FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Simulace strategií spolupráce na oligopolistickém trhu coby pomůcka pro výuku ekonomických předmětů
Student:	Bc. Martin Kluger
Vedoucí:	Ing. Mgr. Pavla Vozárová, Ph.D., M.A.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

1. Nastudujte si mikroekonomickou teorii týkající se spolupráce v oligopolu jako opakované hře, popište různé možné strategie.
2. Proveďte průzkum volně dostupných online simuací podobných strategií, ať už přímo v rámci oligopolu nebo obecně v teorii her (vězňovo dilema).
3. Navrhněte responzivní aplikaci, která by umožňovala v rámci výuky ekonomických předmětů simulovat možné strategie na oligopolistickém trhu. Aplikace by měla umožňovat jak hru dvou hráčů proti sobě tak hru jednoho hráče proti počítači.
4. Aplikaci implementujte a otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 15. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Simulace strategií spolupráce na oligopolistickém trhu coby pomůcka pro výuku ekonomických předmětů

Bc. Martin Kluger

Katedra softwarového inženýrství
Vedoucí práce: Ing. Mgr. Pavla Vozárová, Ph.D., M.A.

27. června 2019

Poděkování

V první řadě bych rád poděkoval vedoucí své práce, paní Ing. Mgr. Pavle Vozárové, Ph.D., M.A. za pomoc při výběru velmi zajímavého tématu, za cenné rady při psaní této práce a také za zpětnou vazbu při vývoji aplikace, která na jejím základě vznikla. Dále bych chtěl poděkovat všem lidem, kteří mi vyjádřili podporu ať už při psaní této práce nebo při náročné cestě studiem.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 27. června 2019 .....

Odkaz na tuto práci

Kluger, Martin. *Simulace strategií spolupráce na oligopolistickém trhu coby pomůcka pro výuku ekonomických předmětů*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Cílem práce je představit problematiku spolupráce na oligopolistickém trhu a umožnit takovou spolupráci simulovat v rámci responzivní webové aplikace. Ta umožňuje uživateli otestovat své schopnosti proti počítačem řízenému protivníkovi nebo proti dalším skutečným hráčům. Jedná se o výpočetně náročnou simulaci, kterou je obtížné předvést běžnými prostředky. Výsledky této práce to však umožňují velmi snadno. Výuka formou hry s sebou také přináší velký počet výhod a množství aplikací podobné této, které zefektivňují výuku po celém světě, každým dnem roste.

Klíčová slova responzivní webová aplikace, kartel, nekalá spolupráce, oligopol, teorie her, cournotův model, strategie, návrh, implementace, javascript, react, express, node.js, socket.io

Abstract

Objective of this thesis is to introduce the topic of cooperation on oligopolistic markets and simulate such cooperation in the form of responsive web-based application. Such application allows the user to test their skill against artificial intelligence or other online users. This kind of simulation is very complex and hard to demonstrate using conventional means. It is however trivial using the web application created as a result of this thesis. Learning through games is widely known to have positive effects on the learning process and the number of applications, such as this one, making learning more accessible, is growing every day.

Keywords responsive web application, cartel, unfair cooperation, oligopoly, game theory, cournot competition, strategy, design, implementation, javascript, react, express, node.js, socket.io

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Kvalitní vzdělávání je pilířem moderní společnosti. V dnešní době se však již není možné vzdělávat bez použití moderních technologií k těmto účelům uzpůsobených. Takové technologie umí vysvětlovanou látku zjednodušit, podat jí uživateli uchopitelnou formou a poskytnou mu dostatek času na porozumění problematiky.

Přestože se většina z nás s webovými technologiemi setkává každý den, při výuce jsou stále využívány v omezené míře. Namísto nich jsou často nasazeny zastaralé vzdělávací postupy, protože pro inovaci není dostatek prostoru. Mým cílem v rámci této aplikace je nabídnout vzdělávací pomůcku, která umožní tyto nedostatky napravit.

Řešením je aplikace, která umožňuje interaktivní formou simulovat spolupráci na trhu s oligopolistickou konkurencí. V režimu pro jednoho hráče může uživatel otestovat svou znalost tématu proti počítačovému protivníkovi využívajícímu komplexní strategie. Alternativně aplikace umožňuje uživatelům soupeřit proti sobě navzájem. Tento režim je také velmi hodný pro demonstrování principů v rámci školní výuky.

V úvodu práce jsou představeny matematické a mikroekonomické principy, na kterých je má aplikace vystavěná. Další kapitoly se poté zabývají analýzou požadavků a volbou vhodných technologií pro implementaci mé aplikace. V závěru práce popisuji samotné provedení a také testování pro zajištění dostatečné kvality. Dále se zaměřuji na rozvoj aplikace a její potenciál do budoucna.

Kapitola 1

Cíl práce

Prvním z cílů je seznámit čtenáře s teorií oligopolu zasazenou do širších mikroekonomických souvislostí. Následuje úvod do matematické teorie her, který objasňuje problematiku spolupráce v různě komplexních situacích. Tato témata jsou následně propojena v rámci spolupráce na oligopolistickém trhu. Jsou představeny problémy takové spolupráce a různé strategie k jejímu dosažení. V neposlední řadě je provedena rešerše existujících řešení na poli simulačních her v rámci mikroekonomie a teorie her.

Druhým cílem je vytvořit aplikaci, která spolupráci na oligopolistickém trhu simuluje v souladu s dříve uvedenou teorií. Nejprve je provedena analýza požadavků na tuto aplikaci, na základě které jsou následně vybrány nejvhodnější technologie pro její implementaci.

Třetím cílem je samotná implementace, testování a nasazení takové simulační aplikace do provozu. To je provedeno tak, aby byla aplikace připravena k nasazení do webového prostředí, kde k ní budou uživatelé přistupovat z mobilních zařízení a osobních počítačů, a kde bude sloužit jako pomůcka pro výuku ekonomických předmětů.

Kapitola 2

Ekonomická teorie

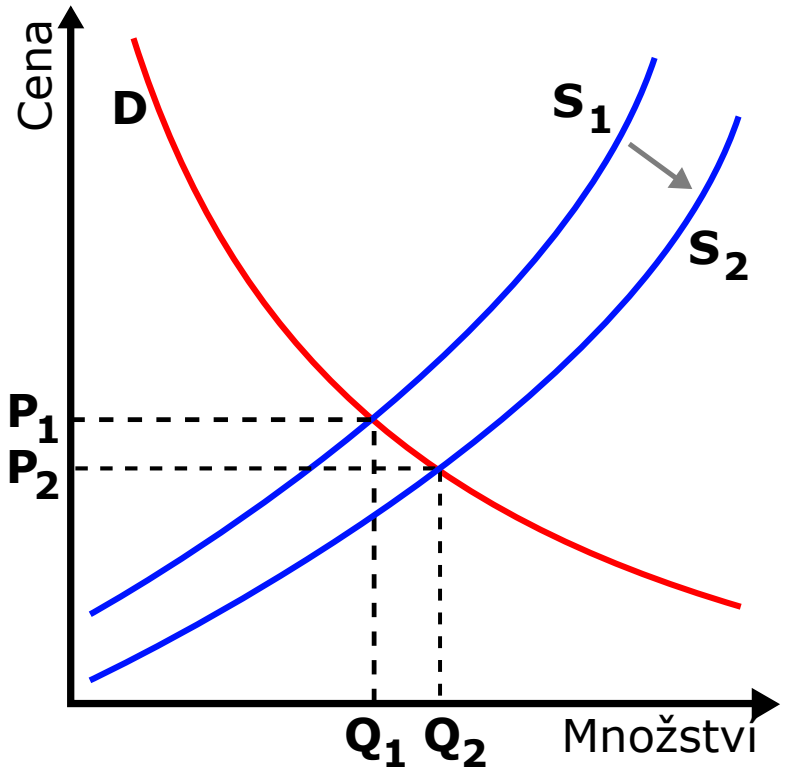
V následující kapitole je uvedena teorie na níž je založena aplikace, kterou je cílem vytvořit v rámci této práce. Tato aplikace má za cíl tuto teorii simulovat a sloužit k jejímu lepšímu uchopení, a je proto nutné ji nejprve vyhranit. Dále tato kapitola obsahuje analýzu existujících řešení, zejména aplikací, které již stejnou nebo podobnou funkcionalitu nabízejí.

2.1 Princip nabídky a poptávky

Nejprve je třeba uvést princip nabídky a poptávky jakožto jeden z fundamentálních principů mikroekonomie.

2.1.1 Nabídka

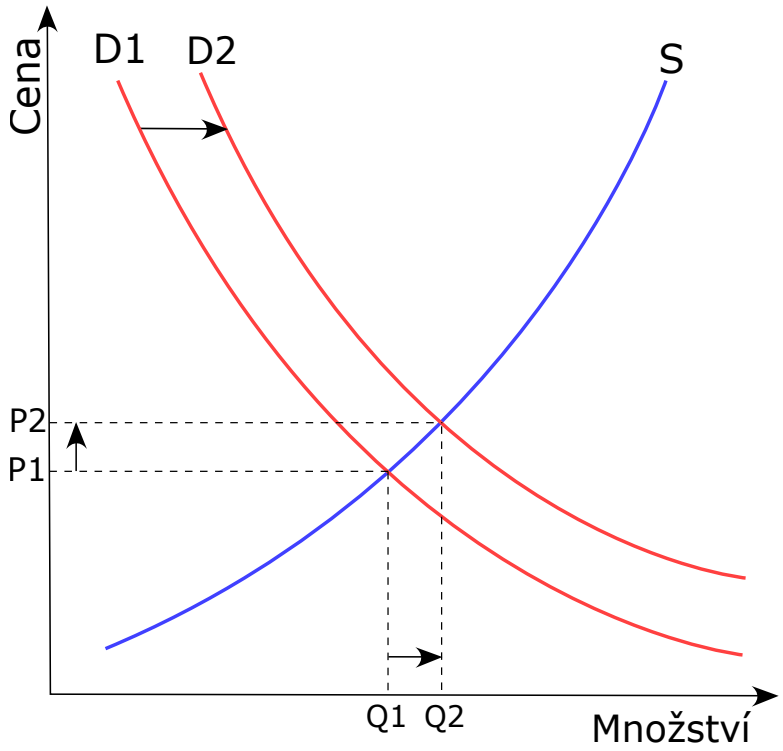
Křivka nabídky nám udává množství zboží, které jsou výrobci ochotni prodat za danou cenu, přičemž ostatní faktory zůstávají konstantní. Ukázková křivka nabídky S_1 na obrázku 2.1 demonstruje tento princip. Na vertikální osu grafu je zanesena cena zboží, na horizontální osu poté dodávané množství. Povšimněme si také, že se jedná o rostoucí funkci, jinými slovy, čím vyšší je cena, tím více budou firmy ochotny vyrábět a prodávat zboží. To si lze představit například tak, že vyšší cena zboží, která vede k vyšším ziskům, umožní firmě investovat zpět do výrobního procesu a vyrobit tak více zboží. Nabízené množství samozřejmě závisí na dalších faktorech kromě ceny, tyto faktory jsou vyjádřeny tvarem křivky. Podívejme se například, co by se stalo, pokud by klesla cena surového materiálu pro výrobu zboží. Tuto situaci máme vyobrazenou na obrázku 2.1 v podobě křivky S_2. Vidíme, že díky sníženým nákladům jsou firmy ochotny vyrobit více zboží za nižší cenu než v případě křivky S_1. Vztah ceny a vyrobeného množství je tedy udán samotnou křivkou, kdežto vnější vlivy na tyto faktory jsou udány změnou křivky (jejím posunem nebo změnou jejího tvaru). [1]



Obrázek 2.1: Křivka nabídky (Zdroj: [2])

2.1.2 Poptávka

Křivka poptávky demonstruje, kolik jednotek zboží jsou zákazníci ochotni koupit v závislosti na ceně za jednotku. Ukázkovou křivku D1 je možné vidět na obrázku 2.2. Opět platí, že na vertikální ose je vynesena cena zboží a na horizontální poptávané množství. Všimněme si, že na rozdíl od křivky nabídky je tato funkce klesající. Tím je vyjádřen fakt, že čím je nižší cena, tím jsou zákazníci ochotnější zboží koupit. S klesající cenou také přibývá zákazníků, kteří jsou ochotní si zboží koupit a v případě vyšší ceny nebyli. Samozřejmě množství poptávaného zboží mimo ceny závisí na dalších faktorech. Jedním z takových faktorů je příjem, zákazníci s vyšším příjmem utratí za jakékoliv zboží více peněz. Situaci, která by nastala v případě zvýšení příjmů, můžeme opět vidět na obrázku 2.2 v podobě křivky D2. Vidíme, že pokud bychom v takovém případě drželi konstantní cenu, množství poptávaného zboží by se oproti křivce D1 zvýšilo. [1]



Obrázek 2.2: Křivka poptávky (Zdroj: [3])

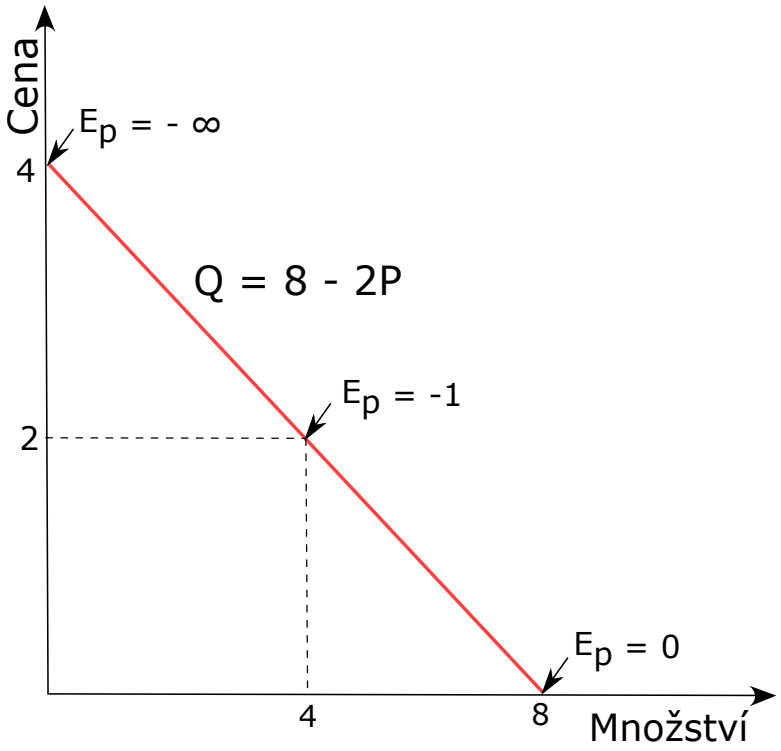
Dalším významným faktorem, který ovlivňuje poptávku jsou **substituty** a **komplementy**. O zboží řekneme, že je **substitutem**, pokud zvýšení ceny jednoho zboží vede k nárůstu poptávaného množství zboží druhého. Příkladem takového zboží jsou měď a hliník, neboť jsou dost často zaměnitelné v industriálním využití. Poptávané množství mědi vzroste, dojde-li ke zvýšení ceny hliníku. Podobně jsou substitučním zbožím kuřecí a hovězí maso, neboť většina zákazníků je ochotna změnit preference a koupit levnější zboží, dojde-li k nárůstu cen jednoho z nich. Za **komplementy** dva druhy zboží označíme, pokud nárůst ceny jednoho z nich vede ke snížení poptávky po tom druhém. Jako příklad můžeme uvést automobily a pohonné hmoty, protože jsou nejčastěji užívány společně. Zvýšení cen pohonných hmot vede ke snížení poptávaného množství automobilů. [1]

2.1.3 Lineární křivka poptávky

Již jsme ukázali, že poptávka po zboží nezávisí pouze na jeho ceně, ale také na příjmech zákazníků a na cenách ostatního zboží. Podobně, nabídka záleží na ceně a proměnných ovlivňujících výrobní cenu. **Elasticita** nám vyjadřuje vzájemnou citlivost těchto proměnných. Přesněji řečeno, jedná se o číslo, které nám říká, k jak velké procentuální změně první proměnné dojde na základě jednoprocentní změny druhé proměnné. Nejprve uvedeme **cenovou elasticitu poptávky**, která je udána následujícím vzorcem:

$$E_p = \frac{\%\Delta Q}{\%\Delta P} = \frac{\Delta Q/Q}{\Delta P/P} = \frac{P\Delta Q}{Q\Delta P}$$

kde %ΔQ chápeme jako procentuální změnu poptávaného množství a %ΔP jako procentuální změnu ceny. Většinou se jedná o záporné číslo, neboť vzroste-li cena zboží, poptávané množství typicky klesá. [1]



Obrázek 2.3: Lineární křivka poptávky Q = 8 - 2P

Vrátíme-li se zpět ke křivce poptávky, z výše uvedené rovnice vyplývá, že v různých bodech křivky bude elasticita různá, proto se princip elasticity nejlépe demonstruje na **lineární křivce poptávky**. Tato křivka má následující zápis:

$$Q = a \ - b \times P$$

kde Q je opět množství a P cena za kus zboží. Také zde figurují kladné reálné koeficienty **a** a **b**, které si můžeme představit jako vlivy na celkovou poptávku. Vezměme tedy jako příklad křivku následující:

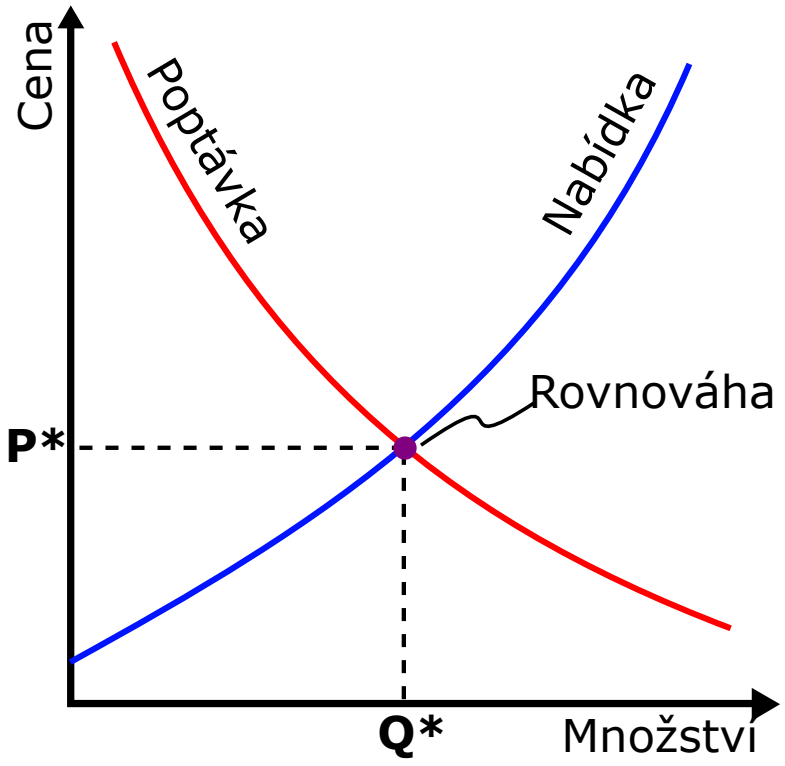
$$Q = 8 - 2P$$

Grafické vyobrazení této křivky je možné vidět na obrázku 2.3. Pro tuto křivku je podíl ΔQ/ΔP konstantní a je roven -2, ale elasticita křivky není konstantní. Jelikož je křivka poptávky klesající funkce, se zvyšujícím se množstvím dochází ke snížení poměru ceny ku množství (P/Q). Vliv na elasticitu je vyznačen na obrázku 2.3. [1]

K extrémním případům by došlo v případě, že by křivka poptávky byla plně horizontální nebo plně vertikální. V případě plně horizontální křivky by se jednalo o nekonečně elastickou poptávku, protože poměr ΔQ/ΔP je v takovém případě nekonečný a také jakákoliv změna ceny vede k obrovské změně množství. Říkáme, že elasticita poptávky je v tomto případě nekonečná. V případě vertikální křivky, poměr ΔQ/ΔP je 0. Jelikož poptávané množství je vždy stejné bez ohledu na cenu, říkáme, že elasticita poptávky je nulová. V rámci mojí aplikace se budeme zabývat pouze lineární křivkou poptávky a trhem s homogenním zbožím. Horizontální ani vertikální poptávkou se dále zabývat nebudeme. [1]

2.1.4 Tržní rovnováha

Nyní, když jsme definovali pojmy nabídka a poptávka, uvedeme je do společného kontextu. Na obrázku 2.4 vidíme průnik křivek nabídky a poptávky. Bod, ve kterém se tyto křivky protínají, nazveme bodem **tržní rovnováhy**. Vidíme, že při ceně P^* je množství nabízeného zboží stejné jako množství poptávaného zboží Q^*.



Obrázek 2.4: Ilustrace tržní rovnováhy (Zdroj: [4])

Jednou z vlastností volných trhů je tendence ceny zboží přibližovat se k výše zmíněnému rovnovážnému stavu P^*, dokud se množství poptávaného a nabízeného zboží nerovnají. Protože v tomto bodě není převis nabídky ani poptávky, není na cenu zboží vyvíjen žádný další tlak a ta se dále nemění. Abychom situaci lépe porozuměli, představme si, že počáteční cena je vyšší než P^*, v této situaci budou výrobci vyrábět a prodávat více, než kolik jsou zákazníci ochotni koupit. Nastalé situaci říkáme **převis nabídky** nad poptávkou, nabízeného zboží je více než poptávaného. Aby se firmy tohoto nadbytečného zboží zbavily, nebo aby alespoň zabránily dalšímu růstu nadbytku, začnou snižovat cenu. Se snížením ceny dojde ke zvýšení poptávaného množství zboží, což následně vede ke snížení množství zboží nabízeného. Takto se pokračuje až do chvíle, kdy je dosažena rovnovážná cena P^*.

V opačném případě se může stát, že počáteční cena bude na úrovni nižší než P^*, v takovém případě nastává **převis poptávky** nad nabídkou, neboli poptávané množství převyšuje nabízené. Zákazníci v takové situaci nemohou nakoupit tolik zboží, kolik by v ideálním případě chtěli, začnou tedy vyvíjet rostoucí tlak na cenu, neboť budou ochotni nabídnout za zboží vyšší než tržní cenu, jen aby ho získali. Firmy na tuto situaci budou reagovat tak, že zvýší cenu zboží a začnou ho vyrábět více, stejně jako v předchozím případě nakonec cena dosáhne rovnovážného stavu P^*. [1]

V rámci simulace vystupují jak křivka nabídky, tak křivka poptávky. V případě nabídky se jedná zejména o nákladovou křivku, samotnou nabídku pak určuje hráč. Poptávku je možné navolit při založení hry nebo využít některá z již existujících nastavení. Průnik těchto křivek je důležitý pro určení optimálního zisku jednotlivých firem. Jak určit takový zisk je popsáno v následujících kapitolách.

2.2 Úvod do tržních struktur

Aby se firma mohla správně rozhodnout jak maximalizovat své zisky, musí kromě svých zákazníků znát také situaci na trhu. Chování firem závisí na tržní struktuře, která mimo jiné zahrnuje počet firem na trhu, schopnost odlišení svého produktu od ostatních a obtížnost, s jakou firmy mohou vstupovat na trh a opouštět jej. V této kapitole vyčleníme základní druhy tržních struktur a poté se více zaměříme na problematiku oligopolu, kterým se tato diplomová práce zabývá.

2.2.1 Dokonalá konkurence

Mluvíme-li o tržní konkurenci, máme obvykle na mysli firmy soupeřící o stejné zákazníky na určitém trhu. Z ekonomického hlediska však o trhu řekneme, že se jedná o dokonalou konkurenci za podmínky, že všechny firmy jsou tzv. **cenovými příjemci**. Pro firmu to znamená, že nemůže výrazně ovlivnit cenu, za kterou prodává zboží ani cenu, za kterou nakupuje suroviny. Pro firmu platí, že musí být cenovým příjemcem. Když čelí horizontální křivce poptávky, nemá na vybranou. Znamená to, že firma může za danou cenu prodat libovolné množství zboží, nemá tedy důvod cenu snížit. Podobně, nemůže cenu zvýšit snížením vyráběného množství, protože čelí nekonečně elastické poptávce. (viz kapitola 2.1.3) Drobné zvýšení ceny vyústí ve snížení poptávky na nulu. Firmy na trhu proto vždy volí vyráběné **množství** a nikdy výrobní cenu. [5]

Důvodů, proč je poptávková křivka firmy v dokonalé konkurenci horizontální, je mnoho. Pro ilustraci uvedu několik společných rysů firem, které jsou na trhu cenovými příjemci. Na trhu typicky figuruje **velké množství firem**, což má za následek, že pokud by jedna z firem trh opustila, nabídka by klesla téměř neznatelně a ostatní firmy by naplnily poptávku. Dalším rysem tohoto trhu je, že prodávané **produkty jsou homogenní** nebo identické. Zákazníkovi nezáleží na tom, který farmář vypěstoval konkrétní odrůdu jablek, pokud dosahují požadované kvality. V kontrastu k tomuto stojí diferencované neboli heterogenní produkty, které nalezneme například v automobilovém průmyslu. Parametry jednotlivých výrobců aut se velmi liší a trh s automobily také nepatří do kategorie dokonalé konkurence. Zákazník také typicky má **přístup ke všem informacím o trhu**, tedy ví, že produkty jsou identické, a je také schopen pozorovat ceny na trhu. Vzroste-li cena u jednoho výrobce, jednoduše nakoupí u jiného. V neposlední řadě **na trhu neexistují bariéry**, firmy mohou na trh volně vstupovat a také jej opouštět, což vede k minimalizaci firemních zisků, neboť v případě, kdy budou zisky vysoké, budou na trh vstupovat další firmy s vidinou výdělku a budou snižovat tržní cenu zboží. [5]

2.2.1.1 Maximalizace zisku

Jedním z klíčových indikátorů úspěchu firmy na trhu je úroveň jejích zisků. V rámci mojí aplikace firmy také generují zisk a jeho velikost jednak určuje vítěze, neboť se jedná o hru, ale zisk firem je také klíčový pro demonstraci ekonomických principů, kvůli kterým byla má aplikace vytvořena. V této kapitole proto ukáži, jak se vypočítá zisk firem, abychom tento princip mohli poté aplikovat na různé tržní struktury.

