

<p style="text-align: center;"><b>Politechnika Świętokrzyska w Kielcach</b>  <b>Wydział Elektrotechniki, Automatyki i Informatyki</b></p>	
<p style="text-align: center;"><b>Projekt systemu odporne na błędy</b></p>	
<p style="text-align: center;"><b>Paxos: Głosowanie</b></p>	
<p><b>Autorzy: Mateusz Mróz</b>  <b>Michał Młodawski</b>  <b>Kamil Pazera</b>  <b>Piotr Nowacki</b></p> <p>Grupa: <b>2ID21A</b></p>	<p>Data wykonania: <b>24.05.2021r.</b></p>

### 1) Cel projektu

Celem naszego projektu było stworzenie systemu głosowania z pomocą protokołu Paxos.

### 2) Krótki opis Paxosa

Paxos to protokół konsensusu w systemach rozproszonych. Konsensus to proces zgadzania się na jeden wynik wśród uczestników głosowania. Problemy pojawiają się gdy jeden lub więcej uczestników może doznać awarii.

Protokół Paxos składa się z następujących etapów:

Etap 1:

a) Prepare – Proposer tworzy wiadomość, którą nazywamy Prepare z przypisanym do niej numerem identyfikacyjnym  $n$ . Następnie wiadomość jest wysyłana do kworum akceptorów.

b) Promise – Akceptory czekają na wiadomość Prepare od Proposera. Jeśli otrzyma wiadomość to istnieją 2 warianty dalszego postępowania w zależności od wartości numeru  $n$ .

Jeśli  $n$  jest większe od poprzednich wiadomości, to Akceptor odsyła Proposerowi wiadomość nazywaną Promise, informującą o ignorowaniu wszystkich przyszłych wiadomości o numerze mniejszym od  $n$ . Jeśli Akceptor zaakceptował wcześniej jakąś wiadomość to dodatkowo odsyła poprzednią wiadomość wraz z jej numerem.

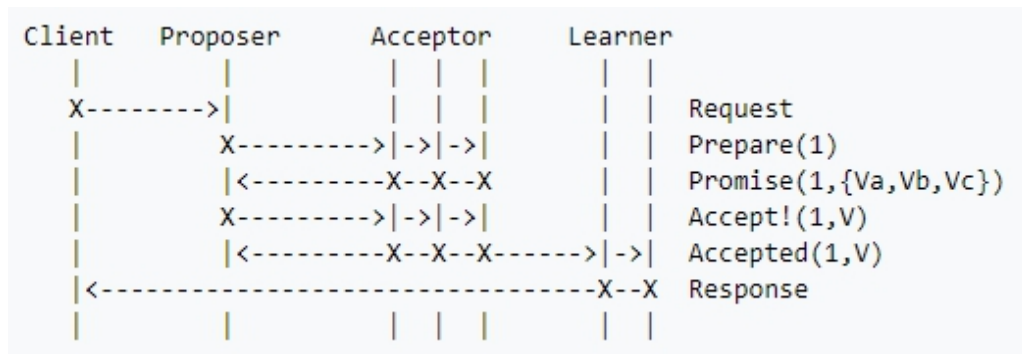
Etap 2:

a) Accept – Jeśli Proposer otrzyma większość Promise'ów z kworum to ustawia wartość  $v$  na propozycję. Jeśli któryś z akceptorów zaakceptował wcześniej jakąś propozycję to Proposer ustawia  $v$  względem najwyższej wartości propozycji Akceptorów  $z$ . Jeśli żaden z akceptorów nie akceptował wcześniej wiadomości to Proposer ustawia wartość na taką jaką chciał oryginalnie  $x$ .

Proposer wysyła wiadomość Accept  $(n, v)$  do kworum, wraz z wybraną wartością propozycji  $v$  i numerem propozycji  $n$ . Accept może przyjmować 2 formy:  $(n, v=z)$  lub  $(n, v=x)$ .

b) Accepted – Jeśli akceptor otrzyma wiadomość  $(n, v)$  od Proposera to akceptuje ją tylko jeśli obiecał

nie rozważać propozycji poniżej lub równej n.

