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 Do udostępniania gier planszowych

Praca inżynierska

Spis treści

[1. Wstęp 2](#_Toc119531277)

[1.1. Cel pracy 3](#_Toc119531278)

[2. Rozwiązania technologiczne 4](#_Toc119531279)

[2.1. XMLHTTPRequest (XHR) 4](#_Toc119531280)

[2.2. WebSocket 4](#_Toc119531281)

[2.3. Funkcjonalność udostępniona przez serwis główny 5](#_Toc119531282)

[2.3.1. Node.js 5](#_Toc119531283)

[2.3.2. Funkcjonalność ogólna 6](#_Toc119531284)

[2.3.3. Funkcjonalność wewnątrz pokoju 8](#_Toc119531285)

[3. Udostępnione API 10](#_Toc119531286)

[3.1. Opis 10](#_Toc119531287)

[3.1.1. Funkcje interfejsu 10](#_Toc119531288)

[3.1.2. Dostępne rodzaje plansz 13](#_Toc119531289)

[3.2. Przykładowa implementacja 14](#_Toc119531290)

[3.2.1. Deklaracja gry 14](#_Toc119531291)

[3.2.2. Serwer https 16](#_Toc119531292)

[4. Opis aplikacji 19](#_Toc119531293)

[4.1. Architektura rozwiązania 19](#_Toc119531294)

[4.2. Model przepływu informacji 21](#_Toc119531295)

[5. Prezentacja 22](#_Toc119531296)

[Rejestracja gry 22](#_Toc119531297)

[Otwarcie pokoju i połączenie użytkownika 22](#_Toc119531298)

[Chińczyk 22](#_Toc119531299)

[6. Wdrożenie 22](#_Toc119531300)

[W chmurze 22](#_Toc119531301)

[Docker + reverse-proxy 22](#_Toc119531302)

[7. Podsumowanie 22](#_Toc119531303)

[8. Wnioski 22](#_Toc119531304)

[9. Bibliografia? 22](#_Toc119531305)

# 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.

# Rozwiązania technologiczne

Serwis do udostępniania gier planszowych jest przykładem rozwiązania problemu komunikacji pomiędzy serwerem aplikacji, a serwerami zewnętrznymi w sposób niezauważalny przez użytkownika. Rozwiązanie zapewnia ustandaryzowany sposób wymiany informacji, który nie naraża klientów na potencjalne zagrożenia ze strony aplikacji zewnętrznych. Serwis główny łączy ze sobą dwa rozwiązania technologiczne XHR oraz WebSocket, udostępniające komunikację asynchroniczną wewnątrz protokołu http.

## XMLHTTPRequest (XHR)

XMLHTTPRequest 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 rozwiązaniach SPA.

Powyższe narzędzie znalazło zastosowanie w implementacji funkcjonalności odpowiedzialnej za nawiązywanie komunikacji pomiędzy serwerami. Proces ten jest zadaniem o bardzo niskiej częstotliwości występowania, natomiast może być czasochłonny w przypadku, gdy serwer zewnętrzny nie odpowiada. Czas oczekiwania na zakończenie tworzenia połączenia wynosi wtedy równowartość limitowi czasu oczekiwania na odpowiedź na żądanie http. Wykonanie tego polecenia w sposób synchroniczny doprowadziło by do zablokowania interfejsu użytkownikowi lub zerwania połączenia przez przeglądarkę z błędem 503 (Serwis niedostępy). Zastosowania narzędzia do komunikacji asynchronicznej gwarantuje użytkownikowi komfortowe korzystanie z serwisu.

## 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.

Cała funkcjonalność serwisu głównego wewnątrz pokoju gier opiera się połączenie WebSocket nawiązane automatycznie po dołączeniu. Dzięki takiemu rozwiązaniu każdy użytkownik otrzymuje powiadomienia o zmianach wywołanych przez innego członka. Pozwala to na znaczące ograniczenie ruchu w sieci, ponieważ nie ma potrzeby wysyłania periodycznych zapytań o stan pokoju, jak w przypadku rozwiązania opartego o technologię XHR.

## Funkcjonalność udostępniona przez serwis główny

Serwis główny, celu sprostania narzuconym na niego wymaganiom wynikającym z założeń projektu, doje dostęp użytkownikowi do funkcjonalności niezbędnych do korzystania z gier dostępnych w serwisie logiki. Serwis udostępnia również funkcje, których zadaniem jest zwiększenie komfortu w czasie używania aplikacji. Interakcja z użytkownikiem obsłużona jest przez środowisko Node.js rozszerzone o dodatkowe pakiety.

