Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Ústav aplikované informatiky

Aplikovaná informatika

Bakalářská práce

Vývoj webové aplikace pro podporu plánování filmu

Developing a web application for film planning

Martin Tomšovský

Vedoucí práce: RNDr. Icha Jaroslav

Větřní, 2013

10. dubna 2013 Martin Tomšovský

Bibliografické údaje
Tomšovský M.: Vývoj webové aplikace pro podporu plánování filmu.
[Developing a web application for film planning] – 40p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech republic

Anotace
Bakalářská práce se zabývá vývojem webové aplikace, jež má ve své první verzi usnadnit, především amatérským filmařům, plánování filmového natáčení. Zaobírá se průzkumem současného trhu, kdy na základě dosažených výsledků dochází k vymezení požadavků webové aplikace vyvíjené v rámci této práce. Čtenář je seznámen se všemi fázemi vzniku této aplikace, její vnitřní strukturou či použitými technologiemi. Výstupem práce je hotový prototyp aplikace schopný testu u vybraných tvůrců, kteří se podíleli na definici potřeb.

In English
The Bachelor's dissertation is interested in web application developing. The application should make the whole film planning process much easier for amateur filmmakers. The web app's functional requirements are based on research of the current marketplace. The reader is introduced to all the phases of web app creation, its inner architecture and technologies it uses. The main output of this work is to develop a prototype of this application, which must be ready for testing by the selected filmmakers.

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

10. 4. 2013

Podpis

Poděkování
Rád bych poděkoval panu RNDr. Jaroslavu Ichovi i ostatním pedagogům Ústavu aplikované informatiky za pomoc a ochotu při odborných konzultacích, které výrazně přispěly k uskutečnění této bakalářské práce. Rád bych také poděkoval filmovým tvůrcům Adamu Dvořákovi, Petru Hálovi a Filipu Vorlovi za jejich ochotu při rozhovorech týkajících se filmového plánování. Poděkování patří též studentům českých filmových škol, kteří ochotně odpovídali na mnou předložený dotazník.

Obsah

1 Úvod

Filmová preprodukce je první fází vývoje filmu, během níž se provádí rozhodnutí nezbytná pro zahájení natáčení. Zahrnuje psaní literárního scénáře[1], technického scénáře[2], tvorbu storyboardu[3] či harmonogramu natáčecích dní, volbu různých strategií, mezi které lze zařadit kupříkladu výběr mikrofonů, světel či kamery a s ní související charakteristiku výsledného videa.[4]

V profesionální i amatérské sféře filmového průmyslu existuje hned několik softwarových aplikací ať už desktopových nebo webových, zaměřujících se na filmovou preprodukci. Někteří čeští filmaři jsou mírně skeptičtí vůči moderním technologiím a z průzkumu vyplynulo, že část z nich vůbec tyto specializované nástroje nevyužívá. Ani v amatérském odvětví, na které se bakalářská práce zaměřuje, nemají preprodukční softwarové prostředky přílišné zastoupení a pokud ano, jedná se spíše o aplikace desktopové, které nepodporují žádnou formu komunikace s ostatními členy filmového štábu. Některé aplikace, internetové i desktopové, nabízejí celou řadu komponent, jež ale nenacházejí u amatérských filmařů uplatnění. Velké množství prvků dává filmovému tvůrci značný počet možností, bohužel na úkor přehlednosti a jednoduchosti programu, kterou amatér upřednostňuje.

Tato bakalářská práce se zabývá průzkumem současného náhledu tvůrců na informační technologie v oblasti filmové preprodukce, ale také návrhem a vývojem prototypu webové aplikace, jež má později ve své první verzi naplnit požadavky dotázaných filmařů, zejména těch amatérských a začínajících.

[1] Literární scénář je první fází filmového díla. Obsahuje příběh – tak jak se v průběhu filmového vyprávění rozvíjí a graduje, charakterizuje ale i specifickou filmovou formu, která je příběhu adekvátní. (podle [1])

[2] Technický (režijní) scénář vychází z literárního scénáře. Píše jej v období příprav filmu režisér případně ve spolupráci s kameramanem, výtvarníkem, střihačem, hudebním skladatelem a zvukařem. Scénář je rozdělen na záběry, u každého je označena jeho délka a velikost, sklon kamer, pohyby kamery, apod. (podle [1]).

[3] Storyboard je jeden z hlavních nástrojů pro přípravu natáčení, jedná se o kreslené reprezentace obrazů produkce. Mají filmovému štábu přiblížit, jak se má vyjádřit kompozice, velikost záběru a střihy mezi scénami (podle [2]).

[4] LONG, Ben a Sonja SCHENK. *Velká kniha digitálního videa*. Vyd. 1. Překlad Magdalena Kolínová. Brno: Computer Press, 2005, s. 23. ISBN 80-251-0580-6.

1.1 Cíle

Hlavním cílem bakalářské práce je navrhnout a implementovat prototyp webové aplikace, která bude obsahovat základní komponenty pro vytvoření kvalitního plánu natáčení, bude dostatečně intuitivní tak, aby jejímu uživateli umožnila efektivně pracovat. Dále tvůrci usnadní organizaci filmového natáčení a dobrou spolupráci mezi všemi účastníky filmového štábu.

1.2 Použitá metodologie

Vývoj webové aplikace je rozfázován do několika částí:

- Úvodní rešerše čili analýza,
- vymezení požadavků vycházejících z úvodní analýzy,
- návrh aplikace,
- implementace,
- testování.

Vývoj aplikace, zahrnující úvodní analýzu, zachycení požadavků, návrh aplikace a její implementaci odpovídá vývojovému procesu vyvozeného z vodopádového modelu[5]. Následné testování je založeno na iterativním modelu[6], kdy se osobě pohybující se ve filmovém průmyslu předá testovací verze aplikace a dle zpětné vazby dojde k její následné úpravě.

[5] Vodopádový model je základním modelem životního cyklu softwaru. Jednotlivé etapy jsou seřazeny za sebou, přičemž následující etapa začíná až po ukončení té předcházející (podle [3]).

[6] Iterativní model rozděluje proces vývoje softwaru do iterací. Po každé iteraci má uživatel k dispozici spustitelnou verzi softwaru (s neúplnou funkcionalitou), která mu pomůže upřesnit požadavky na software, které jsou zapracovány v další iteraci (podle [3]).

2 Analýza

Na základě vlastního pozorování byly vypracovány hypotézy, jež byly do jisté míry potvrzeny, ale i částečně vyvráceny následným průzkumem trhu, rozborem míry používání dostupných aplikací a analýzou názorů tvůrců na konkrétní nástroje.

2.1 Vlastní předpoklad

Samotný nápad k vytvoření systému tohoto ražení vzniknul dle vlastní praxe s natáčením amatérského filmu. U každého dalšího hotového snímku se potvrdila obecně známá teorie, jak důležitou částí preprodukce je. Během úplně prvních zkušeností s filmováním docházelo k výraznému podcenění úvodního procesu, což se znatelně projevilo v postprodukčním střihovém programu při onom sestavování výsledného videa, kdy na povrch vyvěralo několik zásadních problémů v důsledku nedostatečné přípravy. Příkladem může být režisér, který si jednotlivé záběry chce sestavovat z hlavy až během natáčení, snadno se pak může stát, že střihač narazí při své práci na neřešitelnou potíž chybějícího záběru. Čím větší pozornost tvůrce preprodukci věnuje, tím menší zmatky vznikají při práci v terénu a tím jednodušší je poté závěrečný střih.

Celý průběh zhotovení filmového díla, především jeho efektivita, je výrazně ovlivněna počtem lidí, kteří se na filmu podílejí a vhodně zvolenou taktikou vzájemné komunikace. Malý filmový štáb čítající pár jedinců si při domluvě na podrobnostech příštího natáčení vystačí s mobilním telefonem či e-mailem. Jakmile se skupina o poznání rozroste, dochází při použití takových technologií ke značnému poklesu efektivity práce. Při natáčení posledního filmu, prvního celovečerního, bylo téměř nezdolnou překážkou vytvořit harmonogram natáčecích dní, poněvadž celý filmový ansámbl se rozšířil o velký počet členů, z nichž každý měl k dispozici jen určitý počet volných dní. Najít takové časové rozmezí, které vyhovuje všem zúčastněným, je těžký úkol nejen při plánování filmového natáčení, ale obecně při plánování jakéhokoliv typu akcí. Ona synchronizace, jejímž cílem je zvolit natáčecí den, jenž naplňuje představy všech členů, je při užití mobilních telefonů či e-mailů velmi frustrujícím procesem. Konkrétně tyto zážitky vedly k myšlence vyrobit takový softwarový produkt, který by výše uvedené problémy co nejlépe eliminoval, který by byl schopný usnadnit tvůrcům organizaci filmového natáčení a nabízel by snazší způsob spolupráce mezi členy filmového štábu.

Pro amatérského tvůrce, jenž natáčí neziskové filmy a svou tvůrčí činnost provozuje hlavně pro zábavu, je důležitým prvkem při výběru vhodné plánovací aplikace její bezplatnost.

Vlastní hypotézy byly hlavním důvodem k námětu takovou aplikaci vyvinout. Určujícím faktorem pro dosažení toho, jakým směrem se ubírat a jaké požadavky je nutné naplnit není ona hypotéza, nýbrž názory a zkušenosti ostatních osob, pohybujících se ve světě filmu.

2.2 Průzkum trhu

Průzkum byl prováděn prostřednictvím dvou pomůcek pro získávání informací – dotazníkem rozeslaným mezi amatérské a začínající filmaře a třemi rozhovory. První rozhovor byl uskutečněn telefonicky s profesionálním filmovým tvůrcem, producentem a střihačem Adamem Dvořákem[1], druhé a třetí jednání proběhlo při osobních schůzkách s amatérskými filmaři Filipem Vorlem[2] a Petrem Hálou[3]. Na základě získaných informací o nejhojněji využívaných nástrojích přišla na řadu zevrubná analýza dvou z nich – CeltX, Adobe Story a letmý rozbor zbylých třech – Movie Magic, Final Draft, Scripped.

Ve filmovém průmyslu existuje několik aplikací, které slouží amatérům i profesionálům k usnadnění práce ve fázi filmové preprodukce. Jedním z nejoblíbenějších specializovaných software mezi profesionály jsou komplexní Final Draft či Movie Magic. Novinkou na trhu je ve své placené verzi velmi robustní internetová aplikace – Adobe Story. Amatérští tvůrci naproti tomu dávají přednost jednoduššímu CeltX. Velké množství začátečníků, ale i někteří profesionálové nevyužívají žádnou specializovanou aplikaci a preferují textové nebo tabulkové procesory či editory, případně tužku a papír.

2.2.1 CeltX

Populární desktopová aplikace vhodná pro začínající filmové autory. Obsahuje základní nástroje vhodné pro přípravu filmového natáčení – editor literárního scénáře, správce storyboardu či evidenci různých položek jakými jsou rekvizity, kostýmy, kamery, lokace,

[1] Adam Dvořák je producent a střihač, jenž vystudoval filmovou a televizní fakultu Akademie múzických umění. Je producentem filmů Rafťáci, Bobule, Lidice či Martin a Venuše. Jako střihač poté působil u dalších známých českých filmů, jakými jsou např. Výlet, Sametoví vrazi nebo Probudím se včera.

[2] Filip Vorel je herec a amatérský filmový tvůrce. Ztvárnil role ve filmech svého strýce Tomáše Vorla staršího – Gympl a Cesta z města. Jako filmový tvůrce se účastnil kupříkladu soutěže 48 Hour Film Project Praha.

[3] Petr Hála je začínající filmař - režisér a scénárista. Posledním jeho filmem je parodie Brocky Lovec Salvation, v současné době připravuje řadu dalších snímků.

postavy, apod. Jedná se o program velice intuitivní a jednoduchý. Zahrnuje českou podporu a navíc je distribuován bezplatně.

Mezi nevýhody patří zcela jistě téměř nulová možnost natáčení organizovat. Software nabízí v rámci svého příjemného prostředí kvalitní evidenci výše zmíněných komponent pro soukromé účely, ale není samostatně schopný se o uložené informace podělit s ostatními členy filmového štábu.

Úplnou novinkou posledních týdnů je internetová aplikace CeltX, která ve své bezplatné verzi Basic poskytuje jen torzo toho, co její desktopová obdoba. Součástí je pouze editor scénáře, ke kterému mohou mít přístup ostatní uživatelé, jež jsou rozděleni do dvou skupin – čtenáři a autoři. Čtenář má právo scénář pouze číst, autor do něj má možnost zapisovat a upravovat ho. Verze Edge dává filmovému štábu k dispozici více komponent, je však placená a z tohoto důvodu nebyla výrazněji analyzována.

2.2.2 Adobe Story

Zbrusu nová internetová aplikace, která zatím nenalezla přílišné uplatnění na českém trhu. Stejně jako aplikace CeltX je i Adobe Story distribuována ve dvou verzích – bezplatná Adobe Story Free a placená Adobe Story Plus. Důkladnější průzkum byl realizován znovu u bezplatné verze. Adobe Story Free neposkytuje uživateli evidenci žádných předmětů (rekvizit, kamer, mikrofonů), se kterými manipulují členové filmového štábu během natáčení. Zaměřuje se především na literární, ale i technický scénář. Na rozdíl od Celtx Basic nenabízí Adobe Story Free žádnou možnost interakce s ostatními členy filmového štábu. Tu nabízí až placená, velmi robustní Adobe Story Plus. Nebylo tomu tak vždy, v úplných začátcích této aplikace bylo sdílení projektu možné i v bezplatné verzi, až později byla tato komponenta přesunuta do placené edice a v současné době je dostupná jedině zde.

