Poker Master Tool — Sieciowy kalkulator układów dla Texas Holdem

(Poker Master Tool — Web hand calculator for Poker Texas Holdem)

Bartosz Putek

Praca inżynierska

Promotor: dr hab. Artur Jeż

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

23 czerwca 2022

Streszczenie

Celem pracy jest stworzenie kalkulatora układów dla gry Poker Texas Holdem. Wyzwaniem jest przygotowanie szybko działającego rozwiązania w środowisku webowym. Aplikacja ma także na celu zaprojektowanie intuicyjnego interfejsu użytkownika, które zachęci graczy do poznawania teorii pokera. Praca zawiera wprowadzenie do gry poker Texas Holdem, wyjaśnia działanie ewaluatorów oraz kalkulatorów szans, opisuje implementację i architekturę rozwiązania oraz zawiera instrukcję użytkownika.

The purpose of the thesis is creation poker hand calculator for Texas Holdem. The challenge is preparing a fast solution in a web environment. Another goal is to encourage players to expand their knowledge and therefore the application should contain an intuitive user interface. The paper includes an introduction to the card game Poker Texas Holdem, explains the mechanism of poker evaluators and calculators, describes the implementation and architecture of the software with a user manual.

Spis treści

Rozdział 1.

Wstęp

1.1. Zawartość pracy

Pierwszy rozdział zawiera wprowadzenie wyjaśniające jaki problem rozwiązuje aplikacja *Poker Master Tool*. Drugi rozdział objaśnia zasady gry Poker Texas Hold'em, wyjaśnia działanie kalkulatorów szans oraz zawiera porównanie z innymi istniejącymi rozwiązaniami. Trzeci rozdział opisuje algorytm wykorzystany w silniku aplikacji. W czwartym rozdziale omówiono implementację rozwiązania. Piąty rozdział to instrukcja użytkownika z demonstracyjnymi przypadkami użycia. W szóstym rozdziale zostało zawarte podsumowanie.

1.2. Opis problemu i cel powstania aplikacji

Poker [1] to jedna z najpopularniejszych gier karcianych na świecie. Codziennie grają w niego miliony graczy online oraz na żywo. Kluczowymi umiejętnościami jest dobór odpowiedniej strategii, znajomość elementów rachunku prawdopodobieństwa oraz aspekty psychologiczne. Aby udoskonalać swoją strategię, profesjonalni pokerzyści korzystają z szerokiej gamy narzędzi. Jedną z kategorii takich aplikacji są kalkulatory szans (ang. equity calculator), które służą do obliczenia szans wygranej dla poszczególnych graczy przy zadanych kartach graczy oraz kartach wspólnych. Dzięki temu gracze mogą ocenić siłę swojej ręki co ułatwia im podejmowanie decyzji.

Niestety, większość kalkulatorów obecnych na rynku jest nieprzyjazna dla amatorów pokera — zawierają zbyt dużo skomplikowanych opcji, a ich interfejs jest nieczytelny, przez co próg wejścia jest wysoki, jeżeli gracz zechciałby zacząć ich używać. Większość kalkulatorów jest dostępna tylko jako aplikacja desktopowa, a inne w formie aplikacji webowej (sieciowej) są nieprzystosowane do szerokości ekranów urządzeń mobilnych.

Poker Master Tool stara się rozwiązać ten problem — jest to w pełni respon-

sywna aplikacja webowa, z interfejsem dostosowanym do współczesnych standardów, oferująca kalkulację szans na wygraną i szans uzyskania poszczególnych układów. Dzięki prostocie obsługi kalkulatora, mniej doświadczeni gracze mają szansę z zaznajomieniem się z teorią pokera.

Aplikacja dostępna jest pod adresem https://pokermastertool.bartoszputek.pl/, a kod źródłowy na repozytorium https://github.com/bartoszputek/poker-mastertool/.

Rozdział 2.

Poker oraz kalkulatory szans

2.1. Zasady gry Poker Texas Hold'em

Poker Texas Hold'em [2] to najpopularniejsza odmiana pokera. W standardowej rozgrywce uczestniczy od 2 graczy do 9 graczy oraz używa się pełnej talii 52 kart. Celem gry jest wygranie żetonów od innych graczy poprzez skompletowanie silniejszego układu bądź wymuszenie spasowania kart od pozostałych graczy. Każdy z graczy dysponuje dwiema kartami oraz kartami wspólnymi ujawnianymi w kolejnych rundach licytacji. Gracze w każdej turze licytacji mają do dyspozycji następujące możliwości:

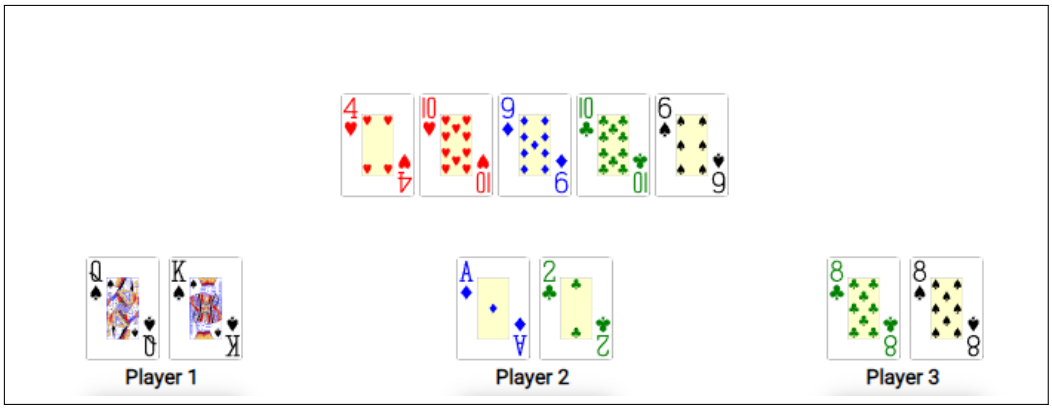
- Spasowanie kart (fold)
- Sprawdzenie (call)
- Podbicie stawki (raise)
- Czekanie (check)

Gracz może czekać tylko, gdy w obecnej turze licytacji nikt wcześniej nie podbił stawki (raise). Pojedyncze rozdanie składa się z co najwyżej 4 tur licytacji, po których dochodzi do ujawnienia kolejno: trzech kart wspólnych (flop), jednej karty wspólnej (turn) i ostatniej kart wspólnej (river). Po ostatniej turze licytacji odbywa się określenie siły rąk wszystkich graczy pozostałych w rozdaniu.

Ewaluacja ręki polega na określeniu układu z poniższej listy rankingowej. Każdy z graczy do określenia swojego układu wybiera pięć kart spośród dwóch kart własnych i pięciu wspólnych.

Układy kart od najsłabszego:

Wysoka karta (High Card): Następuje w przypadku gdy układ nie kwalifikuje się pod żaden z poniższych. Gracz wybiera 5 najwyższych dostępnych dla siebie



Rysunek 2.1: Przykładowy stan gry w trakcie ostatniej tury licytacji (river)

kart. Jeżeli obydwoje z graczy mają wysoką kartę należy porównać kolejno najwyższe karty. W przypadku identycznego układu następuje remis. Podobną zasadę (kicker) stosuje się w przypadku remisów w innych układach, które należy uzupełnić do 5-kartowego układu. (np. $A\clubsuit, T\diamondsuit, 7\spadesuit, 5\heartsuit, 2\heartsuit$)

Para (One pair): dwie karty o tej samej wartości; jeżeli gracze mają tą samą parę, decyduje kicker. (np. $A\clubsuit, A\diamondsuit, J\diamondsuit, 9\spadesuit, 2\spadesuit$)

Dwie pary (Two pairs): Wygrywa gracz z wyższą parą; jeżeli najwyższa para jest identyczna, to wygrywa gracz z drugą najwyższą parą. Gdy druga para jest identyczna — decyduje kicker. (np. $K\clubsuit, K\diamondsuit, T\spadesuit, T\heartsuit, 4\heartsuit$)

Trójka (Three of a Kind): Wygrywa gracz z trójką o wyższej randze; jeżeli ranga jest ta sama, to decyduje kicker. (np. $K\clubsuit, K\diamondsuit, K\spadesuit, 7\heartsuit, 4\heartsuit$)

Strit (Straight): pięć kolejnych kart; jeśli dwóch graczy ma strita to wygrywa ten, który ma wyższą kartę. W stricie as może być zarówno najwyższą jak i najniższą kartą — jedynką. (np. $A\spadesuit, K\diamondsuit, Q\clubsuit, J\spadesuit, T\heartsuit$)

Kolor (Flush): pięć kart w tym samym kolorze; jeśli dwóch graczy ma kolor (uzyskanie dwóch różnych kolorów jest niemożliwe) wygrywa ten, którego kolor ma najwyższą kartę. (np. $K\spadesuit, T\spadesuit, J\spadesuit, 8\spadesuit, 7\spadesuit$)

