Sławomir Klimowski

[Adres e-mail]

Streszczenie

[Przyciągnij uwagę czytelnika interesującym streszczeniem. Zazwyczaj jest to krótkie podsumowanie dokumentu.   
Aby dodać zawartość, wystarczy kliknąć tutaj i zacząć wpisywanie.]

Opracowanie serwisu WWW udostępniającego gry planszowe

Praca inżynierska

Spis treści

[1. Wstęp 2](#_Toc118744800)

[1.1. Cel pracy 2](#_Toc118744801)

[2. Funkcjonalność 2](#_Toc118744802)

[2.1. XMLHTTPRequest 2](#_Toc118744803)

[2.2. WebSocket 2](#_Toc118744804)

[2.3. Node.js 2](#_Toc118744805)

[3. Udostępnione API 3](#_Toc118744806)

[3.1. Opis 3](#_Toc118744807)

[3.2. Implementacja 6](#_Toc118744808)

[4. Opis aplikacji 10](#_Toc118744809)

[Diagram przypadków użycia 10](#_Toc118744810)

[Funkcje wewnątrz pokoju 12](#_Toc118744811)

[Technologie 12](#_Toc118744812)

[Diagram komponentów 12](#_Toc118744813)

[Diagram przepływu danych 14](#_Toc118744814)

[5. Prezentacja 14](#_Toc118744815)

[Rejestracja gry 14](#_Toc118744816)

[Otwarcie pokoju i połączenie użytkownika 15](#_Toc118744817)

[Tworzenie gier 15](#_Toc118744818)

[6. Wdrożenie 15](#_Toc118744819)

[W chmurze 15](#_Toc118744820)

[Docker + reverse-proxy 15](#_Toc118744821)

[7. Podsumowanie 15](#_Toc118744822)

[8. Wnioski 15](#_Toc118744823)

[9. Bibliografia? 15](#_Toc118744824)

# Wstęp

Gry planszowe są jedną z najstarszych form rozrywki znaną człowiekowi, a historia gier takich jak Szachy, Go czy Królewska gra z Ur liczy sobie tysiące lat. Pomimo swojego wieku ta forma rozrywki cieszy się dużą popularnością nie tylko wśród młodzieży, ale również wśród osób dorosłych. Na rynku pojawiają się nowe tytuły, wprowadzające nową fabułę, nie zmieniając opracowanego przez tysiące lat sposobu prezentacji stanu rozgrywki. Znacząca część gier planszowych cechuje się ustandaryzowaną obsługą i formą prezentacji. Gracze przemieszczają pionki po planszy w ściśle określony przez reguły gry sposób. Bieżący stan partii jest łatwo interpretowalny, dzięki czemu gracze nie tracą czasu na zrozumienie sytuacji i mogą się skupić na analizie i opracowaniu strategii wiodącej do zwycięstwa. Dodatkowo podobieństwa w prezentacji planszy pozwalają nowym graczom na szybkie zaznajomienie się z regułami nowego tytułu, korzystając z doświadczenia nabytego w innych tytułach. Porównując ze sobą kilka znanych tytułów tj. Szachy, Warcaby, Kółko i krzyżyk oraz Chińczyk jak również Królewską grę z Ur zauważyć można podobieństwa w aspekcie prezentacji. Najlepiej obrazuje to fakt, że plansza do gry w Szachy jest identyczna do gry w Warcaby, a jedyną widoczną różnicą jest kształt pionków. Większe różnice pojawiają się regułach gier, gdzie każdy tytuł niesie ze sobą nieraz całkiem odmienny zestaw zasad opisujących przebieg rozgrywki.

## Cel pracy

Duża popularność gier planszowych sprawia, że są one również jednym z najczęściej implementowanych rodzajów gier. Są one dobrze opisywalne za pomocą modelu maszyny stanowej, której zaprogramowanie nie stanowi dużej trudności nawet dla początkującego programisty. Ich oczywista schematyczność rodzi pytanie czy można ustandaryzować i ułatwić proces wytwórczy gier planszowych.

Celem niniejszej pracy jest opracowanie serwisu do udostępniania gier planszowych, który pozwoli na podział implementacji gry na dwie główne części, logikę i oprawę graficzną.

Logika przedstawia zbiór reguł, które opisują przebieg rozgrywki. Stanowi ona najbardziej zróżnicowaną pomiędzy różnymi tytułami część implementacji. Jest to również najbardziej formalna część, którą można opisać maszyną stanów. Nie narzuca żadnych znaczących obostrzeń na narzędzia programistyczne.

Druga część zawiera wszystkie aspekty związane z prezentacją danych użytkownikowi. Użytkownikiem jest osoba, która bierze czynny lub bierny udział w rozgrywce. Użytkownik o czynnym udziale nazywany jest graczem, natomiast użytkownik bierny przestawia obserwatora, który wyłącznie ogląda rozgrywaną partię. W skład oprawy graficznej wchodzi prezentacja planszy i pionków, obsługa kości do gry komunikacja z użytkownikiem. Przedstawiana zawartość jest silnie uzależniona od logiki gry, gdyż reguły gry opisują kształt planszy i pionków, liczbę graczy jak również jaką część planszy dany użytkownik może widzieć. Jest to część stawiająca największy rygor na narzędzia programistyczne. Wymagana jest prosta i zrozumiała prezentacja graficzna planszy pozwalająca na intuicyjny i komfortowy sposób wykonywania ruchu.

W ramach pracy przygotowany został serwis, w dalszej części pracy nazywany serwisem głównym, implementujący oprawę graficzną wraz z ustandaryzowanym interfejsem do udostępniania gier planszowych. Serwis pozwala na rejestrację dowolnej gry planszowej spełniającej wymogi interfejsu opisane w rozdziale 3.1. Do zadań serwisu należy zarządzanie listą gier jak również listą pokoi, zarządzaniem połączeniem z użytkownikami oraz ich rolą wewnątrz pokoju. Każdy pokój jest niezależną od innych instancją wybranej gry planszowej.

Dodatkowo powstał przykładowy serwis udostępniający logikę czterech gier planszowych, nazywany serwisem logiki w dalszej części pracy. Serwis ten implementuje interfejs spełniający wymogi serwisu głównego, dla czterech gier, które kolejno wykorzystują różne dostępne funkcjonalności.

Pierwszą zaimplementowaną grą jest Kółko i krzyżyk, która stanowi przykład minimalnej funkcjonalności wymaganej przez interfejs serwisu głównego. Następnym przykładem jest czteroosobowa gra połącz cztery, która ilustruje możliwość zmiennej liczby graczy. Następne na podstawie gry w statki pokazana została możliwość obsługi wielu plansz oraz prezentacja różnej zawartości tych plansz w zależności od roli użytkownika. Dodatkowo jest to pierwsza gra wprowadzająca wielokrotny ruch jednego gracza. Ostatnią opracowaną grą jest Chińczyk, która jako jedyna wykorzystuje planszę o nieregularnych kształtach oraz wprowadza losowość w postaci rzutu kością do gry.

# Funkcjonalność

## XMLHTTPRequest

XMLHTTPRequest (XHR) jest to interfejs programowania aplikacji (API) pozwalający na asynchroniczną komunikację między klientem a serwer. Większość współczesnych przeglądarek internetowych udostępnia go jako obiekt środowiska JavaScript. Interfejs ten implementuje obsługę wysyłania zapytań internetowych nawet po załadowaniu strony HTML. Pozwala również na odbieranie odpowiedzi w sposób asynchroniczny za pośrednictwem pętli zdarzeń języka JavaScript. XHR został zaprojektowany do przesyłania dokumentów XML, w późniejszych wersjach dodano obsługę danych w formacie JSON, jak również wiadomości w formie surowego tekstu. Korzystając z tego interfejsu możliwe jest dynamiczne modyfikowanie zawartości strony HTML. Popularne w rozwiązania SPA.

## WebSocket

WebSocket jest to protokół komunikacyjny, który zapewnia dwukierunkową komunikację wewnątrz jednego połączenia TCP. Protokół WebSocket został zaprojektowany do współpracy z protokołem http. Takie rozwiązanie gwarantuje pełną obsługę funkcjonalności oferowanej przez protokół http, m. in. Serwery proxy. Dodatkowo połączenie WebSocket nie wymaga osobnego portu, tylko wykorzystuje ten sam, z którego korzysta usługa http, standardowo 80 dla połączenia zwykłego oraz 443 w przypadku połączeń szyfrowanych. W celu zapewnienia kompatybilności dwóch protokołów, nawiązanie połączenia WebSocket odbywa się za pośrednictwem nagłówka http Upgrade. Połączenie WebSocket pozwala na komunikację między klientem i serwerem internetowym w znacznie zwięźlejszy sposób niż http, ponieważ nagłówek wysłany jest wyłącznie w czasie zestawiania połączenia. Dodatkowo połączenie jest dwukierunkowe i pozwala na wysyłanie w ustandaryzowany sposób informacji bezpośrednio z serwera do klienta, bez potrzeby ówczesnej inicjacji zapytania przez klienta.

## Node.js

Node.js jest to wieloplatformowe środowisko uruchomieniowe dla aplikacji w języku JavaScript do zastosowań serwerowych. Aplikacje uruchamiane są korzystając z silnika JavaScript Google V8. Node.js zaprojektowany zastał do umożliwienia tworzenia łatwo skalowalnych aplikacji internetowych stosując paradygmat „JavaScript everywhere” (JavaScript wszędzie). Pozwala na ujednolicenie procesu tworzenia aplikacji wykorzystując wyłącznie jeden język programowania zarówno po stronie klienta jak i serwera. Architektura Node.js oparta jest wokół pętli zdarzeń gwarantującej zdolność do równoległego wykonywania zadań asynchronicznych pomimo korzystania wyłącznie z jednego wątku procesora. Posiada bogate repozytorium pakietów, które rozszerzają podstawową funkcjonalność o m.in.:

* Tworzenie aplikacji w stylu REST
* Dynamiczne generacje stron HTML w oparciu o języki szablonów np. Embedded JavaScript (EJS)
* Obsługę połączeń stałych WebSocket

# Udostępnione API

## Opis

Serwis wystawia API za pomocą którego może komunikować się z serwerami logiki gry implementujące API zgodne z definicją openapi.

API opisuje minimalną liczbę stron? REST, które musi zawierać serwis.

Strony REST (w celu umożliwienia obsługi wielu gier planszowych przez jeden serwer ścieżka zawsze musi zaczynać się od nazwy identyfikującej grę oznaczoną jako {GameName}):

* /{GameName}/api  
  Pod tym adresem znajduję się deklaracja gry dostępna metodą http get jak również post. Jest to krótki opis jakie środowisko ma zostać przygotowane, przez serwis by prawidłowo udostępnić grę.   
  Deklaracja gry:
  + Nazwa gry
  + Krótki opis
  + Adres internetowy, pod którym serwis jest dostępny
  + Numer niestandardowego portu, na którym serwis nasłuchuje
  + Minimalna i maksymalna liczba graczy
  + Opis planszy(plansz) wykorzystywanej przez grę
    - Rodzaje plansz opisane w dalszej części
  + Lista ścieżek, pod którymi dostępne określone funkcje
    - API – adres, pod którym serwis udostępnia dokument opisu
    - NewGame – adres, pod którym serwis udostępnia polecenie rozpoczęcia nowej gry
    - NewRound – adres, pod którym serwis udostępnia polecenie rozpoczęcia newej rundy
    - Move – adres, pod którym serwis udostępnia polecenie wykonania ruchu
    - Update– adres, pod którym serwis udostępnia polecenie pobrania stanu gry przez użytkownika
    - Status– adres, pod którym serwis udostępnia polecenie pobrania stanu gry przez serwer
    - Open– adres, pod którym serwis udostępnia polecenie otwarcia nowego pokoju dla gry
    - Close– adres, pod którym serwis udostępnia polecenie zamknięcia otwartego pokoju gry
* /{GameName}/NewGame  
  Pod tym adresem serwis udostępnia polecenie rozpoczęcia nowej gry wewnątrz wybranego pokoju dostępne używając metody http post
  + Ciało zapytania:
    - room– nazwa identyfikująca instancję pokoju
    - players – liczba aktywnych graczy
  + Odpowiedź:
    - accepted – potwierdzenie zaakceptowania zapytania
    - message – opisowa odpowiedź na zapytanie
* /{GameName}/NewRound  
  Pod tym adresem serwis udostępnia polecenie rozpoczęcia nowej rundy wewnątrz wybranego pokoju dostępne używając metody http post
  + Ciało zapytania:
    - room – nazwa identyfikująca instancję pokoju
  + Odpowiedź:
    - accepted – potwierdzenie zaakceptowania zapytania
    - message – opisowa odpowiedź na zapytanie
* /{GameName}/Move  
  Pod tym adresem serwis udostępnia polecenie wykonania ruchu przez gracza wewnątrz wybranego pokoju dostępne używając metody http post
  + Ciało zapytania:
    - room – nazwa identyfikująca instancję pokoju
    - player – numer użytkownika wykonującego zapytanie
    - move – dane wykonanego ruchu  
      Serwis obsługuje trzy rodzaje ruchów:
      * choice – wybór jednego pola planszy  
        Przesłana wartość odpowiada nazwa wybranego pola
      * move – wybór pary dwóch pól planszy  
        Przesłana wartość odpowiada wzorcu:  
        „nazwa\_pierwszego\_pola|nazwa\_drugiego\_pola”
      * throw – rzut kością/kośćmi  
        Przesłana wartość to „throw”
    - board – numer planszy, na której dokonano ruchu
    - dices – lista ścianek rzuconych kości do gry
  + Odpowiedź:
    - accepted – potwierdzenie zaakceptowania zapytania
    - message – opisowa odpowiedź na zapytanie
* /{GameName}/Update  
  Pod tym adresem serwis udostępnia polecenie pobrania stanu gry przez użytkownika wewnątrz wybranego pokoju dostępne używając metody http post
  + Ciało zapytania:
    - room – nazwa identyfikująca instancję pokoju
    - player – numer użytkownika wykonującego zapytanie
  + Odpowiedź:
    - accepted – potwierdzenie zaakceptowania zapytania
    - message – opisowa odpowiedź na zapytanie
    - board/boards – lista stanów dla każdej planszy
    - score – punktacja wewnątrz pokoju
    - state – stan pokoju
      * gameActive – stan opisujący czy w pokoju rozgrywana jest partia gry
      * playerWon – wartość sygnalizująca, że bieżąca runda zakończyła się zwycięstwem gracza o podanym identyfikatorze
      * draw – flaga sygnalizująca, że bieżąca runda zakończyła się remisem
    - nextMove – rodzaj następnego ruchu, który gracz ma wykonać  
      Możliwe wartości: {„choice”, „move”, „throw”}
    - nextPlayer – identyfikator gracza, który ma wykonać następny ruch
* /{GameName}/Status  
  Pod tym adresem serwis udostępnia polecenie pobrania stanu gry przez serwis wewnątrz wybranego pokoju dostępne używając metody http post
  + Ciało zapytania:
    - room – nazwa identyfikująca instancję pokoju
  + Odpowiedź:
    - accepted – potwierdzenie zaakceptowania zapytania
    - message – opisowa odpowiedź na zapytanie
    - score – punktacja wewnątrz pokoju
    - state – stan pokoju
      * gameActive – stan opisujący czy w pokoju rozgrywana jest partia gry
      * playerWon – wartość sygnalizująca, że bieżąca runda zakończyła się zwycięstwem gracza o podanym identyfikatorze
      * draw – flaga sygnalizująca, że bieżąca runda zakończyła się remisem
* /{GameName}/Open  
  Pod tym adresem serwis udostępnia polecenie otwarcia nowej instancji pokoju dostępne używając metody http post
  + Ciało zapytania:
    - room – nazwa identyfikująca instancję pokoju
  + Odpowiedź:
    - accepted – potwierdzenie zaakceptowania zapytania
    - message – opisowa odpowiedź na zapytanie
* /{GameName}/Close  
  Pod tym adresem serwis udostępnia polecenie zamknięcie wybranej instancji pokoju dostępne używając metody http post
  + Ciało zapytania:
    - room – nazwa identyfikująca instancję pokoju
  + Odpowiedź:
    - accepted – potwierdzenie zaakceptowania zapytania
    - message – opisowa odpowiedź na zapytanie

Dostępne rodzaje plansz:

* Prosta  
  Plansza przedstawiająca standardową prostokątną planszę  
  Deklaracja:
  + Type=”simple” – identyfikator rodzaju planszy
  + rowCount – liczba wierszy w zakresie <1, 9>
  + rowLabels – typ znaczników dla wierszy
    - ‘d’ – cyfry
    - ‘l’ – litery
  + columnCount – liczba kolumn w zakresie <1, 9>
  + columnLabels – typ znaczników dla kolumn
    - ‘d’ – cyfry
    - ‘l’ – litery
  + tileWidth – szerokość pola planszy
  + tileHeight – wysokość pola planszy
  + textures – lista tekstur wykorzystywana w czasie gry
* Własna  
  Plansza o dowolnym kształcie i liczbie pól  
  Deklaracja:
  + Type=”custom” - identyfikator rodzaju planszy
  + tileWidth – szerokość pola planszy
  + tileHeight – wysokość pola planszy
  + width – szerokość obszaru planszy
  + height – wysokość obszaru planszy
  + background – rodzaj tła
    - color – tło wypełnione jednolitym kolorem
    - image – adres URI grafiki, która ma pełnić rolę wypełnienia
  + tiles – lista pól planszy  
    Każde pola opisuje trójka wartości:
    - x – pozycja horyzontalna lewego górnego narożnika pola
    - y – pozycja wertykalna lewego górnego narożnika pola
    - name – nazwa identyfikująca pole (nie musi być unikalna)
  + textures – lista tekstur wykorzystywana w czasie gry
* Losowa   
  Plansza przedstawiająca obszar rzutu kością(kośćmi) do gry  
  Deklaracja:
  + Type=”random” – identyfikator rodzaju planszy
  + Real – flaga czy kolejność ścianek ma odpowiadać fizycznej kostce sześciościennej
  + numberOfSides – liczba ścianek kostki
  + numberOfDices – liczba kostek rzucanych na raz
  + rotations – ile razy kostka ma się obrócić po rzucie

## Implementacja

Przykładowa implementacja serwera udostępniającego grę planszową na przykładzie gry Chińczyk.

Pierwszą część implementacji stanowi deklaracja gry zgodna z API:

* Podstawowe informacje:
  + name: 'ManDontGetAngry',
  + hostname: 'bgs-argen-game-server.herokuapp.com',
  + port: 443,
  + description: 'Cztero-osobowa gra planszowa w chińczyka',
  + maxNoPlayers: 4,
  + minNoPlayers: 2,
* Opis wykorzystywanych planszy:
  + Plansza własna, która przedstawia obszar gry:
    - type: 'custom',
    - tileWidth: 64,
    - tileHeight: 64,
    - width: 736,
    - height: 736,
    - background: { image: ‘https://i.pinimg.com/736x/0e/10/b5/0e10b5dee4f4d73f7facac1fac79a9c9.jpg’ } – tło będące grafiką pobraną z zewnętrznego serwera
    - textures: [
      * '/static/images/textures/Tile1.png',
      * '/static/images/textures/Tile1p1.png',
      * '/static/images/textures/Tile1p2.png',
      * '/static/images/textures/Tile1p3.png',
      * '/static/images/textures/Tile1p4.png',
      * '/static/images/textures/Tile1s.png',
      * '/static/images/textures/Tile1sp1.png',
      * '/static/images/textures/Tile1sp2.png',
      * '/static/images/textures/Tile1sp3.png',
      * '/static/images/textures/Tile1sp4.png',
      * '/static/images/textures/TileB.png',
      * '/static/images/textures/TileBp.png',
      * '/static/images/textures/TileG.png',
      * '/static/images/textures/TileGp.png',
      * '/static/images/textures/TileR.png',
      * '/static/images/textures/TileRp.png',
      * '/static/images/textures/TileY.png',
      * '/static/images/textures/TileYp.png',
    - ] – lista grafik wykorzystywana przez grę dostępnych na serwisie głównym
    - tiles: [
      * {x: 16, y: 272, name: 's0’},
      * {x: 80, y: 272, name: 'm1’},
      * {x: 144, y: 272, name: 's2’},
      * {x: 16, y: 656, name: 'b4’},
      * {x: 80, y: 656, name: 'b4’},
      * {x: 80, y: 336, name: 'f1’},
      * {x: 144, y: 336, name: 'f1’},
    - ] – siedem wybranych pól planszy z powtórzeniami nazwy
  + Plansza losowo do obsługi kostki sześciościennej
    - type: 'random',
    - real: true,
    - numberOfSides: 6,
    - numberOfDices: 1,
    - rotations: 10,
* Lista ścieżek pod którymi dostępne są odpowiednie funkcje
  + api: {
    - API: '/ManDontGetAngry/api',
    - NewGame: '/ManDontGetAngry/NewGame',
    - NewRound: '/ManDontGetAngry/NewRound',
    - Move: '/ManDontGetAngry/Move',
    - Update: '/ManDontGetAngry/Update',
    - Status: '/ManDontGetAngry/Status',
    - Open: '/ManDontGetAngry/Open',
    - Close: '/ManDontGetAngry/Close',
  + },

Kolejnym elementem jest opracowanie serwera https w stylu REST, który udostępni wymagane funkcjonalności. W tym celu można wykorzystać pakiet Express.js.

Propozycja implementacji ścieżki dla funkcji API

app.route('/:GameName/api')

.get((req, res) => {

if (!description[req.params.GameName]) {

res.json({ accepted: false, message: 'Gra nie istnieje!' });

return;

}

res.json({ accepted: true, api: description[req.params.GameName] });

})

.post((req, res) => {

if (!description[req.params.GameName]) {

res.json({ accepted: false, message: 'Gra nie istnieje!' });

return;

}

res.json({ accepted: true, api: description[req.params.GameName] });

});

Wszystkie elementy związane z logiką gry są w pełni uniezależnione od serwisu głównego. Dzięki czemu nie ma narzuconej kolejności graczy ani kolejności wykonywanych ruchów. W głównym serwisie nie ma również walidacji informacji przesyłanych przez użytkowników z wartościami oczekiwanymi.

Logikę gry można zaprojektować jako klasę lub pakiet, którego odpowiednie funkcje wywoływane są w zależności od ścieżki.

Przykład obsłużenia funkcji Update oraz odpowiadającej jej części logiki gry:

app.post('/:GameName/Update', (req, res) => {

const gameName = req.params.GameName;

const game = games[gameName];

if (!game) res.json({ accepted: false, message: "Game doesn't exists!" });

const rid = req.body.room;

const roomName = '#' + rid;

const room = rooms.get(roomName);

if (room) {

const player = req.body.player;

const data = **game.getUpdate(room, player);**

if (Object.keys(data).length) {

res.json({

...{ accepted: true, message: 'Request successful' },

...data,

});

} else res.json({ accepted: false, message: 'Request unsuccessful' });

} else res.json({ accepted: false, message: "Room doesn't exists!" });

});

function getUpdate(room, playerId) {

const board = room.board.concat(room.bases).concat(room.finishes);

saveZones.concat(startZones).forEach((zoneId) => {

board[zoneId] += 5;

});

for (var pid = 0; pid < 4; ++pid) {

for (var zid = 0; zid < 4; ++zid) {

board[4 \* pid + zid + 40 + 16] =

10 + 2 \* pid + (board[4 \* pid + zid + 40 + 16] > 0 ? 1 : 0);

}

}

return {

board: board,

score: room.score,

state: room.state,

nextMove: room.nextMove,

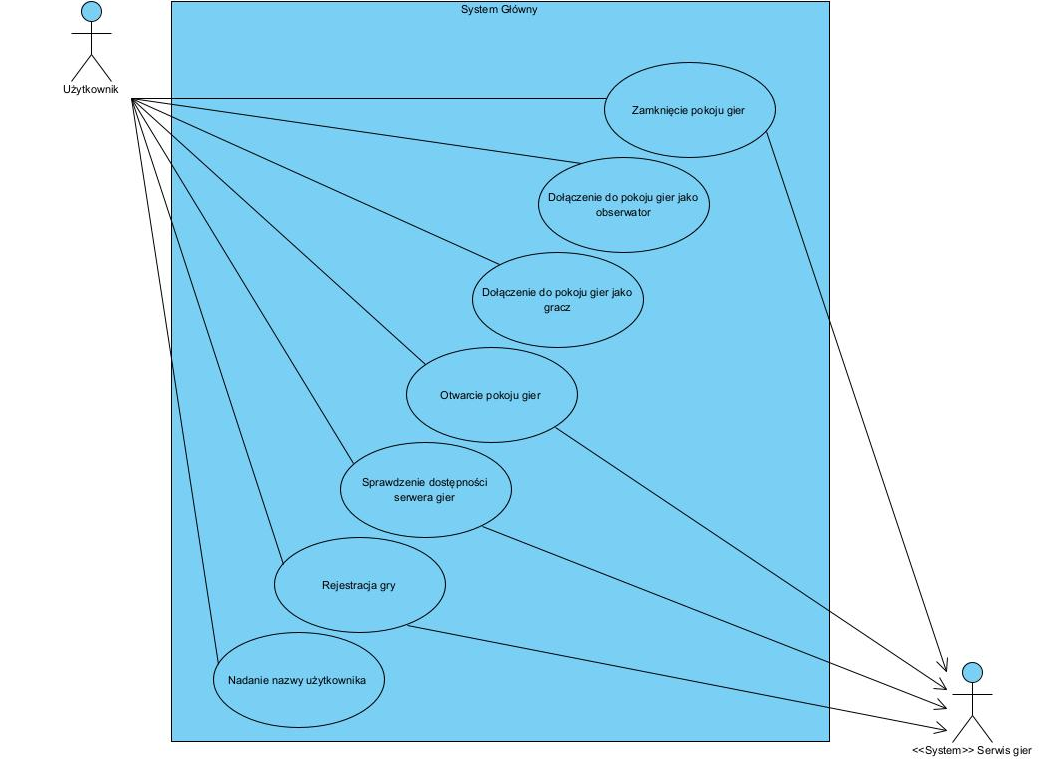
nextPlayer: room.player,

};

}

# Opis aplikacji

* 1. Funkcjonalność udostępniona przez serwis główny
     1. Funkcjonalność ogólna



Serwis główny udostępnia podstawowe funkcjonalności potrzebne do komfortowego korzystania z udostępnianych gier

Lista dostępnych akcji:

* Rejestracja gry planszowej

Podstawową funkcją, którą serwis musi udostępniać jest możliwość rejestracji dowolnej gry planszowej spełniającej wymogi interfejsu. Rejestracja odbywa się korzystając z jednego zapytania http. Serwis główny wysyła zapytanie o deklarację interfejsu na adres wskazany przez użytkownika, następnie, po sprawdzeniu poprawności odpowiedzi, zapisuje otrzymany dokument w pamięci. Dalsza komunikacja między serwisami opiera się o informacje zawarte wewnątrz deklaracji.

* Sprawdzenie dostępności serwisu logiki wybranej gry

Kolejną kluczową funkcją, jest możliwość sprawdzenia stanu połączenia między serwisami. Pozwala ona na sprawdzenie czy serwis gier jest wciąż osiągalny od strony serwisu głównego oraz czy wcześniej oferowana przez niego gra nadal jest dostępna. Odbywa się z użyciem funkcji Status opisanej przez interfejs.

* Otwarcie i zamknięcie pokoju gier

Każda gra możliwa do rejestracji musi posiadać możliwość obsługi co najmniej jednego pokoju gier. Serwis główny udostępnia użytkownikowi funkcje Open służącą do zainicjowania próby otwarcia nowego pokoju oraz Close, która wysyła prośbę o zamknięcie wybranego pokoju.

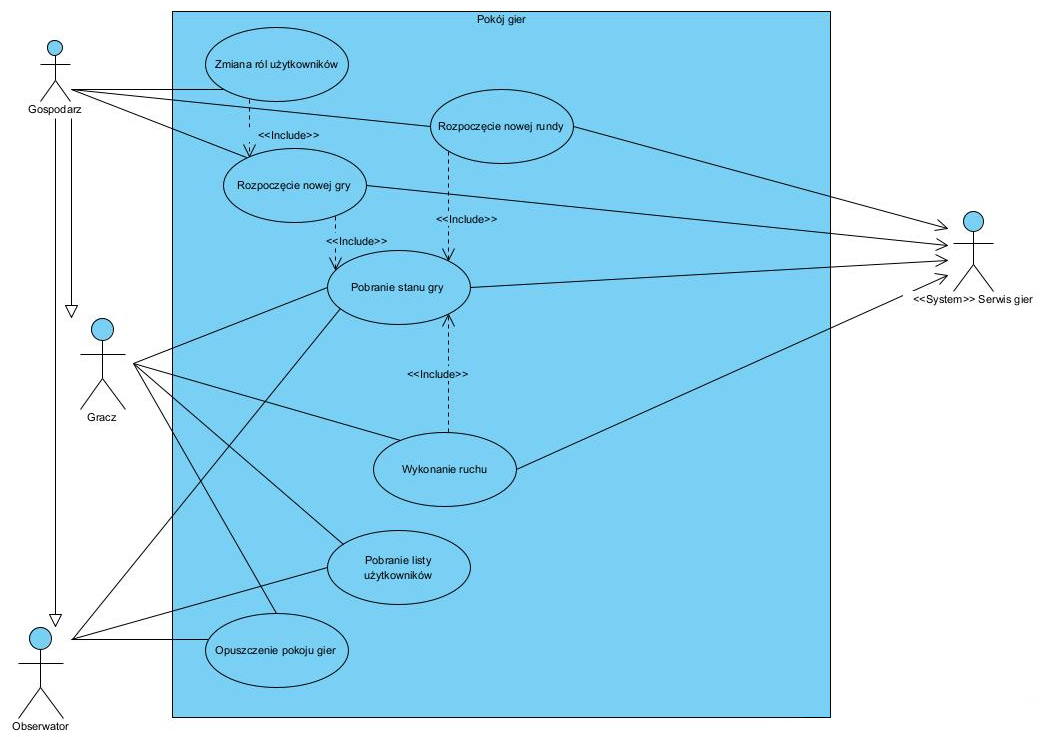
* Dołączenie użytkownika do pokoju gier z preferowaną rolą

Każdy użytkownik ma umożliwione dołączenie do dowolnego ówcześnie otwartego pokoju. Dodatkowo może sprecyzować jaką rolę wewnątrz danego pokoju chciałby objąć. Dołączenie jako gracz jest możliwe wyłącznie, gdy nie przekroczona została jeszcze maksymalna liczba graczy dla wybranego pokoju. Jeżeli dojdzie do sytuacji, że dołączenie jako gracz będzie nie dostępne, wtedy użytkownikowi automatycznie zostanie przydzielona rola obserwatora.

* Nadanie nazwy użytkownika

Ostatnią ogólną funkcjonalnością udostępnianą przez serwis główny jest możliwość wybrania własnej nazwy użytkownika. Nazwa ta jest wykorzystywana jako identyfikator na liście członków pokoju gier.

* + 1. Funkcjonalność wewnątrz pokoju



Osobną grupą funkcjonalności udostępnianej przez serwer główny są funkcje wewnątrz pokoju. Część funkcji dostępne są wyłącznie dla wybranej roli użytkownika. Każdy użytkownik pokoju gier ma przydzieloną jedną z dwóch ról gracz lub obserwator. Gracz to użytkownik, który bierze czynny udział w rozgrywce. Oznacza to, że ma prawo do wykonywania ruchów w czasie trwania partii. Obserwatorem nazywa się użytkownika, który bierze wyłącznie bierny udział w rozgrywce i nie może wykonywać żadnego ruchu. Dodatkowo jednemu użytkownikowi przedzielana jest specjalna dodatkowa rola gospodarza pokoju. Rola ta rozszerza możliwości członka pokoju o zmianę przydzielonych ról za wyjątkiem roli gospodarza, rozpoczęcie nowej partii oraz nowej rundy w czasie partii.

Opis poszczególnych funkcji:

* Funkcje dostępne wyłącznie dla gospodarza:
  + Zmiana ról użytkowników

Funkcja udostępniająca gospodarzowi zmianę przydzielonych użytkownikom ról, pozwalając przeprowadzić rotację graczy wewnątrz pokoju. Gracz nie musi opuścić pokoju, żeby zwolnić swoje miejsce dla innego użytkownika.

* + Rozpoczęcie nowej partii lub rundy

Funkcje udostępniają gospodarzowi możliwość rozpoczęcia nowej partii/gry oraz nowej rundy w czasie partii.

Partia również określana jako gra jest to zbiór jednej lub więcej rund gry planszowej wewnątrz pokoju. Każda partia ma własną punktację, która przechowuję rezultaty wszystkich rund, które odbyły w czasie jej trwania.

Runda jest to pojedyncza rozgrywka gry planszowej, która kończy się zwycięstwem jedno z graczy lub remisem.

* Funkcja dostępna wyłącznie dla gracza
  + Wykonanie ruchu

Funkcja pozwalająca na wysłanie propozycji ruchu przez gracza.

* Funkcje dostępne dla wszystkich ról
  + Pobranie stanu gry

Funkcja wywołująca uaktualnienie lokalnego stanu

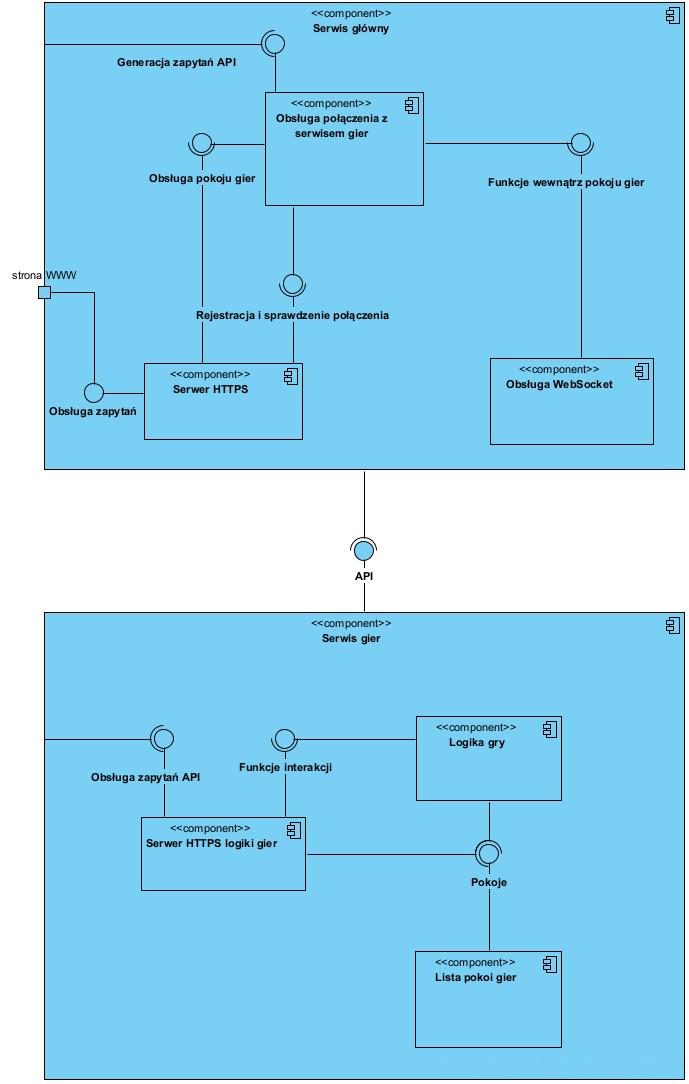
* + Pobranie listy użytkowników

Funkcja wywołująca uaktualnienie lokalnej listy członków pokoju z podziałem na role

* + Opuszczenie pokoju gier

Funkcja wywołująca zwolnienie ówcześnie zajmowanych ról przez użytkownika

* 1. Architektura rozwiązania



System udostępniający gry planszowe jest podzielony na dwa główne komponenty, serwis główny oraz serwis gier, połączone ze sobą interfejsem. Każdy z tych dwóch komponentów przedstawia osobną aplikację, która składa się pomniejszych komponentów.

Opis poszczególnych komponentów składowych:

* Serwis główny
  + Obsługa połączenia z serwisem gier

Komponent ten jest odpowiedzialny przygotowywanie zapytań oraz interpretację odpowiedzi zgodnie ze zdefiniowanym interfejsem.

* + Serwer HTTPS

Komponent ten reprezentuje serwer https, który udostępnia ogólną funkcjonalność serwisu

* + Obsługa WebSocket

Komponent ten reprezentuje serwer WebSocket, który udostępnia funkcjonalność wewnątrz pokoju gier

* Serwis gier
  + Serwer HTTPS

Komponent ten jest odpowiedzialny za udostępnianie funkcjonalności wymaganej przez interfejs

* + Logika gry

Komponent ten zawiera maszynę stanów, która opisuje udostępnianą grę planszową

* + Lista pokoi gier

Komponent ten reprezentuje rozwiązanie przechowywania instancji gry

* 1. Technologia
  2. Model przepływu informacji

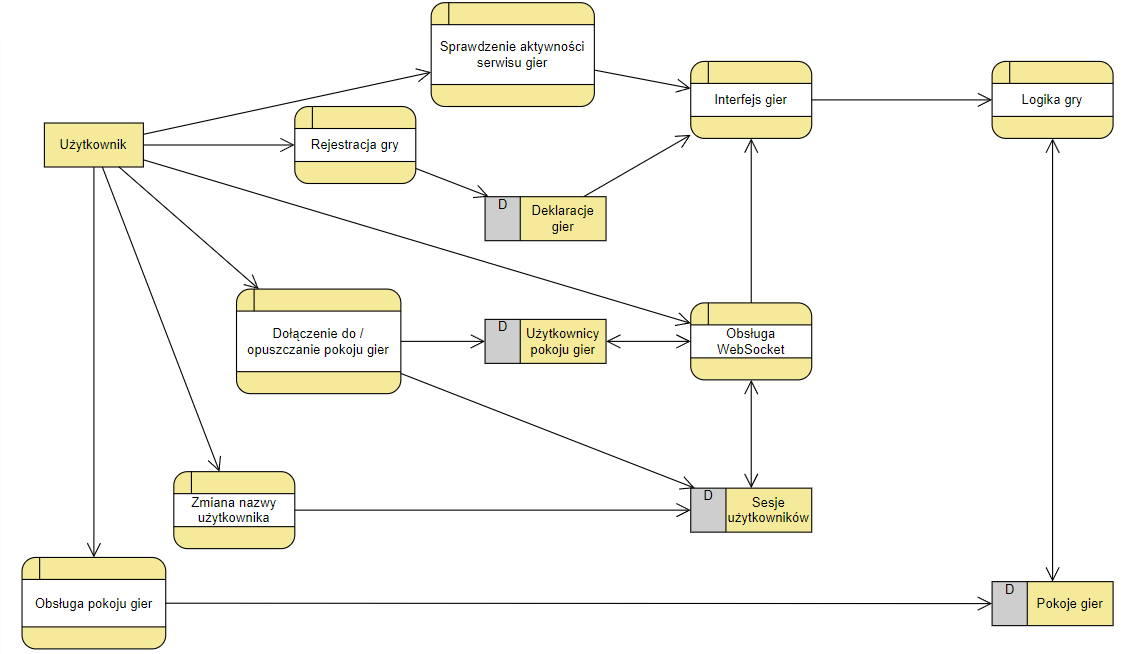


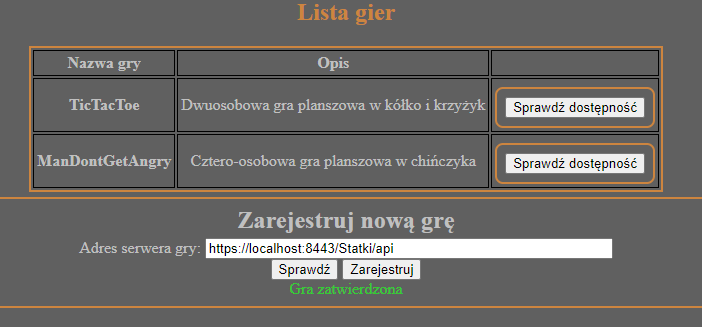
Diagram przepływu danych przedstawia diagram zerowy sytemu od udostępniania gier planszowych, gdzie serwis gier zostaw włączony do serwisu głównego. Tak prezentacja pozwala na lepsze zrozumienie przepływu informacji między funkcjami.

W implementacji sytemu granica między serwisami biegnie przez funkcję Interfejs gier oraz magazyn Deklaracje gier. Obydwa komponenty znajdują się w obu systemach

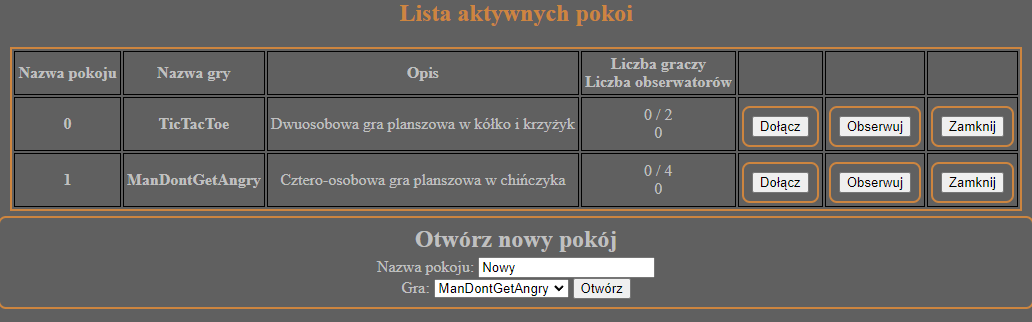
|  |  |  |
| --- | --- | --- |
| Komponent | Serwis główny | Serwis gier |
| Interfejs gier | Część przygotowująca zapytania zgodne z interfejsem | Część przygotowująca odpowiedzi zgodne z interfejsem |
| Deklaracje gier | Lista gier zarejestrowanych w serwisie | Lista gier udostępnianych w serwisie |

# Prezentacja

## Rejestracja gry



## Otwarcie pokoju i połączenie użytkownika



## Tworzenie gier

# Wdrożenie

## W chmurze

## Docker + reverse-proxy

# Podsumowanie

# Wnioski

# Bibliografia?