Velkou nevýhodou je značná neintuitivnost. Během vlastního testování, při kterém byl tento produkt vybrán jako nástroj pro organizaci jednoho z filmů, nastaly problémy hned při úvodním záměru začít psát obyčejný literární scénář, kdy nebylo možné v rozmezí prvních několika minut nalézt volbu pro jeho vytvoření. Při nemožnosti jednoduchého provedení dalšího dílčího úkolu, jehož cílem bylo přidat k filmovému projektu jiného člena, došlo k rychlému pozměnění strategie a zpětnému přechodu k CeltX. Vývojáři se pokoušejí tento nedostatek vynahradit několika tutoriály, jež mají uživatele dovést ke správnému používání programu.

2.2.3 Final Draft

Zřejmě nejznámější program, jehož spotřebiteli jsou například věhlasná americká studia MGM, Pixar, Warner Bros, Paramount, Dreamworks, apod. Podle slov pana Adama Dvořáka je tento software častým nástrojem pro preprodukci i u českých profesionálních filmařů. Final Draft je desktopová placená aplikace, která předkládá odborným tvůrcům vše potřebné pro dokonalý plán natáčení. Jedná se o desktopovou aplikaci, která svým uživatelům neumožňuje vzájemnou komunikaci. Členové filmového štábu si mezi sebou zasílají vyexportované výstupy, se kterými v závislosti na svých filmových rolích dále pracují.

2.2.4 Movie Magic

Své zastoupení ve filmové branži má i desktopový placený program Movie Magic. Byly vyvinuty tři podoby této aplikace – Movie Magic Screenwriter, Movie Magic Budgeting a Movie Magic Scheduling. První z nich, Movie Magic Screenwriter, je klasickým editorem literárního scénáře, naproti tomu Movie Magic Budgeting poskytuje správu financí a rozpočtu u filmového projektu, proto je tato aplikace vhodná obzvláště pro producenty. Movie Magic Scheduling slouží jako plánovač natáčení s podporou importu literárního scénáře z populárních programů Movie Magic Screenwriter či Final Draft.

2.2.5 Ostatní aplikace

Existuje několik dalších aplikací podobného rozsahu a zaměření na cílovou skupinu uživatelů, nicméně výše uvedená čtveřice je podle průzkumu jednoznačným lídrem. Do okruhu amatérských a profesionálních filmařů se snaží proniknout i program Scripped, který zatím neumožňuje spolupráci mezi jednotlivými účastníky filmového projektu a navíc neobsahuje mnoho dalších prvků, kterými disponují konkurenční výrobky. K plnému užívání aplikace Scripped je nutná placená registrace na internetových stránkách produktu. Dalšími programy, zabývajícími se stejnou tématikou, jsou třeba Storyist, Scrivener či Movie Outline. Všechny jsou desktopové a navíc placené.

2.2.6 Míra využití jednotlivých aplikací v profesionální sféře

Informace poukazující na uplatnění jednotlivých aplikací mezi profesionály byly získány během telefonického rozhovoru s producentem a střihačem Adamem Dvořákem, jenž získal své bohaté zkušenosti v průběhu natáčení několika známých českých filmů. Dle jeho

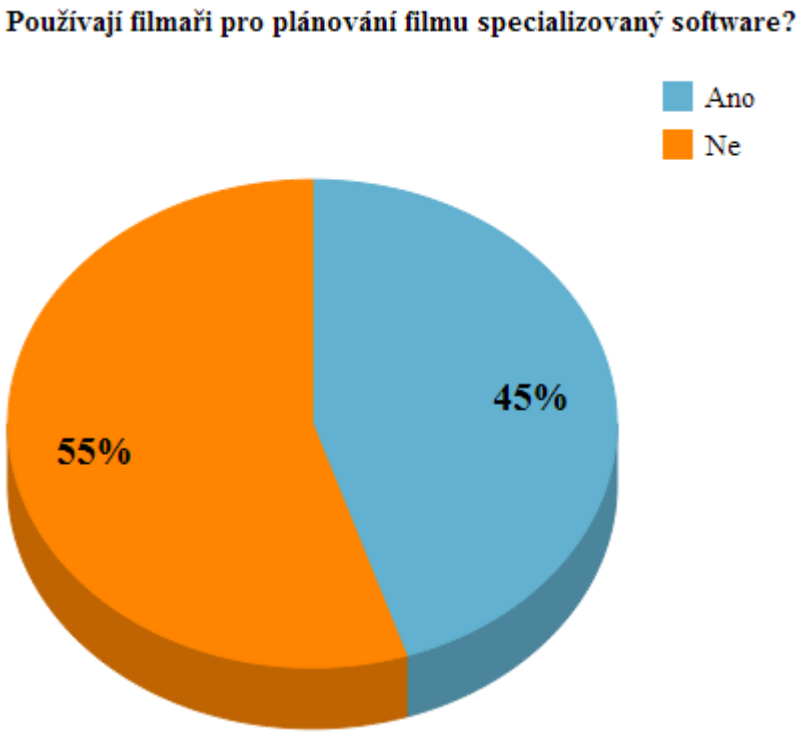
slov je v odborných kruzích nejoblíbenějším nástrojem pro preprodukci desktopový program Final Draft. Sám pan producent naráží ze specializovaných aplikací nejčastěji právě na tento software. Řekl však, že setkání s podobným typem aplikací u něj probíhá pouze v pasivní formě, jelikož on sám v těchto programech nikdy nepracuje, pouze přijímá exportované výstupy od ostatních členů filmového štábu, u kterých je práce se zmíněnými aplikacemi předpokladem. Jako další příklad často používaného software uvedl Movie Magic.

2.2.7 Míra využití jednotlivých aplikací v amatérské sféře

Bakalářská práce se zaměřuje především na amatérskou tvorbu filmů, proto jí byla ve fázi analýzy věnována velká pozornost. Sběr informací byl úspěšně dokončen zásluhou dobré kooperace se začínajícími filmaři Filipem Vorlem a Petrem Hálou, ale také díky mnoha odpovědím na dotazník, jenž byl odeslán do českých filmových škol a amatérských filmových organizací.

Filip Vorel i Petr Hála potvrdili hypotézu, že jako amatéři a tvůrci neziskových snímků upřednostňují výhradně bezplatný software. Z hlediska bezplatnosti vyhovují z výše sepsaného výčtu aplikací pouze dvě – CeltX a Adobe Story, konkrétně desktopový program CeltX, jeho webová analogie CeltX Basic či neplacený produkt společnosti Adobe – Adobe Story Free. Petr Hála používá pro plánování svých filmů desktopovou podobu CeltX, zatímco Filip Vorel sází na základní textové procesory, tužku a papír. Při své tvůrčí činnosti narazil také na internetovou aplikaci Adobe Story Free.

Zásadní roli v celém průzkumu sehrály odpovědi respondentů na dříve zmíněný dotazník, který ukazuje, jaký software je nejčastěji používán a kým. Mezi začátečníky je úvodní etapa vzniku filmu dosti podceňována, což jasně dokazuje 55% dotazovaných, kteří pro své filmové plány nevyužívají vůbec žádnou specializovanou aplikaci (obr. 2.1). Část z nich uvádí, že scénář píší v textových procesorech a ke všemu ostatnímu jim stačí tužka a papír.



Obrázek 2.1. Graf: Používají filmaři pro plánování filmu specializovaný software?

Dotazník též mapoval, jakou konkrétní aplikaci začínající tvůrci nejčastěji používají (obr. 2.2). Mezi amatéry vítězí naprosto jednoznačně program CeltX, majoritně jeho desktopové provedení. Respondenti měli možnost napsat program, který nebyl součástí předchozích voleb. Tato doplňková odpověď neobsahovala ani v jednom případě druh preprodukčního software, dotázaní zde uváděli jako příklad takových aplikací obyčejné textové editory či dokonce e-mail. Na druhém místě se umístila placená aplikace Movie Magic, za ní zaostávají Final Draft, Adobe Story i Scripped.

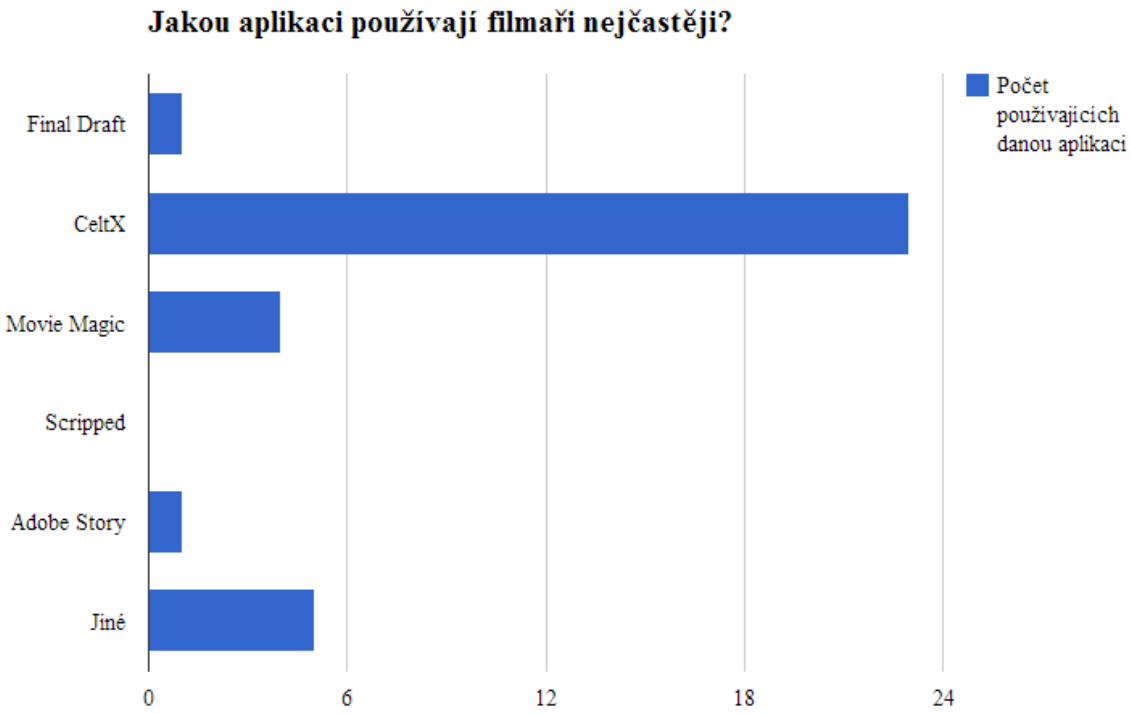
Během důkladnějšího přezkoumání došlo také k analýze využívanosti aplikací vzhledem k jednotlivým filmovým oborům. Na dotazník nejčastěji odpovídali režiséři, producenti a scénáristé.

Jediní, kdo pro svou práci doopravdy potřebují alespoň základní editor scénáře, jsou scénáristé (obr. 2.3). Jen 23,8% z nich nepoužívá žádnou aplikaci zaměřující se na filmovou preprodukci. Naproti tomu 76,2% autorů scénáře takový software aplikují při plánování svých děl, ba co víc, všichni pracují v programu CeltX.

Výsledky šetření dále poukazují na to, že u režisérů je tomu, co se využívanosti plánovacích aplikací týká, téměř opačně než u scénáristů (obr. 2.4). Jen 38,1% dotázaných

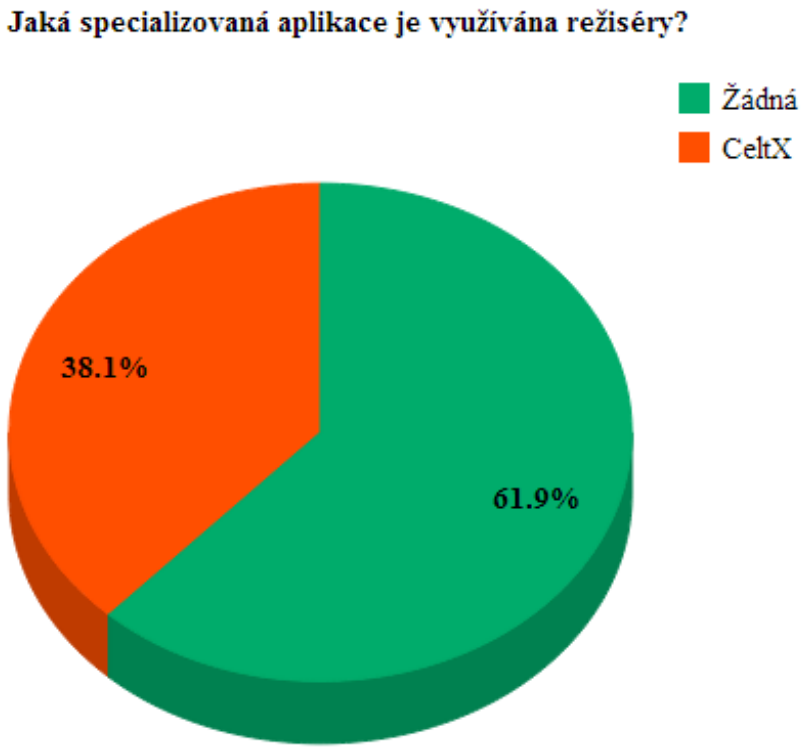
režisérů dělá se specializovaným software, stoprocentní zastoupení má opět CeltX. U 64,9% režisérů nenacházejí preprodukční nástroje žádné uplatnění.

U producentů je situace poněkud rozmanitější (obr. 2.5). Do hry zde vstupuje kromě CeltX (31,8%) také desktopový Movie Magic (18,2%), jenž naplňuje požadavky producentů především v produktech Movie Magic Budgeting a Scheduling, které nabízí nejen plánování natáčení, ale i správu financí a rozpočtu.