Full (Full house): układ składający się z trójki oraz pary; jeśli dwóch graczy ma fulla, to wygrywa ten, który ma wyższą trójkę; jeśli obydwaj mają tę samą trójkę, wygrywa ten, który ma wyższą parę; jeśli i para jest taka sama, to następuje remis. (np. $K\clubsuit, K\diamondsuit, K\spadesuit, 4\heartsuit, 4\diamondsuit$)

Kareta (Quads): cztery karty tej samej rangi; jeżeli obydwoje z graczy mają karetę, wygrywa ten składający się z wyższej rangi. (np. $K\clubsuit$, $K\diamondsuit$, $K\spadesuit$, $K\heartsuit, 2\diamondsuit$)

Poker (Straight flush): pięć kolejnych kart w tym samym kolorze; jeżeli obydwoje z graczy mają pokera, wygrywa ten z ostatnią wyższą kartą. (np. $Q\heartsuit, J\heartsuit$, $T\heartsuit, 9\heartsuit, 8\heartsuit$)

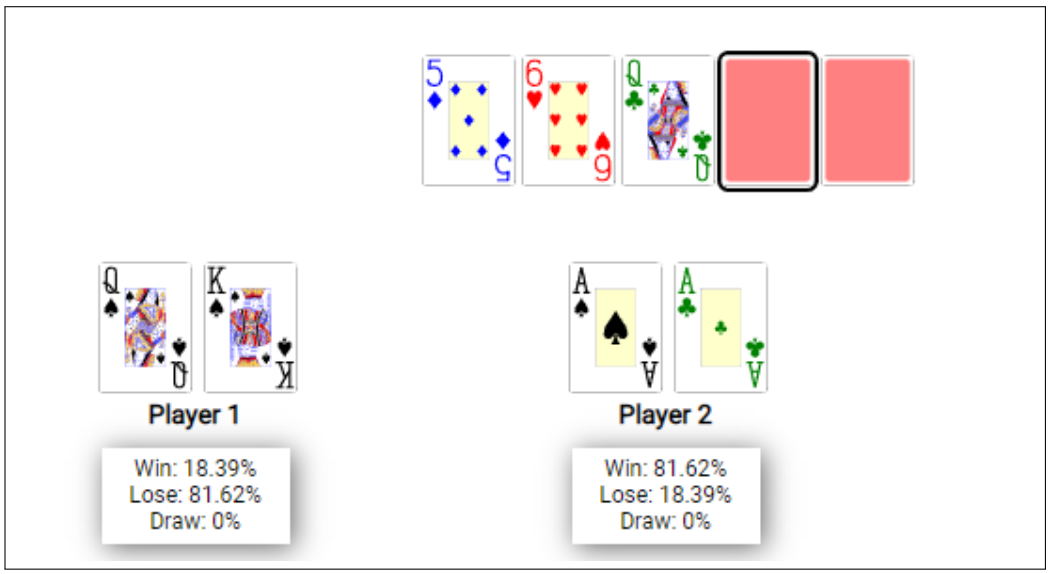
W scenariuszu przedstawionym na rysunku 2.1 gracze dysponują następującymi układami:

- Player 1: Para dziesiątek — $T\clubsuit, T\heartsuit, K\spadesuit, Q\spadesuit, 9\diamondsuit$
- Player 2: Para dziesiątek — $T\clubsuit, T\heartsuit, A\diamondsuit, 9\diamondsuit, 6\spadesuit$
- Player 3: Dwie pary — $T\clubsuit, T\heartsuit, 8\clubsuit, 8\spadesuit, 9\diamondsuit$

Zatem gracz trzeci dysponuje najlepszym układem, drugi po nim jest gracz numer dwa, a najgorszym układem dysponuje gracz numer jeden (gracz numer dwa posiada wyższego kickera — $A\diamondsuit > K\spadesuit$).

2.2. Działanie kalkulatora szans

Zadaniem kalkulatorów szans (ang. equity calculator) jest obliczenie szansy na wygraną każdego z graczy przy wybranych przez użytkownika kartach graczy oraz kartach wspólnych — taką sytuację dalej nazwiemy *scenariuszem*. Zatem zakładamy, że każdy gracz uczestniczący w rozdaniu będzie podejmował tylko jedną akcję — check. Wyniki z kalkulatora przydają się, do określenia rzeczywistej siły ręki przeciwko innym rękom np. przy zagraniu za wszystko (all-in) w sytuacji $Q\spadesuit, K\spadesuit$ vs $A\spadesuit, A\clubsuit$ z kartami wspólnymi $5\diamondsuit, 6\heartsuit, Q\clubsuit$ gracz pierwszy ma tylko 18% szans na wygraną.



Rysunek 2.2: Wizualizacja powyższego scenariusza

Kalkulatory szans bazują na podobnej zasadzie. Należy określić, jakie karty są dostępne w talii, a następnie rozpatrzyć wszystkie możliwe kombinacje i w każdej z nich określić zwycięzcę. W scenariuszu przedstawionym na rysunku 2.2 w talii pozostało 45 kart (52 minus po dwie karty każdego z graczy i trzy na stole). Musimy

wybrać dwie z nich zatem liczba kombinacja bez powtórzeń to $\binom{45}{2}$ = 990. Dla tak prostego scenariusza problem jest łatwy, natomiast w pesymistycznym i najczęstszym przypadku użycia, gdy kart w talii jest najwięcej oraz na stole nie ma żadnych kart, liczba kombinacji drastycznie rośnie, nawet do $\binom{48}{5}$ = 1712304 kombinacji.

Naiwne podejście do tego problemu jest naturalne. W każdej kombinacji można posortować karty każdego z graczy, przyjrzeć się pojedynczo każdej z kart i zebrać w ten sposób potrzebne dane (np. zliczać kolory, pary) do ustalenia układu. Ten sposób ma jedną poważną wadę - jest zdecydowanie zbyt wolny do zastosowania w scenariuszach z wieloma kombinacjami.

Wszystkie szybkie kalkulatory szans wykorzystują zapamiętane wcześniej wyniki przez naiwny algorytm lub korzystają z metody Monte Carlo [3] [4] do przybliżania wyników przy dostatecznie małym błędzie. Aplikacja *Poker Master Tool* oblicza wynik dokładnie, zatem będziemy się zajmować tylko tą metodą.

2.3. Porównanie z innymi dostępnymi kalkulatorami

2.3.1. Poker Master Tool

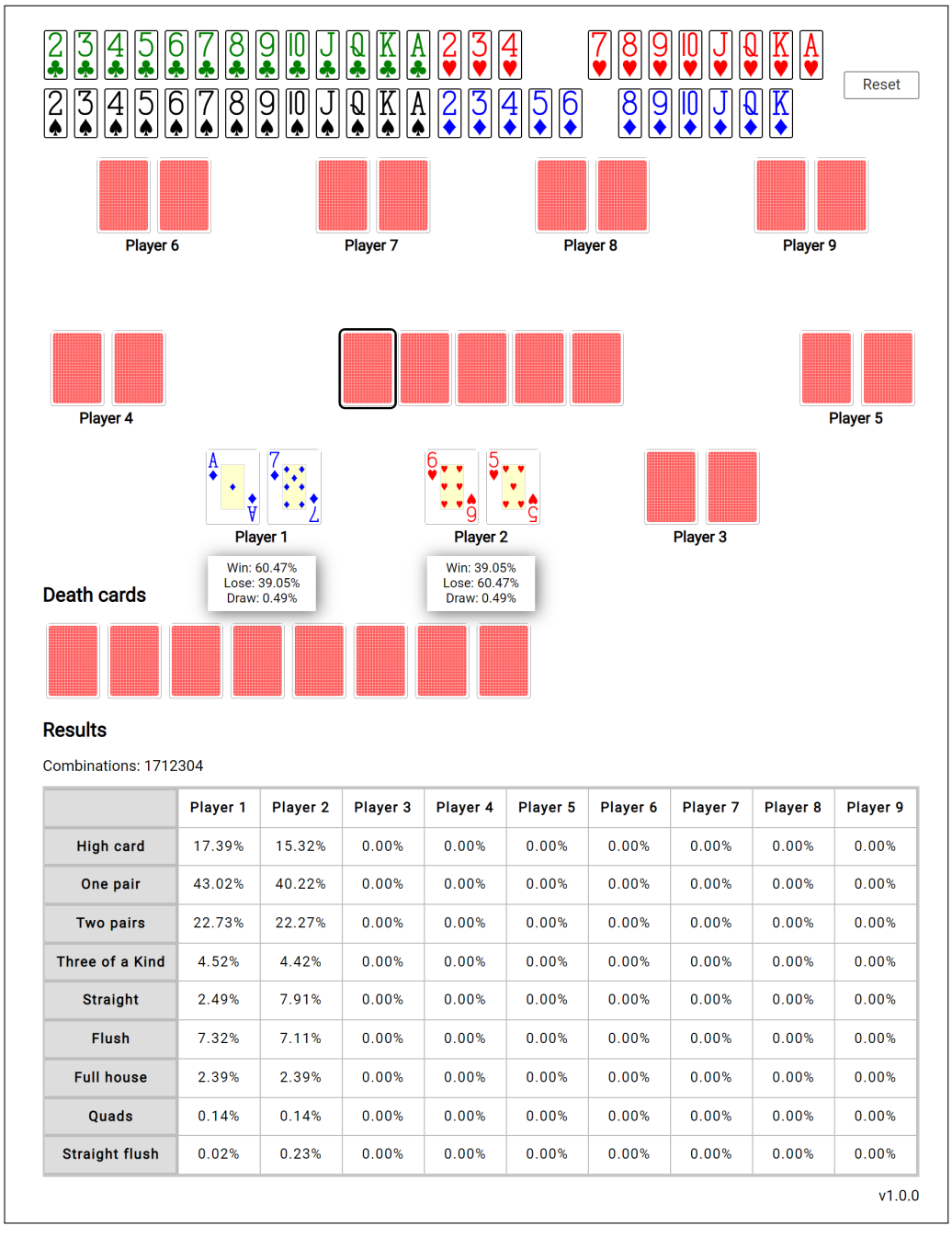
Poker Master Tool jest aplikacją internetową opartą o architekturę klient-serwer. Klient z poziomu przeglądarki w wygodny sposób wybiera interesujący go scenariusz, a następnie informacje są wysyłane do serwera, który dane przetwarza i zwraca wyświetlaną w widoku użytkownika odpowiedź. Z tego powodu do korzystania z aplikacji niezbędne jest połączenie internetowe.