Zisk firmy (Π) je určen jako rozdíl mezi celkovými příjmy firmy, které označíme TR (z anglického total revenue) a mezi jejími celkovými náklady, které budeme značit TC (z anglického total costs). Vzorec pro výpočet zisku pak vypadá následovně:

$$\Pi = TR - TC$$

Pokud je výsledek záporný, tedy $\Pi < 0$, říkáme, že firma utržila ztrátu. Výpočet příjmů TR je snadno vyjádřitelný jako počet prodaných kusů zboží vynásobený jejich cenou. Výpočet nákladů je složitější. Je samozřejmě možné počítat všechny vynaložené zdroje v uvažovaném období, z ekonomického hlediska je ale správný postup také počítat náklady ušlé příležitosti, tedy sumu, kterou by bylo možné získat alternativní investicí zdrojů, které firma vynaložila. Typicky se uvažuje výnos z uložení peněz do banky nebo investice do jiného odvětví a další podobné alternativy. [5]

Aby firma maximalizovala zisk, musí se rozhodnout, jak velké množství výstupů vyrobit a následně prodat. V této situaci firma odpovídá na dvě základní otázky, uvažujeme-li úroveň výstupu q^*, maximalizuje tento výstup zisk nebo minimalizuje ztrátu? Na základě první otázky, je pro firmu výnosné produkovat výstup q^* nebo ukončit výrobu a neprodukovat nic? [5] Existuje množství strategií pomocí kterých lze tyto otázky zodpovědět, jelikož to však není cílem této práce, nebudeme se jim podrobněji věnovat.

2.2.1.2 Zisk v případě dokonalé konkurence

Abychom nyní mohli principy maximalizace zisku aplikovat na trhy s dokonalou konkurencí, je potřeba definovat několik pojmů. **Mezní náklady** (marginal cost - MC) definujeme jako hodnotu, o kterou se změní celkové náklady firmy vlivem změny objemu produkce. Matematicky vyjádříme jako

$$MC = \Delta TC/\Delta q$$

kde ΔTC je změna celkových nákladů a Δq je změna množství. Podobně, **mezními příjmy** (marginal revenue - MR) firmy rozumíme změnu celkových příjmů firmy vlivem změny objemu produkce. Matematicky lze opět vyjádřit takto:

$$MR = \Delta TR/\Delta q$$

kde ΔTR je změna celkových příjmů a Δq je opět změna v produkovaném množství. Podobně je možné definovat **mezní zisk** (marginal profit - MP) jako změnu celkového zisku firmy na základě změny objemu produkce. Zároveň ale platí následující rovnice:

$$MP(q) = MR(q) - MC(q)$$

Z této rovnice vidíme, že mezní zisk mimo jiné získáme odečtením mezních nákladů od mezních příjmů. [5]

Lze ukázat, že jakákoliv firma maximalizuje zisk, produkuje-li takový výstup, pro který je mezní zisk MP nulový. Ekvivalentně se v takovém případě mezní náklady MC rovnají mezním příjmům MR. V případě dokonalé konkurence to znamená, že firma produkuje takový výstup, při kterém platí rovnice $MC(q) = p$, kde p je tržní cena jednoho kusu výstupu. Na základě tohoto výstupu firma přistupuje k druhému kroku rozhodovacího procesu. Firma bude tento výstup produkovat, pokud tím bude dosahovat nezáporného zisku, neboli nebude ve ztrátě. [5]

Podrobnější rozbor trhů s dokonalou konkurencí je ponechán na čtenáři, neboť cílem této práce je simulovat trhy oligopolistického charakteru.

2.2.2 Monopol

V kontrastu s dokonalou tržní konkurencí stojí monopol. **Monopol** je jediným tržním dodavatelem zboží, pro které neexistuje žádný blízký tržní substitut. Hlavním rozdílem je, že taková firma není cenovým příjemcem, ale naopak cenu nastavuje. Stejně tak by mohla volit výrobní množství a z něj odvodit cenu. Lze ukázat, že oba přístupy jsou v případě monopolu ekvivalentní. Protože monopol čelí klesající křivce poptávky a nikoliv horizontální, nepřijde zvýšením ceny o všechny zákazníky, proto nastavuje cenu vyšší než své mezní náklady, aby maximalizoval zisk. Zákazníci sice za tuto cenu nakupují menší množství, ale zisky monopolu jsou vyšší. Samotný proces maximalizace zisku bude analogický jako v případě dokonalé konkurence.

Podobně jako může být na trhu jediná firma, může nastat situace, ve které je na trhu pouze jediný zákazník. Takovému stavu se říká **monopson**. Na takovém trhu má naopak všechnu sílu v rukou kupující, který nastavuje cenu. Takové trhy nejsou typické, ale existují. Může se jednat například o výrobce zbraní, od kterých nakupuje pouze ministerstvo obrany.

2.2.3 Diferencované produkty

Doposud jsme se zabývali tržními strukturami, pro které platilo, že výrobky jednotlivých firem jsou navzájem nerozlišitelné. Takovým trhům se říká trhy s homogenními produkty. Abychom mohli uvést další významné tržní struktury, je třeba nejprve definovat trhy s diferencovanými produkty.

V ekonomii rozlišujeme dva druhy produktové diferenciace. **Vertikální diferenciace** se týká kvality a o dvou produktech řekneme, že jsou vertikálně diferencované, pokud zákazníci považují jeden z nich za lepší nebo horší než ten druhý. **Horizontální diferenciace** vypovídá o zaměnitelnosti dvou produktů. Pokud zákazníci jeden z produktů preferují natolik, že jsou ochotni zaplatit vyšší cenu namísto nákupu substitutu, pak jsou produkty horizontálně diferencované. [6]

2.2.4 Oligopol a monopolistická konkurence

Trh, na kterém operuje relativně málo firem a na nějž je obtížné vstoupit pro nové firmy, nazýváme **oligopolem**. Kvůli relativně nízkému počtu firem má každá z nich vliv na cenu a ovlivňuje také své tržní konkurenty. Protože při maximalizaci zisku musí firma vzít v úvahu chování svých rivalů, je pro ní rozhodnutí, které musí učinit, složitější než v případě monopolu nebo dokonalé konkurence. Firma v oligopolu, která nesprávně odhadne chování svých konkurentů nebo se jej dokonce rozhodne ignorovat, s velkou pravděpodobností utrpí ztrátu. [5]

Pro oligopol je také typické, že firmy mohou jednat nezávisle nebo se mohou rozhodnout spolupracovat. Skupině firem, která se takto rozhodne spolupracovat, se říká **kartel**. Kartelová dohoda umožňuje firmám chovat se na trhu jako monopol a tím maximalizovat své kolektivní zisky. Ve většině vyspělých zemí, včetně České republiky, je taková dohoda nezákonná. Pokud firmy nespolupracují, znamená to pro ně nižší zisk, protože je ale na oligopolistickém trhu firem relativně málo, udržují v dlouhodobém časovém horizontu kladné zisky. To je zásadní rozdíl oproti firmám v rámci dokonalé konkurence, které mají v tomto horizontu zisky nulové. Pro firmy je kartelová dohoda velmi lákavá, zejména proto, že kromě vyšších zisků také vyrábějí nižší množství a ušetří tak na nákladech. [5]

Na oligopolistickém trhu existuje jedna nebo více překážek odrazující nové firmy od vstupu na trh. Pokud na trhu taková bariéra neexistuje, budou na trh vstupovat stále další firmy, dokud se zisky nebudou limitně blížit nule. **Monopolistická konkurence** je taková tržní struktura, ve které firmy mají dostatečnou tržní sílu, aby mohly nastavit cenu vyšší než mezní náklady, ale pokud by na trh vstoupila nová firma, nebude již schopná dosáhnout nenulových zisků. Na rozdíl od dokonalé konkurence však firma čelí klesající reziduální křivce poptávky nikoliv horizontální. To je dáno buď tím, že má na trhu málo rivalů nebo firmy vyrábí diferencované produkty. [5]

2.2.4.1 Cournotův model oligopolu

Francouzský ekonomik a matematik Antoine-Augustin Cournot představil první formální model oligopolu v roce 1838. V tomto modelu popisuje, jak se firmy budou chovat, mají-li všechny zároveň určovat výrobní množství. Každá firma se musí rozhodnout o výrobním množství, aniž by věděla, kolik kusů zboží vyrobí její konkurenti. [7]

Cournotův model předpokládá, že firmy produkují homogenní produkty a operují v rámci oligopolu. To znamená, že firmy čelí stejné křivce poptávky. Na základě zadání bude navíc křivka poptávky tohoto trhu pouze lineární. Nabídku pak určují parametry jednotlivých firem. Předpokládáme-li navíc, že obě firmy mají identické mezní náklady, zbývá jim rozhodnout, jak velké množství vyrábět. Na základě tohoto rozhodnutí bude poté určena společná tržní cena. Cournot v počátku uvažoval **duopol**, tedy trh na kterém operují pouze dvě firmy, model ale lze aplikovat na trh s libovolným počtem firem. Firmy volí vyráběné množství nezávisle na sobě, nespolupracují a nemají žádnou informaci o tom, jaké množství plánuje vyrobit jejich konkurent. Rozhodnutí o výstupu firmy záleží na tržní ceně, která se ale odvíjí od kombinovaného výstupu obou firem. To znamená, že tržní cena není známá, dokud obě firmy neučinily rozhodnutí o vyráběném množství. Obě firmy proto budou maximalizovat zisk na základě odhadu vyráběného množství svého konkurenta. [6]

Zafixujeme-li předpokládané množství, které vyrobí firma B, můžeme pak říci, že firma A čelí poptávce, která je dána **reziduální křivkou poptávky**. Ta nám udává vztah mezi tržní cenou a výstupem firmy A. Firma se potom chová jako monopol vzhledem k reziduální křivce poptávky a stejně jako v případě ostatních tržních modelů řeší rovnici $MR = MC$. Vezmeme-li v úvahu všechny možné výstupy, které firma B může vyrobit, zisk maximalizující odpovědi firmy A na tyto výstupy pak tvoří **reakční křivku** R_S. [6]

2.2.4.2 Rovnováha v rámci Cournotova modelu

V případě dokonalé konkurence je hlavním příznakem rovnováhy fakt, že žádná z firem nemá motivaci odchýlit se od svého zisk maximalizujícího rozhodnutí, jakmile je dosaženo tržní rovnováhy. To samé platí pro **Cournotovo ekvilibrium**, ve kterém je výstup každé firmy optimální odpověď na vyráběné množství ostatních firem na trhu. Toto ekvilibrium se nachází v průniku reakčních křivek všech firem figurujících na trhu. Cournotův model oligopolu je statickým modelem, a z toho důvodu nevysvětluje, jak se firmy dostanou do stavu odpovídajícímu rovnováze. Lze se ale domnívat, že pokud firma zváží svojí zisk maximalizující strategii a tu svého konkurenta, dostane se postupnou eliminací produkovatelných výstupů až do množství, které odpovídá rovnovážnému stavu. [6]

Je také důležité zmínit, že budou-li obě firmy vyrábět množství udané tímto rovnovážným stavem, jejich kombinovaný výstup nebude odpovídat množství, které by vyráběl monopol. Není tedy dosaženo maximálního potenciálu, který trh nabízí. Toho by dosáhly firmy operující v rámci kartelové dohody, které by společně vyrobily výstup odpovídají výstupu monopolu a rovnoměrně by si tržní poptávku rozdělily. [6]

2.2.4.3 Bertrandův model oligopolu

Ukázali jsme si, jak mohou oligopolistické firmy určovat vyráběné množství s cílem maximalizovat zisky. Často však firmy volí cenu svých produktů a nechávají zákazníky rozhodnout, jak velké množství chtějí koupit. V takovém případě je tržní rovnováha odlišná od Cournotova modelu. Joseph Bertrand takto argumentoval v roce 1883, proto se stavu, ve kterém žádná firma nemůže dosáhnout vyššího zisku změnou ceny, za předpokladu, že ostatní firmy ceny také nezmění, říká **Bertrandovo ekvilibrium**.

Budeme-li uvažovat, že obě firmy mají identické náklady a vyrábí nerozeznatelné produkty, pak cena za kus v Bertrandově ekvilibriu je rovna mezním nákladům, stejně jako v případech, kdy je firma cenovým příjemcem. To můžeme snadno demonstrovat. Stanovme konstantní mezní náklady a průměrné náklady za kus například na 50 korun českých a uvažujme, že firma 2 nabízí produkt za 100 korun. Pokud v této situaci firma 1 bude účtovat za kus více než 100 korun, neprodá nic, neboť všichni zákazníci budou nakupovat u firmy 2. Pokud nastaví stejnou cenu jako firma 2, očekáváme, že získá polovinu zákazníků a také tržního zisku. Nic ale nebrání první firmě nastavit cenu jen o něco nižší, než jakou si účtuje druhá firma, řekněme 99 korun za kus. Protože jsou ale produkty identické, firma 1 nyní obslouží celý trh a získá celý tržní zisk. Firma 2 nebude mít zisk žádný. Ta se samozřejmě s nastalou situací nesmíří a sníží cenu svého produktu tak, aby byla nižší než konkurenční. Představme si nyní, že firma 1 bude nabízet produkt za své mezní náklady, tedy 50 korun. Pokud by nyní firma 2 nastavila cenu nižší než svůj konkurent, získala by sice celou tržní poptávku, ale operovalo by ve ztrátě. Ukázali jsme tedy, že jediná situace ve které firmy nemají motivaci změnit cenu je, pokud již obě nabízejí produkt za své mezní náklady. Firmy tedy v dlouhodobém horizontu mají nulový zisk, stejně jako v případě trhu s dokonalou konkurencí. [5]

Z výše uvedeného vyplývá, že rovnovážná situace na Bertrandově trhu je stav, kdy všechny firmy dosahují nulových zisků. Pokud bychom chtěli dosáhnout zisků nenulových, museli bychom uvažovat trh s diferencovanými produkty a různé poptávkové křivky pro jednotlivé firmy na trhu. To by zbytečně komplikovalo simulační hru, jednak co se týče implementace, ale také složitosti na pochopení. Jelikož se jedná o simulaci formou hry, trh s nenulovými zisky se jeví jako atraktivnější volba. Z těchto důvodu je jedním z požadavků zadavatele implementovat v mé aplikaci Cournotův model oligopolu a nikoliv Bertrandův.

2.2.4.4 Další modely oligopolu

V ekonomické teorii existují i další uznávané modely, jako například **Stackelbergův model**, ve kterém jedna firma figuruje jako množstevní vůdce, který vybírá kvantitu pro výrobu jako první a ostatní firmy se poté rozhodují na základě jeho volby. Jelikož hráči v rámci mojí aplikace činí rozhodnutí současně, použít tento model by nedávalo smysl. [5]

2.3 Teorie her

Teorie her je disciplínou aplikované matematiky a zabývá se analýzou optimálního rozhodování v konkurenčním boji, kde každé rozhodnutí má značný vliv na tržní postavení soupeře.

Máme možnost studovat dva základní typy her. **Statické hry**, ve kterých se hráči rozhodují zároveň a nejedná se o opakovanou hru, což znamená, že se hráči již znovu nepotkají, nehrozí jim odplata za vzájemné poškozování a podobně. Příkladem takové hry je vězňovo dilema, které v této kapitole podrobněji rozebereme. Cournotův a Bertrandův model oligopolu, tak jak jsme si je uvedli, také patří mezi modely statických her. Na druhé straně máme **dynamické hry**, ve kterých se hráči v tazích střídají. Alternativně, jako například v případě mojí aplikace, hrají současně, ale opakovaně. V takových hrách mohou hráči využít informace o předchozích tazích svých protivníků. [5]

2.3.1 Nashovo ekvilibrium

V rámci teorie her je naším cílem odpovědět na otázku: Jaký bude pravděpodobně výsledek hry? Pro určení pravděpodobných výstupů hry se používá koncept **Nashova ekvilibria**. V tomto ekvilibriu volí každý hráč takovou strategii, která mu přinese nejvyšší výplatu, vezmeme-li v úvahu rozhodnutí ostatních hráčů. Taková ekvilibria jsme si již představili v rámci modelů oligopolu, neboť Bertrandovo i Cournotovo ekvilibrium je příkladem Nashova ekvilibria. Nejpůsobivější vlastností takové rovnováhy je, že pokud jedna strana od té druhé očekává, že zvolí strategii danou Nashovým ekvilibriem, pak jí zvolí obě strany. To není pravdivé v případě výstupů mimo Nashovo ekvilibrium. [6]

2.3.2 Vězňovo dilema

Nashovo ekvilibrium ne vždy koresponduje s maximálním sdruženým ziskem obou hráčů. Protože hráči racionálně staví své zájmy na první místo, často tím poškozují společné blaho všech hráčů. Této preferenci vlastních zájmů nad kolektivními zájmy se často říká **vězňovo dilema**. Tento termín vznikl na základě následující situace: Dva zločinci podezřelí ze spáchání zločinu, Mario a Luigi, jsou zatčeni a umístěni do oddělených cel. Policie, která proti nim nemá žádné pádné důkazy, jim každému nabídne spolupracovat a svědčit proti tomu druhému. Pokud ani jeden z nich nepromluví, budou oba odsouzeni z drobného zločinu a stráví ve vězení jeden rok. Pokud se oba přiznají, budou odsouzeni ze závažnějšího zločinu, ale protože spolupracovali, půjdou za mříže na pět let. Pokud ale promluví jen jeden z nich, získá imunitu a do vězení nepůjde, zatímco jeho komplic bude odsouzen na 10 let. Všechny možné výstupy této hry je možné vidět v tabulce 2.1. [6]

Tabulka 2.1: Vězňovo dilema

Luigi / Mario	**Spolupracovat**	**Nespolupracovat**
Spolupracovat	-5, -5	0, -10
Nespolupracovat	-10, 0	-1, -1

V případě vězňova dilematu je Nashovo ekvilibrium situace, ve které oba vězni spolupracují. Pokud by Luigi spolupracoval s policií, Mario dostane nižší sazbu, pokud bude také spolupracovat. Stejná situace nastává v opačném směru. Oba podezřelí tedy v rovnovážném stavu budou odsouzeni na pět let, ačkoliv dohromady by na tom byli lépe, kdyby ani jeden z nich nespolupracoval, protože pak by každý z nich byl odsouzen pouze na jeden rok. Vězňovo dilema je často studovaný fenomén, protože věrně odráží problematiku reálných situací. Firmy se například často pouštějí do cenových válek, ačkoliv ve výsledku tím poškodí všechny firmy operující na trhu. Studiem vězňova dilematu máme možnost lépe pochopit, proč k těmto kontraproduktivním výsledkům dochází. [6]

2.3.3 Dynamické hry

Z kategorie dynamických her se nebudeme zabývat sekvenčními hrami, mezi které patří například dříve zmiňovaný Stackelbergův model cenového vůdce, namísto toho se zaměříme na opakované hry, které se má aplikace snaží emulovat.

V případě statických her, jako je například vězňovo dilema, hra trvá pouze jednu periodu a hráči nemají možnost jeden druhého potrestat za spolupráci s policií. Takový přístup neodráží příliš dobře skutečné podmínky. Možnost reagovat na chování soupeře však existuje v případě dynamických her. Jako příklad uvažujme kartelovou dohodu v rámci oligopolu. Neboť hra trvá více než jednu periodu, firmy mají možnost vzájemného trestání v případě odchýlení od dohody s vidinou vlastního zisku. [5]

V opakované hře může firma použít vyráběné množství k tomu, aby signalizovala strategie nebo vyhrožovala svému konkurentovi. Může například snížit vyráběné množství, aby tak signalizovala, že její konkurent má začít také vyrábět nižší množství, neboť jak jsme si již ukázali, celkově nižší množství zboží na trhu vede k vyšším cenám a kolektivně vyššímu zisku pro všechny výrobce na trhu. V této kapitole si dále ukážeme některá typická chování firem, které bude možné demonstrovat jakožto osobnosti počítačového protivníka v rámci této aplikace.

2.4 Oligopol jako opakovaná hra

V každodenním životě se stejné firmy setkávají na trhu opakovaně. Představíme-li si tuto situaci jako simulační hru, musí firma v každém kole této hry zvolit nejlepší strategii pro celý zbytek hry a ne jen pro jedno konkrétní kolo.

Jako **nekalou spolupráci** označíme situaci, ve které firmy záměrně produkují snížené množství s cílem zvýšit cenu zboží a své zisky. Často se jedná o zákonem zakázanou činnost, která poškozuje zájmy spotřebitele. V rámci mojí aplikace se zaměříme na situace, kdy je nekalá kooperace nejlepší tržní strategií, za jakých podmínek je udržitelná a pomocí kterých metod lze protihráče přimět k takové spolupráci.

2.4.1 Předpoklady nekalé spolupráce

Ačkoliv může být pro firmy nekalá spolupráce (obecně označovaná také jako kartelová dohoda nebo tzv. kartel) velmi lukrativní, v mnoha zemích světa, včetně České republiky, je mimo výjimečné situace zakázaná. V České republice jí obecně zakazuje Zákon o ochraně hospodářské soutěže [8].

2.4.1.1 Legální dohody o spolupráci

Jedním příkladem aktivně operujícího kartelu může být Organizace zemí vyvážejících ropu (OPEC [9]), která je mezivládní organizací sdružující 14 zemí exportujících ropu. Jelikož se jedná o mezinárodní sdružení zemí, neaplikují se na něj zákony, které v různých zemích světa kartelové dohody zakazují, a jedná se tak o legální organizaci. Podle statistik z roku 2017 členské země kontrolují přes 81.89% [10] všech zásob ropy a společně rozhodují o objemu a ceně exportované ropy, například pomocí těžebních kvót. Jak jsme již ukázali dříve, takováto dohoda má zásadní vliv na tržní křivku nabídky. Kvůli uměle vytvořenému nedostatku zásadně roste cena produktu a vzniká prostředí, které není výhodné pro spotřebitele. I přesto, že je organizace legální, čelí některým problémům nekalé kooperace, které dále popíši. Členské země mají například zájem na porušování dohodnutých kvót a produkování vyššího množství ropy za účelem krátkodobého zisku i přes to, že tím dlouhodobě poškodí organizaci jako celek.

2.4.1.2 Nekalá spolupráce

V rámci této aplikace budeme uvažovat nekalou spolupráci, kterou není možné zákonně vynutit a pokud bude objevena, dojde k ukončení simulační hry. Navíc se bude vždy jednat o duopol - budou mezi sebou na trhu soupeřit právě dvě firmy. Tyto firmy mají obě za cíl maximalizovat současnou hodnotu toku svých zisků (**Present Value - PV**) vyjádřenou takto:

$$PV(\Pi_i) = \Pi_{i0} + \sum_{t=1}^{n}((\prod_{j=1}^{t}\delta_{ij}) * \Pi_{it}) \quad i \ = 1, 2$$

kde Π_{it} je zisk i-té firmy v období t a δ_{ij} je diskontní faktor i-té firmy, který říká, jak si v období $(j-1)$ považuje firma i prostředky dostupné v období j. Pro zjednodušení výpočtů v rámci této aplikace budeme uvažovat **tržní úrokovou míru** R, za kterou je možné si půjčovat peníze. Můžeme poté uvažovat stálý a jednotný diskontní faktor $\delta = \frac{1}{1+R}$ a zjednodušený vzorec:

$$PV(\Pi_i) = \sum_{t=0}^{n}((\delta)^t * \Pi_{it}) = \sum_{t=0}^{n}\frac{\Pi_{it}}{(1+R)^t} \quad i \ = 1, 2$$

2.4.2 Konečné opakované hry

Uvažujme nyní opakovanou hru, která má známý a konečný počet opakování n. Motivace ke spolupráci existuje, pokud mají firmy jistotu, že mohou potrestat odchýlení od dohody. Tento trest se může projevit například tím, že firma začne místo dohodnutého množství produkovat výstup odpovídající Cournotovu ekvilibriu. Pro zjednodušení budeme uvažovat, že přínos z porušení dohody je menší než diskontovaná újma ze ztráty kooperace v dalším kole.

V posledním kole n však už porušení dohody potrestat nebude možné a proto obě firmy budou v tomto období automaticky produkovat Cournotovy výstupy a nedojde k dohodě. Takto se ale hrozba potrestání stane neefektivní i v předposledním kole $n-1$, neboť obě firmy ví, že v posledním kole ke spolupráci nedojde, musí tedy zvážit produkování Cournotových výstupů i v tomto kole. Indukční metodou se lze postupně propracovat zpět na začátek hry s tím výsledkem, že spolupráce nebude uzavřena v žádném kole.

2.4.3 Opakované hry s neznámým počtem kol

Pro potřeby této aplikace není prakticky možné, aby byl počet období nekonečný. Můžeme ale uvažovat hru, která sice bude mít konečný počet kol, ale tento počet nebude dopředu známý. Označme F **pravděpodobnost selhání koluze** v kterémkoliv kole z jiných důvodů, než je vlastní rozhodnutí firmy pro ukončení spolupráce. Může se jednat například o zákonnou intervenci, selhání trhu atp. Pravděpodobnost přežití firmy v jednom kole lze pak zapsat jako $S = 1 - F$. V rámci simulace bude parametr F volitelný a pokud v daném kole dojde k tomuto selhání, hra bude ukončena a vyhodnocena. V dalším textu se budeme zabývat již jen opakovanými hrami s nejistým počtem opakování.