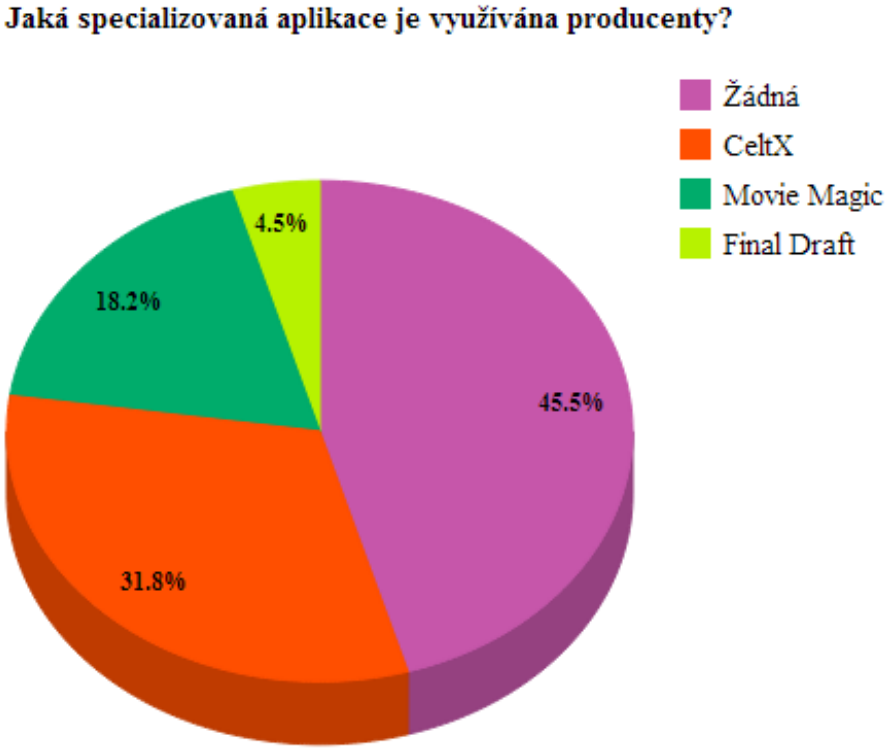


Obrázek 2.2. Graf: Jakou aplikace používají filmaři nejčastěji?

Obrázek 2.3. Graf: Jaká specializovaná aplikace je využívána scénáristy?



Obrázek 2.4. Graf: Jaká specializovaná aplikace je využívána režiséry?



Obrázek 2.5. Graf: Jaká specializovaná aplikace je využívána producenty?

2.3 Postřehy filmových tvůrců

Průzkum trhu dal k dispozici aktuální data o oblíbenosti jednotlivých preprodukčních aplikací. Dalším cílem této analýzy je získat osobní názor na tyto aplikace od osob, které s nimi pracují. Pozornost byla věnována především jejich spokojenosti se softwarovým produktem z hlediska organizace a koordinace s ostatními kolegy. Jiným důležitým aspektem byl stupeň intuitivnosti programů.

Filmaři se také podělili o své dosavadní zkušenosti, v rámci kterých vysvětlili jejich vlastní strategie vedoucí k organizaci natáčení a spolupráci s dalšími účastníky filmového projektu.

2.3.1 Profesionální sféra

Pan Adam Dvořák, s nímž byl veden telefonický rozhovor, byl trochu skeptický ohledně nápadu na vytvoření nové webové aplikace, jež by přinesla vylepšení ve formě dobré spolupráce s ostatními spolupracovníky. Řekl, že v jeho okolí po podobném produktu není téměř žádná poptávka. Vzájemná spolupráce ve filmovém štábu funguje prostřednictvím mobilních telefonů nebo e-mailů, což je v této době podle názoru pana Dvořáka pro českou

filmovou tvorbu dostačující způsob komunikace. Svou současnou spokojenost zdůvodnil na příkladu šíření hotového scénáře mezi osoby podílejícími se na vývoji filmu, kdy scénárista napíše scénář (obvykle v jednom ze specializovaných nástrojů jakým je Final Draft), který vytiskne do formátu pdf a v této podobě ho rozešle elektronickou poštou napříč filmovým štábem. Podobným způsobem postupuje člověk zodpovědný za vytvoření harmonogramu natáčení, který pro svou práci použije kupříkladu některý z dostupných tabulkových procesorů a výsledný soubor propaguje stejnou cestou.

2.3.2 Amatérská sféra

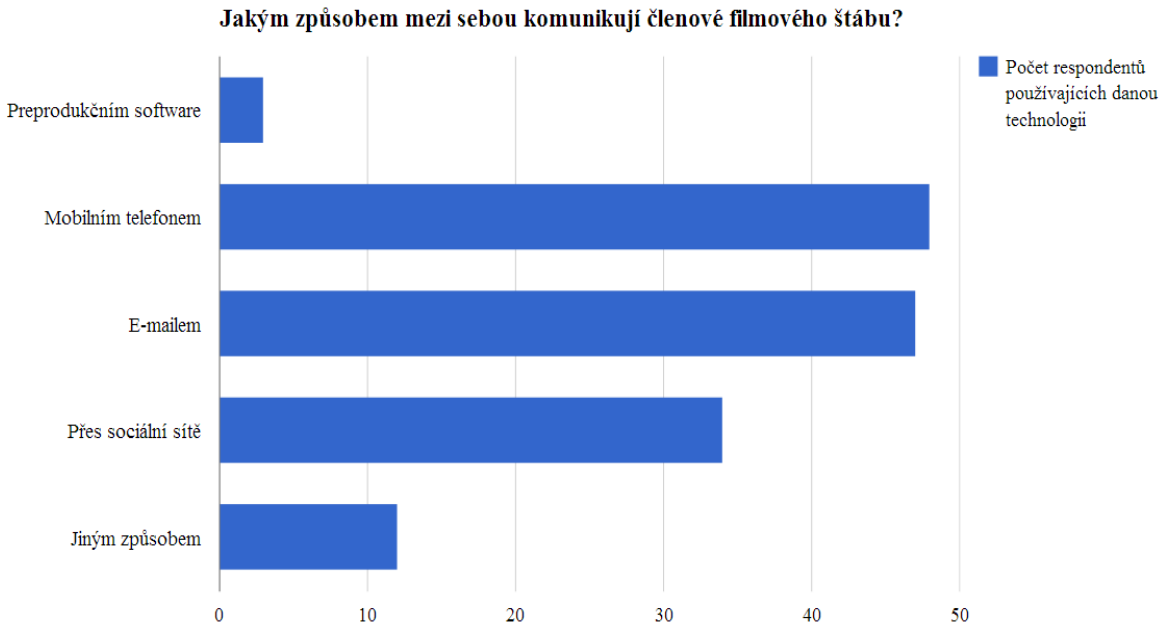
Rozhovory s Petrem Hálou i Filipem Vorlem vedly k několika závěrům, které potvrdily původní teorie. Oba jednoznačně upřednostňují bezplatný software před placeným. Důvod je jasný – filmování neberou jako výdělečnou činnost, nýbrž jako zábavu, na níž však netouží příliš prodělávat. Oba pro spolupráci se svými kolegy preferují mobilní telefony a elektronickou poštu, případně některý ze software pro instant messaging[4]. Oba by uspokojilo jisté vylepšení ve směru kooperace a organizace.

Petr Hála produkuje své scénáře a plány v programu CeltX. Používá jak desktopovou, tak i internetovou aplikaci. Webová verze Celtx Basic mu vyhovuje pouze díky schopnosti sdílení scénáře s ostatními spolupracovníky. Považuje ji ovšem na rozdíl od desktopové obdoby za notně neintuitivní navíc s velmi omezenými možnostmi, mezi které lze počítat pouze sdílený editor scénáře. Jakoukoliv další komponentu k plánování filmu tento začínající režisér postrádá. Samotné psaní skriptu společně s plánováním natáčení provádí v desktopové paralele, přičemž hotový scénář importuje do internetové aplikace ke sdílení. Ideu nového webového systému velmi intenzivně podporuje, uvítal by především harmonogram natáčecích dní s výpisem scén přidruženým ke každému dni.

Filip Vorel se během své praxe nesetkal s žádnou aplikací pro podporu plánování filmu. Se svými kolegy organizují natáčení většinou verbální domluvou nebo komunikací prostřednictvím mobilních telefonů či elektronických zpráv. Pro psaní literárního scénáře si vystačí s obyčejným textovým procesorem. Nedávno však poukázal na program Adobe Story, který měl možnost sám otestovat. Označil ho za příliš složitý a brzy jej opustil. Stejně jako Petr Hála by rád uvítal jednoduchou internetovou aplikaci, pokud možno bezplatnou, jež by splňovala základní požadavky ke kvalitní preprodukci.

[4] Mezi software pro instant messaging lze zařadit kupříkladu ICQ, Skype či Jabber.

Rozsáhlejší a také komplexnější soubor informací vnesl do práce dotazník. Jeho snahou bylo rovněž potvrdit či vyvrátit vlastní hypotézy a dosáhnout ucelenějšího pohledu na věc. Jedna z otázek přibližuje, jakým způsobem mezi sebou filmaři nejčastěji komunikují (obr. 2.6). Komunikace mezi aktéry probíhá nejběžněji přes mobilní telefon či elektronickou poštu, v závěsu za těmito dvěma technologiemi jsou sociální sítě. V sekci „jiné“ respondenti uváděli ve většině případů programy pro instant messaging, např. ICQ. Pouze tři dotázaní používají pro spolupráci s dalšími členy filmového štábu svůj preprodukční software.



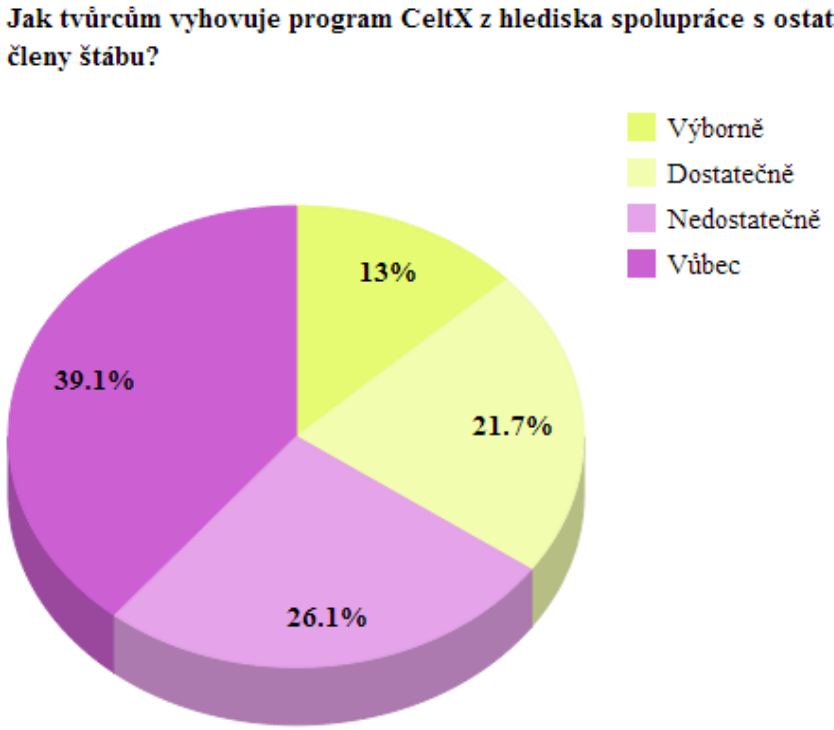
Obrázek 2.6. Graf: Jakým způsobem mezi sebou komunikují členové filmového štábu?

Následující dvě otázky se omezují výhradně na program CeltX, jenž je používán naprosto drtivou většinou dotázaných. Uvádět statistiku zbylých aplikací by bylo bezpředmětné, poněvadž by se jednalo o velmi neobjektivní ukazatele.

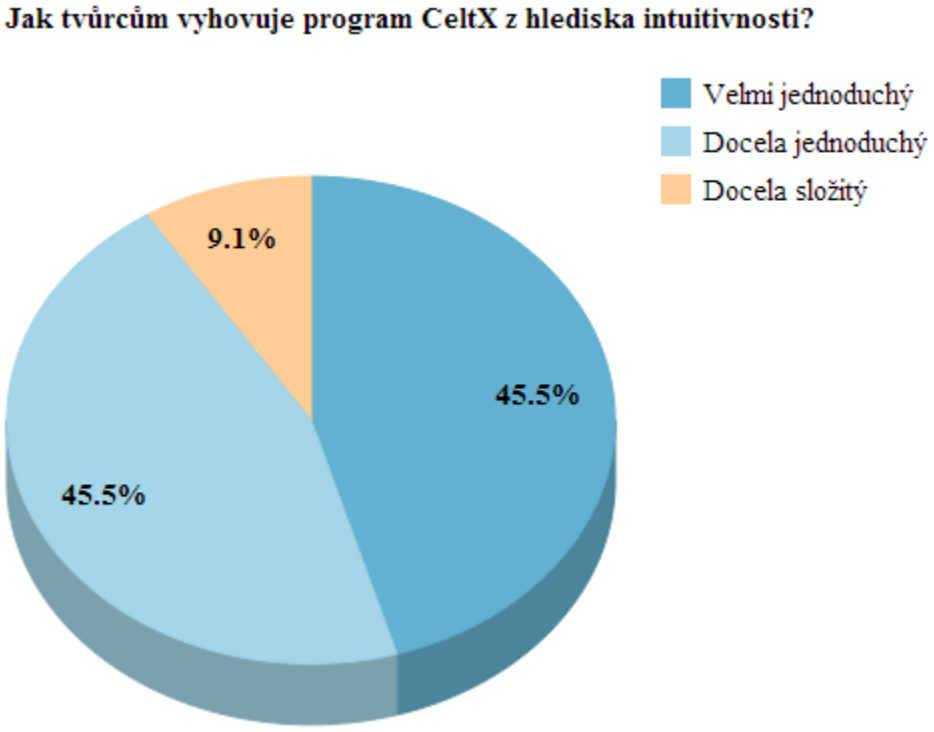
Se způsobem vzájemné interakce mezi spolupracovníky souvisí i dotaz, jehož cílem je zjistit, jak jsou uživatelé spokojení s aplikací CeltX vzhledem k možnosti komunikace se členy filmového štábu (obr. 2.7). Dohromady 65,2% respondentů odpovědělo, že program CeltX jim nenabízí dostatečně uspokojivou podporu spolupráce s ostatními účastníky filmového projektu. Zbylých 34,8% dotázaných je v tomto ohledu spokojených.