Użytkownik za pomocą myszy lub klawiatury jest w stanie wybrać karty dla każdego z graczy, karty wspólne oraz tak zwane karty martwe, wykluczone z pozostałej części talii. Informacje jakie dostajemy w zwrotnej odpowiedzi to:

- Szanse na wygraną, przegraną oraz remis każdego z graczy
- Szanse na poszczególne układy każdego z graczy
- Liczbę kombinacji w danym scenariuszu

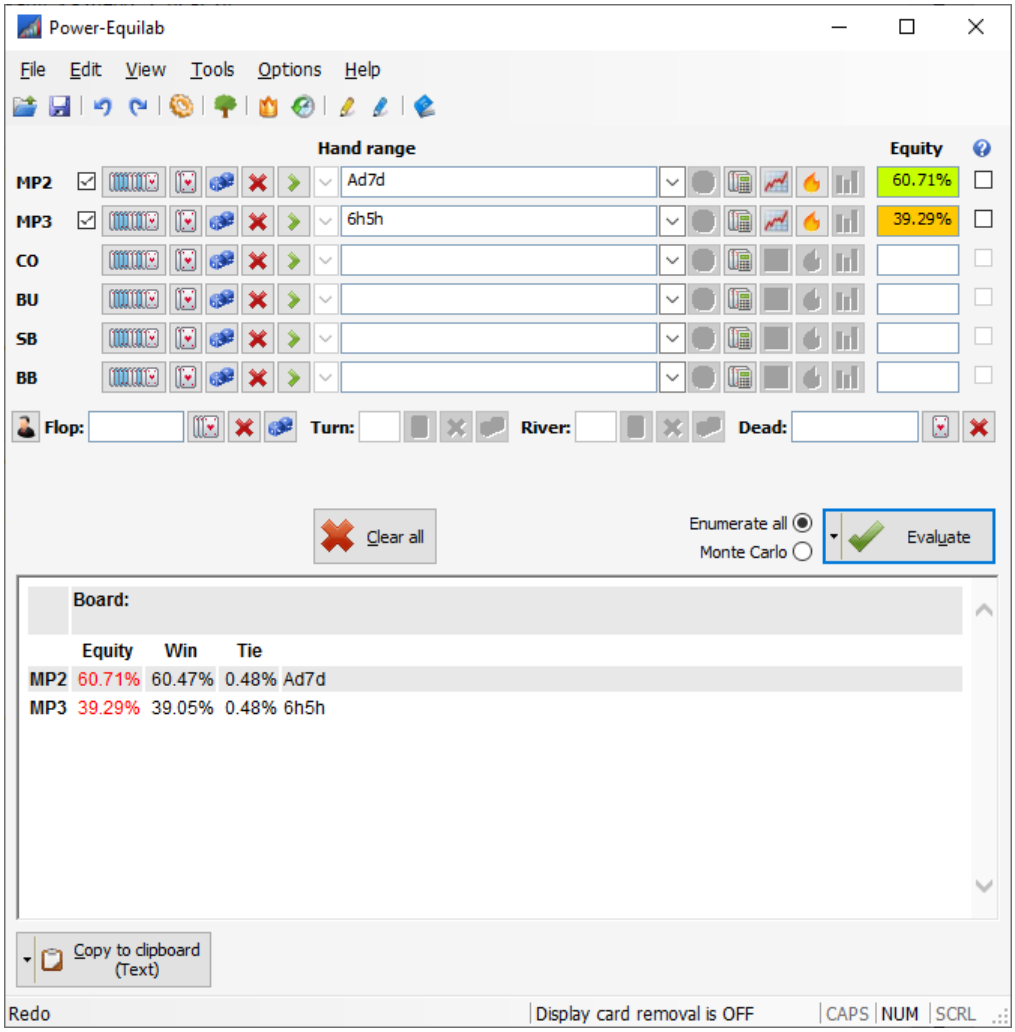
2.3.2. Power Equilab

Power Equilab jest płatnym oprogramowaniem przeznaczonym dla profesjonalistów w formie aplikacji desktopowej. Nie posiada wersji na telefony, za to w odróżnieniu od aplikacji webowej nie potrzebuje do działania połączenia z internetem. Jedną z jego funkcjonalności jest kalkulator szans. Kalkulator posiada wiele rozbudowanych opcji, które mogą być niejasne z perspektywy początkującego gracza, lecz bardzo przydatne dla graczy zaawansowanych. Interfejs aplikacji niestety nie pozwala komfortowo korzystać z klawiatury.



	Player 1	Player 2	Player 3	Player 4	Player 5	Player 6	Player 7	Player 8	Player 9
High card	17.39%	15.32%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
One pair	43.02%	40.22%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Two pairs	22.73%	22.27%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Three of a Kind	4.52%	4.42%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Straight	2.49%	7.91%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Flush	7.32%	7.11%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Full house	2.39%	2.39%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Quads	0.14%	0.14%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Straight flush	0.02%	0.23%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

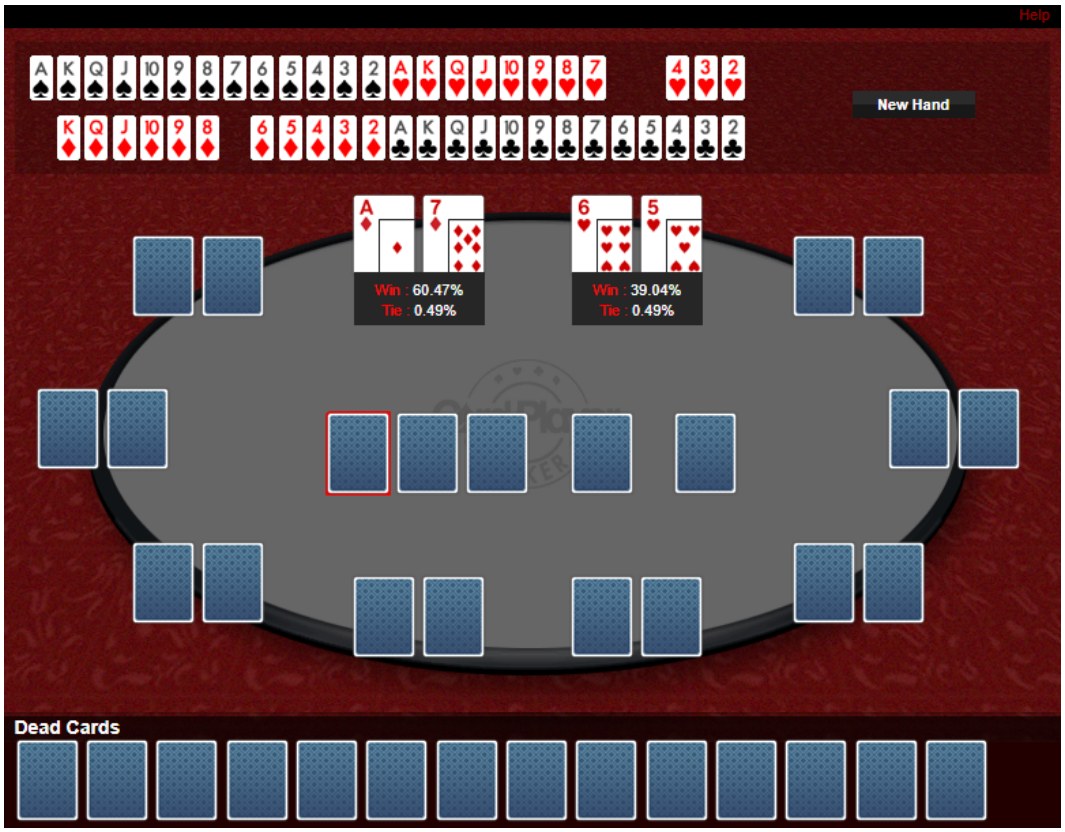
Rysunek 2.3: Standardowy widok aplikacji Poker Master Tool