2.4.4 Strategie trestání odchylky od spolupráce

Předpisu, který stanoví, co výrobce provede v kole následujícím po tom, co jeho partner v koluzi dohodu poruší, se říká **spouštěcí strategie**. Nazývá se tak proto, že definuje způsob, jakým se spouští trestání odchylek od dohody. V následujícím textu si definujeme několik základních spouštěcích strategií, které bude možné v rámci aplikace simulovat. Tyto strategie vychází z nepublikovaného textu Ing. Ivo Koubka z Institutu ekonomických studií Fakulty sociálních věd Univerzity Karlovy. [11]

2.4.4.1 Trestání Cournotovou konkurencí

Uvažujme výchozí situaci, kdy spolu firmy na trhu spolupracují. Pokud jedna z firem v určitý moment dohodu poruší, spolupráce bude trvale ukončena a oba výrobci budou v následujícím kole a všech dalších kolech vyrábět své Cournotovy výstupy. Situaci, kdy se firma od spolupráce takto odklonila označíme jako **renegátství**. Firma i bude spolupracovat v daném kole, pokud přínos z renegátství v tomto kole bude menší než diskontované obětované přínosy ze spolupráce ve všech následujících kolech. Tuto situaci je možno vyjádřit následující nerovnicí:

$$\Pi_{i(rq)} - \Pi_{ik} < \sum_{j=1}^{\infty} \left(\frac{(\Pi_{ik} - \Pi_{ic})}{1+R} \right)^j = \frac{(\Pi_{ik} - \Pi_{ic})}{R} \quad i \ = 1,2$$

kde $\Pi_{i,k}$ je podíl i-té firmy ze společného maximalizovaného zisku, $\Pi_{i(rq)}$ je zisk z porušení dohody a Π_{ic} je zisk v případě, že se obě firmy navrátí k produkování Cournotových výstupů. Vyjádříme-li z této nerovnice R, dostaneme následující výraz:

$$R < \frac{\Pi_{ik} - \Pi_{ic}}{\Pi_{i(rq)} - \Pi_{ik}} \qquad i \ = 1,2 \tag{2.1}$$

který nám udává úrokovou míru, pro kterou bude možné spolupráci udržet libovolně dlouho a pro kterou zároveň bude nekalá kooperace Nashovou rovnováhou této opakované hry. Experimentálně je možné ukázat, že tato podmínka je poměrně snadno splnitelná.

Podívejme se nyní, jak se změní podmínky pro udržení spolupráce, vezmeme-li v úvahu pravděpodobnost selhání koluze z důvodů, které nezávisí na vlastním rozhodnutí firmy. Tuto pravděpodobnost jsme již dříve označili jako F. Pro jednoduchost budeme uvažovat, že se tato hodnota v průběhu hry nebude měnit, potom můžeme pravděpodobnost pokračování koluze v n-tém kole vyjádřit jako $(1-F)^n$. Zahrnutím této pravděpodobnosti získáme následující podmínku udržení nekalé kooperace:

$$\Pi_{i(rq)} - \Pi_{ik} < \sum_{j=1}^{\infty} (\frac{1-F}{1+R})^j (\Pi_{ik} - \Pi_{ic}) \quad i \ = 1,2$$

a po úpravách lze získat zjednodušenou podmínku:

$$R < (\frac{R+F}{1-F}) < \frac{\Pi_{ik} - \Pi_{ic}}{\Pi_{i(rq)} - \Pi_{ik}} \quad i \ = 1,2 \tag{2.2}$$

ze které je na první pohled vidět, že horní mez všech úrokových měr pro které lze udržet koluzi bude mnohem nižší. Tyto podmínky budou zejména důležité při implementaci počítačových protivníků v rámci simulace.

2.4.4.2 Minimaxové trestání

„Minimaxové trestání je taková akce trestající firmy, která v případě, že trestaná firma uskuteční svou nejlepší reakci, přináší trestané firmě ten nejhorší možný výsledek.“ [11] Pokud budeme uvažovat, že firma 1 trestá firmu 2, pak lze hledaný penalizační výstup získat řešením následující úlohy:

$$\min_{q_1} \max_{q_2} \Pi_2(q_1, q_2) = min_{q_1} \Pi_2(q_1, R_2(q_2)) \tag{2.3}$$

kde R_2 reprezentuje nejlepší reakci firmy 2 vzhledem k výstupu první firmy. Problém s touto strategií v rámci oligopolu je, že trestáním renegáta trestáme taky sebe samotné. V případě rostoucích nákladů volbou penalizačního výstupu dokonce sami sebe trestáme více než soupeře. Zavedeme proto následující omezení:

- Žádnou firmu nelze přimět, aby přijala trest v podobě záporného zisku. Nejtvrdší trest, který lze proto udělit je snížení jejího zisku na nulu.
- Firma nemůže zvolit napořád trestání, které by jí jako trestajícímu přineslo záporný zisk. Taková hrozba by také byla těžko uvěřitelná.

Nejprve budeme uvažovat situaci na homogenním trhu (který má inverzní křivku poptávky vyjádřenou jako $P = a - b \times Q$), kde mají obě firmy stejné konstantní mezní náklady c. Zvážíme-li všechny možné tržní konfigurace, dojdeme k závěru, že nejtvrdší penalizační výstup, který může firma zvolit je $q_{i,px} = \frac{a-c}{b}$. Jedná o rovnovážný stav, ve kterém se cena produktu rovná mezním nákladům ($P = c$) a zisky obou výrobců ($\Pi_i = (P - c) \times q_i$) se rovnají nule. Z tohoto vyplývá, že také výstup trestané firmy bude roven nule. Jelikož trestaná firma nemá důvod spolupracovat na svém trestání a není možné zvolit nižší výstup než je nulová výroba, nebude trestaná firma vyrábět nic.

Složitější situace nastávají v případě diverzifikovaného trhu a rostoucích mezních nákladů. V rámci mojí aplikace uvažujeme pouze trhy homogenní, zaměříme se proto na situaci, kdy firmy mají rostoucí mezní náklady. Aby byla v této úloze kooperace udržitelná při trestání penalizačním výstupem q_{ip} musíme dodržet podmínky dané nerovnicí 2.2 a provést minimalizační úlohu 2.3. Uvažujeme-li situace ve které firma 1 (produkující výstup q_1) trestá firmu 2 (produkující výstup q_2) a přidáme-li dříve definovaná omezení, dostaneme následující optimalizační úlohu:

$$\begin{aligned} & min_{q_1} \Pi_2(q_1, R_2(q_2)), \\ & \Pi_1(q_1, R_2(q_2)) \geq 0, \\ & \Pi_2(q_1, R_2(q_2)) \geq 0. \end{aligned} \tag{2.4}$$

Protože lze v úloze $min_{q_1}\Pi_2(q_1, R_2(q_2))$ zvyšováním výstupu dosáhnout záporného zisku, stačí nám řešit splnění zbývajících dvou podmínek. Pro oba výše uvedené příklady však platí, že má-li být kooperace udržitelná při trestání penalizačním výstupem q_{ip} jednou provždy musí splňovat podmínku:

$$\frac{R+F}{1-F} < \frac{\Pi_{jk} - \Pi_{j(px)}}{\Pi_{j(rq)} - \Pi_{jk}}, \quad j = 1,2 \quad \text{kde} \quad \Pi_{j(px)} = \Pi_j(q_{ip}, R_j(q_{ip})) \tag{2.5}$$

která je důležitá z teoretického hlediska, neboť jsme doposud vždy uvažovali, že při porušení dohody bude zahájeno trestání, které bude trvat až do konce hry. V reálných situacích se však nezdá pravděpodobné, že by se jedna z firem nepokusila v budoucnu dohodu obnovit, proto si dále v této kapitole představíme spouštěcí strategie, které tuto eventualitu umožňují.

2.4.4.3 Cukr a bič

Jednou z takových strategií je tzv. strategie cukru a biče (angl. carrot and stick). V rámci této spouštěcí strategie požaduje trestající firma, aby se pro obnovení spolupráce trestaný podílel na svém trestání. Pokud tak učiní, je v následujícím kole spolupráce obnovena. Prospěch z porušení dohody je obecně $\Pi_{ir} - \Pi_{ik}$. Předpokládáme-li, že není rozdíl mezi výstupem při pasivním trestání $q_{i(px)}$ a výstupem, který vyrábí firma podílející se aktivně na svém trestání q_{ip}, pak lze odvodit následující zjednodušenou podmínku udržení nekalé spolupráce:

$$\frac{1+R}{1-F} < \frac{\Pi_{ik} - \Pi_{ip}}{\Pi_{ir} - \Pi_{ik}} \quad i = 1,2 \tag{2.6}$$

Jedná se o požadavek, aby diskontovaná očekávaná újma z penalizace v následujícím kole byla větší než současný přínos z odklonu od spolupráce. Pokud ale existuje rozdíl mezi q_{ip} a $q_{i(px)}$, a tedy i zisky Π_{ip} a $\Pi_{i(px)}$, bude třeba vzít v úvahu ještě podmínku 2.5. Tyto dvě podmínky budou splněny, pokud rozdíl mezi újmou pasivně přijímanou a mezi újmou na níž se trestaná firma aktivně podílí, nebude větší než přínos z odklonu od spolupráce. Matematicky vyjádřeno následovně:

$$\Pi_{ir} - \Pi_{ik} \geq \Pi_{i(px)} - \Pi_{ip} \quad i = 1,2$$

Protože se ale jedná o podmínku postačující nikoliv nutnou, která zbytečně zužuje obor hodnot ze kterých můžeme poté vybírat vyráběné množství, vyplatí se nám většinou ověřit obě původní podmínky.

Dalšími zásadními podmínkami účinnosti této strategie trestání je, aby jednorázový prospěch z renegátství byl menší než újma ze střídavé spoluúčasti na svém trestání a tato újma zase menší než újma z navždy opakovaného pasivního přijímání trestů. Tuto situaci je možné vyjádřit následujícím výrazem:

$$\Pi_{ir} - \Pi_{ik} < \frac{1-F}{1+R}(\Pi_{ik} - \Pi_{ip}) < \frac{1-F}{R+F}(\Pi_{ik} - \Pi_{i(px)}) \quad i = 1,2 \tag{2.7}$$

Při splnění těchto podmínek lze říci, že hrozba navždy vyráběného nekooperativního výstupu přináší větší újmu než výnos získaný jednorázovým porušením dohody, a firmy jsou tedy motivované k obnovení spolupráce. Dalšími úpravami pravé nerovnosti v rámci tohoto výrazu můžeme získat následující podmínku:

$$\Pi_{i(px)} - \Pi_{ip} < \frac{1-F}{1+R}(\Pi_{ik} - \Pi_{ip}) \quad i = 1,2$$

která nám říká, že rozdíl v újmě mezi aktivním sebe potrestáním a pasivním přijímáním trestu nesmí být vyšší než diskontovaný očekávaný přínos z obnovené spolupráce v dalším kole. Pro srovnání s předchozími spouštěcími strategiemi je možno podmínku vyjádřit také následovně:

$$\frac{1+R}{1-F} < \frac{\Pi_{ik} - \Pi_{ip}}{\Pi_{i(px)} - \Pi_{ip}} \quad i = 1,2 \tag{2.8}$$

Zásadní výhodu má tato spouštěcí strategie v situacích, kdy je pro trestajícího velmi nákladné pokoušet se dohodu obnovit (v reálných situacích zejména na trzích s diferencovanými produkty nebo v případě strmě rostoucích mezních nákladů). Představme si, že firma, která dohodu porušila má zájem na jejím obnovení. Pokud trestaný spolupracuje na svém potrestání, trestající firma nemůže na trestání prodělat a bude se jednat o důvěryhodnou hrozbu. Takovou spouštěcí strategii můžeme nazvat **odolnou proti opětovnému vyjednávání**. Protože ale v rámci mé aplikace implementuji pouze trh s homogenním zbožím, spouštěcí strategie cukru a biče není použita.

2.4.5 Herní strategie

Představili jsme si dvě základní metody trestání porušení dohody - Cournotovou konkurencí a penalizačním výstupem. Podobně by mohla být zavedena strategie trestání Bertandovou konkurencí, v případě, že bychom uvažovali oligopol, který namísto výrobního množství stanovuje cenu. Dále jsme představili spouštěcí strategii, která umožňuje obnovení dohody v případech, kdy je renegátská firma ochotná podílet se na vlastním trestání. V této podkapitole se dále zaměříme na různé strategie, které mohou firmy uplatnit v dlouhodobém horizontu. Ty je pak možné spárovat s vhodnou spouštěcí strategií pro trestání odchylek od dohody V rámci této aplikace je potom možné soupeřit proti těmto strategiím ve hře pro jednoho hráče.

Abychom mohli strategie precizně popsat, je potřeba problematiku definovat matematicky. Definujme konečnou **množinu hráčů** $P = \{1, 2, ..., n\}$, přičemž víme, že v rámci mojí aplikace bude vždy omezena na dva hráče ($n = 2$). Dále má každý hráč k z množiny P k dispozici konečnou množinu ryzích strategií S_k. Prvky tohoto prostoru, neboli **strategie** označíme s_k, kde $s_k \in S_k$. Množinu prostorů strategií všech hráčů pak označíme jako $S = S_1 \times ... \times S_n$. Konečně definujeme **výplatní funkci** $u : S_1 \times S_2 \times ... \times S_n \to \mathbb{R}$, která reprezentuje výplatu (zisk nebo ztrátu) daného hráče na konci hry. Množina výplat všech hráčů bude označena analogicky jako předchozí množiny $U = \{u_1, u_2, ..., u_n\}$.

Definice 1 *Trojici* $G = < P, S, U >$ *nazveme* ***hra v normálním tvaru*** *pro n hráčů.*

Nyní máme definovanou simulační hru, kterou můžeme libovolně opakovat. Hra samotná je pak rozdělena do časových period $t = 0, 1, 2, ..., T$. Pokud je $T < \infty$ jedná se o hru s **konečným počtem opakování**. Již dříve jsme si uvedli, že v rámci mé aplikace je sice počet opakování konečný, ale nejistý, díky pravděpodobnosti ukončení hry v daném kole F. Budeme proto předpokládat, že se jedná o hru s **nekonečným počtem opakování**. Nechť $a^t \equiv (a_1^t, a_2^t, ..., a_n^t)$ je množina akcí zahraných v periodě t (a tedy a_i^t je akce zahraná hráčem i v periodě t), pak **historii** opakované hry v časové periodě $t \geq 1$ označíme h^t a bude se jednat o posloupnost množin akcí z časových period předcházejících t:

$$h^t = \left(a^0, a^1, a^2, ..., a^{t-1}\right) \qquad t = 1, 2, ...$$

Pro příklad uveďme pátou periodu hry opakovaného vězňova dilematu $h^4 = ((C,C),(C,D),(C,C),(D,D))$, která se skládá z period $0, 1, 2$ a 3. V každé periodě mají hráči na výběr buď spolupracovat (označeno C z anglického cooperate) nebo druhého hráče zradit (označeno D z anglického defect). Toto označení akcí budeme dále používat i při definování strategií firem v rámci oligopolu. Taková strategie bude definována následovně:

$$s_i = \left(s_i(h^0), s_i(h^1), ..., s_i(h^T)\right)$$

Z čehož vyplývá, že hráč bude mít fixní akci v nulté periodě a v každé další bude zvolena akce $a_i \in A_i$ na základě dostupné historie h^t a tedy akcí ostatních hráčů v předchozích periodách. Definice strategií jsou převzaty ze článku B. Slantcheva zabývajícího se problematikou opakovaných her v rámci vězňova dilematu. [12]

2.4.5.1 Naivní strategie

Nejdříve se podíváme na strategie, které nejsou podložené žádnou matematickou teorií, pouze „selským rozumem“. Jednou takovou strategií může být **vždy spolupracovat**, označme tuto strategii **ALL-C** (z anglického always cooperate). Firma využívající tuto strategii se nikdy neodkloní od kooperačního výstupu. Z teorie, kterou jsme doposud představili, je vidět, že to nebude velmi efektivní strategie, pokud se druhá firma rozhodne dohodu porušit. Firma využívající tuto strategii nejen že druhou firmu nepotrestá za porušení dohody, ale protože bude stále vyrábět kooperativní výstup, bude dosahovat nižšího zisku než její konkurent, který produkované množství zvýšil. Strategii můžeme zapsat jako $s_i(h^t) = C$ ve všech časových periodách t.

Pokud firma aplikuje opak předchozí strategie, bude **vždy podvádět** nezávisle na rozhodnutí ostatních hráčů na trhu. Takovou strategii označíme **ALL-D** (anglicky always defect) a můžeme jí zapsat jako $s_i(h^t) = D$ ve všech časových periodách t. Firma aplikující tuto strategii na tom bude lépe než v případě strategie ALL-C, nicméně nebude nikdy schopná dosáhnout nekalé spolupráce a tím i vyššího zisku v rámci tržního optima. Obě tyto strategie jsou velmi jednoduché a s jistotou můžeme říct, že ani u jedné z nich by nebyla nekalá spolupráce Nashovou rovnováhou v rámci opakované hry.

Další o něco sofistikovanější strategie nám předepisuje zvolit výstup **většinovým hlasováním**. Pokud soupeř většinu tahů porušoval dohodu, vyrobíme nekooperativní výstup. Pokud naopak ve většině kol spolupracoval, budeme v následujícím kole produkovat kooperativní výstup. Strategii označíme **MAJ-V** (z anglického majority vote). U této strategie je třeba také zvolit úvodní výstup, kdy nejsou data k dispozici a typ výstupu v případě shody počtu kooperací a odchylek. Tyto výstupy by mohly například být náhodné, ale v rámci této aplikace bude vyráběn v těchto případech kooperativní výstup. Jak si dále ukážeme, nejúspěšnější strategie budou „přátelské“ a proto se budeme tohoto pravidla držet i u této základní strategie.

Poslední naivní strategii, kterou popíšeme je čistě **náhodná** a přidělíme jí zkratku **ALL-R** (z anglického random). Předepisuje hráči, který jí aplikuje kooperovat s 50% pravděpodobností v každém jednotlivém kole. Šance dohodu porušit je pak ekvivalentní. Jednalo se o základní strategii použitou v rámci Axelrodova turnaje, který si dále představíme. Dlužno říci, že strategie se umístila na posledním celkovém místě, pravděpodobně i proto, že byla dopředu ohlášená a mnoho strategií se zaměřovalo právě na její poražení. Nyní uvedeme sofistikovanější strategie, které umožňují dohodu uzavřít a v mnoha případech i dlouhodobě udržet.

2.4.5.2 Axelrodův turnaj

Okolo roku 1980 uspořádal Robert Axelrod dvojici turnajů, ve kterých se měly proti sobě utkat různé strategie soupeřící ve hře řešící vězňovo dilema. Do prvního turnaje Axelrod [13] shromáždil množství strategií od předních odborníků v oboru. Tyto strategie proti sobě poté soupeřily v 200 iteracích simulační hry a na základě výsledku této simulace byl sestaven žebříček nejúspěšnějších strategií. Vítězem tohoto prvního turnaje se stala překvapivě jednoduchá strategie s anglickým názvem „TIT FOR TAT“ (TFT), kterou do soutěže poslal Anatol Rapoport. Na základě výsledků prvního turnaje byl uspořádán turnaj druhý, kde byla snaha zjistit, jestli přijde někdo s lepší strategií, nicméně turnaj opět vyhrála strategie TFT. Aby byl eliminován vliv konečného počtu kol z prvního turnaje, byla do hry zabudována pravděpodobnost konce hry, stejně jako v rámci této aplikace. V rámci druhého Axelrodova turnaje byla v každém kole rovna 3.46%. Více o výsledcích těchto turnajů se lze dočíst v Axelrodově publikaci „The evolution of cooperation“ původně z roku 1984. Nové zrevidované vydání bylo publikováno v roce 2006. [14]

Strategie použité v rámci Axelrodova turnaje jsou velmi zajímavé a proto stojí za to alespoň některé z nich podrobněji představit.

2.4.5.3 Strategie Oko za oko (TFT)

Z anglického tit for tat (TFT), někdy také překládána jako „Jak ty mně, tak já tobě“ nebo „Půjčka za oplátku“. Jedná se o strategii, která zvítězila v prvním a posléze také ve druhém Axelrodově turnaji, kterého se účastnilo 62 strategií, mnohé z nich s cílem porazit právě tuto strategii.

Pro strategii platí, že v jednoduchosti je síla. Přikazuje hráči spolupracovat v prvním období t_0 a následně kopírovat tahy svého soupeře. To lze zapsat následovně:

$$s_i(h_t) = \begin{cases} C & \text{pokud } t = 0 \\ C & \text{pokud } a_{j,\,t-1} = C, \quad i \neq j \\ D & \text{v ostatních případech} \end{cases} \tag{2.9}$$

Hlavní předností této strategie podle R. Axelroda [13] je, že se jedná o „přátelskou“ strategii, která nikdy neiniciuje zradu a snadno odpouští zrádci, který má zájem na obnovení spolupráce (dokonce hned v následující časové periodě). Ze 14 strategií prvního turnaje by se dokonce 8 nejlepších dalo označit za „přátelské“ strategie.

Nelze ovšem říci, že tato strategie nemá své slabiny. Jedním z nedostatků v turnaji bylo, že hráči neměli úplné informace o tom s kým se utkají a nemohli dostatečně přizpůsobit své strategie. V prostředí skutečného trhu jsou firmy často lépe informované a mohou se plně přizpůsobit soupeři. Tato strategie také nikdy nemůže zvítězit v přímé konfrontaci. Zisk firmy, která jí uplatňuje může být nejvýše stejně vysoký jako zisk soupeře. I přes tyto nedostatky se jedná o překvapivě silnou a zároveň velmi jednoduchou strategii, která má v teoretických modelech své místo.

2.4.5.4 Strategie omezené odplaty (LR)

Dnes již existuje mnoho variací na strategii TFT a bylo by těžko možné je všechny vypsat, natož implementovat. Představme si však alespoň ty nejzajímavější z nich. Jednou takovou je **strategie omezené odplaty** (z anglického limited retaliation), kterou do Axelrodova turnaje poslal Martin Shubik a podrobněji strategii vymezil ve své publikaci z roku 1970 [15]. V prvním turnaji se umístila na 5. místě. Tato strategie si pamatuje, kolikrát se soupeř odklonil od dohody a na základě tohoto parametru stupňuje trestání. Označme tento parametr k (na začátku hry platí $k = 0$). Předpis strategie je pak následující:

1. **První fáze:** Spolupracuj a přejdi do druhé fáze.
2. **Druhá fáze:** Tato fáze se dělí na tři základní scénáře:

- Pokud v předchozím období soupeř spolupracoval, vyrob kooperativní výstup.
- Pokud v předchozím období nespolupracovala ani jedna z firem, vyrob nekooperativní výstup.
- Pokud byl vyroben kooperativní výstup, ale soupeř se od dohody odklonil, přejdi do třetí fáze a nastav $\tau = 0$; $k = k + 1$.

3. **Třetí fáze:** Pokud je $\tau < k$, nastav $\tau = \tau + 1$ a v dané periodě nespolupracuj. Pokud je $\tau \geq k$, vrať se zpět do druhé fáze.

Můžeme si všimnout, že strategie nebere v úvahu, jak se druhý hráč chová ve fázi trestání, tím se strategie odlišuje od původní TFT. Jinak řečeno, délka trestání je fixně daná v moment, kdy ke zradě dojde a dokud není trestání dokončeno, nelze vyjednávat.

Turnaje se zúčastnila i další variace této strategie, kterou zaslali T. Nicolas Tideman a Paula Chieruzzi. Jejich strategie však za určitých podmínek počítá s možností odpuštění soupeři a resetováním strategie do bodu nula. V jejich verzi také dochází automaticky ke zradě v posledních dvou periodách, neboť byl v turnaji počet kol dopředu známý. Přestože se jednalo o úspěšnější strategii, která se umístila na 2. místě, byla v rámci této aplikace upřednostněna Shubikova implementace pro svou jednoduchost. [13]

2.4.5.5 Strategie Grim Trigger (GRIM)

Další strategie, kterou si představíme se také zúčastnila prvního Axelrodova turnaje a skončila na 7. místě ze 14 přihlášených strategií. Do turnaje ji poslal James W. Friedman, který o této strategii publikoval článek [16] v roce 1971 navazující na původní popis strategie od R. J. Harrise z roku 1969 [17]. Z tohoto důvodu se někdy strategie nazývá „Friedman“, ale spíše vešla do veřejného povědomí pod názvem **Grim Trigger**.

Hráč aplikující tuto strategii zahájí hru spoluprací, ve které pokračuje, dokud obě firmy dodržují kooperativní výstup. Pokud jedna z firem dohodu poruší, pak tato strategie přikazuje vyrábět penalizační výstup až do konce hry. Definujeme následovně:

$$s_i(h_t) = \begin{cases} C & \text{pokud } t = 0 \\ C & \text{pokud } a^T = (C, C) \text{ pro } T = 0, 1, ..., t-1 \\ D & \text{v ostatních případech} \end{cases} \tag{2.10}$$

Jak je z této definice vidět, strategie připouští i možnost, kdy kooperaci poruší samotný hráč, nikoliv soupeř. Jedná se o drobnou úpravu původní strategie, kterou bychom mohli nazvat například naivní Grim Trigger, ve které, stejně jako u předchozí strategie, hráč nikdy zradu neiniciuje.

Zásadní slabinou této strategie je, že nikdy neodpouští. Po porušení dohody ji už nikdy nelze obnovit, a jak jsme si ukázali v předchozích kapitolách, obnovení dohody může být často ku prospěchu obou stran. Ze stejného důvodu zde není možné použít spouštěcí strategie cukru a biče.

