UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

BEZPEČNÉ HRY VIACERÝCH HRÁČOV V PROSTREDÍ REACT

Diplomová práca

2020 **Júlia Kázsmérová**

UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

BEZPEČNÉ HRY VIACERÝCH HRÁČOV V PROSTREDÍ REACT

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Ústav informatiky
Vedúci práce:	RNDr. Jozef Jirásek, PhD.

Košice 2020 **Júlia Kázsmérová**

Abstrakt

V dnešnej dobe trh online hier je stále väčší, kde tieto hry sú často slabo zabezpečené. To môže viesť k rôznym stratám, preto je čoraz dôležitejšie sa venovať aj bezpečnosti v hrách. V práci sme sa rozhodli venovať tejto problematike. Upozorníme na rôzne potencionálne hrozby a navrhneme možné riešenia. Súčasťou tejto práce je aj implementácia a vytvorenie hry v prostredí React s použitím vybraných autentifikačných a bezpečnostných protokolov.

Kľúčové slová: *online hry, hry viacerých hráčov, bezpečnosť, bezpečnostné protokoly, autentifikačné protokoly, React.*

Obsah

Úvod

V dnešnej dobe sa vyvíja veľa online hier a vďaka rôznym vymoženostiam už hry môžu vyvíjať aj úplní začiatočníci. Preto na trhu sa objavujú aj hry, ktoré nie sú dobré navrhnuté, sú teda pomalé a často sú aj málo zabezpečené[7]. To môže viesť k rôznym problémom najmä, keď v hre hrá viacero hráčov. Ak si vezmeme napríklad nejaké kartové hry s trochou znalosti môžeme ovplyvniť svoj ťah alebo ťah iných [5]. Môžeme teda podvádzať, čo niekedy môže viesť aj k strate financií hráčov [4]. Ak užívatelia si všimnú podvádzanie, môže ich to odradiť a môže to znamenať stratu financií aj pre správcov hry. Ďalšou potencionálnou hrozbou je zisk osobných údajov útočníkmi. Ak však je hra správne zabezpečená kryptograficky, hry viacerých hráčov nemôžu byť ovplyvnené a nemôže dôjsť k strate osobných údajov alebo financií.

V tejto práci sa venujeme známym metódam bezpečnej komunikácie viacerých hráčov cez počítačovú sieť. Súčasťou práce je aj návrh a realizácia scenára hry v prostredí React, ktorá pomáha pri tvorbe používateľských rozhraní. Cieľom je použitie vybraných autentifikačných a bezpečnostných protokolov v tejto hre a taktiež otestovanie hry v rôznych systémových platformách.

V prvom kroku sme sa rozhodli venovať výberu hry. Pri výbere sme sa snažili zvoliť takú hru, kde by sa ukázal správnosť fungovania vybraných autentifikačných a bezpečnostných protokolov. Najvhodnejšie sa nám zdali kartové alebo kockové hry, keďže funkčnosť kryptografických protokolov sa dá na nich jednoducho ukázať a ich implementácia nie je komplikovaná. Nakoniec sme si zvolili implementovať hru Srdce, ktorá je kartová hra pre štyroch hráčov. Všeobecné pravidlá a priebeh hry sa nachádzajú v 1. kapitole.

Ďalším krokom je implementácia hry v prostredí React. V tomto kroku sa najprv sústredíme na základné vlastnosti hry a na ich funkčnosť. Použijeme niekoľko hlavných čŕt Reactu, akým je napríklad schopnosť hneď reagovať na zmeny. Takými zmenami môžu byť napríklad výmena kariet alebo vyloženie karty. Ak niektorá z týchto zmien nastane u hráča, pomocou Reactu vieme veľmi rýchlo a jednoducho zobraziť tieto zmeny ostatným hráčom.

Neskôr do hry zahrnieme vybrané autentifikačné a bezpečnostné protokoly. Tento

krok obsahuje získanie prehľadu existujúcich a použitých protokolov v hrách slúžiace na bezpečnú komunikáciu a autentifikáciu hráčov. Súčasťou je aj ich analýza týchto protokolov a následný výber najvhodnejších z nich pre našu implementáciu hry.

Posledným krokom bude otestovanie funkčnosti našej vytvorenej hry a použitých protokolov v rôznych systémových platformách, prehliadačoch a na rôznych zariadeniach.

Obr. 1: Jedným z cieľov je otestovanie funkčnosti hry na rôznych zariadeniach a systémových platformách.

1 Srdce

Hra Srdce je kartová hra inak nazývaná aj ako Hearts, Black Lady, Black Maria alebo Calamity Jane. Pochádza z 19. storočia z Beneluxu. Obľúbenejšou a známejšou sa stála po tom, čo ju Microsoft zaradil do štandardného príslušenstva Windowsu [14].

1.1 Základné vlastnosti

Táto hra je zvyčajne pre štyroch hráčov ale nájdu sa varianty aj pre iný počet hráčov. Hrá sa s jedným balíčkom francúzskych kariet, čiže s 52 kartami. Pri verzii pre štyroch hráčov na začiatku hry každý hráč dostane 13 kariet. Pre iný počet hráčov, každý hráč dostane rovnaký počet kariet a zvyšné sa vylúčia. Karty sa rozdávajú v smere hodinových ručičiek. Karty sú ohodnotené od esa (najvyššia) po dvojku (najnižšia) [12].

1.2 Odovzdávanie kariet

Pôvodná hra nezahŕňa odovzdávanie kariet na začiatku kola, ale tie najbežnejšie varianty zahŕňajú. Na začiatku kola si hráč vyberie 3 karty, ktoré chce odovzdať. V prvom kole sa odovzdajú karty hráčovi naľavo, v druhom napravo, v treťom hráčovi oproti a vo štvrtom sa neodovzdávajú karty. Takéto štvorfázové cykly sa opakujú do konca hry.

Cieľom odovzdávanie je posunúť nebezpečné karty. Preto sa odporúča odovzdať najvyššie karty najmä farby pikovej a srdcovej a pikovú kráľovnú. Ďalším cieľom je mať čo najmenej kariet jednotlivých farieb.

1.3 Priebeh hry

Hra sa väčšinou začína podobne ako u iných kartových hrách. Po rozdaní a odovzdaní kariet hru začne hráč, ktorý sa nachádza naľavo od dealera. V najznámejšej

verzii hru začína hráč, ktorý vlastní krížovú dvojku vyložením tejto karty. Hráč, ktorý začína daný ťah sa nazýva vedúci ťahu (tzv. lead).

V každom ťahu spoluhráči musia pokračovať s kartou rovnakej farby akou začal hráč ťah. Ak niekto to nemôže dodržať, lebo nemá kartu takej farby, tak hráč môže použiť ľubovoľnú kartu. V niektorých verziách je stanovené, že ak to je prvý ťah daného kola a spoluhráč nemá kartu danej farby, tak môže vyložiť ľubovoľnú kartu okrem bodových kariet (srdcové a piková kráľovná) [17].

Ťah spolu s prípadnými trestnými bodmi vyhráva hráč, ktorý vyložil kartu s najväčšou hodnotou tej farby, ktorou začal vedúci ťahu. Tento hráč sa stane nový vedúcim ťahu. Ak vedúci ťahu si nezvolil stratégiu čistej hry, tak aby nezískal neželané bodové karty, odporúča sa začať ťah s najmenšou kartou jednotlivých farieb, ktoré má k dispozícii.

Hráč začínajúci ťah môže začať s ľubovoľnou kartou. Výnimkou sú srdcové karty. Kartou takejto farby sa nemôže začať až kým nie sú „zlomené“ („Breaking hearts“). Tento pojem sa používa na situáciu, keď je použitá srdcová karta iným hráčom v predchádzajúcom ťahu z dôvodu, že nemal kartu potrebnej farby. Toto pravidlo môže byť porušené iba vtedy, keď vedúci ťahu už nemá žiadnu kartu inej farby k dispozícii.

1.4 Bodovanie

V tejto hre body sú považované za trestné body. Za každú srdcovú kartu v získanom ťahu hráč dostane bod. Hráč, ktorý vyhral ťah obsahujúci pikovú kráľovnú dostáva 13 bodov. V každom kole je teda 26 trestných bodov.

Môže nastáť situácia nazývaná „čistá hra“ („Shooting the moon“), kde sa hráčovi podarí získať všetky srdcové karty spolu s pikovou kráľovnou [13]. V tomto prípade tomuto hráčovi sa nepripíšu žiadne body a trestné body sa pripočítajú všetkým ostatným hráčom. Táto situácia je znázornená na obrázku 2 v šiestom kole.

	Západ	Sever	Východ	Juh
1.	0	1	25	0
2.	1	0	13	12
3.	0	0	20	6
4.	0	0	18	8
5.	0	0	25	1
6.	0	26	26	26

Obr. 2: Bodovanie a koniec hry.

1.5 Koniec hry

Koniec hry môže nastať po určitom počte kôl, po nejakom časovom limite alebo po dosiahnutí určitého počtu trestných bodov niektorým z hráčov. V najznámejšej variante na ukončenie hry sa používa limit 100 trestných bodov.

Hru vyhráva hráč s najmenším počtom bodov. Ak nie je jasný výherca, t.j. keď ide o verziu s limitovaným počtom trestných bodov, kde dvaja majú rovnako najmenší počet bodov, tak hra pokračuje ďalej, kým nebude jasný výherca. Takýto prípad môžeme vidieť aj na obrázku 2, kde hráč „Východ“ v piatom kole presiahne limit (100 bodov), ale hra pokračuje ďalej, keďže hráči „Západ“ a „Sever“ majú rovnako najmenší počet bodov. V ďalšom kole už je jasný výherca a teda hra sa skončí.

