzód. Kilka procesów może być uznanych za lidera, ale protokół

gwarantuje postęp tylko wtedy, gdy zostanie wybrany jeden z nich. Jeśli dwa procesy uważają, że są liderem, mogą zablokować protokół poprzez ciągłe wysyłanie sprzecznych propozycji. Ale w tym przypadku integralność danych jest nadal zachowana.

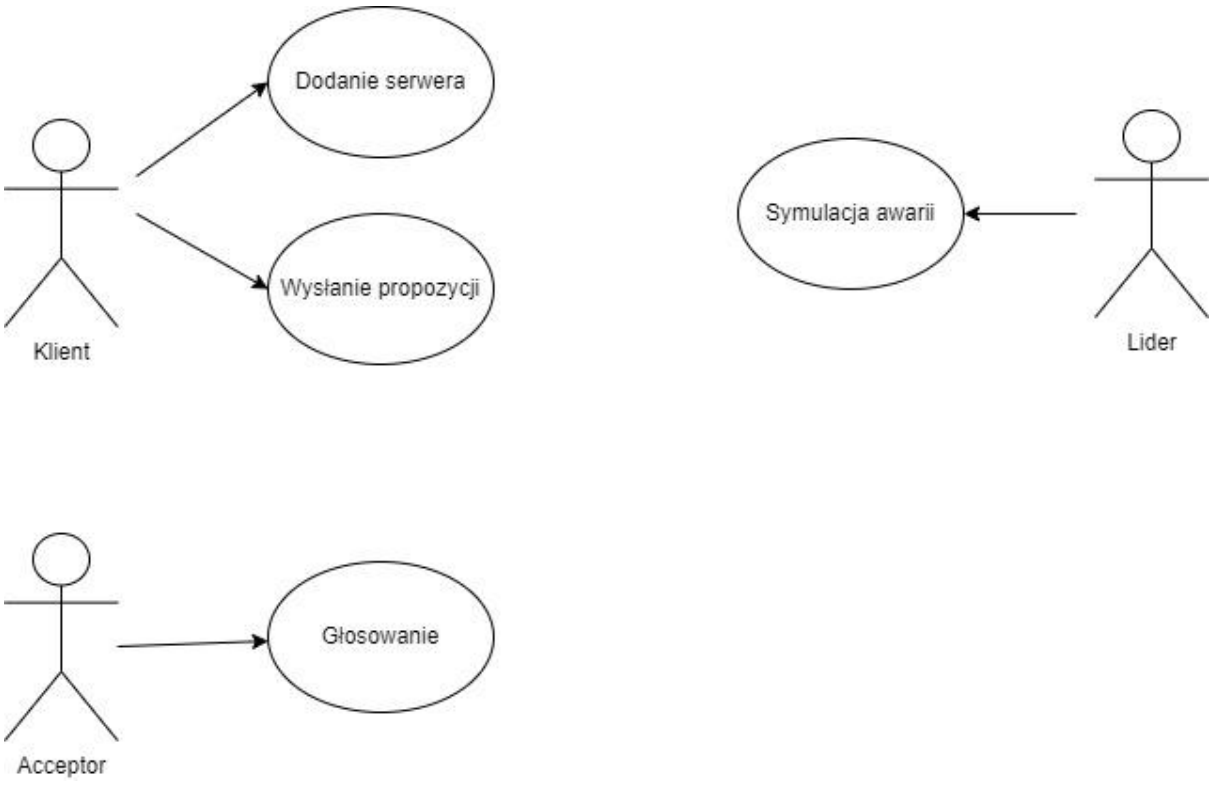
3. Diagram klas

W diagramie uwzględniono klasy pełniące rolę kontrolerów (RoleController, PaxosController, ServerController) w których odbierane są żądania od uczestników komunikacji związane z m.in dodawaniem serwera, głosowaniem, wyborem lidera. W Klasie DataManager zaimplementowana jest logika dla większości funkcji. Pozostałe klasy niżej są modelem serwera i głosu.