2.4.5.6 Gladsteinova strategie (SD-TFT)

Abychom neopomenuli i druhý Axelrodův turnaj, představíme si některé strategie, které v něm vystupovaly. Jednu takovou strategii do turnaje poslal David Gladstein a v rámci aplikace jí označíme **SD-TFT**. Opět se jedná o adaptaci strategie TFT, která ovšem hru zahájí porušením dohody, aby otestovala reakci soupeře. Pokud soupeř kdykoliv v průběhu hry vyrobí penalizační výstup, strategie se až do konce hry změní v TFT. Pokud však soupeř pokračuje v kooperaci, tato strategie předepisuje střídavě vyrábět kooperativní a renegátský výstup, tak aby poměr renegátských výstupů ku celkové počtu period nebyl vyšší než 0.5, přičemž první perioda je ignorována. Strategie neměla valný úspěch, skončila až na 46. místě ze 63 soutěžících strategií. Zde jí uvádím předně proto, že se jedná o inovativní modifikaci nejpopulárnější strategie TFT a lze na ní demonstrovat, které parametry byly pro vítězství v turnaji klíčové a které naopak příliš důležité nebyly. [14]

2.4.5.7 Championova strategie (C-TFT)

Již bylo zmíněno, že i druhý Axelrodův turnaj vyhrála strategie TFT, uveďme proto strategii, která skončila v tomto turnaji na druhém místě. Do turnaje jí zaslal Danny Champion. Nepřekvapivě se opět jedná o modifikaci strategie oko za oko. V této strategii je hráči předepsáno kooperovat v prvních 10 tazích, následně se držet strategie TFT v dalších 15 tazích. Zaveďme proměnnou C_RATE, která nám udává poměr soupeřových kooperačních výstupů ku jeho renegátským výstupům. Od 26. tahu pak hráč bude kooperovat, dohodu poruší, jen pokud jsou splněny všechny tři následující podmínky:

1. Soupeř v předchozí iteraci vyrobil renegátský výstup.
2. Počet iterací, ve kterých soupeř kooperoval je menší než 60%.
3. Nechť x je náhodné číslo mezi 0 a 1, pak musí platit $x > C\_RATE$.

Stejně jako ostatní strategie na předních příčkách můžeme tuto strategii označit za „přátelskou“. Nikdy neiniciuje zradu a je dokonce opatrnější v trestání zrady. To si můžeme vyložit například tak, že počítá s možností náhodného selhání. V turnaji však ztratila body zejména na tom, že její přílišná důvěřivost v prvních 10 iteracích byla zneužita. [14]

2.4.5.8 Strategie Pavlov (WS-LS)

Snaha předčit strategii TFT pokračovala i nadále a v roce 1993 přišla dvojice Nowak a Sigmund se strategií operující na principu „win-stay, lose-shift“ [18]. Strategii autoři pojmenovali **Pavlov** a podobně jako TFT je překvapivě jednoduchá. Předepisuje hráči spolupracovat v první periodě a poté v každé další, pokud se akce hráčů shodovaly v periodě předchozí. Vyjádřeno následovně:

$$s_i(h_t) = \begin{cases} C & \text{pokud } t = 0 \\ C & \text{pokud } a^{t-1} \in \{(C,C),(D,D)\} \\ D & \text{v ostatních případech} \end{cases} \tag{2.11}$$

Z tohoto předpisu je vidět, že strategie uvažuje při svém rozhodování pouze předchozí kolo a nikdy nepromýšlí dopředu své tahy, odtud pochází její název. Autoři byli překvapení její úspěšností, neboť si vedla velmi dobře proti pokročilým strategiím využívajícím evoluční algoritmy.

Hlavní výhody této strategie oproti TFT jsou dvě. Za prvé je odolná vůči nahodilým chybám. Představme si, že na trhu figurují dvě firmy, které obě aplikují strategii oko za oko. Nejprve mezi nimi probíhá spolupráce, představme si ale, že jedna z firem chybně vyhodnotí výstup té druhé jako odklon od spolupráce a začne proto produkovat penalizační výstup. Druhá firma dohodu v následující periodě také opustí a díky tomu, jak je strategie navržená, se firmy k dohodě již nikdy nevrátí, přestože se jednalo pouze o chybné vyhodnocení tržních podmínek a dohoda porušena nebyla. Druhou výhodou je, že strategie umí zneužít příliš „přátelské“ hráče. Pokud hráč zjistí, že odklon od spolupráce nebude potrestán (například díky neúmyslnému porušení dohody), potom už se strategie ke spolupráci nevrátí, ale bude nadále zneužívat svého soupeře. To je zásadní rozdíl oproti strategii TFT, která by se ke spolupráci vrátila. [18]

2.4.5.9 Další strategie

Množství strategií, které lze pro řešení opakovaného vězňova dilematu použít, je ve své podstatě neomezené. Dokonce i pro výše uvedené strategie bylo navrženo nesčetně úprav a omezení. Tyto základní strategie však patří mezi nejvýznamnější ve svém oboru a opakovaně se objevují v literatuře. Z těchto důvodů jsem je vybral a adaptoval pro opakovanou hru v rámci oligopolu. Design aplikace je však modulární a připouští do budoucna přidání libovolného množství dalších strategií, tak jak se problematika bude dále vyvíjet.

Ve hře pro jednoho hráče je pak možné každou uvedenou herní strategii zkombinovat s relevantní spouštěcí strategií, vybranou z těch, které jsme si představili v podkapitole 2.4.4. Strategie jsou také vázané tržními parametry, zejména **tržní úrokovou mírou** R a **pravděpodobností konce hry** F. Ačkoliv budou tyto parametry nastavitelné, zásadně ovlivňují, jestli bude spolupráce realisticky dosažitelná bez ohledu na zvolenou strategii.

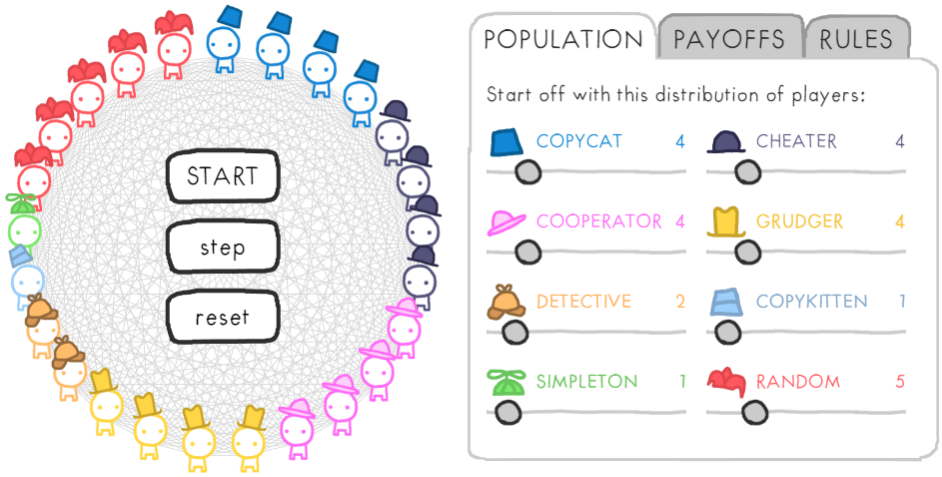
2.5 Existující řešení

Podívejme se nyní jaká řešení již existují v oblastech doposud zmíněných v této práci. Představíme si řešení simulačních her na poli mikroekonomie, matematiky a obecného vzdělávání.

2.5.1 Vězňovo dilema a důvěryhodnost

Velká část informací, které jsme doposud uvedli, pochází z oblasti teorie her. Zabývali jsme se zejména problematikou spolupráce v různých prostředí, jejím vynucením a důvěryhodností hrozby trestáním při porušení dohody.

Zajímavou aplikací, která demonstruje principy probírané v Axelrodově publikaci „The Evolution of Cooperation“ [14], je aplikace **The Evolution of Trust** [19]. V této aplikaci je představena problematika vězňova dilematu a umožňuje proti sobě postavit různé strategie inspirované Axelrodovým turnajem. Mezi ně patří například ALL-C, ALL-D a TFT, které byly dříve představeny v této práci. Celkem umožňuje simulovat 8 různých osobnostních profilů a nechat je soupeřit proti sobě ve vysoce přizpůsobitelné simulační hře. To vše je navíc podáno hravou a uchopitelnou formou v podobě pestrobarevných obrázků a animací. Ukázka z aplikace je vidět na obrázku 2.5. Jedná o velmi všestrannou aplikaci, která jednak dokáže problematiku vysvětlit a demonstrovat, ale zároveň vyzkoušet pokročilé scénáře. Z těchto důvodů aplikaci velmi doporučuji jako úvod do problematiky. Aplikace je dostupná v anglickém jazyce.



Obrázek 2.5: Ukázka aplikace „The Evolution of Trust“ (Zdroj: [19])

Byl jsem překvapen nedostatkem kvalitních aplikací pro simulaci vězňova dilematu v českém jazyce. Na téma je k dispozici množství článků, závěrečných prací a dalších publikací. Z těch je možné čerpat informace a extrapolovat nová řešení, neumožňují však interaktivně testovat a demonstrovat různé scénáře. Jednou aplikací, která simulace umožňuje v českém jazyce je **Server pro podporu výuky teorie her**. [20] Tato aplikace vznikla na základě diplomových prací Martina Hurába a Marka Záškodného z Fakulty elektrotechniky a informatiky v rámci VŠB - Technické univerzity Ostrava. Aplikace umožňuje simulaci kombinatorických her, her s nulovým součtem pro dva hráče a také her s nenulovým součtem, kterými jsme se zabývali v rámci této práce a patří mezi ně například zmiňované vězňovo dilema. Přestože se jedná primárně o výukovou pomůcku v rámci VŠB, může být přínosná pro obecné studium problematiky, obzvláště pro čtenáře, kteří cizojazyčných aplikací nemohou využít. Server je výsledkem kolaborace v rámci několika diplomových prací a dá se předpokládat, že bude v budoucnu dále rozšířen, aby zahrnoval nová řešení a nové oblasti.

V případě, že stručný úvod do problematiky není dostačující, máme možnost využít pokročilejších aplikací. K tomu nám může posloužit například původní zdrojový kód a dokumentace k druhému Axelrodově turnaji. [21] Jelikož byl ale vytvořen v roce 1996, jedná se o program v jazyce Fortran, který je do značné míry odlišný od dnes používaných technologií. Pokud bychom pro snazší uchopení chtěli použít modernější implementaci, je k dispozici knihovna, kterou v roce 2015 vytvořil Vincent Knight a kolektiv autorů. [22] Jedná se o knihovnu v jazyce Python, který patří mezi moderní, aktuálně používané technologie, knihovna je navíc stále udržována a průběžně aktualizována.

2.5.2 Simulace ekonomických modelů

Již jsme si uvedli některé příklady aplikací simulujících teorii her, zaměřme se nyní také na simulace z oblasti mikroekonomie. V první řadě tato práce nepřímo navazuje na mou předchozí práci z roku 2016 nazvanou **Simulace modelu monopolistické konkurence**. [23] Stejně jako u této aplikace se jedná o simulační hru, která má sloužit jako pomůcka pro výuku ekonomických předmětů. Ve zmíněné práci je simulován Salopův kruhový model monopolistické konkurence ve variantách hry pro více hráčů a hry jednoho hráče proti počítači. Díky podobným požadavkům na funkcionality slouží aplikace jako předloha této aplikaci. Konkrétní technologické rozdíly budou dále popsány v implementační části této práce.

Na poli mikroekonomie samozřejmě existuje množství dalších simulačních her s různými úrovněmi realističnosti, hratelnosti a ostatních kritérií. Připomeňme například populární deskovou hru **Monopoly**, poprvé vydanou v roce 1935, která má však ještě hlubší kořeny. Hra vznikla na základě konceptu z roku 1903, který vytvořila Elizabeth Magie v rámci své hry „Landlord's Game". Tato hra měla poukazovat na nebezpečí neregulovaných monopolů a benefity, kterých se dostává spotřebitelům na konkurenčních trzích. Ačkoliv se dnes jedná převážně o rodinnou hru, která nemá příliš vzdělávací charakter, uspěla v upevnění konceptu monopolů v rámci široké veřejnosti. Hry z oblasti oligopolu nikdy nesklidily zdaleka takový úspěch jako Monopoly. V roce 1973 byla vydána desková hra **Cartel**, která se zabývá dynamikou kartelových dohod. Nebyla však velmi populární a ani nedošlo k jejímu vydání v českém jazyce. V pozdějších letech byly několikrát vytvořeny hry s podobným nebo dokonce stejným jménem a konceptem, nikdy však s valným úspěchem. Adaptace pokračují i v roce 2019, z čehož je možné soudit, že trh pro hry s mikroekonomickou tématikou není doposud saturován a je možné, že se v budoucnu úspěšné hry z oblasti probírané touto prací teprve dočkáme.

Přesuneme-li se nyní do oblasti softwaru, v roce 1987 vyšla strategická hra **Oligopoly** pro platformu MS DOS. Hru vydalo americké vývojářské studio XOR Corporation, protože ale jejich poslední titul vyšel v roce 1989, je velmi nepravděpodobné, že v budoucnu bude vydána nová verze této hry pro moderní systémy. Přesto se jednalo o relativně populární hru, která využívala výpočetní možnosti nově dostupných technologií k simulaci reálných tržních situací. Novějším příkladem simulačních her dostupných online je webový portál **Economics Games**. [24] Za projektem stojí **Nicolas Gruyer**, který působí v rámci Toulouské školy ekonomie (Toulouse School of Economics) a **Nicolas Toublanc**, který má na starosti technické zpracování. Aplikace aktuálně nabízí 14 simulačních her pro jednoho hráče a 48 her pro více hráčů, které pokrývají množství ekonomických oblastí, včetně teorie her a problematiky oligopolu. Simulace jsou vytvořené za účelem vzdělávání a podrobně jednotlivé oblasti vysvětlují. K výuce je používají organizace jako Toulouse School of Economics, Paris Dauphine University a další, zejména francouzské, vzdělávací instituty. Aplikace je dostupná ve francouzském a anglickém jazyce.

2.6 Uplatnění v rámci existujících řešení

V této kapitole jsem ukázal, že existuje množství různých simulačních her jak v oblasti teorie her, tak v rámci mikroekonomie. Jedním z nedostatků, které často sdílejí je nedostupnost v českém jazyce, který je v rámci mojí aplikace k dispozici. Dalším důvodem k vytvoření mojí aplikace je unikátní spojení Cournotova modelu oligopolu s oblastí teorie her. Rešerše ukázala, že aplikace, která by se zabývala simulací oligopolu jako opakované hry nebyla doposud vytvořena nebo není veřejně známá.

Aplikace je vytvořena se vzděláváním na prvním místě, navíc společně s aplikací [23] z roku 2016, má potenciál být základem širšího vzdělávacího webu v rámci mikroekonomie a ekonomických předmětů obecně.

Kapitola 3

Analýza požadavků

Zadáním této práce je vytvořit aplikaci simulující spolupráci v oligopolu jako opakované hře vycházející z dříve uvedené teorie. Hlavním požadavkem je, aby aplikace byla dostupná online a využívala moderních technologií, které zajistí stejnou nebo lepší funkčnost jako dříve představená již existující řešení. Aplikace má dále dodržovat principy responzivního návrhu, aby byla dostupná nejen z počítačového prohlížeče, ale zejména z mobilních přístrojů a případně dalších zařízení s přístupem k internetu. Aplikace simulaci umožňuje v režimu hry pro jednoho hráče proti počítačovému protivníkovi využívajícímu některou z dříve uvedených herních strategií. Dále umožňuje hru po síti v režimu dvou hráčů soupeřících proti sobě.

Druhotnými požadavky je pak, aby byla aplikace snadno přeložitelná do cizích jazyků, v prvotní verzi je dostupná v českém a anglickém jazyce s možností budoucího překladu do dalších jazykových mutací. V návaznosti na již existující simulaci monopolistické konkurence [23] byly zvoleny kompatibilní technologie tak, aby byla v budoucnu možná případná integrace do větší vzdělávací platformy.

3.1 Logické členění aplikace

Abychom mohli vybrat vhodné technologie pro realizaci výše uvedených požadavků, je potřeba popsat funkcionality systému. Systém můžeme rozdělit do dvou základních celků nebo také modulů. Prvním z nich je modul statický, který slouží jako úvod do aplikace, obsahuje informace o aplikaci, základní instruktážní texty a možnost přepínat mezi českým a anglickým jazykem. Obsah tohoto modulu je vesměs statického charakteru a vyžaduje minimální interakci s uživatelem. Druhým logickým celkem je herní modul, který disponuje dynamickým obsahem, a jehož součástí je samotná simulační hra. Dále v této kapitole popíši funkční požadavky na systém formou případů užití a uvedu výčet nefunkčních nebo také doplňkových požadavků.

3.2 Případy užití systému

Případy užití má smysl uvažovat pouze u herního modulu, protože statický modul obsahuje jen triviální případy užití. Herní modul můžeme dále rozdělit do dvou stavů. Prvním je přípravná fáze, ve které uživatel čeká na připojení do hry nebo hru zakládá. Druhým je simulační fáze, ve které se uživatel aktivně účastní herní simulace. Pro tyto dva stavy případy užití detailněji popíši.

3.2.1 Přípravná fáze herního modulu

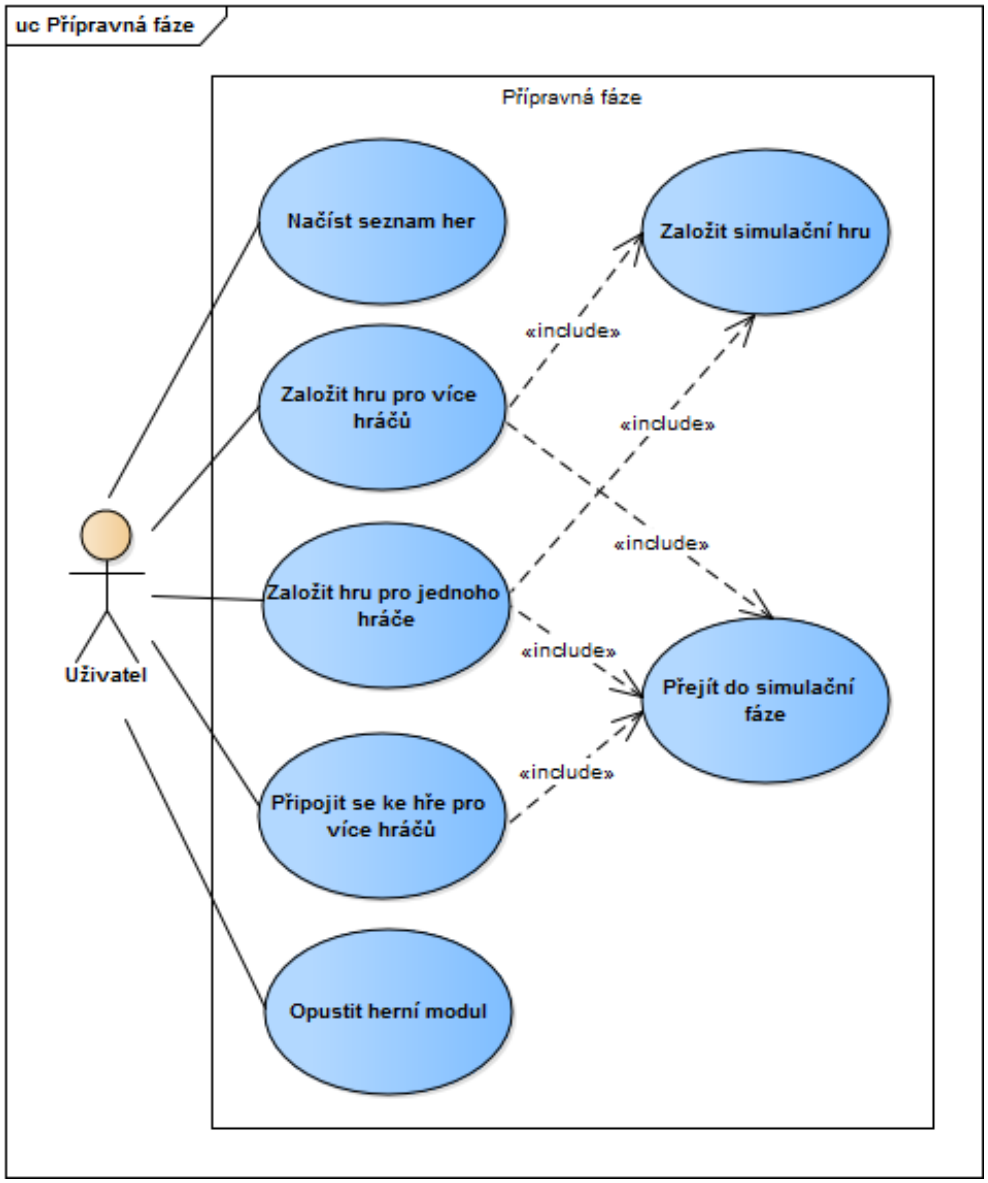
Přípravná fáze herního modulu má jednoho aktéra - uživatele. Ten má možnost stát se hostitelem simulační hry. V případě hry pro jednoho hráče to znamená založit novou hru, čímž získá roli hostitele, ale zároveň bude v roli hráče proti počítačem řízené firmě. V případě hry pro více hráčů se při založení nové hry stane uživatel pouze hostitelem a ve hře působí jako nezávislý pozorovatel. Ke hře se dále mohou připojit ostatní uživatelé, kteří zaujmou role hráčů a to až do dosažení kapacity hry. Všechny scénáře dostupné v této fázi jsou vyobrazeny na obrázku 3.1 a také popsány dále v této kapitole.

3.2.2 Scénáře přípravné fáze

1. **Načíst seznam her**

   - **Popis:** Případ užití umožňuje uživateli vypsat hry pro více hráčů ke kterým se lze připojit.
   - **Aktéři:** Uživatel
   - **Podmínky spuštění:** Uživatel se nachází v přípravné fázi herního modulu
   - **Základní tok událostí:**
     a) Uživatel zvolí záložku „Seznam her“
     b) Uživatel stiskne tlačítko pro obnovení seznamu her
     c) Systém na straně uživatele odešle serveru zprávu o události
     d) Systém na straně serveru zpracuje žádost a odešle uživateli zprávu obsahující seznam aktivních her
     e) Systém na straně uživatele přijme zprávu, zpracuje seznam her a zobrazí jej uživateli
   - **Alternativní tok událostí:** Zaznamená-li server vytvoření nové hry a aktér se nachází v přípravné fázi herního modulu, seznam her je obnoven automaticky bez nutnosti stisknutí tlačítka.

2. **Založit hru pro více hráčů**



Obrázek 3.1: Model případů užití v přípravné fázi herního modulu

- **Popis:** Případ užití umožňuje uživateli stát se hostitelem hry pro více hráčů. Hostitel má přístup k ovládacím prvkům hry a je pozorovatelem po dobu trvání hry. Hra je aplikací ukončena v momentě, kdy jí hostitel opustí.
- **Aktéři:** Uživatel
- **Podmínky spuštění:** Uživatel musí vyplnit vstupní údaje nutné pro běh hry a na serveru musí být volné místo pro novou hru.

- **Základní tok událostí:**
  a) Uživatel v přípravné fázi herního modulu zvolí záložku pro založení nové hry pro více hráčů
  b) include(Založit simulační hru)
  c) include(Přejít do simulační fáze)
  d) Hra je zpřístupněna uživatelům, kteří se k ní mohou připojit ze záložky „Seznam her“. Systém na straně serveru čeká na zahájení hry hostitelem
- **Alternativní tok událostí:** Nastane-li chyba v bodě b), pak je scénář přerušen a uživatel se jej může pokusit spustit znovu od začátku.

3. **Založit hru pro jednoho hráče**

- **Popis:** Případ užití umožňuje uživateli zahájit simulační hru proti počítačem řízené firmě. Uživatel přechází do role hostitele a má k dispozici ovládací prvky hry. Zároveň ale také získává roli hráče a má k dispozici hráčské ovládání. Hra je ukončena ve chvíli, kdy jí uživatel opustí.
- **Aktéři:** Uživatel
- **Podmínky spuštění:** Uživatel musí vyplnit vstupní údaje nutné pro běh hry.
- **Základní tok událostí:**
  a) Uživatel v přípravné fázi herního modulu zvolí záložku pro založení hry pro jednoho hráče
  b) Uživatel vyplní nastavení specifická pro hru jednoho hráče
  c) include(Založit simulační hru)
  d) Systém do hry doplní na pozici druhého hráče počítačového protivníka podle nastavení daných uživatelem
  e) include(Přejít do simulační fáze)
  f) Systém na straně serveru čeká na zahájení hry hostitelem
- **Alternativní tok událostí:** Nastane-li chyba v bodě c), pak je scénář přerušen, systém vypíše chybovou hlášku a uživatel se jej může pokusit spustit znovu od začátku.

4. **Připojit se hře pro více hráčů**

- **Popis:** Případ užití umožňuje uživateli připojení ke hře pro více hráčů, kterou jiný uživatel hostuje. Uživatel získává roli hráče a s ní spojené ovládací prvky v rámci hry. Ke hře je připojený, dokud jí on nebo hostitel neopustí.

- **Aktéři:** Uživatel
- **Podmínky spuštění:** Hra na serveru existuje a není naplněna její kapacita hráčů.
- **Základní tok událostí:**
  a) Uživatel v přípravné fázi herního modulu zvolí záložku pro seznam probíhajících her
  b) Uživatel vyplní své herní jméno (přezdívku)
  c) Uživatel zvolí ze seznamu hru, ke které se chce připojit
  d) Uživatel potvrdí připojení ke hře stiskem tlačítka
  e) Systém na straně serveru ověří existenci hry a její kapacitu
  f) Systém přidělí uživateli ovládání jedné ze dvou společností v herním objektu (podle pořadí ve kterém se hráči připojili)
  g) Systém pošle ostatním uživatelům připojeným ke hře informace o novém hráči
  h) Systém předá uživateli údaje k vykreslení herní plochy
  i) include(Přejít do simulační fáze)
  j) Systém a uživatel čekají na zahájení hry hostitelem
- **Alternativní tok událostí:** Pokud v bodě b) není vyplněno uživatelské jméno, aplikace vypíše chybovou hlášku a uživatel musí před pokračováním chybu napravit. Pokud v bodě c) není vybrána žádná hra, tlačítko pro připojení ke hře je znepřístupněno. Pokud v bodě d) nastane chyba (kapacita hry byla naplněna, hra neexistuje, ...), aplikace vypíše chybovou hlášku a uživatel musí proces opakovat od bodu b).

