



ZADÁNÍ BAKALÁ SKÉ PRÁCE

Název:	Simulace koncentrovaného trhu – aplikace pro výuku ekonomických p edm t na FIT
Student:	Michal Kocánek
Vedoucí:	Ing. Mgr. Pavla Vozárová, Ph.D., M.A.
Studijní program:	Informatika
Studijní obor:	Informa ní systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Formou rešerše se seznámte se základními mikroekonomickými teoriemi popisujícími r zn koncentrované trhy. Prozkoumejte a popište webové aplikace, které umož ů ují student m ekonomie si tyto principy osvojit. Navrh ů te a implementujte responzivní interaktivní webovou aplikaci, ve které budou studenti na cvi eních z ekonomických p edm t na FIT vypl ovat svá rozhodnutí ve h e simulující rovnováhu na více i mén koncentrovaném trhu, popsané v J. Parker: Using Laboratory Experiments to Teach Introductory Economics. Aplikace musí být schopna v reálném ase zaznamenat hodnoty vypl ované jedním až osmi hrá i, vyhodnocovat výsledek jednotlivých kol hry na základ daného algoritmu a dostate n ilustrativní formou prezentovat pr b žné i kone né výsledky hry v etn teoreticky optimálního ešení. Aplikaci navrh ů te tak, aby vyu ujícím umož ovala m nit parametry hry. Implementujte ji v eském jazyce i v anglické mutaci.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdí k, CSc.
d kan

V Praze dne 11. listopadu 2015

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
KATEDRA SOFTWAREVÉHO INŽENÝRSTVÍ



Bakalářská práce

**Simulace koncentrovaného trhu – aplikace
pro výuku ekonomických předmětů na FIT**

Michal Kocánek

Vedoucí práce: Ing. Mgr. Pavla VOZÁROVÁ, Ph.D., M.A.

16. května 2016

Poděkování

Zde bych rád poděkoval své vedoucí Ing. Mgr. Pavle Vozárové za cenné rady a podnětné připomínky při tvorbě mé práce. Velké díky patří také rodině a přátelům za podporu při studiu.

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 Lázních Bohdaneč dne 16. května 2016

.....

České vysoké učení technické v Praze
Fakulta informačních technologií

© 2016 Michal Kocánek. Všechna práva vyhrazena.

Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Kocánek, Michal. *Simulace koncentrovaného trhu – aplikace pro výuku ekonomických předmětů na FIT*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato bakalářská práce se zabývá tvorbou webové aplikace na podporu výuky ekonomických předmětů, která simuluje koncentrovaný trh. Popis experimentu se nachází v teoretické části. Součástí praktické části práce je analýza, návrh, implementace, testování a nasazení aplikace. Zvolený problém jsem vyřešil pomocí frameworku Java Spring. Výstupem je aplikace spuštěná na veřejně dostupném serveru a příloha této práce obsahuje její zdrojový kód.

Klíčová slova Java, Spring Framework, webová aplikace, ekonomická hra

Abstract

This bachelor thesis deals with development web application of economic game. In theoretical part is discussed the economic game. Users of this application will be students of economy at FIT CTU in Prague. This thesis describes analysis, design, implementation, testing and deployment. Result of this thesis is application prototype deployed on public available server in test mode.

Keywords Java, Spring Framework, web application, economic game

Obsah

Úvod	1
1 Současný stav řešení problematiky	3
1.1 economics-games.com	3
1.2 MobLab	5
1.3 Veconlab	5
1.4 FEELE	6
1.5 Shrnutí současného stavu	6
2 Popis experimentu	7
2.1 Popis experimentu	7
2.2 Role kupujícího	7
2.3 Role prodávajícího	8
2.4 Průběh hry	10
2.5 Výsledky experimentu	10
3 Analýza	11
3.1 V čem je přínos aplikace	11
3.2 Požadavky na aplikaci	11
3.3 Model případů užití	13
3.4 Analytický doménový model	17
4 Návrh	19
4.1 Použité technologie	19
4.2 Architektura aplikace	22
4.3 Databázový model	22
4.4 Návrhový model tříd	23
4.5 Popis balíčků	23
4.6 Autentizace a autorizace	25

5	Realizace	27
5.1	Implementace algoritmu trhu	27
5.2	Dependency Injection	29
5.3	Spring Security	29
5.4	Validace	31
5.5	Responzivní web	31
6	Ukázka aplikace	33
6.1	Úvodní stránka	33
6.2	Stránka pro studenta	34
6.3	Stránka pro učitele	35
7	Testování	37
7.1	Unitové testování	38
7.2	Funkční testování	38
8	Nasazení	41
8.1	Webový hosting	41
8.2	Nasazení aplikace	41
8.3	Nasazení aplikace na univerzitní server	42
	Závěr	43
	Literatura	45
A	Seznam použitých zkratk	49
B	Obsah příloženého CD	51

Seznam obrázků

1.1	Ukázka z economics-games.com	4
3.1	Uživatelé aplikace	14
3.2	Případ užití Učitel	15
3.3	Případ užití Student	16
3.4	Případ užití Obecné	16
3.5	Analytický doménový model	18
4.1	Návrhový model tříd - backend	23
4.2	Návrhový model tříd - backend	24
6.1	Úvodní stránka aplikace	33
6.2	Stránka pro studenta	34
6.3	Stránka pro učitele	35
7.1	Relativní cena opravy chyby	37
7.2	Selenium	39

Seznam tabulek

2.1	Poptávková funkce	9
2.2	Nákladová funkce	9

Úvod

Výuka ekonomických předmětů na FIT ČVUT nepatří k těm nejdůležitějším na naší fakultě. Studenti často na tyto předměty ani nechodí, což je dle mého názoru velká škoda, protože ekonomie nabízí celou řadu oblastí, které by mohly zaujmout i člověka zaměřeného čistě technickým směrem. S nápadem zatraktivnit výuku využitím ekonomické hry přišla vedoucí mé bakalářské práce, ale nyní musí vyhodnocovat jednotlivé hry ručně a to velmi zdržuje. Proto jsem se za její podpory rozhodl naimplementovat webovou aplikaci, která studenty seznámí se základními principy na koncentrovaném trhu.

Hry při výuce jsou skvělým způsobem jak studenty vtáhnout do probírané látky a zároveň představit klíčové ekonomické koncepty. Především dělají výuku zajímavější a příjemnější jak pro žáky, tak i pro učitele. Tyto experimenty mohou sloužit jako základ k pozorování, které učitel může využít ke konceptualizaci, tj. k vysvětlení nového teoretického konceptu. Vědomosti nabyté tímto způsobem si budou studenti pamatovat ještě dlouho poté, co dokončí předmět. Experimenty nutí studenty nejprve hry prožít a pak je analyzovat.

Aplikace bude zlepšovat úroveň výuky především tím, že umožní cvičícím se soustředit na vysvětlení jevů, které hra zprostředkovává, a výrazně urychlí celou hodinu. Studenti budou mít k dispozici moderní aplikaci, která bude dostupná jak z klasického počítače nebo notebooku, tak i z mobilních zařízení.

První kapitola se věnuje analýze dostupných webových portálů zaměřených na podporu výuky ekonomických předmětů formou experimentů. Zde budou uvedena srovnání současných řešení problematiky a hodnocení jejich kladů a záporů.

Druhá část vysvětluje principy implementované hry. V této kapitole bude čtenář seznámen s pravidly experimentu a jeho průběhem v současné podobě bez podpory webové aplikace.

Třetí část se zabývá analýzou problému. Čtenář zde nalezne požadavky, diagram případů užití a analytický doménový model aplikace.

Čtvrtá část popisuje problematiku návrhu. V této kapitole jsou sepsány všechny technologie použité k vytvoření aplikace. Dále je zde uveden návrhový

a databázový model a zamýšlená architektura aplikace. Také jsou zde stručně popsány komponenty, ze kterých se aplikace skládá.

Páta část této práce se zabývá implementací aplikace. Je zde stručně popsán algoritmus vyhodnocení hry a způsob, jakým jsou vyřešeny problémy spojené s vytvářením aplikace.

Následující část se věnuje testování. Jsou zde popsány způsoby testování a využití technologie, které zajišťují kvalitu aplikace.

Poslední část popisuje proces nasazení aplikace na veřejně dostupný server.

Cílem práce je vytvořit fungující webovou aplikaci, která bude sloužit při výuce na Fakultě informačních technologií ČVUT. Snahou je zaujmout studenty při výuce, zlepšit obsah cvičení a ulehčit práci vyučujících.

Cílem této práce není vymyslet novou ekonomickou hru podporující výuku na školách, ale pouze převedení jedné konkrétní do elektronické podoby dostupné ve webovém prostředí.

Současný stav řešení problematiky

Tato kapitola pojednává o současném stavu řešení, zejména hodnotí klady a zápory jednotlivých webových aplikací a jejich využitelnost při výuce. Jediným kritériem, které aplikace musela splnit, byla dostupnost z webového prohlížeče.

1.1 economics-games.com

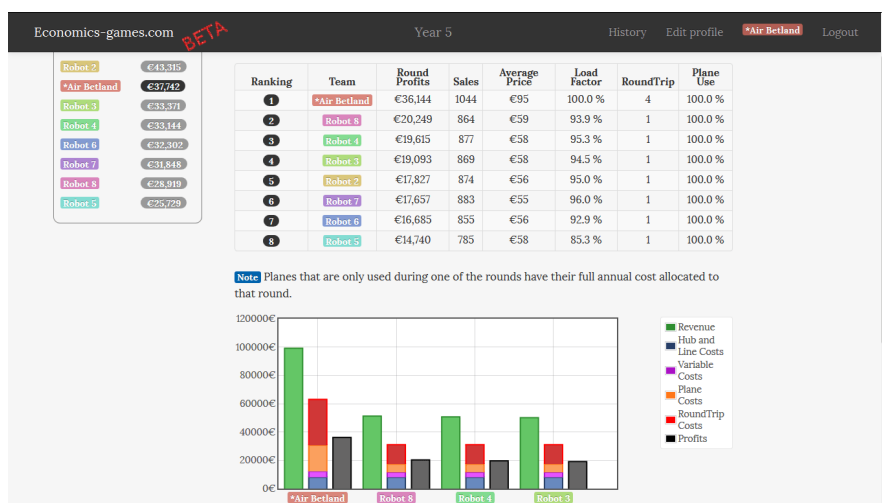
Tento portál [1] umožňuje výběr až z přibližně patnácti různých her z ekonomického prostředí. Hry jsou jednoduché na pochopení a není zapotřebí žádná registrace, navíc je celá služba zdarma.