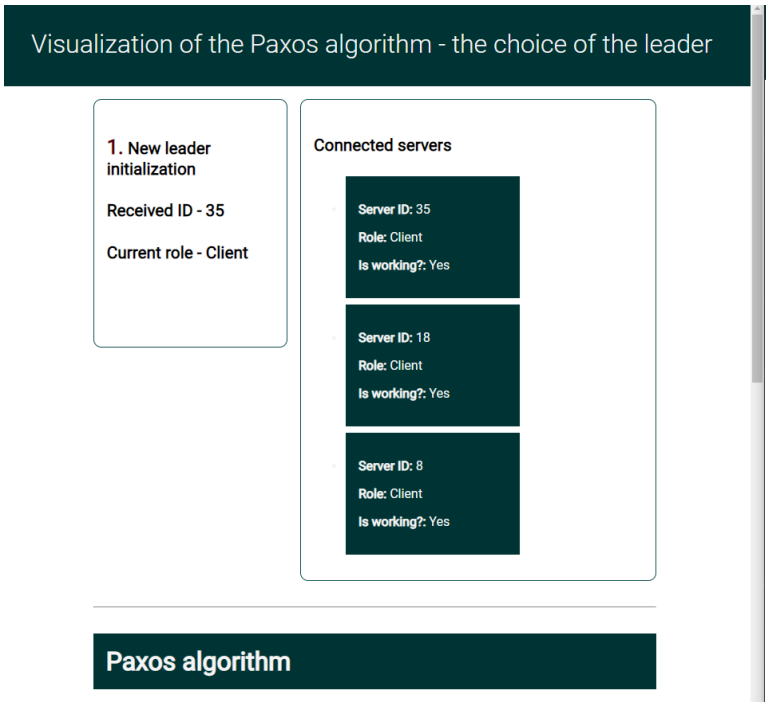
4. Diagram przypadków użycia

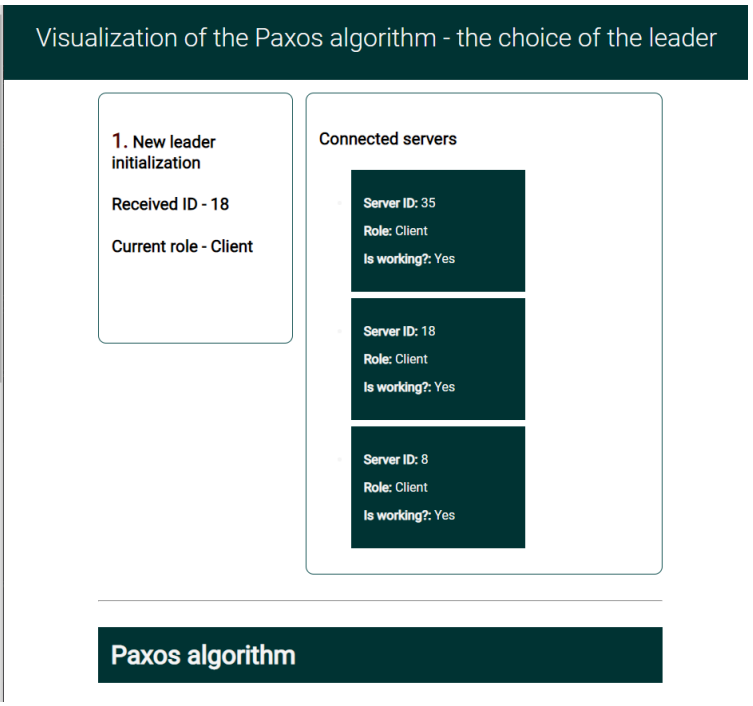
Użytkownik po uruchomieniu aplikacji automatycznie staje się klientem. Może on dodać jeden serwer do listy, aby zaznaczyć swoje uczestnictwo w głosowaniach. Klient po naciśnięciu przycisku może stać się Proposerem. Wówczas bierze udział w głosowaniu. Acceptorzy, czyli pozostali klienci mogą oddać głos na jednego z Proposerów. Po tym, gdy wszyscy oddadzą głos, wybierany jest nowy lider, który może symulować swoją awarię (na przykład wyłączenie okna przeglądarki) aby wznowić procedurę głosowania.



