



# **Państwowa Wyższa Szkoła Informatyki i Przedsiębiorczości w Łomży**

**Wydział Informatyki i Nauk o Żywności**

**Kierunek studiów: Informatyka I stopień**

**Specjalność: Systemy oprogramowania**

**Adrian Sokołowski**

7118

## **GRA SIECIOWA W ŚRODOWISKU APLIKACJI WEBOWEJ A NETWORK GAME IN A WEB APPLICATION ENVIRONMENT**

**Promotor:**

.....  
.....

(podpis promotora)

**Łomża .....**



# SPIS TREŚCI

<b>1. Wstęp .....</b>	<b>2</b>
1.1. Uzasadnienie wyboru tematu .....	3
1.2. Cel pracy .....	3
<b>2. Gry sieciowe .....</b>	<b>4</b>
2.1. Historia gier .....	5
<b>3. Aplikacje webowe .....</b>	<b>6</b>
<b>4. Wykorzystane technologie .....</b>	<b>7</b>
4.1. HTML i CSS .....	7
4.2. Bootstrap .....	8
4.3. JavaScript i jQuery .....	8
4.4. NodeJS .....	10
4.5. Socket.io .....	11
4.6. ExpressJS .....	11
<b>5. Projekt gry .....</b>	<b>12</b>
5.1. Wymagania funkcjonalne .....	13
5.2. Wymagania pozafunkcjonalne .....	13
5.3. Logika gry .....	14
<b>6. Architektura aplikacji .....</b>	<b>14</b>
6.1. Klient .....	19
6.2. Serwer .....	29
<b>7. Opis stworzonej aplikacji .....</b>	<b>35</b>
<b>8. Podsumowanie .....</b>	<b>36</b>

<b>Literatura .....</b>	<b>37</b>
<b>Spis rysunków .....</b>	<b>38</b>
<b>Spis tabel .....</b>	<b>39</b>
<b>Spis załączników .....</b>	<b>40</b>

# 1. Wstęp

W obecnych czasach Internet (ang. *Inter-network*) wywiera na nas ogromny wpływ. Zmienia się nasza kultura oraz sposób postrzegania rzeczywistości. Nie tak dawno dostęp do Internetu za pośrednictwem komputerów miało wąskie grono odbiorców, co zmieniał się z roku na rok. Wraz z postępem technologicznym powstawały kolejne pomysły i stopniowo stawały się one możliwe do realizowania. [1] Na nauczanie i rozwój nauk technicznych przeznaczono duże sumy pieniędzy, z których skorzystali również informatycy. Internet został stworzony przez ARPA (ang. *Advanced Research Project Agency*) dla celów wojskowych. Internet umożliwia błyskawiczną wymianę danych z zawartymi informacjami pomiędzy komputerami na całym świecie. Informacje w postaci danych możemy przekazywać poprzez dostępne narzędzia. Najpowszechniejszym narzędziem obsługującym zasoby Internetu dostępnym dla każdego użytkownika stała się przeglądarka internetowa (ang. *web-browser*), jako niezbędny program do wyświetlania zawartości stron internetowych znajdujących się na serwerach podłączonych do sieci W3. Z upływem lat strony internetowe rozwijały się, czego wynikiem było powstanie nowych pojęć określających typ danej strony. Interaktywną stronę internetową, bardziej skomplikowaną od zaprogramowanej głównie w języku HTML nazwano aplikacją webową.

Pierwsze komputery stworzone przez wynalazców zajmowały ogromne pomieszczenia w granicach kilkunastu metrów kwadratowych. Dziś komputer można zmieścić w ludzkiej dłoni. Komputerem nazwano urządzenie przeznaczone do przetwarzania informacji. Główną cechą odróżniającą komputery od innych urządzeń mechanicznych jest możliwość ich programowania, wprowadzając do ich pamięci list instrukcji, które mają być wykonane. Komputer po podłączeniu do sieci internetowej może komunikować się z innymi urządzeniami poprzez wysyłanie lub odbieranie danych w postaci liczb binarnych. Na przestrzeni ostatnich lat można zaobserwować ogromny wzrost technologiczny. Informatyka (ang. *computer science*) jako dziedzina nauk ścisłych i technicznych nieustannie się rozrasta. Współczesna informatyka jest obecnie bardzo szeroką dziedziną, często towarzyszącą nam w różnych aspektach życia codziennego.

## **1.1. Uzasadnienie wyboru tematu**

W trakcie edukacji na uczelni Państwowej Wyższej Szkole Informatyki i Przedsiębiorczości w Łomży obok wielu dziedzin z obszaru informatycznego autora szczególnie zainteresowały technologie internetowe. Dlatego zdecydował się na próbę utworzenia aplikacji webowej podążając za trendami w środowisku programowania. Statyczne aplikacje stają się technologią przestarzałą, więc autor uznał, że utworzenie gry sieciowej z wykorzystaniem dotychczas zdobytej przez niego wiedzy w zakresie technologii informatycznych, stworzyłoby mu drogę do poznania bardziej szczegółowo zasad panujących przy tworzeniu nowoczesnych aplikacji.

## **1.2. Cel pracy**

Celem pracy inżynierskiej jest stworzenie aplikacji przy użyciu dostępnych narzędzi w środowisku internetowym. Autor podjął się stworzenia prostej implementacji gry sieciowej działającej w czasie rzeczywistym. Cechą każdej gry jest to, że zazwyczaj składają się one z skomplikowanych procesów, dając możliwość wykorzystania środowiska programistycznego w pełnej okazałości. Rozgrywka w grze powinna być przyjazna graczom. W ogólnym założeniu powinna ona zawierać przyjazne sterowanie, poprawne kolizje oraz przewidywalną fizykę. Wszystkie te elementy powinny współgrać ze sobą. Ponadto dodanie opcji rozgrywki dla wielu graczy czyni ją znacznie bardziej złożoną, ponieważ na etapie pracy nad projektem trzeba utworzyć także serwer obsługujący wszystkie zdarzenia synchronizując je ze sobą. W projekcie autor omówił oraz stworzył podstawową funkcjonalność gry w oparciu o środowisko aplikacji webowych. W wieloosobowej grze sieciowej każdy gracz powinien łączyć się z serwerem jako klient, informując serwer o wykonanych działaniach. Serwer z kolei musi interpretować otrzymane dane i przysyłać je do innych klientów połączonych z serwerem. Trzeba mieć na uwadze czynniki, na które nie mamy wpływu jak np. opóźnienie klient-serwer.

## 2. Gry sieciowe

Grą komputerową nazywamy program, którego główną cechą jest ingerencja użytkownika. Stan gry zmienia się na podstawie podjętych przez gracza czynności. Każda gra powinna mieć cel do wykonania. Gry sieciowe jak sama nazwa wskazuje są grami, w których udział bierze więcej niż jeden gracz przy użyciu łącza internetowego. Do niedawna przeglądarki internetowe służyły głównie do przeglądania statycznych treści zamieszczonych przez innych użytkowników. Dziś technologia pozwala na użycie przeglądarki do pisania o wiele bardziej złożonych aplikacji. W niedalekiej przeszłości przy pisaniu gier w przeglądarkach internetowych najczęściej sięgano po technologie takie jak Flash, PHP czy Java. Po wydaniu kolejnej wersji języka znaczników HTML, rozwijając go o element Canvas powstał trend tworzenia gier bezpośrednio w przeglądarce. Jednak gry sieciowe z użyciem Canvas tworzą dopiero po udostępnieniu frameworka NodeJS, gdyż sama przeglądarka nie mogła obsługiwać zdarzeń interaktywnie w czasie rzeczywistym.

Gry sieciowe należą bez wątpienia do bardziej skomplikowanego oprogramowania jakiego można stworzyć. Oprócz elementarnych funkcji, jakie oferuje nam środowisko, dochodzi ingerencja użytkownika, na którą składa się więcej procesów. W przeciwieństwie do gier lokalnych gry sieciowe mają zazwyczaj charakter konkurencyjny z uwagi na fakt, że bierze w nich udział więcej niż jeden gracz. Choć istnieje również wiele odmian trybów gry, w których gracze nie rywalizują ze sobą, lecz np. współpracują w celu spełnienia wymagań dostarczonych przez daną grę komputerową.

### 2.1. Historia gier

Pierwszy prototyp gry komputerowej powstał w 1947 roku. Był to symulator pocisku rakietowego. Jednakże pierwsze gry w formie graficznej powstały dopiero po wynalezieniu komputera EDSAC. Pionierem w tworzeniu gier był wówczas Alexander Sandy Douglas. Stworzył on w ramach swojej pracy doktorskiej na uniwersytecie w Cambridge adaptację gry w kółko i krzyżyk. Następnie powstawały coraz to bardziej rozbudowane wersje gier. William Higinbotham stworzył grę opartą o mechanikę gry w tenisa, którą nazywał Tennis for Two. Historia gier komputerowych obejmuje kilka generacji konsol. Wraz z produkowaniem coraz

to nowszych urządzeń powstawały bardziej zaawansowane gry. Oprócz konsol, popularne były automaty do gier. Kolejne innowacje w branży gier wideo zapoczątkowały komputery osobiste był to okres na przełomie lat 80 i 90. XX wieku. [2] W tamtych czasach użytkownicy komputerów mieli do dyspozycji w zasadzie wyłącznie tryb tekstowy, tzn. możliwości graficzne sprzętu ograniczały się do wyświetlania na ekranie liter i cyfr.

Niedługo po wyprodukowaniu komputerów osobistych, gry zaczęły udostępniać graczom możliwość rozgrywki sieciowej. Co było wynikiem rozpowszechnienia Internetu. Z biegiem czasu powstały nowsze generacje konsol, służące głównie do gier. Jednak gry były implementowane nie tylko na konsolach, ale również na innych urządzeniach jak np. telefony komórkowe, które stały się urządzeniami wielofunkcyjnymi. Przykładem takiego telefonu jest Nokia 3310 bez kolorowego wyświetlacza, w której znajdowała się gra Snake.



### 3. Aplikacje webowe

Aplikacje webowe (ang. web applications) nazywane także aplikacjami internetowymi to aplikacje, które uruchamiane są w przeglądarkach internetowych. Każda aplikacja webowa powinna posiadać własny interfejs, dzięki któremu użytkownik ma możliwość w sposób interaktywny wykonać czynności do których została ona stworzona. Co nie jest możliwe w przypadku statycznych stron internetowych mających charakter informacyjny, gdzie użytkownik może jedynie przeglądać umieszczone na niej treści. W odróżnieniu od aplikacji desktopowych zaletą aplikacji internetowych jest łatwy dostęp, ponieważ do skorzystania z danej aplikacji użytkownikowi wystarczy przeglądarka internetowa oraz dostęp do Internetu. W dzisiejszej dobie Internetu aplikacje webowe stają się coraz bardziej powszechne ze względu na wymagania stawiane przez użytkowników. Większość aplikacji webowych komunikuje się z serwerem. Użytkownik korzystający z aplikacji wywołuje zdarzenia po stronie przeglądarki, które z kolei wysyłane są do serwera. Serwer natomiast interpretuje otrzymane komunikaty i przesyła odpowiedź ingerując w jej stan.