1.1.1 Příklad hry

Zde popíšeme konkrétní hru, která se nejvíce blíží experimentu implementovanému v této práci. Jedná se o simulaci trhu, kde producenti nabízejí své zboží a aplikace automaticky nabídky vyhodnocuje a simuluje chování kupců. Její název na těchto stránkách je *Competition Game: Impact of fixed costs and capacity constraints on price and profit, with differentiated goods*. Existuje zde ale několik rozdílů:

- V jednom kole lze dodat na čtyři trhy s lehce odlišnými vlastnostmi (rozdíly jsou v existenci fixních nákladů a velikostí produkce).
- Hráči nerozhodují o množství dodaného na trh - to se určuje druhem trhu (viz předchozí bod)
- Producenti se v pozdější fázi neslučují, takže se nevytváří oligopol ani monopol

1. SOUČASNÝ STAV ŘEŠENÍ PROBLEMATIKY



Obrázek 1.1: Ukázka stránek economics-games.com. Na obrázku je příklad jedné z dostupných her

Založení hry je velice jednoduché a intuitivní - nutné je pouze vybrat druh experimentu, následně nastavit počet participujících studentů a počet na sobě nezávislých her. Poté aplikace náhodně určí, jak budou studenti mezi různé hry rozprostřeni, pouze s omezením, že každý student může být přidělen právě k jedné hře. Například pro 30 studentů je ideální zvolit pět různých her, protože v každé hře bude šest různých hráčů. Systém nageneruje unikátní přihlašovací údaje pro všechny hráče plus jeden pro správce.

Cílem každého studenta je maximalizovat svůj zisk. Toho může dosáhnout tím, že se bude účastnit obchodování na trzích, kterých je ve hře několik druhů a liší se svými vlastnostmi - zejména fixními náklady a velikostí produkce, které se na trh dodává (studenti nemají možnost ovlivnit množství vyrobeného zboží). Kolo končí až všichni hráči ze stejné hry zvolí svoji nabídku a tu potvrdí přechodem na další stránku, kde se studentům zobrazí graf s výsledky. Tento cyklus se opakuje.

Studenti mají ve hře možnost změnit své jméno, které se zobrazuje ostatním, a své přihlašovací údaje.

Učitel kurzu, který se přihlásí jako administrátor, je schopen sledovat průběh hry, měnit studentům přihlašovací údaje a vynutit konec kola, pokud nějaký student neodešle data ke zpracování a tím zabrání aplikaci automaticky ukončit kolo.

1.1.2 Hodnocení

Celý portál hodnotím kladně především díky snadnosti s jakou lze novou hru založit, nejsou zde žádné větší překážky, jako je například registrace. Vzhled webových stránek je moderní, ale bohužel příliš nepodporuje responzivní de-

sign. Všechny stránky jsou uzpůsobeny pouze velkým obrazovkám a jestliže je zobrazíme např. na mobilním telefonu, komponenty se špatně přizpůsobí menší velikosti a začnou se překrývat a to celkově kazí dojem z jinak povedeného designu.

Jistou nevýhodou může být absence české lokalizace. Celý portál je dostupný pouze v anglické nebo francouzské verzi. Další nevýhodou je již zmíněná absence responzivního designu.

Economics games doporučuji zmínit jako alternativu ke klasické výuce, může studenty snadno nadchnout. Všechny hry jsou jednoduché na pochopení a nezaberou dlouhou dobu.

1.2 MobLab

Moblab [2] je portál zaměřený na podporu výuky. Lze zde nalézt spousty výukových her z oblasti ekonomie, managementu nebo teorie her.

Pokud chce učitel spustit libovolnou hru, tak se musí nejprve registrovat, vytvářet kurz a vybranou hru asociovat se založeným kurzem. Celé to působí komplikovaným a zdoluhavým dojmem. Velkou nevýhodou je skutečnost, že služba není zdarma.

Pozitivní dojem dělá zpracování stránek. Design působí moderně a je plně responzivní. Moblab dokonce vytvořil aplikace pro mobilní platformu Android a iOS, které jsou bezplatně ke stažení v Google Play nebo v AppStore.

1.3 Veconlab

Další portál, který cílí na podporu výuky, je Veconlab [3]. Najdeme zde tučt her z různých oblastí ekonomie, které jsou volně dostupné.

Učitel si nejprve zvolí kategorii a pak typ experimentu, který chce využít ve svých hodinách. Na výběr je přibližně deset různých kategorií, každá s průměrným počtem pěti her. Ke spuštění jakékoliv hry je nejdříve nutná krátká registrace, teprve potom lze přistoupit k založení hry. Aplikace poté vygeneruje unikátní *session name*, s jehož pomocí se studenti zaregistrují do hry. U většiny her existuje konfigurace, díky níž si učitel upraví hru podle svých představ.

Po úspěšném registrování do hry jsou studentům zobrazeny instrukce, co se od nich očekává. Tyto instrukce se mohou lišit pro různé role v experimentu, např. jiné jsou pro kupujícího a jiné pro prodávajícího.

Design stránek působí zastarale. Bohužel zobrazení stránek na mobilních přístrojích se nepřizpůsobí velikosti, tudíž text velice malý a musí se přibližovat. I přes tyto drobné nedostatky jsou stránky plně funkční a použitelné při výuce.

1.4 FEELE

Finance and Economics Experimental Laboratory at Exeter (FEELE) [4] je webová aplikace s množstvím ekonomických experimentů převedených do elektronické podoby. Návrh těchto stránek záměrně vychází z portálu Veconlab [5], ale nabízí menší portfolio her.

Experimenty lze založit pouze jako přihlášený uživatel, nicméně registrace je zdarma. Studenti se do aplikace přihlásí na základě údajů, které získají od učitele.

Přestože design není nejmodernější, je vzhled stránek plně dostačující. Jistou obtíž může působit přístup na tyto stránky z mobilních zařízení, protože nejsou pro malé zařízení přizpůsobené, přesto se lze experimentů účastnit pouze s těmito zařízeními.

1.5 Shrnutí současného stavu

Výhodou všech portálů je velká různorodost nabízených ekonomických experimentů. Bohužel se přesto nepodařilo nalézt hru, která by přesně odpovídala popisu v následující kapitole. Na všech stránkách lze vybírat z desítek různých experimentů, které pokrývají většinu ekonomické praxe. Kladně hodnotím bezplatnou dostupnost aplikací, kromě jediné jsou všechny zdarma.

Naopak největší slabinu hodnocených aplikací shledávám v chybějícím responzivním designu, který byl implementován pouze aplikací v Moblabu. Nepříjemná je i absence české lokalizace u všech hodnocených stránek, ale to je pouze drobný nedostatek, protože dnes již většinu studentů nedělá problém číst pokyny v anglickém jazyce.

Aplikace implementovaná v této práci by měla skloubit moderní prostředí s responzivním designem, které u většiny z uvedených příkladů chyběla. Hotová aplikace nemá za cíl konkurovat ostatním portálům co se týče různorodosti nabídky experimentů, ale zaměřit se pouze na jeden konkrétní experiment, který bude popsán v další části.

Popis experimentu

V této kapitole bude čtenář seznámem s experimentem, jehož převedení do webové aplikace je cílem této práce. Kompletní popis lze nalézt v textu J. Parkera [6]. V odkazované práci je možné naleznout i další hry využitelné při výuce, nejenom tu, které se kapitola věnuje.

2.1 Popis experimentu

Hra simuluje chování na posted-offer trhu, kde producenti zveřejňují své nabídky. Ceny jsou zde konečné a kupující je nemohou přímo ovlivnit smlouváním, ale pouze se rozhodují jaké množství produktu jsou ochotni koupit při daných cenách. Tento model trhu je běžný pro většinu rozvinutých ekonomik.

Role nabízejícího je mnohem zajímavější než role kupce. Jsou to právě oni, kdo stanovují ceny a určují úroveň produkce. Kupci pak pouze nakupují za nejnižší cenu takové množství, ze kterého mají profit. Proto je role kupce často v těchto hrách simulována. To je i případ tohoto experimentu, kdy studenti hrají pouze roli prodávajícího a roli kupujícího přebírá učitel.

Simulace se sestává z konečného počtu opakujících se kol. Každé kolo začínají prodávající, kteří se rozhodují jaké množství jsou ochotni na trh dodat a za jakou cenu. Až se každý z nich rozhodne, tak své nabídky odevzdají vyučujícímu. Ten potom nabídku vyhodnotí a výsledek kola zveřejní například ve formě tabulky, kterou napíše na tabuly.

2.2 Role kupujícího

Jak bylo již zmíněno, tak je role kupujících simulována učitelem, který rozhoduje čím zboží se prodá a čím ne podle následující metody:

Nejprve se roztrídí nabídky podle ceny a postupně se začíná od té nejnižší. Pro nejnižší nabídku učitel zjistí z tabulky poptávky 2.1 ochotu trhu nakupovat za danou cenu, tj. jaké množství se při dané ceně nakoupí. Pokud je

2. POPIS EXPERIMENTU

množství u podané nabídky s nejnižší cenou menší než množství, které učitel zjistil z tabulky, pak student s touto nabídkou prodal všechno zboží. Například student nabídne 15 kusů za cenu 60 a tomu odpovídá 62 prodaných jednotek zboží na trhu, student tedy prodá všech 15 kusů. Pokud ale student nabídl více než je trh ochoten koupit, tak prodal pouze množství poptávané trhem. Výsledek transakce si učitel poznamená.

Jestliže student s nejnižší nabídnutou cenou selhal v prodeji všech jednotek zboží, tak student s druhou nejnižší cenou automaticky neprodá nic. Tato informace může usnadnit rozhodovací proces. Na druhou stranu, pokud student s nejnižší cenou prodal vše, což je obvyklý případ, tak učitel vyhodnotí druhou nejvýhodnější nabídku. Nyní učitel zjistí poptávaný objem při dané ceně, ale snížený o již prodané zboží. Výsledek opět porovná s nabízeným množstvím - pokud je poptávané množství nižší než studentem nabízené, tak učitel vyhodnotí maximum, které se prodá. Například pokud se v předchozím kroku prodalo množství o deseti kusech a nabízená cena je 70, tak se na trhu prodá pouze 32 kusů místo 42. Opět platí, že pokud se nepodařilo prodát vše, tak je již zbytečné pokračovat, protože ostatní studenti nabízejí za cenu vyšší než trh akceptuje.

Tento proces se opakuje, dokud studenti zboží dokáží prodat. V každém kroku se musí brát v potaz již prodané zboží - nejprve se zjistí, kolik se toho při dané ceně prodá a od tohoto množství se odečte prodané zboží studenty s nižší nabídkou a výsledek se porovná s nabízeným objemem transakce.

V pozdějších částech hry mohou studenti nabízet zboží za stejnou cenu. V případě, že tato situace nastane, musí být skupina studentů vyhodnocena najednou. Pokud není všechno zboží prodáno, tak se učitel musí postarat o spravedlivé rozdělení mezi všechny studenty.

Na konci budou všechny výsledky napsány na tabuli a studenti si budou moci udělat přehled, za jaké ceny se nakoupilo a za jaké ne. Tabulka poptávky se za celou hru nemění a to ani v případě sloučení studentů do oligopolu nebo monopolu v pozdějších částech hry.

2.3 Role prodávajícího

Roli prodávajícího budou hrát studenti, ale nejprve je učitel musí seznámit s pravidly experimentu.

Ta jsou jednoduchá a žáci většinou nemají problém je pochopit. Jejich úkolem je dodat výrobky na trh a určit jejich cenu. Vyrobit toho můžou jen tolik, jaká je kapacita výrobní linky. Výrobní náklady jednoho kusu nejsou stejné a s každým dalším vyrobeným kusem stoupají. Přesně se řídí touto funkcí 2.2, kterou by studenti měli mít neustále u sebe, aby se jí mohli řídit. Jednou z nejdůležitější informací, kterou by si měli uvědomit, je to, že zboží nelze skladovat, tedy že neprodané zboží nelze přenášet z jednoho kola do druhého.