5. **Opustit herní modul**

- **Popis:** Případ užití zajišťuje zpracování dat, které zůstanou v systému po odchodu uživatele.
- **Aktéři:** Uživatel
- **Podmínky spuštění:** Uživatel ukončil aplikaci.
- **Základní tok událostí:**
  a) Uživatel se nachází v herním modulu a zavře záložku prohlížeče s aplikací
  b) Systém na straně serveru přijme informační zprávu o odchodu uživatele
  c) Systém vyprázdní firmy, které uživatel ovládal a předá informaci o odchodu uživatele ostatním uživatelům, kteří s ním hru sdíleli
  d) Systém ukončí hry, které uživatel hostoval

6. **Založit simulační hru**

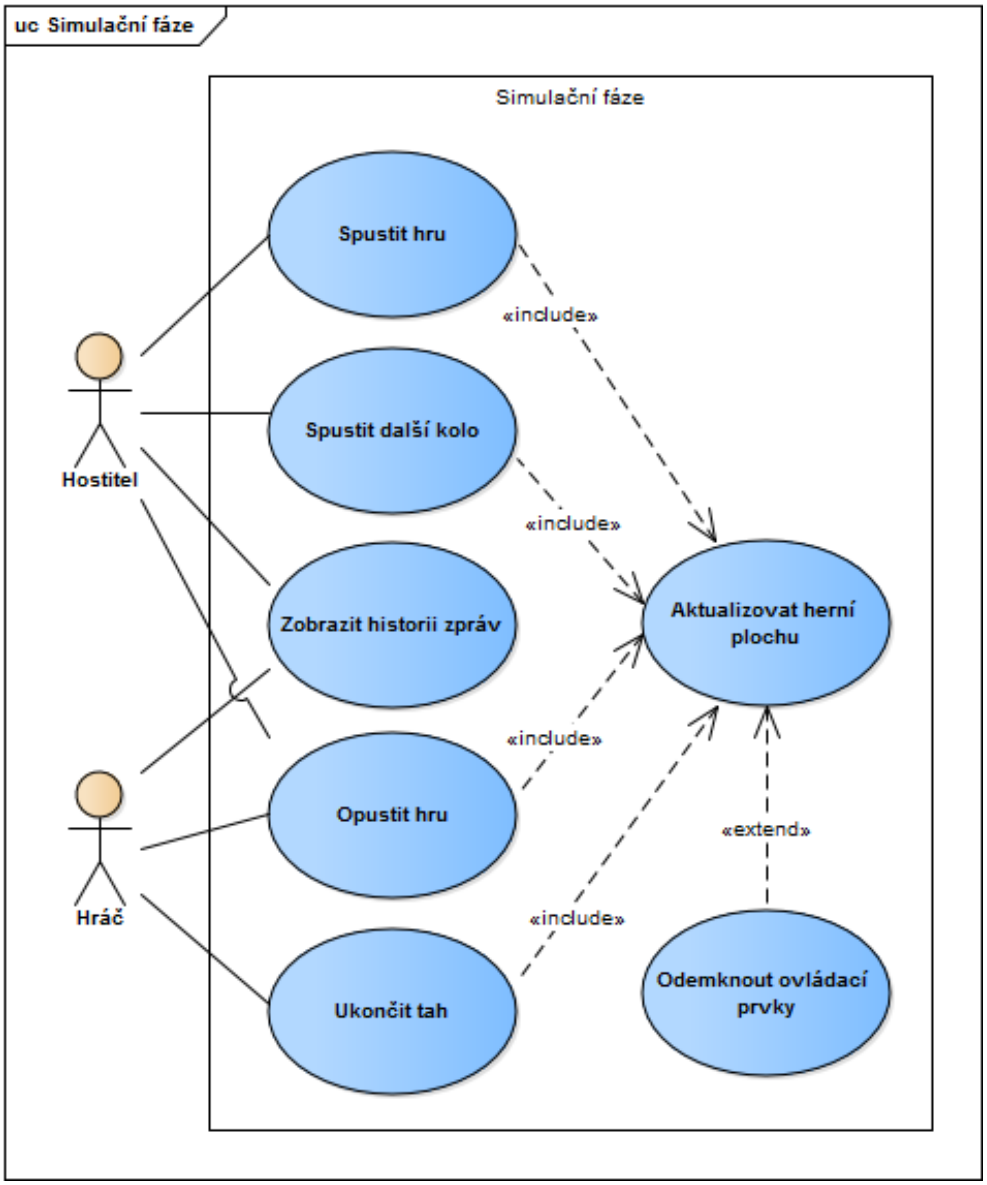
   - **Popis:** Případ užití popisuje proces vytvoření hry sdílený mezi hrou pro jednoho a více hráčů.
   - **Aktéři:** Uživatel
   - **Podmínky spuštění:** Uživatel se nachází na stránce určené pro vytvoření hry
   - **Základní tok událostí:**
     a) Uživatel vyplní všechny položky formuláře
     b) Uživatel potvrdí vytvoření nové hry stiskem tlačítka
     c) Systém ověří platnost údajů vložených uživatelem a ověří, jestli hra může být založena
     d) Systém založí herní objekt pomocí uživatelových vstupních dat a předá uživateli údaje k vykreslení herní plochy
     e) Uživatel je zaznamenán jako hostitel hry a s jeho spojením je svázána existence hry
   - **Alternativní tok událostí:** Pokud v bodě a) nejsou vyplněny všechny položky formuláře, systém vypíše chybovou hlášku a uživatel formulář musí opravit. Pokud je v bodě c) dosažen na serveru maximální počet souběžných her nebo nastane jiná chyba, uživateli je vypsána chybová hláška a může se pokusit proces opakovat od začátku.

7. **Přejít do simulační fáze**

   - **Popis:** Případ užití zajišťuje první vykreslení herní plochy
   - **Aktéři:** Uživatel
   - **Podmínky spuštění:** Uživatel vytvořil novou hru nebo se k ní připojil
   - **Základní tok událostí:**
     a) Uživatel se nachází v herním modulu
     b) Uživatel se stane hostitelem nebo hráčem v rámci systému
     c) Systém pošle uživateli informace nutné pro vykreslení herní plochy
     d) Systém na straně uživatele (prohlížeč) vykreslí herní plochu

3.2.3 Simulační fáze herního modulu

Simulační fáze má aktéry dva, uživatel může vystupovat jako hostitel, hráč nebo může působit v obou rolích zároveň. Cílem této fáze je zahájit hru na vykreslené herní ploše, zpracovat tahy jednotlivých hráčů a při opuštění hry vrátit uživatele zpět do přípravné fáze. Scénáře popisující realizaci těchto cílů jsou uvedeny na obrázku 3.2 a dále popsány.



Obrázek 3.2: Model případů užití v simulační fázi herního modulu

3.2.4 Scénáře simulační fáze

1. **Spustit hru**

   - **Popis:** Případ užití umožňuje hostiteli zahájit hru. Zahájením hry se novým uživatelům uzavře přístup do hry a prvnímu hráči na tahu se odemknou ovládací prvky.
   - **Aktéři:** Hostitel

- **Podmínky spuštění:** Hostitel se musí nacházet v platné hře, musí být jejím vlastníkem a hra musí mít naplněnou kapacitu hráčů
- **Základní tok událostí:**
  a) Hostitel se nachází v simulační fázi herního modulu
  b) Hostitel stiskne tlačítko pro zahájení hry
  c) Systém na straně serveru ověří identitu hostitele
  d) include(Aktualizovat herní plochu)
  e) Systém předá kontrolu nad herní plochou prvnímu hráči v pořadí
  f) Systém čeká na přijetí informací od hráče na tahu nebo jeho odpojení
- **Alternativní tok událostí:** Pokud se v bodě a) hráč nenachází ve správné fázi herního modulu, systém vypíše chybovou hlášku a hostitel musí pokračovat jiným scénářem. Pokud v bodě c) selže ověření na straně serveru, systém vypíše na straně klienta chybovou hlášku a hostitel může scénář opakovat nebo zvolit jiný scénář.

2. **Spustit další kolo**

- **Popis:** Případ užití umožňuje hostiteli zahájit další herní periodu. Zahájením nového kola je předána kontrola nad herní plochou dalšímu hráči v pořadí.
- **Aktéři:** Hostitel
- **Podmínky spuštění:** Hostitel se musí nacházet v platné hře a musí být jejím vlastníkem. Navíc je nutné, aby při založení hry hostitel zvolil manuální spouštění tahů.
- **Základní tok událostí:**
  a) Hostitel se nachází v simulační fázi herního modulu
  b) Hostitel stiskne tlačítko pro zahájení nového kola
  c) Systém na straně serveru ověří identitu hostitele
  d) include(Aktualizovat herní plochu)
  e) Systém předá kontrolu nad herní plochou dalšímu hráči v pořadí
  f) Systém čeká na přijetí informací od hráče na tahu nebo ukončení hry
- **Alternativní tok událostí:** Pokud se v bodě a) hráč nenachází ve správné fázi herního modulu, systém vypíše chybovou hlášku a hostitel musí pokračovat jiným scénářem. Pokud v bodě c) selže ověření na straně serveru, systém vypíše na straně klienta chybovou hlášku a hostitel může proces opakovat nebo pokračovat jiným scénářem.

3. **Zobrazit historii zpráv**

   - **Popis:** Případ užití umožňuje hráči nebo hostiteli zobrazit historii informačních zpráv získaných ze serveru.
   - **Aktéři:** Hostitel, Hráč
   - **Podmínky spuštění:** Aktér se musí nacházet v platné hře.
   - **Základní tok událostí:**
     a) Aktér se nachází v simulační fázi herního modulu
     b) Aktér stiskne tlačítko pro otevření historie zpráv
     c) Systém na straně klienta zobrazí uloženou historii komunikačních zpráv ze serveru
   - **Alternativní tok událostí:** Pokud se v bodě a) aktér nenachází ve správné fázi herního modulu, tlačítko pro zobrazení historie nebude přístupné. Pokud v bodě b) žádná historie zpráv neexistuje, systém zobrazí informaci o této skutečnosti.

4. **Opustit hru**

   - **Popis:** Případ užití umožňuje hráči nebo hostiteli opustit aktivní hru v rámci simulační fáze herního modulu.
   - **Aktéři:** Hostitel, Hráč
   - **Podmínky spuštění:** Aktér se musí nacházet v platné hře.
   - **Základní tok událostí:**
     a) Aktér se nachází v simulační fázi herního modulu
     b) Aktér opustí hru stisknutím tlačítka pro opuštění hry
     c) Pokud aktérem byl hráč, server jej vyjme ze hry, ve které se nacházel a oznámí jeho odchod ostatním účastníkům hry
     d) Pokud aktérem byl hostitel, server ukončí hru, kterou hostoval a oznámí ukončení hry ostatním hráčům
     e) include(Aktualizovat herní plochu)
   - **Alternativní tok událostí:** Pokud se v bodě a) aktér nenachází ve správné fázi herního modulu, tlačítko pro ukončení není k dispozici. Pokud je v bodě b) hra opuštěna uzavřením záložky v prohlížeči, spustí se scénář 5 přípravné fáze (**Opustit herní modul**).

5. **Ukončit tah**

   - **Popis:** Případ užití umožňuje hráči, který je momentálně na tahu, svůj tah ukončit a předat aplikaci informace o produkovaném množství.
   - **Aktéři:** Hráč

- **Podmínky spuštění:** Hráč se musí nacházet v platné hře a musí být na tahu.
- **Základní tok událostí:**
  a) Hráč se nachází v simulační fázi herního modulu a je v rámci hry na tahu
  b) Hráč vyplní množství, které chce v daném kole vyprodukovat
  c) Hráč stiskne tlačítko pro ukončení tahu
  d) Systém ověří identitu hráče a jeho oprávnění
  e) Systém předá tah dalšímu hráči v pořadí nebo hostiteli, pokud hráč byl posledním hráčem v daném kole
  f) Pokud došlo k ukončení kola, potom include(Aktualizovat herní plochu)
  g) Systém čeká na ukončení tahu následujícího hráče, jeho odpojení nebo na zahájení dalšího kola
- **Alternativní tok událostí:** Pokud se bodě a) hráč nenachází ve správné fázi herního modulu, systém vypíše chybovou hlášku a hráč musí pokračovat jiným procesem. Pokud v bodě c) nemá hráč oprávnění ukončovat tah, systém vypíše chybovou hlášku a hráč musí scénář zopakovat až přijde na tah. Pokud v bodech e) nebo g) nezbývají žádní hráči, systém hru ukončí.

6. **Aktualizovat herní plochu**

- **Popis:** Případ užití umožňuje aktérovi aktualizovat svou herní plochu tak, aby byla synchronizovaná se stavem na serveru.
- **Aktéři:** Hráč, Hostitel
- **Podmínky spuštění:** Aktér se musí nacházet v platné hře.
- **Základní tok událostí:**
  a) Aktér A se nachází v simulační fázi herního modulu
  b) Aktér B provede změnu na herním plánu
  c) Systém aktualizuje herní objekt a rozešle všem aktérům přítomným ve hře aktuální informace o stavu herní plochy
  d) Systém na straně klienta (prohlížeč) vykreslí aktuální stav herní plochy
  e) extend(Odemknout ovládací prvky)
- **Alternativní tok událostí:** Pokud se v bodě a) aktér nenachází ve správné fázi herního modulu, scénář se neprovede.

7. **Odemknout ovládací prvky**

- **Popis:** Případ užití umožňuje aktérovi získat přístup k ovládacím prvkům, pokud má oprávnění je použít.

- **Aktéři:** Hráč, Hostitel
- **Podmínky spuštění:** Aktér se musí nacházet v platné hře a musí být k akci zpřístupněné odemčením oprávněn.
- **Základní tok událostí:**
  a) Aktér A se nachází v simulační fázi herního modulu
  b) Aktér B provede akci a předá kontrolu systému
  c) Systém vyhodnotí, že vyžaduje akci od aktéra A a odešle zprávu o odemčení ovládání
  d) Systém na straně klienta (prohlížeč) vykreslí aktualizované ovládací prvky
- **Alternativní tok událostí:** Pokud se v bodě a) aktér nenachází ve správné fázi herního modulu, scénář se neprovede.

3.3 Nefunkční požadavky

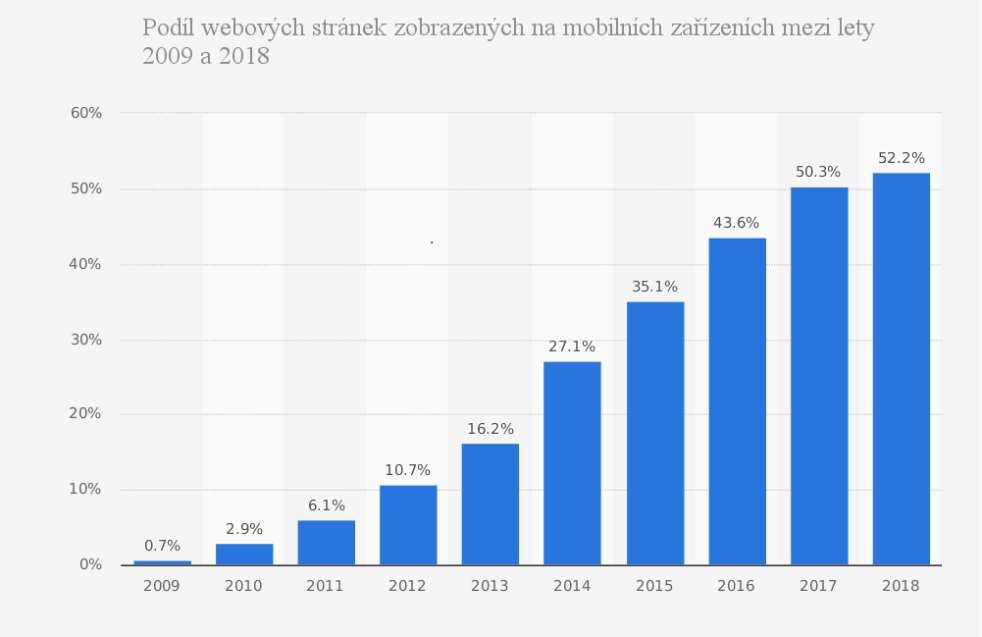
V této kapitole se zaměříme na další požadavky, které nebyly vymezeny v rámci případů užití. Jedná se například o požadavek, aby byla aplikace přístupná z mobilních zařízení. Tyto požadavky je důležité vymezit, abychom na jejich základě mohli následně zvolit vhodné řešení.

3.3.1 Responzivní návrh

Centrálním požadavkem na aplikaci je, aby byla dostupná z webového prohlížeče na co nejširším spektru zařízení. Podíl přístupů k webovým stránkám z mobilních zařízení se každoročně zvyšuje. Zatímco v roce 2014 tvořila mobilní zařízení zhruba čtvrtinu všech návštěvníků, v roce 2018 se jednalo o více než polovinu [25]. Prudký růst tržního podílu můžeme vidět na obrázku 3.3. Díky tomuto vývoji existuje množství vhodných technologií, které se specializují na zajištění responzivního návrhu a v dalších kapitolách si představíme ty, které byly použity po vývoj této aplikace.

3.3.2 Stabilita a výkon aplikace

Od každé moderní aplikace se očekává, že bude dostupná s nízkou odezvou a vysokou úrovní dostupnosti. Při výběru technologií je třeba zvážit, kolik uživatelů bude aplikaci používat současně, a jak bude aplikace nakládat s daty. Zde se jedná o aplikaci malého až středního rozsahu a očekává se nízký počet souběžných uživatelů (desítky až potenciálně stovky). Aplikace je také navržena tak, aby mohlo běžet více instancí aplikace a přirozeně tak distribuovat zátěž. Jedním z důvodů je například to, že aplikace nevyžaduje ukládání dat do databáze, ale pracuje pouze s daty v paměti, které jsou relativně malého rozsahu s krátkou dobou platnosti. Tyto požadavky mají vliv na výběr serverové technologie a programovacích jazyků.



Obrázek 3.3: Vývoj trhu s mobilními zařízeními (Zdroj: [25])

3.3.3 Komunikace server-klient

Pro komunikaci mezi serverem a klientem také existuje velké množství komunikačních protokolů. Aplikace podporuje jednak přenos dat, která nejsou citlivá na rychlé doručení, ale také obsahuje simulační hru, která probíhá mezi uživateli v reálném čase. Zde je potřeba využít takové technologie, které zajistí precizní přenos dat s nízkou odezvou, tak aby uživatelé měli kvalitní zážitek z používání aplikace.

3.3.4 Bezpečnost aplikace

Dalším běžným požadavkem je, aby technologie použité při vývoji dodržovaly moderní bezpečnostní standardy. Zde se jedná zejména o použití takových externích knihoven, které neobsahují žádná známá bezpečnostní rizika.

3.3.5 Zpětná kompatibilita

Vedlejším požadavkem je také použít takové technologie, které zaručí zpětnou kompatibilitu v případě, že by v budoucnu byla aplikace integrována se simulátorem monopolistické konkurence [23], případně dalšími aplikacemi podobného charakteru do širšího vzdělávacího systému. Nejedná se však o prioritní požadavek a mohou být zvoleny modernější technologie, budou-li to vyžadovat ostatní požadavky.

Kapitola 4

Návrh řešení

V této kapitole popíši technologie, které jsem zvolil na základě provedené analýzy požadavků. To povede k získání uceleného přehledu o systému.

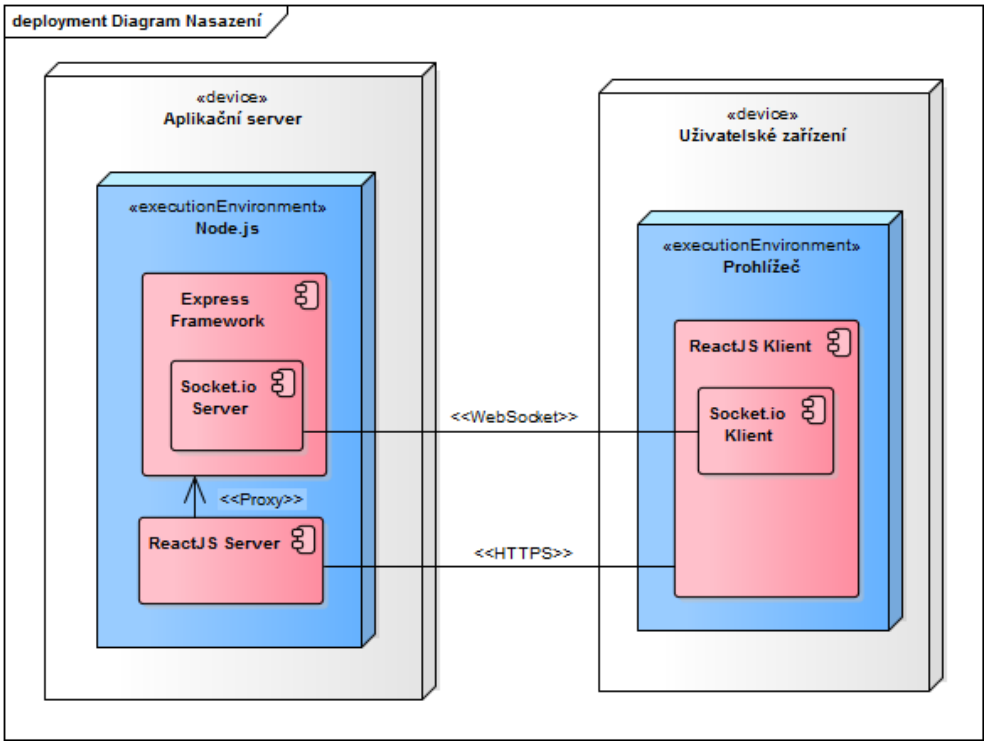
4.1 Jádro systému

Systém můžeme rozdělit na serverovou a klientskou část. Serverová část zajišťuje výpočetní operace, komunikaci s klientem a různé další procesy nutné pro úspěšný běh aplikace. Často se také označuje jako **backend**. Klientská část běží na uživatelském zařízení v prohlížeči. Jejím úkolem je správně vykreslit aplikaci a umožnit bezproblémovou komunikaci se serverem. Jedná se o takzvaný **frontend**. Jak tyto části interagují je vidět na diagramu nasazení 4.1. Použité technologie dále představím v této kapitole.

Jednou z technologií, která se vyskytuje v serverové i klientské části je programovací jazyk **Javascript**. Někdy je také prezentován jako jazyk skriptovací a umožňuje vytvářet webové stránky s dynamickým obsahem. Je proto součástí drtivé většiny moderních aplikací spouštěných ve webovém prohlížeči. Později byl Javascript přizpůsoben i pro psaní serverového kódu a této skutečnosti využívám ve své aplikaci pro snížení počtu programovacích jazyků, které v ní figurují. To vede ke zvýšení přehlednosti a udržitelnosti zdrojového kódu.

4.2 Backendové technologie

Vybrané technologie zajišťují běh aplikace, umožňují komunikaci klientů se serverem a provádějí zpracování dat. Vybrány byly s ohledem na rozsah aplikace, komerční dostupnost a schopnost splnit vymezené požadavky.



Obrázek 4.1: Diagram nasazení systému

4.2.1 Prostředí Node.js

Node.js je prostředí vycházející z Chrome V8 Javascript engine a umožňuje spouštět kód v jazyce Javascript mimo prohlížeč. Node.js používá událostmi řízený, neblokující I/O model, který zajišťuje vysokou efektivitu. Node.js má vlastní systém pro správu knihoven **npm**, který je jedním z největším ekosystémů s volně dostupnými knihovnami na světě. [26]

Toto prostředí se pro provoz serveru osvědčilo již v aplikaci pro simulaci monopolistické konkurence [23]. Popularita Node.js od té doby stále roste, proto již dnes nabízí velmi pestrou škálu volně dostupných knihoven. Použití takových knihoven snižuje rozsah kódu, který je nutné implementovat a v případě volby kvalitních knihoven také zvyšuje spolehlivost. Pro implementaci klientské části je Javascript nutnou technologií, má proto smysl, aby serverová implementace byla ve stejném programovacím jazyce (ačkoliv to není nutné). Navíc bude zajištěna zpětná kompatibilita s výše zmíněnou simulací monopolistické konkurence. Následující odstavce jsou přeloženy přímo z webových stránek projektu Node.js. [27]

Node je navržený k vývoji škálovatelných síťových aplikací. Umožňuje paralelní zpracování velkého množství připojení. Takřka žádná funkce v Node neprovádí I/O operace přímo, nehrozí tedy zablokování procesu. Designově se podobá systémům jako stavový automat v jazyce Ruby nebo Twisted v jazyce Python. Node ale model událostí dále prohlubuje, prezentuje smyčku událostí jako konstrukci runtime prostředí namísto toho, aby byla implementována pomocí externí knihovny. Implementace smyčky událostí je velmi podobná chování Javascriptu v prohlížeči - je uživateli skryta.

Node je navržený s nízkou odezvou a přenosem dat na paměti a je tedy samozřejmostí, že protokol HTTP je v něm občanem první třídy. Node tak tvoří perfektní základ jakékoliv webové knihovny nebo rozhraní. Výhody vícejádrového programování je v rámci Node možné využít s pomocí „fork()“ rozhraní. To je navrženo tak, aby práce s ním byla co nejsnazší. Na stejném principu je postavený cluster modul, který umožňuje vyvažování zátěže na jádrech. I v roce 2019 se Node.js stále aktivně vyvíjí, v nejbližší době tedy nehrozí zastarání technologie. Díky své rychlosti, flexibilitě a rozšířenosti se jedná o perfektní technologii pro vývoj mojí aplikace.

4.2.2 Framework Express

Pro Node.js bylo vytvořeno velké množství volně dostupných technologií. Pro vývoj mojí aplikace jsem zvolil framework Express 4 [28], který má nejen bohatou historii vývoje, ale zároveň z něj vzešli další velmi populární řešení jako jsou Sails, Kraken nebo KeystoneJS.

Framework Express poskytuje řadu integrovaných komponent pro vývoj webových aplikací a jeho velkou předností je rychlost. Express přidává pouze vrstvu těch nejnutnějších webových funkcionalit, aniž by přepracovával nebo skrýval původní funkcionality prostředí. Díky tomu lze plně využít potenciálu prostředí Node.js, na kterém je vystavěn. Express je také nezaujatou technologií, což znamená, že není vystavěn pouze na jednom architektonickém stylu, ale přizpůsobí se potřebám vývoje.