Základem úspěchu webové i desktopové aplikace je její dostatečná intuitivnost. Čelí-li uživatel při svém prvním kontaktu s aplikací chaosu, je velmi rychle odrazen a program opustí. Odběratelé výrobků CeltX jsou po této stránce se svým preprodukčním softwarem

velice spokojení (obr. 2.8). 90% odpovídajících pokládá ovládání aplikace CeltX za jednoduché a intuitivní. Pouhá desetina z nich si stěžuje na přílišnou složitost tohoto produktu.



Obrázek 2.7. Graf: Jak tvůrcům vyhovuje program CeltX z hlediska spolupráce s ostatními členy štábu?



Obrázek 2.8. Graf: Jak tvůrcům vyhovuje program CeltX z hlediska intuitivnosti?

3 Vymezení požadavků

Předchozí průzkum prokázal, že v amatérské sféře je mezi preprodukčními softwarovými produkty jednoznačným lídrem program CeltX. Aby byla nová webová aplikace, vytvářená v rámci bakalářské práce, schopna tomuto výrobku konkurovat, bylo zapotřebí vyvarovat se jeho chyb, poučit se z jeho nedostatků a naopak se inspirovat kladnými vlastnostmi onoho produktu, které byly uživateli nejčastěji oceňovány.

Dvěma hlavními a nejdůležitějšími cíli vyvíjené aplikace jsou snadná uživatelská použitelnost a schopnost organizovat a nejen evidovat. Program by měl být minimálně stejně intuitivní jako výše zmíněný CeltX (především jeho desktopová obdoba) a jeho přidanou hodnotou by měla být snadná možnost podílet se na filmovém díle i s ostatními členy filmového štábu.

3.1 Funkční požadavky

Důkladný rozbor problému, na kterém se podíleli také amatérští filmaři Filip Vorel a Petr Hála, přinesl konečné řešení ohledně vymezení funkčních požadavků, jež by měla aplikace splňovat. Program by měl jeho uživateli poskytnout možnost:

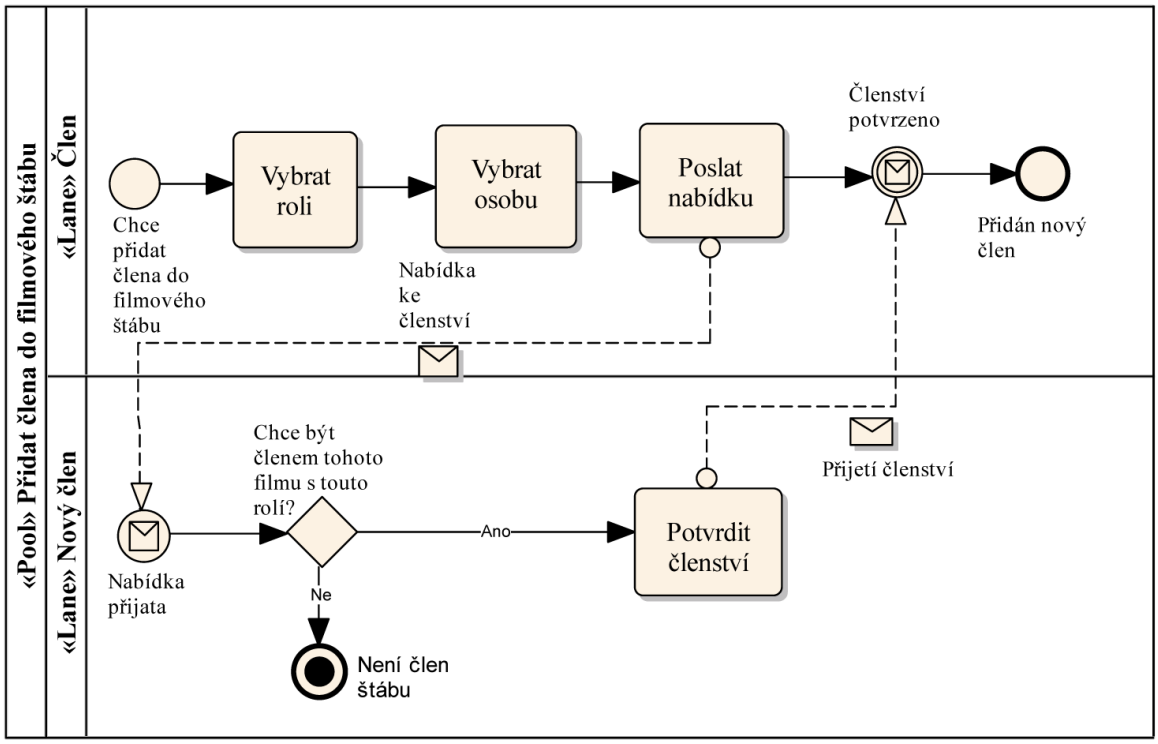
- Vytvořit si filmový projekt,
- poskládat si vlastní filmový štáb v rámci daného projektu,
- přiřadit každému členovi filmového štábu různá práva nad úpravou projektu,
- organizovat natáčení podle natáčecích dní,
- psát a sdílet literární scénář,
- evidovat veškeré předměty a další náležitosti nutné pro samotné natáčení (rekvizity, lokace, kamery, mikrofony).

Další informace jsou zaneseny jednak v logickém rámci či modelech podnikových procesů (tzv. BPMN), ale také v jednotlivých diagramech jazyka UML strukturovaného i behaviorálního typu (diagram případů užití, diagram tříd, diagram aktivit).

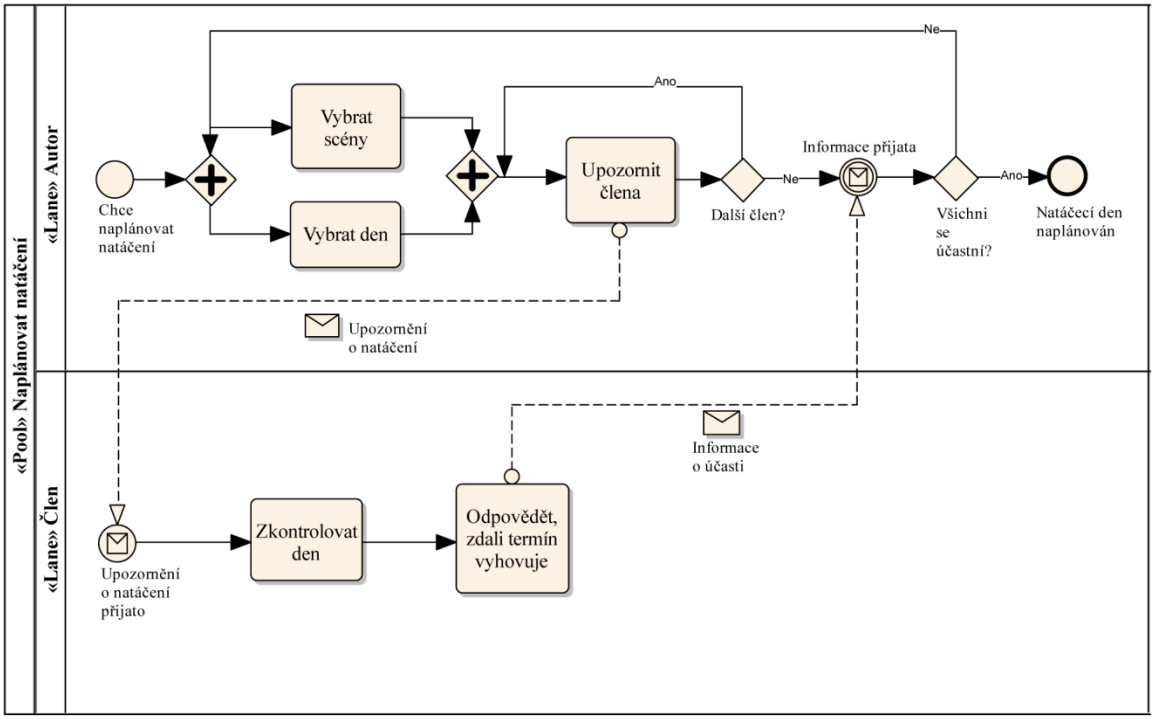
3.1.1.1 Business Process Model and Notation (BPMN)

Hlavním cílem BPMN je poskytnout notaci, která je snadno srozumitelná všem podnikovým uživatelům – od podnikových analytiků, kteří vytvářejí počáteční návrhy

procesů přes vývojáře odpovědné za implementaci technologie, která tyto procesy bude vykonávat až po obchodníky, kteří je budou spravovat a sledovat.[1]



Obrázek 3.1. BPMN: Přidání nového člena do filmového štábu



Obrázek 3.2. BPMN: Plánování natáčecího dne

[1] Object Management Group, Inc. *Business Process Model and Notation (BPMN): Version 2.0*. [online]. [cit. 2013-04-16]. s.1. Dostupné z: http://www.omg.org/spec/BPMN/2.0/.

3.1.1.2 Logický rámec

Metoda logického rámce (LR) slouží jako pomůcka při stanovování cílů projektu a jako podpora k jejich dosahování.[2]

Název projektu:	Webová aplikace pro plánování filmu	Zpracoval:	Martin Tomšovský
Vypracováno dne:	15. 1. 2013	Verze:	1.00

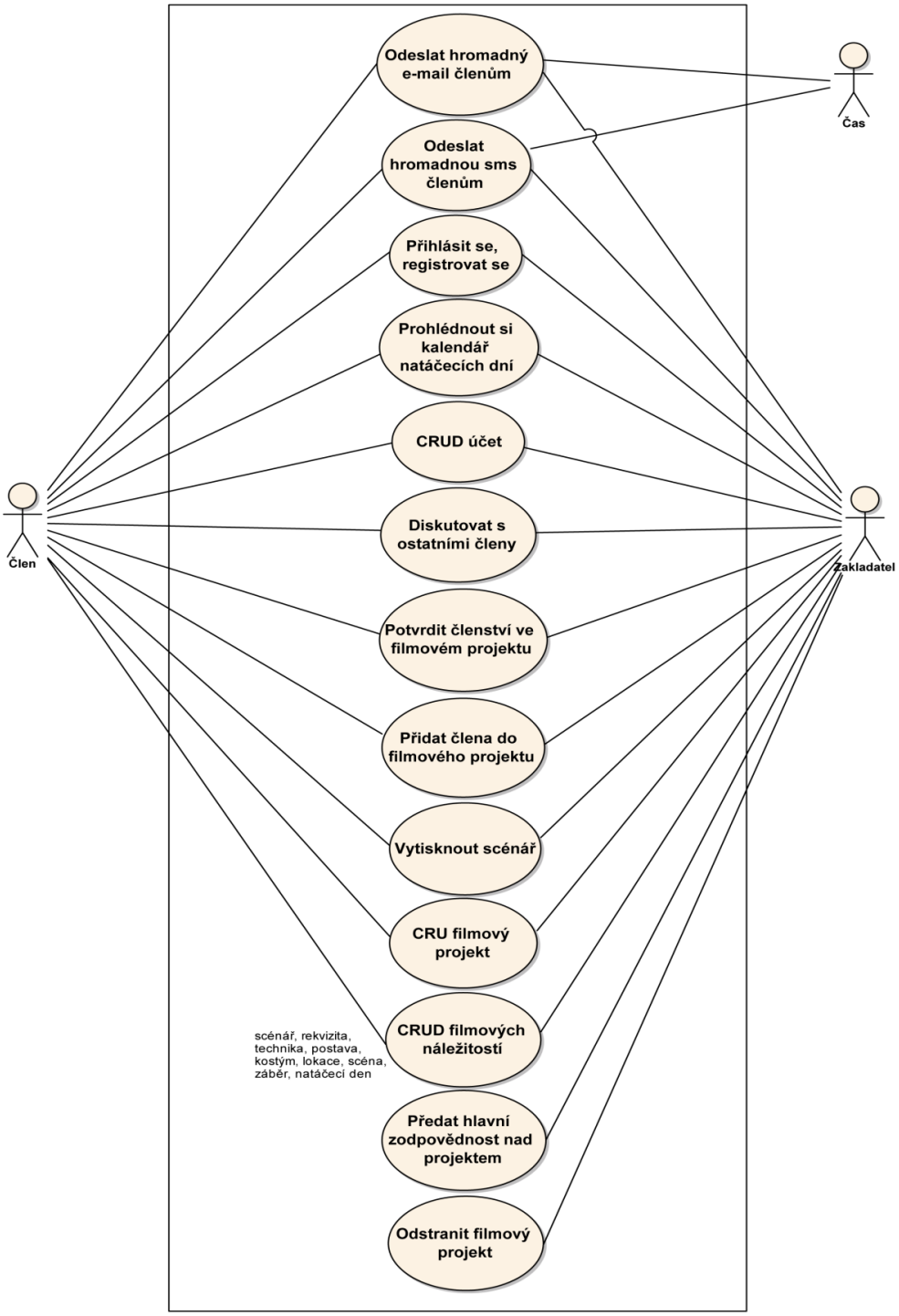
Popis projektu	**Objektivně ověřitelné ukazatele**	**Prostředky ověření**	**Předpoklady**
Cíl projektu *Zjednodušit a zefektivnit amatérským filmařům práci při plánování jejich filmu.*	*Prototyp aplikace, který je připravený k otestování vybranými amatérskými filmaři.*	*Data získaná na základě rozhovorů s filmaři, kteří aplikaci otestovali.*	
Účel projektu *Vytvořit internetovou aplikaci, která bude pro uživatele dostatečně intiuitivní a jednoduchá, bude nabízet dobrou organizaci projektů s možností podílet se na nich s ostatními členy filmového štábu.*	*Vybraní filmaři, kteří otestují aplikaci, budou spokojeni s její jednoduchostí i možností podílet se na organizaci filmu s ostatními členy filmového štábu.*	*Data získaná na základě rozhovoru s filmaři, kteří aplikaci otestovali.*	*Filmaři, kteří testují aplikaci, jsou ochotni tento software opravdu vyzkoušet takovým způsobem, aby bylo možné dostatečně dobře zjistit, došlo-li ke splnění jeho účelu a cílů.*
Výstupy Uživatel bude moci: *Vytvořit si filmový projekt,* *poskládat si vlastní filmový štáb v rámci daného projektu,* *přiřadit každému členovi různá práva nad projektem,* *organizovat natáčení podle natáčecích dní,* *psát a sdílet literární scénář,* *evidovat veškeré předměty.*	*Prototyp aplikace musí být dodán do konce dubna, v květnu je naplánován test aplikace vybranými amatérskými filmaři.*	*Data získaná na základě rozhovoru s filmaři, kteří aplikaci otestovali.*	*Dodavatel softwaru pracuje efektivně a plní harmonogram.*
Analýza současného trhu a z ní vycházející vyvození funkčních i nefunkčních požadavků, *návrh,* *implementace,* *testování.*	**Prostředky** *UML diagramy* *instalace v lokálním prostředí: vývojové prostředí, databáze, frameworky, atd.* *webhosting s podporou vybraných technologií, doména* *lidské zdroje: dodavatel, testovací filmový štáb*	**Dokončení činnosti:** *Analýza a vymezení požadavků: 30. 1. 2013,* *studium: 30. 12. 2012,* *návrh: 15. 2. 2013,* *implementace: 30. 4. 2013,* *testování: 31. 5. 2013.*	*Dostatek času na vypracování,* *odbornost dodavatele ve vybraných technologických postupech,* *vybrané technologické postupy jsou zároveň vhodné,* *dostatek finančních zdrojů na zaplacení webhostingu a domény.*

Tabulka 3.1. Logický rámec projektu

[2] DOLEŽAL, Jan, Pavel MÁCHAL a Branislav LACKO. *Projektový management podle IPMA*. 2., aktualiz. a dopl. vyd. Praha: Grada, 2012, s.64. Expert (Grada). ISBN 978-80-247-4275-5.