2 Bezpečnosť v online hrách

V minulosti bolo potrebné riešiť bezpečnosť v hrách hlavne kvôli autorským právam, aby nevznikali nelegálne kópie hry. V dnešnej dobe hráči namiesto kupovania hier si radšej zahrajú online hry. Tým vznikli nové problémy, ktoré treba riešiť. Čím je väčší počet užívateľov o to viac sa treba starať o bezpečnosť hry, keďže útočníci môžu chcieť získať ich osobné údaje. Ďalej, v hrách často sú prepojené aj reálne peniaze s menami a nákupmi predmetov týkajúce sa hry. Tento fakt taktiež využívajú útočníci, či už na zisk reálnych peniazi alebo virtuálnych majetkov v hre.

Takéto a ďalšie možné hrozby opísali aj Woo J. a Kim H.J. v práci [10]. Najčastejšie útoky podľa vlastností rozdelili do kategórií, popísali ich a označili závažnosť útoku vzhľadom na poskytovateľov hry a užívateľov. Túto tabuľku môžeme vidieť na obrázku 3. V ďalšej tabuľke (obrázok 4) uviedli protiopatrenia voči týmto útokom. Súčasťou ich práce je aj popis ako by mohli byť niektoré z útokov detegované.

Category	Description	Severity (effect on game service providers)	Severity (effect on users)
Game bot	- A game bot is an automated program that plays the games in a human's stead. - Game bots can be categorized by physical type: sqecifically, software type, USB type, and mouse type. - Game bots can also be categorized by running type: specifically, out-of-game (OOG) client bots and in-game (IG) client bots [A.R. Kang and Kim 2012]	High	High
In-game hack	- In-game hack is the manipulation of a computers memory and game process(es) to get advantages. (Hacks include memory hacks, speed hacks, HP hacks, wall-hacks, aim-hacks, etc.)	High	High
Gold farming	- Gold farming is done by highly industrialized game sweatshops. - Gold farming is categorized into two types: Labor intensive and Automated. - Labor-intensive gold farming relies on cheap human workers game plays. - Automated gold farming relies on highly crafted game bot programs.	High	High
Private servers	Private servers are categorized into two types: - Servers created by reverse engineering analysis (emulated version, not related to hacking incidents). - Servers that are the same as the genuine one (usually obtained via a system hack or internal file leakage).	High	Low
System or network hacking	- These are remote exploit attacks directly targeted at the games servers, especially the database system that contains users in-game cyber assets and equipment data. - Once this hacking has succeeded, hackers usually run an update query to manipulate the asset or inventory records.	High	Medium
Identity theft (account theft)	- In this attack, the attacker logs in with stolen user accounts. - This attack is enabled by malware (such as dropper or password stealer).	Medium	High
Misc.	- In-game forgery or hoax. - In-game spamming. - Harassment.	Low	Low

Obr. 3: Najčastejšie útoky v online hrách podľa Woo J. a Kim H.J. [10].

V ďalšom článku Chen a spol. [3] analyzovali 613 kriminálnych prípadov, ktoré sa vyskytli v Tajvane v roku 2002. Na základe výsledkov zistili, že okrem iných, najčastejšie útoky sú krádeže (mena, hesla, virtuálneho majetku) a podvody. Ich výsledky zahŕňajú aj najčastejšie miesta a časy, kde a kedy boli útoky vykonané. O ďalších zaujímavých zistených faktoch sa môžeme dočítať v ich práci. V článku zverejnili obrázok, v ktorom kategorizovali trestné činy týkajúce sa online hier. Z množstva zaujímavých výsledkov ich štatistiky je pre nás najužitočnejšia tabuľka, ktorá vyjadruje ich rozdelenie útokov a počet výskytov daného útoku. Túto tabuľku môžeme vidieť na obrázku 5. Na záver uverejnili pre užívateľov aj pre poskytovateľov hry rady, ako by sa mali brániť proti týmto útokom.

Category	Countermeasures	Types
Game bot	Client-side: Install bot detection programs. Network-side: Protect network traffic. Server-side: Data mining and analysis to find out game bots play pattern.	- Running of detection software on the client - Prevention of malicious program injection attempts in client game software. - Frequent changing of games network protocol. - Application of cryptography to encrypt/decrypt network transmission. - Log analysis at the server-side to reveal anomalous user activities.
In-game hack	Client-side: Install game security solutions. Server-side: Implement investigation code that inspects all incoming packets.	- Protection of memory space from unauthorized programs. - Protection of games running processes from dll or process injection. - Protection of files and integrity checks to detect unauthorized modification. - Checking of all input packets and analysis of the input range at the server-side.
Gold farming	Install bot detection programs.	- Running of detection software on the client. - Prevention of malicious program injection attempts in the game client software. - Log analysis at the server-side to detect suspicious transactions for items and money trade.
Private servers	Implement private server detection module.	- Running of detection software on the client. - Evidence collection support for legal investigation.
System or network hacking	Enforce system and network security.	- Deployment of traditional security solutions such as firewalls, IDPSs, database security solutions, PMS, etc.
Identity theft (account theft)	Enforce individual PC security or provide multi-factor authentication.	- Implementation of secondary authentication methods such as security card and one time password (OTP). - Running of on-demand antivirus software to detect malware (e.g., keylogger or password stealer programs).
Misc.	Enforce in-game monitoring by game masters (GMs)	- Positive gathering of users petitions to resolve conflicts among players.

Obr. 4: Protiopatrenia proti jednotlivým útokom podľa Woo J. a Kim H.J. [10].

Measure	Value	Frequency	Percentage
Type of crimes	Theft	452	73.7
	Fraud	124	20.2
	Robbery	9	1.5
	Threat	2	0.3
	Others	26	4.2
Total		613	100.0

Obr. 5: Rozdelenie útokov podľa Chen a spol. [3] a frekvencia ich výskytu.

Yan a Randell [11] sa tiež rozhodli venovať tejto problematike a svojim článkom chceli pomôcť hlavne bezpečnostným odborníkom a vývojárom hier. V článku uviedli a charakterizovali 15 najčastejších útokov ako napr. podvádzanie pomocou zmeny kódu alebo podvádzanie spoluprácou. Tieto metódy podvádzania následne rozdelili podľa

toho aká zraniteľnosť je využitá, aké sú dôsledky a podľa toho kto podvádza. Takéto rozdelenie môžeme vidieť na obrázku 8.

CLASSIFICATION OF VARIOUS TYPES OF CHEATING	VULNERABILITIES				POSSIBLE FAILURES					EXPLOITERS			
	SYSTEM DESIGN INADEQUACY		PEOPLE		FAIRNESS VIOLATION	MASQUERADE	INTEGRITY VIOLATION	SERVICE DENIAL	THEFT OF INFORMATION OR POSSESSIONS	INDEPENDENT		COOPERATIVE	
	In underlying systems	In game system	Player	Game operator						Single player	Game operator	Multiple players	Operator and player
A. Exploiting misplaced trust		•					•		•	•			
B. Collusion		•			•				•			•	
C. Abusing the game procedure		•			•					•			
D. Cheating related to virtual assets			•		•					•			
E. Exploiting machine intelligence		•			•					•			
F. Modifying client infrastructure	•						•			•			
G. Denying service to peer players	•	•						•		•			
H. Timing cheating		•			•		•			•			
I. Compromising passwords			•						•	•			
J. Exploiting lack of secrecy		•					•		•	•			
K. Exploiting lack of authentication		•				•				•			
L. Exploiting a bug or design loophole		•			•					•			
M. Compromising game servers	•						•			•			
N. Internal misuse				•			•				•		•
O. Social engineering			•						•	•			

Obr. 6: Kategórie podvádzanie a ich rozdelenie na základe rôznych aspektov podľa Yan a Randell [11].

Tieto rozdelenia útokov a podvádzania nám môžu pomôcť pri práci. S niektorými z nich sme sa už aj zaoberali (napr. krádežou mena a hesla alebo s pokusmi o podvod klamaním) a po podrobnom preskúmaní ostatných hrozieb zvážime aké protiopatrenia by boli najvhodnejšie.

3 Návrh riešenia

V tejto kapitole je opísaný priebeh hry, konkrétny výber pravidiel, ktoré budeme implementovať a taktiež návrhy riešenia jednotlivých kľúčových bodov.

3.1 Pravidlá hry

Keďže existuje veľa rôznych verzií hry Srdce, v tejto podkapitole bude opísaný nami zvolený priebeh hry.

Hra bude slúžiť pre štyroch hráčov, na začiatku teda každý hráč dostane 13 kariet. V každom kole hráč odovzdá 3 karty svojim susedom podľa vyššie uvedených pravidiel (kap. 1.2). Hru bude otvárať hráč, ktorý vlastní pikovú dvojku. Tento hráč musí vyložiť práve túto kartu v prvom ťahu. Ak budú môcť, spoluhráči budú musieť pokračovať kartou rovnakého druhu. Ak takúto kartu nebudú mať, budú môcť použiť ľubovoľnú inú kartu. Špeciálnym prípadom tohto pravidla bude možnosť použitia karty ľubovoľného druhu okrem srdcových a pikovej kráľovnej, ak pôjde a prvý ťah daného kola. Ťah vyhrá hráč podobne ako vo všetkých verziách. V našej implementácií hry sa taktiež objaví pravidlo „Breaking Hearts“ (kap. 1.3).

Bodovanie bude ako zvyčajne a zohľadní sa aj to, ak hráčovi sa podarí zahrať „čistú hra“. Koniec hry nastane po tom, čo niektorý z hráčov dosiahne 100 bodov. Víťazom sa stane hráč s najmenším počtom bodov. Ak viacerí by mali rovnako málo počet bodov, tak budú prebiehať ďalšie kolá, kým víťaz nebude jasný.

3.2 Architektúra

Pre našu štruktúru hry sme zvolili klient-server architektúru. Tento model je vhodný pre naše účely hlavne kvôli manažovaniu hráčov a hry.