Istotną różnicą pomiędzy statyczną stroną, a aplikacją webową jest środowisko, w którym została ona zaprogramowana. Statyczne strony internetowe buduje się w oparciu o języki HTML i CSS. Natomiast do tworzenia aplikacji internetowych stosuje się bardziej zaawansowane języki programowania jak np. PHP, NodeJS czy Java. Definiując aplikację webową stwierdza się, że jest to program pracujący na serwerze, który komunikuje się z urządzeniem użytkownika.

## 4. Wykorzystane technologie

Do stworzenia aplikacji użyto kilku technologii informatycznych. Wszystkie technologie, z których skorzystano ściśle współpracują ze sobą, aby osiągnąć wymierzony cel pracy dyplomowej. Każdy element jest niezbędny do funkcjonowania aplikacji webowej jaką jest stworzona gra sieciowa. Do wyświetlenia aplikacji w przeglądarce wykorzystano HTML, CSS oraz biblioteki Bootstrap. Bez tych technologii użytkownik nie widziałby procesów przebiegających w aplikacji, tudzież sam nie mógłby wpływać na stan rozgrywki. Największą rolę aplikacji spełnia JavaScript wraz z biblioteką JQuery. Język ten odpowiada za całą mechanikę gry. Dużą rolę w aplikacji odgrywa także środowisko uruchomieniowe Node.js, które działa jako serwer napisany w języku JavaScript. Język JavaScript był początkowo stworzony do działania wyłącznie w przeglądarkach internetowych. Do rozwiązania tego problemu wykorzystano Node.js, aby aplikacja mogła funkcjonować jako odosobniony serwer. Wizualizacja gry w szachy opiera się na komponencie chessboard.js, który wyświetla na ekranie szachownicę wraz z figurami, które można przemieszczać na ekranie. Chess.js sprawdza czy ruch jest poprawny i zgodny z zasadami przebiegającymi w tejże grze.

### 4.1. HTML i CSS

HTML (ang. Hypertext Markup Language) został stworzony przez organizację WW3 jako unikalny język znaczników obsługiwany przez przeglądarkę internetową. Wykorzystuje się go jako podstawę każdej strony WWW. Język ten służy jako frontend do zaprezentowania użytkownikowi treści w sposób wizualny. HTML jest [3] używany przez programistów od ponad 20 lat, przez pierwszych kilka był poddawany radykalnym zmianom, ale w końcu minionego stulecia jego rozwój się spowolnił. Przeglądarki, jako programy na urządzeniach różnego typu, parsują ten język. Dzięki czemu wiedzą, w jaki sposób wyświetlić dany element na stronie internetowej. Idea jest taka, aby każdy użytkownik korzystający z urządzenia widział aplikację w tej formie, w której autor kodu chciał nam przekazać. Aktualną wersją języka HTML jest wersja 5. Wersję określa się razem z nazwą jako HTML5.

CSS (ang. *Cascading Style Sheets*) to lista dyrektyw określających w jaki sposób przeglądarka ma wyświetlać elementy na stronie internetowej. Zarówno jak HTML został

opracowany przez organizację W3. Pierwsza implementacja CSS odbyła się w roku 1991 jako potomek języka DSSSL. Dyrektywy zazwyczaj pisze się w specjalnie do tego przeznaczonych plikach o rozszerzeniu .css, którego celem jest lepsza organizacja pisanego kodu zachowując przy tym semantykę tworzenia webowego oprogramowania.

## **4.2. Bootstrap**

Bootstrap jest to biblioteka o otwartym kodzie źródłowym służąca usprawnieniu tworzenia interfejsu graficznego stron internetowych oraz aplikacji webowych. Rozwijana jest przez programistów Twittera na licencji MIT. Biblioteka ta zbudowana jest głównie z HTML, CSS oraz JavaScript do budowania frontendu aplikacji.

## **4.3. JavaScript i jQuery**

JavaScript jest językiem służącym do zapewnienia interakcji poprzez reagowanie na zdarzenia, walidacji danych wprowadzanych w formularzach lub tworzenia złożonych efektów wizualnych. [6] Język JavaScript powstał w 1995 roku jako technologia umożliwiająca dodawanie programów do stron internetowych w przeglądarce Netscape Navigator. Dopiero później został zaimplementowany we wszystkich przeglądarkach internetowych. W tradycyjnych stronach internetowych JavaScript dodaje do nich wszelkiego rodzaju interaktywność. Podstawową cechą kodowania w tym języku jest rozróżnienie wielkości liter na małe i duże. Jest to język skryptowy i dynamiczny. Choć ma wiele różnych zastosowań, najczęściej stosuje się go przy tworzeniu stron internetowych lub aplikacji webowych.

jQuery to szybka, mała i bogata w funkcje biblioteka JavaScript. Sprawia, że takie rzeczy jak przechodzenie i manipulowanie dokumentami HTML, obsługa zdarzeń, animacja i Ajax są znacznie prostsze dzięki łatwemu w użyciu API, który działa w wielu przeglądarkach.

Według przeprowadzanych analiz przez Stackoverflow.com Javascript jest szósty rok z rzędu najpopularniejszym językiem programowania. W Tabeli 4.1 przedstawiono dziesięć najczęściej wybieranych języków programowania w roku ubiegłym, w kolejności od najpopularniejszego.

Tabela 4.1. Dziesięć najpopularniejszych języków programowania w 2018 roku według ankiety przeprowadzonej przez Stackoverflow.com

Język programowania	Popularność
Javascript	69.8%
HTML	68.5%
CSS	65.1%
SQL	57.0%
Java	45.3%
Bash/Shell	39.8%
Python	38.8%
C#	34.4%
PHP	30.7%
C++	25.4%

Źródło: <https://insights.stackoverflow.com/survey/2018>

#### 4.4. NodeJS

NodeJS jest łatwym w użyciu, elastycznym i wieloplatformowym narzędziem zbudowanym w oparciu o skryptowy język programowania JavaScript. Został stworzony do budowania skalowalnych aplikacji webowych. [5] Od chwili wydania przeglądarki Google Chrome w roku 2008 nieustannie i bardzo szybko poprawia się wydajność działania JavaScript, co jest wynikiem ogromnej konkurencji między producentami poszczególnych przeglądarek internetowych (Mozilla, Microsoft, Apple, Opera i Google).

Tabela. 4.2. Dziesięć najpopularniejszych frameworków, bibliotek i narzędzi w 2018 roku według ankiety przeprowadzonej przez Stackoverflow.com

Framework, biblioteka lub narzędzie	Popularność
Node.js	49.6%
Angular	36.9%
React	27.8%
.NET Core	27.2%
Spring	17.6%
Django	13.0%
Cordova	8.5%
TensorFlow	7.8%
Xamarin	7.4%
Spark	4.8%

Źródło: <https://insights.stackoverflow.com/survey/2018/>

## 4.5. Socket.io

Socket.io to biblioteka open source stworzona przez Guillermo Raucha. Zbudowana przy użyciu Engine.io, [8] który jest abstrakcją niższego poziomu na szczycie technologii Web-Socket. Zaletą tego rozwiązania jest szybka komunikacja dwukierunkowa niezbędna w grach w trybie rzeczywistym oraz znacznie mniejsze obciążenie niż standardowe żądanie protokołu HTTP.

Biblioteka służy do komunikacji w czasie rzeczywistym między serwerem Node.js, a przeglądarką internetową użytkownika. Komunikacja między serwerem, a klientem odbywa się dwukierunkowo, czyli dane wysyłane są z serwera do klienta i na odwrót w tym samym czasie, w odróżnieniu od protokołu bezstanowego, który umożliwia tylko wykonanie pojedynczych,

krótkich żądań dla serwera. Utworzenie gry, która pozwala graczom na wykonywanie określonych działań w tym samym momencie stało się możliwe dzięki technologii WebSocket. Biblioteka ta minimalizuje ilość kodu, który trzeba umieścić na podstawowym serwerze HTTP. Składnia biblioteki opiera się na odsłuchiowaniu i wykonywaniu zdarzeń.

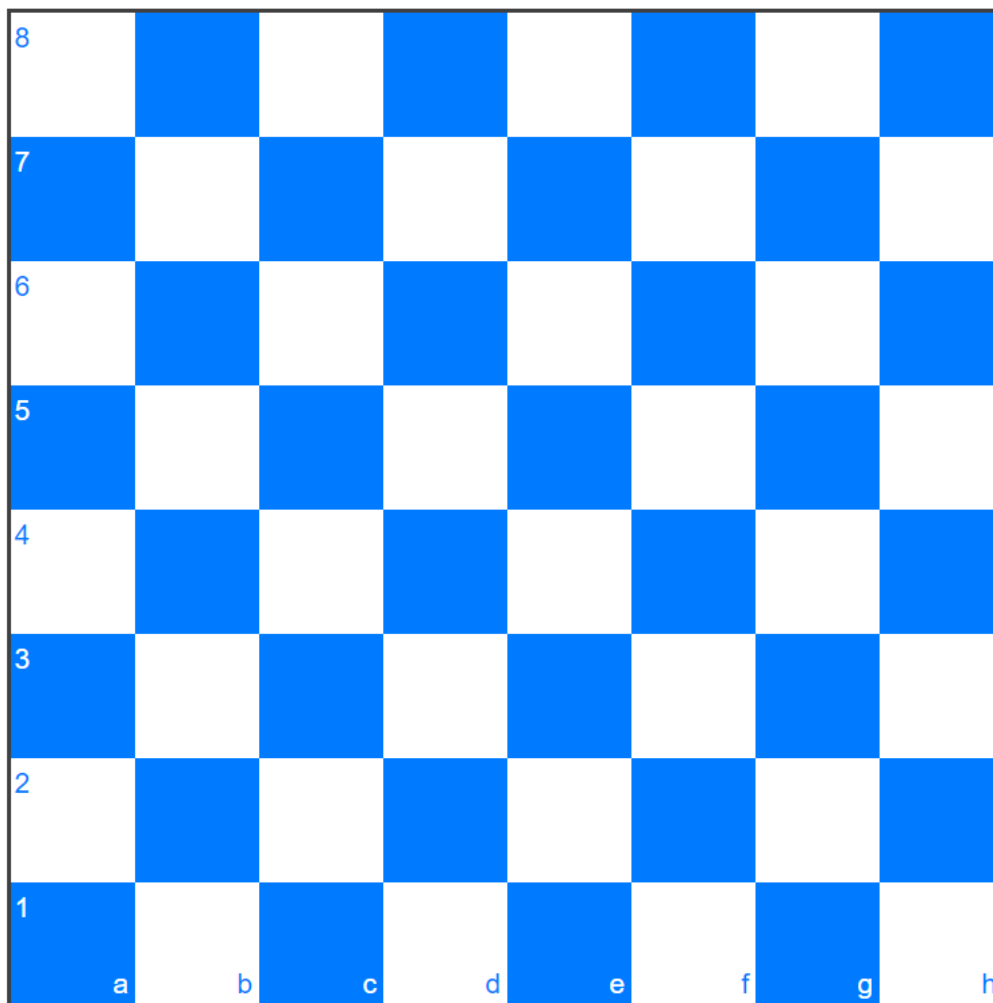
Celem Socket.io jest zapewnienie natywnej warstwy emulacji, która wykorzysta implementację WebSocket w przeglądarkach, które ją obsługują i powróci do innych metod takich jak HTTP long polling, aby symulować natywny interfejs API w przeglądarkach, które go nie obsługują. W przeciwieństwie do natywnej implementacji interfejsu WebSocket API socket.io wymaga niestandardowej biblioteki klienta: socket.io.js w przeglądarce internetowej oraz identycznego modułu po stronie serwera Node.js

## **4.6. ExpressJS**

ExpressJS jest minimalistycznym frameworkiem do Node.js stworzonym przez TJ Holowaychuk. Służy do usprawnienia pracy przy pisaniu kodu. Do korzystania z tego frameworka wymagane jest zainstalowanie platformy Node.js

## 5. Projekt gry

Projekt obejmuje klasyczną grę w szachy z graficznym interfejsem użytkownika. Rozgrywka w trakcie gry opiera się na podstawowych zasadach gry w szachy, a wszystkie bierki szachowe poruszają się tylko zgodnie ich przeznaczeniem. Implementacja szachów przeznaczona jest dla dwóch graczy. Gra rozgrywana jest na dwuwymiarowej szachownicy w kratkę 8 na 8 z kwadratem o kolorze niebieskim w lewym dolnym rogu okna gry każdego z graczy.