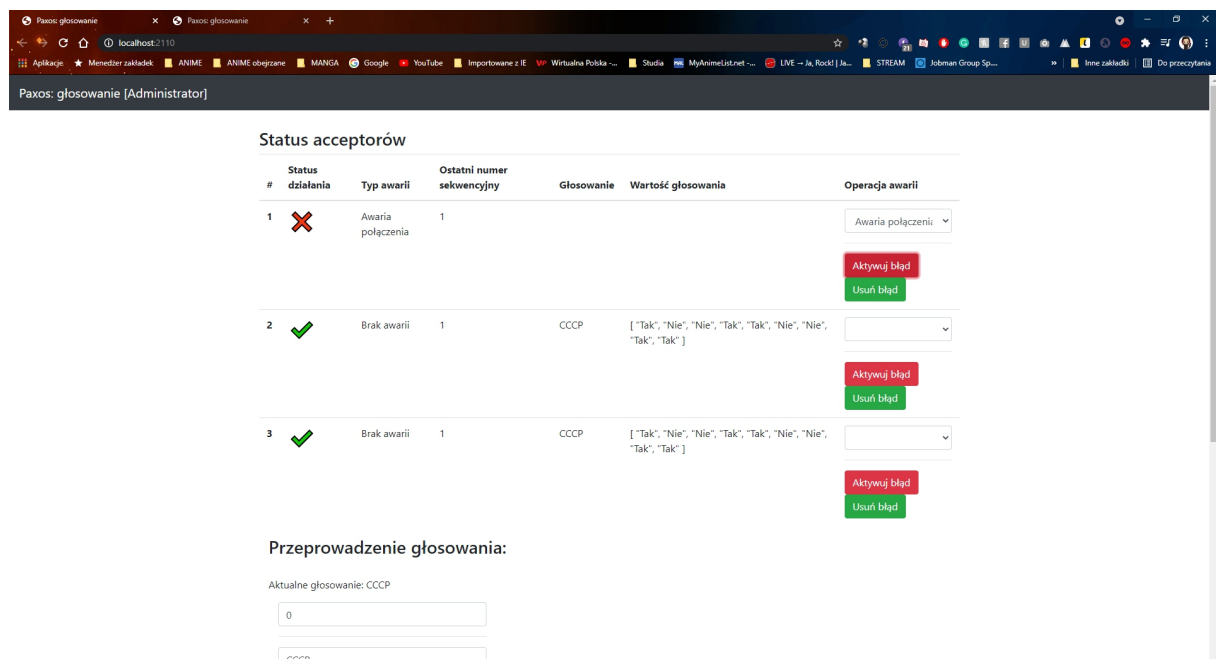


Rysunek 1

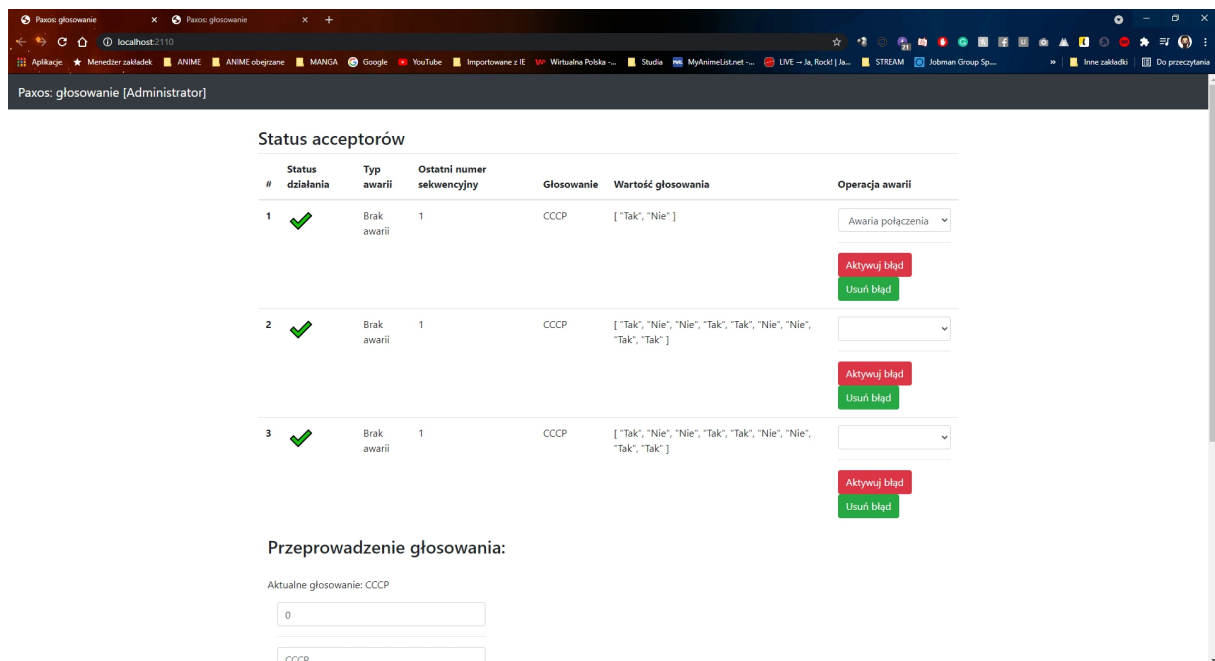
### 3) Rodzaje błędów i ich działanie

- Awaria połączenia - po aktywacji tego błędu następuje rozłączenie acceptora z serwerem (rysunek 2), w czasie gdy błąd ten jest aktywny acceptator nie jest odświeżany i zatrzymuje pobieranie nowego numeru sekwencyjnego, oraz nie pobiera nowych głosowań i odpowiedzi od klientów (rysunek 3).

- Szalony acceptator - po aktywacji tego błędu acceptator zwraca losowy numer sekwencyjny oraz nie można na nim ustawić nowego głosowania i nie odbiera odpowiedzi od klientów. Na rysunku 4 i 5 pokazane są zmiany w wartościach numeru sekwencyjnego.



Rysunek 2



Rysunek 3

#	Status działania	Typ awarii	Ostatni numer sekwencyjny	Głosowanie	Wartość głosowania	Operacja awarii
1	✗	Szalony akceptor	213			Szalony akceptor Aktywuj błąd Usuń błąd

Rysunek 4

1	✗	Szalony akceptor	995		Szalony akceptor	Aktywuj błąd Usuń błąd
---	---	------------------	-----	--	------------------	---------------------------

Rysunek 5

#### 4) Klient

Pierwszą z aplikacji wchodzącą w skład naszego projektu jest klient. Odpowiada ona za symulację 3 klientów podpiętych do głosowania, ich funkcjonalność oraz pełni funkcje Proposera. Sercem aplikacji jest klasa `ProposerPaxosLogicService.java` w której zaimplementowana jest logika Paxosa po stronie proposer'a. Najważniejsze metody to:

- startNewVotingProblem
- addNewVote
- sendProposeAndHandlePaxos
- sendProposeToAcceptor
- proposeRequestAccepted

startNewVotingProblem odpowiada za rozpoczęcie nowego głosowania. Jako argument przyjmuje nazwę nowego problemu i id klienta.

```
public void startNewVotingProblem(String problemName, Integer clientId) {
    ProposeRequestModel proposeRequestModel = new ProposeRequestModel(problemName,
seqNumber);
    seqNumber++;
    sendProposeAndHandlePaxos(clientId, AcceptorApi.ADD_NEW_PROBLEM, proposeRequestModel);
}
```

Na początku tworzony jest nowy `ProposeRequestModel` któremu podajemy nazwę problemu i aktualny numer sekwencyjny. Następnie zwiększamy numer sekwencyjny o 1 a następnie wywołujemy metodę `sendProposeAndHandlePaxos`.

addNewVote odpowiada za dodanie nowego głosu. Jako argument przyjmuje treść głosu i id klienta.

```
public void addNewVote(String vote, Integer clientId) {
    ProposeRequestModel proposeRequestModel = new ProposeRequestModel(vote,
    proposerAppState.getCurrentClient(clientId).getSequenceNumber());
    sendProposeAndHandlePaxos(clientId, AcceptorApi.ADD_NEW_VOTE, proposeRequestModel);
}
```

Na początku tworzony jest nowy `ProposeRequestModel` któremu przypisywany jest id klienta i treść głosu. Następnie wywołujemy metodę `sendProposeAndHandlePaxos`.