Rysunek 2.4: Standardowy widok aplikacji Power Equilab

2.3.3. Card Player Poker Odds Calculator

Card Player Poker Odds Calculator jest webowym kalkulatorem szans. Jego główną zaletą jest prostota oraz brak konieczności instalacji oprogramowania na własnym komputerze. Niestety, mimo architektury webowej trudno z niego korzystać na urządzeniach mobilnych. Nie posiada żadnych zaawansowanych funkcji oraz wynik zawiera tylko kalkulację szansy na zwycięstwo dla każdego z graczy. Interfejs nie wspiera obsługi klawiatury, a aplikacja wymaga stałego połączenia internetowego, podobnie jak w przypadku *Poker Master Tool*.



Rysunek 2.5: Standardowy widok aplikacji Card Player Poker Odds Calculator

	Poker Master Tool	Power Equilab	Card Player Calc.
Typ aplikacji	Webowa	Desktopowa	Webowa
Połączenie internetowe	Tak	Nie	Tak
Obsługa klawiatury	Tak	Nie	Nie
Liczba funkcji	Średnia	Duża	Niska
Urządzenia mobilne	Tak	Nie	Nie
Przeznaczenie	Początkujący	Profesjonaliści	Początkujący
Model	Darmowy	Płatny	Darmowy

Tablica 2.1: Podsumowanie prezentowanych kalkulatorów

Rozdział 3.

Ewaluacja układów

3.1. Wprowadzenie

Podstawą działania kalkulatorów szans jest algorytm odpowiedzialny za określanie siły ręki. W przypadku odmiany Poker Texas Hold'em należy określić ją na podstawie 7 kart - 2 gracza oraz 5 wspólnych. Program, który realizuje powyższe zadanie nazwiemy ewaluatorem. Szybkość ewaluacji jest najistotniejsza z punktu widzenia naszej aplikacji, ponieważ to właśnie ten proces zajmuje zdecydowaną większość czasu odpowiedzi serwera.

3.2. Ewaluator Cactus Kev's

Kevin Suffecool, autor ewaluatora *Cactus Kev's* [5] [6] zauważył, że 5-kartowe kombinacje, których jest $\binom{52}{5} = 2598960$ można pogrupować w według ich siły. Na przykład układ $A\clubsuit, K\clubsuit, Q\clubsuit, J\clubsuit, T\clubsuit$ jest identyczny pod względem siły z $A\heartsuit, K\heartsuit$, $Q\heartsuit, J\heartsuit, T\heartsuit$, gdyż każdy z układów to najwyższy możliwy do uzyskania poker. Grupując w taki sposób wszystkie możliwe kombinacje uzyskujemy tylko 7462 porównywalnych wartości. Grupy te nazwiemy *klasami równoważności*.

Ewaluator *Cactus Kev's* przyjmując 5 kart zwraca liczbę od 1 do 7462, reprezentującą układ, gdzie 1 oznacza najsilniejszy układ, a 7462 najsłabszy.

Pojedyncza karta reprezentowana jest jako 32 bitowa liczba:

-	-	-	B	B	B	B	B	B	B	B	B	C	D	H	S	R	R	R	R	X	X	P	P	P	P	P	P

gdzie:

- Pierwsze 6 bitów P reprezentują rangę karty jako liczba pierwsza (dwójka - 2, trójka - 3, czwórka - 5, ... , as - 41)
- Cztery bity R reprezentują rangę karty jako kolejna liczba naturalna (dwójka

Układ	Unikalne	Zredukowane
Poker	40	10
Kareta	624	156
Full	3744	156
Kolor	5108	1277
Strit	10200	10
Trójka	54912	858
Dwie pary	123552	858
Para	1098240	2860
Wysoka karta	1302540	1277
Razem	2598960	7462

Tablica 3.1: Redukcja 5-kartowych kombinacji według ich siły

- 1, trójka - 2, czwórka - 3, ... , as - 12), nie są one wykorzystywane przez ewaluator w żaden sposób

- Jeden z zapalonych bitów CDHS oznacza kolor karty (C - trefl, D - karo, H - kier, S - pik)

- Dwanaście bitów B reprezentuję rangę karty jako kolejny zapalony bit (pierwszy zapalony bit to dwójka, drugi to trójka itd.)

Karty otrzymane na wejściu w powyższym formacie oznaczmy kolejno jako $C1$, $C2$, $C3$, $C4$, $C5$. W pierwszej kolejności ewaluator sprawdza, czy wszystkie karty są tego samego koloru korzystając z poniższej formuły:

```
isSuited = C1 AND C2 AND C3 AND C4 AND C5 AND 0xF000
```

Jeżeli wartość `isSuited` nie jest równa zero, to wszystkie karty są tego samego koloru. W przypadku gdy układ jest kolorem lub pokerem obliczenie wartości polega na zajrzenie do tablicy `flushes`, w której indeksem jest wyrażenie `index`. Na przykład wartość przechowywana w tablicy `flushes[0x1F00]` jest równa 1, gdyż 1 jest *klasą równoważności* odpowiadającą asowemu pokerowi.

```
index = (C1 OR C2 OR C3 OR C4 OR C5) >> 16
```

Zauważmy, że najmniejsza możliwa wartość wyrażenia `index` to `0x001F` (31), a największa `0x1F00` (7936). Tablica `flushes` ma zatem długość 7936 i została wypełniona obliczonymi wcześniej *klasami równoważności* za pomocą naiwnego algorytmu. Większość komórek w naszej tablicy to puste wartości, gdyż możliwych kolorów i pokerów jest $\binom{13}{5} = 1287$. W technice tablicowania [8] zyskujemy szybkość działania, w zamian za niewykorzystaną pamięć.

W przypadku, gdy wszystkie karty nie są tego samego koloru, możemy użyć tej samej techniki do sprawdzenia układów zawierające same unikalne karty — Strit oraz Wysoka karta. Została skonstruowana tablica `unique5` zawierająca porównywalne wartości dla Strita oraz Wysokiej karty. Jej długość oraz indeksy pod którymi znajdują się niezerowe wartości są identyczne z tablicą `flushes`, ponieważ układy poker — strit, kolor — wysoka karta różnią się tylko obecnością koloru. Indeksem w tej tablicy jest wartość wyrażenia `index`. Oznacza to, że pod indeksem `flushes[0x1F00]` znajduje się wartość 1600, która oznacza asowego strita.

Pozostają nam układy, w których występuje co najmniej jedno powtórzenie karty o tej samej randze. Do obliczenia takich układów posłużą nam bity P, reprezentujące poszczególne rangi jako liczby pierwsze. Aby uniknąć kosztownego sortowania kart, ewaluator korzysta z Podstawowego twierdzenia arytmetyki [7] oraz przemienności mnożenia.

```
product = (C1 AND 0xFF) * (C2 AND 0xFF) * (C3 AND 0xFF) *
        (C4 AND 0xFF) * (C5 AND 0xFF)
```

Największa wartość wyrażenia `product` to $41^4 \cdot 37 = 104553157$, zatem skonstruowanie tak dużej tablicy będzie wymagało za dużo pamięci. Autor rozwiązania obliczył wszystkie iloczyny liczb pierwszych dla pozostałych 4888 układów (7462-2574), posortował je oraz zapisał w tablicy `products`. Następnie jest wyszukiwana binarnie [9] wartość `product` w tablicy, a otrzymany w ten sposób indeks posłuży nam jako indeks w tablicy `value`, w której znajdują się wcześniej obliczone przez naiwny algorytm *klasy równoważności*.