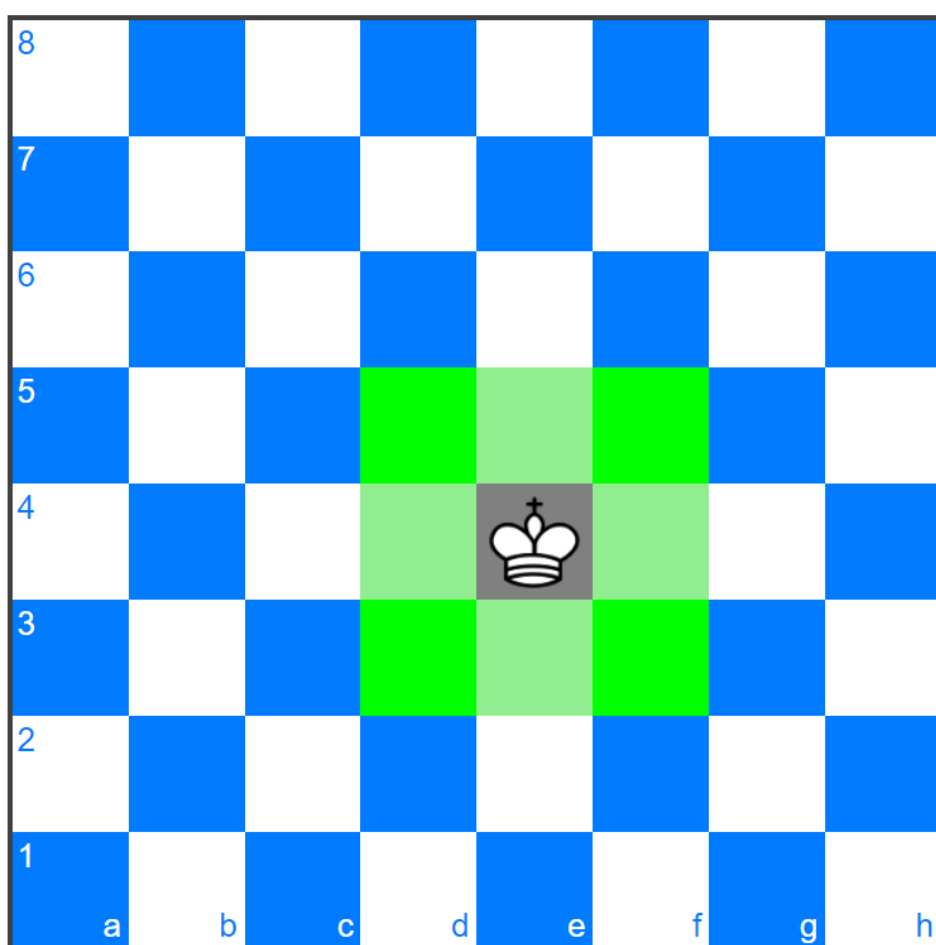
Rys. 5.1. Dwuwymiarowa szachownica 8x8

Najważniejszą figurą na szachownicy jest król. Decyduje on o wygraniu lub przegraniu partii. W początkowym ustawieniu znajduje się on pomiędzy hetmanem, a gońcem na pierwszej lub ostatniej linii. Może poruszać się o jedno pole w dowolnym kierunku. Nie może jednak

przemieszczać się na pola atakowane przez bierki przeciwnika. Jeśli w kolejnym posunięciu gracz zaatakuje króla nazywane jest to szachem. W takiej sytuacji atakowany król musi się bronić, poprzez zasłonięcie własną figurą, zbiciem atakującej figury lub ucieczką na inne pole. Uniemożliwienie królowi ucieczki nazywane jest matem i oznacza koniec rozgrywki.



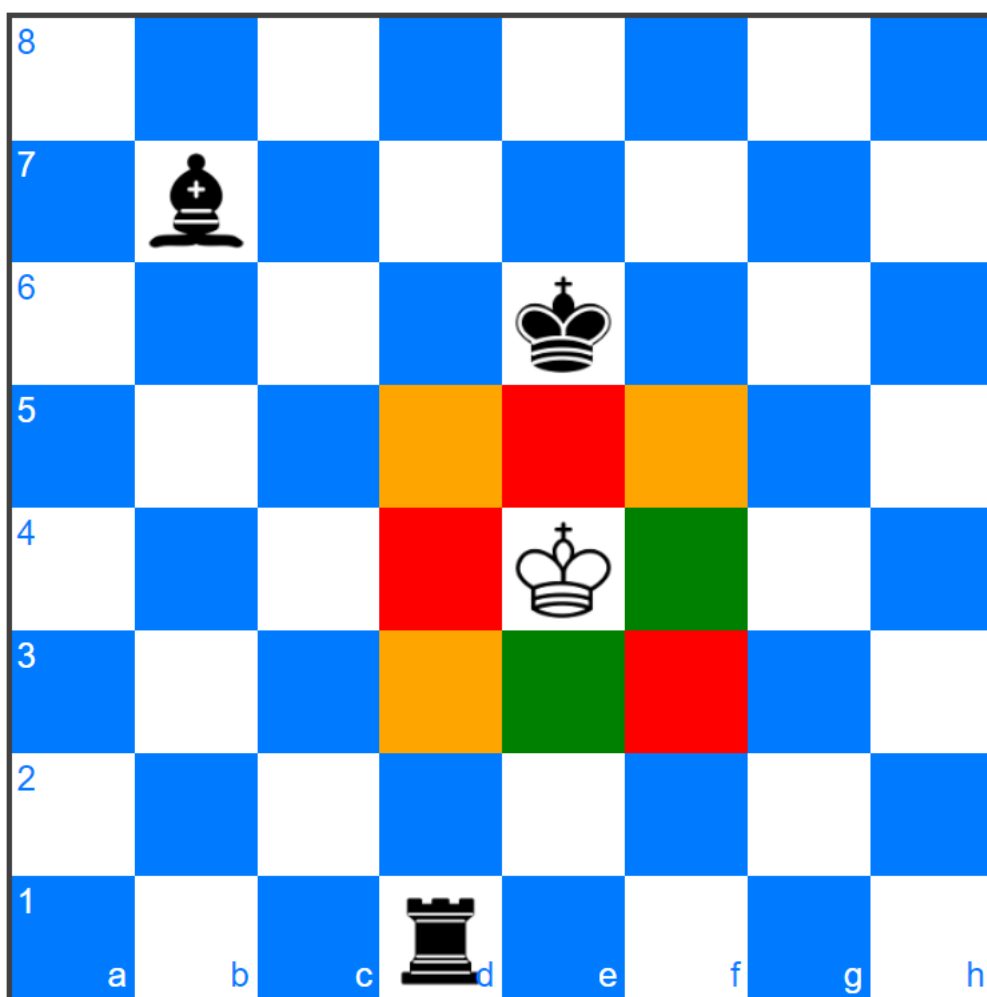
Rys. 5.2. Wizualizacja graficzna figury król.



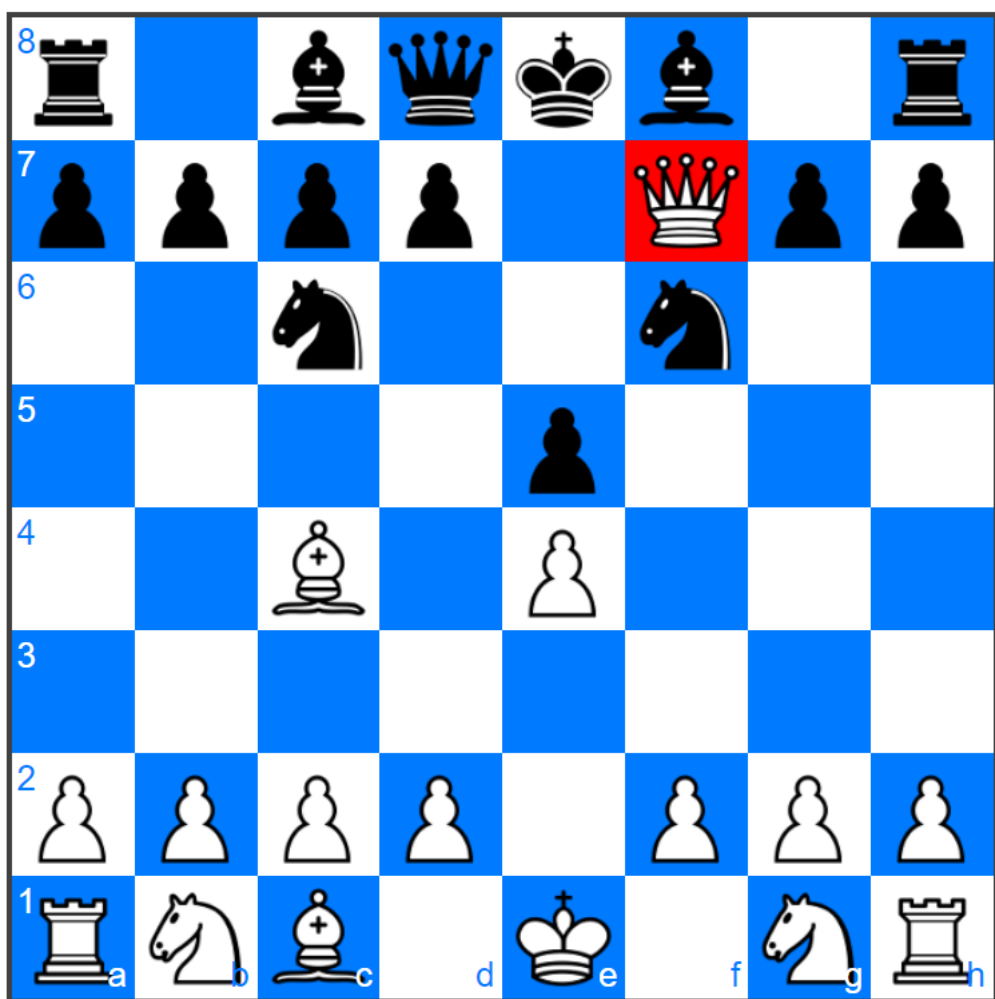
Rys. 5.3. Schemat poruszania się króla po szachownicy.

Na poniższym rysunku przedstawiona została sytuacja szachowa. Ruch mają teraz białe. Król może uciec tylko na pola koloru zielonego.





Rys. 5.4. Sytuacja szachowa.