Tabulka 2.1: Tabulka simuluje chování kupců

Cena	Množství	Cena	Množství	Cena	Množství
90	2	75	32	60	62
89	4	74	34	59	64
88	6	73	36	58	66
87	8	72	38	57	68
86	10	71	40	56	70
85	12	70	42	55	72
84	14	69	44	54	74
83	16	68	46	53	76
82	18	67	48	52	78
81	20	66	50	51	80
80	22	65	52	50	82
79	24	64	54	49	84
78	26	63	56	48	86
77	28	62	58	47	88
76	30	61	60	46	90

Tabulka 2.2: Nákladová funkce pro šest různých hráčů. N značí náklady na n-tý kus. Například pro hráče A jsou náklady na výrobu tří kusů 60.

n	A	B	C	D	E	F
1	15	30	7	24	12	15
2	20	33	14	28	18	20
3	25	36	21	32	24	25
4	30	39	28	36	30	30
5	35	42	35	40	36	35
6	40	45	42	44	42	40
7	45	48	49	48	48	45
8	50	51	56	52	54	50
9	55	54	63	56	60	55
10	60	57	70	60	66	60
11	65	60	77	64	72	65
12	70	63	84	68	78	70
13	75	66	91	72	84	75
14	80	69	98	76	90	80
15	85	72	105	80	96	85

2.4 Průběh hry

Hra je rozdělena na kola. Na začátku studenti dostanou nákladovou funkci, kterou se musí řídit. Tato funkce určuje náklady a kapacitu výroby. V každém kole studenti stanovují cenu a množství, které chtějí dodat na trh. Na konci kola učitel vyhodnotí nabídku a nakoupí za nejvýhodnější cenu. Velikost transakce se stanoví na základě cen, které hráči učinili, a také podle poptávkové funkce. Aby studenti mohli maximalizovat svůj zisk, měli by se snažit tuto funkci zjistit. K tomu jim pomůžou výsledky kola, které vyučující zveřejní např. formou tabulky napsané na tabuli.

Po určitém počtu kol cvičící vyhlásí sloučení firem do dvou. Studentům je k dispozici nová nákladová funkce, která reflektuje sloučení výrobních linek, takže mají možnost dodat na trh až 45 kusů, ale nyní musí spolupracovat v týmu. Samotná hra pokračuje dál podle nezměněných pravidel. Ideálně by k prvnímu sloučení firem mělo dojít až poté, co na trhu vznikne rovnováha mezi nabídkou a poptávkou. Ale protože tato rovnováha může nastat až po velmi dlouhé době, není to nutnou podmínkou.

V poslední části hry se všichni spojí do jednoho monopolu s výrobní kapacitou všech dílčích společností. Studenti v této části budou zejména bojovat s efektivní komunikací.

2.5 Výsledky experimentu

Z první části experimentu by si studenti měli odnést znalosti o tom, jak se tvoří rovnováha mezi nabídkou a poptávkou.

Pozdější část experimentu by měla prakticky ukázat, co se stane s výnosy, pokud se na trhu sníží konkurence.

V průběhu hry a každého kola spolu hráči budou moci komunikovat a domlouvat se na cenách - v některých hrách dokonce může vzniknout kartel. Studenti mohou domluvu dodržet a stanovenou cenu dodržet, nebo mohou nabídku nastavit o něco níže a zvýšit svůj zisk. Tento jev byl již pozorován a posán jako tzv. věžňovo dilema [7].

Analýza

Tato kapitola se věnuje problematice domény aplikace, dále je zde uvedena krátká studie o požadavcích kladených na aplikaci a způsobu, jakým bude s aplikací zacházeno. Výsledné poznatky budou základem pro návrh v další kapitole.

3.1 V čem je přínos aplikace

Hlavním přínosem aplikace je automatizování průběhu experimentu pomocí výpočetní techniky. Cílem aplikace je ušetřit čas, který se při hodinách musí vynaložit na vyhodnocení experimentu a poskytnout učitelům snadno dostupný prostředek ke zlepšení výuky.

3.2 Požadavky na aplikaci

Požadavky na aplikaci nám poskytují rámcový přehled o budoucích vlastnostech aplikace. Hlavním úkolem požadavků je zachytit, jakou funkčnost bude informační systém poskytovat. Zpřesňují představu o aplikaci jak na straně zákazníka, tak na straně dodavatele. To je důležité pro specifikování rozsahu implementovaného systému a zabraňuje pozdějším sporům o funkčnosti aplikace mezi oběma stranami. Požadavek by měl odpovídat na otázku, co by měl systém zvládnout, nikoliv jak. Existuje několik druhů požadavků, podle [8] např. tyto:

- **Funkční** - co bude systém umět, např. přihlásit se, odhlásit se
- **Nefunkční** - vlastnost systému, např. bezpečnost, výkon, dostupnost
- **Na rozhraní** - uživatelské, softwarové, hardwarové, komunikační, . . .
- Další požadavky - legislativní, vícejazyčnost, . . .

3. ANALÝZA

Sběru požadavků je v softwarovém inženýrství přikládána velká důležitost, protože je hlavním podkladem pro vyhodnocování rozsahu aplikace, finanční a časové náročnosti. Podle několika studií jsou chyby při sběru požadavků častým důvodem pro neúspěch projektu [9].

Přestože jazykem UML není specifikováno, jak bychom měli požadavky zapisovat, autoři [10] doporučují následující formát zápisu:

`<id> <název systému> bude <popis funkce>`

Důležitá je i forma požadavků, všechno přesně a jasně specifikovat, aby se předešlo pozdějším sporům o funkčnost mezi zákazníkem a dodavatelem. Např. v tomto textu nalezneme několik užitečných rad, co by měl popis každého požadavku obsahovat [11].

3.2.1 Funkční požadavky

Seznam funkčních požadavků.

F01 Založení hry

Aplikace bude umožňovat vytvářet hry. Každé hry se bude moci účastnit až šest různých hráčů. Všechny hry budou od sebe oddělené a nesmí se jakkoliv ovlivňovat. Aplikace musí generovat unikátní přihlašovací údaje, pomocí kterých se budou studenti přihlašovat. Jeden uživatel může založit i více her.

F02 Registrace do hry

Aplikace musí umožňovat studentům přihlásit se a registrovat se do hry. Studenti využijí přihlašovací údaje, které jsou nagenеровány při založení hry a které získají od vyučujícího.

F03 Přihlášení

Aplikace bude umožňovat přihlášení. Pro přihlášení je nutné zadat správné uživatelské jméno a heslo.

F04 Odhlášení se

Aplikace bude umožňovat přihlášenému uživateli se odhlásit.

F05 Editace jména

Aplikace umožní studentům změnit si jméno, které se bude zobrazovat spolu s výsledky.

F06 Podání nabídky

Aplikace bude umožňovat uživatelům přihlášeným jako student podávat nabídku na trh.

F07 Sloučení studentů do týmu

Hráči se mají v pokročilých částech hry spojit a vytvořit tým. Přesnou dobu, kdy k tomu dojde, určí učitel a ne aplikace.

F08 Vyhodnocení kola

Uživatelům, kteří budou přihlášení jako učitel, aplikace umožní ukončit jednotlivá kola ve hře. Aplikace nesmí sama od sebe vyhodnotit kola a musí vždy čekat na akci vyučujícího. Teprve pak je kolo vyhodnoceno.

F09 Zobrazení výsledku kola

Aplikace bude zobrazovat výsledky kol ve formě sloupcového grafu.

F10 Zobrazení výsledku hry

Aplikace bude zobrazovat průměrnou cenu v jednotlivých kolech ve formě liniového grafu.

3.2.2 Nefunkční požadavky

Seznam nefunkční požadavků.

N01 Webová dostupnost

Aplikace bude dostupná z webového prostředí pomocí prohlíže Google Chrome 41 [12], Mozilla Firefox 37 [13] a Internet Explorer 11 [14] a jejich novější verze.

N02 Responzivní design

Aplikace bude muset podporovat zobrazení na mobilních platformách a to zejména na mobilních verzích Google Chrome v systému Android a Safari v systému iOS. Podpora je garantována frameworkem Bootstrap. Ačkoliv mobilní platforma Windows není oficiálně podporována, přesto studenti s touto platformou se pravděpodobně budou moci účastnit bez omezení.

N03 Dvojjazyčnost

Aplikace bude lokalizována do českého a anglického jazyka. Tyto jazykové mutace musí být v systému jasně odlišeny a nesmí se zobrazovat stránky ve dvou různých jazycích. Výchozím jazykem bude čeština.

3.3 Model případů užití

Modelování případu užití je dalším doplňkovým způsobem zachycení požadavků na systém. Výstupem modelování případů užití by měly být následující komponenty [10]:

- **Účastníci** - množina rolí uživatelů, kteří budou využívat systém. Účastníkem může být také jiná aplikace, která s vyvíjenou aplikací komunikuje nebo spolupracuje. V případě, že systém spouští některé operace závislé na čase např: každý den o půlnoci, může být účastníkem také čas.
- **Případy užití** - činnosti, které mohou uživatelé systému vykonávat.

3. ANALÝZA

- **Relace** - vztahy mezi účastníky a případy užití.
- **Hranice systému** - ohraničení kolem případů užití, které znázorňuje hranice funkčnosti systému. Účastníci jsou vždy znázorněni až za hranicí systému.

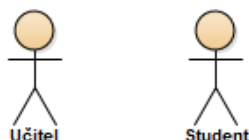
3.3.1 Účastníci

Učitel

Učitel bude moci zakládat a kontrolovat hru. Bez jeho účasti nelze hru spustit ani dohrát.

Student

Student se bude moci registrovat do hry a změnit si svoje jméno. Dále během hry bude moci podávat nabídku na trh. Každý student se účastní pouze jedné hry.



Obrázek 3.1: Uživatelé aplikace

3.3.2 Případy užití

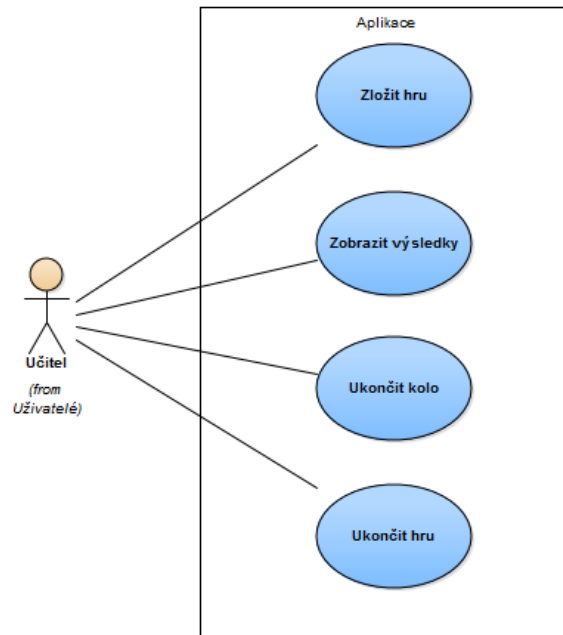
Účelem případu užití je popsat část souvislého chování systému bez odhalení vnitřní implementace. Pro přehlednost byly případy užití rozděleny do několika skupin.