4.2.3 Knihovna Socket.io

Socket.io je síťová technologie v jazyce javascript, která umožňuje obousměrnou komunikaci mezi klientem a serverem v reálném čase. Aby toto efektivně umožnila, skládá se z klientské a serverové knihovny. Komunikace je řízena událostmi, přičemž obě strany mohou události vyvolávat i zpracovávat. Každá taková událost ve své podstatě funguje jako datová zpráva a lze reagovat čistě na událost samotnou nebo zpracovat i data, která jsou k ní připojena. [29]

Pro vytvoření spojení využívá Socket.io síťových socketů a komunikačního protokolu WebSocket, odtud také pochází název technologie. Jedná se o přímou komunikační linku s minimální ztrátovostí a odezvou. Typicky je tato technologie využívána pro hraní online her nebo v rámci tzv. „chatovacích" programů. V rámci mojí aplikace taktéž probíhá simulační hra, která vyžaduje vysokou úroveň interakce mezi jednotlivými hráči. Uživatelé se potřebují v reálném čase dozvědět, jaký je stav hry, jestli jsou zrovna na tahu, čekají na soupeře nebo například jestli nebyla hra již ukončena.

Využití Socket.io má i jednu další nečekanou výhodu. Každé okno prohlížeče otevírá unikátní spojení se serverem a funguje tedy jako nezávislý hostitel nebo hráč. Proto v případě nedostatku fyzických hráčů je možné simulovat dalšího hráče jednoduchým otevřením nového okna v prohlížeči. To může být velmi efektivním nástrojem v rámci demonstrací principů při výuce a dobře se osvědčilo také při testování aplikace.

4.2.4 Technologie pro zajištění kvality

V rámci každé aplikace je důležité zajistit, aby byla vytvořena pomocí kvalitního otestovaného kódu. Kód by měl být zdokumentovaný a aplikace otevřena případným budoucím rozšířením. Nyní popíši některé z technologií, které jsem využil k dosažení těchto cílů.

4.2.4.1 Typescript

Jakožto programovací jazyk má Javascript velké množství nevýhod. Některé nedostatky jsou postupně opravovány s novými verzemi, ale přesto není čistý Javascript vhodný pro vývoj větších aplikací. Proto jsem se rozhodl v mojí aplikaci využít volně dostupnou jazykovou nadstavbu, která se nazývá **Typescript**. Tato nadstavba umožňuje například vytvářet plnohodnotné třídy a rozhraní včetně dědičnosti a generických datových typů. To umožňuje dodržovat principy objektově orientovaného programování. Typescript také implementuje staticky typované proměnné v kontrastu s čistým Javascriptem a dynamicky typovanými proměnnými. To vše slouží k značnému zvýšení přehlednosti a snížení množství chyb ve zdrojovém kódu.

Důležité je také uvést, že kód napsaný v jazyce Typescript je následně zkompilován zpět do plnohodnotného Javascriptu. Aplikace psané v Typescriptu a Javascriptu jsou proto plně kompatibilní a Typescript slouží pouze jako pomůcka pro značné zvýšení efektivity vývoje aplikace.

4.2.4.2 TSLint

Při psaní zdrojového kódu je vhodné dodržovat jisté zásady, aby další osoba, která s ním přijde do styku, mohla provádět rozšíření a údržbu kódu efektivně. Každý vývojář má však osobitý styl, a proto je dobré dodržovat určitě společné standardy. Pro Typescript je naštěstí k dispozici knihovna s názvem **TSLint**. Jedná se o nástroj, který analyzuje zdrojový kód a vynucuje při jeho psaní dodržování zadaných pravidel. Nástroj je vysoce konfigurovatelný a je možné dopředu určit, jak striktní budou tato pravidla. Mimo to TSLint také kontroluje překlepy a funkční chyby.

4.2.4.3 Testovací framework Jest

Aplikace by také měla být řádně otestována. V mé aplikaci se jedná zejména o testování výpočetních modulů, které kalkulují výrobní množství různých strategií představených v předcházejících kapitolách. Jedná se často o složité výrazy a rovnice, kontrola je tedy opravdu nezbytná. K tomuto účelu existuje v prostředí Node.js velké množství testovacích knihoven. Pro svou aplikaci jsem zvolil framework **Jest**, který využívá například společnost Facebook.

Knihovna nebyla zvolena jen pro svou velkou popularitu. Jest je vyvíjen nativně pro prostředí Node, ale zároveň podporuje další technologie použité v mé aplikaci jako například Typescript nebo React, který bude dále zmíněný v klientské části aplikace. Jedná se také o knihovnu zaměřující se na jednoduchost a efektivitu.

4.3 Frontendové technologie

Technologie v klientské části aplikace mají za úkol prezentovat data uživateli. Definuje se zde jejich struktura a grafický design aplikace. Musí být samozřejmě také zajištěna efektivní komunikace se serverem. Technologie vybrané pro tyto účely nyní představím.

4.3.1 Standardní webové technologie

Nejprve představím trojici technologií, které jsou dnes součástí naprosté většiny webových aplikací. Jedná se o stavební kameny moderního webu.

4.3.1.1 Hypertext Markup Language (HTML)

HTML je značkovací jazyk, který webovým stránkám dává základní strukturu. Text je vložen mezi různé značky (anglicky tag), které udávají jak se má text zobrazit. Tyto značky jsou následně interpretovány webovým prohlížečem a každá z nich má na výsledný text jiný vliv. Důležitá je také přítomnost tzv. hypertextových odkazů, které umožňují uživateli navigaci v rámci webové stránky nebo přesun na úplně jinou stránku.

4.3.1.2 Cascading Style Sheets (CSS)

Ve svém počátku sloužilo HTML i pro vytváření vzhledu webové stránky, značkám je možné specifikovat například velikost a barvu textu, vzhled pozadí a další parametry. Postupem času se tento postup ukázal jako nedostačující a byly vytvořeny kaskádové styly, které existují nezávisle na značkovacím jazyce. Jednotlivým značkám v rámci značkovacího jazyka přidělují styl. Kaskádový se stylu říká, protože všechny značky zanořené do té původní (její potomci), podědí styl této značky (svého rodiče). Samozřejmě může ale být takovým potomkům přidělen i styl vlastní. V případě konfliktu je pak zvolen styl, který je nejvíce zanořený (nejblíže vnitřnímu textu).

Při vývoji webu se jedná o velmi silný nástroj, který mimo jiné umožňuje měnit zobrazení na základě velikosti obrazovky, čehož využívám pro zajištění responzivního zobrazení na mobilních zařízeních. Styly je také možné dynamicky měnit, toho je v mé aplikaci využito například pro implementaci dvojího barevného schématu. Prvním je světlý vzhled, který je vhodnější pro zobrazení na projektoru. Druhým je pak tmavý vzhled, který je vizuálně atraktivní. Uživatel mezi vzhledy může volně přepínat.

4.3.1.3 Javascript

Výše uvedený jazyk HTML umožňuje v kombinaci s kaskádovými styly vytvářet velmi dobře vypadající webové stránky. Ty mohou však zobrazovat pouze statický obsah, což už v dnešní době není dostačující. Abychom mohli uživatelům nabídnout obsah dynamický, je potřeba přidat ještě třetí vrstvu a využít jednu z již dříve představených technologií - **Javascript**.

Jedná se o objektově orientovaný skriptovací jazyk, který je mimo jiné možné využít k dynamické manipulaci obsahu webových stránek. Tu zajišťují skripty psané v tomto jazyce. Můžeme pomocí nich například vytvářet animace, překreslovat grafické komponenty bez nutnosti obnovení stránky nebo získávat měnící se data od jiných uživatelů a ze serveru. Javascript nám umožňuje reagovat na podněty uživatele jako je například kliknutí myší nebo stisknutí tlačítka na klávesnici. Oproti statickým stránkám je také manipulace s obsahem efektivnější a plynulejší, neboť je překreslován pouze obsah, který se změnil. V případě statického modelu by bylo nutné znovu načíst celou stránku, aby se mohla projevit jakákoliv změna.

4.3.2 Bootstrap

Vysvětlil jsem již důležitost a potenciál kaskádových stylů, vytvořit s jejich pomocí ucelený, dobře vypadající vzhled však není jednoduchý úkol. Za tímto účelem vzniklo množství komplexních řešení, která zajišťují konzistentní vzhled jednotlivých komponent nejen v rámci jedné webové stránky, ale dost často také napříč různými zařízeními. Jedním takový řešením je frontendový framework **Bootstrap** [30]. Pro tuto aplikaci jsem jej vybral zejména pro svou jednoduchost a kompatibilitu s ostatními použitými technologiemi. Jedná se o řešení, které je „mobile-first“, což znamená, že nám zaručuje, že vytvořený design bude dobře fungovat zejména na mobilních zařízeních. Vzhledem k tomu, že responzivita je jedním z hlavních požadavků na aplikaci, jedná se o velmi důležitou vlastnost.

Bootstrap byl původně vytvořen pro interní potřeby sociální sítě Twitter v roce 2010 [31]. Volně dostupným se však stal již 19. srpna 2011 a od té doby prochází kontinuálním rozvojem. Například v roce 2013 vyšla verze Bootstrap 3, která implementuje již zmiňovaný „mobile-first“ přístup. V mé aplikaci používám aktuální verzi Bootstrap 4, která je dostupná od ledna 2018. Nyní již probíhají práce na páté hlavní verzi, která má například za cíl nahradit knihovnu jQuery nativními funkcemi v jazyce Javascript. Na rozdíl od různých experimentálních řešení tedy nehrozí, že by v nejbližší době Bootstrap zastaral a věřím, že je cennou součástí mé aplikace.

4.3.3 React

Stejně jako není v případě serverové části vhodné začínat psát aplikaci od nuly, podobně je tomu v případě klientské části. Je dobré vybrat již existující sadu nástrojů, která vývoj zpřehlední, urychlí a obecně umožní lepší práci se zdrojovým kódem. V dnešní době množství dostupných nástrojů stále rapidně roste a učinit správný výběr není jednoduché. V mé aplikaci jsem se rozhodl použít knihovnu **React**, která je v roce 2019 vůbec nejrozšířenějším řešením pro frontendový vývoj v jazyce Javascript. Původně byl React vytvořen společností Facebook, ale již v roce 2013 se stal volně dostupným pro veřejnost a od té doby jeho popularita strmě stoupá. K vývoji jej využívají společnosti jako například Netflix, Airbnb nebo PayPal.

React je velmi revoluční technologií a to dokonce tolik, že se původně po svém vydání dočkal velkého nepochopení a výsměchu. React totiž fundamentálně mění způsob, jakým vývojář a aplikace pracují se strukturou nazývanou **Document Object Model** (dále jen DOM). Abych mohl tuto odlišnost vysvětlit, je nejdříve třeba definovat, o co se vlastně jedná. DOM nám umožňuje manipulovat existující prvky HTML dokumentu, typicky pomocí skriptů psaných v jazyce Javascript. Ve své podstatě jsou jednotlivé prvky uspořádány do virtuální stromové struktury, kterou je následně možné libovolně procházet a modifikovat. Vytváříme tak objektově orientovanou reprezentaci dokumentu původně vytvořeného pomocí značkovacího jazyka. DOM je obecně velmi komplexní téma a výše uvedená definice je pouze hrubým nástinem jeho funkčnosti.

React se od dřívějších řešení liší v tom, že pro vytváření dynamického webu není přímá manipulace DOM nutná. Dříve by bylo nutné definovat statickou stránku a následně bychom dynamicky měnili její obsah manipulací struktury DOM tak, jak by přicházela data. V Reactu místo toho popíšeme, jak by měla vypadat výsledná stránka na základě aktuálních a budoucích dat. React z tohoto popisu následně vytvoří virtuální DOM a s jeho pomocí poté provede aktualizaci skutečného DOM webové stránky. K tomu využije efektivní interní algoritmy a vše tak proběhne přehledněji a efektivněji.

4.3.3.1 JSX

Mimo výše uvedené výhody jsou jádrem Reactu komponenty. Každá komponenta funguje jako samostatný modul s interní logikou a produkuje nějaký výstup. Výhodou je, že můžeme vytvořit komponentu se specifickým účelem a následně jí znovu využít na různých místech v aplikaci. To je princip, který je v souladu s objektově orientovaným programováním. Spojením mnoha takových komponent pak vytvoříme komplexní webovou stránku.

Psát takové komponenty v čistém Javascriptu je poměrně zdlouhavé a nepraktické, React proto obsahuje rozšíření Javascriptu, které se nazývá **JSX** (JavaScript XML). Jedná se o nadstavbu Javascriptu, která se vizuálně a logicky podobá HTML a XML. React za nás pak už zajistí překlad kódu do čistého Javascriptu (pomocí knihovny Babel). To nám umožňuje snazší, rychlejší a preciznější vývoj webu.

4.3.3.2 Knihovna React Bootstrap

Aby mohl Bootstrap plně využít prostředí knihovny React, bylo by ideální jednotlivé komponenty přepsat do JSX. K tomu dnes již naštěstí také existují volně dostupné knihovny. Já jsem pro tento účel vybral knihovnu **React Bootstrap** [32]. Její výhodou je, že plně nahrazuje původní Bootstrap a vytváří úplně nové React komponenty. Nevzniká tak žádná závislost na původním Javascriptovém kódu nebo knihovně jQuery.

4.3.3.3 Lokalizace

V době globalizace pomůže k rozšíření aplikace, bude-li dostupná ve větším počtu jazyků a nikoliv pouze v jazyce českém. Zároveň se jedná o jeden z dříve zmíněných požadavků. Proto je v mojí aplikaci lokalizace zajištěna pomocí knihovny **react-intl**, která je součástí projektu FormatJS [33]. Za projektem stojí tým ze společnosti Yahoo a knihovna dodržuje globální standardy pro lokalizaci aplikací. Každá část jazykově závislého textu je speciálně označena pomocí komponent této knihovny. Texty jsou následně extrahovány za pomoci knihovny Babel a uloženy do speciálního souboru. Z tohoto souboru je následně možné provést překlad textů do libovolného světového jazyka. V základní verzi bude má aplikace dostupná v českém a anglickém jazyce.

4.3.3.4 Ostatní balíčky

Na závěr této kapitoly bych chtěl uvést některé další knihovny, které aplikace využívá. Jedná se spíše o vedlejší pomůcky při vývoji, přesto jsou ale nedílnou součástí aplikace. Patří mezi ně například dříve zmiňovaná knihovna **Jest**, která slouží k testování aplikace. Pro navigaci je v aplikaci použita knihovna **React Router DOM**, která umožňuje plynulý přechod mezi statickým a herním modulem. Klíčovou komponentou je také klientská část knihovny **Socket.io**, která zajišťuje komunikaci se serverem pomocí protokolu WebSocket a byla detailněji popsána v kapitole 4.2.3.

Kapitola 5

Implementace

V předchozích kapitolách jsem specifikoval požadavky kladené na aplikaci a technologie zvolené k jejich řešení. Zbývá tedy popsat samotnou implementaci aplikace vzniklé v rámci této práce. Kapitola je opět rozdělena na serverovou část, která implementuje framework Express a klientskou část, v níž je implementován framework React.

5.1 Serverová část systému

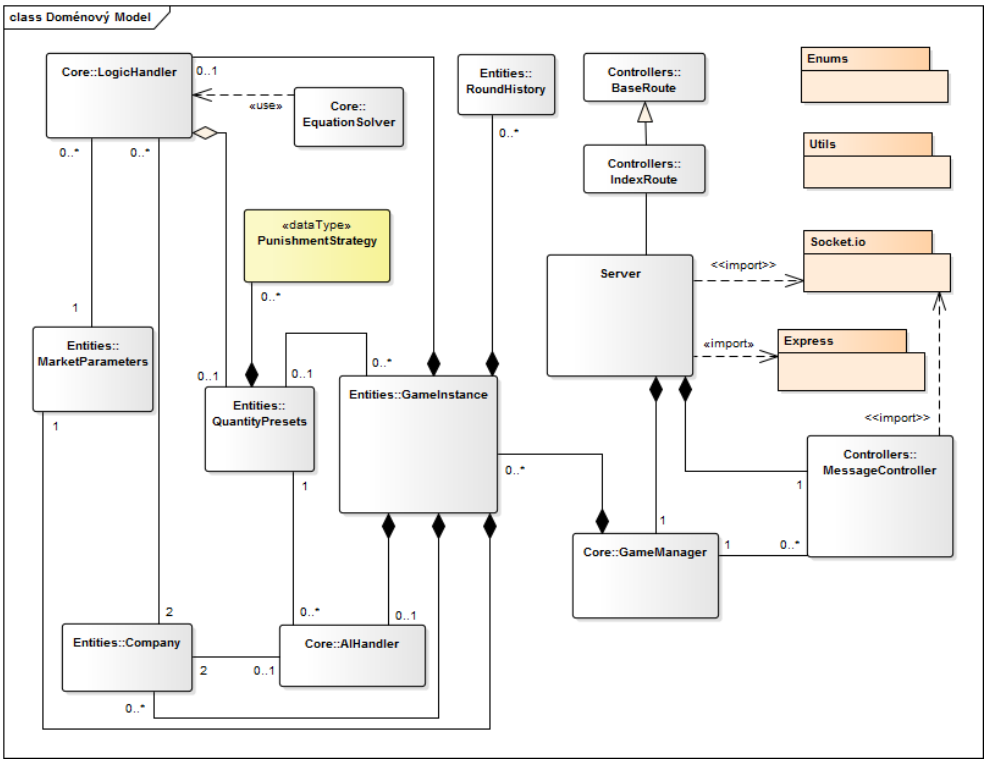
Pro vymezení serverové části systému jsem zvolil doménový model, který popisuje objekty implementované v rámci systému a vztahy mezi nimi. Ty je možné vidět na doménovém diagramu 5.1. Jednotlivé vyobrazené objekty nyní detailněji popíši.

5.1.1 Objekt Server

Centrálním objektem mojí aplikace je třída Server. Instance této třídy je vytvořena při spuštění Node.js serveru a existuje až do jeho ukončení. V rámci této třídy jsou implementovány knihovny Express a Socket.io. Obsahuje také reference na objekty GameManager a MessageController, které spravuje.

5.1.2 Objekt GameManager

Cílem práce je simulovat oligopolistickou konkurenci v režimu pro jednoho nebo dva hráče. Správu těchto her má na starosti třída GameManager neboli Správce her. Na serveru existuje právě jedna instance této třídy, která je spjata s jeho existencí. Třída udržuje dvě nezávislá pole objektů GameInstance, jedno pole pro hry jednoho hráče a druhé pro hry více hráčů. Dále zajišťuje, aby každá instance hry měla svůj unikátní identifikátor a manipulovat se hrou je možné pouze prostřednictvím správce. Mezi manipulační operace patří například vytvoření nové hry, získání herního objektu nebo zrušení existující hry.



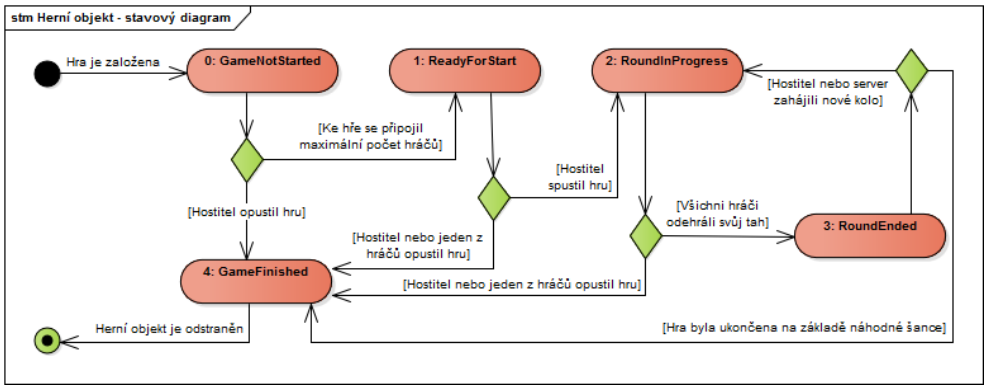
Obrázek 5.1: Doménový model serverové části aplikace

5.1.3 Objekt MessageController

Manipulace s herními objekty probíhá na základě komunikace s uživatelem. Tuto komunikaci zajišťuje třída MessageController. V rámci této třídy je využita serverová část knihovny Socket.io a komunikace probíhá pomocí protokolu WebSocket. Třída nese odpovědnost jak za přijímání, tak za odesílání zpráv. Detailní seznam komunikačních událostí je uveden na závěr této kapitoly.

5.1.4 Objekt GameInstance

Srdcem serverové části mé aplikace je třída GameInstance, která reprezentuje simulační hru. Každá instance této třídy je unikátní a Správce her na ní udržuje odkaz, dokud je k ní připojený její hostitel. Třída obsahuje odkazy na jednotlivé firmy, které v simulaci vystupují a na tržní parametry nastavené uživatelem při založení hry. Dále obsahuje odkaz na výpočetní modul, který na základě tržních parametrů vypočítá produkční kvantity pro různé rovnovážné stavy uvedené v teoretické části této práce. Mezi ně patří například Maximalizace tržního zisku, Cournotova rovnováha nebo Minimaxové trestání.



Obrázek 5.2: Stavový diagram herního objektu

Pokud se jedná o hru pro jednoho hráče, udržuje také odkaz na modul umělé inteligence, který na základě zvolené strategie určí vyráběné množství v daném kole. Nahradí tak roli jednoho z hráčů. Posledním navázaným objektem je třída QuantityPresets, která obsahuje množství získaná v rámci modulu výpočetního. Typicky je tyto hodnoty potřeba vypočítat pouze jednou na začátku hry a po zbytek hry je možné využívat hodnoty uložené v rámci této instance. To vede k šetření výpočetních prostředků a času na straně uživatele.

Herní objekt také spravuje svůj vnitřní stav. Po založení hry uživatelem probíhá tzv. herní smyčka, v rámci které objekt prochází různými stavy, dokud není ze serveru odstraněn Správcem her. Tento proces je přehledně znázorněný ve stavovém diagramu 5.2.

5.1.5 Objekt Company

Instance tohoto objektu reprezentuje jednu tržní firmu v rámci oligopolu. Na základě požadavků na mou aplikace se vyskytují firmy vždy v párech, neboť se jedná o duopol. Třída obsahuje informace jako název společnosti, identifikátor hráče, který společnost ovládá, jeho jméno a nejdůležitějším atributem jsou parametry cenové křivky, na základě které se poté vypočítává zisk. Instance třídy také obsahuje informaci o produkci a zisku v daném kole a za celou hru. Tyto údaje se v průběhu jednotlivých herních kol přepočítávají.

5.1.6 Další podpůrné objekty

Výše uvedené objekty mají v systému klíčové role, v této sekci poté uvádím objekty, které plní podpůrnou roli těmto klíčovým třídám a dohromady tak tvoří výsledný systém.

5.1.7 Objekty zajišťující logické a výpočetní operace

Pro zajištění výpočetních operací existuje v aplikaci třída **LogicHandler**. Ta zajišťuje dříve zmiňované výpočty rovnovážných stavů, ale má také na starosti například výpočet tržní ceny na základě zvolených produkčních výstupů v daném kole nebo výpočet tržního zisku jednotlivých firem. Pro některé složitější výpočty pak využívá statické třídy EquationSolver, která se specializuje na řešení parametrických rovnic. Třída také udržuje odkaz na objekt **QuantityPresets**, do kterého ukládá výsledky výpočtů. Je-li pak požadována operace, pro kterou jsou hodnoty již vypočítány, LogicHandler tyto hodnoty použije a výpočet zbytečně neopakuje.

Ve hře pro jednoho hráče je využita třída **AIHandler**, která implementuje zvolenou strategii počítačového hráče a udržuje informace o kooperativním a nekooperativním výstupu firmy, kterou počítač ovládá. Na základě této strategie, volitelných parametrů a historie předchozích kol pak určí výstup počítačem ovládané firmy v daném kole.

5.1.8 Ostatní objekty

Na diagramu 4.1 jsou vyobrazeny balíčky **Enums** a **Utils**. V rámci balíčku Enums jsou uloženy různé číselníky, které aplikace využívá. Balíček Utils pak obsahuje pomocné funkce různého charakteru. Detailní popis těchto balíčků, stejně jako výše představených tříd je možné nalézt v dokumentaci ke zdrojovému kódu mé aplikace.

5.2 Klientská část systému

V následující části stručně popíši strukturu aplikace na klientské straně. Dále uvedu, jaké možnosti se nabízí uživateli, který aplikaci bude používat.

5.2.1 Struktura klientské části

Tato část systému je implementována pomocí technologie React. Protože se jedná o vizuální komponenty, nemá zde smysl použít doménový model jako v předchozí části. Místo toho zde popíši klíčové komponenty a celkové chování klientské části systému.

5.2.1.1 Přepínání vzhledu a jazyků

V záhlaví aplikace jsou uživateli k dispozici ovládací prvky. První z nich umožňuje přepínat mezi světlou a tmavou verzí aplikace. Druhé menu nabízí výběr jazyka, ve kterém se aplikace zobrazí. V úvodní verzi jsou k dispozici česká a anglická varianta.

5.2.1.2 Navigace

Navigace je implementována pomocí knihovny **React Router DOM** a umožňuje navigaci použitím lišty s odkazy v horní části obrazovky. Alternativně je možné přejít přímo na následující odkazy v rámci domény:

```
\
\theory\
\game\
```

Při použití jiného URI je uživateli zobrazena stránka s chybovým hlášením.

5.2.1.3 Klíčové komponenty

Centrální komponentou této části systému je třída **App**. Ta má za úkol udržovat informaci o aktuálně zvoleném vzhledu a jazyce aplikace a dále řídit přesměrování do jednotlivých sekcí aplikace. Komponenta **Header** obsahuje ovládací prvky, které tyto změny a navigaci usnadňují uživateli.

Komponenty **About** (URI \) a **Theory** (URI \theory\) jsou součástí statického modulu a vyřizují zobrazení informačních textů sloužících jako úvod do aplikace. Komponenta **Game** (URI \game\) poté obsluhuje celý herní modul a zajišťuje bezchybné zobrazení simulační hry.