SendProposeAndHandlePaxos odpowiada za wysyłanie Proposów do Akceptorów i Requestów do klientów. Jako argument przyjmuje id klienta, url operacji i model requesta.

```
private void sendProposeAndHandlePaxos(Integer clientId, AcceptorApi operation,
ProposeRequestModel proposeRequestModel) {
    for (int i = 0; i < 3; i++) {
        sendProposeToAcceptor(i, operation, proposeRequestModel);
        proposeRequestsAccepted(i, clientId, operation, proposeRequestModel.getMessage());
    }
}
```

W pętli `for` trzykrotnie wywołujemy metody `sendProposeToAcceptor` i `proposeRequestAccepted`.

sendProposeToAcceptor odpowiada za wysyłanie propozycji na akceptory.

Jako argumenty przyjmuje id akceptora, url operacji, model requesta.

```
private boolean sendProposeToAcceptor(int acceptorId, AcceptorApi operation,
ProposeRequestModel proposeRequestModel) {
    String effectiveUrl = communicationService.processToEffectiveUrl(operation.getProposeUrl(),
acceptorId);
    return communicationService.sendProposeAndAwaitResponse(effectiveUrl,
proposeRequestModel);
}
```

Najpierw tworzymy url z użyciem urla operacji i id akceptora. Następnie wysyłamy requesta na utworzony url akceptora.

`proposeRequestAccepted` odpowiada za akceptowanie requestów. Jako argumenty przyjmuje id akceptora, id klienta, url operacji, model requesta i wiadomość.

```
private void proposeRequestsAccepted(int acceptorId, Integer clientId, AcceptorApi operation, String
acceptedMessage) {
    ClientModel currentClient = proposerAppState.getCurrentClient(clientId);
    currentClient.setSequenceNumber(seqNumber);
    String effectiveUrl = communicationService.processToEffectiveUrl(operation.getAcceptedUrl(),
acceptorId);
    AcceptedRequestModel acceptedRequestModel = new
AcceptedRequestModel(currentClient.getSequenceNumber(), acceptedMessage);
    communicationService.sendAccepted(effectiveUrl, acceptedRequestModel);
}
```

Najpierw pobierany jest klient. Następnie ustawiany mu jest numer sekwencyjny. Potem tworzony jest adres url na podstawie urla operacji i id akceptora. Na koniec za pomocą numeru sekwencyjnego klienta i wiadomości tworzony jest nowy request, a następnie jest on wysyłany na utworzony url.

## 5) Akceptor

Drugą aplikacją projektu jest Akceptor. Odpowiada ona za działanie akceptora. Sercem aplikacji jest klasa `AcceptorPaxosLogicService.java`. Jej najważniejsze metody to:

- getStateDto
- isSequenceCorrect
- acceptNewVotingSession
- acceptNewVote

`getStateDto` odpowiada za odpowiedź akceptorów. Jako argument przyjmuje id akceptora.

```
public AcceptorResponseModel getStateDto(Integer acceptorId) {
    AcceptorModel acceptor = acceptorAppState.getAcceptor(acceptorId);
    prevSeqNumber = BigInteger.valueOf(acceptor.getCurrentSequenceNumber());

    if (acceptor.getCurrentError() == 1) {
        seqNumberError.add(acceptor.getCurrentSequenceNumber());
        if (seqNumberError.size() > 1) {
            seqNumberError.remove(1);
        }
        return new AcceptorResponseModel(
            true,
            null,
            null,
            seqNumberError.get(0),
            acceptor.getCurrentError()
        );
    } else if (acceptor.getCurrentError() == 2) {
        Random rand = new Random();
        return new AcceptorResponseModel(
            true,
            null,
            null,
            rand.nextInt(999),
            acceptor.getCurrentError()
        );
    }
}
```

```

        acceptor.getCurrentError()
    );
}
VotingSession currentVotingSession = acceptorAppState.getCurrentVotingSession(acceptorId);
if (Objects.isNull(currentVotingSession)) {
    return new AcceptorResponseModel(
        true,
        null,
        null,
        prevSeqNumber.intValue(),
        acceptor.getCurrentError()
    );
}
return new AcceptorResponseModel(
    true,
    currentVotingSession.getCurrentProblem(),
    currentVotingSession.getVotes(),
    acceptor.getCurrentSequenceNumber(),
    acceptor.getCurrentError()
);
}

```

Na początku pobierany jest akceptor i aktualny numer sekwencyjny.