3.3.2.1 Příklad užití Učitel

Balíček případů užití Učitel je vyobrazen na obrázku 3.2.

UC01 Založit hru

1. Příklad užití začíná, když učitel chce založit hru.
2. Uživatel vybere možnost Založit hru.
3. Aplikace zobrazí pole, kam učitel napíše kolik chce založit her a stiskne tlačítko Založit.
4. Aplikace zobrazí vygenerované přihlašovací údaje, které učitel sdělí žákům. Potom ho aplikace přesměruje na přihlašovací stránku.
5. Uživatel se přihlásí vyplněním formuláře. Ten ho pak přesměruje na stránku, kde může řídit založené hry.



Obrázek 3.2: Příklad užití Učitel

UC02 Konec kola

Konec každého kola bude řízen ručně učitelem. Aplikace nebude upozorňovat učitele na případné nepodané nabídky žáků na konci kola.

UC03 Sloučit hráče do týmu

Učitel bude moci sloučit studenty do týmů. Tento úkon bude muset být spuštěn ručně.

UC04 Ukončit hru

Aplikace ukončí hru na pokyn učitele.

UC05 Zobrazit výsledek

Aplikace zobrazí výsledek ve formě grafu.

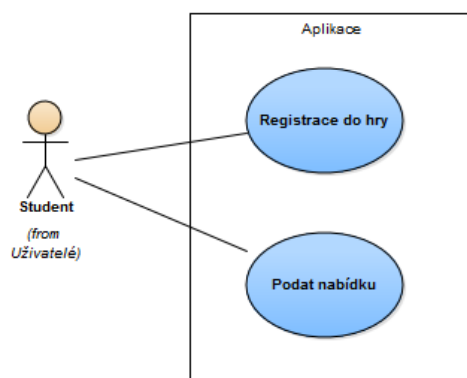
3.3.2.2 Příklad užití Student

Balíček případů užití Student je vyobrazen na obrázku 3.3.

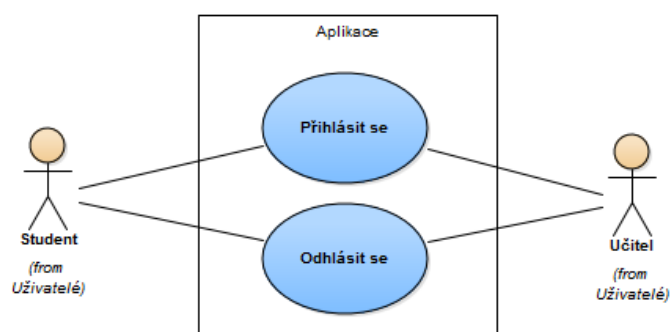
UC06 Registrovat se do hry

1. Příklad užití začíná, když se student chce přihlásit do hry a má platné přihlašovací údaje od učitele.
2. Student vybere možnost registrovat se do hry.
3. Aplikace zobrazí přihlašovací formulář, kam student zadá své údaje.

3. ANALÝZA



Obrázek 3.3: Příklad užití Student



Obrázek 3.4: Příklad užití Obecné

4. Student zadá své jméno do políčka, které se zobrazí.

UC07 Změnit svoje jméno

Aplikace umožní studentům měnit své jméno, které se bude zobrazovat u výsledků. Pokud student nezadá žádné jméno, tak mu bude automaticky vygenerováno na začátku hry.

UC08 Podat nabídku

1. Aplikace zobrazí nákladovou tabulku.
2. Student vyplní počet kusů, které chce dodat na trh, a jejich cenu do formuláře a stiskne tlačítko Podat nabídku.
3. Student počká, dokud učitel neukončí kolo.

3.3.2.3 Příklad užití Obecné

Balíček případů užití Obecné je vyobrazen na obrázku 3.4.

UC09 Přihlásit se

1. Případ užití začíná, když uživatel chce začít pracovat s aplikací a zná správné přihlašovací údaje.
2. Uživatel vybere možnost Přihlásit se.
3. Aplikace zobrazí formulář přihlášení.
4. Uživatel vyplní formulář a po stisknutí tlačítka Přihlásit se může začít využívat aplikaci.

UC10 Odhlásit se

1. Případ užití začíná, když uživatel chce ukončit práci s aplikací.
2. Uživatel vybere možnost Odhlásit se.
3. Aplikace uživatele odhlásí a zobrazí formulář přihlášení.

UC11 Změnit jazyk

1. Případ užití začíná, když uživatel chce změnit jazyk.
2. Uživatel klikne na možnost English, pokud jsou stránky v češtině a chce změnit jazyk na angličtinu, nebo Česky, pokud je aplikace v angličtině a uživatel chce češtinu.
3. Aplikace změní jazyk.

3.4 Analytický doménový model

Doménový model slouží jako náhled na daný problém z vyšší úrovně abstrakce. Skládá se z doménových tříd a asociací mezi nimi. Diagram zachycuje pouze nejdůležitější atributy a metody, implementační detaily zakresleny nejsou. Název doménové třídy by měl co nejpřesněji odrazet její účel. Doménový model aplikace je znázorněn na obrázku 3.5.

Z analytického doménového modelu vyplývá, že hru řídí právě jeden administrátor a každá hra obsahuje jednu implementaci algoritmu simulujícího trh. Mnohem zajímavější je část s třídami Player a Team, kdy se v jedné hře může vyskytnout až šest různých týmů. To odpovídá požadavku na šest různých studentů ve hře. V jednom týmu může být různý počet hráčů, od jednoho do šesti, a to odpovídá požadavku na sloučení hráčů do jednoho týmu. Model umožňuje v jedné hře až 36 různých hráčů, ale to v reálné aplikaci nebude možné, protože každá hra začíná s maximálním počtem šesti hráči a nebude povoleno hráče přidávat.

Game

Doménová třída reprezentující hru.

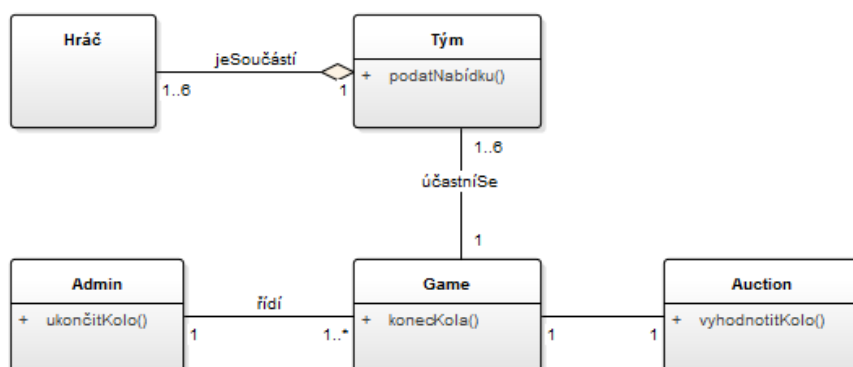
Auction

Doménová třída reprezentující implementaci trhu.

Admin

Doménová třída reprezentující uživatele v roli administrátora.

3. ANALÝZA



Obrázek 3.5: Analytický doménový model

Player

Doménová třída reprezentující uživatele v roli hráče. Každý hráč je součástí nějakého týmu.

Team

Doménová třída reprezentující tým, který se účastní obchodování na trhu.

Návrh

4.1 Použité technologie

Aplikace používá open-source technologie založené na platformě Java. Jejich největší výhodou je pořizovací cena, nicméně u open-source technologií musíme počítat se složitější konfigurací a neuceleností jednotlivých řešení.

4.1.1 Java

Java [15] je velmi populární objektově orientovaný programovací jazyk. Původně tento jazyk vyvíjela firma Sun Microsystems a poprvé byl představen 23. května 1995. V současnosti je spravován firmou Oracle Corporation.

Syntaxe je inspirována jazyky C a C++. Programy napsané v Javě se kompilují do tzv. *byte code*, který lze spustit všude, kde je dostupný JRE (Java Runtime Environment) [16]. V jazyce je implementována automatická správa paměti, tzv. *garbage collector*, díky které se programátoři nemusí starat o alokování a uvolňování paměti - programy jsou kvůli tomu více robustní a odolnější vůči častým chybám se správou paměti.

Jazyk Java se dělí na několik edic [17]:

- **Java ME** - edice vyvinutá pro mobilní a vestavěné aplikace
- **Java SE** - edice určená pro vývoj desktopových aplikací
- **Java EE** - nejrobustnější edice určená pro vývoj podnikových systémů

4.1.2 Spring Framework

Při programování větších projektů je většinou nežádoucí vytvářet těsné vazby mezi objekty. Tyto vazby vadí zejména při udržování projektů, kdy znepřehledňují strukturu, způsobují problémy při výměně implementace za jinou a celkově znesnadňují vývoj enterprise aplikací. Spring Framework [18] pomáhá s řešením tohoto problému.

Spring je lightweight kontejner, který umožňuje centralizované řízení objektů, tzv. Spring bean. Toto centralizované řízení závislostí umožňuje oddělit a snadno nahradit různé implementace a také ulehčuje testování. Za Spring bean považujeme jakýkoliv objekt kontrolovaný Springem. Ten se stará o jejich vznik a zánik a poskytuje je ostatním objektům.

Důvodem využití této technologie je injekce závislostí, tzv. *Dependency Injection*, známá též jako Inversion of Control. Místo toho, aby třídy sami o sobě vytvářely objekty, které potřebuje, Spring Framework zařídí, aby ji byla nějaká implementace poskytnuta. S takto dosazeným objektem třída komunikuje přes rozhraní společné všem pro všechny implementace.

Závislosti mezi Spring beanly jsou definovány v samostatném XML souboru. Protože závislosti nejsou popsány přímo ve zdrojovém kódu, aplikace se rozpadne do nezávislých částí, které můžeme znovu využít nebo testovat. U závislosti lze také určovat životnost jednotlivých beanů, např. zda se jedná o singleton (objekt společný pro všechny třídy v celém projektu) nebo prototype (každé třídě, která tuto závislost potřebuje, bude vyrobena vlastní).

4.1.3 Hibernate

Hibernate [19] je framework pro objektově-relační mapování v jazyce Java. Mapuje objekty POJO (Plain Old Java Object) v Javě do relační databáze, ale dokáže i vytvářet objekty z uložených dat. Mapování můžeme provádět v XML souboru nebo pomocí anotací. Další výhodou je, že vývojáře odstíní od databáze. Toho pak nemusí zajímat, jestli jsou data uložena v databázi MySQL, Oracle nebo případně v databázi jiného výrobce. Vše, co je potřeba udělat při výměně databáze, je změnit dialekt v konfiguraci Hibernate.

Objektově-relační mapování nám umožňuje přistupovat k datům přímo v programovacím jazyce ve formě obvyklých objektů. V Javě pro tento způsob existuje standard zvaný Java Persistence API a Hibernate ho implementuje.

4.1.4 MySQL

Aplikace využívá open-source databázový systém MySQL [20]. Vývoj byl započat švédskou firmou MySQL AB, ale v současné době je spravován korporací Oracle Corporation. K datům lze přistupovat přímo pomocí SQL dotazů, ale je možné využít i objektově-relační mapování.

4.1.5 Apache Tomcat

Apache Tomcat [21] je lightweight servlet kontejner vyvíjený pod licencí Apache Licence version 2. Mezi jednu z jeho nevýhod oproti klasickým aplikačním serverům patří absence Enterprise Java Beans kontejneru. Tato skutečnost je při použití Springu zcela zanedbatelná. Mezi výhody patří nízké nároky na webhosting, díky které je Apache Tomcat často nabízen u zdarma dostupných hostingů.

4.1.6 JavaServer Pages

JavaServer Pages [22], zkráceně JSP, je technologie pro vytváření šablon, které jsou ve většině případů použity ke generování HTML stránek.

V JSP lze využívat javovský kód, tzv. *Scriptlets*, ke generování obsahu, ale od této techniky se v současné době upouští, protože později byla do JSP přidána technologie JSP Standard Tag Library (zkráceně JSTL), která scriptlety nahrazuje. JSTL je set tagů pro zjednodušení vývoje JSP stránek a které implementují běžné operace. Pro tvoření stránek je také možné využít Expression Language, který poskytuje důležité mechanismy pro komunikaci s aplikační logikou. V současné době je doporučováno využít JSTL a EL namísto scriptletů.

4.1.7 Respozivní web

Respozivní webový design používá CSS a HTML ke přizpůsobení, skrytí, zmenšení, zvětšení nebo přesunutí obsahu na stránce tak, aby celek vypadal dobře pro jakoukoliv velikost obrazovky. Výsledkem je, že stránky jsou optimalizovány i pro mobilní přístroje, popř. pro tablety.

Framework pro tvorbu responzivního webu byl zvolen Bootstrap [23], volně stažitelná sada nástrojů pro tvorbu responzivního webu. Balíček obsahuje návrhářské šablony založené na HTML, CSS a JavaScriptu, pomoci kterých lze vytvářet hezké stránky bez nutnosti vzhled komponent a chování jakkoliv implementovat. Velikou výhodou tohoto řešení je responzivní vlastnost všech stránek, tj. prvky na stránce se svou velikostí a pozicí automaticky přizpůsobují velikosti obrazky. Stejná stránka se jinak zobrazí pro desktopový prohlížeč a pro mobilní zařízení. Vše probíhá samo od sebe bez nutnosti zásahu programátora.

Bootstrap je vydán pod MIT licenci. Tato licence vznikla na půdě Massachusetts Institute of Technology (MIT) a zajišťuje volnou šířitelnost softwaru.

Existují i další technologie pro tvorbu responzivního designu, například Foundation [24]. I tento framework lze využít při tvorbě této aplikace.

4.1.8 Vykreslování grafů

Pro vykreslování grafů byla použita knihovna Chartist.js [25]. Tato knihovna k vykreslování grafů používá JavaScript a CSS styly. Důvodem výběru této knihovny byla především vizuální podobnost s Bootstrapem.

Dále samozřejmě existují i jiné knihovny pro vykreslování grafů, zmiňme například knihovny Flot [26], Chart.js [27] nebo Google Charts [28].

4.1.9 Maven

Apache Maven [29] je nástroj pro správu, řízení a automatizaci buildů aplikací. Ačkoliv je možné použít tento nástroj pro projekty psané v různých

programovacích jazycích, podporován je převážně jazyk Java.

Maven byl vytvořen jako nástroj pro zjednodušení buildů pro projekt Jakarta Turbine. Hlavním impulzem pro vznik byla snaha o standardizaci a znovupoužitelnost buildovacích skriptů.

4.1.10 Git

Git [30] je nástroj pro správu verzování zdrojového kódu. Ačkoliv ho lze použít i k verzování binárních souborů, není k tomu stavěn a neměl by být k tomu používán. Autorem Gitu je Linus Torvalds.

4.2 Architektura aplikace

Architektura zachycuje realizace nefunkčních požadavků. Při návrhu bychom měli dbát zvýšené opatrnosti, protože chyby při špatném návrhu se mohou v budoucnu velice komplikovat a prodražit údržbu aplikace [31].

Architektura aplikace vychází z analytického návrhového modelu. Odlišnost spočívá v rozdělení aplikace na dvě samostatné části – na front-end a back-end. Výhodou tohoto řešení je zpřehlednění zdrojového kódu a rozdělení aplikace na dva nezávislé celky.

Frontend

Frontend je webová aplikace, která obsahuje uživatelské rozhraní dostupné z internetu. Kostrou této části je Spring MVC - je zde naimplementované controller třídy a webové stránky, které jsou dynamicky generované. Frontend má vazby na backend, bez kterého nemůže fungovat.

Backend

Backend je dále rozdělen na několik modulů. Mezi nejdůležitější patří modul, který nám umožňuje přistupovat k datům z datového úložiště. V dalším modulu je naimplementovaná třída, která vyhodnocuje nabídky hráčů.

4.3 Databázový model

Úkolem tohoto diagramu je zachytit strukturu uložení dat v databázi. Model vychází z návrhového modelu tříd z předcházející kapitoly. Z databázového modelu byl vytvořen DDL (Data Definition Language) skript, který obsahuje definici tabulek v jazyce SQL. Pomocí tohoto skriptu byly vytvořeny potřebné tabulky v databázi. Tvorbu DDL skriptu lze automatizovat použitím vhodného CASE nástroje.

4.4 Návrhový model tříd

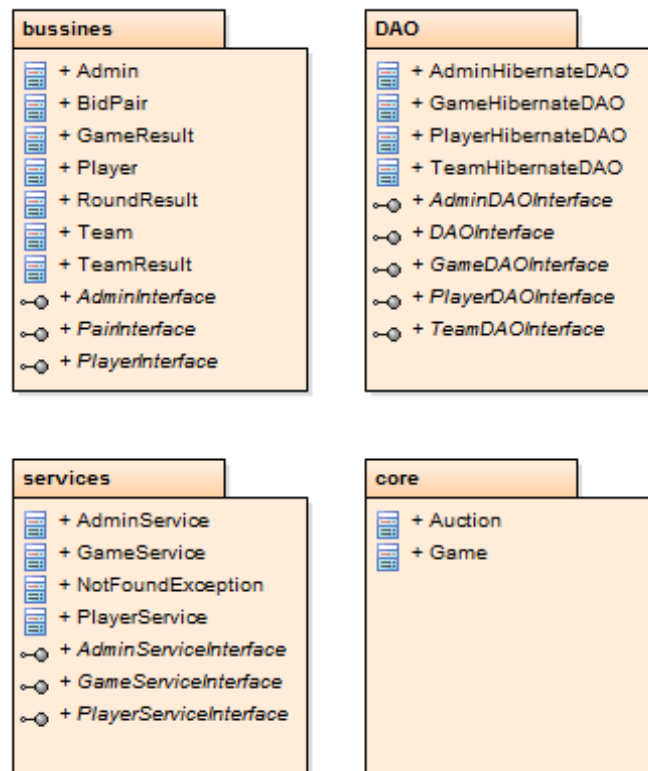
Základ návrhového modelu tříd vychází z analytického doménového modelu, popsaného v předcházející kapitole. Tento model je již detailnější, obsahuje například typy atributů včetně jejich viditelnosti. Dále v návrhovém modelu přiřazujeme třídám jednotlivé metody včetně vstupních a výstupních parametrů a tím definujeme zodpovědnost tříd.

Pomocí vhodného CASE (Computer-Aid Software Engineering) nástroje např.: Enterprise Architect můžeme pro namodelované třídy vygenerovat předlohy, které usnadní práci programátorům. Proto je vhodné tento model modelovat co nejdětalněji.

4.5 Popis balíčků

Třídy byly rozděleny do několika balíčků tak, aby výsledný zdrojový kód byl co nejprehlednější.

4.5.1 Backend



Obrázek 4.1: Návrhový model tříd - backend

4.5.1.1 Balíček core

Zde jsou uloženy třídy, které implementují core funkcionalitu. Třída Auction implementuje algoritmus, podle kterého se řídí trh. Pomocí třída Game se řídí celá hra.

4.5.1.2 Balíček dao

Tento balíček definuje rozhraní a dále obsahuje třídy implementující toto rozhraní, pomocí kterých aplikace přistupuje k datům v databázi. Tyto třídy jsou navrženy podle návrhového vzoru Data Access Object (DAO).

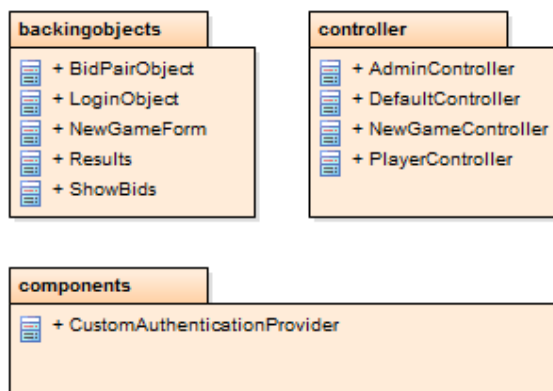
4.5.1.3 Balíček bussines

Balíček bussines obsahuje třídy, které reprezentují doménový model aplikace. Tyto třídy jsou pomocí tříd z balíčku dao mapovány do tabulek databáze.

4.5.1.4 Balíček services

Třídy uložené v tomto balíčku zajišťují logiku aplikace. Třídní metody jsou volané z frontendu a vykonávají akce, které zaručují funkčnost.

4.5.2 Frontend



Obrázek 4.2: Návrhový model tříd - backend

4.5.2.1 Balíček controller

Tento balíček obsahuje třídy, které podle MVC zastupují controllery. Úkolem těchto tříd je obsluhovat požadavky uživatele a předávat je backendové části. Jejich další úlohou je aktualizovat model a udržovat aktuální data ve view.

4.5.2.2 Balíček components

Třída v tomto balíčku se stará o autentizaci a autorizaci v rámci celé aplikace.

4.5.2.3 Balíček backingobjects

Tento balíček obsahuje několik pomocných tříd a validaci vstupních dat. Některé třídy jsou uložena data, které se prezentují uživateli.

4.6 Autentizace a autorizace

Aplikace musí zajistit, aby se oddělily role. To znamená, že uživatel, který je přihlášen jako administrátor, nesmí podávat nabídky a naopak uživatel s právy účastníka aukce nesmí ukončit kola. Dále aplikace musí oddělit jednotlivé hráče tak, aby se nestalo, že by jeden mohl hrát za druhého.

Těchto požadavků je dosaženo pomocí modulu Spring Security.

Realizace

Jak již bylo zmíněno, tak vývoj aplikace byl rozdělen do dvou logických celků.

První část, backend, zajišťuje aplikační logiku a ukládání dat do databáze. Nachází se zde důležitá třída Auction, která implementuje rozhodovací algoritmus trhu.

Druhá část, frontend, se stará o obsluhu požadavků uživatele a o prezentaci dat. Zde jsou uloženy šablony webových stránek aplikace. Pro vývoj bylo použito vývojové prostředí NetBeans IDE 8.0.2 [32].

5.1 Implementace algoritmu trhu

Třída, která simuluje chování kupců, se jmenuje Auction a nachází se v balíčku s názvem core. Tato třída je nezávislá na okolních třídách, protože obsahuje veškeré funkce, na které se spoléhá. Třída Game, která se nachází ve stejném balíčku, pak třídu Auction řídí – volá metodu makeDecision.

Třída obsahuje statický inicializační blok, který inicializuje pole, kam se vloží poptávková funkce. Pole musí být static a final, ale to nevádí, protože je sdílené pro všechny instance.

Nejdůležitější metody jsou tyto:

```
public void makeOffer();  
public RoundResult makeDecision();
```

- Metodu makeOffer využívají hráči, kteří chtějí podat nabídku na trh.
- Metodu makeDecision simuluje rozhodování kupujících. Jakmile je spuštěna, tak již nelze podat nabídku. Výsledky kola jsou uloženy ve třídě RoundResult.

5.1.1 Implementace metody makeOffer

```
public void makeOffer(Integer id, BidPair pair) {  
    bids.put(id, pair);  
    roundResult.get(id).setPrice(pair.getPrice());  
    roundResult.get(id).setBidAmount(pair.getAmount());  
}
```

Ukázka kódu 5.1: Implementace metody makeOffer.

Metoda makeOffer je velice jednoduchá. Prakticky jenom uloží nabídku ve formě objektu BidPair do instanční proměnné. Nabídka se také uloží do výsledku kola, které je reprezentováno proměnnou roundResult. To se dělá z důvodu, aby se vědělo, jaké množství za jakou cenu studenti nabízeli.

5.1.2 Implementace metody makeDecision

Algorismus nejprve nalezne nejnižší nabídku a všechny kupce, kteří za nalezenou cenu na trhu nabízejí. Pak se vyhodnotí, kolik toho za danou nejnižší cenu trh koupí – tato informace se uloží do proměnné amount. Dále se mezi kupci s nejnižší cenou nalezne nejmenší nabízený objem zboží a ten se uloží do proměnné lowestAmount. Tato informace bude využita při vyhodnocování, jakým způsobem aplikace nakoupí zboží.

Nyní se algoritmus snaží koupit množství uložené v lowestAmount od všech kupců s nejnižší cenou. Zde mohou nastat situace:

- Množství, které trh nakoupí (hodnota získaná z poptávkové funkce) je větší než objem uložený v proměnné lowestAmount násobený počtem všech, kteří nabízejí za stejnou cenu. Algorismus může toto množství nakoupit ode všech a následně aktualizuje nabídky na trhu (odečte prodané množství a vyřadí nabídky, které prodali vše).
- Nelze nakoupit od všech nejmenší nabízené množství. Program musí určit největší množství, které lze koupit od všech. Pokud nelze od všech nakoupit stejné množství, tak algoritmus náhodně upřednostní některé prodávající.

Vždy, když se provede transakce, tak aplikace navýší počet koupeného zboží v proměnné bought a výsledek se uloží do objektu RoundResult.

Výše popsaný proces se provádí stále dokola, dokud je trh ochoten nakupovat. Poté se veškeré atributy vynulují a třída je připravena znovu přijímat nabídky od hráčů. V ukázce lze vidět rozdělení algoritmu na dvě disjunktní možnosti, které odpovídají dvou situacím popsaným výše.

Poslední uvedená podmínka testuje, jestli trh může ještě nakoupit poslední zbytek. Ten bude nakoupen od náhodně vybraného prodávajícího. Pro získání náhodného kupce byla použita třída Random z balíku java.util.Random.

```

public RoundResult makeDecision() {
    int bought = 0;
    while (true) {
        ...
        if (lowestAmount * sellers.size() <= (amount - bought)) {
            //buy lowestAmount from all sellers with the same price
        }
        else {
            lowestAmount = (amount - bought) / sellers.size();
            //buy fair amount from all sellers
            if (bought < amount) {
                //buy rest from random seller
            }
        }
    }
}
    ...
}

```

Ukázka kódu 5.2: Ukázka implementace metody makeDecision. Ukázka je zkrácena.

5.2 Dependency Injection

Dependency Injection byl v aplikaci použit ke správě objektů a vytváření vazeb mezi nimi. Definovat Java beana lze dvěma způsoby a to buď anotacemi nebo v samostatném XML souboru. Obě metody jsou ekvivalentní a záleží na preferencích programátora.

Jednou z nejdůležitějších objektů, které Spring spravuje, je třída vytvářející spojení s databází. V ukázce 5.3 jsou definovány dvě třídy, které jsou potřeba pro databázi. První Spring bean, pojmenovaný dataSource, vytvoří spojení s databázovým serverem. Zde musí být uvedeny platné přihlašovací údaje, jinak se spojení neuskuteční. Tento bean je pak předán v definici druhého beanu, který inicializuje Hibernate třídu používanou v metodách pracujících s databází.

5.3 Spring Security

Modul Spring Security byl použit k rozlišení uživatelů v aplikaci. V aplikaci jsou definovány dvě role – jedna pro žáka a druhá pro učitele. Role slouží k oddělení práv studenta od práv učitele, který se stará o chod hry. Žádný uživatel nesmí mít obě role současně.

V ukázce 5.4 je uvedena část konfiguračního souboru. Takto nastavená aplikace se postará o to, aby každý, kdo si bude chtít zobrazit stránku začínající vzorem /player/, se bude muset přihlásit a prokázat, že má právo tyto stránky zobrazit.

5. REALIZACE

```
<!-- MySQL data source -->
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
    destroy-method="close">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url"
        value="jdbc:mysql://webdev.fit.cvut.cz:3306/kocanmi4" />
    <property name="username" value="username" />
    <property name="password" value="password" />
</bean>

<!-- Hibernate session factory -->
<bean id="sessionFactory"
    class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="annotatedClasses">
        <list>
            <!-- list all classes that are persisted -->
        </list>
    </property>
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">
                org.hibernate.dialect.MySQLDialect
            </prop>
        </props>
    </property>
</bean>
```

Ukázka kódu 5.3: Příklad konfigurace Java Spring, která je použita k vytvoření spojení s databází. Zde uvedená adresa databáze je použita pouze jako ukázka.

Obdobně je nastavená část pro učitele, s výjimkou stránky dostupné na `/admin/newGame/` – zde se zobrazuje formulář pro vytvoření nových her. Formulář je přístupný bez nutnosti přihlášení, protože v době, kdy učitel vytváří nové hry, ještě nezná své přihlašovací údaje, a proto by se nemohl přihlásit.

```
<http auto-config="true" use-expressions="true">
    <intercept-url pattern="/admin/newGame/"
        access="hasRole('ROLE_ANONYMOUS')" />
    <intercept-url pattern="/admin/**"
        access="hasRole('ROLE_ADMIN')" />
    <intercept-url pattern="/player/**"
        access="hasRole('ROLE_PLAYER')" />
</http>
```

Ukázka kódu 5.4: Ukázka z konfiguračního souboru XML Spring Security.

5.4 Validace

Všechny informace, které uživatel zadá do formuláře je třeba zkontrolovat, aby se zamezilo vložení dat, které neodpovídají požadovanému formátu nebo mohou způsobit případný pád aplikace.

Validaci dat v Springu lze provést několika způsoby. Jedním z nich je použití anotací, konkrétně anotace z balíku `javax.validation.constraints`, též známé pod označením JSR-303. Tento způsob je vhodný pro jednoduchou validaci, například pro kontrolu jestli uživatel nenechal prázdné políčky nebo aby nebyl vložen záporný, či jinak nesmyslný věk.

Pokud je k validaci potřeba vytvářet složitější aplikační logiku, je vhodnější implementovat rozhraní `Validator`.

Například pro validaci čísla, které nesmí být prázdné a záporné, lze anotace využít tímto způsobem 5.5.

```
//form object
@NotNull
@Min(0)
private Integer number;

//controller
@RequestMapping(value = "/newGame", method = RequestMethod.POST)
public String postForm(@Valid @ModelAttribute("formObject") NewGameForm fo,
    BindingResult result) {
    if (result.hasErrors()) {
        // do something
    } else {
        // do something else
    }
}
```

Ukázka kódu 5.5: Anotace zobrazené v ukázce validují proměnnou `number`, kam se uložila data z formuláře. V druhé části, která zobrazuje kus implementace kontroleru, anotace `Valid` spouští kontrolu dat v objektu `NewGameForm`.

5.5 Responzivní web

K využití `Bootstrap Frameworku` je potřeba zajistit několik věcí:

- Předně zahrnout potřebné soubory – do hlavičky každé HTML stránky bylo potřeba zahrnout zkompileované soubory `Bootstrapu` a `JavaScriptu`. Toho šlo docílit několika způsoby, ten nejjednodušší je využít služby `CDN` (`Content Delivery Network`) jako je zobrazeno v ukázce. Další způsob byl nutné soubory stáhnout a v odkazu uvést cestu k souboru.

5. REALIZACE

- Hlavní obsah je potřeba vložit do kontejneru, který jej obaluje. Tento kontajner je potřeba označit stylem container (jak je vidět v ukázce) nebo container-fluid.

```
<head>
  <!-- Latest compiled and minified CSS -->
  <link rel="stylesheet"
        href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/
              bootstrap.min.css">
  <!-- jQuery library -->
  <script
        src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/
              jquery.min.js">
  </script>
  <!-- Latest compiled JavaScript -->
  <script
        src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/
              bootstrap.min.js">
  </script>
</head>
<body>
  <div class="container">
    ...
  </div>
</body>
```

Ukázka kódu 5.6: Použití frameworku Bootstrap při tvorbě HTML stránek.

Jakýkoliv element, který se nachází v kontajneru, automaticky podědí styl od Bootstrapu. Některé HTML prvky mohou dále definovat svůj vzhled - pro prvek button Bootstrap vytvořil několik různých schémat. Další příklad je úzký barevný pás, který má informativní charakter - pro varovnou zprávu je definován červeně, pro pozitivní zprávu je zelený, atd. Jediné, co tyto prvky musí specifikovat, aby byly takto nastýlovány, je dědit od správné css třídy. Programátor nemusí tuto explicitně definovat, vše již bylo vytvořeno ve frameworku.

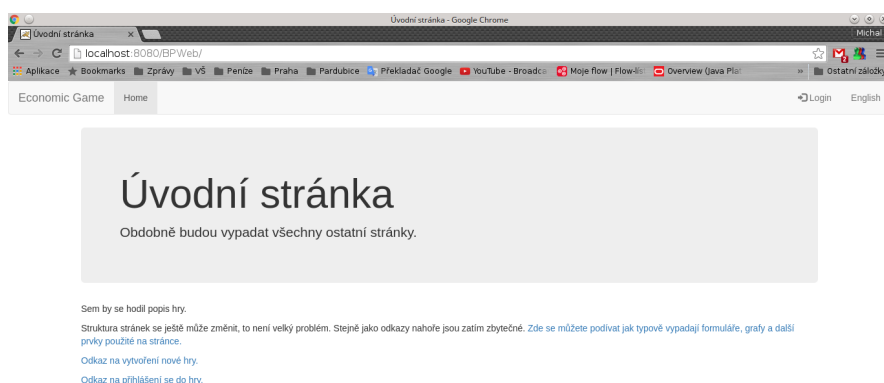
Bootstrap definuje i pokročilé JavaScriptové komponenty ve formě jQuery pluginů. Zmíňme například vyskakující seznamy, rozbalující políčka s nabídkou nebo několik různých prvků z nichž je zobrazen pouze jeden. Všechny zmíněné komponenty by normálně programátor musel pracně vytvořit sám, ale díky Bootstrapu pouze využije hotové řešení. Tento postup značně urychluje vývoj webových stránek.