Užívatelia budú mať účty pomocou ktorej sa budú prihlasovať do hry. Pred prvou hrou sa teda budú registrovať pomocou emailovej adresy. Na túto adresu dostanú overovací mail. Ak registrácia prebehne úspešne, údaje užívateľa sa uložia do databázy na server. Následne hráč bude verifikovaný a bude sa môcť prihlasovať do hry pomocou

údajov, ktoré zadal pri registrácii.

Server v rámci manažovania užívateľov sa stará aj o spojenie hráčov. Ak sa prihlásia štyria hráči, spojí ich do jednej hry [8]. Paralelne môže prebiehať aj viac nezávislých hier. Niekedy, keď sa nenájde dostatočný počet hráčov, hráči by mohli príliš dlho čakať. Preto pre väčšie pohodlie používateľov by sa mohli neskôr vytvoriť boti, ktorý by sa dogenerovali po istej dobe čakania. Vytvorenie tejto vlastnosti ale nie je zatiaľ isté.

3.3 Riadenie hry

Hra sa skladá z rôznych pravidiel, ktoré musia byť dodržané. To ako môžu byť jednotlivé pravidlá overené je opísané v tejto podkapitole. Veľmi dôležitou súčasťou je aj miešanie kariet, čo tiež musí prebiehať spoľahlivo. Taktiež sa vyskytnú návrhy riešenia problémov týkajúce sa bezpečnosti, podvádzania a možnosti ovplyvňovania hry.

3.3.1 Miešanie kariet

Miešanie kariet prebieha na začiatku každého kola. V tomto bode sme analyzovali otázku, či je lepšie aby miešala serverová časť, klientská časť poprípade či by sa mala použiť nejaká ich kombinácia.

Prvým najjednoduchším riešením sa zdalo byť nechať miešanie kariet na server. To by mohlo prebiehať tak, že server zamieša karty a každému hráčovi pošle 13 kariet zašifrované klientovým verejným kľúčom. Tieto karty následne daný hráč odšifruje svojim súkromným kľúčom. Tým by sme zabezpečili, že aj keby bola správa odchytená, útočník by ju aj tak nemohol prečítať ani pomeniť bez klientovho súkromného kľúča. Túto možnosť sme ale vylúčili z dôvodu, že ak server bude napadnutý, hráči by to nemali ako zistiť a hra by mohla byť ovplyvnená.

Ďalšou možnosťou je použiť dve dvojice kľúčov. Jedna dvojica by slúžila na autorizovanie klienta na strane servera, druhá by slúžila na autorizovanie servera na strane klienta. S ďalšími pridanými bezpečnostnými prvkami by sa mohlo zabezpečiť, aby aj klient mohol dôverovať serveru. Tým pádom by klienti si boli istý, že server nie je napadnutý a teda miešanie kariet by mohol vykonávať aj server.

Tretia alternatíva je bez prítomnosti servera, teda miešanie kariet by vykonávali iba klienti. Ak by miešal iba jeden z klientov, stále by tu bola možnosť, že tento klient môže manipulovať s kartami. Z tohto dôvodu by bolo vhodné zakomponovať všetkých klientov.

3.3.2 Rozdávanie kariet

Po zamiešaní kariet nasleduje fáza rozdávania. Tento problém sa dá riešiť rôznymi spôsobmi. Jedným z nich je rozdávanie kariet klasicky dealer-om, ktorým by mohol byť server alebo stále iný klient. Ďalšou možnosťou je rozdávanie, kde pomocou náhodných čísel by sa stanovilo, ktorý hráč ktorú kartu dostane. V tomto bode nastáva veľa možností ohľadom toho ako vytvoriť dané náhodné čísla a že čo by tieto čísla znamenali. Tieto čísla by mohli byť generované serverom alebo klientmi. Pre oba prípady máme ďalšie možnosti. Ak je náhodné číslo generované:

- Serverom:
  - mohli by sme generovať čísla od 1 po počet hráčov (n) bez opakovania. Tieto vygenerované čísla (r) by sa rozdelili medzi hráčmi a i-ty hráč by vybral svoju j-tu kartu (k) z balíku nasledovne:

$$k_{i,j} = r[i] + (j-1) * 4. \tag{1}$$

  - rovnako by sa vygenerovali toľko čísel, koľko je počet hráčov, ale balík kariet by sa rozdelil na intervaly vzhľadom ku počtu hráčov. Teda i-ty hráč by dostal svojich 13 kariet nasledovne:

$$k_i =< (r[i]-1) * 13 + 1, r[i] * 13 > . \tag{2}$$

- Klientmi:
  - mohli by byť rovnaké metódy na rozdelenie kariet. To by ale znamenalo, že všetci hráči by museli vygenerovať rôzne čísla. Tento problém sa dá jednoducho vyriešiť rôznymi spôsobmi.

Rozdávanie kariet sa teda dá riešiť rozlične, ale čím je väčšia náhodnosť, o to ťažšie vie útočník ovplyvňovať hru. Preto v našej implementácii by sa mohla použiť kombinácia vyššie spomenutých možností.

3.3.3 Dodržanie pravidiel

V hre sa vyskytujú rôzne pravidlá, ktoré musia byť dodržané každým hráčom. Táto kontrola taktiež môže prebiehať pomocou servera aj pomocou klientov.

Overovanie správnosti krokov serverom by znamenalo, že server musí poznať všetky karty jednotlivých hráčov, teda aj tých, ktoré sú pre všetkých ostatných spoluhráčov tajné. Tento prístup by bol najjednoduchší z hľadiska implementácie , avšak kvôli bezpečnosti nie príliš vhodný [9].

Niektoré z pravidiel by mohli byť kontrolované klientmi. Keďže hráči majú skryté karty pred ostatnými spoluhráčmi, iba pravidlá týkajúce sa vyložených kariet môžu byť nimi kontrolované. Takými pravidlami sú napríklad či prvá karta bola piková dvojka alebo pravidlo „Breaking Hearts“. Klienti by mohli odsúhlasiť správnosť kroku potvrdzovacou správou. Ak vedúci ťahu dostane od všetkých spoluhráčov potvrdenie, hra by mohla ďalej pokračovať, inak klienti by mohli oznámiť serveru podozrivé správanie, ktorý by hru ukončil.

Ostatné pravidlá by mohli klienti odkontrolovať spätne pred koncom hry. Ak by sa našiel nejaký neplatný krok, hra by bola vyhlásená za neplatnú a klientom sa body nezarátajú.

3.3.4 Bodovanie

Na konci každého kola sa zrátajú body pre každého hráča na základe získaných kariet. Ďalšie kolá hrajú hráči dovtedy, kým jeden z nich nedosiahne 100 trestných bodov a kým nie je jasný výherca.

Keďže počet bodov závisí iba od získaných kariet jednotlivých hráčov, teda od kariet, ktoré už nie sú tajné, tak body môžu byť zrátané aj serverom aj klientmi. Pri kontrole klientmi, by sa po každom kole mohlo skontrolovať, či sa počet bodov zhoduje u každého klienta. Inak by mohli klienti nahlásiť svoje pochybnosti serveru.

4 Technológie

V tejto práci budeme hru vyvíjať v prostredí React a taktiež použijeme iné technológie, ktoré uľahčujú prácu s React-om a pomocou ktorých môžeme pridať rôzne funkcionality. Použité knižnice a rozšírenia opíšeme v tejto kapitole.

4.1 React

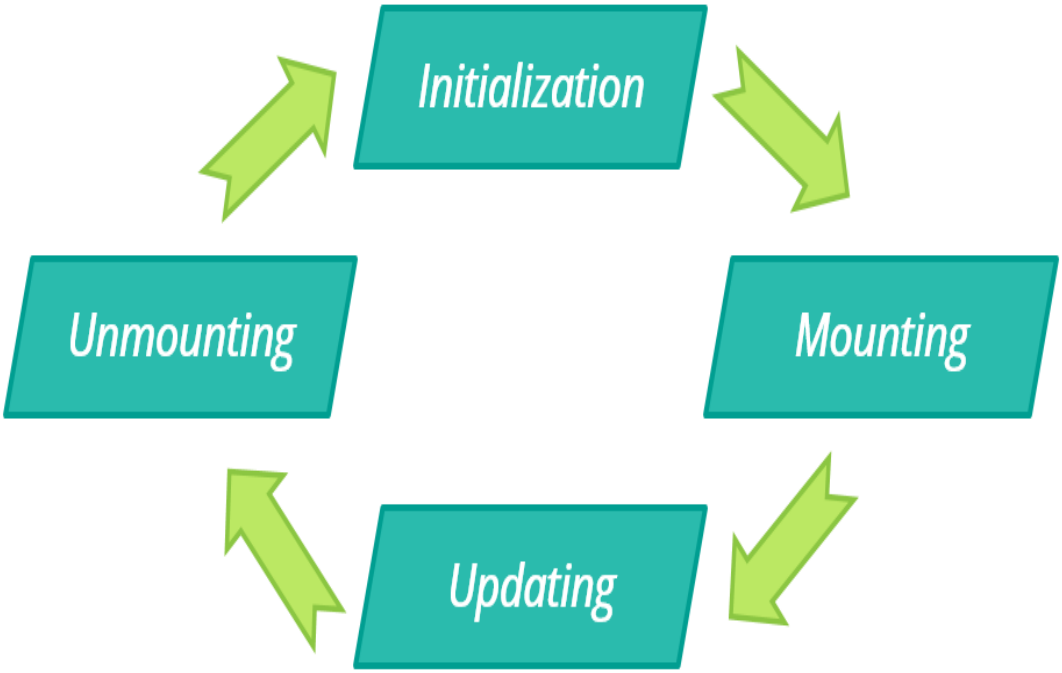
Je knižnica pre vytváranie používateľských rozhraní [1]. React bol vytvorený spoločnosťou Facebook. Snažili sa vyriešiť problémy, ktoré sa vyskytujú pri vytváraní zložitejších užívateľských rozhraní s dátami meniacimi sa z času na čas. Avšak môže byť použitý aj pre vytváranie jednoduchých stránok.