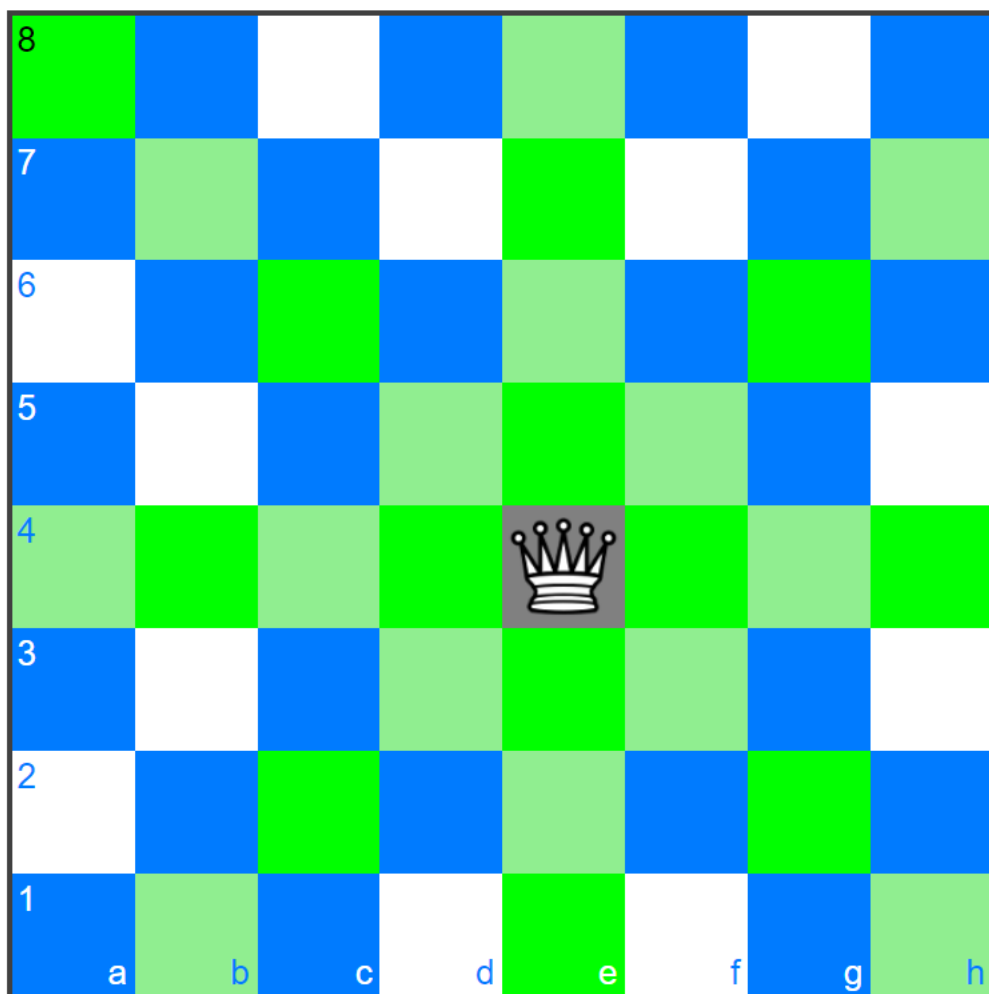


Rys. 5.5. Sytuacja matowa.

Najsilniejszą figurą w szachach jest hetman, nazywany również królowa lub dama. W ustawieniu początkowym znajduje się na pierwszym bądź ostatnim polu szachownicy. Porusza się we wszystkich kierunkach na szachownicy o dowolne pole.



Rys. 5.6. Wizualizacja graficzna figury hetman.

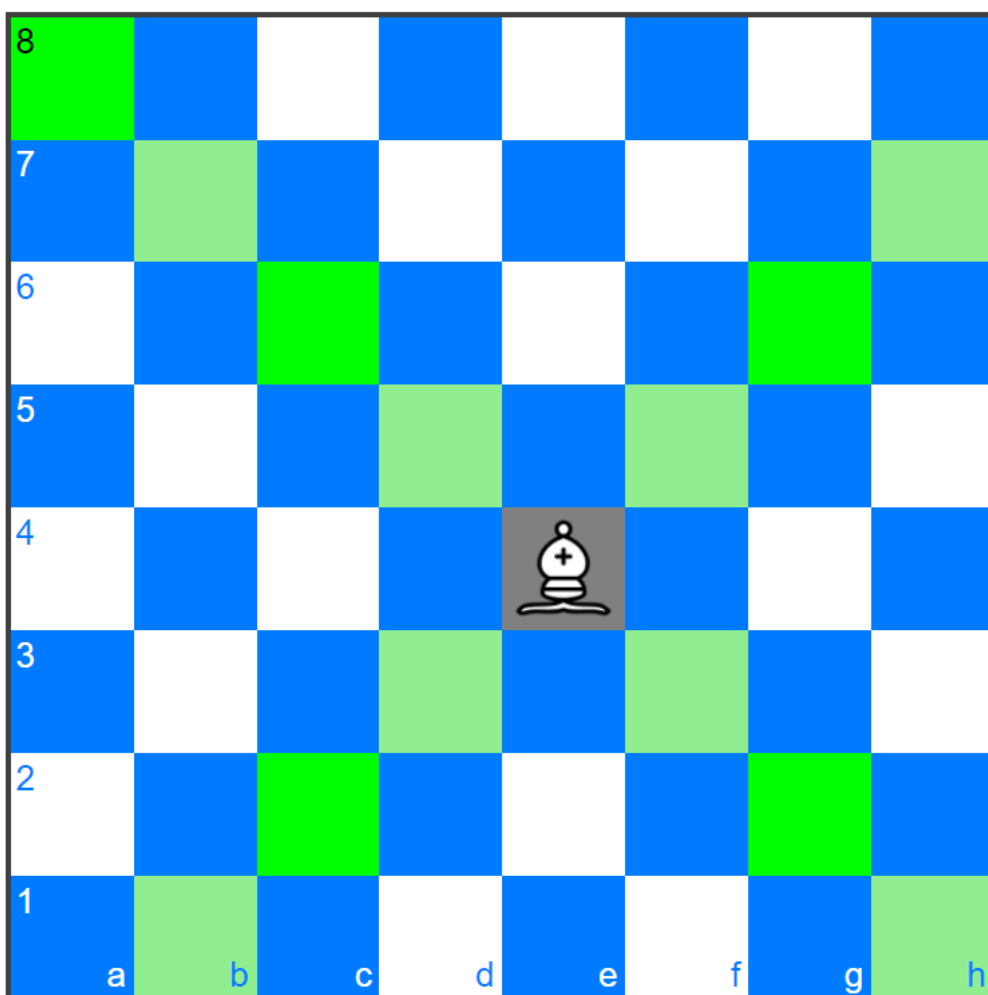


Rys. 5.7. Schemat poruszania się hetmana na szachownicy.

Goniec nazywany również laufrem porusza się tylko na ukos o dowolne pole we wszystkich kierunkach. Na szachownicy w początkowej fazie partii każdy z graczy posiada dwie figury na polach różnego koloru.



Rys. 5.8. Wizualizacja graficzna figury goniec.

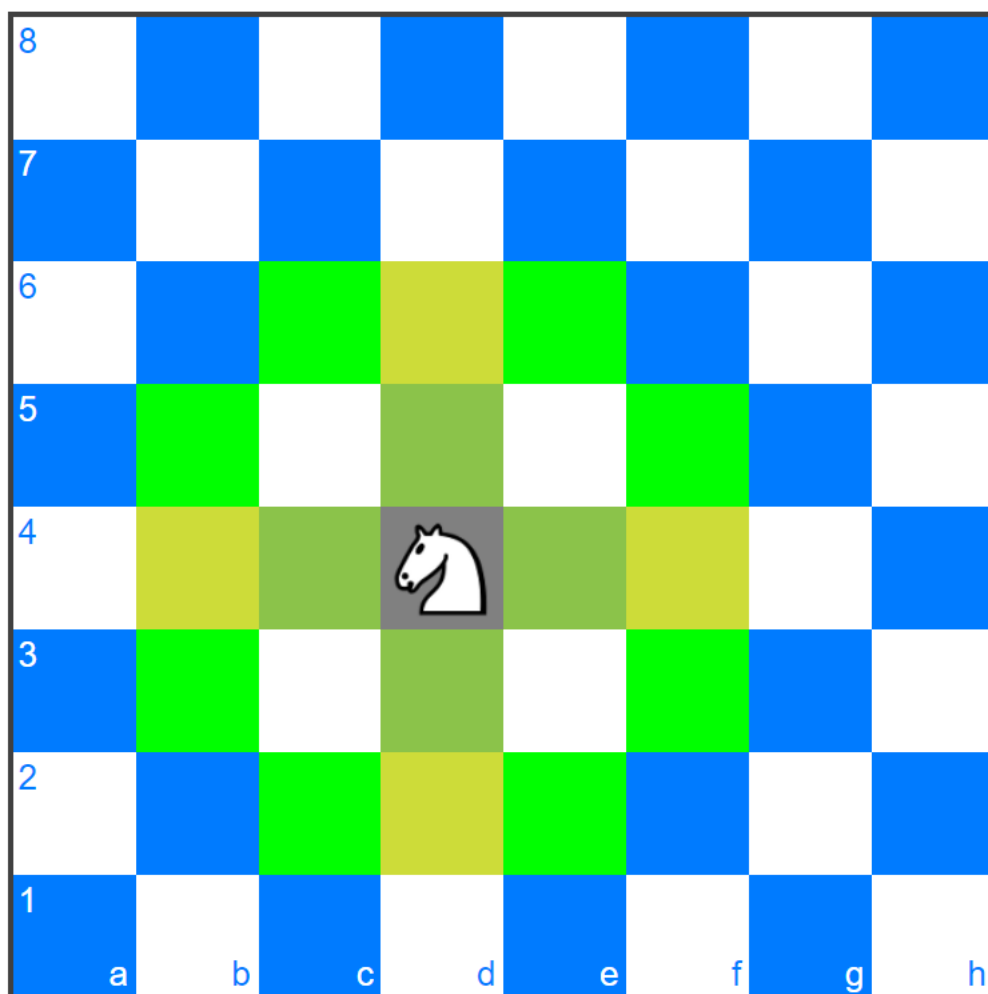


Rys. 5.9. Schemat poruszania się gońca na szachownicy.

Na początku rozgrywki, podobnie jak w przypadku gońca każdy z graczy posiada po dwie sztuki tej figury na pierwszej lub ostatniej linii szachownicy. Figura może poruszać się we wszystkich kierunkach. Ruch skoczka można opisać jako przesunięcie o jedno pole w pionie lub poziomie, a następnie na ukos. Jako jedyna figura może przeskakiwać przez inne bierki na szachownicy.



Rys. 5.10. Wizualizacja graficzna figury skoczek.