Ukázka aplikace

V této kapitole je uveřejněno několik ukázek aplikace. Zobrazené ukázky jsou z testovacího provozu a výsledná vizuální stránka se ještě může změnit, zejména co se obsahu týče.

6.1 Úvodní stránka

Na úvodní stránce 6.1 bude vysvětleno k čemu stránky slouží. Dále zde bude odkaz na vytvoření nové hry pro učitele a odkaz na přihlášení do hry pro studenty. V horní části se nachází hlavička, kde je základní navigace portálem, plus možnost přihlásit se nebo odhlásit se a nakonec možnost přepnout jazyk do angličtiny. Tato hlavička je dostupná na každé stránce a lehce se mění na základě uživatelského nastavení aplikace.

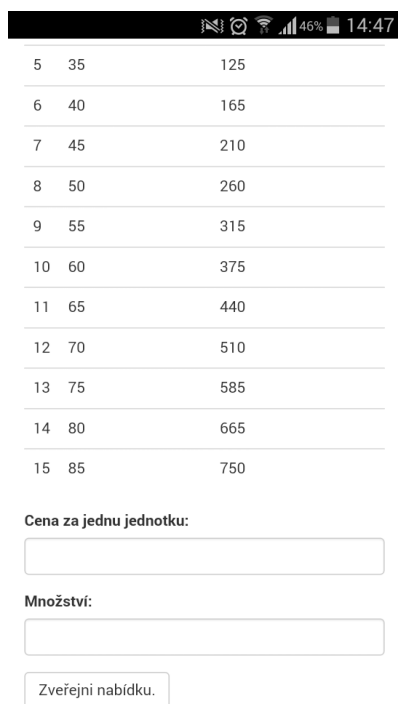


Obrázek 6.1: Úvodní stránka aplikace.

6.2 Stránka pro studenta

Poté, co student klikne na odkaz na registraci do hry na úvodní straně a úspěšně ji dokončí, tak mu aplikace zobrazí nákladovou funkci a formulář k podání nabídky. Tabulka a formulář jsou optimalizovány pro klasické prohlížeče na osobních počítačích, ale i pro mobilní přístroje 6.2.

Po odeslání nabídky, student musí počkat na ostatní kolegy a na učitele, který musí ukončit kolo ručně.



The screenshot shows a mobile application interface. At the top, there is a status bar with icons for signal strength, Wi-Fi, battery (46%), and time (14:47). Below the status bar is a table with three columns: a number (5 to 15), a unit price (35 to 85), and a total price (125 to 750). The table is followed by a form with two input fields: 'Cena za jednu jednotku:' and 'Množství:'. Below the input fields is a button labeled 'Zveřejni nabídku.'

5	35	125
6	40	165
7	45	210
8	50	260
9	55	315
10	60	375
11	65	440
12	70	510
13	75	585
14	80	665
15	85	750

Cena za jednu jednotku:

Množství:

Zveřejni nabídku.

Obrázek 6.2: Stránka pro studenta zobrazená na mobilním zařízení s Androidem.

6.3 Stránka pro učitele

Na stránce pro učitele se nachází prvky k ovládní experimentu. Jedním z požadavků na aplikaci je, aby učitel měl kontrolu nad průběhem aplikace. Jedná se o tlačítko pro konec kola, sloučení hráčů do skupin a konec hry.

Poté, co učitel ukončí kolo, zobrazí se nejprve graf všech nabídek studentů učiněných v kole a studenti mohou porovnat svoji nabídku s ostatními. Teprve na další stránce se objeví celý výsledek ve formě sloupcového grafu 6.3 a studenti se dozví, jak v daném kole dopadli.

Liniový graf s průměrnou cenou v jednotlivých kolech se zobrazí po ukončení celého experimentu.



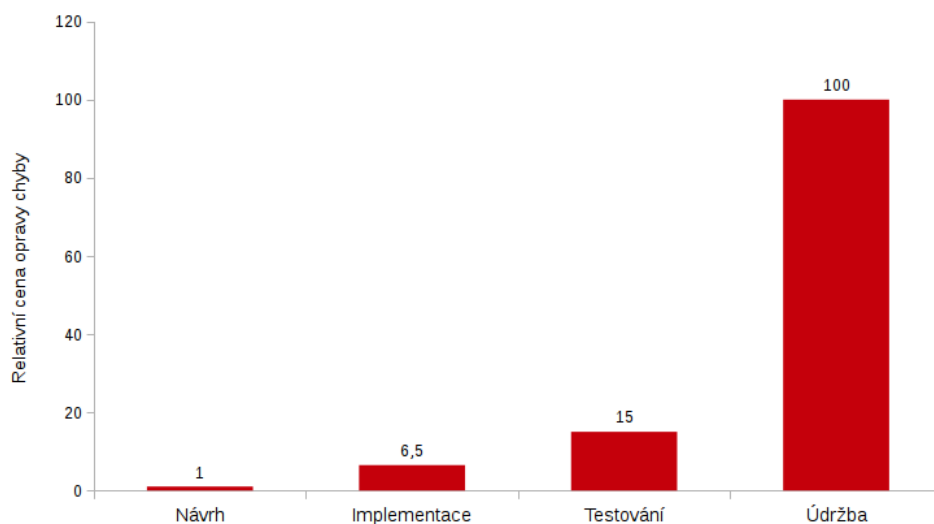
Obrázek 6.3: Stránka pro učitele zobrazující výsledky kola.

Testování

Testování je jednou ze základních oborů quality assurance a softwarového inženýrství. Protože lidé nejsou neomylní a produkují chyby, které ovlivňují kvalitu softwaru, vzniklo testování. Chyby nemusí být pouze v implementaci, ale i v návrhu, analýze nebo samotných testech. Protože cena opravy chyb roste s dobou, kdy jsou chyby neodhaleny, je vhodné chyby odhalovat co nejdříve. S každou opravenou chybou roste kvalita softwaru.

Testování má za úkol odhalovat chyby v zdrojovém kódu, které mají negativní vliv na kvalitu výsledného softwaru. Během relativně krátké doby vzniklo mnoho testovacích frameworků a přístupu k testování obecně. Obor zabývající se kontrolou kvality software nazýváme Quality Assurance (QA).

Hodnoty v grafu 7.1 převzaty z [9].



Obrázek 7.1: Relativní cena opravy chyby.

7.1 Unitové testování

Unitové (česky také jednotkové) testování je první úrovní ověřování kvality kódu. Krátké testy jsou zaměřené především na správné fungování metod a tříd. Tyto testy většinou píše autor kontrolovaného zdrojového kódu, proto se jedná o tzv. *White box* techniku.

7.1.1 JUnit

Framework pro unit testování jsou jazykově specifické. Protože je aplikace naprogramovaná v Javě, tak musí být otestována v JUnit frameworku [33]. Kolektivně se nazývají xUnit a mezi nejznámější patří již zmíněný JUnit, PHPUnit (jazyk PHP) nebo například JSUnit (jazyk JavaScript). První framework pro jednotkové testování byl vytvořen pro SmallTalk.

7.1.2 Mockito

Další technologie použitá v jednotkovém testování je tzv. mockování. Mockito [34] je framework pro vytváření mock objektů. Díky této technice můžeme oddělit testované třídy od jejich okolí.

Mock objektu nastavíme očekávané vlastnosti a návratové hodnoty. Dále pomocí tohoto frameworku můžeme verifikovat, jestli testovaný objekt volal správnou metodu se správnými parametry.

```
import java.util.List;
import org.junit.Test;
import static org.mockito.Mockito.*;

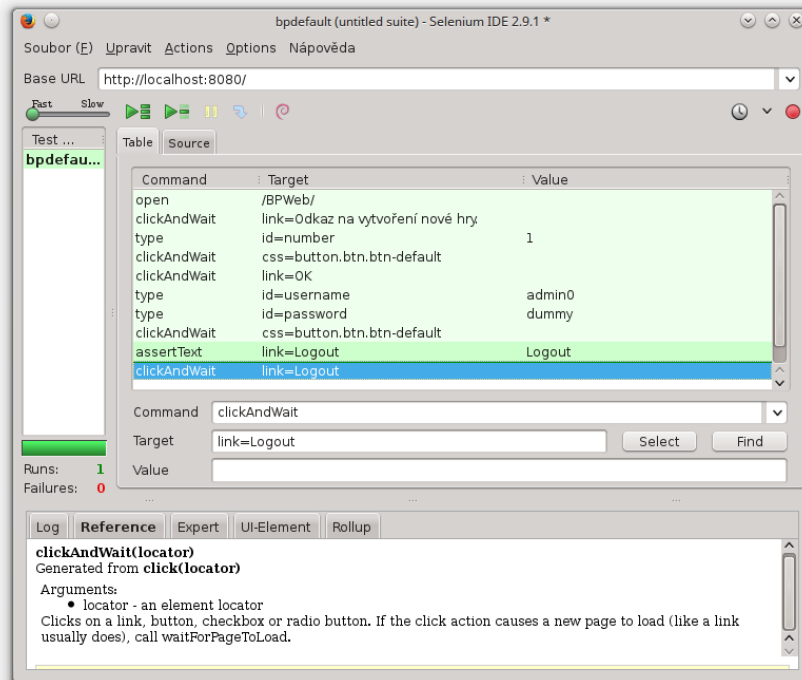
public class ListTest {

    @Test
    public void testAdd() {
        List mockedList = mock(List.class);
        mockedList.add("one");
        mockedList.clear();
        verify(mockedList).add("one");
        verify(mockedList).clear();
    }
}
```

Ukázka kódu 7.1: Ukázka testovací metody.

7.2 Funkční testování

Jedná se typ testování, při kterém nás zajímá veškerá funkčnost zkoumaného systému. Funkční testování je podmnožinou akceptačního testování, které vychází z akceptačních kritérií zákazníka.



Obrázek 7.2: Na obrázku lze vidět jednoduchý testovací příklad, který otestuje, že se vytvoří hra a uživatel se může přihlásit jako administrátor.

Při tomto druhu testování nás zajímá jak se aplikace chová a co dělá, ne jak to dělá. Uplatňuje se zde tzv. *Black box* technika, kdy se tester nestará o vnitřní strukturu implementace (tu většinou ani nezná), ale o rozhraní systému. Ve většině případů se testuje uživatelské rozhraní - GUI, nebo pokud chybí, tak se může testovat například příkazová řádka.

7.2.1 Selenium

Selenium [35] je nástroj pro automatické testování webového rozhraní a podporuje množství platform. Tento software se skládá z několika komponent, zejména ze Selenium IDE a z Selenium WebDriver API.

Selenium IDE lze doinstalovat pouze do webového prohlížeče Mozilla Firefox, jiné platformy bohužel nejsou podporovány. Uživatelské rozhraní je vidět v ukázce 7.2. V Selenium IDE bylo vytvořeno několik jednoduchých testů zaměřených na základní úkony aplikace, např. přihlášení nebo odhlášení.

Složitější testy byly udělány v jazyce Java za pomoci Selenium WebDriver, knihovny pro práci s webovými elementy. V těchto testech byly dodrženy návrhové vzory Page Object a Page Factory [36] pro testování webových stránek.