### 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.

Ważniejsze pakiety użyte w aplikacji:

* Express.js

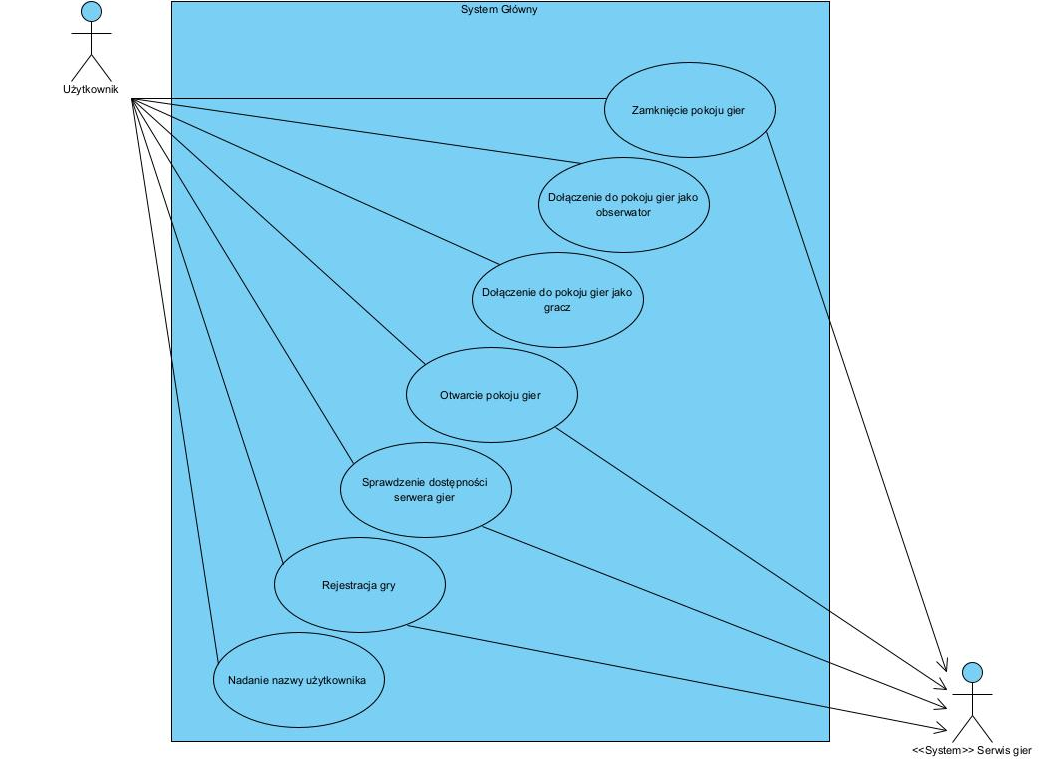
Odpowiedzialny za utworzenie aplikacji w stylu REST, pozwala na szybką implementację ścieżek. Do pakietu dołączona została również obsługa sesji i walidacja dokumentem openAPI oraz dynamiczna generacja stron HTML wykorzystując języki szablonów oparty na języku JavaScript .

* WebSocket

Pakiet implementuje obsługę protokołu WebSocket wykorzystywanego przez połączenia wewnątrz pokoi gier.

### Funkcjonalność ogólna

Pierwsza grupa udostępnianych funkcjonalności odpowiada za przygotowanie środowiska dla pokoi gier. Rysunek 1 przedstawia wszystkie funkcje dostępne dla użytkownika poza instancją pokoju gier. Rysunek sygnalizuje również, które funkcje wymagają aktywnego połącznie z serwisem logiki, a które wykonują wyłącznie operacje wewnętrzne.



Rysunek Funkcjonalność ogólna

* Sprawdzenie dostępności serwisu logiki wybranej gry

Fundamentalną funkcja, którą serwis musi udostępniać. Wywołuje ona próbę nawiązania lub sprawdzenie stanu połączenia między serwisami. Pozwala to stwierdzić czy serwis logiki jest osiągalny od strony serwisu głównego oraz czy wcześniej oferowana przez niego gra nadal jest dostępna. Tę funkcję użytkownik może wykonać na dwa sposoby. Po pierwsze może wysłać zapytanie na dowolny adres internetowy w procesie rejestracji gry, jak również na adres zapisany w pamięci serwisu dla zarejestrowanej gry. Żądania http wysyłane są asynchronicznie za pomocą narzędzia XHR, co umożliwia sprawdzenie wielu gier naraz. Wszystkie pozostałe funkcjonalności wykonywane są synchronicznie.

* Rejestracja gry planszowej

Druga podstawowa funkcja, która odpowiada za rejestrację dowolnej gry planszowej spełniającej wymogi interfejsu. Wysłane zostaje przez serwis główny żądanie http o deklarację interfejsu gry na adres wskazany przez użytkownika. Jeżeli walidacja odpowiedzi zakończy się sukcesem otrzymany dokument zapisywany jest w pamięci. Dalsza komunikacja między serwisami opiera się o informacje zawarte wewnątrz deklaracji.