Komponenta Game se dále skládá z množství dalších komponent, které implementují jednotlivé funkcionality. Jejich podrobný popis je k dispozici v dokumentaci ke zdrojovému kódu.

5.2.2 Realizace statického modulu

Tento modul se skládá ze dvou webových stránek (vytvořených pomocí komponent uvedených výše). Při spuštění aplikace se uživateli zobrazí úvodní stránka, kterou tvoří komponenta About. Na této stránce jsou základní informace o aplikaci, odkaz na tuto práci a kontakt na autora. Druhou statickou stránku tvoří komponenta Theory. Jak název vypovídá, na této stránce uživatel nalezne stručný úvod do teorie k aplikaci a popis jejího ovládání. Obě tyto stránky obsahují pouze statický obsah, odtud název tohoto modulu a z obou je možné vstoupit do dynamického herního modulu.

5.2.3 Realizace herního modulu

Po připojení do herního modulu se uživatel nalezne v přípravné fázi a zobrazí se mu seznam dostupných her pro více hráčů.

5.2.3.1 Přípravná fáze

Tato fáze herního modulu slouží k zakládání nových her a připojování k existujícím hrám. Skládá se ze tří záložek. Na první je k dispozici seznam existujících her a tlačítka pro obnovení seznamu a připojení ke hře.

Na druhé záložce lze vytvořit hru pro více hráčů. Uživatel musí vyplnit své jméno, název hry a množství herních parametrů vycházejících z teoretické části této diplomové práce. Pro pohodlí uživatele je také možné vybrat jeden z předem připravených scénářů a vyhnout se tak nastavování vlastních tržních parametrů. Po vyplnění všech údajů je potřeba vytvoření hry potvrdit stiskem tlačítka.

Třetí záložka slouží k vytvoření hry pro jednoho hráče a obsahuje stejný formulář jako druhá záložka. Navíc je ale od uživatele požadováno nastavit parametry počítačového protivníka. Zde je možné zvolit herní strategii a formu nekooperativního výstupu. Aplikace také umožňuje náhodný výběr těchto parametrů. Vytvoření hry je opět potřeba potvrdit stiskem tlačítka.

5.2.3.2 Simulační fáze

Uživateli se tato fáze zobrazí pouze po úspěšném založení hry nebo po připojení k existující hře. Stránka obsahuje velké množství informací, které se dá rozdělit do čtyř základních sekcí. V záhlaví stránky se nachází informace o tržních parametrech a zprávy přijaté ze serveru. Na levé a pravé straně obrazovky se nachází informace o firmách jako produkovaný výstup, nákladová funkce a dosažený zisk. Na levé straně jsou informace o firmě prvního hráče, na pravé potom hráče druhého.

Centrální komponentou v této fázi jsou informace o nekalé spolupráci zobrazené uprostřed obrazovky. Zobrazují například, jestli je kartel aktivní nebo jaká množství je třeba vyrábět pro obnovení dohody. Nabízí hráči také vypočítaná výrobní množství odpovídající různým rovnovážným stavům, aby sám nemusel počítat složité rovnice. V zápatí stránky se pak dle požadavků nachází historie všech odehraných kol v rámci hry.

Mimo tyto informační prvky stránka obsahuje také prvky ovládací. Všichni hráči mají přístup k tlačítku pro opuštění hry. Hostitel pak ovládá tlačítko pro spuštění hry a pro zahájení dalšího herního kola. V případě, že je uživatel hráčem, má k dispozici tlačítko pro ukončení kola a formulář pro vyplnění množství, které chce v daném kole produkovat. Aby bylo více zdůrazněno, kterou firmu vlastně ovládá, tyto prvky se zobrazí na straně obrazovky, která odpovídá jeho firmě.

5.3 Komunikace mezi serverem a klientem

Klientská a serverová část spolu komunikují pomocí množství událostí vyvolaných v rámci knihovny Socket.io. Protože se jedná o zcela klíčový mechanismus pro fungování aplikace, jednotlivé typy zpráv představím v dalším textu.

5.3.1 Společné události

Knihovna socket.io definuje několik druhů událostí, které vznikají automaticky, je možné je zpracovat na klientské i serverové straně, ale systém by je neměl sám generovat.

Událost „connection“

Jedná se o událost, která je vyvolána při úspěšném vytvoření spojení mezi klientskou a serverovou částí aplikace.

Událost „disconnect“

Opakem předchozí události je zpráva oznamující odpojení uživatele. Server na tuto zprávu reaguje ukončením všech her, ve kterých uživatel figuroval.

5.3.2 Události, které přijímá server od klienta

Tato sada událostí je generována na klientské straně, většinou jako důsledek stisku tlačítka. Server tyto události zpracovává a reaguje na ně generováním odpovídající události.

Událost „create-game“

Tato zpráva serveru říká, že má vytvořit novou herní instanci. Obsahuje data potřebná k jejímu vytvoření a hostitelem hry se stává socket, který událost vyvolal.

Událost „list-games“

Jedná se o událost vyvolanou stiskem tlačítko pro obnovení seznamu her. Zpráva serveru říká, že má patřičnému socketu předat aktuální seznam her pro více hráčů.

Událost „game-join“

Požadavek oznamující serveru, že se klient chce připojit ke hře, která je ve zprávě specifikována. Server uživatele ke hře připojí, je-li to možné a odešle zprávu „game-update“ s informacemi pro vykreslení herní plochy. Pokud se ke hře z nějakého důvodu připojit nelze, vyvolá namísto toho událost „error-message“.

Událost „game-start“

Událost vyvolaná hostitel hry, který po serveru požaduje, aby hru zahájil. V případě, že tento požadavek není oprávněný, server odešle zprávu „error-message“. V opačném případě provede všechny kroky nezbytné k zahájení hry a informování přítomných uživatelů o začátku této hry.

Událost „turn-end“

Zde se jedná o událost, kterou vyvolá uživatel stiskem tlačítka pro ukončení tahu. Součástí zprávy je množství, které chce v daném tahu vyrobit. Server musí ověřit, že se jedná o oprávněnou akci, uložit vyráběné množství a předat kontrolu dalšímu hráči na tahu. Pokud se jednalo o posledního hráče v daném kole, server kolo ukončí a rozešle odpovídající aktualizace připojeným uživatelům.

Událost „round-start“

V případě, že uživatel při vytvoření hry vybere automatický start nového kola, tato událost nebude vůbec využita. Pokud však zvolí manuální start, server po skončení kola vyvolá událost „round-start-enable“, která hostiteli odemkne přístup k tlačítku pro zahájení dalšího kola. Stiskem tohoto tlačítka je pak vyvolána tato událost, která zahájí nové herní kolo. Pokud uživatel nemá oprávnění nové herní kolo zahájit, server opět vyvolá událost „error-message“.

Událost „leave-game“

Poslední událostí, kterou server zpracovává, je žádost o opuštění aktuální hry. Server hráče odejme z aktivní hry, pokud v nějaké figuruje a ostatním hráčům hru ukončí. Také zajistí, aby se uživatel, který událost vyvolal, vrátil zpět do přípravné fáze herního modulu.

5.3.3 Události, které přijímá klient ze serveru

Jedná se o zprávy získané ze serveru za různých okolností. Některé události vznikají jako odpovědi na zprávy odesílané klientskou částí systému. Jiné jsou generovány jako důsledek procesů, které na serveru probíhají.

Událost „game-update“

Tato událost klientské straně systému říká, že pokud se nachází v přípravné fázi herního modulu, je potřeba se přepnout do simulační fáze. Pokud se v této fázi již nachází, stačí pouze aktualizovat herní plochu. Informace pro vykreslení plochy jsou součástí zprávy.

Událost „refresh“

Událost přikazuje klientské části systému obnovit seznam her v přípravné fázi herního modulu. Událost je buď vyvolána automaticky, když dojde ke změnám na serveru nebo je odeslána jako odpověď na událost „list-games“, kterou uživatel vyvolal stiskem tlačítka.

Událost „update-info“

Aby bylo zajištěno, že uživatel vnímá aplikaci jako interaktivní, server průběžně posílá zprávy obsahující informace o stavu hry. K tomuto účelu slouží právě událost „update-info“. Událost je klientem zpracována pouze nachází-li se v simulační fázi herního modulu.

Událost „game-start-enable“

Základní událost, která umožňuje uživateli zahájit hru. Je serverem vyvolána okamžitě po naplnění hry plným počtem hráčů. Na základě jejího přijetí je na herní desce odemčeno tlačítko pro zahájení hry.

Událost „turn-end-enable“

Podobně jako v případě předchozí události, umožňuje hráči, který je na tahu, svůj tah ukončit.

Událost „round-start-enable“

Funguje na stejném principu jako událost „game-start-enable“ s tím rozdílem, že se jedná o zahájení nového kola a nikoliv celé hry. V případě, že hostitel při založení hry zvolil automatický start dalšího kola, tato událost není nikdy spuštěna.

Událost „turn-timer-enable“

V případě, že hostitel zvolil jako metodu začátku nového kola automatický časovač, tato událost zajišťuje vykreslení odpočtu na herní ploše. Server tuto událost vyvolá okamžitě po skončení kola, pokud hra nebyla dosud ukončena. V případě, že hostitel zvolil manuální start nového kola, tato událost není nikdy spuštěna.

Událost „return-to-list“

Tuto událost server vyvolá jako důsledek stisku tlačítka pro opuštění hry na straně klienta. Má za následek, že klient opustí simulační fázi herního modulu a přesune se zpět do přípravné fáze.

Událost „game-end“

Událost, kterou klient obdrží ve chvíli, kdy je na serveru zrušena hra, ve které se nachází. Jedinou reakci, kterou může klient provést, je opustit hru tlačítkem nebo zavřením záložky v prohlížeči.

Událost „error-message“

Poslední zprávou, kterou klientská část systému zpracovává je zpráva chybová. Tuto událost server vyvolá pokaždé, když detekuje nějakou chybu způsobenou vstupem uživatele.

Kapitola 6

Testování

Každá kvalitní aplikace by měla být řádně otestována před nasazením do provozu. V této kapitole popíši, jakým způsobem byla testována má aplikace.

6.1 Testování bílé skříňky

U tohoto druhu testování je plně známá vnitřní struktura systému. Tím je myšleno složení tříd, vazby mezi nimi a další implementační detaily. Cílem testování v této fázi je odhalit chyby ve zdrojovém kódu jednotlivých komponent, ale také otestovat jak spolu komponenty interagují a jak se chovají v mezních situacích.

6.1.1 Jednotkové testy

Již dříve byla zmíněna knihovna **Jest**. Ta je použita jak v serverové, tak v klientské části pro implementaci jednotkových testů (anglicky unit tests). Z výše uvedených scénářů se jednotkové testy hodí nejlépe k otestování izolovaného chování metod v rámci tříd nebo samostatně stojících funkcí.

Aktuálně jsou využity primárně k testování třídy **LogicHandler**. Třída má na starosti výpočet složitých parametrických rovnic a na její přesnosti závisí bezchybná funkčnost mé aplikace. Jako vstupní data pro tyto testy jsou využity scénáře tržních podmínek z literárních zdrojů použitých v rámci této práce. Zde se můžeme spolehnout na kvalitní vstupní data a správné výsledky výpočtů, které pak lze využít k ověření testů.

Do budoucna jsou v aplikaci další segmenty, které by bylo vhodné pokrýt jednotkovými testy, jedná se zejména o třídu **AIHandler**, která implementuje strategie pro umělou inteligenci. Bez ohledu na komplexnost strategie je jediným výstupem produkované množství, metody třídy jsou tedy perfektními kandidáty pro jednotkové testování. V klientské části se pak jedná o testy, které ověřují správné vykreslení komponent a vracení správných stavových kódů protokolu HTTP.

6.2 Testování černé skříňky

Druhou možností je testovat aplikaci bez znalosti vnitřní struktury systému. Jedná se o testy, které s aplikací pracují tak, jak by s ní pracoval uživatel a na základě toho se snaží odhalit chyby. Tester zde nemá přístup ke zdrojovému kódu ani k žádným funkcionalitám systému, ke kterým by neměl přístup běžný uživatel.

6.2.1 Testování případů užití

Test, který jsem se v této části rozhodl provést, je testování případů užití. Jejich správná funkčnost je totiž zásadní pro kvalitu výsledné aplikace. Postup byl otestovat všechny případy užití vymezené v rámci softwarového návrhu a zaznamenat jejich výsledky. Následuje popis jednotlivých testů a z nich vyvozené závěry.

6.2.1.1 Herní modul - přípravná fáze

Jako první jsem otestoval přípravnou fázi herního modulu v souladu s pořadím uvedeným v kapitole 3.2, kde byly případy užití vymezeny.

1. **Načíst seznam her**
   - **Popis:** Uživatel stiskem tlačítka načte aktuální seznam her.
   - **Aktéři:** Uživatel
   - **Průběh testu:** Po stisknutí tlačítka se zobrazí seznam aktuálně dostupných her. V případě, že žádné na serveru nejsou k dispozici, zobrazí se informace o této skutečnosti.
   - **Výsledek testu:** Základní tok událostí splnil všechna očekávání.
2. **Založit hru pro více hráčů**
   - **Popis:** Případ užití umožňuje uživateli stát se hostitelem hry pro dva hráče. Hra má k dispozici dvě místa pro hráče a je možné se k ní připojit ze seznamu her.

- **Aktéři:** Uživatel
- **Průběh testu:** Základní tok událostí nevykazuje žádné nedostatky. V alternativním toku událostí se zobrazí chybová hlášení u nesprávně vyplněných polí. Po opravení formuláře automaticky zmizí.
- **Výsledek testu:** Po založení hry je uživateli signalizováno, že je hostitelem například tím, že nemá zobrazena tlačítka pro ukončení kola. Dále není jeho jméno uvedeno u žádné z firem. Přesto by pro uživatele, který aplikaci vidí poprvé mohla být situace matoucí. Aplikace oznamuje připojení nových hráčů a po naplnění hry zpřístupní tlačítko pro spuštění hry. Proces je však dostatečně efektivní pro nasazení do provozu.

3. **Založit hru pro jednoho hráče**

   - **Popis:** Případ užití umožňuje uživateli zahájit simulaci proti počítačem řízené firmě. Firma operuje na základě zvolené strategie. Uživatel vystupuje jako hostitel a hráč zároveň. Hra je serverem ukončena jakmile ji uživatel opustí.
   - **Aktéři:** Uživatel
   - **Průběh testu:** Základní tok událostí byl proveden bez chyb.
   - **Výsledek testu:** V záhlaví aplikace je jasně viditelné, že se jedná o hru jednoho hráče. Toto signalizují i další prvky, jako například označení soupeře popiskem pro umělou inteligenci. Ovládací prvky jsou snadno přístupné a funkční. Vytváření hry vyžaduje širší znalost tématu a bylo by vhodné vytvořit instruktážní materiály v případě nasazení pro širší veřejnost.

4. **Připojit se ke hře pro více hráčů**

   - **Popis:** Případ užití umožňuje uživateli připojení ke hře pro více hráčů, kterou jiný uživatel hostuje. Ke hře je připojený, dokud on nebo hostitel hru neopustí.
   - **Aktéři:** Uživatel
   - **Průběh testu:** Po vypsání seznamu her je tlačítko pro připojení ke hře nepřístupné, dokud uživatel nevybere hru ze seznamu. Vybraná hra je přehledně označena v tabulce her. Tlačítko je zpřístupněno po vybrání hry. Po stisku tlačítka je vykreslena herní plocha.
   - **Výsledek testu:** Test neobjevil žádné závažné chyby. Po připojení do hry vidí uživatel přehledně, kterou firmu ovládá, neboť je u ní napsáno jeho jméno. Zároveň je tlačítko pro ukončení tahu situováno na obrazovce u firmy, kterou ovládá.

5. **Opustit herní modul**

- **Popis:** Uživatel opustí aplikaci zavřením záložky v prohlížeči, obnovením stránky nebo přechodem na jiné URL.
- **Aktéři:** Uživatel
- **Průběh testu:** Uzavření aplikace proběhlo bez chybových hlášení.
- **Výsledek testu:** Scénář nevykázal žádné nedostatky.

6. **Založit simulační hru**

   - **Popis:** Jedná se o podproces při vytváření hry pro jednoho nebo více hráčů.
   - **Aktéři:** Uživatel
   - **Průběh testu:** Funkcionalita byla dostatečně otestována v rámci vytváření hry pro více hráčů a hry pro jednoho hráče.
   - **Výsledek testu:** Při přechodu do hry je korektně vykreslena hrací plocha. Scénář nevykazuje žádné chyby.

7. **Přejít do simulační fáze**

   - **Popis:** Uživateli je na základě jeho žádosti vykreslena herní plocha.
   - **Aktéři:** Uživatel
   - **Průběh testu:** Vykreslení herní plochy bylo otestováno v rámci případů užití, které tento scénář zahrnují.
   - **Výsledek testu:** Při připojení do hry je korektně vykreslena herní plocha. Herní plocha se správně aktualizuje při změnách v rámci hry.

6.2.1.2 Herní modul - simulační fáze

Následuje testování simulační fáze, do které lze přejít vytvořením nové hry nebo připojením se ke hře již existující.

1. **Spustit hru**

   - **Popis:** Případ užití umožňuje hostiteli zahájit hru. Zahájením hry je předáno ovládání dalšímu hráči na tahu.
   - **Aktéři:** Hostitel
   - **Průběh testu:** Po stisknutí tlačítka pro spuštění hry je přehledně vypsáno oznámení o tom, který hráč je na tahu. Tlačítko „Spustit hru“ se změní na „Zahájit kolo“, což dále signalizuje hostiteli, že hra započala. Hru správně není možné spustit, pokud není naplněna její kapacita.
   - **Výsledek testu:** Scénář byl splněn bez závažných chyb.

2. **Spustit další kolo**

   - **Popis:** Případ užití umožňuje hostiteli spustit nové kolo. Zahájením nového kola je předána kontrola nad hrací plochou dalšímu hráči v pořadí (podobně jako při spuštění nové hry).
   - **Aktéři:** Hostitel
   - **Průběh testu:** Poslední hráč (v daném kole) ukončí tah a aplikace automaticky zahájí odpočet nového kola nebo předá kontrolu nad hrou hostiteli, aby spustil další kolo (podle nastavení zvoleného při vytváření hry).
   - **Výsledek testu:** Spuštění nového kola probíhá bez komplikací. V případě, že je hra ukončena, tlačítko je správně deaktivováno.

3. **Zobrazit historii zpráv**

   - **Popis:** Případ užití umožňuje uživateli zobrazit historii zpráv přijatých ze serveru včetně času, kdy byla zpráva přijata.
   - **Aktéři:** Hostitel, Hráč
   - **Průběh testu:** Po stisknutí tlačítka na herní ploše se otevře modální okno obsahující zprávy. Nové zprávy od posledního otevření jsou přehledně označeny. V případě, že doposud nebyly přijaty žádné zprávy, je zobrazena náhradní zpráva.
   - **Výsledek testu:** Scénář byl splněn bez vizuální nebo systémové chyby.

4. **Opustit hru**

   - **Popis:** Případ užití umožňuje hráči nebo hostiteli opustit aktivní hru v rámci simulační fáze herního modulu.
   - **Aktéři:** Hostitel, Hráč
   - **Průběh testu:** Hra je ukončena pokud ji hostitel nebo hráč opustí. V případě odchodu hostitele je navíc kompletně odstraněna z paměti serveru. Hráči, kteří v takové hře zůstanou ji mohou opustit stiskem tlačítka nebo pomocí navigace.
   - **Výsledek testu:** Test splnil očekávání, na serveru nevznikají paměťové úniky a uživatelům se při opouštění her neobjevují chyby.

5. **Ukončit tah**

   - **Popis:** Případ užití umožňuje hráči, který je na tahu svůj tah ukončit a předat aplikaci množství, které chce v daném kole produkovat.
   - **Aktéři:** Hráč

- **Průběh testu:** Potom, co na hráče přijde řada, zpřístupní se tlačítko pro ukončení tahu. Při vyplnění nesmyslného produkčního množství je zobrazena chybová hláška. Po úspěšném stisknutí tlačítka se tlačítko znepřístupní a kontrola je předána dalšímu hráči na tahu.
- **Výsledek testu:** Scénář proběhl bez chyb, předávání tahů je zobrazeno přehledně.

6. **Aktualizovat herní plochu**

- **Popis:** Případ užití umožňuje hráči nebo hostiteli aktualizovat svou herní plochu, tak aby odpovídala stavu na serveru.
- **Aktéři:** Hostitel, Hráč
- **Průběh testu:** Tento případ užití byl otestován v rámci případů užití, které ho obsahují. Ty jsou uvedeny na obrázku 3.2.
- **Výsledek testu:** Hrací plocha byla vykreslena bezchybně.

7. **Odemknout ovládací prvky**

- **Popis:** Případ užití umožňuje hráči nebo hostiteli odemknout ovládací prvky herního pole.
- **Aktéři:** Hostitel, Hráč
- **Průběh testu:** Tlačítko pro spuštění hry se odemkne hostiteli při naplnění hry. Tlačítko pro ukončení tahu se odemkne hráči, který je na tahu. Tlačítka pro zobrazení historie zpráv a opuštění hry jsou přístupná po celou dobu simulace.
- **Výsledek testu:** Všechny ovládací prvky se chovají podle očekávání. Scénář nevyvolal žádné chyby.

Závěr

Prvním cílem této práce bylo představit problematiku spolupráce na oligopolistických trzích a provést rešerši existujících simulátorů na poli mikroekonomie a teorie her. Tento cíl jsem splnil v kapitole 2 této práce. Je v ní detailně popsána problematika oligopolu včetně toho, čím se odlišuje od jiných tržních struktur. Dále uvádím stručný úvod do teorie her a jakým způsobem je aplikována v rámci spolupráce na oligopolistických trzích. Herní a spouštěcí strategie uvedené v této kapitole jsem poté integroval do mé aplikace. V závěru kapitoly jsem provedl průzkum existujících řešení, který ukázal, že má aplikace nalezne uplatnění na poli ekonomických simulátorů a vzdělávacích pomůcek.

Druhým cílem bylo analyzovat požadavky a na jejich základě navrhnout a implementovat responzivní webovou aplikaci. V rámci tohoto cíle se mi podařilo splnit nejen původní požadavky na aplikaci jako například responzivní design, ale také dodatečné požadavky jako překlad do anglického jazyka a alternativní světlý vzhled aplikace pro zobrazení na projektoru. Simulační hra je dostupná v režimech pro jednoho nebo dva hráče a přístupná je z prostředí počítače i mobilních zařízení. Aplikaci jsem implementoval s důrazem na jednoduchost, funkčnost a možnost budoucího rozšíření.

Třetím a závěrečným cílem bylo aplikaci řádně otestovat a připravit pro nasazení do provozu. K testování jsem využil vývojářské metody jako například jednotkové a integrační testování. Dále jsem provedl uživatelské testování formou ověření funkčnosti případů užití. Dalším testováním, kterým by měla aplikace projít je testování v širším okruhu uživatelů, které by mohlo objevit drobnější chyby. Má aplikace však aktuálně neobsahuje žádné kritické chyby a je připravena k nasazení, čímž je splněn i poslední cíl.

Plány do budoucna

Aplikace má mnoho potenciálu pro budoucí rozšíření. Jelikož například nebyl implementován trh s diferencovanými produkty, nebylo možné využít k vynucení kooperace strategii cukru a biče. Přestože tato skutečnost neubírá na využitelnosti aplikace, rozšíření pro diferencované produkty by aplikaci přineslo větší variabilitu.

Díky zvoleným technologiím se také jedná o aplikaci, která je připravena pro budoucí integraci do širšího vzdělávacího systému, pokud bude takový systém vytvořen. I kdyby se tak nestalo, aplikace má díky flexibilnímu designu potenciál dobře fungovat i sama o sobě. Je například velmi snadné doplnit nové strategie pro umělou inteligenci, nehrozí tedy, že by nízký počet možností vedl ke ztrátě zájmu o simulaci. Tímto stylem je vyvíjena většina komponent, a tak věřím, že aplikace bude plnit svůj účel po dlouhou dobu a stane se cenným nástrojem pro výuku.

Literatura

[1] Pindyck, R. S.; Rubinfeld, D. L.: *Microeconomics*. Pearson, 9 vydání, 2018, ISBN 978-1-292-21331-6.

[2] Dasva: *Přeloženo z původního 'Supply Demand Right Shift Supply' [online]*. Wikimedia Commons, [cit. 2019-04-25], Licence: Creative Commons BY-SA 3.0. Dostupné z: `https://commons.wikimedia.org/wiki/File:Supply-demand-right-shift-supply.svg`

[3] Paweł Zdziarski (faxe); Astarot: *Adaptováno z původního 'Supply and demand' [online]*. Wikimedia Commons, [cit. 2019-04-25], Licence: Creative Commons BY-SA 3.0. Dostupné z: `https://commons.wikimedia.org/wiki/File:Supply-and-demand.svg`

[4] SilverStar: *Přeloženo z původního 'Supply demand equilibrium" [online]*. Wikimedia Commons, [cit. 2019-04-25], Licence: Creative Commons BY-SA 3.0. Dostupné z: `https://commons.wikimedia.org/wiki/File:Supply-demand-equilibrium.svg`

[5] Perloff, J. M.: *Microeconomics*. Pearson, 6 vydání, 2012, ISBN 978-0-13-139263-2.

[6] Besanko, D. A.; Braeutigam, R. R.: *Microeconomics*. Wiley, čtvrté vydání, 2010, ISBN 978-0-470-56358-8.

[7] Cournot, A.: *Researches Into the Mathematical Principles of the Theory of Wealth (Classic Reprint)*. Forgotten Books, 2017, ISBN 978-1-5279-2084-2.

[8] Česko. Ministerstvo vnitra: Zákon č. 143/2001 Sb. ze dne 4. dubna 2001, Zákon o ochraně hospodářské soutěže. *Sbírka zákonů České republiky*, 2001, částka 58/2001, ISSN 1211-1244.