Jeśli błąd akceptora wynosi 1 (przerwane połączenie) to:

Aktualny numer sekwencyjny jest zapisany w liście. Następnie zwracana jest odpowiedź z numerem sekwencyjnym z listy.

Jeśli błąd akceptora wynosi 2 (szalony akceptor) to:

Tworzony jest generator liczb pseudolosowych. Następnie zwracana jest odpowiedź z losowym numerem sekwencyjnym.

Jeśli sesja głosowania nie istnieje to zwracana jest odpowiedź z numerem sekwencyjnym zapisanym na początku metody.

Jeśli żadna z powyższych sytuacji nie nastąpiła to zwracana jest odpowiedź zawierająca:

zaakceptowanie requesta, obecny problem, listę głosów, aktualny numer sekwencyjny oraz kod obecnego błędu.

isSequenceCorrect sprawdza czy numer sekwencyjny requesta zgadza się z akceptorem. Jako argumenty przyjmuje id akceptora oraz requesta.

```

public boolean isSequenceCorrect(Integer acceptorId, ProposeRequestModel requestModel) {
    prevSeqNumber =
    BigInteger.valueOf(acceptorAppState.getAcceptor(acceptorId).getCurrentSequenceNumber());
    return acceptorAppState.getAcceptor(acceptorId).getCurrentSequenceNumber() ==
    requestModel.getSequenceNumber();
}

```

Na początku pobierany jest numer sekwencyjny akceptora. Następnie porównywane są numer sekwencyjny akceptora oraz ten zapisany w requestcie. Jeśli się zgadzają zwracane jest true, a jeśli nie false.

acceptNewVotingSession odpowiada za akceptację nowego głosowania. Jako argumenty przyjmuje id akceptora i wiadomość.

```
public void acceptNewVotingSession(Integer acceptorId, AcceptedNotificationModel accepted) {
    AcceptorModel acceptor = acceptorAppState.getAcceptor(acceptorId);

    acceptor.getVotingSessions().add(new VotingSession(accepted.getAcceptedValue()));
    acceptor.setCurrentSequenceNumber(accepted.getNewSequenceId());
}
```

Najpierw pobierany jest akceptor. Następnie do listy zapisanych głosowań w akceptorze dodawane jest nowe głosowanie z wartościami z wiadomości, oraz ustawiany jest numer sekwencyjny akceptora.

acceptNewVote odpowiada za akceptację nowego głosu. Jako argumenty przyjmuje id akceptora i wiadomość.

```
public void acceptNewVote(Integer acceptorId, AcceptedNotificationModel accepted) {
    AcceptorModel acceptor = acceptorAppState.getAcceptor(acceptorId);
    if (shouldAcceptMsg(accepted, acceptor)) {
        acceptor.getCurrentVotingSession().getVotes().add(accepted.getAcceptedValue());
        acceptor.setCurrentSequenceNumber(accepted.getNewSequenceId());
    }
}
```

Najpierw pobierany jest akceptor. Następnie jeśli wiadomość powinna być zaakceptowana do listy głosów zapisanych w akceptorze dodawana jest wartość z wiadomości, oraz ustawiany jest numer sekwencyjny akceptora.

## 6) Działanie systemu

Po wejściu na stronę administratora (rysunek 6) o adresie (<http://localhost:2110/>) dostajemy możliwość stworzenia nowego głosowania, które jest ustawiane na acceptatorach i w którym mogą uczestniczyć klienci (rysunek 7). Strona ta pozwala również zarządzać błędami acceptatorów. Z listy można aktywować błąd szalonego acceptatora oraz awarie połączenia, a także usunąć błąd. Ostatnią rzeczą jest możliwość pobrania historii głosowania klientów. Po pobraniu w liście pokazują się opcje wybrane przez klientów na temat stworzonego głosowania oraz ostateczna wartość głosowania, która może mieć 3 wartości (Tak, Nie, Nie rozstrzygnięto). Wartość ostateczna jest określana na podstawie wartości większości głosów (rysunek 8).