* Otwarcie i zamknięcie pokoju gier

Każda gra możliwa do rejestracji musi posiadać możliwość obsługi co najmniej jednej instancji pokoju gier. Serwis główny udostępnia użytkownikowi funkcje Open służącą do zainicjowania próby otwarcia nowego pokoju oraz Close, która wysyła prośbę o zamknięcie wybranego pokoju. Ich dokładne działanie jest uzależnione od implementacji serwisu logiki. Serwis główny narzuca wyłącznie ograniczenie od dołu na dokładną liczbę obsługiwanych pokoi na grę. Nie jest wymaga również walidacja nieaktywności pokoju w czasie jego zamknięcia, wszyscy aktywni użytkownicy zostaną automatycznie przeniesieni na zewnątrz.

* Dołączenie użytkownika do pokoju gier z preferowaną rolą

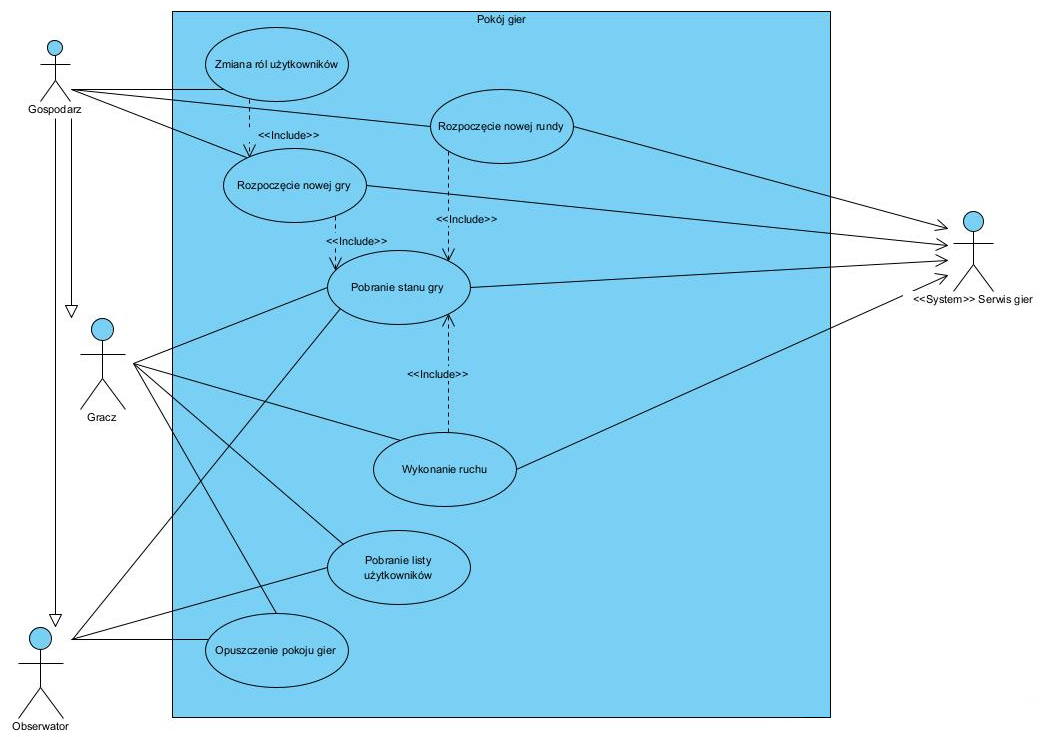
Każdy użytkownik ma umożliwione dołączenie do dowolnego ówcześnie otwartego pokoju. Dodatkowo może sprecyzować jaką rolę wewnątrz danego pokoju chciałby objąć. Dołączenie jako gracz jest możliwe wyłącznie, gdy nie przekroczona została jeszcze maksymalna liczba graczy dla wybranego pokoju. Jeżeli dojdzie 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. Nadawanie unikalnej nazwy nie jest wymagane, służy ona wyłącznie użytkownikom.

### Funkcjonalność wewnątrz pokoju

Osobną grupą funkcjonalności 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 z wyjątkiem roli gospodarza, rozpoczęcie nowej partii oraz nowej rundy w czasie partii.



Rysunek Funkcjonalności wewnątrz pokoju

Na Rysunku 2 przedstawione zostały funkcje z podziałem na role członków. Poniżej znajdują szczegółowe opisy każdej funkcjonalności.

* Funkcje dostępne wyłącznie dla gospodarza:
  + Zmiana ról użytkowników

Odpowiada za udostępnienie możliwości zmiany przydzielonych użytkownikom ról, pozwalając przeprowadzić rotację graczy wewnątrz pokoju. Jest alternatywną metodą rozpoczęcia nowej gry, gdzie w ciele zapytania wysłana zostaje nowa lista graczy. Dzięki temu 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 jednego z graczy lub remisem.

* Funkcja dostępna wyłącznie dla gracza
  + Wykonanie ruchu

Funkcja pozwalająca na wysłanie propozycji ruchu. Każdy gracz w swojej turze zostanie poproszony o wykonanie ruchu lub serii ruchów według oczekiwań serwisu logiki. W przypadku, gdy ruch będzie nie prawidłowy, gracz zostanie o tym poinformowany i poproszony o wykonanie innego.