3.3. Poprawa wydajności Paula Senzee'a

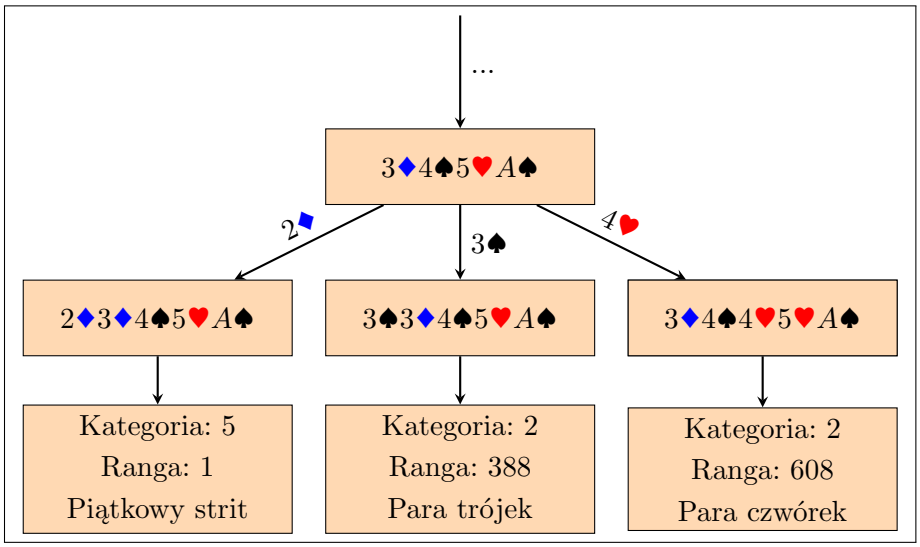
Paul Senzee [10] zauważył, że ostatni krok algorytmu jest najbardziej czasochłonny oraz większość rąk wymaga jego wykonania. Zastąpił on zatem wyszukiwanie binarne korzystając z doskonałej minimalnej funkcji haszującej [11]. Skonstruował ją za pomocą programu do generowania tablic haszujących autorstwa Boba Jenkinsa [12]. Dodatkowo, w sposób naiwny dodał możliwość ewaluacji 6 i 7 kartowych układów wywołując obliczenia dla każdej możliwej kombinacji i zwrócenie maksymalnej wartości.

3.4. Ewaluator TwoPlusTwo

W 2006 roku na jednym z najpopularniejszych forów pokerowych *TwoPlusTwo* [13] [14] została rozpoczęta praca nad *ewaluatorem* przeznaczonym do sekwencyjnego obliczania siły układów. Ideą jest wykorzystanie wcześniej obliczonych częścio-

wych wyników, aby jeszcze bardziej przyspieszyć proces ewaluacji dla dużej liczby kombinacji.

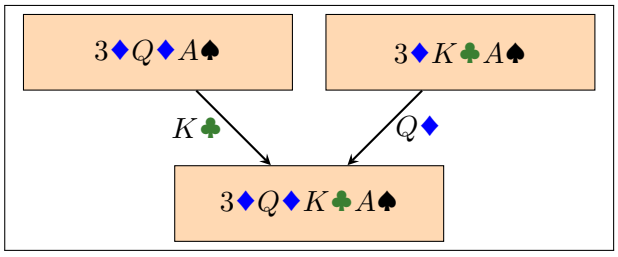
Ewaluator *TwoPlusTwo* skupia się na konstrukcji automatu skończonego [15], w którym kolejne przejścia wyznaczane są na podstawie kolejnych kart. Od piątego wierzchołka możliwe jest zwrócenie typu układu razem z jego miejscem w rankingu względem tych samych układów. Na przykład dla asowego strita zwrócona zostanie kategoria 5, gdyż strit jest 5 układem licząc od najsłabszego oraz ranga 10, gdyż jest najwyższym stritem, a rozróżnialnych stritów jest 10. Podczas konstruowania grafu wykorzystywany jest ewaluator Cactus Kev razem z poprawkami Paul'a Senzee do obliczania siły układów .



Rysunek 3.1: Przykładowa część grafu

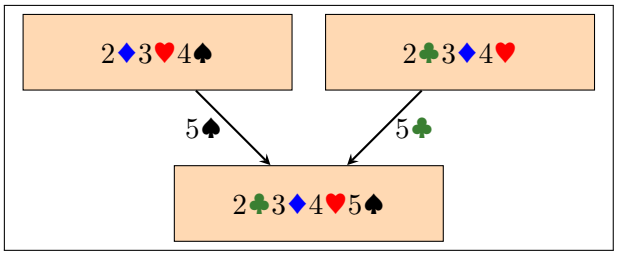
Tworzenie grafu wykorzystuje dwie poniższe techniki, redukujące znacząco liczbę wierzchołków:

- Te same karty ułożone w dowolnej kolejności wskazują na ten sam wierzchołek (rysunek 3.2)



Rysunek 3.2: Te same karty w różnej kolejności wskazują na ten sam wierzchołek

- Aby kolor był znaczący $n-2$ kart musi być tego samego koloru, gdzie n to długość układu. Gdy kolor jest nieznaczący następuje redukcja wierzchołków, ignorując kolor we wszystkich kartach.



Rysunek 3.3: Redukcja wierzchołków po wykryciu nieznaczącego koloru. Dla 4 kartowego układu, aby kolor był znaczący przynajmniej dwie karty powinny mieć ten sam kolor.

Aplikując powyższe zasady, graf zawiera 32487834 wierzchołków, przez co jego rozmiar na dysku to około 123 MB, w zamian uzyskując bardzo szybką ewaluację układów. Korzystając z ewaluatora *TwoPlusTwo* wystarczy tylko 7 odczytów z naszego grafu, aby uzyskać siłę ręki dla 7 kartowego układu.

W poniższych pseudokodach wykorzystane są następujące oznaczenia zmiennych:

Cards jest zmienną zawierającą karty, dla których chcemy uzyskać siłę układu.

HR to automat skończony reprezentowany jako tablica.

Pointer oznacza zmienną, która wskazuje na aktualny element w grafie.

Value jest obliczoną siłą naszego układu.

```
1  int LookupHand(int* cards)
2  {
3      int pointer = HR[53 + *cards++];
4      pointer = HR[pointer + *cards++];
5      pointer = HR[pointer + *cards++];
6      pointer = HR[pointer + *cards++];
7      pointer = HR[pointer + *cards++];
8      pointer = HR[pointer + *cards++];
9
10     int value = HR[pointer + *playerCards++];
11
12     return value;
13 }
```

Dodatkowo sekwencyjne przeszukiwanie rąk jest bardzo szybkie wykorzystując wcześniej otrzymane wskaźniki. Na przykład posiadając karty $3\heartsuit, T\spadesuit, Q\clubsuit, A\diamondsuit$ chcemy obliczyć wszystkie możliwe 5-kartowe uzupełnienia . Obliczamy zatem wskaźnik do wcześniej wymienionego układu, a następnie przechodzimy po wszystkich pozostałych kartach w talii, wykorzystując ten wskaźnik:

```
1    int pointer = HR[53 + *cards++];
2    pointer = HR[pointer + *cards++];
3    pointer = HR[pointer + *cards++];
4    pointer = HR[pointer + *cards++];
5
6    for(int i=0; i<deck.length(); i++){
7        second_pointer = HR[pointer + i];
8        value = HR[second_pointer];
9    }
```

W powyższym przykładzie wykonując $4 + (52 - 4) \cdot 2 = 100$ operacji odczytu z tablicy, której złożoność jest równa $O(1)$ otrzymaliśmy informację o sile 48 5-kartowych układów. Im większa jest liczba kart, do których chcemy uzupełnić układ tym więcej powyższa optymalizacja zyskuje na tle innych ewaluatorów, gdzie nie jesteśmy w stanie skorzystać z częściowo wcześniej obliczonych wyników. Technika ta jest podstawą działania silnika gry aplikacji *Poker Master Tool*.

Rozdział 4.

Opis Implementacji

4.1. Architektura rozwiązania

Poker Master Tool składa się z trzech odrębnych modułów:

- Backend — część serwerowa odpowiedzialna obsługę żądań HTTP
- Silnik gry — moduł służący do obliczeń układów z wykorzystaniem ewaluatora *TwoPlusTwo*
- Frontend — widok użytkownika aplikacji webowej

Kod źródłowy aplikacji dostępny jest jako repozytorium git pod adresem https://github.com/bartoszputek/poker-master-tool/.

Do szybkiego uruchomienia aplikacji na lokalnej maszynie należy zainstalować oprogramowanie Docker, służące do wirtualizacji kontenerów. Aby uruchomić aplikację należy zainstalować sklonować repozytorium oraz wykonać polecenie — `docker-compose up`.

Innym sposobem na zbudowanie projektu jest zainstalowanie środowiska uruchomieniowego Node.js i wykonanie kolejnych poleceń:

```
1 npm install
2 npm run build:all
3 npm run start
```

Po uruchomieniu aplikacja będzie dostępna jako serwer HTTP pod adresem lokalanym `http://localhost:3000`.

Repozytorium zawiera testy jednostkowe oraz integracyjne umieszczone w katalogu `tests`. Testami jednostkowymi zostały pokryte wszystkie klasy poza klasami

fasadowymi, które obudowują funkcje standardowe oraz biblioteki zewnętrzne w wygodny interfejs. Testy zostały napisane według zasad szkoły Londyńskiej [16] z użyciem narzędzi do imitowania (ang. mock/stub) zależności, testując zachowanie i interakcje pomiędzy obiektami. Uruchomienie testów wymaga wykonania polecenia `npm run test`.

Język	Pliki	Liczba linii
TypeScript	36	1621
JavaScript	18	745
C++	5	597
C/C++ Header	4	439
HTML	1	414
CSS	5	317
JSON	4	143
YAML	2	57
Dockerfile	1	14
make	1	6
Razem	77	4353

Tablica 4.1: Statystyki kodu wygenerowane za pomocą narzędzia cloc

4.2. Aplikacja serwerowa

Aplikacja serwerowa została napisana w statycznie typowanym języku TypeScript kompilowanym do języka JavaScript z użyciem biblioteki Express.js. Serwer działa w środowisku uruchomieniowym Node.js. Kod źródłowy modułu backendowego został umieszczony w katalogu `src`.

Aplikacja serwerowa ma za zadanie:

- Przyjmować żądania HTTP
- Serwować widok klienta w formie statycznie skompilowanych plików
- Logować działanie aplikacji
- Przechowywać w pamięci Cache ostatnie zapytania w celu optymalizacji
- Obsługiwać błędy aplikacyjne oraz programistyczne

Aplikacja wystawia jeden REST'owy endpoint POST pod adresem / , którego celem jest zwrócenie informacji o szansie wygranych, porażek oraz szansie uzyskania poszczególnych układów przez graczy. Endpoint oczekuje dostarczenia danych o kartach graczy, wspólnych oraz kart martwych, które nie biorą udziału w obliczeniach.

Po otrzymaniu żądania następuje walidacja danych. Jeżeli takie samo żądanie zostało wykonane niedawno, wynik zostanie zwrócony z pamięci podręcznej (cache) w celu przyspieszenia działania programu. W przeciwnym wypadku obliczany jest wynik za pomocą silnika gry.

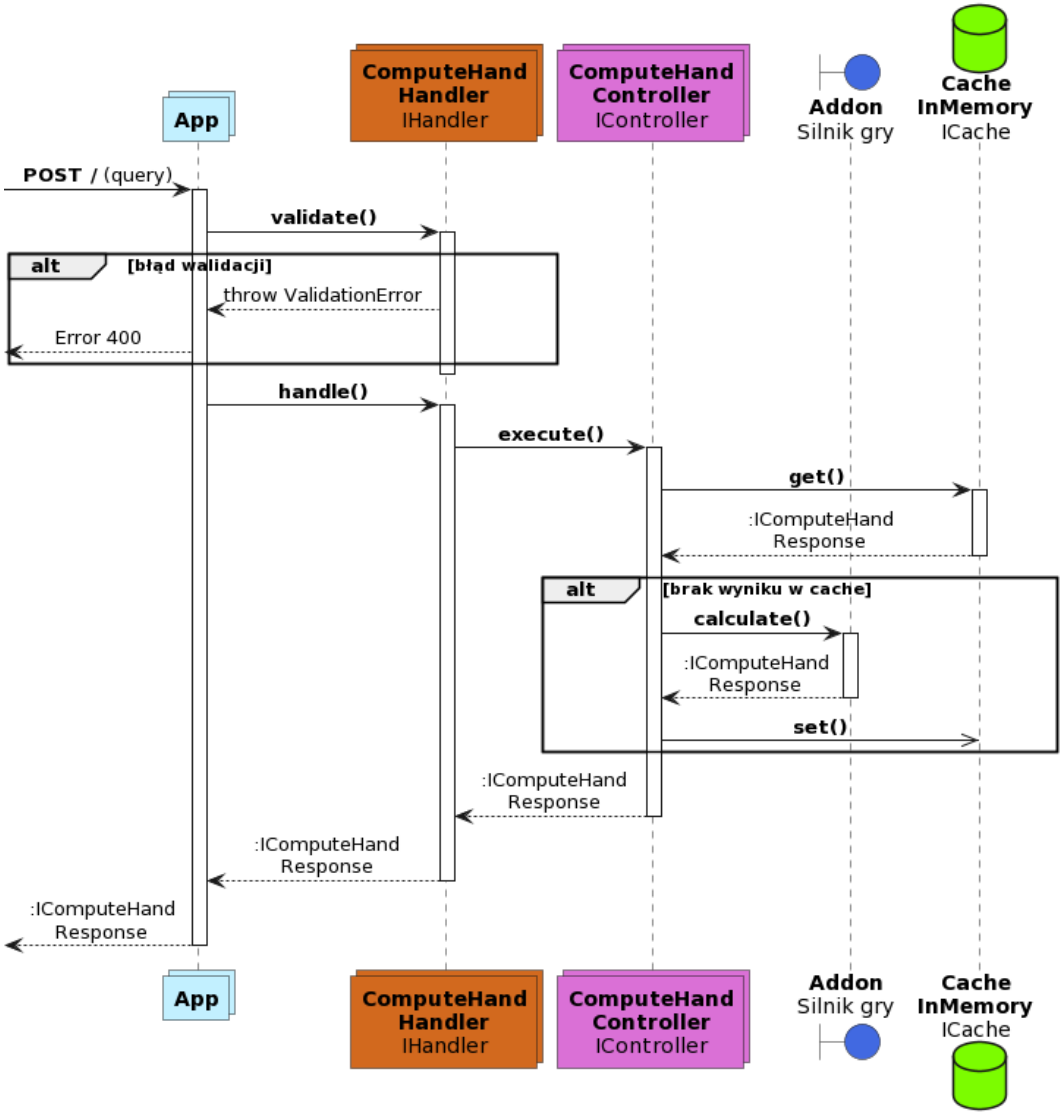
4.3. Silnik gry

Silnik gry, wykorzystywany do kalkulacji układów to natywny moduł C++ napisany jako rozszerzenie Node.js. Algorytm wykorzystany do generacji grafu został napisany przez autorów ewaluatora *TwoPlusTwo*, z naniesionymi modyfikacjami poprawiającymi czytelność oraz szybkość działania z wykorzystaniem standardu C++ 17. Decyzja o wykorzystaniu rozszerzenia została podjęta z przyczyn wydajnościowych, odpowiednik algorytmu w języku JavaScript był zbyt wolny dla pesymistycznych przypadków.

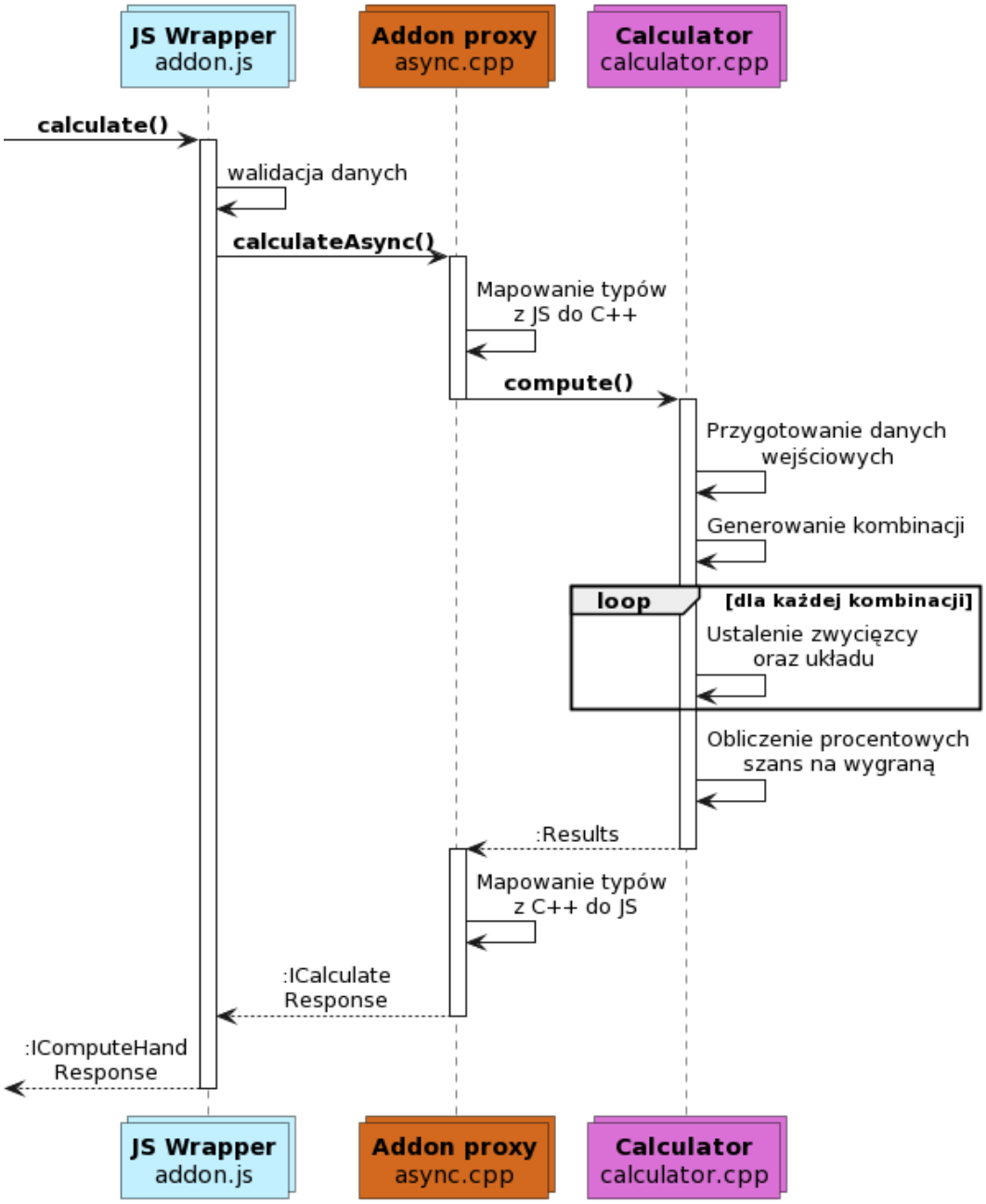
Przed rozpoczęciem kalkulacji, poleceniem `npm run addon:generate-table`. należy zbudować graf. Następnie, graf musi zostać wczytany do pamięci aplikacji korzystając z funkcji rozszerzenia `initLookUpTable()` .

Dla silnika gry zostały napisane testy wydajnościowe, które zwracają czasy obliczeń dla wcześniej przygotowanych przypadków użycia. Uruchomienie testów wymaga skorzystania z polecenia `npm run test:performance`.

Rozszerzenie zostało obudowane fasadą upraszczającą korzystanie z obliczeń silnika oraz walidującą poprawność danych wejściowych. Rozszerzenia wymagają mapowania typów — do tego zadania zostało stworzone proxy.



Rysunek 4.1: Diagram sekwencji prezentujący interakcję pomiędzy obiektami podczas wywołania metody POST na ścieżkę /.



Rysunek 4.2: Diagram sekwencji prezentujący przepływ informacji podczas wywołania funkcji calculate().

4.4. Aplikacja klienta

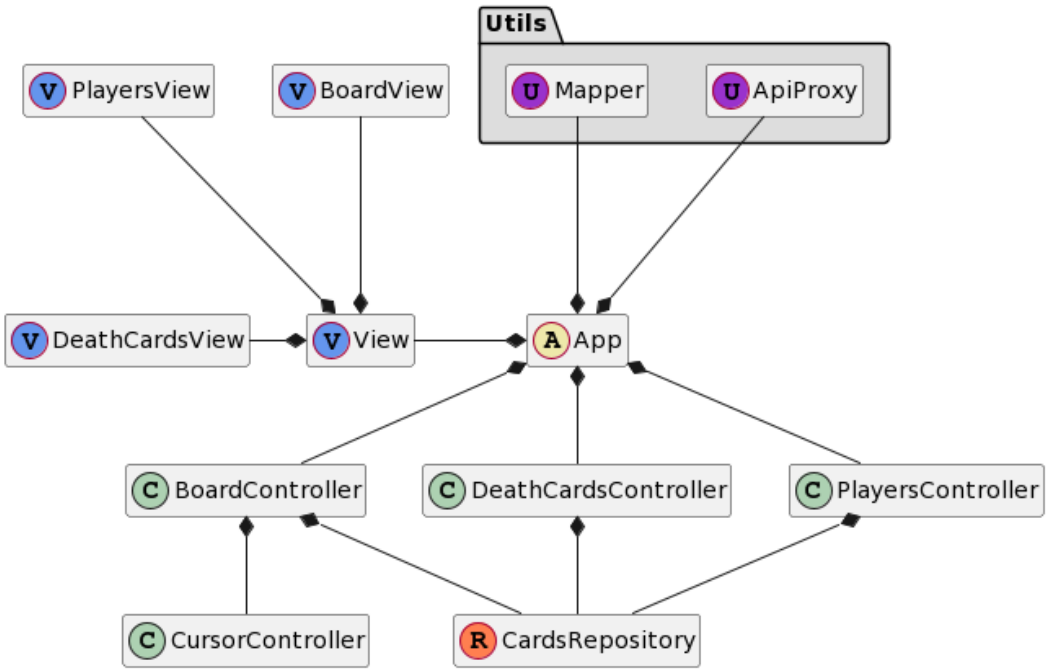
Rolą aplikacji klienta jest udostępnienie interfejsu przyjaznego dla użytkownika. Widok dostosowuje się do szerokości ekranów urządzeń mobilnych oraz umożliwia obsługę kalkulatora za pomocą klawiatury. Część frontendowa zaprojektowana jest z użyciem technologii JavaScript/HTML/CSS oraz Webpack, który jest narzędziem do budowania projektu (ang. bundler). Kod źródłowy modułu znajduje się w katalogu `frontend`.

Kod podzielony jest na niezależne komponenty, zgodnie z zasadą pojedynczej odpowiedzialności. Widoki utrzymywane są jako klasy `View`, komponenty logiczne umieszczone są w klasach `Controller`, a tymczasowy stan aplikacji utrzymywany jest w repozytoriach.

Na podstawie stanu aplikacji podejmowana jest decyzja o wykonania żądania `POST /` i wyświetleniu odpowiedzi. Żądanie wykonywane jest gdy:

1. Każdy z graczy jest gotowy (liczba jego kart to 0 lub 2)
2. Gotowy jest co najmniej jeden gracz
3. Liczba kart wspólnych jest zgodna z poszczególnymi licytacjami (0/3/4/5)

Wykorzystując powyższą logikę unikamy wykonywania błędnych żądań oraz nie używamy nadmiernie zasobów obliczeniowych serwera.



Rysunek 4.3: Diagram klas wykorzystywanych przez aplikację klienta.

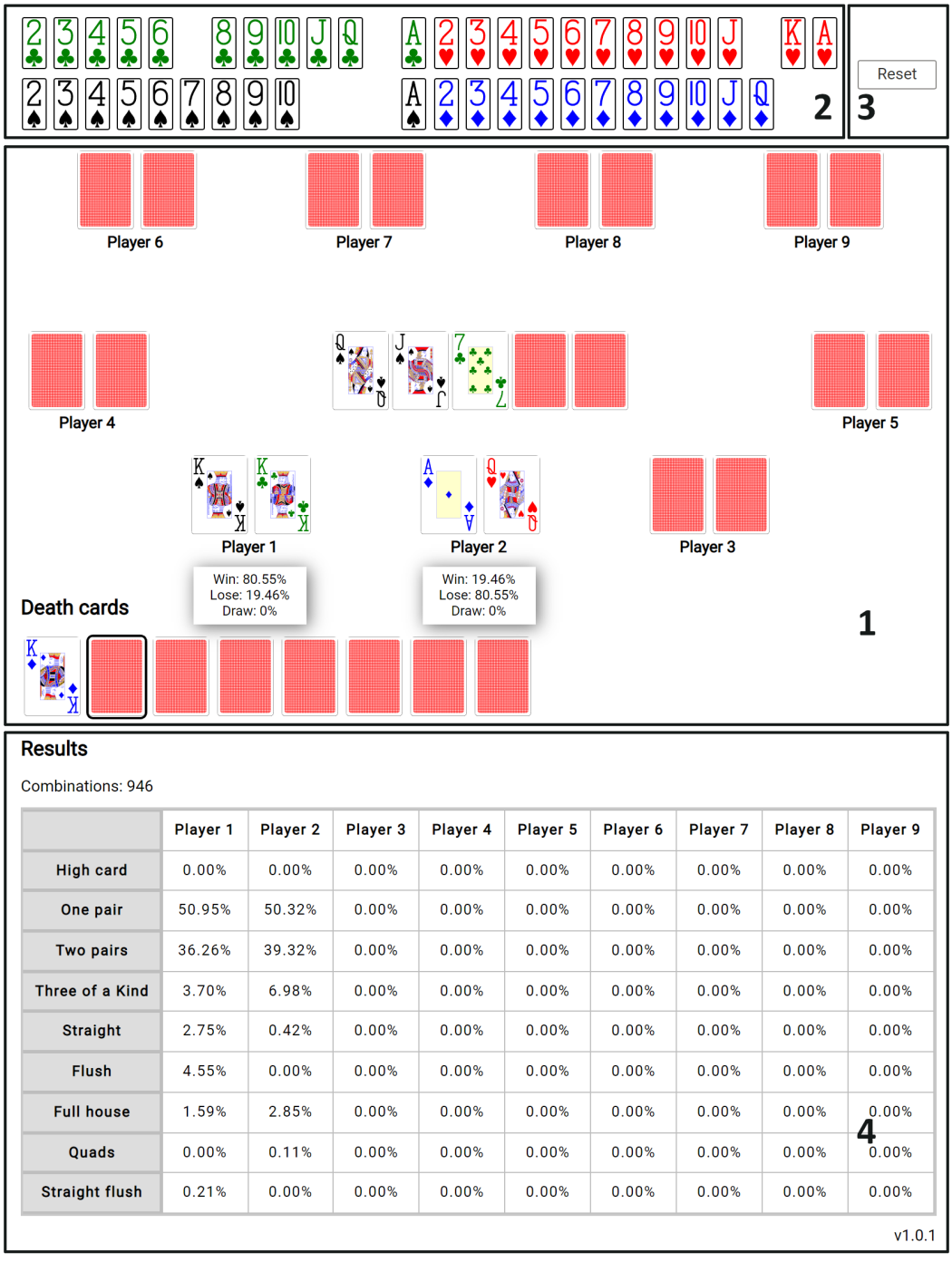
Rozdział 5.