7. TESTOVÁNÍ

Page Object nám pomáhá s prací se stránkou, protože zapouzdřuje stránku. Tester pak pouze volá metody objektu, pokud chce udělat nějakou akci na stránce. Page Factory nám pak pouze zjednodušuje práci, protože můžeme použít anotace místo volání metody `findElement`.

Jedním z nejsložitějších problémů, které se při testování webových stránek vyskytují je identifikace webových elementů. Prvky můžeme identifikovat podle id, jejich vlastností nebo můžeme použít jazyk XPath. XPath je jazyk pro dotazování v XML dokumentech (musí být stromová struktura), ale je pouze pro čtení – nepodporuje vytváření nebo editaci elementů.

Nasazení

Posledním krokem je nasazení aplikace na webový hosting. Aplikace je dostupná z URL:

- <http://economicgame-kocanek.rhcloud.com/>

8.1 Webový hosting

Aplikace byla nasazena na zdarma dostupný webový hosting OpenShift [37] provozovaný firmou RedHat. Po registraci zde může uživatel nasadit několik aplikací. Hosting podporuje širokou škálu dostupných technologií, například Java, PHP, ale i Python nebo Pearl. Z datáází můžeme uvést MySQL, včetně nástroje phpMyAdmin, nebo PostgreSQL. Server taktéž podporuje continuous integration nástroj Jenkins.

8.2 Nasazení aplikace

Existují dva hlavní způsoby, jak nahrát aplikaci na hosting OpenShift:

- **Použít technologii GIT**
Lokální repozitář se musí pushnout do vzdáleného, který je na serveru. Ten se pak automaticky zkompile podle souboru pom.xml. Pokud při kompilaci programu nenastane nějaká chyba, pak je aplikace úspěšně nasazena.
- **Zkopírovat soubor typu Web Archive (WAR) na server.**
Nejprve je nutné připravit vzdálený repozitář, tak aby se server nesnažil nasadit zdrojový kód v Gitu. Poté je nutné pomocí nástroje scp zkopírovat WAR soubor z lokálního zdroje na server.

Obě varianty jsou stejně silné a je jen na uživateli, jakou si vybere.

8.3 Nasazení aplikace na univerzitní server

Později, až bude aplikace plně připravena na produkční provoz, bude aplikace nasazena na univerzitní server. V tomto režimu bude přístupná při výuce. Naplánované nasazení a použití je na zimní semestr 2016/2017

Závěr

Hlavním cílem této práce bylo navrhnout a implementovat webovou aplikaci, která by usnadnila učitelům provádět ekonomické experimenty při výuce na FIT ČVUT. Hra, která byla předlohou aplikace, simuluje koncentrovaný trh a již se na univerzitě využívá. Součástí práce byla analýza podobných, existujících řešení a rozbor zmíněného experimentu. Dále práce popisuje analýzu požadavků na aplikaci, implementaci a testování. Posledním krokem bylo nasazení aplikace.

Cíl byl naplněn, aplikace byla vytvořena a splňuje všechny funkční a nefunkční požadavky vytyčené v analýze. Implementovány jsou rovněž všechny případy užití. Aplikace je nasazená v testovacím provozu na veřejně dostupném serveru a testování aplikace neodhalilo žádné závažné nedostatky.

Přínosem této práce pro mě jako autora, bylo především pochopení principů ekonomických experimentů a získání zkušeností s implementací informačního systému v technologii Java Spring. Během psaní jsem se také naučil, jak shromažďovat zdroje informací a z nich vybrat ty, které jsem potřeboval a získané poznatky popsat. Osobním úspěchem je pro mě i dokončení práce takového rozsahu.

Vývoj celé aplikace bude pokračovat tak, aby byla připravená k nasazení do ostrého provozu. Především je potřeba zaplnit aplikaci textem, který bude napomáhat uživatelům v orientaci a to v českém i anglickém jazyce.

Literatura

- [1] Gruyer, N.; Toublanc, N.: economics-games.com. [cit. 2016-04-11]. Dostupné z: <http://economics-games.com/>
- [2] MobLab. [cit. 2016-04-11]. Dostupné z: <https://www.moblab.com/>
- [3] Holt, C.: VeconLab. [cit. 2016-04-23]. Dostupné z: <http://veconlab.econ.virginia.edu/>
- [4] Finance and Economics Experimental Laboratory at Exeter (FEELE). [cit. 2016-04-23]. Dostupné z: <http://projects.exeter.ac.uk/feeel/>
- [5] Balkenborg, D.; Kaplan, T.: *Economic Classroom Experiments [online]*. University of Exeter, 2009, [cit. 2016-04-08]. Dostupné z: <https://www.economicnetwork.ac.uk/handbook/printable/experiments.pdf>
- [6] Parker, J.: *Laboratory Experiments to Teach Introductory Economics [online]*. 1995, [cit. 2016-04-11]. Dostupné z: <http://academic.reed.edu/economics/parker/ExpBook95.pdf>
- [7] Dixit, A.; Nalebuff, B.: *Prisoners' Dilemma [online]*. 2008, [cit. 2015-11-29]. Dostupné z: <http://www.econlib.org/library/Enc/PrisonersDilemma.html>
- [8] Krátký, T.; Zoubek, B.: *Requirements Engineering [online]*. 2015, [cit. 2016-04-30]. Dostupné z: https://drive.google.com/file/d/0B8dlIg_n7aaCS3BPTEJma2g4U00/view?pref=2&pli=1
- [9] Krigsman, M.: *CIO analysis: Why 37 percent of projects fail [online]*. 2011, [cit. 2015-04-30]. Dostupné z: <http://www.zdnet.com/article/cio-analysis-why-37-percent-of-projects-fail/>
- [10] Arlow, J.; Neustadt, I.: *UML 2 a unifikovaný proces vývoje aplikací*. Brno: Computer Press, a.s., první vydání, 2007, ISBN 978-80-251-1503-9.

- [11] Seibert, P.: *How do you write software requirements? What are software requirements? What is a software requirement? [online]*. 2011, [cit. 2016-04-30]. Dostupné z: <https://hubtechinsider.wordpress.com/2011/07/28/how-do-you-write-software-requirements-what-are-software-requirements-what-is-a-software-requirement/>
- [12] Google, Inc.: *Chrome [software]*. 2016, [cit. 2016-04-13]. Dostupné z: <https://www.google.cz/chrome/browser/desktop/>
- [13] Mozilla Foundation: *Firefox [software]*. 2016, [cit. 2016-04-13]. Dostupné z: <https://www.mozilla.org/cs/firefox/new/>
- [14] Microsoft Corporation: *Internet Explorer [software]*. 2016, [cit. 2016-04-13]. Dostupné z: <http://windows.microsoft.com/cs-CZ/internet-explorer/download-ie>
- [15] Oracle Corp.: *Java SE [software]*. 2016, [cit. 2016-05-14]. Dostupné z: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- [16] Balík, M.: *Programování v jazyku Java [online]*. 2015, [cit. 2016-04-30]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-PJV/_media/lectures/01/pjv01.pdf
- [17] Oracle Corp.: *Java Overview [online]*. 2016, [cit. 2016-05-14]. Dostupné z: <http://www.oracle.com/technetwork/java/javase/overview/index.html>
- [18] Pivotal Software, Inc.: *Spring [software]*. 2016, [cit. 2016-05-14]. Dostupné z: <https://spring.io/>
- [19] Red Hat, Inc.: *Hibernate [software]*. 2016, [cit. 2016-05-14]. Dostupné z: <http://hibernate.org/orm/>
- [20] Oracle Corp.: *MySQL [software]*. 2016, [cit. 2016-05-14]. Dostupné z: <http://www.mysql.com/downloads/>
- [21] The Apache Software Foundation: *Apache Tomcat [software]*. 2016, [cit. 2016-05-14]. Dostupné z: <http://tomcat.apache.org/whichversion.html>
- [22] Oracle Corp.: *Java Server Pages [software]*. 2016, [cit. 2016-05-14]. Dostupné z: <https://jsp.java.net/download.html>
- [23] *Bootstrap [software]*. [cit. 2016-04-23]. Dostupné z: <http://getbootstrap.com/>
- [24] *Foundation [software]*. [cit. 2016-04-23]. Dostupné z: <http://foundation.zurb.com/>

-
- [25] Kunz, G.: *CHARTIST.JS [software]*. 2013, [cit. 2016-04-23]. Dostupné z: <https://gionkunz.github.io/chartist-js/>
- [26] Flot. [cit. 2016-04-23]. Dostupné z: <http://www.flotcharts.org/>
- [27] Chart.js. [cit. 2016-04-23]. Dostupné z: <http://www.chartjs.org/>
- [28] Google Charts. [cit. 2016-04-23]. Dostupné z: <https://developers.google.com/chart/>
- [29] Maven. [cit. 2016-04-23]. Dostupné z: <http://maven.apache.org/>
- [30] Git. [cit. 2016-04-23]. Dostupné z: <https://git-scm.com/>
- [31] Krátký, T.; Zoubek, B.: *Architecture and Design [online]*. 2015, [cit. 2016-04-30]. Dostupné z: https://drive.google.com/file/d/0B8d1Ig_n7aaCS3BPTEJma2g4U00/view?pref=2&pli=1
- [32] Oracle Corp.: *NetBeans IDE [software]*. 2016, [cit. 2016-04-23]. Dostupné z: <http://www.netbeans.org/>
- [33] JUnit: *JUnit [software]*. 2016, [cit. 2016-05-14]. Dostupné z: <http://junit.org/junit4/>
- [34] Faber, S.: *Mockito [software]*. 2016, [cit. 2016-05-14]. Dostupné z: <http://mockito.org/>
- [35] OpenQA: *Selenium [software]*. 2016, [cit. 2016-05-14]. Dostupné z: <http://www.seleniumhq.org/>
- [36] Štrobl, R.: *Návrhové vzory pro Selenium [online]*. 2015, [cit. 2016-04-30]. Dostupné z: https://qa-automation.net/bi-ats/prednaska_09/BI-ATS-P09.pdf
- [37] Red Hat, Inc.: *OpenShift*. 2016, [cit. 2016-04-22]. Dostupné z: <https://www.openshift.com/>

Seznam použitých zkratk

- UML** United Modeling Language
- JRE** Java Runtime Environment
- POJO** Plain Old Java Object
- API** Application Programming Interface
- JSP** JavaServer Pages
- JSTL** JSP Standard Tag Library
- HTML** HyperText Markup Language
- CSS** Cascading Style Sheets
- MVC** Model View Controller
- SQL** Structured Query Language
- DDL** Data Definition Language
- CASE** Computer-Aid Software Engineering
- IDE** Integrated Development Environment
- DAO** Data Acces Object
- CDN** Content Delivery Network
- GUI** Graphical user interface
- XML** Extensible markup language
- WAR** Web Archive

Obsah přiloženého CD

readme.txt.....	stručný popis obsahu CD
impl	
├ backend.....	zdrojové kódy backendové části
├ frontend.....	zdrojové kódy frontendové části
thesis	
├ thesis.pdf.....	text práce ve formátu PDF
├ thesis.tex.....	zdrojová forma práce ve formátu \LaTeX
docs	
├ installation	