* Funkcje dostępne dla wszystkich ról
  + Pobranie stanu gry

Podstawowa funkcja wywołująca uaktualnienie lokalnego stanu oraz z odświeżeniem interfejsu użytkownika. Uruchamiana automatycznie przy każdej zmianie stanu pokoju. Może również zostać wykonana na żądanie.

* + Pobranie listy użytkowników

Funkcja wywołująca uaktualnienie lokalnej listy członków pokoju z podziałem na role. Uruchamiana automatycznie podczas dołączenia lub opuszczenia pokoju przez dowolnego użytkownika.

* + Opuszczenie pokoju gier

Automatyczna funkcja wywołująca zwolnienie ówcześnie zajmowanych ról przez użytkownika.

# Udostępnione API

Interfejs programistyczny aplikacji (ang. Application Programming Interface, API) jest to zbiór zasad opisujący sposób komunikacji pomiędzy wieloma programami komputerowymi. Umożliwia ustandaryzowanie procesu wymiany informacji zapewniając hermetyzację poszczególnych aplikacji. Dobrze zaprojektowane API pozwala na korzystanie w pełni z udostępnianej funkcjonalności, nie narażając poszczególne podsystemy na szkodliwe interakcję pomiędzy nimi. Elementy systemu traktują się jak czarne skrzynki.

Kluczowym elementem API jest specyfikacja będąca dokumentem opisującym sposób korzystania z interfejsu oraz jakie funkcjonalności aplikacja musi implementować, by móc wystawić określony interfejs.

Serwis do udostępniania gier planszowych jest systemem składającym się z dwóch serwisów internetowych, które komunikują się za pomocą interfejsu internetowego. Serwer logiki gier jest fragmentem implementującym API, a serwer główny jest aplikacją korzystającą z wystawionego interfejsu.

## Opis

Specyfikacja interfejsu internetowego została opracowana zgodnie z projektem OpenAPI (dawniej Swagger). Specyfikacja OpenAPI opisuje format plików, które pozwalają na projektowanie serwisów internetowych w stylu RESTful. Bogata gama narzędzi obsługujących specyfikację pozwala na automatyzację procesu testowania oraz walidacji interfejsu.

Dokument API opisuje wszystkie funkcje niezbędne do prawidłowej komunikacji między serwisami. Składa się z listy ścieżek REST-owych, pod którymi znajdują się odpowiednie funkcjonalności implementowane przez logikę gier. Opracowany dokument zakłada ściśle określony format zasobów internetowych. Wynika to z limitacji specyfikacji OpenAPI, która wymaga podania wszystkich punktów końcowych serwisu, natomiast interfejs pozwala na dowolny format ścieżek. Proponowana forma zasobów umożliwia obsługę wielu gier planszowych przez jeden serwis logiki. Każdy punkt końcowy zaczyna się od nazwy gry do której się odnosi. Takie rozwiązanie pozwala na pominięcie jednego parametru zapytania http, odpowiedzialnego na identyfikację gry.

### Funkcje interfejsu

W ramach interfejsu zostało opisane osiem funkcji: Functions, NewGame, NewRound, Move, Update, Status, Open, Close. Dla każdej z nich zdefiniowano przyjmowane parametry oraz format odpowiedzi. Wszystkie funkcjonalności dostępne są metodą http post. Wyjątek stanowi polecenie Functions, które jest obsłużone również metodą get, co pozwala użytkownikom na łatwe podejrzenie zawartości deklaracji.

* API

Bezparametrowa funkcja, w której odpowiedzi znajduje się deklaracja gry, zawierająca opis środowiska niezbędnego do prawidłowego obsłużenia gry przez serwis główny. Poprawna deklaracja składa się z następujących pól:

* + Nazwa gry
  + Krótki opis
  + Adres internetowy, pod którym serwis jest dostępny
  + Numer niestandardowego portu protokołu TCP, na którym serwis nasłuchuje
  + Minimalna i maksymalna liczba graczy
  + Opis planszy(plansz) wykorzystywanej przez grę
    - Rodzaje plansz opisane w Rozdziale 3.1.2.
  + Lista ścieżek, pod którymi dostępne określone funkcje

Informacje zawarte w deklaracji pozwalają na udostępnienie jej przez serwisy pośredniczące lub zapisanie w pamięci serwisu głównego, bez utraty możliwości kontynuacji komunikacji.

* + Odpowiedź:
    - Flaga określająca stan zaakceptowania zapytania
    - Deklaracja gry
* NewGame

Funkcja udostępnia polecenie rozpoczęcia nowej gry wewnątrz wybranego pokoju. Odpowiada za przywrócenie stanu początkowego plansz oraz wyzerowanie punktacji.