Drugą stroną jest panel klienta (rysunek 9) o adresie (<http://localhost:2110/client.html>) dostajemy możliwość wpisania swojego identyfikatora oraz zagłosowania na dany temat opcjami tak i nie. Strona ta oferuje także podgląd na stan acceptatorów oraz historie głosowania i jej wynik w danym temacie głosowania (rysunek 10).

Paxos: głosowanie [Administrator]

Status acceptorów

#	Status działania	Typ awarii	Ostatni numer sekwencyjny	Głosowanie	Wartość głosowania	Operacja awarii
1	✓	Brak awarii	1			<div><div></div><div>Aktywuj błąd</div><div>Usuń błąd</div></div>
2	✓	Brak awarii	1			<div><div></div><div>Aktywuj błąd</div><div>Usuń błąd</div></div>
3	✓	Brak awarii	1			<div><div></div><div>Aktywuj błąd</div><div>Usuń błąd</div></div>

Przeprowadzenie głosowania:

Aktualne głosowanie:

Identyfikator klienta

Wartość głosowania

Rozpocznij proces głosowania

Rysunek 6

Paxos: głosowanie [Administrator]

Status acceptorów

#	Status działania	Typ awarii	Ostatni numer sekwencyjny	Głosowanie	Wartość głosowania	Operacja awarii
1	✓	Brak awarii	1	CCCP	[]	<div><div></div><div>Aktywuj błąd</div><div>Usuń błąd</div></div>
2	✓	Brak awarii	1	CCCP	[]	<div><div></div><div>Aktywuj błąd</div><div>Usuń błąd</div></div>
3	✓	Brak awarii	1	CCCP	[]	<div><div></div><div>Aktywuj błąd</div><div>Usuń błąd</div></div>

Przeprowadzenie głosowania:

Aktualne głosowanie: CCCP

0

CCCP

Rozpocznij proces głosowania

Rysunek 7

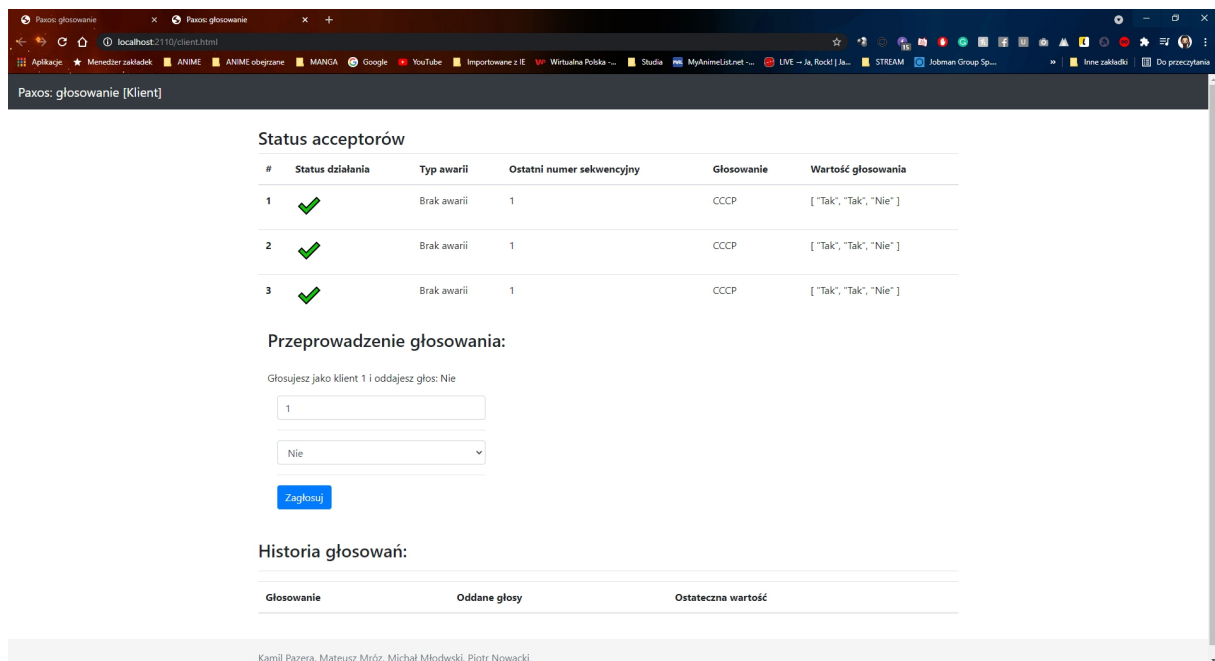
## Historia głosowań:

Pobierz wynik głosowania

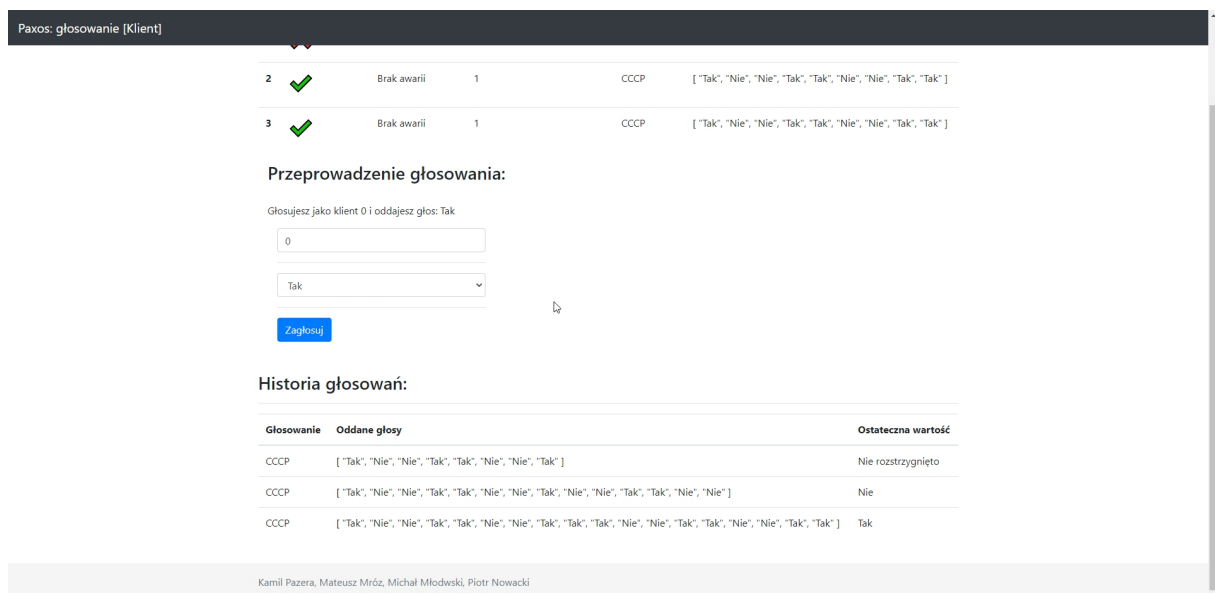
Głosowanie	Oddane głosy	Ostateczna wartość
CCCP	[ "Tak", "Nie", "Nie", "Tak", "Tak", "Nie", "Nie", "Tak" ]	Nie rozstrzygnięto
CCCP	[ "Tak", "Nie", "Nie", "Tak", "Tak", "Nie", "Nie", "Tak", "Nie", "Nie", "Tak", "Tak", "Nie", "Nie" ]	Nie
CCCP	[ "Tak", "Nie", "Nie", "Tak", "Tak", "Nie", "Nie", "Tak", "Tak", "Tak", "Nie", "Nie", "Tak", "Tak", "Nie", "Nie", "Tak", "Tak" ]	Tak

Rysunek 8





Rysunek 9



Rysunek 10

## 7) Bibilografia

- [https://en.wikipedia.org/wiki/Paxos\\_\(computer\\_science\)](https://en.wikipedia.org/wiki/Paxos_(computer_science)) (dostęp 24.05.2021r.)
- [https://www.wikiwand.com/en/Paxos\\_\(computer\\_science\)?fbclid=IwAR3m5QwgFyQDwrxB0blIEjmdNqaZiYebAW2SiKDwVvuKul06WE4Ph1a9qH4#/Roles](https://www.wikiwand.com/en/Paxos_(computer_science)?fbclid=IwAR3m5QwgFyQDwrxB0blIEjmdNqaZiYebAW2SiKDwVvuKul06WE4Ph1a9qH4#/Roles) (dostęp 24.05.2021r.)