3.1.1.3 Případy užití

Případ užití (use case) je metodou pro zachycení funkčních požadavků na systém. Případy užití popisují typické interakce mezi uživateli systému a samotným systémem.[3]



Obrázek 3.3. Diagram případů užití k tomuto projektu

[3] FOWLER, Martin. *Destilované UML*. 1. vyd. Praha: Grada, 2009, s. 53-56. Knihovna programátora (Grada). ISBN 978-80-247-2062-3.

Scénáře k některým případům užití se nacházejí v příloze.

3.1.1.4 Diagram aktivit

Diagramy aktivit jsou technikou určenou k popisu procedurální logiky, business procesů a toku práce.[4]

Diagramy aktivit, které byly vytvořeny v tomto projektu, jsou uvedeny v příloze.

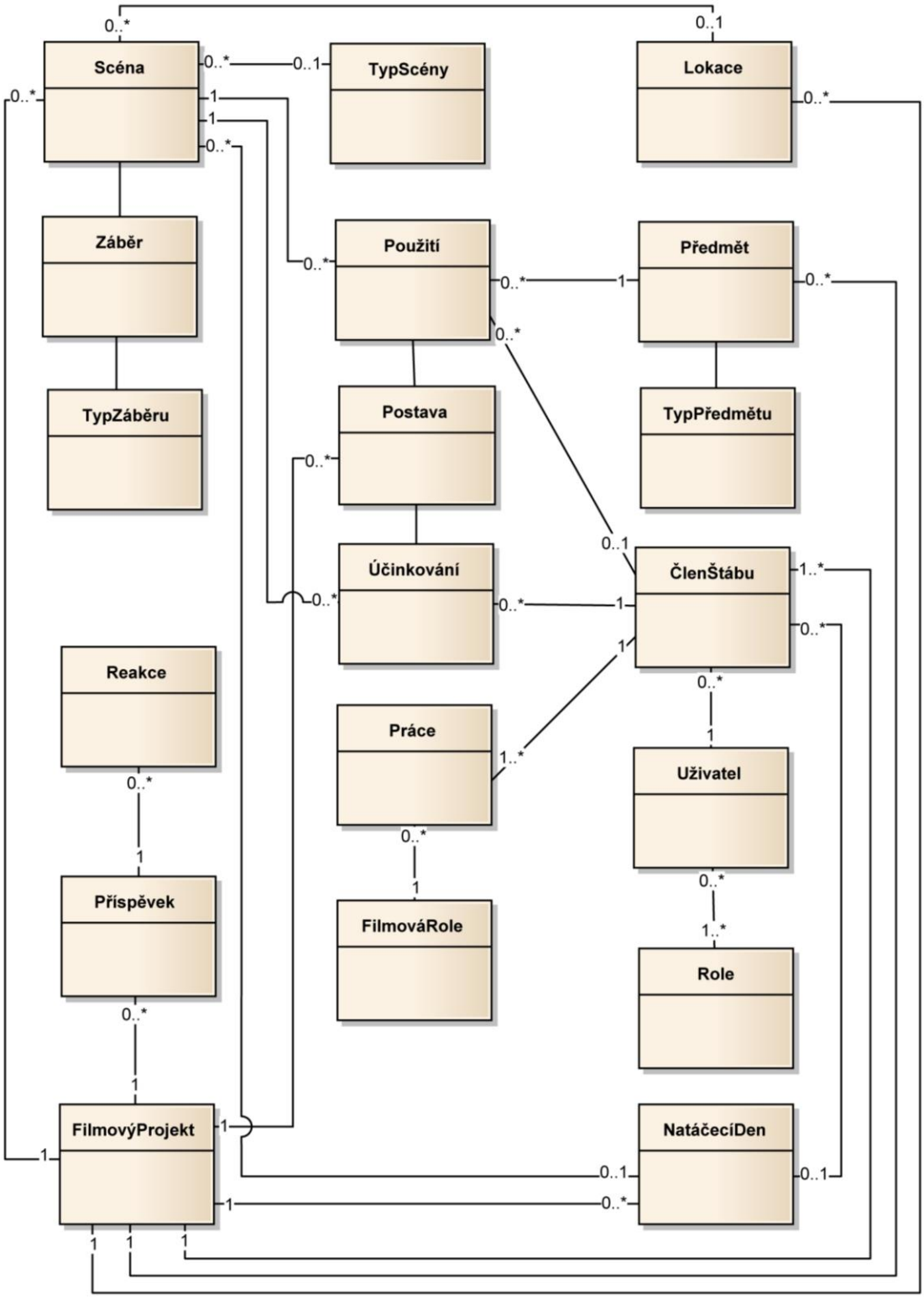
3.1.1.5 Diagram tříd

Diagram tříd (class diagram) popisuje typy objektů v systému a různé druhy statických vztahů, které mezi nimi existují.[5]

Diagram tříd k aplikaci vyvíjené v rámci této práce (obr. 3.4) znázorňuje filmový projekt (entita `FilmovýProjekt`), který v sobě uchovává veškeré informace o vytvářeném filmu – scény, lokace, postavy, natáčecí dny, členy štábu, apod. Předmětem (entita `Předmět`) je myšlena jakákoliv položka, která se bude používat během natáčení, může jít o rekvizity, kamery, mikrofony, kostýmy, atd. Zajímavý je problém dvou typů rolí – role a filmová role. Entita `Role` reprezentuje autoritu, která se přihlášenému objektu po celou dobu jeho přítomnosti nemění, jedná se o klasické případy rolí – user či admin. Zatímco filmová role (entita `FilmováRole`) je role, která se může měnit s přístupem uživatele k různým filmovým projektům. V jednom filmu může mít uživatel roli herec, zatímco v jiném může být režisérem.

[4] FOWLER, Martin. *Destilované UML*. 1. vyd. Praha: Grada, 2009, s. 53-56. Knihovna programátora (Grada). ISBN 978-80-247-2062-3.

[5] FOWLER, Martin. *Destilované UML*. 1. vyd. Praha: Grada, 2009, s. 53-56. Knihovna programátora (Grada). ISBN 978-80-247-2062-3.



Obrázek 3.4. Diagram tříd k tomuto projektu

3.2 Nefunkční požadavky

Aplikace, vyvíjená v rámci bakalářské práce, by měla splňovat následující nefunkční požadavky:

- Dostupnost,
- bezpečnost,
- škálovatelnost,
- rozšiřitelnost.

3.2.1 Dostupnost

Jelikož hlavní devízou tohoto softwaru by měla být schopnost organizace filmového projektu, musí být dostupný více uživatelům najednou bez nutnosti jakkoli zasahovat do stávajícího programového vybavení počítače.

3.2.2 Bezpečnost

Dalším neoddiskutovatelným požadavkem aplikace je její zabezpečení. Už z názvu této práce je patrné, že předmětem zájmu je nástroj, který usnadňuje tvůrčí činnost. Filmoví tvůrci budou chtít prostřednictvím této aplikace vytvářet díla, která jsou před samotným vydáním filmu (mnohdy i trvale) ryze privátního charakteru. Právě onu privátnost a soukromí musí vyvíjený program bezpodmínečně zajišťovat.

3.2.3 Škálovatelnost

V případě úspěchu aplikace je třeba počítat s nárůstem uživatelů, kteří ji budou používat. Je proto nezbytné, aby byl vyvíjený software dostatečně škálovatelný neboli pružný pro případný vzrůst poptávky.

3.2.4 Rozšiřitelnost

Softwarový produkt, který bude výsledkem bakalářské práce, je sám o sobě prototypem, jenž bude přímo vybízet k pozdějšímu zvyšování počtu funkčních požadavků. Proto je nasnadě navrhnout aplikaci tak, aby byla snadno rozšiřitelná a flexibilní vůči případným dodatečným změnám.

4 Design

Design aplikace vycházel z vymezených požadavků a závěrů vyvozených z analýzy. Na základě získaných poznatků a dosavadních zkušeností autora aplikace následně pokračoval výběr programovacího jazyka, frameworků, nástrojů a dalších technologií, které jsou použity pro samotnou implementaci a testování.

Dalším úkolem spadajícím do této kapitoly je návrh architektury aplikace, při kterém bylo mimo jiné zapotřebí zohlednit ony použité technologie.

4.1 Proč internetová aplikace?

Podstata výběru internetové aplikace oproti desktopové tkví v jednom z nefunkčních požadavků - dostupnosti. Program musí být dostupný více uživatelům najednou tak, aby nebylo nezbytné jakkoli zasahovat do stávajícího programového vybavení jejich počítačů. Desktopová aplikace, jakou je kupříkladu CeltX nebo Final Draft, umožňuje ukládání souborů do formátu k ní přidruženého. Tento soubor je pak možné rozesílat dle libosti prostřednictvím elektronické pošty, programů pro instant messaging, apod. Nevýhodou takového přístupu je nutnost přítomnosti stejné aplikace na každém stroji, kde se má takový soubor otevřít a dále na něm pracovat. Aplikace jako je Final Draft či CeltX jsou navíc produkty cílené specifické skupině uživatelů a tím pádem přímo vyžadují explicitní instalaci na jednotlivých strojích. Proto dávají někteří filmoví tvůrci (nutno podotknout, že i profesionální) přednost obyčejným textovým nebo tabulkovým editorům, které jsou obvyklou součástí operačního systému. Internetová aplikace je naopak instalovaná na jediném stroji – serveru, ke kterému přistupují všichni její uživatelé prostřednictvím např. internetového prohlížeče.

4.2 Použité technologie

Strategie volby jednotlivých nástrojů pro implementaci a testování se odvíjela od počátečního výběru platformy a programovacího jazyka. Z něj poté vycházela rozhodnutí o vývojovém prostředí, v němž bude implementace probíhat, použití či nepoužití frameworků, případně jakých frameworků a na kterém konkrétním místě. K vývoji webové aplikace také neodmyslitelně patří volba webového serveru, na nějž bude program nasazen.

4.2.1 Programovací jazyk a platforma

Rozhodování o programovacím jazyce, v němž bude aplikace napsána, nebyl proces příliš časově náročný. S ohledem na to, jakými jazyky se autor projektu během své krátké praxe dosud zabýval, došlo na výběr ze dvou alternativ – skriptovacího jazyka PHP a kompilovaného jazyka Java. Jelikož se programátor primárně věnuje platformě Java a navíc k ní zachovává vřelejší vztah, než je tomu u jazyka PHP, bylo nakonec rozhodnuto, že aplikace bude sestavena na platformě Java, konkrétně na její součásti Java Enterprise Edition[1].

Jako jedna s nejsilnějších stránek jazyku Java je často zdůrazňována jeho bezpečnost, jež je také jedním z bodů nefunkčních požadavků. Mezi základní bezpečnostní mechanizmy se dá uvést kupříkladu typová kontrola, dohled nad velikostí polí, správa paměti, apod. Cílem této práce není popsat výhody a nevýhody jednotlivých programovacích jazyků, každý z nich má své silné i slabé stránky a názory různých vývojářů bývají mnohdy dosti subjektivní.

4.2.2 Frameworky

Volba frameworků oproti předchozímu výběru nevycházela z jejich znalosti. Schopnost používání různých druhů frameworků je v dnešní době důležitým předpokladem každého programátora, pokud chce uspět ve sféře softwarového inženýrství. Cílem této bakalářské práce nebylo pouze vyvinout aplikaci, která by splňovala požadavky plynoucí z uskutečněného průzkumu. Jedním z dílčích záměrů práce bylo vzdělat se v moderních a hojně využívaných technologiích, mezi které se dnešní frameworky rozhodně řadí.

4.2.2.1 Základní struktura - Spring Framework

Základní struktura projektu byla vytvořena na základě frameworku Spring. Na tuto počáteční kostru se později nabalují další části aplikace s použitím jiných frameworků. Spring Framework, ve své verzi 3.2.2, však hraje roli hlavního průvodce při vývoji této aplikace.

Spring Framework je jedním z projektů softwarové společnosti SpringSource, jedná se o nástroj pro budování podnikových aplikací založený na specifikacích Javy EE[2]. Patří

[1] Java Enterprise Edition je platforma pro vývoj podnikových aplikací, napsaných v jazyce Java, vycházející ze standardní edice Javy - Java SE (podle [15]).