React má rôzne náročné aj menej náročné funkcie a idey. Základné črty React-u sú opísané v nasledujúcich podkapitolách.

4.1.1 Elementy a komponenty

Element opisuje to, čo chceme vidieť na obrazovke. Elementy sú nemenné, to znamená, že po ich vytvorení nemôžeme meniť jeho deti alebo atribúty. Aj keď vytvoríme element, ktorý popisuje celý UI strom, iba tá časť sa bude aktualizovať, kde sa mení obsah. Elementy zobrazujú DOM (Document Object Model) tagy a aj používateľom definované komponenty.

Pomocou komponentov sa dá rozdeliť užívateľské rozhranie na nezávislé, znovu použiteľné kúsky. Komponenty sú ako JS funkcie. Dostanú vstup (props) a vrátia React elementy, ktoré popisujú čo sa má zobraziť. Komponent sa môže odkazovať aj na ďalšie komponenty. Často je dobré rozdeliť väčšie komponenty na menšie a neskôr ich poskladať ich zavolaním, aby mohli byť znovu použiteľné. Mali by sa pomenovávať preto skôr z pohľadu funkcionality komponentu a nie podľa kontextu v ktorom sa používajú. Komponent môže byť aj funkcia aj trieda [15].



Obr. 7: Životnosť komponentov – fázy, cez ktoré prechádza komponent počas svojho „života“.

4.1.2 State a props

Parametre pre komponenty nazývame props. Do komponentu môžeme poslať ľubovoľne veľa parametrov, ku ktorým ma komponent stále prístup pomocou this.props. Komponenty ale nikdy nemôžu meniť premennú props. Môžeme im nastaviť aj defaultné hodnoty pre prípad, keď niektoré parametre chceme až neskôr nastaviť.

State je podobný props premennej, ale je privátny (private), teda sú dostupné iba v rámci daného komponentu. Používa sa hlavne vtedy, keď zmeny daného parametra dávajú zmysel iba pre daný komponent. Pre znovu vykreslenie komponentu môžeme zmeniť premennú state pomocou setState().

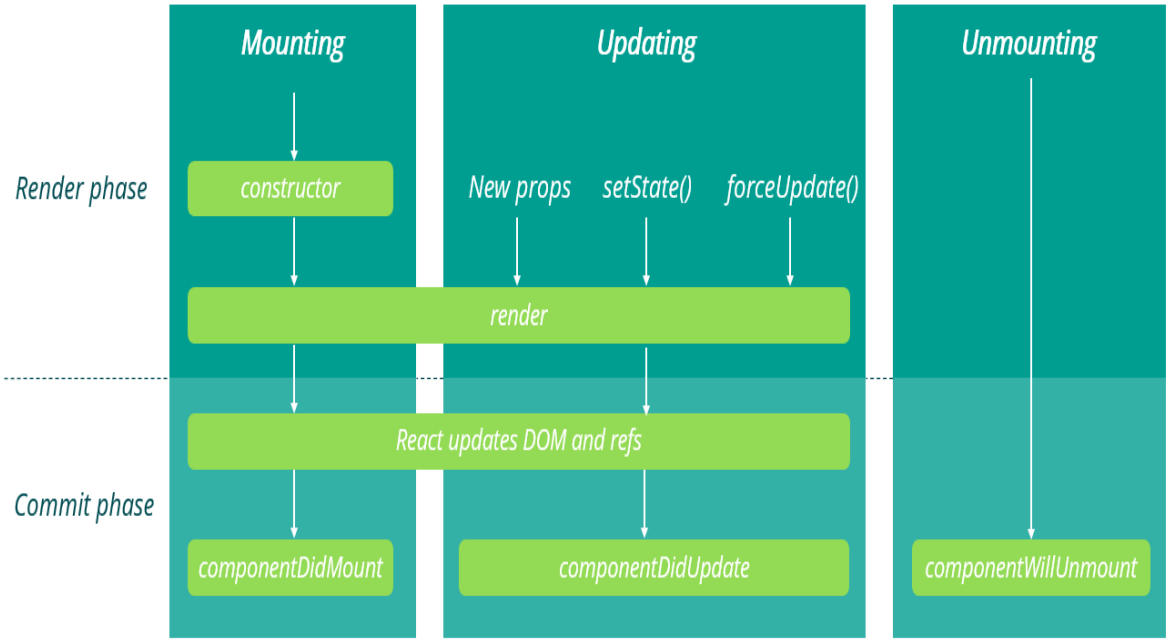
This.props a this.state môžu byť aktualizované asynchrónne, takže nemôžeme sa spoliehať na ich hodnoty pre výpočet nasledujúceho stavu. Preto treba použiť setState() radšej s parametrami state (predošlý state) a props (aktuálny). Ak state má viacero nezávislých premenných, tak iba tá premenná bude zmenená, pre ktorú bola volaná setState(). State je dostupný iba len pre ten komponent, ktorý to vlastní a nastavuje. Komponent môže poslať nižšie state ako props jeho dieťaťa. Dieťa nebude vedieť o tom, či je to state, props alebo či to bolo užívateľom zadané. Komponenty teda iba tie komponenty vedia ovplyvniť, ktoré sú pod nimi v strome [15].

4.1.3 Lifecycle metódy

Lifecycle metódy sú bežné funkcionality, ktoré sa vykonávajú počas rôznych fáz komponentu. Sú metódy, ktoré sú dostupné keď je komponent vytváraný („mounting“), keď je aktualizovaný alebo keď je odstránený („unmounting“).

Pri aplikáciách s veľkým množstvom komponentov je dôležité uvoľniť zdroje, keď komponenty sú zničené. Mounting je nastavenie komponentu, keď komponent je po prvý krát vykreslený v DOM. Unmounting je zmazanie komponentu, keď DOM vytvorený komponentom je odstránený.

Po tom, čo je výstup komponentuu prvýkrát vykreslení do DOM, je dostupná metóda componentDidMount. Po aktualizovaní komponentu je dostupná metóda componentDidUpdate(), ktorá je vhodná na prácu s DOM. Metóda componentWillUnmount() sa zavolá rovno pred tým, než sa komponent zmaže. Používa sa napríklad na odstránenie spojení alebo uvoľnenie pamäti.



Obr. 8: Lifecycle metódy rozdelené podľa fázy, v ktorom sa nachádza React a podľa úlohy metódy.

4.2 Redux

Je Javascript knižnica navrhnutá pre správu stavu aplikácie. Pomáha, aby sa aplikácia chovala konzistentne, fungovali v rôznych prostrediach a boli jednoduché na testovanie [16].

4.2.1 Princípy

Ak používame Redux, mali by sme dbať na to, aby sme dodržali nasledujúce tri princípy:

1. Jediný zdroj pravdy – Týmto zdrojom je Store. Je to ukladací priestor, kde je uložený stav celej aplikácie. Má stromovú štruktúru, aby sa jednoduchšie hľadali chyby v aplikácii. Taktiež zmeny ako undo a redo sú ľahšie implementovateľné.

2. Stav je len na čítanie – Jediný spôsob ako zmeniť stav je pomocou odosielania akcií. To zaisťuje, že ani pohľad (view) ani sieťové spätné volania nebudú nikdy prepisovať priamo stav. Namiesto toho vyjadria zámer o zmene stavu. Keďže všetky zmeny sú centralizované a vykonávajú sa jeden po druhom v poradí, nie sú žiadne súbehy (race condition), na ktoré by ste si mali dávať pozor. Keďže akcie sú iba obyčajnými objektmi, môžu sa zaznamenávať, serializovať, uložiť a neskôr znovu prehrať na účely ladenia alebo testovania.

3. Zmeny sú vykonávané jednoduchými funkciami – Pre špecifikovanie ako sa zmení stavový strom akciami sa píše tzv. reducer. Je to jednoduchá funkcia, ktorá pomocou predošlého stavu a akcie vráti nasledujúci stav. Vráti sa nový stav ako objekt namiesto toho aby sa menil starý. Stačí začať s jedným reducer-om, no postupným zväčšovaním aplikácie je vhodné ho rozdeliť na menšie reducer-y, ktoré budú kontrolovať konkrétne časti stavového stromu. Tým, že sú to jednoduché funkcie, dá sa ľahko kontrolovať v akom poradí sú volané, môžeme poslať dodatočné údaje alebo vytvoriť znovu použiteľné reducer-y pre bežné úlohy ako napríklad stránkovanie.

4.2.2 Komponenty

Redux sa skladá z troch základných komponentov, ktorými sú:

1. **Akcia** – Ak chceme zmeniť dáta v store, vykonáme to pomocou akcií. Posielajú dáta z aplikácie do store. Pre store sú to jediným zdrojom informácií. Odosielajú sa pomocou funkcie store.dispatch(). Je to JavaScript objekt. Musí mať zadefinovaná vlastnosť „type“, ktorá naznačí aký typ akcie bude vykonaný. Tieto typy by mali byť typicky definované ako reťazové konštanty.

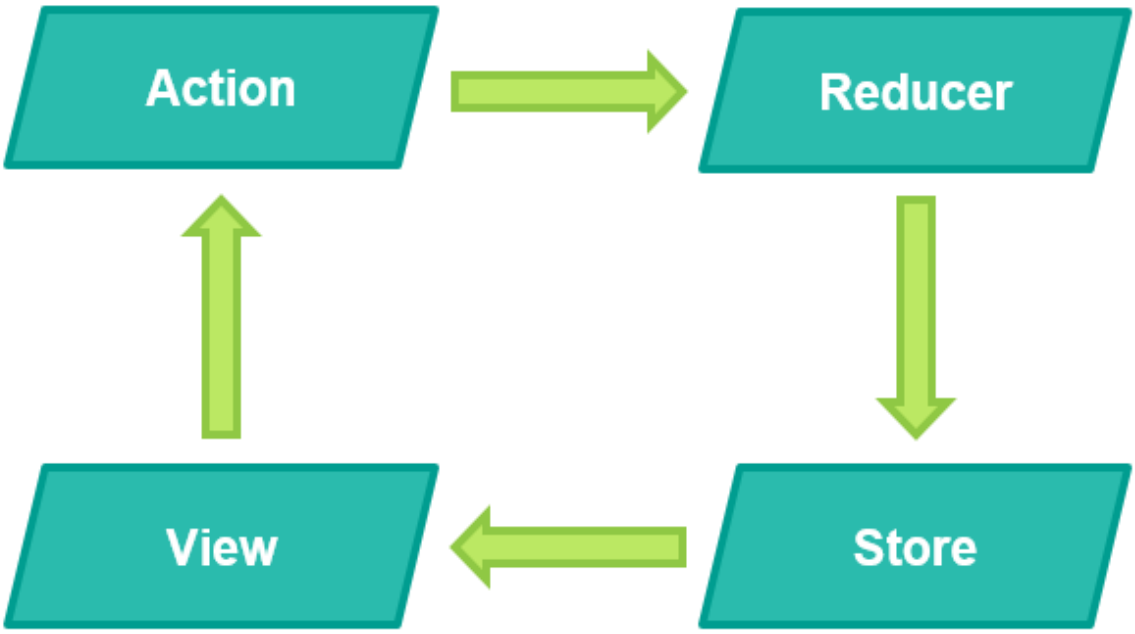
2. **Reducer** – špecifikuje ako bude stav aplikácie zmenený po akcii poslanej do store. Akcie popisujú iba to, že čo sa stalo a nie to, že ako sa zmenil stav aplikácie. Je to jednoduchá funkcia, ktorá zoberie predošlý stav, jednu akciu a vráti

nasledujúci stav ako nový objekt. V tejto funkcii by sme nemali meniť jej argumenty, vykonávať vedľajšie účinky ako API volania a volať nie jednoduché funkcie. Reducer je dobré rozdeliť na menšie funkcie, kde každá jedna funkcia spravuje inú časť globálneho stavu. Parameter „state“ je rôzny pre každý reducer a zodpovedajú za tú časť stavu, ktorý spravujú. Tieto menšie funkcie sa nakoniec spoja pomocou funkcie combineReducers(). Táto funkcia vygeneruje funkciu, ktorá zavolá všetky reducery s časťami state-u vybrané na základe ich kľúčov a skombinuje ich výsledky do jedného objektu znova. Tak isto ako reducer-y ani combineReducer() nevytvára nový objekt, ak poskytnuté reducer-y nemenili stav.

3. **Store** – je objekt, ktorý spája akcie a reducers. Má nasledujúce zodpovednosti: pamätá stav aplikácie, povoľuje prístup k stavu pomocou getState(), povoľuje aktualizovať stav pomocou dispatch(action), registruje a odhlasuje listenery pomocou subscribe(listener). V celej aplikácii je iba jeden Store. Aj keby sme chceli rozdeliť logiku spravovania údajov, tak použijeme kompozíciu reducer-ov namiesto viacerých store-ov. Store vytvárame pomocou funkcie createStore().

Použitím týchto troch komponentov vytvára jednosmerný tok údajov, pomocou ktorej je aplikácia viac predvídateľná a jednoduchšia na pochopenie. Je to predvídateľný kontajner, v zmysle, že dá sa predvídať čo bude každá akcia aplikácie robiť a taktiež aj to, ako sa zmení stav. Životný cyklus dát prechádza nasledujúcimi fázami:

- Zavolá sa funkcia store.dispatch(action)
- Store zavolá zadanú reducer function
- Hlavný reducer skombinuje výstupy viacerých reducer-ov do jedného stromu stavov.
- Store uloží celý strom stavov vrátený hlavným reducer-om. Tento strom sa stane nasledujúcim stavom aplikácie.



Obr. 9: Priebeh práce Redux-u a tok údajov.

4.2.3 Asynchrónne akcie

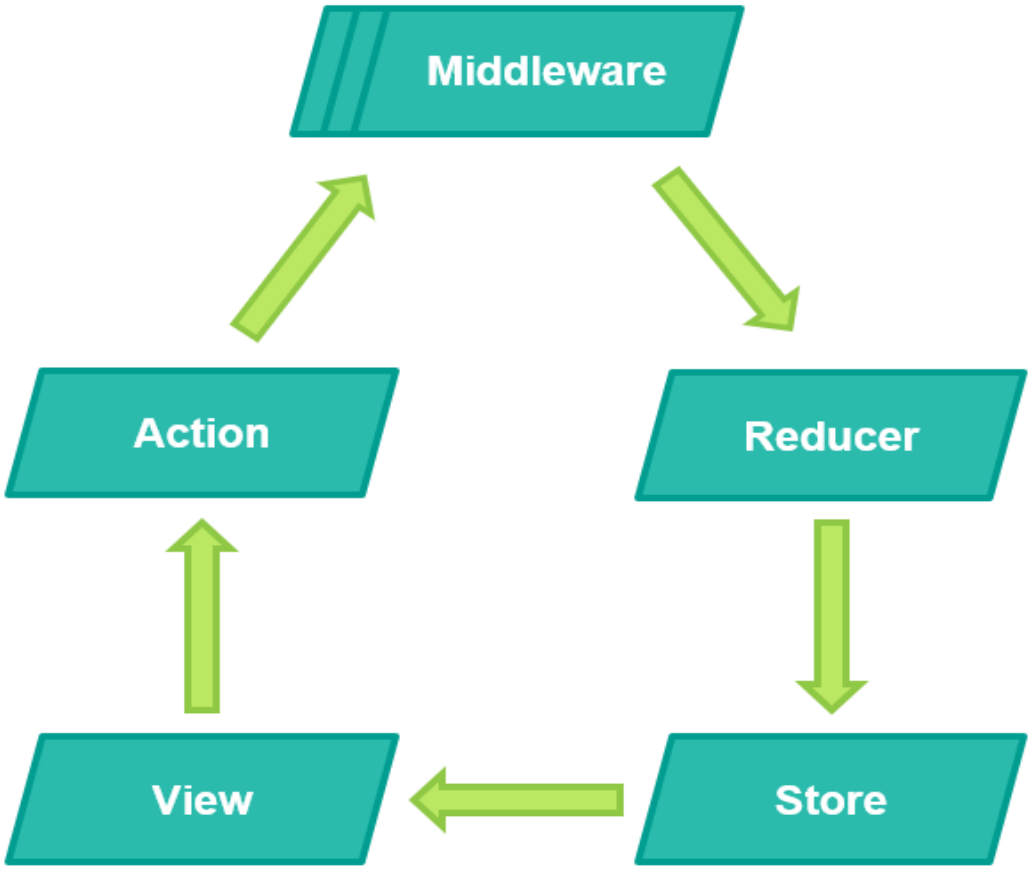
Aplikácie často sú asynchrónne. Pri volaní asynchrónnej API dve dôležité momenty nastávajú:

- moment, kedy začne volanie a
- moment, keď príde odpoveď.

Tieto momenty väčšinou požadujú aj zmenu stavu aplikácie, čiže je potrebné zavolať akcie, ktoré budú synchrónne spracované reducer-om. Väčšinou sa zavolajú aspoň tieto tri rôzne akcie:

- Akcia, ktorá informuje reducer, že požiadavka sa začala. Reducer zvyčajne zmení stav aplikácie a na užívateľskom rozhraní sa zobrazí načítavacím krúžkom.
- Akcia, ktorá oznámi reducer-u, že požiadavka bola úspešná. Reducer zvyčajne získané dáta uloží a odstráni stav naznačujúci čakanie odpovede. Na užívateľskom rozhraní sa to väčšinou zobrazí vykreslením týchto dát.
- Akcia, ktorá oznámi reducer-u, že požiadavka nebola úspešná. Reducer zvyčajne odstráni stav indikujúci čakanie odpovede. Môže uložiť chybovú správu, aby to mohol zobraziť na užívateľskom rozhraní.

Ak chceme, aby store podporoval aj asynchrónny tok údajov, tak potrebujeme použiť middleware. Middleware poskytuje rozšírenie medzi bodmi, kedy je akcia zavolaná a kedy sa dostane do reducera. Medzi týmito bodmi môžeme vložiť aj viaceré rôzne middleware-y (viď. obrázok 10). Tieto middleware-y majú rôzne funkcie, môžu slúžiť napríklad na jednoduchšie zaznamenávanie vykonaných akcií a zmien stavu, komunikáciu s asynchrónnymi API, manažovanie vedľajších účinkov, čítanie histórie zo Store a pod.



Obr. 10: Priebeh práce Redux-u pri použití viacerých middleware-ov.

4.3 JSON Web Token

JWT je norma, ktorá definuje spôsob bezpečného prenosu informácií medzi dvoma stranami ako objekt JSON. Tieto informácie môžu byť overené a sú dôveryhodné, lebo je digitálne podpísaný. Tokeny môžu byť podpísané pomocou tajného kľúča alebo dvojice verejného/súkromného kľúča. Tento objekt obsahuje všetky potrebné informácie na povolenie alebo odmietnutie danej API požiadavky [2].

4.3.1 Použitie

JSON Web Tokeny majú dve hlavné využitia, ktorými sú:

1. **Autorizácia** - je jedným z najčastejších prípadov, kedy sa to používa. Po tom čo sa používateľ úspešne prihlási, každá nasledujúca žiadosť bude obsahovať tento token. Tým umožní vykonávať kroky, ktoré sú dostupné iba pomocou toho tokenu.

2. **Výmena informácií** - keďže je to dobrý spôsob ako bezpečne posielať informácie medzi dvoma stranami. Keďže môžu byť digitálne podpísané, môžeme si byť istý, že odosielajúcim je naozaj ten, ktorým tvrdí, že je. Navyše môžeme zistiť aj to, či obsah nebol zmenený, keďže podpis sa počíta z hlavičky a dátovej časti objektu.

4.3.2 Štruktúra

JWT sa skladá z troch častí oddelené bodkami. Tými sú:

1. **Hlavička** - typicky sa skladá z dvoch častí:

   (a) typ tokenu, čo je JWT a

   (b) algoritmus, ktorý je použitý pri digitálnom podpise.

   Následne tento JSON je zakódovaný pomocou Base64 kódovania, aby tak tvoril prvú časť JWT.

2. **Dáta** - tvoria druhú časť tokenu. Obsahuje najmä informácie o užívateľovi nazývanie aj ako tvrdenia (claims). Tieto tvrdenia môžu byť typu registrované, verejné alebo súkromné. Tento JSON je taktiež zakódovaný pomocou Base64 kódovania.

3. **Podpis** - pre jeho vytvorenie je potrebné mať zakódovanú hlavičku, zakódovanú dátovú časť, kľúč, algoritmus, ktorý je definovaný v hlavičke a pomocou nich to môžeme podpísať. Podpis, pri ktorom chceme použiť algoritmus HMAC SHA256 vyzerá teda nasledovne:

```
HMACSHA256(
  base64UrlEncode(header) + "."+
  base64UrlEncode(payload),
  secret
)
```

4.4 Spring

Je aplikačný rámec a kontajner pre Java platformu [6]. Jeho hlavné funkcie teda môžu byť používané hocijakými Java aplikáciami. Má viaceré výhody ako napríklad ľahšie testovanie, možnosť použitia POJO (Plain old Java object), poskytuje predefinované šablóny, rôzne možnosti vytvárania aplikácii vďaka vkladaniu závislostí (dependency injection) a iné.

4.4.1 Architektura

Spring je modulárny, čo znamená, že môžu sa využívať iba niektoré časti rámca, ktoré sú potrebné pre danú aplikáciu. Ostatné moduly môžu byť úplne vynechané. Moduly, z ktorých sa skladá Spring sú rozdelené do nasledujúcich skupín:

- Spring Core - je najdôležitejšou častou. Poskytuje vkladanie závislostí a obsahuje BeanFactory, čo je implementácia návrhového vzoru factory.

- Spring AOP (Aspect Oriented Programming) - je modul, ktorý implementuje podporu pre aspektovo orientované programovanie. Umožňuje rozdeliť kód

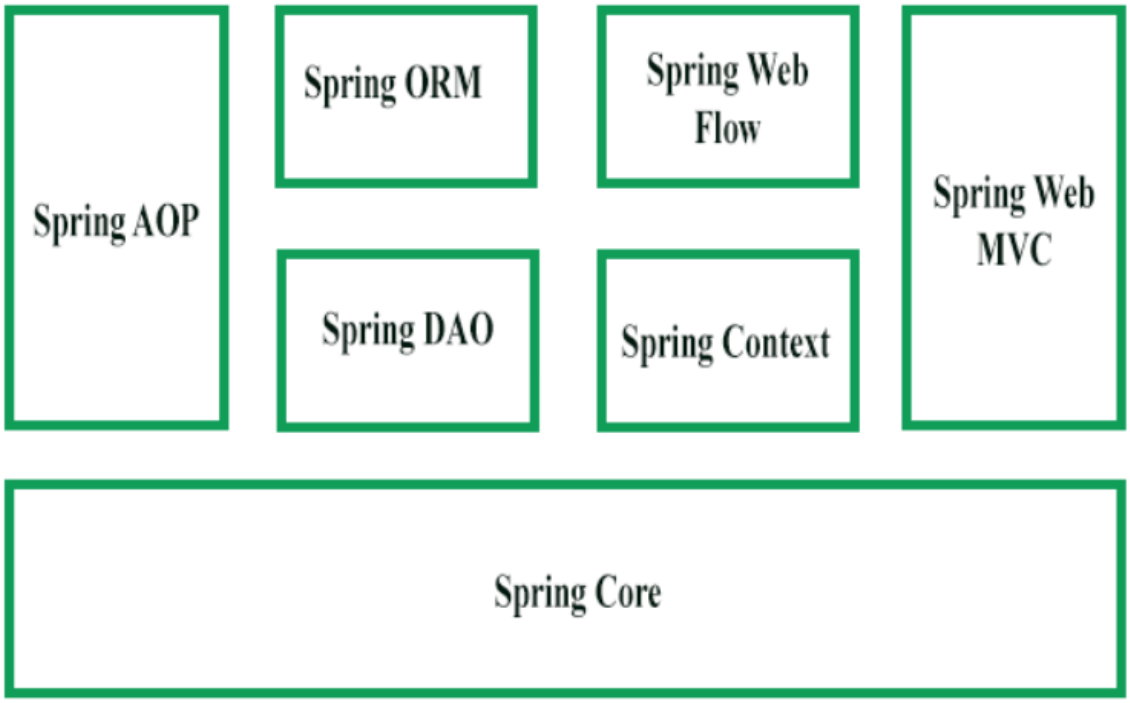
- Spring ORM (Object Relational mapping) - poskytuje integračné vrstvy pre populárne objektovo relačného mapovanie API

- Spring Web MVC (Model-View-Controller) - obsahuje moduly pre uľahčenie tvorby webových stránok. Rozdeľuje kód pre model a pre pohľad

- Spring Web Flow - je rozšírením pre moduly nachádzajúce sa v predošlej skupine. Pomáha hlavne pri definovaní Java tried, ktoré spravujú priebeh práce medzi rôznymi stránkami webovej aplikácie.

- Spring Web DAO (Data access object) - poskytuje abstrakčnú vrstvu pre úlohy týkajúce vytvárania, ukončenia spojenia s databázou a pod.

- Spring Application context - je založený na Core moduloch. Slúži ako médium pre prístup k všetkým definovaným a nakonfigurovaným objektom.



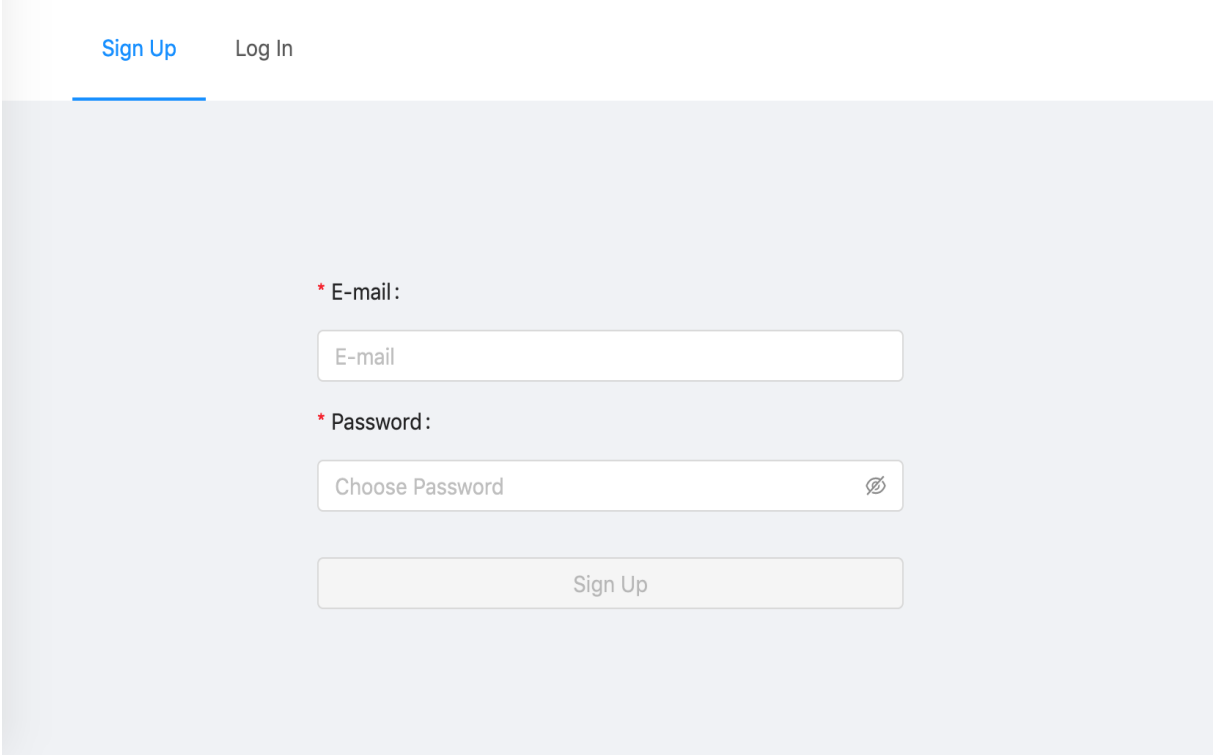
Obr. 11: Skupiny, do ktorých sú rozdelené všetky moduly Spring-u.

5 Implementácia

V tejto kapitole sa nachádza stručne opísaná implementácia. Pre jasnejšie vysvetlenie najprv slovne opíšeme priebeh hry a následne implementáciu.