Instrukcja użytkownika

Interfejs aplikacji jest podzielony na 4 sekcje, zaznaczony na rysunku 5.1.

1. Widok stołu: w tym miejscu użytkownik może zaznaczyć kursorem miejsce na karty gracza, wspólne, bądź „martwe", wyjęte z talii na czas obliczeń. Po procesie kalkulacji pojawiają się tu także szanse na zwycięstwo graczy biorących udział w rozdaniu.
2. Menu kart: po wybraniu karta zostaje umieszczona w miejscu zaznaczonym przez kursor.
3. Przycisk reset: służący do resetu widoku oraz otrzymanych wyników
4. Sekcja wyników: zawiera liczbę kombinacji oraz tabelę z szansami na poszczególne układy dla każdego z graczy

Użytkownik zaznacza poszczególne elementy interfejsu lewym przyciskiem myszy, bądź korzystając z klawisza Enter. Dodatkowo interfejs umożliwia nawigację po wszystkich elementach korzystając z klawiszy Tab oraz Tab+Shift. Poza przyciskiem do resetu scenariusza można usuwać pojedyncze karty lewym przyciskiem myszy lub klawiszem Enter. Po umieszczeniu karty nie trzeba kolejny raz wybierać miejsca docelowego — kursor aplikacji przemieszcza się z kolejnymi wyborami kart.



	Player 1	Player 2	Player 3	Player 4	Player 5	Player 6	Player 7	Player 8	Player 9
High card	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
One pair	50.95%	50.32%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Two pairs	36.26%	39.32%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Three of a Kind	3.70%	6.98%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Straight	2.75%	0.42%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Flush	4.55%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Full house	1.59%	2.85%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Quads	0.00%	0.11%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Straight flush	0.21%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Rysunek 5.1: Interfejs użytkownika z kolejno zaznaczonymi sekcjami

Rozdział 6.

Podsumowanie

W ramach niniejszej pracy został zbudowany pokerowy kalkulator szans, który wyróżnia się na tle konkurencyjnych rozwiązań i wypełnia lukę w obecnym rynku. Odświeżony kod ewaluatora z gotowym rozszerzeniem dla środowiska Node.js może być przydatny do implementacji rozwiązań pokerowych przez innych programistów.

Architektura aplikacji wykorzystuje wiele technologii oraz korzysta z szerokiej gamy wzorców projektowych, aby usprawnić proces rozwoju oprogramowania oraz wydajnie wykonywać obliczenia.

Dalszy praca nad aplikacją mogłaby skupić się na jeszcze szybszym działaniu. Ztablicowanie przypadków dla dwóch graczy wydaje się rozsądnym kierunkiem, biorąc pod uwagę fakt, że są to najczęstsze przypadki użycia. Liczba zapamiętanych kombinacji jest równa $\binom{52}{2} \cdot \binom{50}{2} \cdot \frac{1}{2} = 812175$ bez wykonywania możliwych prób redukcji tej liczby. Dodanie do istniejącej infrastruktury bazy danych byłoby wskazane w celu przechowania wcześniej obliczonych wartości.

Następną sugerowaną poprawką jest uproszczenie generowanie grafu dla silnika gry. Reprezentacja karty w ewaluatorze *Cactus Kev's* nie wykorzystuje bitów oznaczonych jako R, co nie potrzebnie zaciemnia działanie algorytmu.

Kolejnym mechanizmem, który jest przydatny to utrzymywanie odpowiedzi z wykonanych żądań po stronie aplikacji klienta. Gdy wykonujemy żądanie z tymi samymi parametrami, spodziewamy się tego samego wyniku, zatem nie trzeba wykonywać ponownie zapytania do serwera.

Mimo istnienia powyższych ulepszeń istotnie usprawniających działanie aplikacji, oprogramowanie jest w pełni gotowe do użycia dla użytkowników i zawiera wszystkie przewidziane funkcjonalności.

Bibliografia

[1] *Artykuł Poker.* w: Wikipedia, the free encyclopedia [online], dostępny w: `https://en.wikipedia.org/wiki/Poker`, dostęp 16.06.2022.

[2] *Artykuł Texas Hold'em.* w: Wikipedia, the free encyclopedia [online], dostępny w: `https://pl.wikipedia.org/wiki/Texas_Hold%E2%80%99em`, dostęp 16.06.2022.

[3] *Artykuł Monte Carlo Method.* w: Sciencedirect [online], dostępny w: `https://www.sciencedirect.com/topics/medicine-and-dentistry/monte-carlo-method`, dostęp 16.06.2022.

[4] Dr hab. inż. Łukasz Madej *Prezentacja Metoda Monte Carlo - podstawy* w: agh.edu.pl [online], dostępny w: `http://home.agh.edu.pl/~lmadej/wp-content/uploads/wyklad_9_MC.pdf`, dostęp 16.06.2022.

[5] Kevin Suffecool *Cactus Kev's Poker Hand Evaluator* dostępny w: `http://suffe.cool/poker/evaluator.html`, dostęp 16.06.2022.

[6] Kevin Suffecool *Cactus Kev's Equivalence Classes* dostępny w: `http://suffe.cool/poker/7462.html`, dostęp 16.06.2022.

[7] *Artykuł Fundamental theorem of arithmetic.* w: Wolfram MathWorld. [online], dostępny w: `https://mathworld.wolfram.com/FundamentalTheoremofArithmetic.html`, dostęp 16.06.2022.

[8] *Lookup table.* w: Wikipedia, the free encyclopedia [online], dostępny w: `https://en.wikipedia.org/wiki/Lookup_table`, dostęp 16.06.2022.

[9] *Binary search algorithm.* w: Wikipedia, the free encyclopedia [online], dostępny w: `https://en.wikipedia.org/wiki/Binary_search_algorithm`, dostęp 16.06.2022.

[10] Paul Senzee *Wpis na blogu - Some Perfect Hash.* w: Senzee Blogspot [online], dostępny w: `http://senzee.blogspot.com/2006/06/some-perfect-hash.html`, dostęp 16.06.2022.

[11] Ben Coleman *Artykuł Minimal perfect hash functions.* w: Randorithms [online], dostępny w: `https://randorithms.com/2019/09/12/MPH-functions.html`, dostęp 16.06.2022.

[12] Bob Jenkins *Code for perfect hashing.* w: Burtleburtle.net [online], dostępny w: `http://burtleburtle.net/bob/index.html`, dostęp 16.06.2022.

[13] Wątek na forum *7 Card Hand Evaluators.* w: Twoplustwo.com [online], dostępny w: `https://forumserver.twoplustwo.com/45/general-software-discussion/7-card-hand-evaluators-597`, dostęp 16.06.2022.

[14] *The Great Poker Hand Evaluator Roundup.* w: Codingthewheel.com/ [online], dostępny w: `https://www.codingthewheel.com/archives/poker-hand-evaluator-roundup`, dostęp 16.06.2022.

[15] *Artykuł Finite-state machine.* w: Wikipedia, the free encyclopedia [online], dostępny w: `https://en.wikipedia.org/wiki/Finite-state_machine`, dostęp 16.06.2022.

[16] Martin Fowler *Mocks Aren't Stubs* w: Martinfowler.com [online], dostępny w: `https://martinfowler.com/articles/mocksArentStubs.html`, dostęp 16.06.2022.