[9] Organization of the Petroleum Exporting Countries: *OPEC [online]*. [cit. 2019-04-20]. Dostupné z: https://www.opec.org/

[10] Organization of the Petroleum Exporting Countries: *OPEC Share of World Crude Oil Reserves [online]*. [cit. 2019-04-20]. Dostupné z: https://www.opec.org/opec_web/en/data_graphs/330.htm

[11] Ing. Ivo Koubek: *Oligopol jako opakovaná hra*. (mimeo).

[12] Slantchev, B. L.: *Game Theory: Repeated Games [online]*. Březen 2004, [cit. 2019-04-25]. Dostupné z: http://slantchev.ucsd.edu/courses/gt/07-repeated-games.pdf

[13] Axelrod, R.: Effective Choice in the Prisoner's Dilemma. *The Journal of Conflict Resolution*, ročník 24, Březen 1980: s. 3 – 25.

[14] Axelrod, R.: *The Evolution of Cooperation: Revised Edition*. Basic Books, revidované vydání, Prosinec 2006, ISBN 978-0-465-00564-2.

[15] Shubik, M.: Game theory, behavior, and the paradox of the Prisoner's Dilemma: Three solutions. *The Journal of Conflict Resolution*, ročník 14, 1970: s. 181 – 193.

[16] Friedman, J. W.: A Non-cooperative Equilibrium for Supergames. *The Review of Economic Studies*, ročník 38, Leden 1971: s. 1 – 12.

[17] Harris, R. J.: Note on 'optimal policies for the Prisoner's Dilemma.'. *Psychological Review*, ročník 76, 1969: s. 363 – 375.

[18] Nowak, M.; Sigmund, K.: A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoner's Dilemma game. *Nature*, ročník 364, Srpen 1993: s. 56 – 58.

[19] Case, N.: *The Evolution of Trust [online]*. [cit. 2019-04-26]. Dostupné z: https://ncase.me/trust/

[20] Záškodný, M.; Huráb, M.: *Server pro podporu výuky teorie her [online]*. [cit. 2019-04-26]. Dostupné z: http://teh-vsb.aspone.cz/Default.aspx

[21] Axelrod, R.: *Complexity of Cooperation [online]*. [cit. 2019-05-04]. Dostupné z: http://www-personal.umich.edu/~axe/research/Software/CC/CC2.html

[22] Knight, V.; Harper, M.; Campbell, O.; aj.: *Axelrod Python library [online]*. [cit. 2019-05-04]. Dostupné z: https://axelrod.readthedocs.io/en/stable/index.html

[23] Kluger, M.: *Simulace modelu monopolistické konkurence – aplikace pro výuku ekonomických předmětů na FIT*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2016.

[24] Gruyer, N.; Toublanc, N.: *Economics Games [online]*. [cit. 2019-05-08]. Dostupné z: `https://economics-games.com/`

[25] We Are Social; Statcounter: *Adaptováno z původního 'Digital in 2018' [online]*. [cit. 2019-06-20]. Dostupné z: `https://www.slideshare.net/wearesocial/digital-in-2018-global-overview-86860338`

[26] Node.js Foundation: *Node.js [online]*. [cit. 2019-06-25]. Dostupné z: `https://nodejs.org/en/`

[27] Node.js Foundation: *About Node.js [online]*. [cit. 2019-06-25]. Dostupné z: `https://nodejs.org/en/about/`

[28] StrongLoop, IBM and contributors: *Express Framework for Node.js [online]*. [cit. 2019-06-25]. Dostupné z: `http://expressjs.com/`

[29] Přispěvatelé: *Socket.io [online]*. [cit. 2019-06-25]. Dostupné z: `http://socket.io/`

[30] Bootstrap team: *Bootstrap frontend framework [online]*. [cit. 2019-06-25]. Dostupné z: `http://getbootstrap.com/`

[31] Bootstrap team: *Bootstrap history [online]*. [cit. 2019-06-25]. Dostupné z: `http://getbootstrap.com/about/`

[32] Různí přispěvatelé: *React Bootstrap [online]*. [cit. 2019-06-25]. Dostupné z: `https://react-bootstrap.github.io/`

[33] Yahoo Presentation Technologies (YPT): *FormatJS [online]*. [cit. 2019-06-25]. Dostupné z: `https://formatjs.io`

Příloha A

Seznam použitých zkratek

API Application Program Interface

CSS Cascading Style Sheets

DOM Document Object Model

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

I/O Input/Output

JS JavaScript

JSX Javascript XML

MC Marginal Cost

MP Marginal Profit

MR Marginal Revenue

MVC Model View Controller

NPM Node Package Manager

OPEC Organization of the Petroleum Exporting Countries

PV Present Value

RQ Renegade Quantity

STM State Machine

TC Total Cost

TR Total Revenue

TS TypeScript

UC Use Case

UML Unified Modeling Language

URI Uniform Resource Indentifier

URL Uniform Resource Locator

XML Extensible Markup Language

Příloha **B**

Seznam použitých pojmů

Babel Nástroj sloužící k překladu kódu psaném v nadstavbových jazycích (jako například JSX) do čistého javascriptu.

Backend Serverová část aplikace se kterou neinteraguje uživatel přímo. Typicky zajišťuje výpočetní a datové operace.

Bootstrap Frontendový framework specializující se na vykreslování obsahu pro mobilní zařízení.

Controller Někdy také označován jako řadič, Controller je komponenta, která reaguje na události (typicky pocházející od uživatele) a zajišťuje změny v modelu (Model) nebo v pohledu (View).

Duopol Forma oligopolu při které na trhu figurují pouze dvě firmy.

Express Express je framework operující v prostředí Node.js a je vytvořený v jazyce Javascript.

FormatJS Knihovna pro zajištění překladu aplikace do jiných jazyků.

Framework Je komplexní řešení usnadňující vývoj aplikace.

Frontend Frontend je část aplikace, kterou přímo vidí její uživatel.

Javascript Programovací jazyk primárně sloužící pro vytváření dynamického obsahu v prohlížeči.

Jest Knihovna sloužící pro vytváření testů (zejména jednotkových) v prostředí Node.js.

Kartel Tržní dohoda, která narušuje hospodářskou soutěž. Má za cíl minimalizovat konkurenci v daném odvětví a co nejvíce maximalizovat zisk.

Mobile first design Jedná se o filosofii vývoje aplikací. Aplikace je v prvé řadě optimalizována na malých obrazovkách mobilních zařízení a až poté na velkých obrazovkách.

Node.js Node je runtime prostředí, které ve své podstatě umožňuje běh aplikace.

Oligopol Forma tržní spolupráce při které malé množství firem na straně nabídky zabírá většinový podíl trhu.

React Knihovna pro vytváření webových komponent.

Routing V českém jazyce se také označuje jako směrování. Určuje, kam má být uživatel odkázán pomocí příslušného URI.

Socket Softwarový objekt propojující aplikaci se síťovým protokolem.

TSLint Nástroj sloužící pro kontrolu dodržování standardů psaní kódu v jazyce Typescript.

Typescript Nadstavba programovacího jazyka Javascript umožňující plně využít principy objektově orientovaného programování.

Unit Test V českém jazyce také překládán jako jednotkový test. Má za úkol otestovat izolovanou funkčnost segmentu kódu.

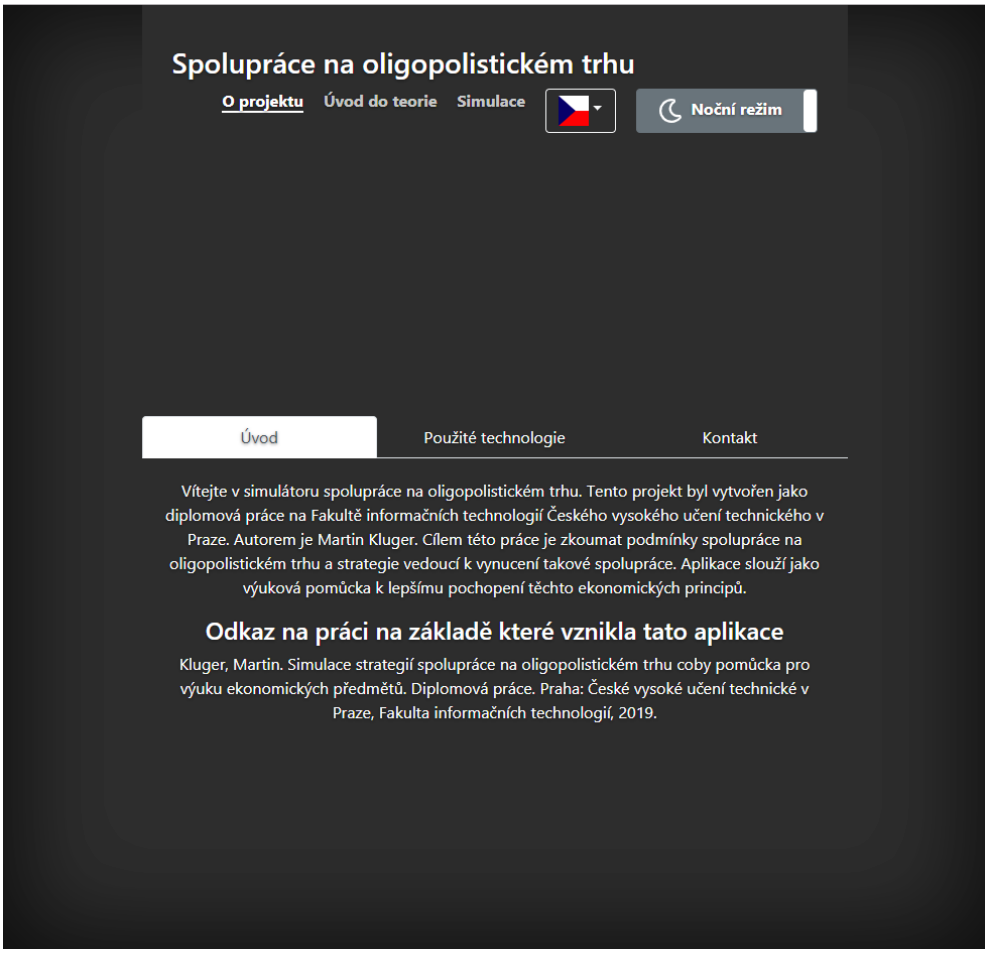
Vězňovo dilema Typ hry s nenulovým součtem v rámci teorie her.

WebSocket Komunikační protokol sloužící pro přímou komunikaci mezi klienty s nízkou dobou odezvy a ztrátovostí.

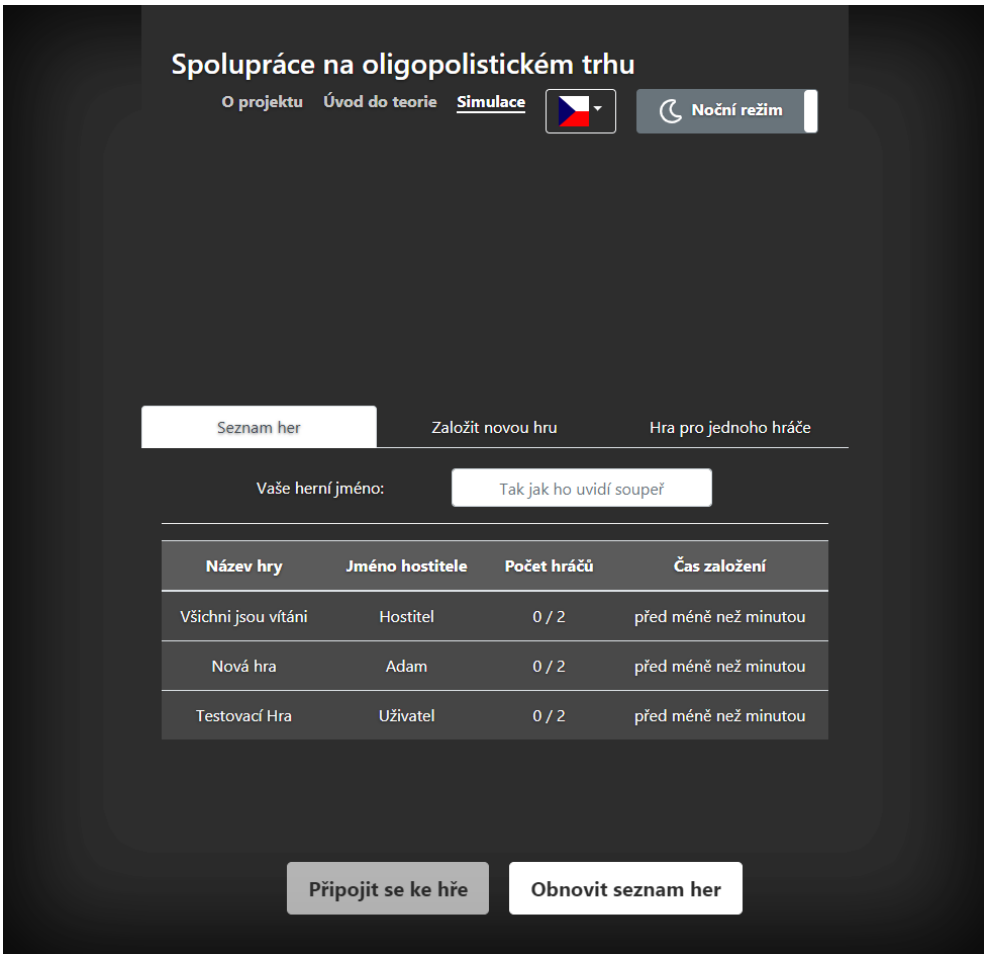
Příloha C

Vizuální vzhled aplikace

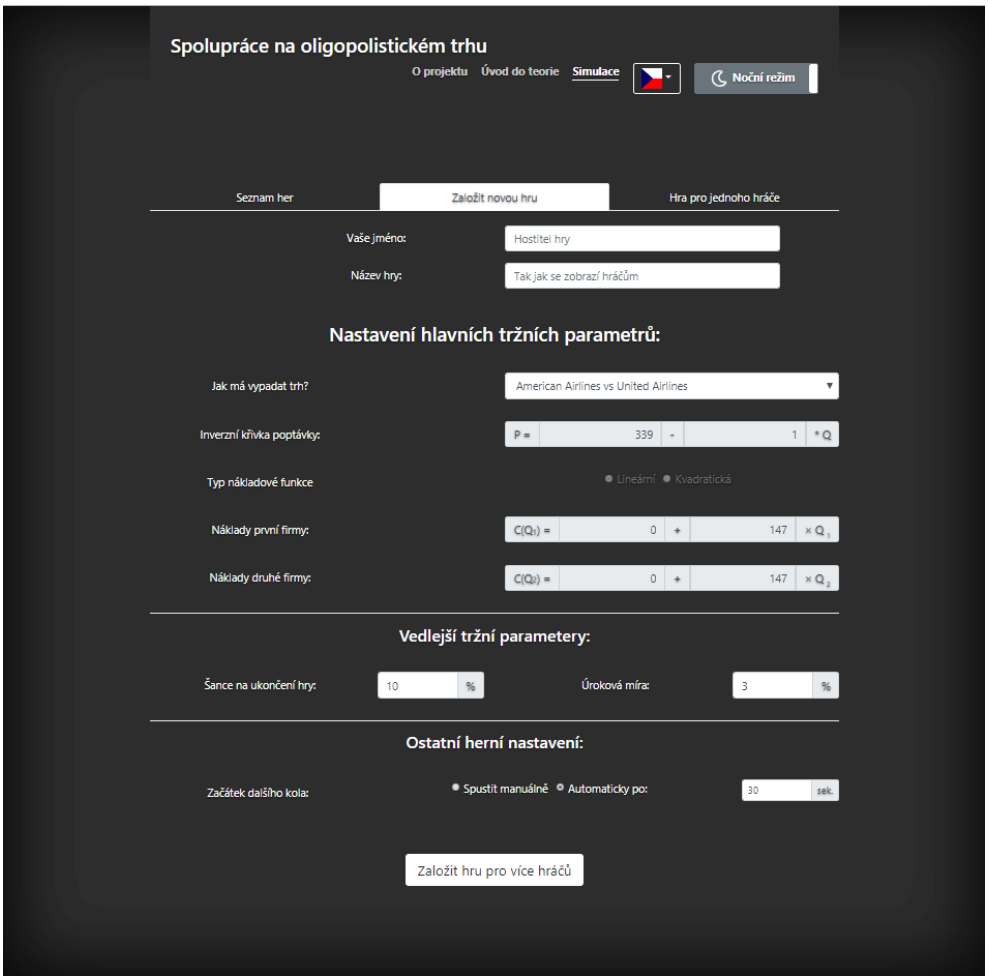
Tato kapitola obsahuje snímky jednotlivých obrazovek použitých v aplikaci. Jsou zde uvedeny proto, aby čtenáři daly lepší představu o finální podobě aplikace.



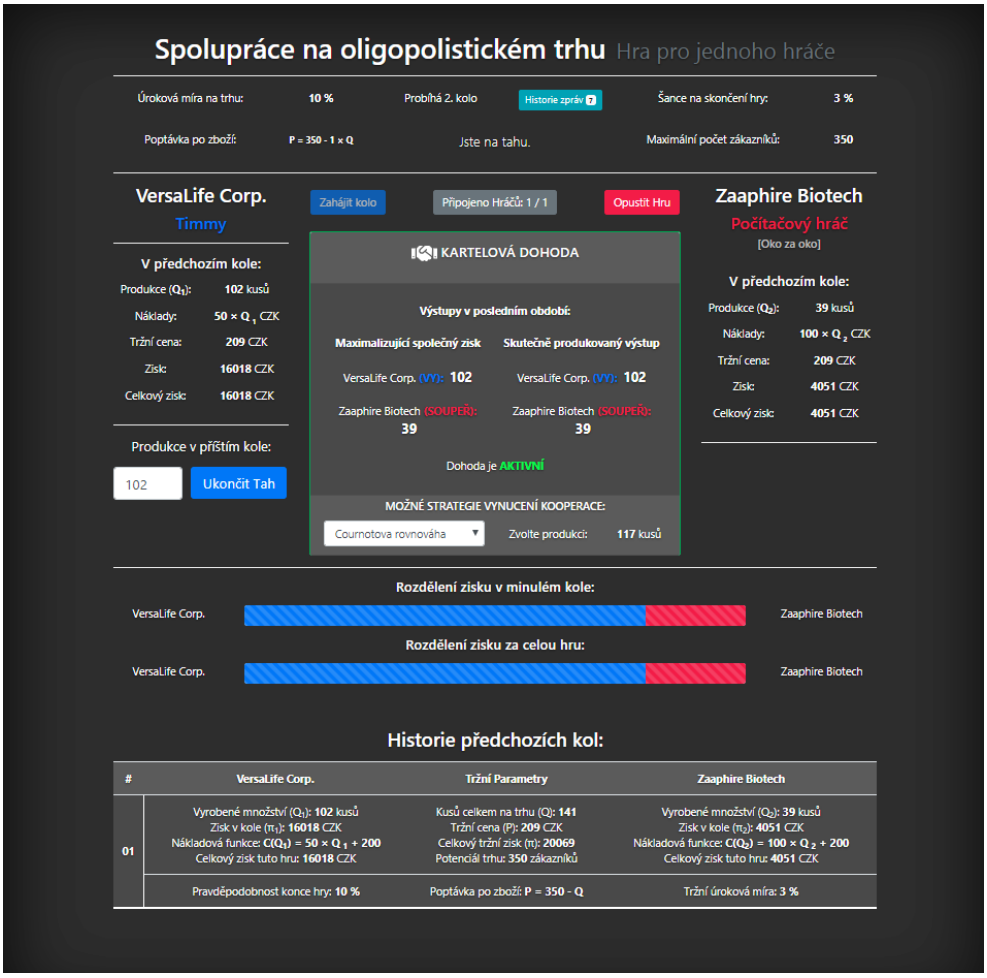
Obrázek C.1: Úvodní obrazovka aplikace v českém jazyce.



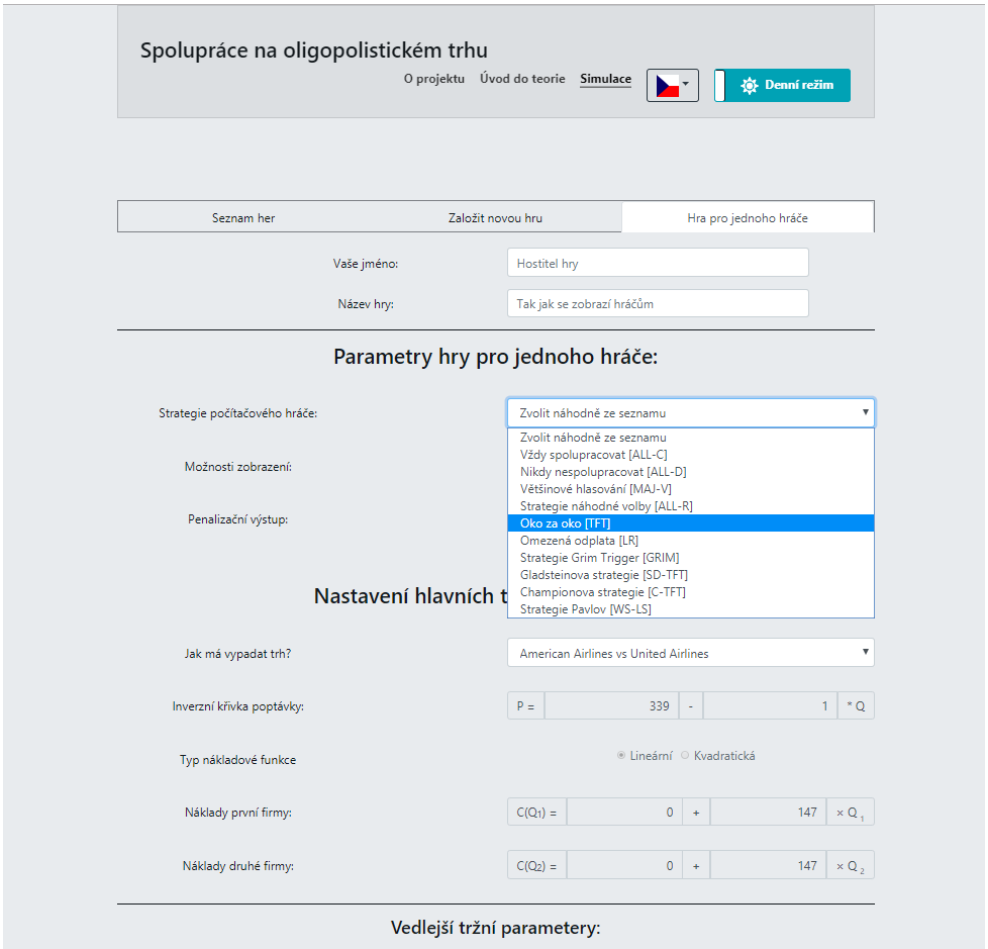
Obrázek C.2: Seznam her včetně ukázkových her v přípravné fázi herního modulu aplikace.



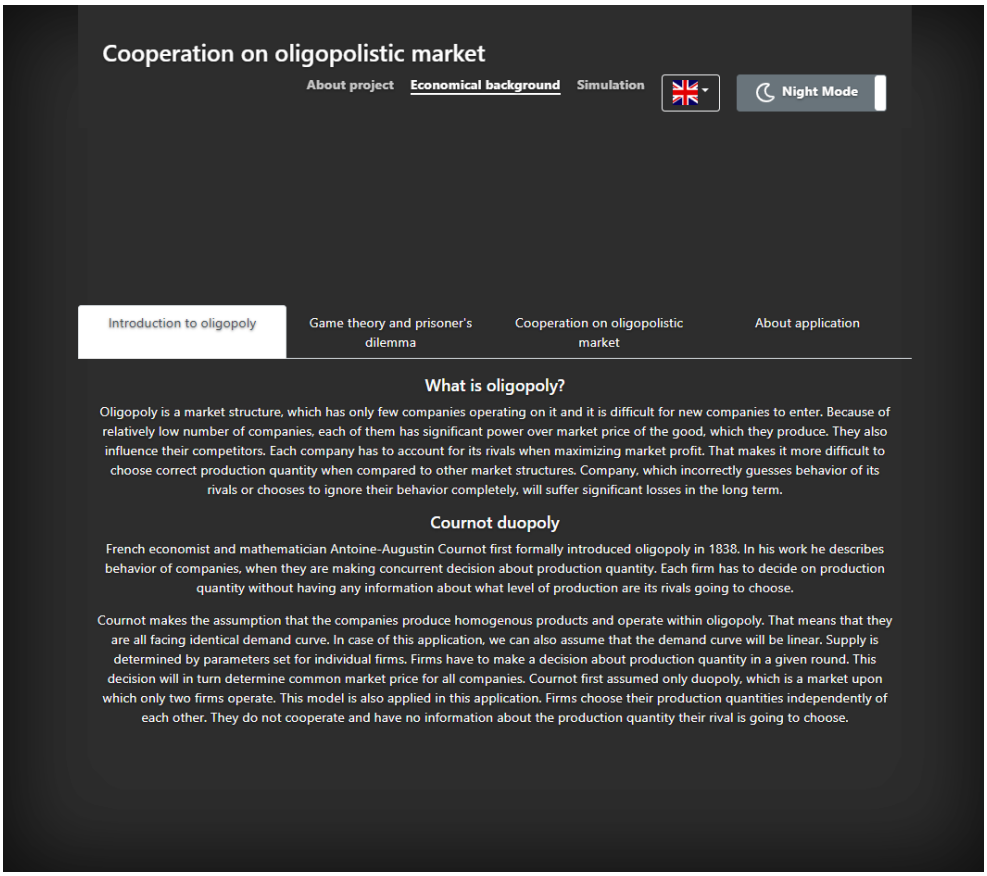
Obrázek C.3: Obrazovka pro vytvoření hry pro dva hráče.



Obrázek C.4: Ukázka obrazovky v probíhající hře pro jednoho hráče.



Obrázek C.5: Ukázka světlého designu aplikace. Uvedena je obrazovka pro vytvoření hry pro jednoho hráče.



Obrázek C.6: Ukázka překladu aplikace do anglického jazyka. Uvedena je obrazovka vysvětlující teorii oligopolu.

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF