Gymnázium, Praha 6, Arabská 14
předmět programování, vyučující Ing. Daniel Kahoun

Simulátor stolní karetní hry pro více hráčů na síti

ročníkový projekt

Jan Hlavnička, 2.F

2025/26

Abstrakt

Your abstract.

Poděkování

Chtěl bych poděkovat především mému vedoucímu práce ing. Danieli Kahounovi za vedení práce a vstřícnost při konzultacích. Dáloe bych chtěl poděkovat své rodině a kamarádům za to, že mojí hru spoustakrát vyzkoušeli a upozornili na její nedostatky.

Prohlášení

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne _____

_____

Jan Hlavnička

1 Úvod

Tady bude úvod

Obsah

Obsah

2 O hře Bang!

Dle wikipedie: Bang! je populární karetní hra na motivy Divokého západu. Hru v roce 2004 vytvořil ital Emiliano Sciarra, vydalo ji nakladatelství DV Giochi. Pro Českou repuliku *Bang!* vydává Albi. V České republice se stal velice oblíbená.

2.1 Pravidla hry

Hra implementuje upravenou verzi pravidel. Ta bude v této kapitole popsána. Celé originální pravidla jde najít (odkaz).

2.1.1 Příprava hry

Na začádku hry si každý hráč vybere jednu ze dvou vylosovaných postav. Tato postava mu zůstane po celou dobu hry. Postava má svůj efekt a počet životů. Hráč také dostane roli, kterou (pokud není šerif) ostatní nevidí. Každému hráči bude podle postavy přidán počet karet.

2.1.2 Cíl hry

Cíl hry má každý napsaný na své postavě – šerif s pomocníky musí zabít bandity, banditi šerifa a odpadlík všechny hráče. Hráč umírá tím, že mu dojdou životy a nemá žádné *pivo*.

2.1.3 Průběh kola

Začíná šerif a hráči se střídají v pořadí ve kterém se připojili. Kolo začíná dostáním dvou karet. V průběhu kola může hráč spálit, zahrát a vyložit libovolný počet karet. Jediná. Vyjímkou je karta *bang*, jejíž zahráním tah končí. Tah může také končit zmáčknutím **ukončit tah**.

3 Vyzkoušení hry

Kompletní hra včetně klienta, serveru a všech pluginů zmíněných v této práci je dostupná na webu bang.honzaa.cz. Zdrojové soubory jsou umístěním v GitHub repozitáři HonzaHlavnicka/Bang, kde také jdou přidávat *Issues* pro nalezené problémy.

4 Technologie

Na serveru používá Engine Javu 11, která byla vybraná, aby mohl server běžet i na školním serveru, ktrý novější nepodporoval. Java byla zvolena z důvodu, že je na Gymnáziju Arabská vyučována a protože z programovacích jazyků, které autor zná byla pro projekt nejideálnější.

Knihovny na serveru se používá převážně *Java websocket*, která řeší serverovou logiku pro připojení k websocketu a vytvoření websocketového serveru. Dále se používa *org.json* pro složitější práci s JSON objekty.

Na straně klienta se používá *React* s *Vite* a *TypeScriptem*. Nejdříve se používal *vanilla JavaScript*, který byl ovšem nevhodný kvůli těžší rozšiřitelnosti, modularitě s chybějící typové kontrole. Tato původní verze se v repozitáři stále nachází, jelikož v základu funguje. Z frameworků byl zvolen právě *React*, protože ho autor používal v té době na jiný projekt a s jiným neuměl pracovat. Navíc byl *React* pro tento typ projektu vhodný a poskytoval dostatečné funkce i moduly. Z modulů byl použit ———

5 Cíle práce

Cíle práce byli definovány hned na začádku projektu. Byli zvolený tak, aby obsahovali původní hlavní cí – vyrobit online *Bang!* – ale aby se zaměřil primárně na možnost vytvoření celého herního enginu, ne pouze jednoduché hry. Cíle byli formulovány takto:

Cíle a specifikace práce

- Vytvořit síťovou karetní hru s architekturou klient-server, kde server bude zajišťovat logiku hry a klient bude sloužit k interakci hráče.
- Naprogramovat server v jazyce Java, který bude mimo jiné:
  - spravovat herní logiku,
  - kontrolovat platnost akcí hráčů,
  - zajišťovat průběh hry a aplikaci pravidel,
  - komunikovat s klienty pomocí websocketového protokolu.
- Implementovat webového klienta, který poběží v prohlížeči a umožní hráčům:
  - zobrazit mimo jiné karty v ruce a na stole,
  - sledovat stav a informace o ostatních hráčích,
  - provádět akce a ovládat hru prostřednictvím uživatelského rozhraní,
  - a další potřebné informace pro průběh hry.
- Podporovat variaci karetní hry Bang! s možností rozšíření o další hry.
- Otestovat aplikaci s reálnými hráči a vyhodnotit její funkčnost i uživatelskou přívětivost.
- Vypracovat dokumentaci a návod k použití, včetně popisu architektury a zdrojového kódu.

6 Předem stanovená struktura práce

Již na začátku školního roku 2025–26 bylo potřeba stanovit si osnovu a průběhu práce. Byli zvoleny takto:

Průběh zpracování práce

1. vytvoření jednoduchého serveru
2. vytvoření jednoduchého klienta
3. přidání logiky a ovládacích prvků
4. testování hry s dalšími lidmi

5. opravení nedostatků na základě získaných podmětů
6. sepsání dokumentace a průběhu práce
7. přidání dodatečných méně potřebných funkcí
8. publikace práce

Osnova práce

- o hře Bang! (popřípadě jiných implementovaných hrách)
- návod k programu
- průběh tvorby
- dokumentace
- závěr

7 Práce se zdroji

Ve stejné době bylo třeba deklarovat jaké zdroje budou pro projekt použity. Deklarace zněla takto:

> K práci budu používat oficiální pravidla hry, dokumentaci Javy (docs.oracle.com), dokumentaci Reactu (react.dev) a stránky w3schools.com. Dále pro řešení problémů využiji internetových webových stránek jako jsou například stackoverflow, nástrojů umělé inteligence (chatGPT, copilot) a rad od lidí, kteří mají s problémem zkušenost včetně vedoucího práce pana Ing. Daniela Kahouna.

8

9 Implementace serveru

Server se dělí do několika balíčků a projektů. Hlavní balíček je `bang`, který propojuje `sdk` se `server`em. Sdk je klíčová součást celého projektu, protože umožňuje pluginům nebýt závislých na konkrétní implementaci serveru a pracovat jen s interfaci. Balíček `server` je poté konkrétní implementací servových tříd.

9.1 `KomunikatorHry` a `SocketServer`

Pro komunikaci s klienty se používá websocket. Pro jeho správu v javě se používá knihovna Java websocket. Díky této knihovny stačí dědit třídu WebsocketServer a Implementovat metody jejích akcí. To dělá právě třída `SocketServer`. Třída obsluhuje zachycení výjimek, vytvoření her, načítání pluginů a komunikaci s hráči na mimoherní úrovni. Tou je myšleno stahování informací o instalovaných pluginech, získávání informací o serveru a podobně.

Nejduležitější metodou je `onMessage`, která zpracovává příslušné zprávy. Dokumentace celého protokolu a všech zpráv je dostupná v dokumentech na GitHubu. Tato metoda umí vytvářet hry a spojovat zprávy s příslušními komunikátoryHer, které si drží v mapě `komunikatoryHracu`.

`KomunikatorHry` je třída, která spojuje `SocketServer` a samotnou hru. Drží si seznami hráčů, tokeny hráčů pro připojení, mapy hráčů se sockety a podobně. Je používaná k tomu, aby mohli servové nebo pluginové metody kominikovat s klientem. Nemusí k tomu znát přímo `Websocket` objekt, protže ten si spojí v metodě `posli()` sám komunikátor. Komunikátor také zpracovává události posílané klientem přes `SocketServer`. Tyto události rovnou předává do metod hráče, kterého se týkají.

10 interface pro balíčky `HerniPlugin` a `HerniPravidla`

Engine je navrhnutý tak, že v základu neobsahuje žádnou hru, ale pouze obecné ulity a funkce. Z toho důvodu je potřeba, aby hra načítala jednotlivé pluginy. To zajišťují třídy `NacitacPliuginu` a `SpravceHernichPravidel`. Aby tyto třídy fungovaly, tak musí každý plugin implementovat `HerniPlugin`. Ten vrací název hry, popis hry, odkaz na pravidla a hlavně metoda, která vytvoří `HerniPravidla`. `HerniPravidla` jsou hlavní třída pluginu, jejíchž metody hra volá, aby se chovala jak autor pluginu očekává. Mezi některé její metody, které může autor používat jsou:

10.0.1 `void pripravBalicek(Balicek<Karta> balicek)`

Tato metoda se volá při přípravě hry a měla by naplnit dobírací balíček `Karta`mi.

10.0.2 `public UIPrvek[] getViditelnePrvky()`

`GetviditelneMetody` je metoda, která vrací seznam prvků, které hráči mohou v klientovy vidět. Přestože by měl být server oddělený od logiky klienta, tak je tato metoda potřebná, aby například ve hře, která nepoužívá princip životů nezabíraly na obrazovce místo ukazatele životů.

10.0.3 `boolean hracChceUkoncitTah(Hrac kdo)`

Metoda, která se volá poté, co hráč klikne na tlačítko ukončit tah. Vrací boolean, který určuje, zda hráč tah ukončit může nebo ne. Spousta her hráči neumožní ukončovat tah když chce, protože hráč musí například odhodit kartu, proto by tyto hry měli v metodě `return false`. Jiné hry, ale nechají hráče udělat kolik akcí chce a proto by jen například kontrolovaly jestli nemá hráč v ruce kartu, kterou před koncem tahu musí odhodit.

10.1 `zacalTah()`

`zacalTah()` je metoda, ve které si může plugin připravit vše, co se odehrává na začátku každého tahu.

Kromě těchto metod třída další, které jsou ale pricipiálně podobné a nemá cenu je zde vysvětlovat. Kompletní seznam možností jde najít v javadoc dokumentaci.

10.2 Karta

Karta je abstraktní třída, jejíchž objekt reprezentuje jednu kontrétní kartu ve hře. Každá karta musí mít jméno a identifikátor obrázku (v sočastnosti cesta k obrázku bez přípony souboru). Každému objektu karty je automaticky přiřazeno unikátní id, které umožňuje její identifikaci v rámci serveru.

Karta sama o sobě nejde zahrát ani vyložit. K tomu poslouží interfaci `HratelnaKarta` a `VylozitelnaKarta`.

10.2.1 `HratelnaKarta`

`HratelnaKarta` je interface, který musí implementovat všechny karty, které může hráč zahrát. Obsahuje metodu `boolean odehrat(Hrac kdo)`, která se volá, když se hráč pokusí kartu zahrát. Vrací boolean zda se karta může zahrát (a rovnou již proběhl její efekt), nebo se v současném kontextu zahrát nemůže.

10.2.2 `VylozitelnaKarta`

`VylozitelnaKarta` je interface, který musí implementovat všechny karty, které může hráč vyložit. Obsahuje metodu `boolean vylozit(Hrac kdo,Hrac kym)`, která se volá, když se hráč pokusí kartu vyložit. Vrací boolean zda se karta může vyložit (a rovnou se k tomu připravý všechny náležitosti), nebo se v současném kontextu vyložit nemůže. Pokud vrátí true, tak je na VylozitelnouKartu zavolána ještě její metoda `getEfekt()`, která vrací objekt implementující interface `Efekt`. Ve většině případů se jedná o objekt karty, ale autor pluginu se může rozhodnout, že logiku karty a efektu bude sržet odděleně, například pokud stejné efekty používá i nějaká postava.

11 `NacitacPluginu` a `SpravceHernichPravidel`

12 Implementace pluginů

K enginu bylo vyrobeno několik pluginů – Bang, prší a UNO. Pluginy jsou v repozitáři ve složce `/pluginy/`.

12.1 Bang!

Tento plugin je pro projekt klíčový, jelikož je engine přízpůsobený hlavně jemu. V původní fázi byl dokonce do enginu hardcodován. Teď už je většina jeho prvků přesunuta do jeho herních pravidel.

V Bangu bylo potřeba zařídit, aby měl správně nakonfigurované pravidla. Dále bylo potřeba Implementovat všechny karty, efekty a postavy.

12.2 Prší

Prší je hra, ktera byla kompletně přidána jako první a používala se v pro testování. Karty byli stažený z webu ()doplnit zdroj.

Implementace karet byla jednoduchá. Existuje základní karta `PrsiKarta`, která má svoji barvu () a hodnotu (). Karta je Implementace interface `HratelnaKarta`. Metoda volání při odebrání neprovádí žádný speciální efekt, pouze kontroluje předchozí kartu v odhalování balíčku, zda se shoduje hodnotou, barvou, či zda vůbec není instancí Prší karty (To by nastat nemělo, ale kdyby nějaký balíček kombinovat karty, tak se hodí mít definované chování.

Tuto základní kartu implementuje PrsiSedmicka. Ta se liší v tom, že následující hráč si musí lízat, či sedmičku přebít. Naprogramovat tuto kartu samostatně nejde, protože mění stav hry. Z toho důvodu bylo třeba do herních pravidel přidat oro sedmičku metodu. (doplnit)

Dalším dítětem Prší karty je `PrsiSvrsek`. Ten se liší tomu, že jde vložit nad libovolnou kartu, tudíž jeho metoda `odehrat()` vrátíte vždy true. Než to ale udělá, tak u hráče co ji zahrál vyvolá dialog na výběr barvy. Do té doby, než vybere barvu, tak jeho getter pro ostatní karty vrací `null`. Poté co hráč vybere barvu, tak se svršek změní tu svoji "falešnou"barvu na barvu, kterou hráč vybral. Také se ostatním pošle upozornění, jaká barva byla vybraná.

Poslední speciální kartou je `PrsiEso`. Jelikož autor hry naprogramoval prší podle pravid3l, které používá on, tak se eso nepřebíjí. Z toho důvodu karta po zkontrolování podmínek zahrání pouze zavolá `hra.getSpravceTahu().eso()`, jelikož eso je funkce v enginu zabudovaná.

12.3 Uno

Poslední implementovaná hra je únor, které je pouze starší verzí prší, bez další složité naprogramované mechaniky. Nejrozlišnější karta je změna směru, ale její funkčnost je zabudovaná ve správci tahu.

13 Implementace klienta

13.1 Struktura

13.2 O aplikaci

Jak již bylo zmíněno, tak klient je napsán pro framework *React* pro web. Obsahuje hlavní soubor `main.tsx`, který obsahuje různé providery a celoaplikacové komponenty. (zde bude odkaz na moduly) Hlavní komponenotou celé aplikace je `App`, která přepíná mezi stránky a řeší lazyloading. Jednotlivé stránky se nacházejí v `/src/pages/`.

Hlavní kontext, který obsahuje jak funkce pro celou aplikaci a všechny data je `GameKontext`. Ten obsahuje jednotlivé funkce pro posílání dat na sever, úpravu dat, akce hráče a podobně. Jeho provider se jmenuje `GameProvider`. V něm nejsou implementovány tyto funkce, ty jsou spolu s logikou příchozích zpráv přemístěny do `gameActions`. Provider ovšem obsahuje efekt, pomocí kterého se připojí k websocket serveru.

13.3 Připojovací stránkka `LoginPage`

Tato stránka by pro uživatele měla být první stránkou, kterou uvidí. Obsahuje nějaké základní informace o hře a formuláře pro připojení se. Tyto formuláře existují tři – první z nich slouží pro připojování se ke hře a zobrazí se jen pokud má hráč v `localStorage` uložený token hry, který mu je přidělen v průběhu připojování se. Druhý slouží k připojení se pomocí kódu hry a jména a třetí slouží k vytvoření hry. U vytvoření hry je vybírací nabídka dostupných pluginů, která se musí nejdříve dostat od serveru, aby se mohli zobrazovat názvy her, popisy a pravidla.

13.4 Hlavní objekt `gamestate`

GameState je objekt držený v `GameKontextu`, který obsahuje všechny informace o probíhající hře. Objekt též obsahuje informaci zda je hráč připojen ke hře (`inGame`), zda je hra spuštěna (`gameStarted`) a seznam pluginů, které jsou na serveru nainstalované, včetně jejich metadat (`gameTypesAvailable`).

13.5 Herní stránka `GamePage`

Herní stránka je stránka, která se zobrazuje, když je hráč připojen ke hře a hra je spuštěna. Obsahuje všechny herní prvky, které jsou zobrazené hráči. Kromě prvků obsahuje utility drag & drop kontext `DndContext` a `<GlobalNotifications />`.

Stránka se dělí do tří částí pod sebou – panel hráčů `Players` s informacemi ke každému hráči, centrální panel `CentralPanel` s kartami na stole a dalšími prvky společné pro všechny hráče a panel s prvky související pouze s hrajícím hráčem `MyThings`.

13.6 Drag & drop

Hra používá drag & drop logiku pro přesun karet, díky kterým může hráč jedním pohybem určit jak co a kam chce předělat mnohem přirozeněji než několika kliknutími. Pro tento účel se používá knihovna *react-dnd*. Původně bylo učiněno několik pokusů o vlastní implementaci, ale nakonec se ukázalo, že pro tento projekt je lepší použít hotové řešení, které je dobře zdokumentované, podporované a obsahuje řešení všech hraničních případů i typů uživatelského vstupu.

13.7 Notifikace

Hra používá více systémů notifikací. Prní z nich jsou toast notifikace v pravém horním rohu pro informování hráče. Používají se, když se stane nějaká událost, která nemusí být jasně patrná z drobné změny UI, která proběhla. Druhým typem jsou notifikace, které se zobrazují přímo uprostřed. Cílem těchto notifikací je upozornit na sebe i když hráč například nedávvá pozor. Tyto notifikace se používají například pro upozornění na začátek tahu.

Pro toasty se používá knihovna *react-hot-toast*, která poskytuje jednoduché API pro zobrazení toastu. Knihovna byla vybrána i přes to, že by šla tato funkčnost vyrobyt bez ní, ale nebyl důvod „znovu vynalézat kolo“. Toast jde po importu vyvolat pomocí funkce `toast()`, která má několik variant pro různé typy notifikací, například `toast.error()` pro chybové notifikace.

Pro notifikace uprostřed se používá vlastní komponenta `CentralNotification`, která zobrazuje zprávu jednoduchý text, který se uprostřed obrazovky začnezvětšovat a zprůhledňovat. Komponenta využívá portál, kterým do `<body>` stránky vkládá absolutně pozicovaný element, do kterého vykresluje samotnou notifikaci. Po importu komponenty jde notifikace vyvolat metodou `notify()`.

Pokud je třeba hráče informovat tak, že musí přečtení potvrdit, tak se použije dialog.

13.8 Dialog

Dialog je komponeta se svím kontextem, která poskytuje api pro informování hráče a získávání vstupu od něj. Skádá se z `DialogProvider`u, který vyrábí kontext a poskytuje metody `openDialog` a `closeDialog`. `DialogProvider` vrací kontext `DialogContext`, který exportuje tyto metody a potřebné typy.

Samotná komponenta `Dialog` používá kontext dialogu a v případě otevření vytváří portál, ve kterém dává obal dialogu, ztmavené pozadí a samotný obsah dialogu. Ten se určí podle typu dialogu. Existují typy `SELECT_CARD`, `SELECT_PLAYER`, `CONFIRM` a `INFO`.

Dialog má různé možnosti, například zda se může zavřít bez akce, kolik položek může maximáálně i minimálně vybrat a podobně.

13.9 Tmavý režim

Celý web je vyroben spodporou tmavého režimu. Pro tmavý režim se nepoužívá `prefers--color-scheme`,aby uživatelé mohli používat přepínač v aplikaci. Stav tmavého režimu ovládá přidáváním třídy `darkMode` na `<body>` element. Nepoužívá se kontext, aby se nemusel překreslovat framework a všechny komponenty, ale pouze css interně v prohlížeči. V `.module.css` souborech se pak používá `:global(.darkMode)` pro funkčnost i po kompilaci, která by jinak změnila názvy tříd.

Přepínač je komponenta `DarkModeSwitch`, která se vykresluje jako měsíček či sluníčko v pravém horním rohu. Tento přepínač se neukazuje na normální hrací stránce, kde by byl zablokovaný, ale režim jde přepínat například při otevřeném dialogu.

13.10 Server info

Nepřímou součástí klienta je i html stránka `serverInfo.html`, která se nachází v `/public/` adresáři. Tato stránka slouží pro zobrazení informací o serveru, které jsou užitečné pro správu serveru, například pro zjištění aktuálního počtu připojených hráčů, seznamu her a podobně. Stránka získává data pomocí websocketového připojení k serveru a zobrazuje je v jednoduchém formátu, který poskytuje server.

13.11 Původní klient

V zaříjové fázi projektu se místo současného klienta v *React*u používal klient ve vanilla JavaScriptu. I přes to, že již není vyvíjen, tak některé základní části stále podporuje a proto se v repozitáři stále nachází ve složce `klient`.

14 Utility pro vývoj

Kromě hlavních částí projektu bylo během vývoje vyrobeno několik užitečných python programů, které pomáhaly s různými úkoly. Například pro generování obrázků karet byl pomocí umělé inteligence vtvořen python skript, který pomocí knihovny *PIL* generoval obrázky karet na základě zadaných parametrů. Tento skript umožnil rychle vytvořit velké množství karet pro různé pluginy bez nutnosti ručního designování každé karty. Další užitečný script stahoval karty z prší do konkrétní složky, což urychlilo proces získávání potřebných obrázků pro tento plugin.

15 Testování

Během vývoje byla hra nesčetněkrát testována jejím autorem. Kromě toho se mnohokrát testovala i s jeho kamarády a rodinou, kteří hru zkoušeli a upozorňovali na nedostatky. Hra se testovala jak na lokálním serveru, tak i na veřejném serveru, aby se otestovalo chování v různých sítích a s různými počty hráčů.

Díky tomu, že se do enginu přidaly rychle jednoužší hry, tak bylo testování zábavné a proto se mohlo testovat často a s různými lidmi, což pomohlo odhalit spoustu nedostatků a zlepšit hru.

Pro testování existuje i několik nástrojů, například pro zobrazení boxů kolem všech prvků DOM stačí přidat body třídu `testovani`, nebo po zobrazení celé websocketové komunikace stačí otevřít konzoli prohlížeče a podívat se na zelené zprávy.

16 Budoucí rozšiřování

Projekt je navržený tak, aby se dal snadno rozšiřovat. Díky pluginům je možné přidat další hry, které budou fungovat na stejném enginu. V plánu je přidat některá rozšíření pro *Bang!*, které již jsou předpřipravené. Dále by bylo možné přidat i další funkce pro klienta, například možnost přizpůsobení vzhledu, přidání zvuků a podobně.

17 Závěr