5.1 Priebeh implementovanej hry

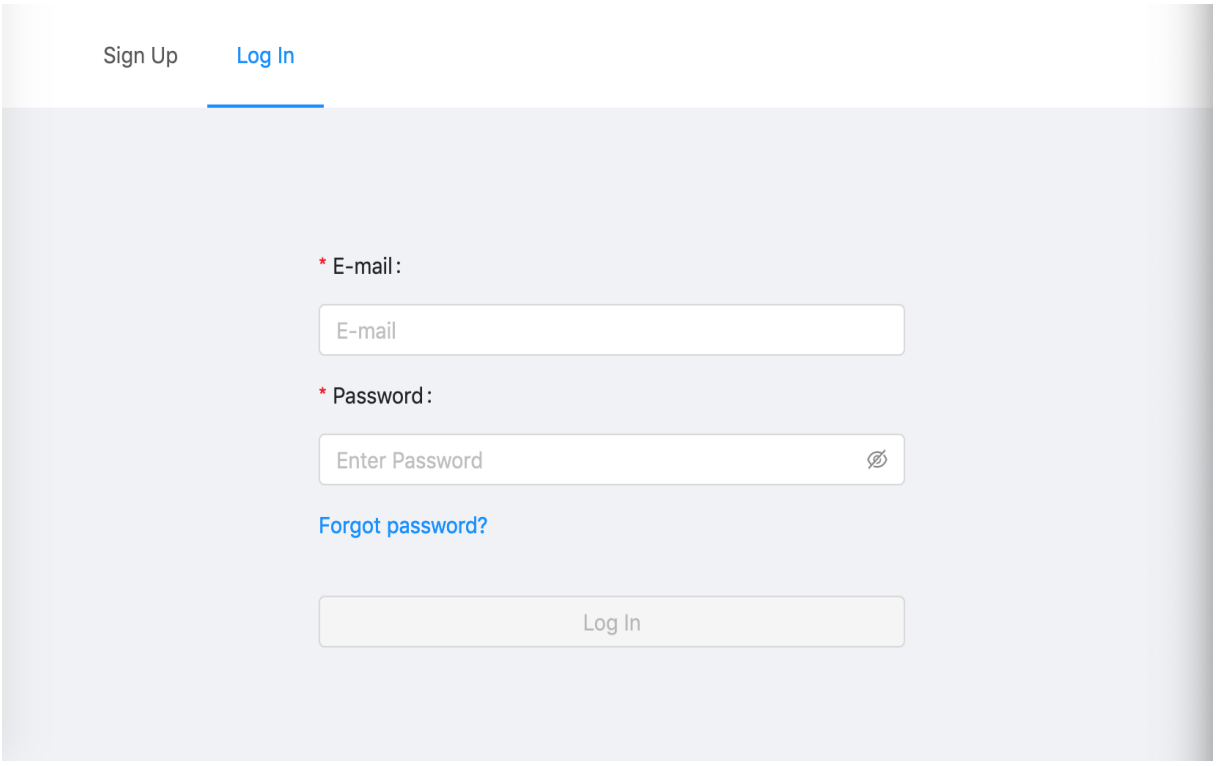
Hráč po otvorení stránky má na výber sa zaregistrovať alebo prihlásiť, ak už má vytvorené konto. Ak sa hráč chce zaregistrovať, je potrebné aby vyplnil registračný formulár so základnými údajmi ako emailová adresa a heslo (viď. obrázok 12).



Obr. 12: Registrácia hráča.

Používateľ následne dostane na zadanú emailovú adresu overovací token vo forme linku. Po kliknutí na daný link je používateľ presmerovaný na stránku hry a už bude

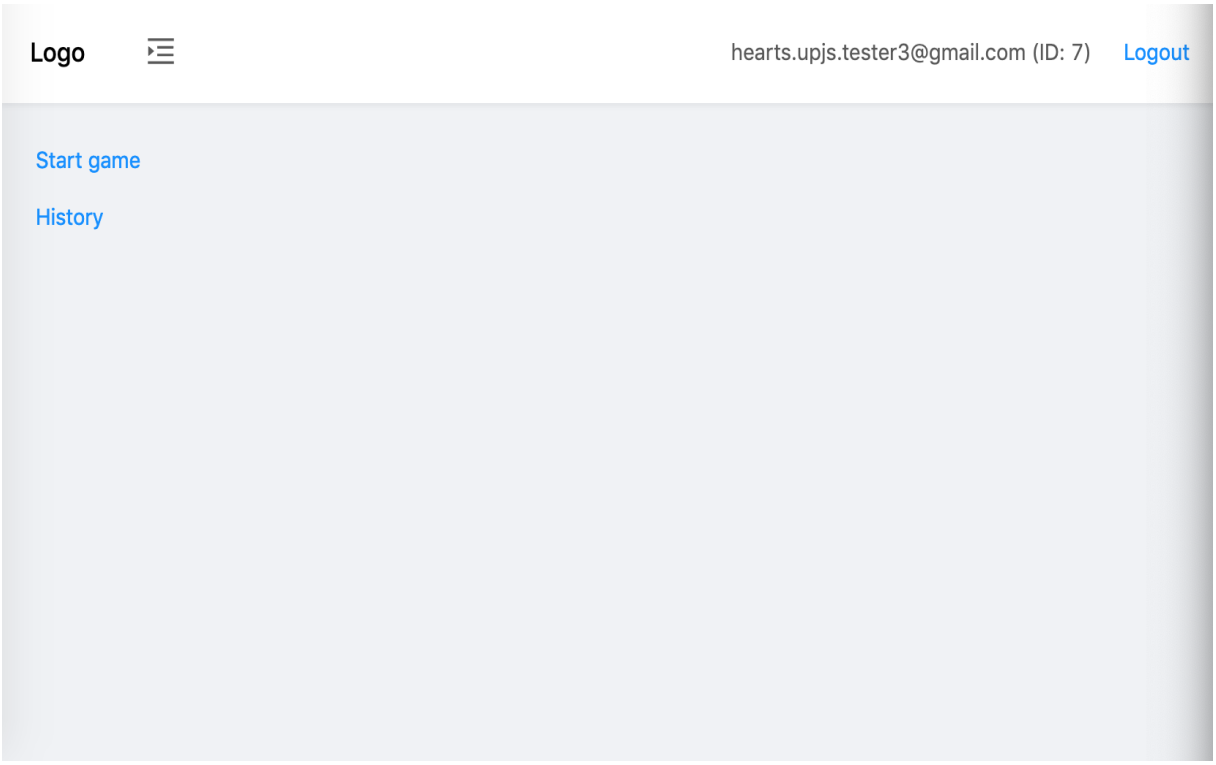
prihlásený. Následne sa môže prihlasovať klasicky už iba pomocou emailovej adresy a hesla (viď. obrázok 13).



Obr. 13: Prihlasovanie hráča.

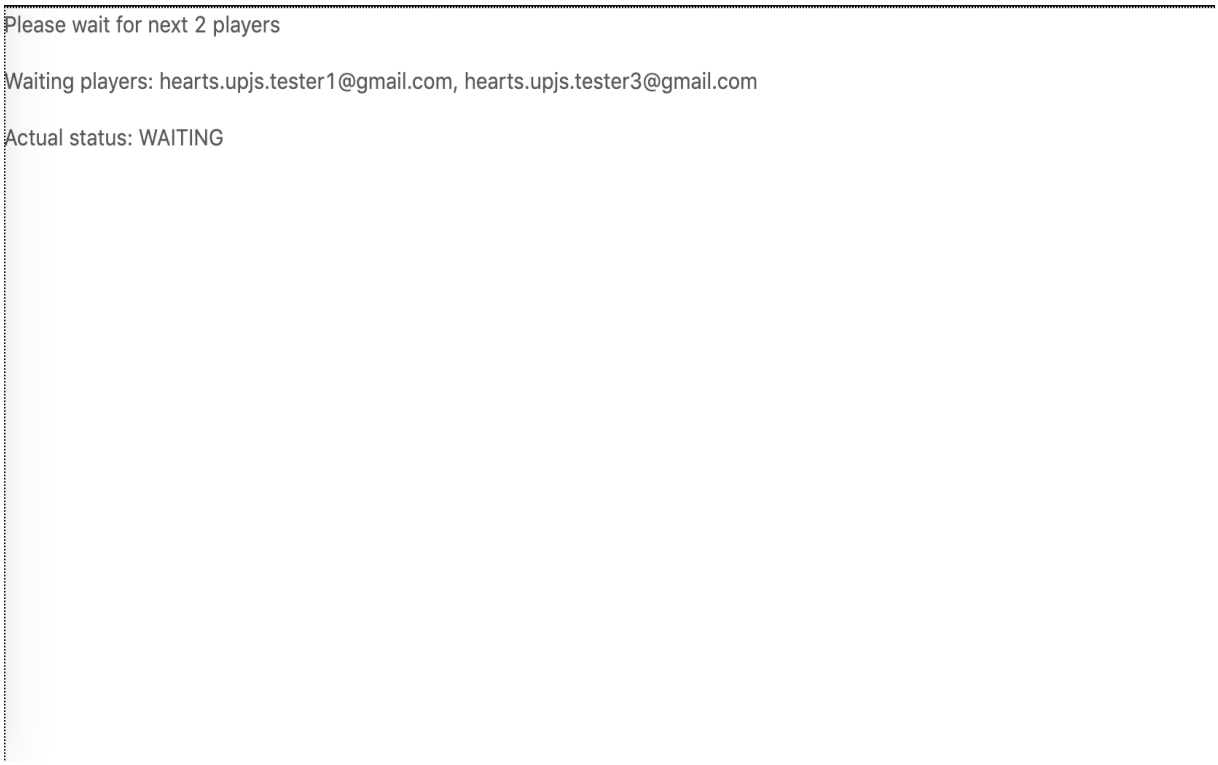
Ak užívateľ klikne na daný overovací link až po expirácii platnosti, tak nebude zaregistrovaný a bude musieť znovu zadať údaje.

Po prihlásení sa užívateľ dostane do menu. Tu môže vidieť históriu odohraných hier s výsledkami. V menu sa nachádza aj tlačítko pre začatie hry (viď. obrázok 14).



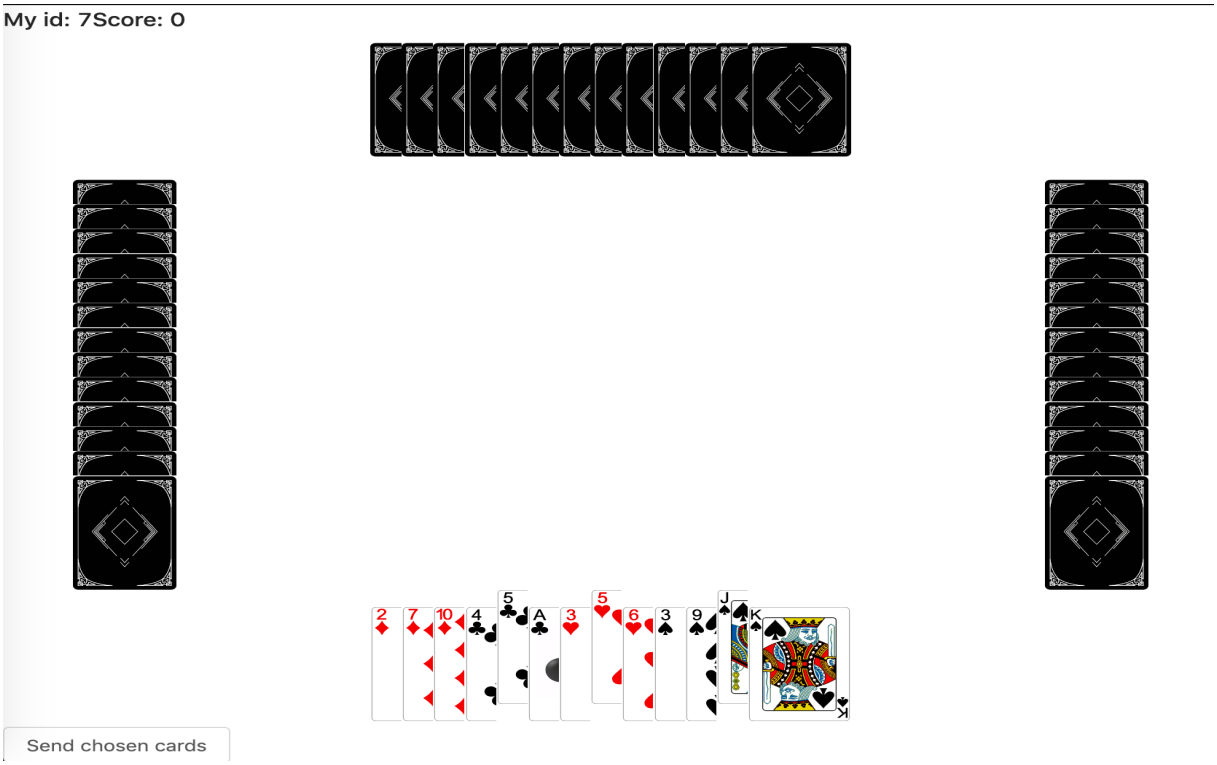
Obr. 14: Menu po prihlásení hráča.

Po stlačení hráč je presmerovaný do čakacej izby, kde čaká na pripojenie ostatných súperov (viď. obrázok 15). Ak v čakacej izbe sa už nachádzajú štyria hráči, tak server ich spojí a vytvorí im izbu pre hru. Od tohto momentu server slúži iba ako broker, čiže iba na preposielanie správ.



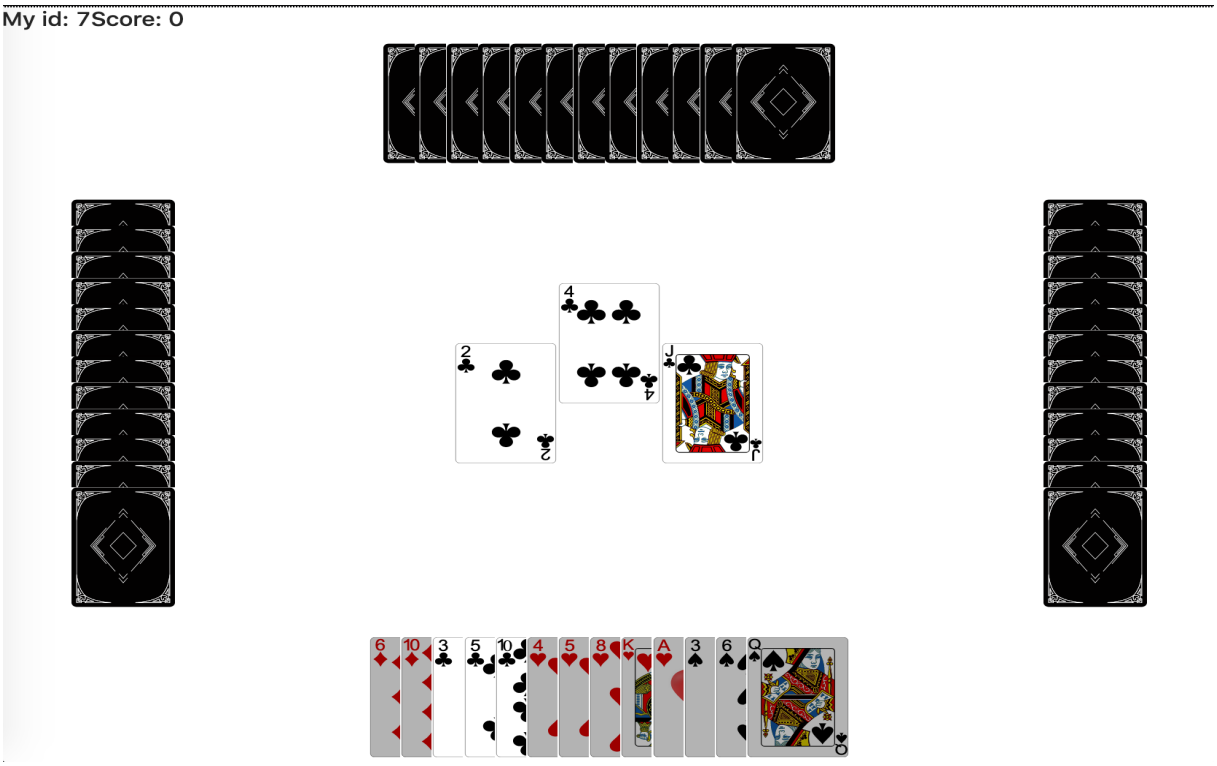
Obr. 15: Čakacia izba po stlačení tlačidla Start game.

Pred začiatkom hry sa zamiešajú karty a každý hráč dostane 13 kariet. Z týchto 13 kariet každý hráč si musí ešte vybrať 3. Tieto karty pomocou tlačidla "Send chosen cards"pošle hráčovi, ktorý sa nachádza naľavo od neho pri stole (manual shuffle) (viď. obrázok 16).



Obr. 16: Posielanie troch kariet susedovi.

Hru začína hráč, ktorý vlastní pikovú dvojku. Následne hra prebieha podľa vybraných pravidiel popísaných v kapitole 3.1. Pre zjednodušenie hry sú stmavené tie karty, ktoré nemôžu byť použité v danom ťahu. Priebežné výsledky hry sú taktiež zobrazené (viď. obrázok 17).



Obr. 17: Hráč má stmavené tie karty, ktoré v danom ťahu nemôže použiť.

Na konci je hráč presmerovaný naspäť do menu, kde už môže vidieť v histórii výsledky hier.

Záver

V tomto článku sme opísali problematiku týkajúcu sa bezpečnosti v online hrách. Súčasťou práce je aj vytvorenie hry, pomocou ktorej ukážeme ako by sa dal riešiť tento problém. Analyzovali sme možné útoky, ktoré sa môžu vyskytnúť počas celej hry a snažili sme sa navrhnúť rôzne metódy ako zabezpečiť hry. V tomto článku sme sa pozreli aj na podobné práce, kde sa tiež zaoberali bezpečnosťou v online hrách. V týchto prácach ale skôr išlo iba o kategorizáciu hrozieb.

Ďalším krokom je implementácia hry v prostredí React a výber autentifikačných a bezpečnostných protokolov pre jednotlivé body v hre. Počas výberu budeme analyzovať najčastejšie používané protokoly v hrách.

Zoznam použitej literatúry

[1] Boduch, A. *React and React Native*. Packt Publishing Ltd, 2017.

[2] Bradley, J., Sakimura, N., and Jones, M. B. Json web token (jwt).

[3] Chen, Y.-C., Chen, P. S., Hwang, J.-J., Korba, L., Song, R., and Yee, G. An analysis of online gaming crime characteristics. *Internet Research 15*, 3 (2005), 246–261.

[4] Du, W., and Atallah, M. J. Secure multi-party computation problems and their applications: a review and open problems. In *Proceedings of the 2001 workshop on New security paradigms* (2001), ACM, pp. 13–22.

[5] Jeff Yan, J., and Choi, H.-J. Security issues in online games. *The Electronic Library 20*, 2 (2002), 125–133.

[6] Johnson, R., Hoeller, J., Donald, K., Sampaleanu, C., Harrop, R., Risberg, T., Arendsen, A., Davison, D., Kopylenko, D., Pollack, M., et al. The spring framework–reference documentation. *interface 21* (2004), 27.

[7] Ki, J., Hee Cheon, J., Kang, J.-U., and Kim, D. Taxonomy of online game security. *The Electronic Library 22*, 1 (2004), 65–73.

[8] Law, M., and Deane, G. General card game playing. *Imperial College London* (2013).

[9] Stamer, H. Efficient electronic gambling: An extended implementation of the toolbox for mental card games. *WEWoRC 74* (2005).

[10] Woo, J., and Kim, H. K. Survey and research direction on online game security. In *Proceedings of the Workshop at SIGGRAPH Asia* (2012), ACM, pp. 19–25.

[11] Yan, J., and Randell, B. An investigation of cheating in online games. *IEEE Security & Privacy 7*, 3 (2009), 37–44.

[12] z: `https://bicyclecards.com/how-to-play/hearts/`, D. How to play hearts.

[13] Z: HTTPS://EN.WIKIBOOKS.ORG/WIKI/CARD_GAMES/HEARTS/STRATEGY, D. Hearts strategy.

[14] Z: HTTPS://EN.WIKIPEDIA.ORG/WIKI/HEARTS_(CARD_GAME), D. Hearts card game.

[15] Z: HTTPS://REACTJS.ORG/, D. React.

[16] Z: HTTPS://REDUX.JS.ORG/, D. Redux.

[17] Z: HTTPS://VIPHEARTS.COM/RULES/, D. Hearts rules.