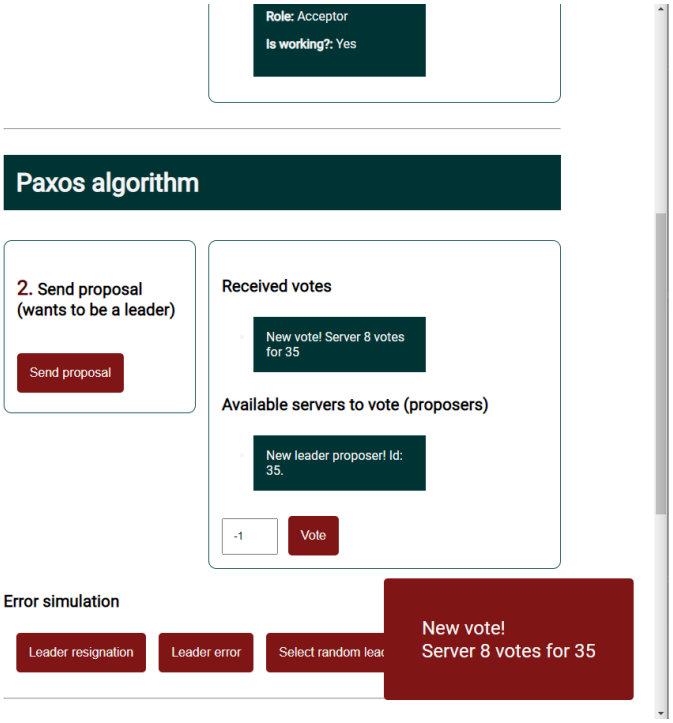
5. Przedstawienie działania aplikacji

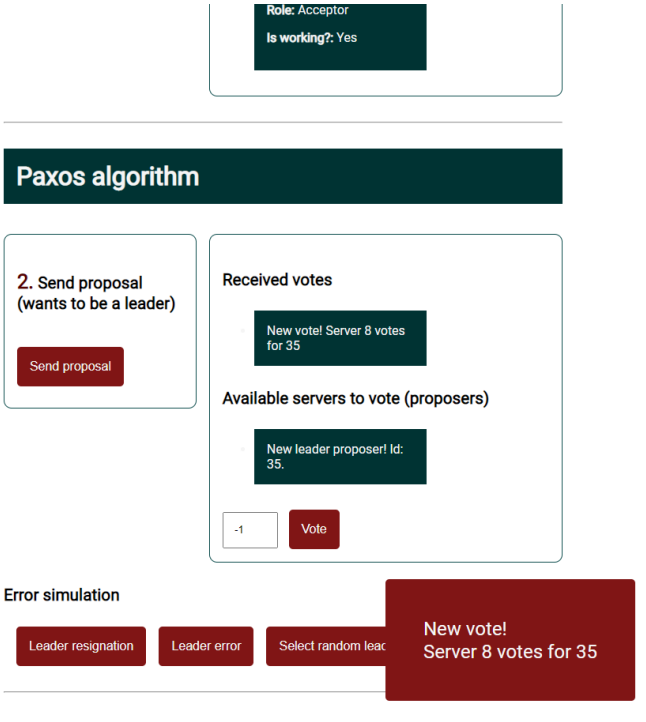
Uruchomienie aplikacji





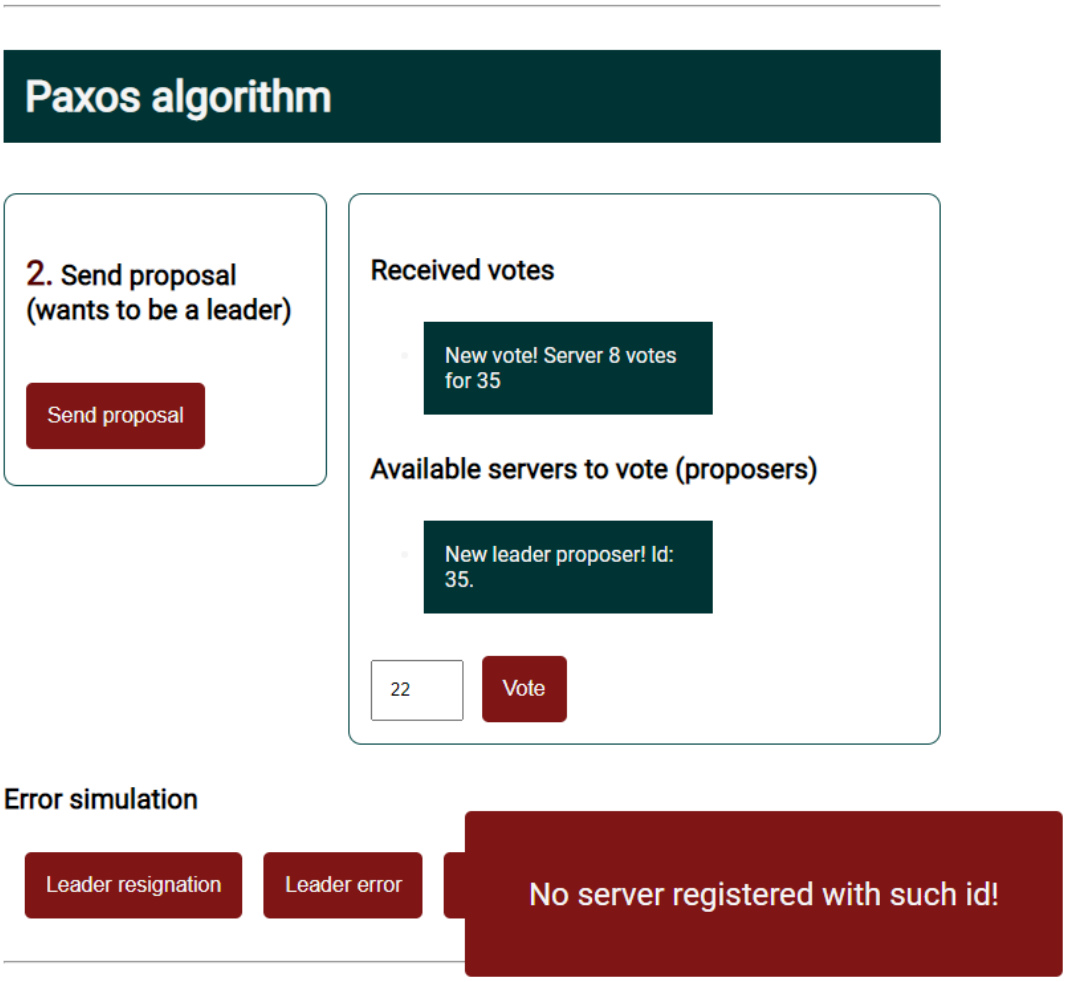
Głosowanie





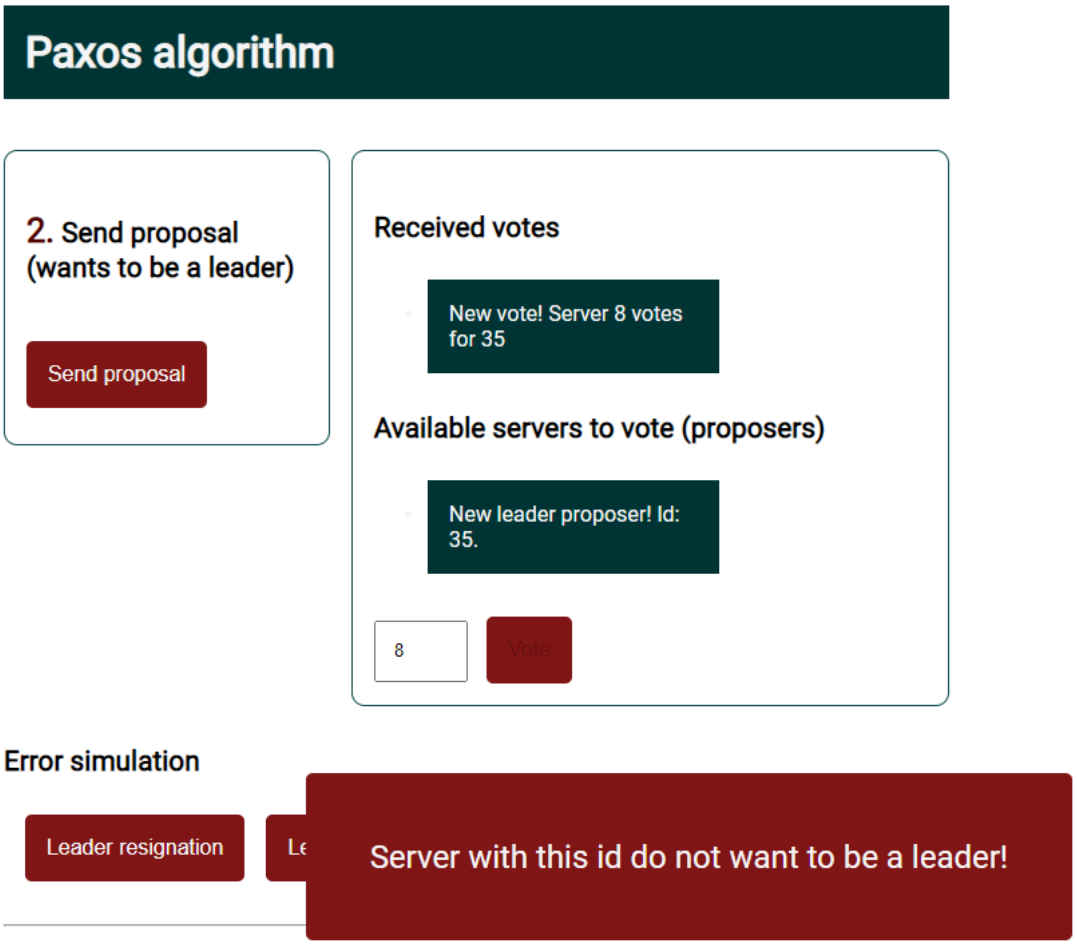
Błąd przy głosowaniu #1

Program zabezpieczony jest przed niepoprawnym oddawaniem głosów - nie można oddać głos na serwer, który nie istnieje.

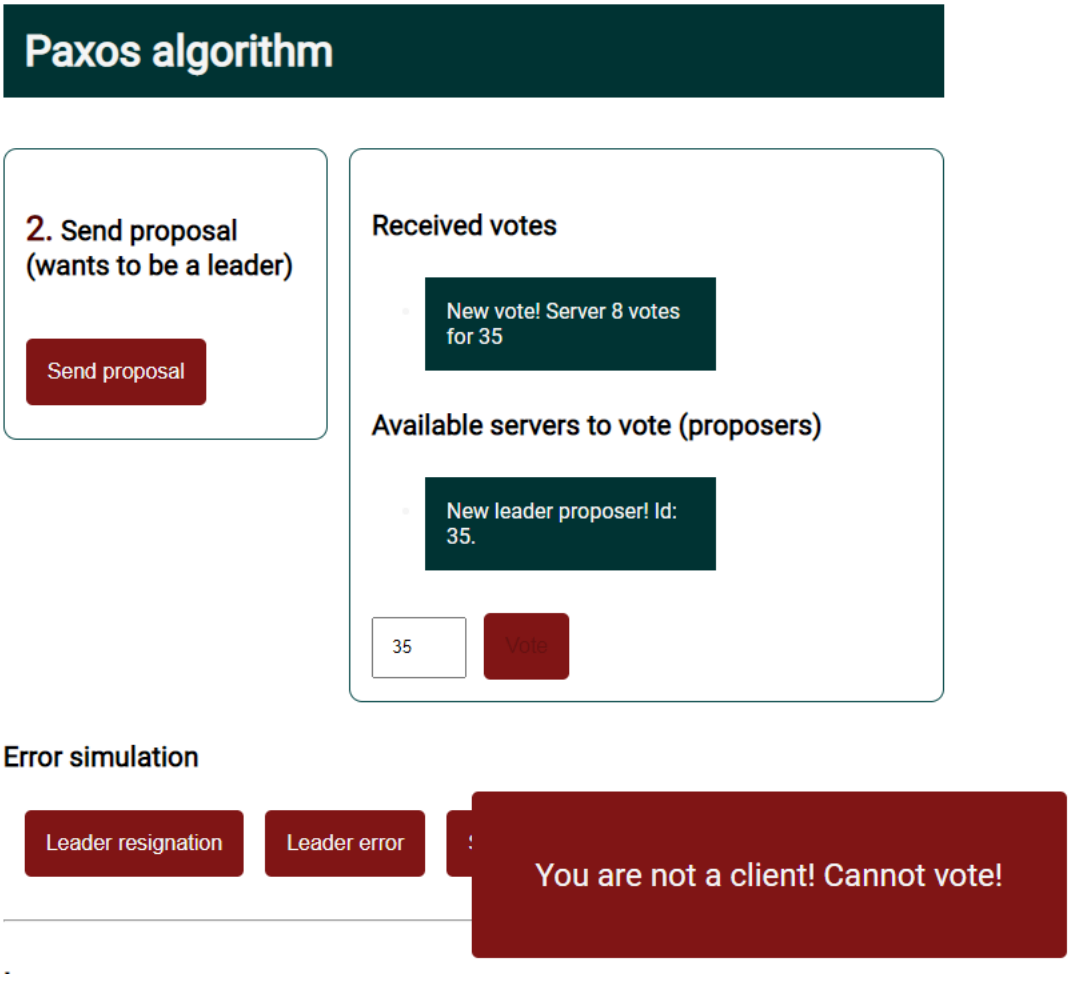


Błąd przy głosowaniu #2

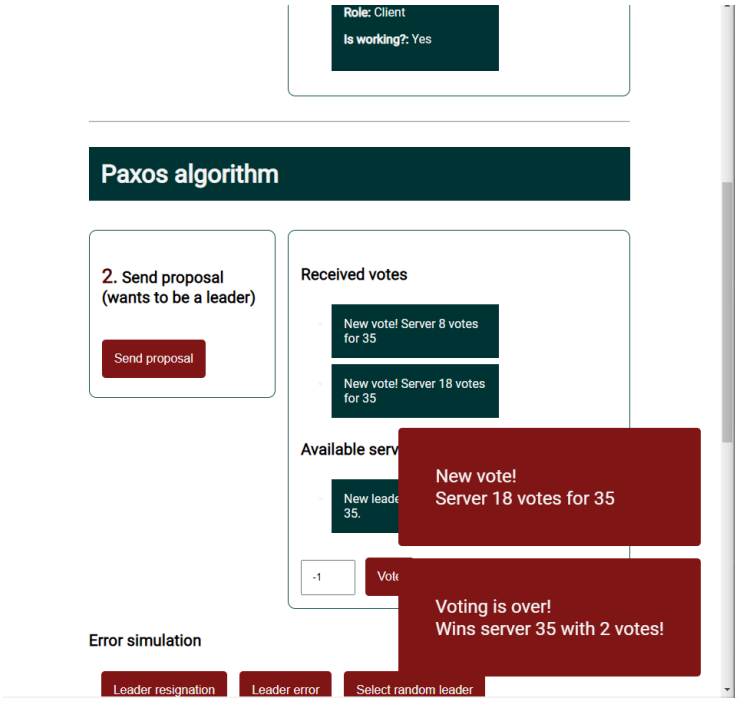
Do głosowania niezbędni są kandydaci na liderów. Jeśli któryś z serwerów wyrazi chęć zostania liderem, jego rola zmiania się na “proposer”. Nie można oddać głosu na serwer, który nie chce zostać liderem.

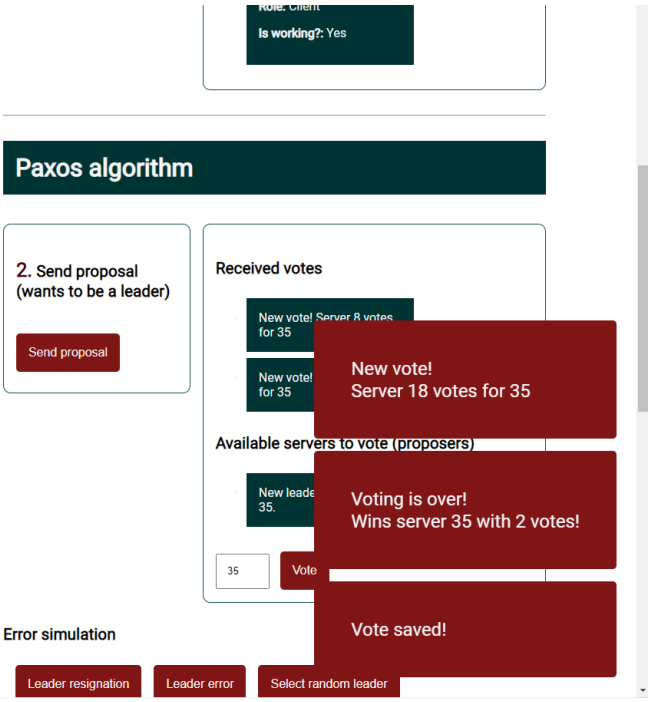


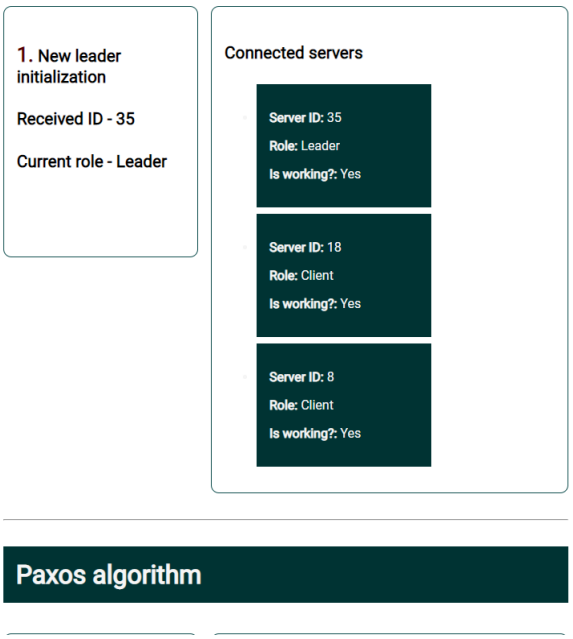
Oprócz dwóch powyższych błędów głosowania, są w systemie również inne zabezpieczenia z głosowaniem związane. Gdy serwer wyrazi chęć zostania liderem, nie może on oddać głosu. Nie można również drugi raz oddać głosu przez akceptora. Nie można również oddać głosu gdy nie ma aktywnego głosowania, tzn. gdy lider istnieje i nie ma powodów do głosowania. Ponownie głosowanie zostanie otwarte przy awarii obecnego lidera.

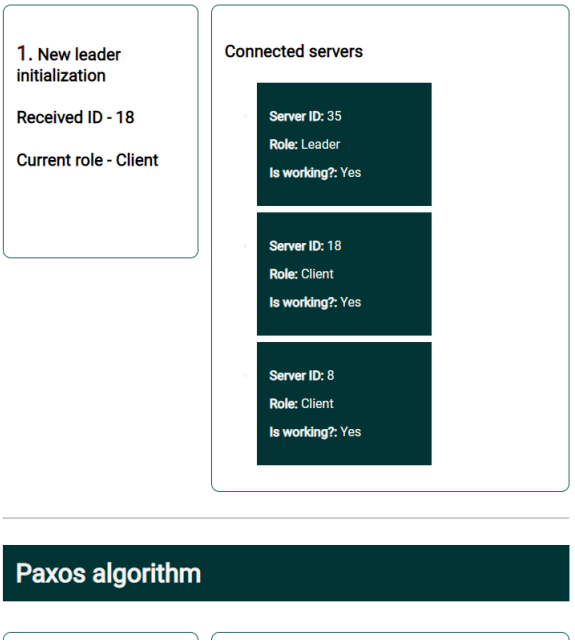


Po głosowaniu każdemu z serwerów wyświetla się komunikat o tym, który serwer wygrał i ile było oddanych na niego głosów. Głosy zostają wyczyszczone, głosowanie się zamyka i wygranemu nadawana jest rola lidera.



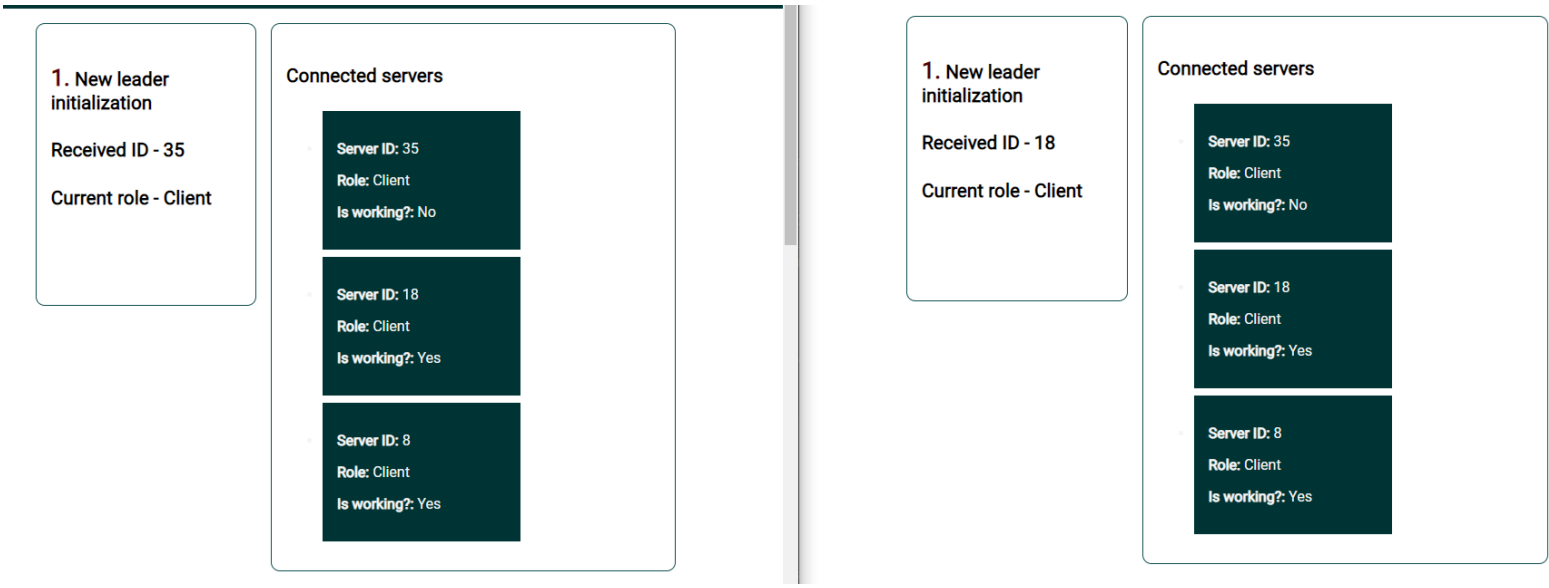






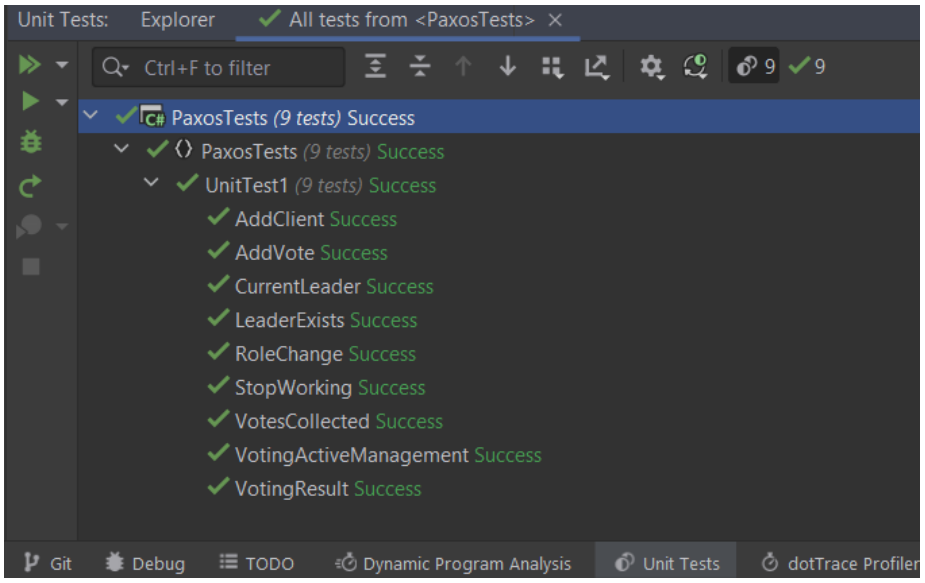
Awaria lidera

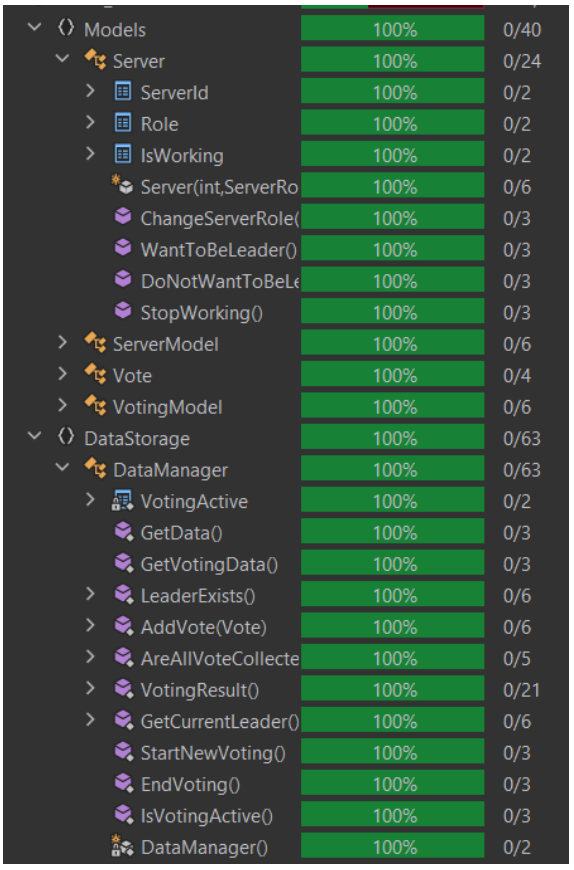
System ma wbudowane wprowadzenie awarii lidera. Zmienia ona status obecnego lidera na niedziałający i wysyła komunikat do wszystkich odbiorców o tym zdarzeniu. Rozpoczyna wtedy także nowe głosowanie.



6. Testy

Do aplikacji wykonane zostały również testy jednostkowe sprawdzające poprawność wykonywania się metod związanych z zarządzaniem danymi i modelami. Przetestowane zostały wszystkie metody modeli serwera i głosowania jak i metody dotyczące samego głosowania, zmiany ról, dodawania serwerów i inne.





7. Wnioski

Symulacja działania algorytmu Paxos przebiegła zgodnie z założeniami. Interfejs umożliwia przeprowadzenie wszystkich kluczowych dla algorytmu Paxos operacji. Dzięki “logom” łatwiejsze jest śledzenie przebiegu działania algorytmu. Do systemu możliwe jest wprowadzenie nowych wartości, jest to dodanie głosu, oraz dodanie nowego serwera do puli. Do systemu wprowadzone zostały wszystkie trzy rodzaje błędów, które zostały właściwie obsłużone. Są nimi: awaria bieżącego lidera, błędnie wprowadzona wartość w polu do głosowania, głosowanie na serwer który pełni inną rolę niż Proposer.

Spełniono podstawowe założenia algorytmu. Zasymulowano awarię i wznawianie pracy serwera, nie występują błędy bizantyjskie, każdy procesor może przesłać wiadomość do każdego innego asynchronicznie (poprzez technologię web socket), a wiadomości nie są przekłamywane w trakcie przesyłania.

Wykonane testy potwierdzają poprawne działanie algorytmów i metod, jak również mogą pomóc przy ewentualnym dalszym rozwijaniem programu.

8. Źródła

1. https://pl.frwiki.wiki/wiki/Paxos_%28informatique%29