[2] SPRINGSOURCE. *SpringSource Community* [online]. [cit. 2013-04-22]. Dostupné z: http://www.springsource.org/spring-framework

v dnešní době k lídrům mezi frameworky pro vývoj aplikací běžících na platformě Java. Řada dnešních IT firem řeší své projekty právě pomocí tohoto prostředku, proto je také znalost Spring frameworku podmínkou mnoha podniků při výběru zaměstnanců na pozici Java programátora.

Spring Framework nebyl jedinou eventualitou při výběru podobného nástroje založeného na platformě Java. Do hry vstoupil také convention-over-configuration[3] framework Groovy on Rails (Grails). Grails je framework pro vývoj webových aplikací, který využívá dynamického programovacího jazyka Groovy. Právě kombinace dynamického jazyka a strategie convention-over-configuration zajišťuje podobným frameworkům jejich popularitu, protože dle dodavatelů i spokojených spotřebitelů zaručuje vysokou efektivitu práce. Taktika convention-over-configuration podle nich přináší do projektu onu efektivitu práce zejména na jeho začátku, kdy se mohou oprostit od řady konfigurací, jež jsou naopak jedním z typických rysů programů psaných ve Spring Frameworku. Projekt tak při svém zrodu nepotřebuje téměř žádná explicitní nastavení a jeho základní kostra je pomocí několika příkazů sestavena během pár minut. Tato strategie ale předpokládá dodržování konvencí, které ovšem nejsou implicitně nijak kontrolovány. Použitím dynamického skriptovacího jazyka Groovy může zase projekt ztratit na bezpečnosti, kterou by mu přinesl silně typovaný jazyk Java. K zachování zabezpečení aplikace musí být v průběhu implementace bezpodmínečně přítomny skutečně kvalitní průběžné testy. Aplikace psaná ve Springu vyžaduje na svém začátku definování několika konfigurací ať už prostřednictvím XML souborů nebo anotací. Z tohoto důvodu není vytvoření hlavní struktury projektu záležitostí několika minut, i když právě výše zmíněné anotace přinesly s příchodem nové specifikace Java Servlet 3.0 výrazný pokrok i v tomto směru.

Výběr frameworku je znovu otázkou velmi subjektivního charakteru, kdy byla sice zvážena mnohá pro a proti u obou kandidátů, ale opět převážily spíše vývojářovy osobní preference nad každým z nich. Spring Framework byl vybrán z důvodu možnosti psát aplikaci v programovacím jazyce Java, který je na rozdíl od Groovy silně typovaným jazykem, jenž udržuje programátorovu fantazii více na uzdě.

Samotný framework Grails je shodou okolností též produktem z rodiny SpringSource a jeho základní struktura je do velké míry inspirována Spring Frameworkem, z čehož plyne,

[3] Convention over configuration je přístup založený na upřednostňování konvencí před konfiguracemi.

že obeznámenost s principy frameworku Spring jsou dobrou vstupní výhodou pro začátky s vývojem v Groovy on Rails.

Základními vlastnostmi jádra Springu je velmi propracovaný způsob implementace návrhového vzoru Dependency Injection (dále DI) přes technologii springovských bean a aspektově orientované programování (dále AOP) přítomné v modulu Spring AOP. Součástí Spring Frameworku je také front-end modul Spring MVC.

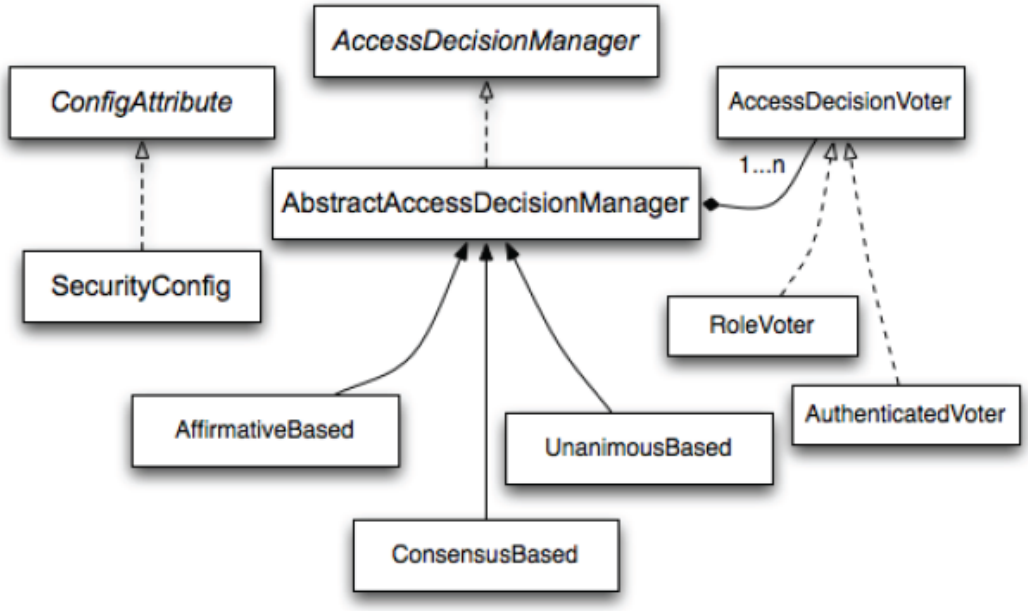
Pro potřeby projektu byla použita verze Spring Framework 3.2.2.

4.2.2.2 Spring Security

Další z velice sofistikovaných frameworků, jehož výrobcem je znovu firma SpringSource. Tento nástroj pomáhá programátorovi vyřešit bezpečnost aplikace. Zprostředkovává jak autentizaci, tak i autorizaci, nicméně je spíše prostředkem ke speciálně uzpůsobené implementaci zabezpečení, jež vychází ze specifických potřeb konkrétního projektu. Obsahuje řadu rozhraní, které ukazují vývojáři cestu, jak aplikaci zajistit proti vnějším útokům. Spring Security má v sobě zabudovanou i základní implementaci těchto rozhraní, jež ale slouží pouze pro velmi obecné případy autentizace či autorizace. Čím specifičtější strategie zabezpečení je, tím více je nutné navrhnout vlastní řešení daných rozhraní.

Hlavním rozhraním, které poskytuje autentizaci, je `AuthenticationManager`. Implementace rozhraní `AuthenticationManager` autentizaci sice poskytuje, ale neprovádí ji. Deleguje proces autentizace do jednotlivých implementací rozhraní `AuthenticationProvider`, a až na základě jejich odpovědi `AuthenticationManager` sestaví nebo nesestaví autentizační objekt, jenž poskytuje informace o právě přihlášeném objektu a je k dispozici po celou dobu, dokud se daný objekt zase neodhlásí. V případě, že `AuthenticationManager` autentizační objekt nesestaví, znamená to, že autentizace byla neúspěšná.

Za autorizaci je zodpovědné rozhraní `AccessDecisionManager`, deklarující metodu `decide(…)`, jejíž implementace rozhoduje, zdali udělit příchozímu objektu přístup na základě jednotlivých instancí tříd, tzv. voterů, které implementují rozhraní `AccessDecisionVoter` (obr. 4.1).



Obrázek 4.1. Autorizace ve Spring Security[4]

Z předchozích dvou odstavců je jasné, že právě konkrétní implementace mohou být vývojářem různým způsobem uzpůsobeny podle jeho vlastní potřeby. Programátor může kombinovat vlastní implementace s těmi, co jsou přítomné již v samotném frameworku.

Pro potřeby projektu byla použita verze Spring Security 3.1.3.

4.2.2.3 Hibernate ORM

Framework pro objektově relační mapování, který je používán v datové vrstvě aplikací. Hibernate mj. implementuje specifikaci JPA (Java Persistence Api)[5], jež je součástí Javy EE. Prostřednictvím Hibernate frameworku se jednotlivé doménové objekty mapují za přispění anotací do databáze.

Jedním z nejdůležitějších rozhraní frameworku je `SessionFactory`, jehož konkrétní implementace v sobě uchovává informace týkající se zdroje dat, dialektu, znakového kódování či balíčků, v nichž se mají hledat jednotlivé doménové objekty, které jsou mapovány do databáze. Rozhraní `SessionFactory` také obsahuje funkcionalitu pro získání objektu typu `Session`, který poskytuje metody pro komunikaci s databází. Jedná se kupříkladu o jednotlivé CRUD operace, ale také metody pro řízení transakcí.

[4] SPRINGSOURCE. *Spring Security: Reference Documentation* [online]. [cit. 2013-04-19]. Dostupné z: http://static.springsource.org/spring-security/site/docs/3.2.x/reference/springsecurity-single.html

[5] Java Persistence API poskytuje persistentní model pro objektově relační mapování (podle [15]).

Spring Framework v sobě v rámci svého datového modulu zahrnuje podporu pro Hibernate ORM. Hibernate ve spojení se Springem tak radikálně zjednodušuje práci s vytvořením spolupráce mezi projektem a relační databází.

Pro potřeby projektu byla použita verze Hibernate 4.1.9.

4.2.2.4 Prezentační vrstva – JSTL, Apache Tiles a Bootstrap

V prezentační vrstvě byla využita technologie JSP, dále pak také template framework Apache Tiles a framework Bootstrap.

JSP (JavaServer Pages) je technologie užívaná v rámci projektu, pomocí které lze kombinovat klasický HTML kód s programovým kódem psaným v jazyce Java. Samotné JSP je v projektu implementováno prostřednictvím tagové knihovny JSTL. JSTL svými tagy nahrazuje javovský programový kód, díky čemuž zůstává daná stránka v klasickém XML formátu.

Apache Tiles je další framework figurující na prezentační vrstvě. Pomocí tohoto frameworku je možné vytvořit základní layout, ze kterého poté vycházejí jednotlivé měnící se view implementace.

Bootstrap je ideální volbou pro vývojáře, kteří nemají po ruce žádného grafika. Při zvolení správné kombinace barev tento framework vygeneruje kolekci souborů s kaskádovými styly a javascriptovým kódem, jež společnými silami vytvoří graficky velmi pěkně vypadající stránky. Programátor se nemusí v tomto ohledu o nic starat, pokud pro něj vzhled aplikace není klíčovou záležitostí.

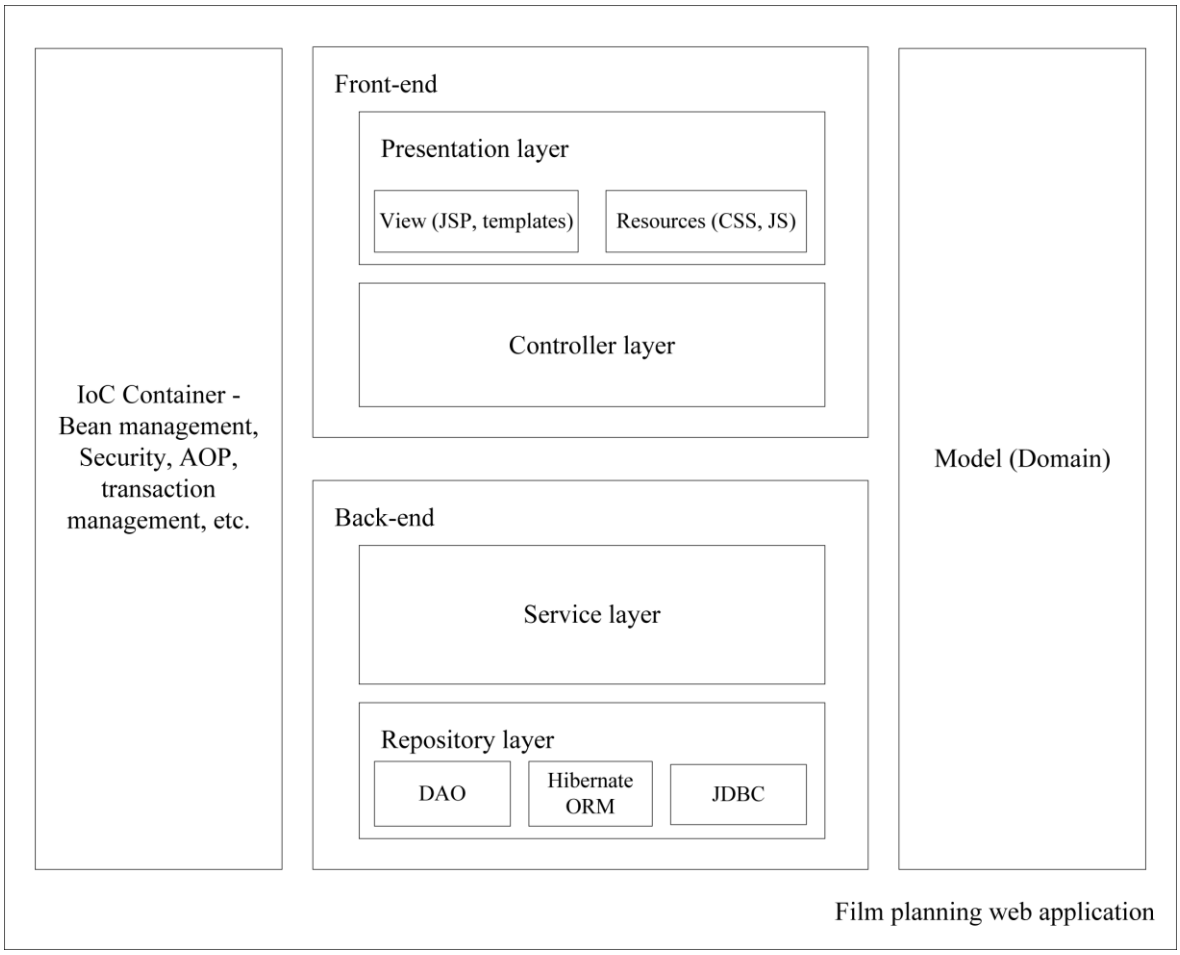
4.2.3 Ostatní technologie

Pro vývoj podobných robustních aplikací je nezbytností použití některého z build manažerů, jež usnadňuje správu struktury aplikace. Pro tyto účely byl v aplikaci vybrán produkt společnosti Apache - Apache Maven. Webovým serverem této aplikace je Apache Tomcat.

4.3 Architektura aplikace

Architektura aplikace (obr. 4.2) přímo vychází z použitých technologií, které ke konkrétnímu sestavení struktury programu vývojáře navádějí. Jedná se o klasickou aplikaci klient-server, přičemž celá serverová část je umístěna na jediném stroji. Aplikace se dělí na front-endovou a back-endovou část. Tyto dvě části jsou rozděleny do dalších logických

vrstev, přičemž každá vrstva má svou vlastní úlohu, které by se měla držet. Nad jednotlivými logickými vrstvami je umístěn IoC Container.



Obrázek 4.2. Architektura aplikace

4.3.1 Front-end

Na front-endu aplikace figuruje jeden z modulů Spring Frameworku - Spring MVC. Již z názvu samotného lze odvozit, že struktura front-endu aplikace vychází z obecně známé softwarové architektury model-view-controller.

4.3.1.1 Model

Jako model jsou označována data, putující napříč celou aplikací, v níž jsou různým způsobem zpracovávána. Modelové, resp. doménové objekty (DTO – domain transfer object) v sobě nezahrnují žádnou aplikační ani business logiku. Třídy, jejichž instancemi jsou právě doménové objekty, obsahují pouze deklarace instančních proměnných a k nim přidružených getterů a setterů. Každá taková třída, označená anotací `@Entity`, je

prostřednictvím ORM frameworku mapována do databáze jako tabulka, její instanční proměnné pak reprezentují tabulkové sloupce.

4.3.1.2 View

Prezentační vrstva využívá pro své fungování template-frameworku Apache Tiles. Aplikace obsahuje dvě základní vzorové stránky, jedna patří administrační části a druhá té uživatelské. Vzorové stránky obsahují základní logické rozvržení, ze kterého vychází každý view, jež od příslušné vzorové stránky dědí. Informace o tom, jaká view implementace dědí od kterého vzoru, je zanesena v definicích uvnitř XML konfiguračních souborů frameworku Apache Tiles. Vzorová stránka také obsahuje odkazy na javascriptové soubory a soubory s kaskádovými styly, jejichž obsah je z velké části výsledkem generování, které provedl framework Bootstrap.

4.3.1.3 Controller

Controllery jsou komponenty, jejichž hlavním úkolem je předávat řízení aplikace dle informací, jež jim byly zaslány v argumentech jejich metod. Controllery jsou volány instancí třídy `DispatcherServlet`. Po dokončení úlohy vrací logické jméno view implementace, která se má zobrazit. Každý controller obsahuje odkaz na servisní objekt, jemuž předává řízení, je-li zapotřebí jakékoliv business logiky k vyřešení některého úkolu.

4.3.2 Back-end

Tato část aplikace je rozvržena do dvou hlavních logických vrstev: Datové a servisní. V servisní vrstvě je schována business logika aplikace, zatímco datová vrstva zajišťuje práci s daty.

4.3.2.1 Datová (repository) vrstva

Datovou vrstvou je myšlena ta část aplikace, která komunikuje s databází. Pro onu komunikaci jsou ve struktuře programu přítomné tzv. DAO (data access object) objekty. Díky frameworku Hibernate ORM je práce s daty velmi intuitivní a zjednodušená.

V projektu se používá databáze PostreSql.

4.3.2.2 Servisní vrstva

Jak již bylo řečeno, servisní vrstva má v sobě uchovánu business logiku aplikace. Každá servisní třída obsahuje závislosti na objekty, které zprostředkovávají operace nad databází - DAO.

4.3.3 IoC Container

Vrstva, jež je umístěna nad logickými vrstvami aplikace. Zahrnuje správu bean (DI), odkud dochází k vytvoření a spojení všech objektů, čímž se utváří celá aplikace.

4.3.3.1 DI ve Spring Frameworku

Podstatou DI (v tomto projektu i obecně) je odebrat třídám zodpovědnost za získávání konkrétních objektů, které potřebují ke své činnosti. Místo toho, aby si třídy vytvářely objekty samy, přenechávají tuto zodpovědnost jiné komponentě, jejíž totožnost jim nemusí být známa. Snižují se tak závislosti mezi jednotlivými částmi systému.

V aplikacích založených na Spring Frameworku se o řízení DI stará tzv. aplikační kontext, který je v jádru Springu reprezentován rozhraním `ApplicationContext`. Konkrétní implementace rozhraní `ApplicationContext` načítá definice bean a spojí je dohromady. Je plně zodpovědná za vytváření a spojování objektů, které utvářejí celou aplikaci. Spring přináší několik implementací rozhraní `ApplicationContext`, přičemž každá se liší pouze v tom, jakým způsobem je provedena konfigurace.[6]

V případě tohoto projektu je konfigurace kombinována bean definicemi v XML souborech a anotacemi uvedenými v programovém kódu.

4.3.4 AOP

AOP (aspektově orientované programování) je technika, která umožňuje snímat funkcionalitu napříč celou aplikací a obalit ji znovupoužitelnými komponentami. Každá tato komponenta je zodpovědná za specifickou část funkcionality, může se jednat např. o řízení transakcí, logování či zabezpečení.[7]

Právě např. transakce nebo i kontrola přístupu k objektům jsou v tomto projektu řešeny prostřednictvím springovské implementace AOP.

[6] WALLS, Craig. *Spring in action*. 3rd ed. Shelter Island: Manning, c2011, s. 9. ISBN 19-351-8235-8.
[7] WALLS, Craig. *Spring in action*. 3rd ed. Shelter Island: Manning, c2011, s. 10. ISBN 19-351-8235-8.

4.3.5 Zabezpečení

Kapitolou samou pro sebe je zabezpečení aplikace, které zasahuje do obou dvou částí – front-endu i back-endu. Bezpečnost je zajišťována prostřednictvím frameworku Spring Security popsaném v dřívějších kapitolách. Dle potřeb projektu bylo na různých místech buď využito vnitřních implementací frameworku nebo došlo k vlastní implementaci, pokud si ji daná část aplikace vyžadovala.

5 Implementace

Tato část práce zahrnuje způsob implementace jednotlivých úseků a vrstev, které byly popsány v předchozí kapitole o designu. K vývoji bylo rovněž zapotřebí vybrat vhodné vývojové prostředí.

Aplikace obsahuje uživatelskou i administrační část. V uživatelské části mohou zaregistrovaní uživatelé vytvářet a dále organizovat své filmové projekty nebo být jejich členem. Mohou provádět vše, co je popsané ve funkčních požadavcích. Do administrační části mohou přistupovat pouze uživatelé s administrátorskou autoritou. Administrátor upravuje různá nastavení aplikace, např. může přidávat, upravovat či mazat typy scén či záběrů dle filmařských specifikací a konvencí.

5.1 Vývojové prostředí (IDE)

I v průběhu rozhodování nad vývojovým prostředím, ve kterém se samotná praktická část uskuteční, byla brána na zřetel především programátorova zběhlost v jednotlivých z nich. Dané IDE muselo samozřejmě splňovat i požadavek, jenž se týká jeho podpory použitého programovacího jazyka, frameworků a dalších nástrojů. Na základě předchozích nároků postoupila do užší volby opět pouze dvě vývojová prostředí – Eclipse a Netbeans. S Netbeans IDE se vývojář tohoto projektu za své programátorské praxe setkal pouze jednou, zatímco prostředí Eclipse je mu známo daleko lépe, jeho prostřednictvím dokonce několik projektů již vyvinul. Proto bylo i pro tuto práci zvoleno vývojové prostředí Eclipse, konkrétně jeho verze Eclipse JEE Juno.

Během fáze vývoje došlo v tomto ohledu k drobné změně. Vzhledem k přímé podpoře vybraných technologií byla implementace přesunuta do vývojového prostředí Spring Tool Suite. Jedná se o prostředí založené na Eclipse s tím, že zajišťuje přímou podporu programování ve frameworku Spring.

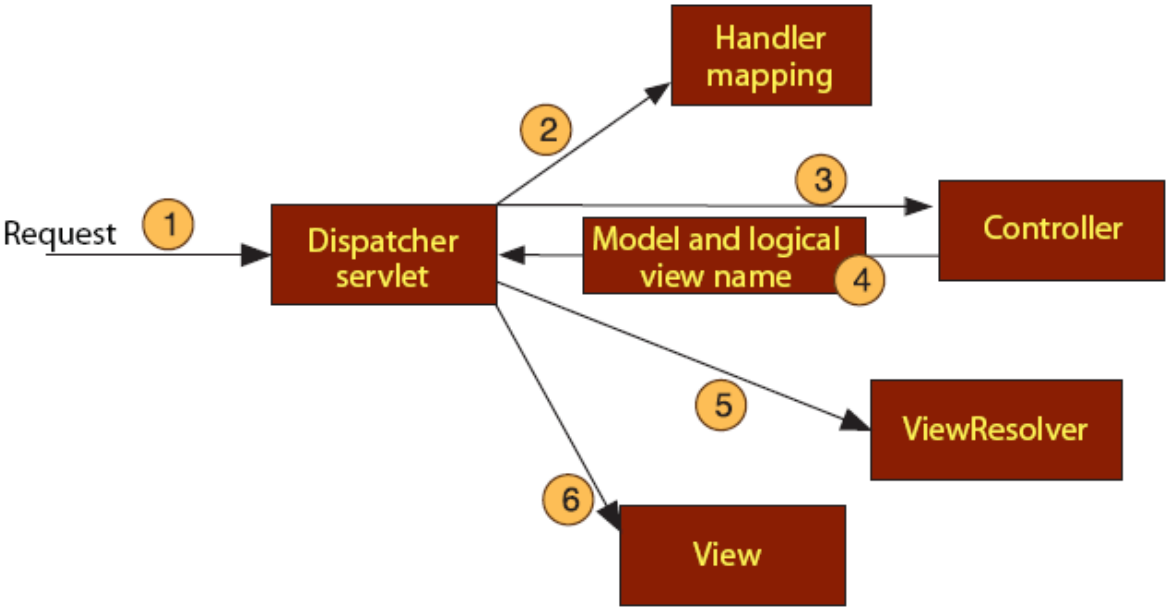
5.2 Adresářová struktura

Veškerý webový obsah (javascript, CSS, JSP) a konfigurační soubory jsou umístěné v root adresáři webové aplikace. Všechen programový kód s javovskými třídami a rozhraními, lokalizační soubory pro překlad aplikace do různých jazyků a unit testy jsou zařazeny v adresáři src. Vrstvy, které byly charakterizovány v návrhu (designu) aplikace, jsou

logicky odděleny prostřednictvím javovských balíčků (např. servisní vrstva je reprezentována balíčkem `cz.tomsovsky.bc.filmplanning.service`)

5.3 Implementace front-endu

Princip fungování front-endu je vysvětlen na putování http požadavku a jeho odezvy napříč aplikací (obr. 5.1). Středobodem front-endu je springovská třída `DispatcherServlet`, přes kterou prochází veškeré požadavky i odezvy. Jakmile požadavek dorazí do instance této třídy, je jejím úkolem rozhodnout na základě předložené URL adresy, do jakého controlleru ho deleguje. Toto rozhodování probíhá za pomoci vybraných implementací rozhraní `HandlerMapping`, které mají za úkol mapovat URL adresu k příslušnému controlleru. V této aplikaci je použita instance třídy `DefaultAnnotationHandlerMapping`, která mapuje URL do konkrétních metod controlleru prostřednictvím anotace `@RequestMapping`, jež bývá umístěna nad deklaracemi controllerů a jejich metod (obr. 5.2). Ve chvíli, kdy je patřičná metoda controlleru nalezena, přepošle jí instance třídy `DispatcherServlet` daný požadavek. Dozor nad aplikací přebírá controller, jehož jediným úkolem je podle typu přijatých informací rozhodnout, kam předat řízení aplikace společně s daty, které obdržel. Může kupříkladu komunikovat s back-endem aplikace, který zpracuje controllerem odeslaná data a po provedení svých úkolů je následně vrátí zpět. Jakmile metoda controlleru dokončí svou úlohu, zašle data společně s logickým jménem view implementace, která se má zobrazit, zpět do instance třídy `DispatcherServlet`. `DispatcherServlet` následně mapuje prostřednictvím instance třídy `TilesViewResolver` logické jméno na příslušnou stránku čili konkrétní view implementaci (JSP), která zobrazí obdržená modelová data. Instance třídy `TilesViewResolver` mapuje logické jméno na příslušnou stránku prostřednictvím tiles definic obsažených v konfiguračních souborech, jež vycházejí z frameworku Apache Tiles.