* + Ciało zapytania:
    - Nazwa identyfikująca instancję pokoju
    - Liczba aktywnych graczy
  + Odpowiedź:
    - Flaga określająca stan zaakceptowania zapytania
    - Opisowa odpowiedź na zapytanie
* NewRound

Pod tym adresem znajduje się polecenie rozpoczęcia nowej rundy wewnątrz wybranego pokoju. Akcja resetuje wyłącznie plansze.

* + Ciało zapytania:
    - Nazwa identyfikująca instancję pokoju
  + Odpowiedź:
    - Flaga określająca stan zaakceptowania zapytania
    - Opisowa odpowiedź na zapytanie
* Move

Funkcja odpowiadająca za przekazanie propozycji ruchu przez gracza wewnątrz wybranego pokoju. Na parametry narzucona jest wyłącznie walidacja formy, sprawdzenia poprawności ruchu należy do części implementującej logikę gry.

* + Ciało zapytania:
    - Nazwa identyfikująca instancję pokoju
    - Numer użytkownika wykonującego zapytanie
    - Dane proponowanego ruchu  
      Interfejs obsługuje trzy rodzaje ruchów:
      * choice – wybór jednego pola planszy  
        Przesłana wartość odpowiada nazwie wybranego pola.
      * move – wybór pary dwóch pól planszy  
        Pierwsze pole określa, z którego miejsca zaczyna się ruch. Natomiast drugie jest miejscem docelowym. Nazwy pól są złączone ze sobą znakiem pionowej kreski („|”).
      * throw – rzut kością/kośćmi  
        Ten rodzaj ruchu przyjmuje wyłącznie wartość „throw”, która wyraża chęć wykonania rzutu kośćmi przez gracza.
    - Numer planszy, na której dokonano ruchu
    - Lista ścianek rzuconych kości do gry
  + Odpowiedź:
    - Flaga określająca stan zaakceptowania zapytania
    - Opisowa odpowiedź na zapytanie
* Update

Polecenie pobrania stanu bieżącej gry przez członka wybranego pokoju. W odpowiedzi znajdują się wszystkie niezbędne informacje do odświeżenia interfejsu użytkownika.

* + Ciało zapytania:
    - Nazwa identyfikująca instancję pokoju
    - Numer użytkownika wykonującego zapytanie
  + Odpowiedź:
    - Flaga określająca stan zaakceptowania zapytania
    - Opisowa odpowiedź na zapytanie
    - Lista stanów pól dla każdej planszy  
      Wartość stanu jest tożsama z indeksem wyświetlanej w miejscu pola planszy tekstury z listy podanej w deklaracji gry.
    - Punktacja wewnątrz pokoju dla bieżącej partii
    - Stan pokoju
      * Flaga opisująca czy w pokoju rozgrywana jest runda gry
      * Wartość sygnalizująca, że bieżąca runda zakończyła się zwycięstwem gracza o podanym identyfikatorze
      * Flaga sygnalizująca, że bieżąca runda zakończyła się remisem
    - Rodzaj następnego ruchu oczekiwanego od gracza  
      Interfejs pozwala wyłącznie na wartości: „choice”, „move”, „throw”.
    - Identyfikator gracza, który ma wykonać następny ruch
* Status

Funkcja wywoływana wyłącznie na potrzeby serwisu głównego, której odpowiedź służy do określenia aktywności wybranej gry w serwisie logiki.

* + Odpowiedź:
    - Flaga określająca stan zaakceptowania zapytania
    - Opisowa odpowiedź na zapytanie

Możliwe jest przekazanie identyfikatora pokoju, wtedy, podobnie jak polecenie Update, wykonywana jest procedura pobrania stanu gry. W odpowiedzi znajduje się minimalna liczba informacji pozwalającą na określenie bezczynności pokoju.

* + Ciało zapytania:
    - Nazwa identyfikująca instancję pokoju
  + Odpowiedź:
    - Flaga określająca stan zaakceptowania zapytania
    - Opisowa odpowiedź na zapytanie
    - Punktacja wewnątrz pokoju
    - Stan pokoju
      * Flaga opisująca czy w pokoju rozgrywana jest partia gry
      * Wartość sygnalizująca, że bieżąca runda zakończyła się zwycięstwem gracza o podanym identyfikatorze
      * Flaga sygnalizująca, że bieżąca runda zakończyła się remisem
* Open

Funkcja udostępnia polecenie otwarcia nowej instancji pokoju gier. Domyślnie, to polecenie nie powoduje rozpoczęcia nowej partii.

* + Ciało zapytania:
    - Nazwa identyfikująca instancję pokoju
  + Odpowiedź:
    - Flaga określająca stan zaakceptowania zapytania
    - Opisowa odpowiedź na zapytanie
* Close

Analogicznie do funkcji Open wywołanie tej procedury powoduje zamknięcie instancji pokoju gier.