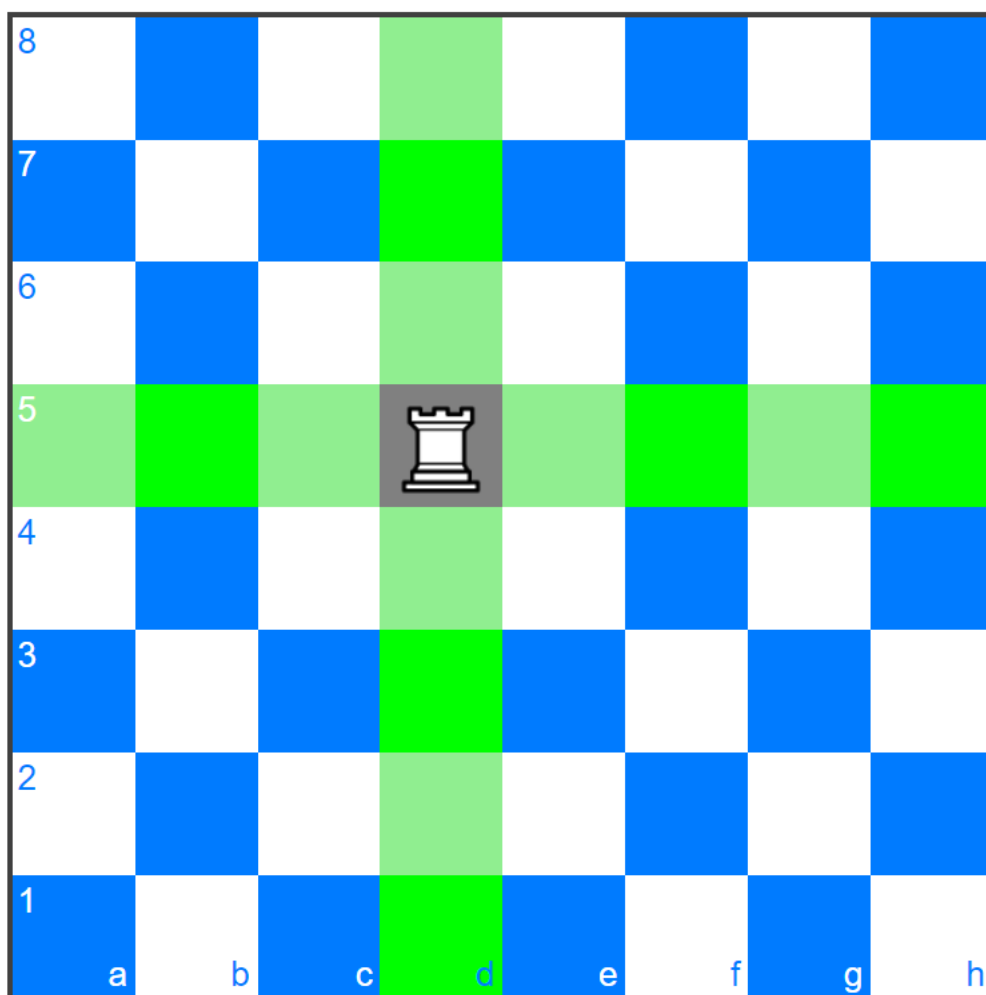


Rys. 5.11. Schemat poruszania się skoczna na szachownicy.

Wieża porusza się tylko w poziomie lub w pionie o dowolne pole we wszystkich kierunkach szachownicy. Na początku każdej partii gracze posiadają dwie wieże na pierwszej lub ostatniej linii w rogach szachownicy.



Rys. 5.12. Wizualizacja graficzna figury wieża.



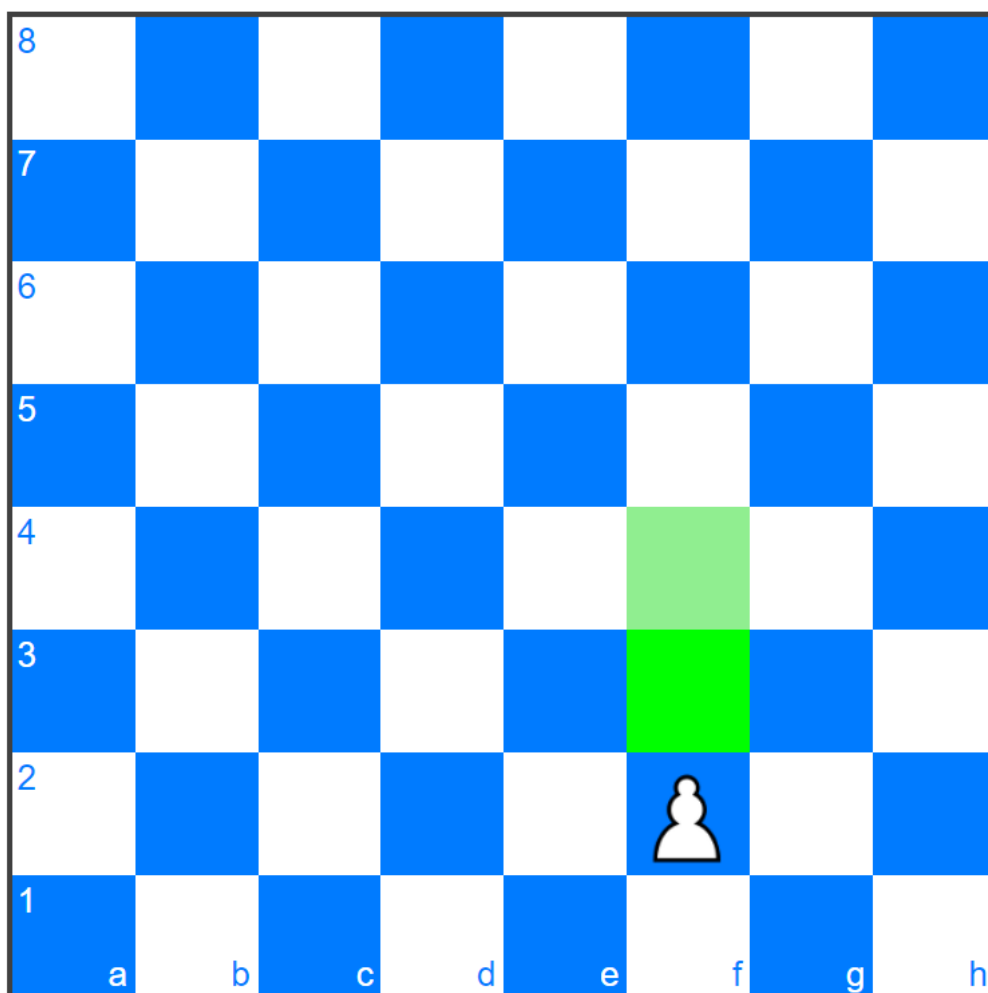
Rys. 5.13. Schemat poruszania się wierzcy na szachownicy.

Pion jest najsłabszą figurą, a zarazem najliczniejszą w rozgrywce szachowej. Może poruszać się tylko do przodu i bić na ukos w kierunku strony przeciwnika. Na początku partii każdy z graczy posiada osiem tych figur. Pion w odróżnieniu od innych figur szachowych posiada ruchy specjalne takie jak bicie i poruszenie się o dwa pola do przodu z pozycji startowej.



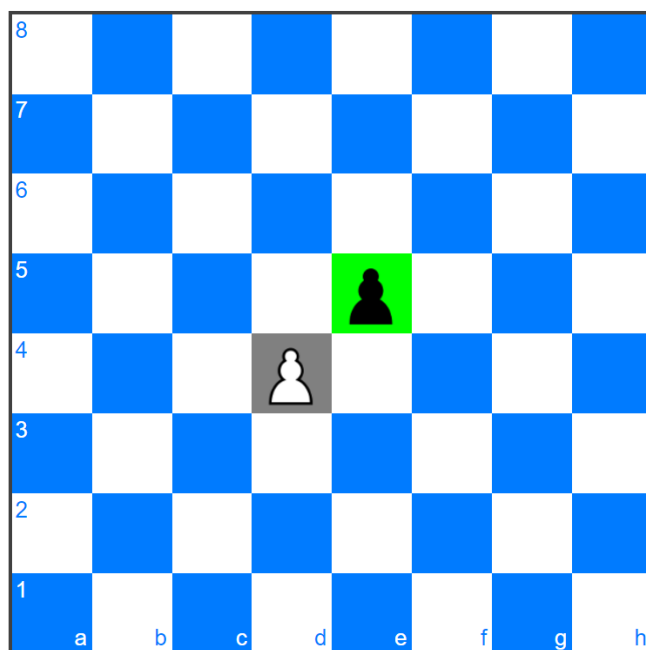
Rys. 5.14. Wizualizacja graficzna bierki pion.

Sposób w jaki porusza się pion na szachownicy przedstawiony jest na Rys. 5.15. W przedstawionej sytuacji pion znajduje się na polu F2. Jest to pozycja startowa klasycznej partii szachów dla tej bierki, więc oprócz ruchu o jedno pole do przodu, możliwe jest również wykonanie ruchu o dwa pola. W tej sytuacji gracz może zdecydować czy chce umieścić piona na polu F3, czy polu F4.



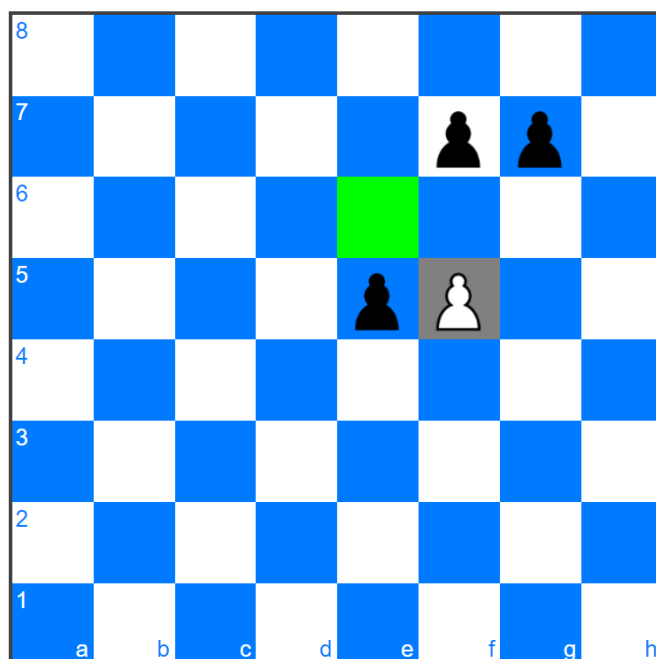
Rys. 5.15. Schemat poruszania się piona.

Pion może zbijać inne bierki tylko na skos.



Rys. 5.16. Bicie pionem.

Bicia w przelocie można dokonać tylko w pierwszym posunięciu po ruchu Piona przeciwnika o dwa pola. Jeżeli Białe ruszą się inną figurą, to tracą przywilej bicia w przelocie.



Rys. 5.17. Bicie w przelocie.



## 5.1. Wymagania funkcjonalne

Z perspektywy gracza ważne są wymagania funkcjonalne gry, gdyż określają one wszystkie możliwości jakie oferuje aplikacja.

Wymagania funkcjonalne:

- Użytkownik łączy się z serwerem, posiada własny identyfikator w postaci loginu
- Po wejściu do aplikacji gracz znajduje się w lobby.
- Gracz ma możliwość dołączenia do wybranego pokoju gry.
- Gra ma być symulacją szachownicy i bierek.
- Bierki są przesuwane na ekranie za pomocą wskaźnika myszy.
- Aplikacja nie pozwala na nieprawidłowe ruchy bierek.
- Gracz nie pobiera na swoje urządzenie żadnego oprogramowania do uruchomienia gry, wykorzystuje tylko przeglądarkę internetową.
- Gracz otrzymuje dane w czasie rzeczywistym o połączonych graczach i ich pozycjach.
- Gracz jest usunięty z lobby po rozłączeniu z serwerem.
- Ruch na szachownicy przebiega turowo.

## 5.2. Wymagania pozafunkcjonalne

Oprócz wymagań funkcjonalnych opisane zostały wymagania pozafunkcjonalne. Wymagania te mają wpływ na działanie gry. Ale nie są aż tak istotne dla gracza.

Wymagania pozafunkcjonalne gry:

- Gra na serwerze powinna działać płynnie.
- Gra działa na serwerze NodeJS
- Gra nie może używać zewnętrznych aplikacji do renderowania gry, renderowanie powinno przebiegać w przeglądarce.
- Gra ma działać na każdym serwerze z zainstalowanym NodeJS.
- Dostęp do aplikacji wyłącznie przez przeglądarkę internetową.

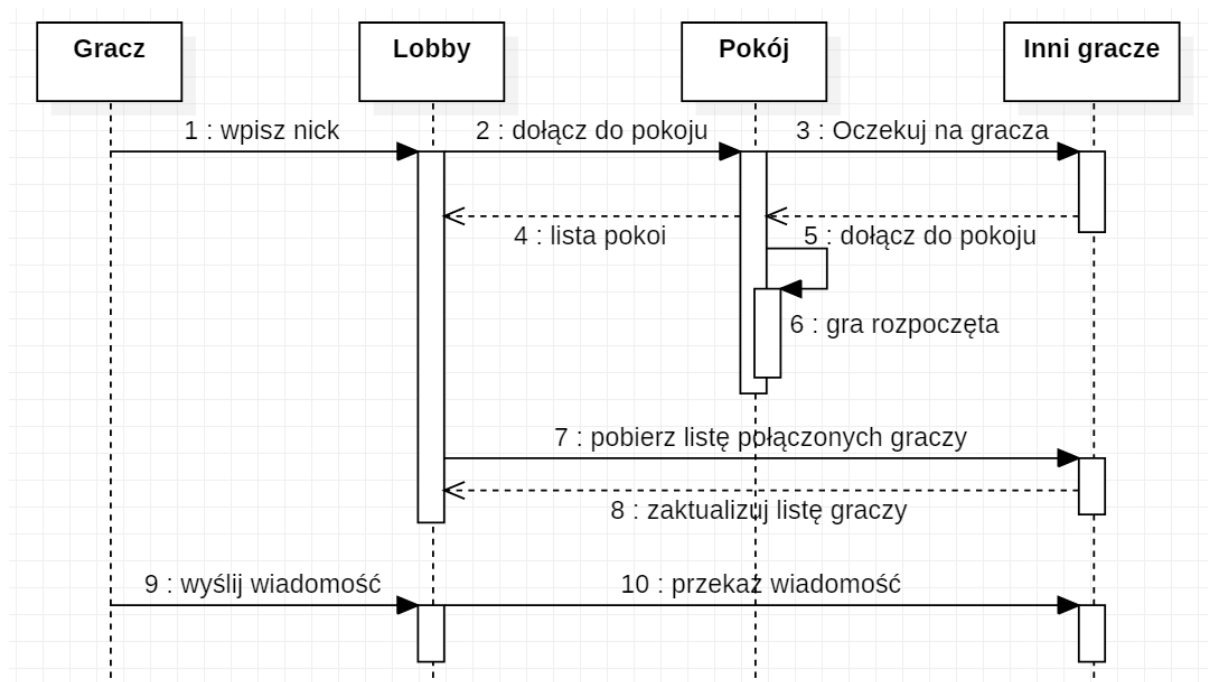
### 5.3. Logika gry

Poniższy diagram przypadków użycia obrazuje czynności jakie może wykonać użytkownik. Po połączeniu z aplikacją, oczom użytkownika ukazuje się ekran startowy, w którym wymagana jest nazwa użytkownika, której gracz użyje jako swój identyfikator w systemie lobby. Po podłączeniu do lobby, gracz otrzymuje listę nazw oraz ilość graczy połączonych już z serwerem, może komunikować się na czacie z innymi graczami, a także dołączyć do pokoju gry, w którym znajduje się gracz, aby rozpocząć rozgrywkę lub do pokoju, w którym nie ma żadnego gracza w oczekiwaniu aż któryś dołączy. Gdy w pokoju znajduje się wymagana liczba graczy gra zostaje rozpoczęta i jeden z graczy może wykonać posunięcie. Rozgrywkę rozpoczyna gracz, który posiada białe figury.



Rys. 5.18. Diagram przypadków użycia.

Poniższy diagram sekwencji zawiera 4 obiekty są nimi kolejno Gracz, czyli klient podłączony do serwera, lobby, pokój oraz inni gracze, jako pozostali klienci. Lobby oraz pokój działają natomiast po stronie serwera.

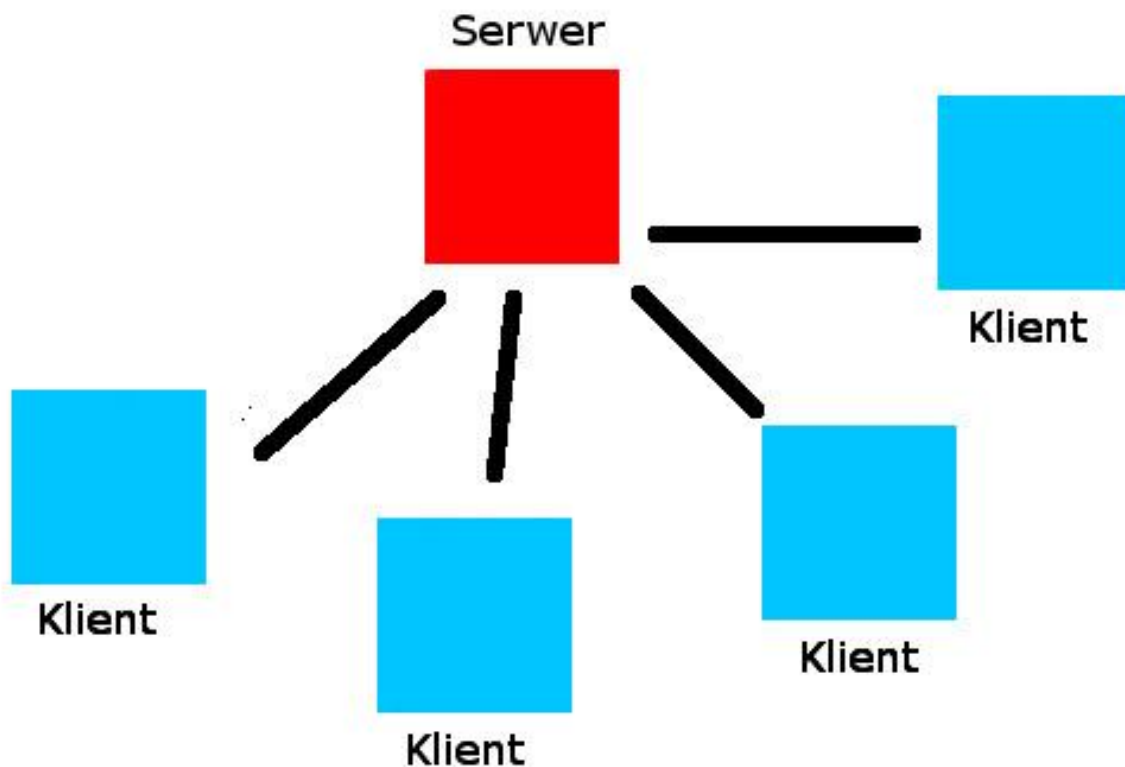


Rys. 5.19. Diagram sekwencji.

## 6. Architektura aplikacji

Grę można podzielić na dwa elementy. Tymi elementami są klienci oraz serwer na którym znajduje się aplikacja. W przypadku gry sieciowej klientem jest urządzenie obsługiwane przez użytkownika. Do połączenia z serwerem użytkownik musi posiadać dostęp do sieci internetowej oraz przeglądarkę. W trakcie łączenia urządzenia z serwerem klient wysyła zapytanie do serwera, następnie serwer zwraca komunikat o prawidłowym połączeniu. Stan przebiegu gry tj. aktualna pozycja graczy jest zapisywana na serwerze. Klienci informowani są o stanie rozgrywki otrzymując komunikaty z serwera. Jeśli w tym samym momencie do serwera podłączonych jest pięciu klientów, łączący się szósty gracz otrzyma dane z serwera o aktualnej pozycji innych graczy.

Poniższy rysunek obrazuje mechanizm działania klient-serwer.



Rys. 6.1. Schemat działania klient-server.

## 6.1. Klient

Klientem jest gracz podłączony do serwera. Wszystkie czynności, które podejmuje wykonywane są z poziomu przeglądarki internetowej.

Listing 6.1. przedstawia fragment kodu języka znaczników HTML sekcji head pliku index.html. Znajdują się w nim podstawowe informacje dotyczące wyświetlania aplikacji po stronie klienta, kodowanie strony utf-8, ponieważ aplikacja ma wyświetlać polskie znaki, autora, tytuł oraz ścieżki do plików zewnętrznych, z których korzysta aplikacja.

```
<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, in-
itial-scale=1, shrink-to-fit=no">

<meta name="description" content="Projekt na pracę in-
żynierską">

<meta name="author" content="Adrian Sokołowski">

<title>Praca inżynierska</title>

<link                href="./libs/bootstrap.min.css"
rel="stylesheet">

<link href="./main/css/lobby.css" rel="stylesheet">

<link                href="./main/css/logowanie.css"
rel="stylesheet">

<script src="./libs/chessboard-0.3.0.min.js"></script>

<link    rel="stylesheet"    href="./libs/chessboard-
0.3.0.min.css">

<script src="./libs/chess.js"></script>

</head>
```

Listing. 6.1 Sekcja head pliku index.html

W pliku index utworzone są dwa identyfikatory: logowanie, oraz lobby w celu identyfikacji poszczególnych znaczników przy pomocy JavaScript i CSS. Gdy gracz zaloguje się do systemu, wówczas znika zawartość umieszczona pomiędzy znacznikiem o id logowanie, a pojawia się zawartość umieszczona między znacznikiem o identyfikatorze lobby. Jest to korzystne rozwiązanie przy budowaniu aplikacji z wykorzystaniem socket.io ze względu na to, że aplikacja nie powinna przeładowywać całej strony. W języku HTML występują dwa sposoby odwoływania się do konkretnego znacznika, identyfikator oraz klasa. Różnicą jest to, że id jest unikalne i może być wykorzystane tylko raz. Na Listingu 6.2 pokazano wymienione znaczniki. Klasa text-center odwołuje się do pliku css biblioteki Bootstrap, której zadaniem jest wyśrodkowanie całego tekstu.

```
<div id="lobby" class="text-center">

. . .

</div>

<div id="logowanie"

. . .

</div>
```

Listing. 6.2. Identyfikatory zawartości, którą widzi użytkownik.

Całość kodu napisanego w lobby.js umieszczona jest w funkcji biblioteki jQuery przedstawionej na listingu 6.3 w celu upewnienia się, że kod zostanie wykonany dopiero po załadowaniu wszystkich elementów aplikacji.

```
$(function() {

. . .

});
```

Listing. 6.3. Uproszczona wersja funkcji document.ready biblioteki jQuery.

W celu uproszczenia pisania kodu na początku dokumentu zadeklarowano zmienne, które są wielokrotnie wykorzystywane w kodzie aplikacji m.in. funkcjach, czy klasach.

```

var socket = io();

var KOLORY = [
    '#007bff', '#6610f2', '#6f42c1', '#e83e8c',
    '#dc3545', '#fd7e14', '#ffc107', '#28a745',
    '#20c997', '#17a2b8', '#a700ff', '#d300e7'
];

var LICZNIK_PISANIA = 500;
var CZAS_ZANIKANIA = 200;
var $oknoPrzegladarki = $(window);
var $nazwa_uzytkownika = $('.nazwa_uzytkownika');
var $wiadomosci = $('.wiadomosci');
var $napiszWiadomosc = $('.napiszWiadomosc');
var $logowanieStrona = $('#logowanie');
var $lobbyStrona = $('#lobby');
var uzytkownik;
var polaczony = false;
var pisze = false;
var ostatniaWiadomosc;

var $aktywnePoleTekstowe = $nazwa_uzytkownika.focus();

```

Listing. 6.4. Deklaracja zmiennych w kliencie aplikacji.

Za każdym razem, gdy użytkownik połączy się z serwerem lista graczy jest przetwarzana na nowo, wysyłając komunikat z serwera do podłączonych klientów.

```

socket.on('update', function (gracze){
    userList = gracze;
    $('nav-link').empty();
    for(var i=0; i<userList.length; i++) {
        $('#gracze_online ul').attr('class', 'nav flex-column').append(
        $('<li>').attr('class', 'nav-item').append(
            $('<a>').attr({'class': 'nav-link', 'href': '#'}).append('<span data-feather="user"></span>' +
            ' ' + userList[i])
        ));
        feather.replace()
    });
});

```

Listing. 6.5. Metoda wyświetlania graczy dostępnych w aplikacji

```

const sendMessage = () => {
    var message = $napiszWiadomosc.val();
    message = cleanInput(message);
    if (message && polaczony) {
        $napiszWiadomosc.val('');
        addChatMessage({
            uzytkownik: uzytkownik,
            message: message
        });
        socket.emit('nowa wiadomosc', message);
    }
});

```

Listing. 6.6. Funkcja wysyłania wiadomości na czacie.



Kiedy gracz przeciągnie bierkę na szachownicy z pola A na pole B sprawdzane jest czy ruch jest dozwolony, a następnie wysyłane jest zdarzenie do serwera.

```
var onDrop = function (source, target) {
    removeGreySquares();

    var move = game.move({
        from: source,
        to: target,
        promotion: 'q'
    });

    if (game.game_over()) {
        state.innerHTML = 'GAME OVER';
        socket.emit('gameOver', roomId)
    }

    if (move === null) return 'snapback';
    else
        socket.emit('move', { move: move, board:
game.fen(), room: roomId });
};
```

Listing 6.7. Kod sprawdzający ruch na planszy.

## 6.2. Serwer

Serwer jest centrum aplikacji. Działa jako pośrednik między wszystkimi klientami. Zdarzenia emitowane przez klienta trafiają do serwera, a następnie do pozostałych klientów.

```
var express = require('express');
var app = express();
var path = require('path');
var server = require('http').createServer(app);
var io = require('socket.io')(server);
var port = process.env.PORT || 8080;

server.listen(port, () => {
  console.log('Serwer został uruchomiony na porcie %d',
port);
});
```

**Listing. 6.8.** Konfiguracja serwera NodeJS.

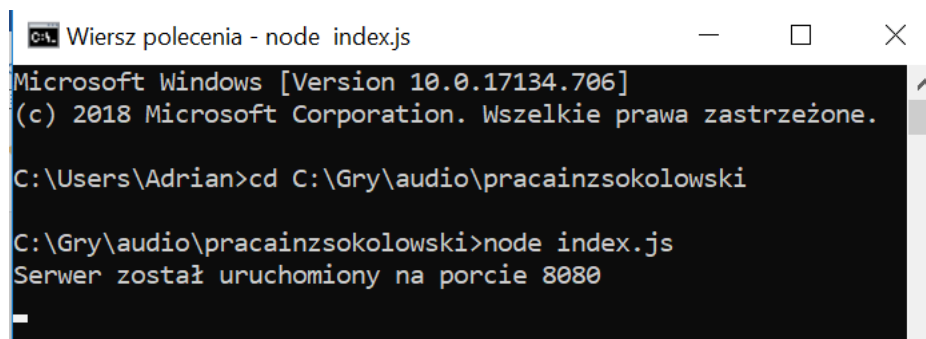
```
socket.on('dodaj gracza', (uzytkownik) => {
  if (graczDodany) return;

  socket.uzytkownik = uzytkownik;
  gracze.push(socket.uzytkownik);
  updateClients();
  ++iloscGraczy;
  graczDodany = true;
  socket.emit('login', {
    iloscGraczy: iloscGraczy
```

**Listing. 6.9.** Dodawanie gracza z poziomu serwera.

## 7. Opis stworzonej aplikacji

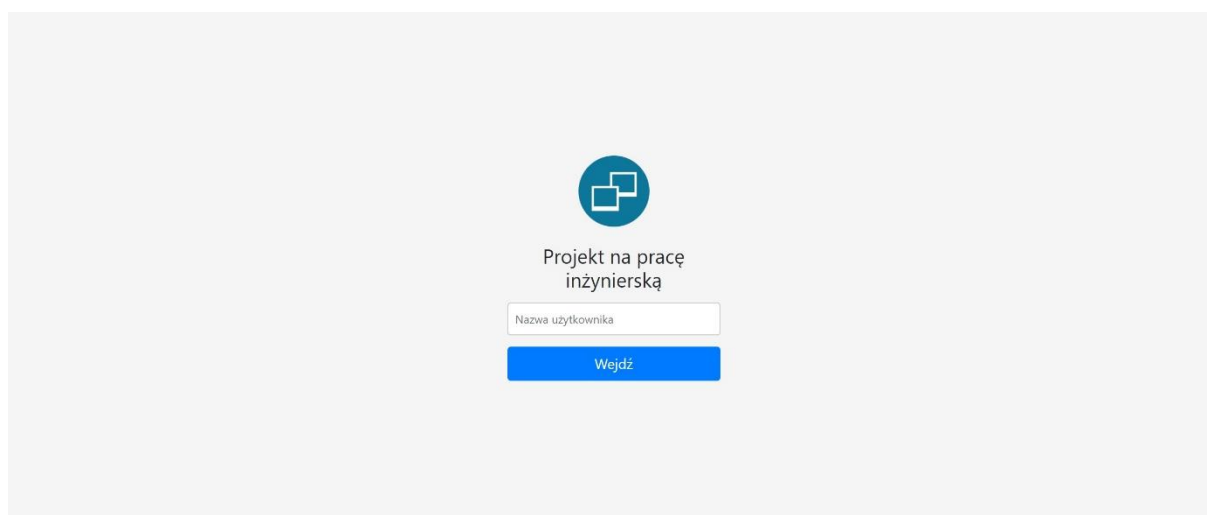
Stworzona aplikacja w ramach pracy inżynierskiej jest prostym odzwierciedleniem gry komputerowej z wykorzystaniem wyłącznie środowiska webowego. Do uruchomienia aplikacji potrzebny jest zainstalowany moduł NodeJS. Aplikację powinno uruchomić się na serwerze, aby mogły dołączyć inne osoby. Można ją także uruchomić na własnym komputerze jako localhost. Aplikację uruchamiamy bezpośrednio z wiersza polecenia komendą: `node server.js`



```
C:\Users\Adrian>cd C:\Gry\audio\pracainzsokolowski
C:\Gry\audio\pracainzsokolowski>node index.js
Serwer został uruchomiony na porcie 8080
```

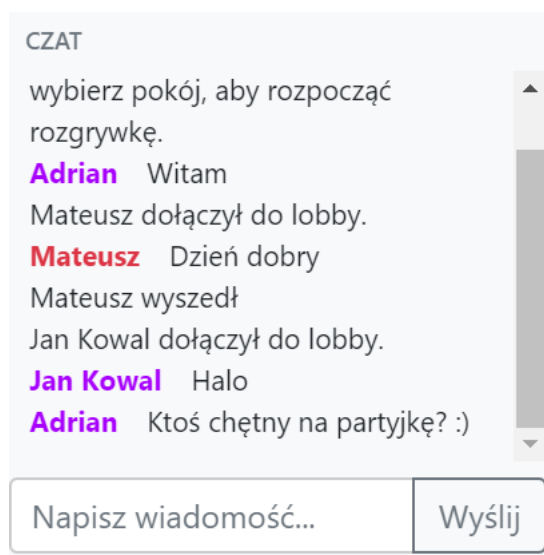
Rys. 7.1. Okno wiersza polecenia

Po wejściu w aplikację umieszczoną na serwerze NodeJS na ekranie pojawia się ekran startowy, na którym znajduje się logo aplikacji oraz input, służący do wpisania nazwy użytkownika, który posłuży jako identyfikator podłączonego urządzenia w obrębie aplikacji.



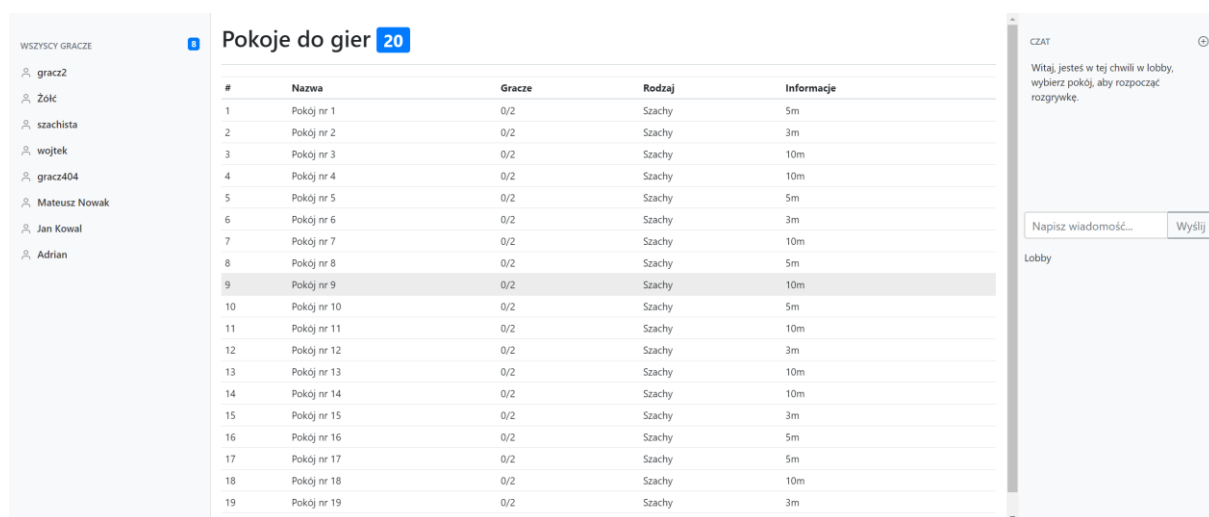
Rys. 7.2. Obrazek przedstawiający widok ekranu startowego aplikacji.

Gdy użytkownik wybierze swoją nazwę użytkownika przechodzi do lobby graczy, w którym znajduje się lista graczy podłączonych do aplikacji, pokoje do gier oraz czat globalny, w którym każdy może wysyłać między sobą wiadomości w formie tekstu. Przy liście wszystkich graczy widoczna jest liczba symbolizująca ich aktualną ilość na serwerze.

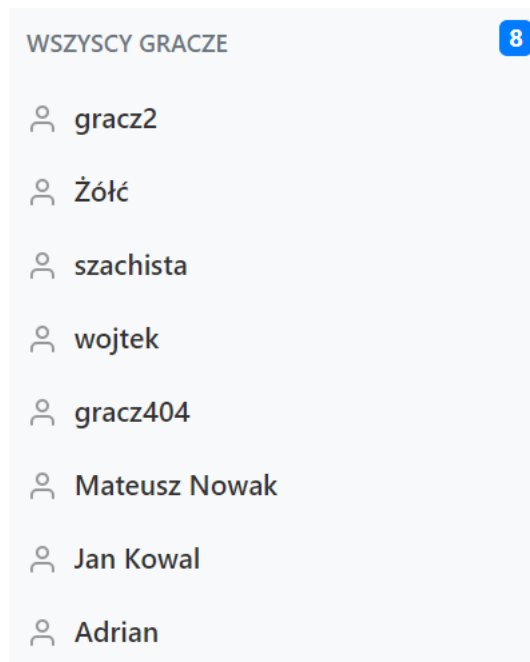


Rys. 7.3. Okno czatu.

Wiadomości docierają do wszystkich użytkowników w lobby. Scroll przewija się automatycznie po przesłaniu kolejnej wiadomości.

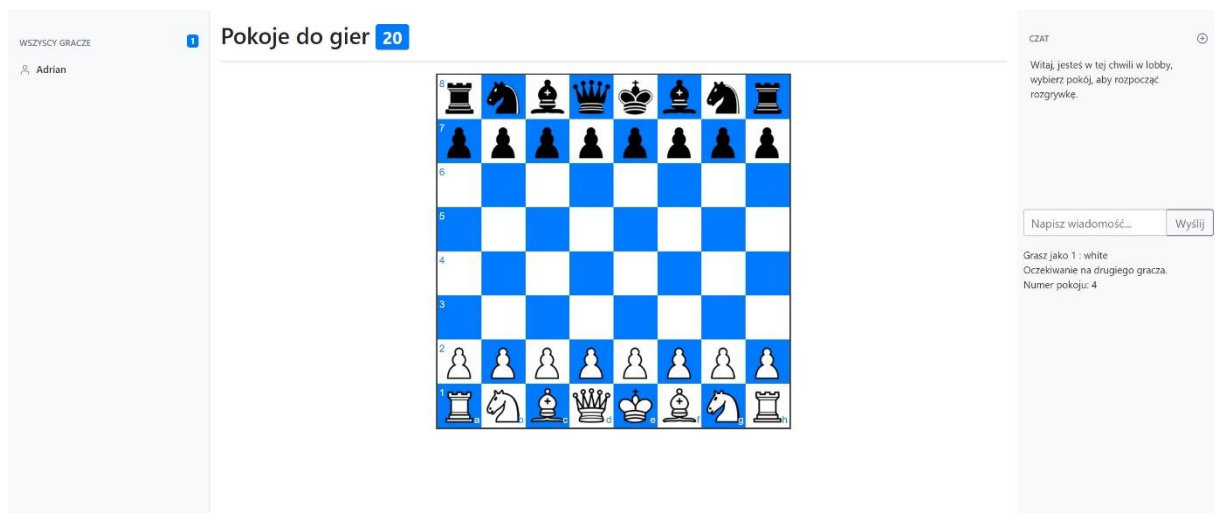


Rys. 7.4. Obrazek przedstawiający widok okna gry z połączonym wieloma klientami.



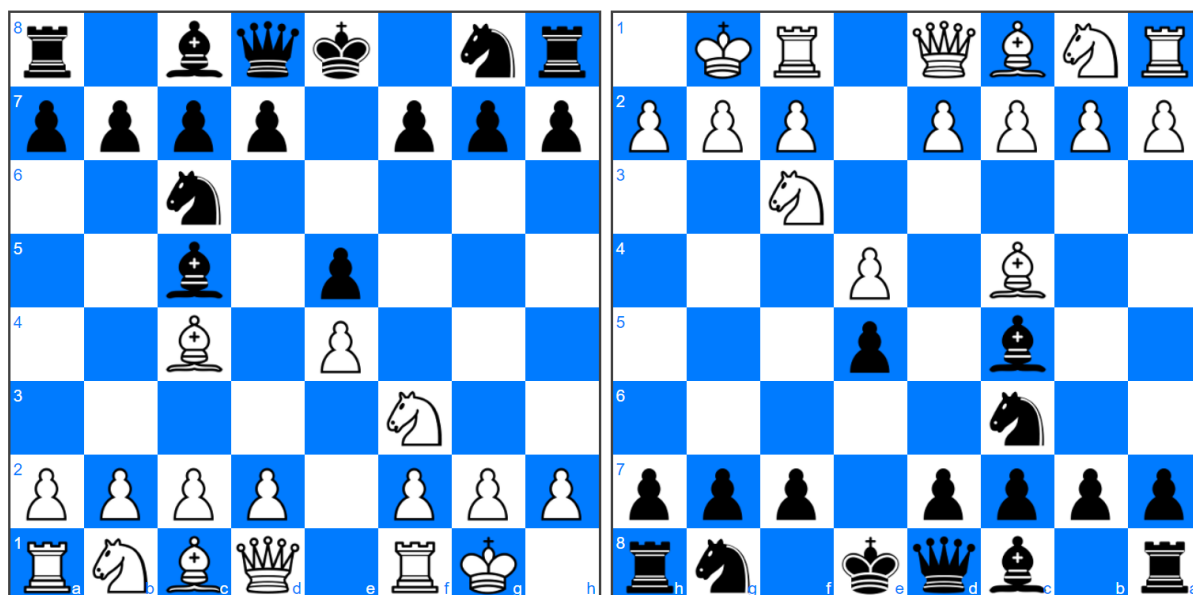
Rys. 7.5. Lista graczy.

Każdy z pokoi posiada szachownicę do gry w szachy do którego może dołączyć maksymalnie dwóch graczy, aby rozpocząć rozgrywkę. Każdy z graczy wykonuje ruch na przemian według zasad obowiązujących w szachach w czasie rzeczywistym.



Rys. 7.6. Obrazek przedstawiający widok okna rozgrywki w szachy, z poziomu jednego z klientów oczekującego na drugiego gracza.

Rozgrywka odbywa się w czasie rzeczywistym, gdy jeden z graczy wykona ruch, drugi natychmiast otrzymuje dane o wykonanym, dozwolonym posunięciu szachowym w formie graficznej na ekranie urządzenia oraz możliwość wykonania ruchu.



Rys. 7.7. Obrazek przedstawiający widok dwóch okien rozgrywki w szachy, z poziomu każdego z graczy znajdujących się w pokoju.

## 7. Podsumowanie

Celem pracy dyplomowej było przedstawienie zasad towarzyszących przy tworzeniu gry w środowisku aplikacji webowych oraz stworzenie prototypu gry wykorzystując wyłącznie technologie webowe. Korzystając z możliwości opisanych technologii udało mi się utworzyć grę sieciową w trybie rzeczywistym z systemem lobby, w którym znajduje się czat użytkowników, lista graczy oraz pokoje bez wykorzystywania zewnętrznych aplikacji renderujących w przeglądarce. Język skryptowy JavaScript, HTML i CSS oraz NodeJS jako serwer okazały się wystarczające do funkcjonowania aplikacji, która wykorzystuje wyłącznie przeglądarkę internetową i dostarczone do niej biblioteki. Podejmując się pisanie pracy na temat w planach miałem utworzenie gry, która może działać w obrębie przeglądarki, a użytkownik korzystający z niej nie musi posiadać dodatkowych technologii oprócz przeglądarki internetowej by móc z niej korzystać, a także instalować dodatkowych wtyczek czy pluginów lub pobierać czegoś na swoje urządzenie.

Praca dyplomowa była dla mnie okazją do zdobycia dodatkowego doświadczenia. Zanim jeszcze przystąpiłem do pisanie aplikacji w planach miałem utworzenie gry, w której wszyscy gracze wykonywaliby jakąś czynność jednocześnie na jednym ekranie, jednak później zdecydowałem się utworzyć system lobby, gdzie gracz mógłby sam zdecydować co chce zrobić oraz komunikować się z innymi poprzez czat. Więc utworzyłem ekran startowy, w którym gracz podaje swój identyfikator, a później może dzięki niemu wykonywać inne czynności na serwerze. Podczas tworzenia projektu natknąłem się na problemy. Zauważyłem, że najwięcej problemów miałem podczas synchronizacji klientów z serwerem. Gracz A widział zupełnie coś innego niż gracz B. Cel pracy został osiągnięty. Aplikacja ma potencjał w przyszłości, można między innymi rozbudować lobby poprzez dopisanie dodatkowych gier jak np. warcaby, gra w statki itd.

## Literatura

1. M. Pudełko, *Prawdziwa Historia Internetu*, ITStart 2017
2. B. Kluska, *Dawno temu w grach. Czas pionierów: szkice z historii gier komputerowych*, Orka 2008
3. C. Hudson, T. Leadbetter, *HTML5. Podręcznik programisty*, Helion 2013
4. E. Rowell, *HTML5 Canvas. Receptury*, Helion 2013
5. M. Cantelon, M. Harter, T. Holowaychuk, N. Rajlich, *Node.js w akcji*, Helion 2015
6. M. Haverbeke, *Zrozumieć JavaScript. Wprowadzenie do programowania*, Helion 2011
7. Socket.io, <https://socket.io>, stan z dnia 02.01.2019
8. T. Cadenhead, *Socket.io Cookbook*, PACKTPUB 2015
9. A. Mardan, *Express.js. Tworzenie aplikacji sieciowych w Node.js*, Helion 2016
10. Bootstrap, <https://getbootstrap.com>, stan z dnia 27.03.2019
11. Stackoverflow, <https://insights.stackoverflow.com/survey/2018/>, stan z dnia 15.02.2019
12. NodeJS, <https://nodejs.org/en/docs/>, stan z dnia 02.01.2019
13. JQuery, <https://api.jquery.com>, stan z dnia 04.03.2019

## Spis rysunków

**Rys. 5.1.** Dwuwymiarowa szachownica 8x8.

**Rys. 5.2.** Wizualizacja graficzna figury król.

**Rys. 5.3.** Schemat poruszania się króla po szachownicy.

**Rys. 5.4.** Sytuacja szachowa.

**Rys. 5.5.** Sytuacja matowa.

**Rys. 5.6.** Wizualizacja graficzna figury wieża.



**Rys. 5.7.** Wizualizacja graficzna bierki pionek.

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

**Rys. 5.1.** Wizualizacja graficzna

## **Spis tabel**

Tabela 4.1. Dziesięć najpopularniejszych języków programowania w 2018 roku według ankiety przeprowadzonej przez Stackoverflow.com

Tabela 4.2. Dziesięć najpopularniejszych języków programowania w 2018 roku według ankiety przeprowadzonej przez Stackoverflow.com

## **Spis załączników**

**Załącznik 1.** Kod źródłowy programu

**Załącznik 2.** Oświadczenie studenta