* + Ciało zapytania:
    - Nazwa identyfikująca instancję pokoju
  + Odpowiedź:
    - Flaga określająca stan zaakceptowania zapytania
    - Opisowa odpowiedź na zapytanie

### Dostępne rodzaje plansz

Wewnątrz specyfikacji interfejsu zdefiniowane zostały trzy interaktywne dla użytkownika obszary nazywane planszami. Każdy z nich przedstawia wybraną część wizualną gry planszowej. Rodzaj obszaru identyfikowany jest przez wartość pola type w deklaracji.

Obsługiwane typy plansz:

* „simple”  
  Obszar przedstawiający standardową prostokątną planszę. Składa z informacji o liczbie pól w pionie i poziomie, rozmiarze w pikselach pojedynczego pola oraz listy tekstur wykorzystywanych do reprezentacji stanu. Pozycja na planszy jest reprezentowana przez parę znaków, gdzie pierwszy znak oznacza numer kolumny, a drugi wiersza. Wymiar może być indeksowany cyframi lub literami, dodatkowo rodzaje znaków są niezależne od siebie.
* „custom”  
  Reprezentacja planszy o dowolnym kształcie i liczbie pól. Deklarację rozpoczyna się od podania wymiarów obszaru oraz pojedynczego pola wraz z listą tekstur do wizualizacji stanów. Kolejno precyzuje się rodzaj wypełnienia tła. Do wyboru jest jednolity kolor w formacie zgodnym z kolorami CSS lub grafika, która może być dowolnym adresem URI osiągalnym przez użytkowników. Ostatnim elementem definicji jest lista wszystkich pól planszy, gdzie jedno pole reprezentowane jest przez trójkę wartości: współrzędne x i y lewego górnego wierzchołka pola oraz jego nazwa, którą będzie identyfikowane. Nazwy nie muszą unikalne, pozwala to na grupowanie pól o tych samych własnościach.

Ze względu na politykę Same-origin wszystkie względne adresy grafik, są obsługiwane przez serwis główny. Dopuszcza się stosowanie adresów absolutnych do serwerów zewnętrznych.

* „random”  
  Ostatnim dostępnym rodzajem jest obszar reprezentujący pole rzutu kością(kośćmi) do gry. Podczas deklaracji podaje się liczbę kości oraz ile razy mają się przetoczyć po wykonanym rzucie. Kolejno określa się sposób generacji kolejnych ścianek podczas toczenia. Dostępna jest uproszczona symulacja rzeczywistej kości sześciościennej oraz arbitralnej o dowolnej liczbie oczek.

## Przykładowa implementacja

W ramach projektu został opracowany serwis logiki gier wraz z przykładową implementacją interfejsu zgodną ze specyfikacją. W efekcie powstało rozwiązanie pozwalające na obsługę popularnej gry planszowej w Chińczyka (ang. Man, Don’t Get Angry).

### Deklaracja gry

Pierwszym elementem implementacji jest przygotowanie deklaracji gry przesyłanej w funkcji API interfejsu. Definicję można podzielić na trzy funkcjonalne fragmenty. Na początku znajdują się informacje podstawowe tj. nazwa gry w formacie akceptowalnym jako adres URL, adres internetowy lub nazwa domenowa serwera, port protokołu TCP na którym nasłuchuje serwer https, krótki opis dla użytkowników oraz liczba obsługiwanych graczy na pokój.

W drugiej części są opisane wszystkie wykorzystywane plansze na potrzeby interfejsu użytkownika. Gra korzysta z jednej planszy o nieregularnych kształtach. Płótno jest kwadratem o boku długości 736px, wypełnionym grafiką pobraną z serwisu logiki. Pozostałe tekstury pochodzą z repozytorium serwera głównego. Następnie wylistowane są wszystkie pola planszy, gdzie każde z nich jest kwadratem o boku długości 64px. Przykładowe osiem pól w formacie JSON:

* {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’},

Miejsca odpowiadające domkowi lub mecie dla danego gracza współdzielą nazwę, dzięki czemu gracz może wybrać dowolne z nich, aby uzyskać ten sam efekt.

Drugim interaktywnym obszarem jest przestrzeń do rzutu pojedynczą kością sześciościenną. Wykorzystywana jest symulacja rzeczywistego toczenia z dziesięcioma obrotami na rzut.

Na końcu podane są ścieżki do każdej z funkcji interfejsu zgodne ze specyfikacją openAPI. Dla przykładu polecenie API dostępne jest pod adresem /ManDontGetAngry/api.

### Serwer https

Kolejnym elementem implementacji jest opracowanie serwera https w stylu REST, który udostępni wymagane funkcjonalności. W tym celu można wykorzystać pakiet Express.js środowiska Node.js. Opracowany serwis logiki przystosowany został do obsługi kilku gier na raz, z tego powodu proponowane rozwiązanie jest parametryzowane nazwą gry.

Przykładowa implementacja ścieżki dla funkcji API:

1. app.route('/:GameName/api')
2. .get((req, res) => {
3. if (!description[req.params.GameName]) {
4. res.json({ accepted: false,
5. message: 'Gra nie istnieje!' });
6. return;
7. }
8. res.json({ accepted: true,
9. api: description[req.params.GameName] });
10. })
11. .post((req, res) => {
12. if (!description[req.params.GameName]) {
13. res.json({ accepted: false,
14. message: 'Gra nie istnieje!' });
15. return;
16. }
17. res.json({ accepted: true,
18. api: description[req.params.GameName] });
19. });

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.

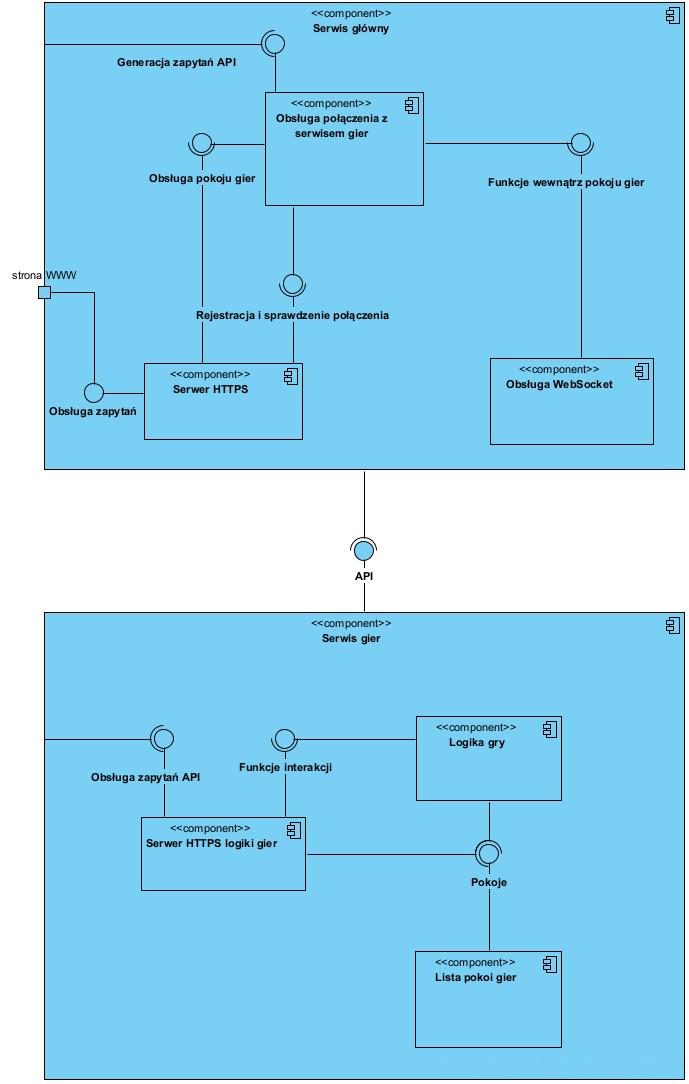
Logikę gry można zaprojektować jako klasę lub pakiet, którego odpowiednie funkcje wywoływane są w zależności od ścieżki.

Przykład obsłużenia funkcji Update oraz odpowiadającej jej części logiki gry:

1. app.post('/:GameName/Update', (req, res) => {
2. const gameName = req.params.GameName;
3. const game = games[gameName];
4. if (!game) res.json({ accepted: false,
5. message: "Game doesn't exists!" });
6. const rid = req.body.room;
7. const roomName = rid;
8. const room = rooms.get(roomName);
9. if (room) {
10. const player = req.body.player;
11. const data = **game.getUpdate(room, player);**
12. if (Object.keys(data).length) {
13. res.json({
14. ...{ accepted: true,
15. message: 'Request successful' },
16. ...data,
17. });
18. } else
19. res.json({ accepted: false,
20. message: 'Request unsuccessful' });
21. } else
22. res.json({ accepted: false,
23. message: "Room doesn't exists!" });
24. });
25. function getUpdate(room, playerId) {
26. const board =
27. room.board.concat(room.bases)
28. .concat(room.finishes);
29. saveZones.concat(startZones).forEach((zoneId) => {
30. board[zoneId] += 5;
31. });
32. for (var pid=0; pid<4; ++pid) {
33. for (var zid=0; zid<4; ++zid) {
34. const pos = 4\*pid+zid+40+16;
35. board[pos] = 10+2\*pid+(board[pos]>0?1:0);
36. }
37. }
38. return {
39. board: board,
40. score: room.score,
41. state: room.state,
42. nextMove: room.nextMove,
43. nextPlayer: room.player,
44. };
45. }

# Opis aplikacji

## 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

## Model przepływu informacji

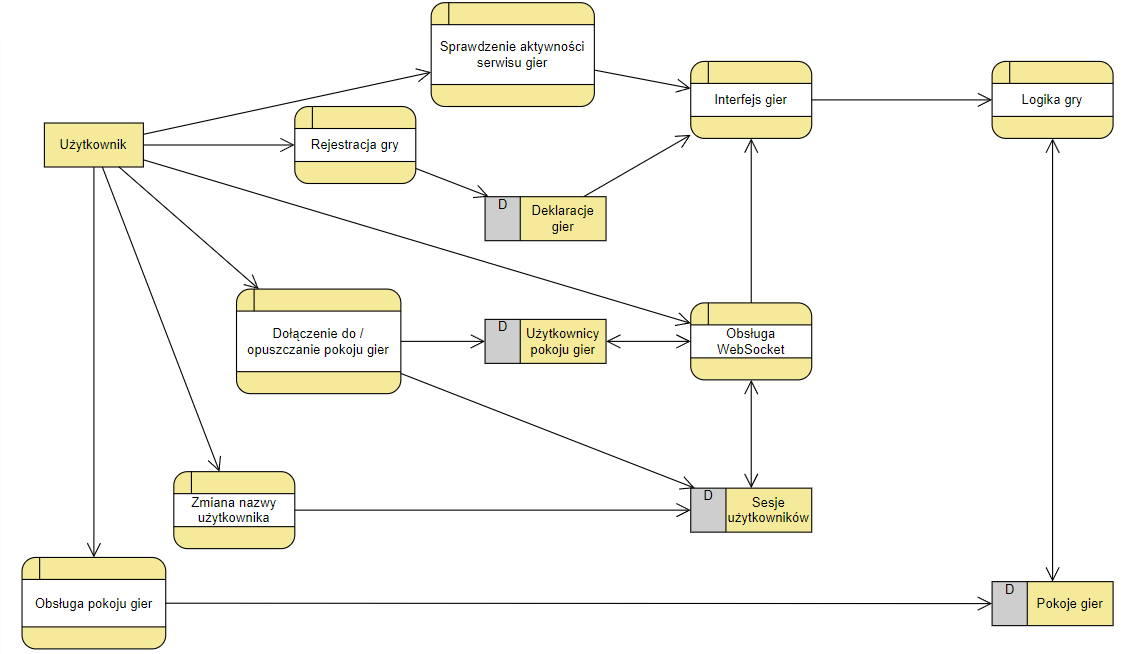


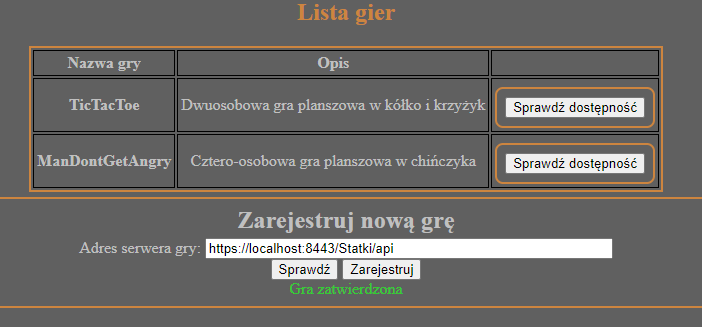
Diagram przepływu danych przedstawia diagram zerowy sytemu od udostępniania gier planszowych, gdzie serwis gier zostaw włączony do serwisu głównego. Tak prezentacja pozwala na lepsze zrozumienie przepływu informacji między funkcjami.

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

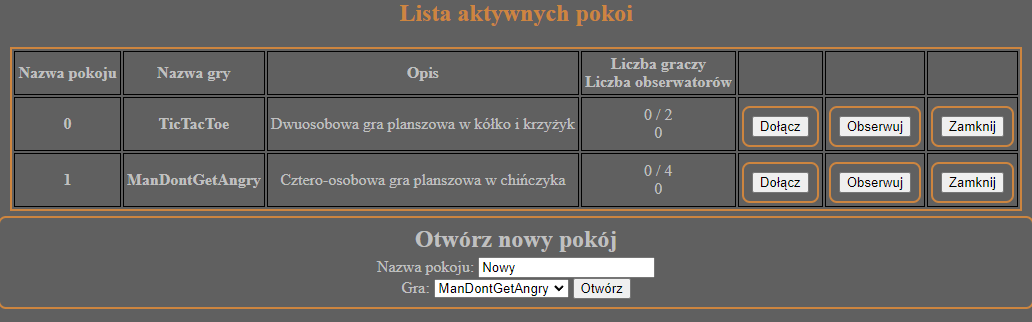
|  |  |  |
| --- | --- | --- |
| Komponent | Serwis główny | Serwis gier |
| Interfejs gier | Część przygotowująca zapytania zgodne z interfejsem | Część przygotowująca odpowiedzi zgodne z interfejsem |
| Deklaracje gier | Lista gier zarejestrowanych w serwisie | Lista gier udostępnianych w serwisie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

# Prezentacja

## Rejestracja gry



## Otwarcie pokoju i połączenie użytkownika



## Chińczyk

# Wdrożenie

## W chmurze

## Docker + reverse-proxy

# Podsumowanie

# Wnioski

# Bibliografia?