Obrázek 5.1. Princip front-endu prostřednictvím Spring MVC[1]

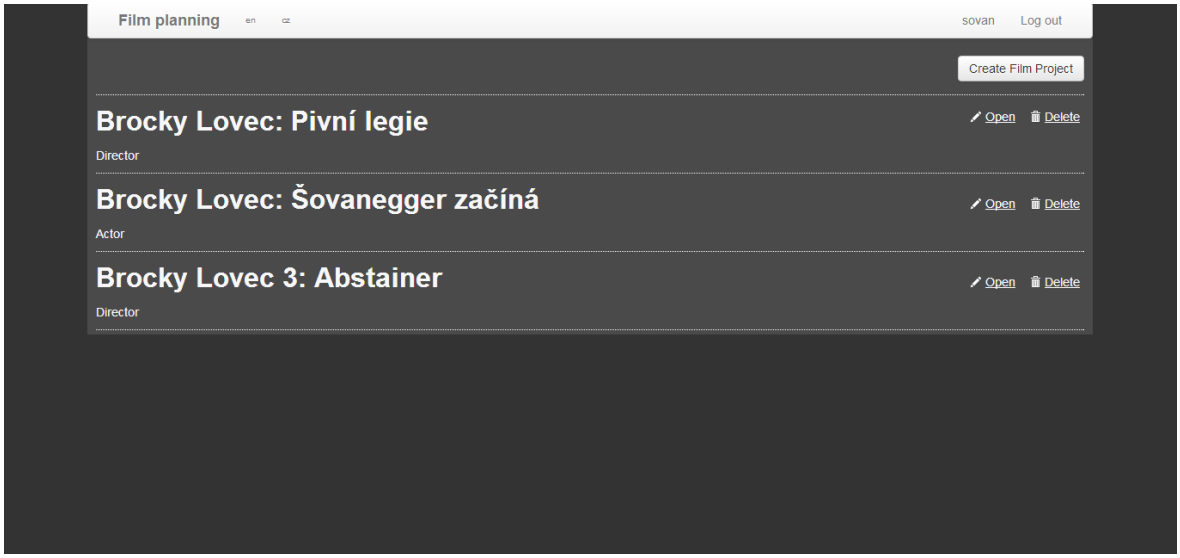
```
@Controller
@RequestMapping(value="/filmprojects")
public class FilmProjectController {
    private final FilmProjectService filmProjectService;

    @Autowired
    public FilmProjectController(FilmProjectService filmProjectService) {
        this.filmProjectService = filmProjectService;
    }

    @RequestMapping(value="/add", method=RequestMethod.GET)
    public String getAddForm(Model model) {
        model.addAttribute(new FilmProject());
        return "filmprojects/add";
    }
```

Obrázek 5.2. Ukázka controlleru

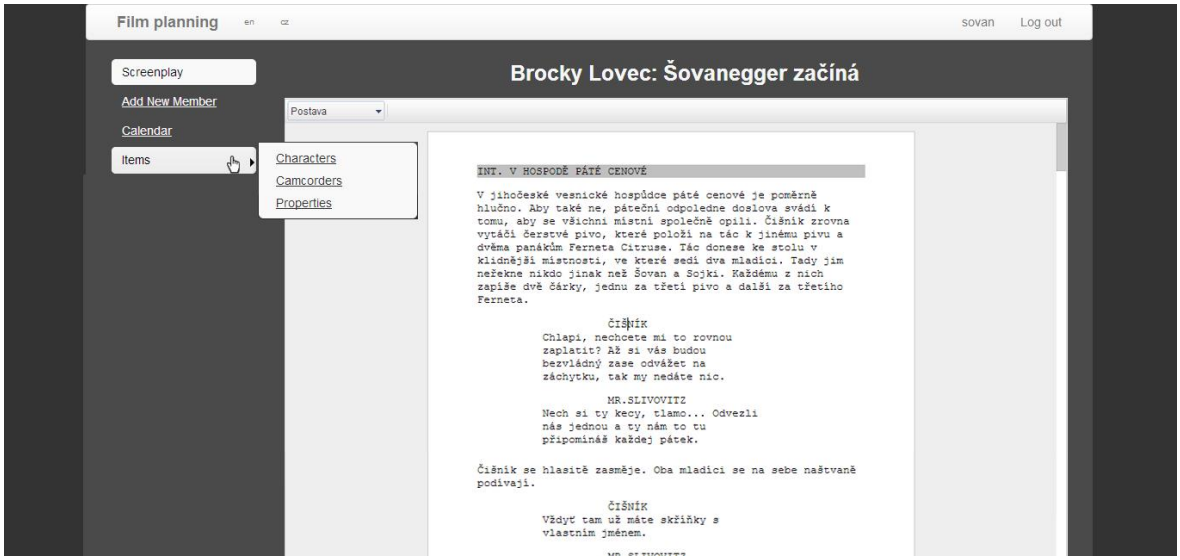
5.3.1 GUI

Grafické uživatelské rozhraní aplikace je nastaveno tak, aby byla práce s ním co nejintuitivnější. Uživatel se přihlásí do aplikace a hned se mu zobrazí všechny projekty, kterých se účastní (obr. 5.3). Jakmile jeden z projektů otevře, zobrazí se mu poslední podoba literárního scénáře a nabídka, podle které může s projektem dále pracovat (obr. 5.4).

[1] WALLS, Craig. *Spring in action*. 3rd ed. Shelter Island: Manning, c2011, s. 165. ISBN 19-351-8235-8.



Obrázek 5.3. GUI: Úvodní strana po přihlášení



Obrázek 5.4. GUI: Psaní scénáře

5.4 Implementace back-endu

Každá DAO třída (obr. 5.5) této aplikace implementuje rozhraní `DomainDao`[2], jež obsahuje deklarace metod reprezentujících CRUD operace – uložení, úprava, odstranění a získání záznamu. Konkrétní implementace DAO rozhraní v rámci této aplikace potřebuje pro spolupráci s databází udržovat instanci typu `SessionFactory`, pomocí které získává jednotlivé instance třídy `Session`, s podporou kterých provádí databázové operace. Aby

[2] DAO třída neimplementuje rozhraní `DomainDao` přímo, ale přes rozhraní, které odpovídá konkrétní doméně, aby tak bylo dosaženo dobré vývojářské taktiky programování proti rozhraní a ne proti implementaci. Tzn. např. `FilmProjectDaoHibernate` implementuje rozhraní `FilmProjectDao`, které rozšiřuje rozhraní `DomainDao`.

každá taková třída nemusela udržovat svůj vlastní odkaz na instanci `SessionFactory`, je vytvořena společná deklarace této reference ve třídě `HibernateDao`, od níž poté jednotlivé DAO třídy dědí, díky čemuž je instance typu `SessionFactory` společná pro všechny DAO třídy.

```
@Repository(value="filmProjectDao")
public class FilmProjectDaoHibernate extends DaoHibernate
                                implements FilmProjectDao {
    @Override
    @AclUpdate
    public void save(FilmProject filmProject) {
        currentSession().save(filmProject);
    }

    @Override
    public void update(FilmProject filmProject) {
        currentSession().update(filmProject);
    }

    @Override
    public FilmProject getById(long id) {
        return (FilmProject) currentSession().get(FilmProject.class, id);
    }
```

Obrázek 5.5. Ukázka DAO třídy

Servisní třídy (obr. 5.6) pak obvykle uchovávají odkazy na DAO. Pokud přes tento odkaz konkrétní servisní metoda přistupuje k databázi, tak je označena anotací `@Transactional`, která zajistí, že se při každém volání metody spustí transakce. Při dokončení metody pak dochází ke commitu či rollbacku dle úspěšnosti databázových operací.

```
@Service(value="filmProjectService")
public class FilmProjectServiceImpl implements FilmProjectService {
    @Autowired
    private FilmProjectDao filmProjectDao;

    @Override
    @Transactional
    public void saveFilmProject(FilmProject filmProject) {
        filmProject = prepareAttributes(filmProject);
        filmProjectDao.save(filmProject);
    }

    @Override
    @Transactional
    public void updateFilmProject(FilmProject filmProject) {
        filmProjectDao.update(filmProject);
    }
```

Obrázek 5.6. Ukázka servisní třídy

5.5 Bean management

Jak již bylo objasněno, za vytváření a spojování objektů, utvářejících celou aplikaci, je plně zodpovědná springovská implementace rozhraní `ApplicationContext`, čímž je dosaženo návrhového vzoru DI, též dříve vysvětleného. Spojování a vytváření objektů provádí aplikační kontext tak, že načítá definice bean a spojí je dohromady.

Definice bean jsou v této aplikaci umístěny v jednotlivých XML konfiguračních souborech nebo prostřednictvím anotací přímo v programovém kódu.

XML konfigurační soubory, nezbytné pro chod celé aplikace, jsou kvůli dobré orientaci logicky rozvrženy do částí:

- Servlet,
- service,
- repository,
- aop,
- security,
- acl.

Definice bean přes anotace je prováděna jejím umístěním nad deklaraci třídy v programovém kódu. Každý controller je tedy označen anotací `@Controller`, servisní třída anotací `@Service` a DAO třída anotací `@Repository`.

Objekty, na kterých je třída závislá, jsou označeny anotací `@Autowired`, umístěnou nad deklarace referencí v dané třídě. Anotace `@Autowired` zajistí, že takto označená instanční proměnná je spojena s beanou stejného jména jako je jméno oné proměnné.

Při spuštění aplikace je odstartován mechanizmus vkládání závislostí.

5.6 Zabezpečení aplikace

Program využívá pro zabezpečení framework Spring Security, který poskytuje jak autentizaci, tak i autorizaci. Aplikace pro oba procesy využívá vnitřní implementace frameworku, ale na některých místech musela být tato základní implementace explicitně rozšířena pro specifické potřeby aplikace.

5.6.1 Autentizace

Pro záměry této aplikace byla v její uživatelské části použita autentizace, která je vnitřní součástí frameworku Spring, Pro administrační úsek byla poté vytvořena vlastní implementace rozhraní `AuthenticationManager`.

5.6.2 Autorizace

Autorizace je v aplikaci řešena na třech úrovních:

- URL autorizace,
- autorizace metod (method security),
- ACL (access control list).

Přístup k jednotlivým URL adresám je ošetřen v konfiguračních souborech Springu, kde je každému vzoru URL adresy přidělena autorita, která k ní má povolený přístup. Vzory URL jsou vytvářeny pomocí regulárních výrazů.

Autorizace metod je řešena anotacemi, umístěnými nad každou metodou, kterou je potřeba zabezpečit. I jim je přidělena autorita, jež může k těmto metodám přistupovat. Anotace, které zajišťují zabezpečení metod, jsou umístěny nad deklaracemi metod servisních tříd.

Třetí úroveň se týká kontrolování přístupů k objektům. Ta je prováděna prostřednictvím doplňkového modulu frameworku Spring Security - Spring Security ACL. Přístup k objektům je sledován opět uvnitř servisní vrstvy.

6 Testování

Testování aplikace probíhalo v několika fázích. Nutno dodat, že poslední etapa (testování uživatelem) bude vykonána až po uzávěrce této práce.

6.1 Ruční testování

Nejprimitivnější způsob otestování, zdali je část kódu funkční. Jedná se o testování vývojářem samotným. Programátor při dokončení vývoje každého modulu ověřil, zdali skutečně funguje tím způsobem, že zkoumal chování své aplikace přímo v jeho grafickém rozhraní. Např. jakmile dovršil vývoj vytváření typu scény, zkusil typ scény vytvořit přímo v aplikaci, a pokud se typ scény uložil správně do databáze, bylo ruční testování tohoto modulu označeno za úspěšné.

6.2 Unit testování

Daleko sofistikovanějším způsobem testování aplikace v průběhu jejího vývoje je Unit testování. Jedná se o testování jednotek, tedy tříd a metod. Každá třída má k sobě přidruženou testovací třídu, jež ji testuje, tudíž kupříkladu třída `FilmProjectServiceImpl` je testována testovací třídou `FilmProjectServiceImplTest`, resp. metoda `addFilmProject(…)` je testována testovací metodou `testAddFilmProject(…)` těchto tříd.

K testování byl použit testovací framework JUnit 4.

6.3 Testování uživatelem

Testování tohoto typu bude vykonáno až po odevzdání bakalářské práce. Proto zde nejsou jeho výsledky uvedeny.

Princip je založen na iterativní metodě vývoje. Testujícím uživatelům aplikace, kterými jsou filmoví tvůrci Petr Hála a Filip Vorel, je předložena sada úkolů, jimiž mají uvnitř aplikace projít. Prostřednictvím osobního pohovoru poté zhodnotí, zdali nebyla nalezena chyba jakéhokoliv charakteru, jak obtížné bylo daný úkol splnit, případně doplní, jakým způsobem by jednotlivé části aplikace mohly být pozměněny. Na základě výsledků pohovorů je aplikace upravena a znovu předložena k otestování.

7 Závěr

Vzhledem k provedenému průzkumu trhu se mi potvrdil úvodní předpoklad, že většina současných amatérských či začínajících tvůrců nepoužívá žádný specializovaný preprodukční software a pokud ano, jedná se především o desktopovou verzi aplikace CeltX, která však nepodporuje sdílený přístup k filmovému projektu.

Na základě této analýzy jsem vytvořil specifikaci požadavků na aplikaci, podle kterých jsem se řídil při návrhu a následné implementaci programu. Vyvinul jsem prototyp aplikace se základními funkcemi, který je schopný úvodního otestování externími uživateli, amatérskými filmovými tvůrci Filipem Vorlem a Petrem Hálou.

V dohledné době je cílem udělat z prototypu aplikace hotový produkt, schopný nasazení. Až dle jeho úspěchu či neúspěchu se rozhodnu, zdali budou do projektu implementovány další moduly, vhodné pro usnadnění filmařovy práce.

I kdyby tato aplikace nedosáhla úspěchu, nebyl pro mě čas strávený nad vytvořením bakalářské práce zbytečně odevzdaný. Míra získaných poznatků o současných hojně využívaných technologiích byla velmi vysoká a zajistila mi zpětnou vazbu v samotné praxi, na níž mě má vysokoškolské studium připravit.

8 Literatura

[1] DOHNAL, Lubor. Úvodem do scénáristiky a dramaturgie. Studijní materiál filmové školy. PRAHA: Famu

[2] LONG, Ben a Sonja SCHENK. *Velká kniha digitálního videa*. Vyd. 1. Překlad Magdalena Kolínová. Brno: Computer Press, 2005, 478 s. ISBN 80-251-0580-6.

[3] KŘENA, Bohuslav a Radek KOČÍ. *Úvod do softwarového inženýrství: IUS*. Studijní opora. [online]. Brno, 2006. Vysoké učení technické, Fakulta informačních technologií. 100 s.

[4] Object Management Group, Inc. *Business Process Model and Notation (BPMN): Version 2.0*. [online]. 508 s. Dostupné z: http://www.omg.org/spec/BPMN/2.0/.

[5] FOWLER, Martin. *Destilované UML*. 1. vyd. Praha: Grada, 2009, 173 s. Knihovna programátora (Grada). ISBN 978-80-247-2062-3.

[6] DOLEŽAL, Jan, Pavel MÁCHAL a Branislav LACKO. *Projektový management podle IPMA*. 2., aktualiz. a dopl. vyd. Praha: Grada, 2012, 526 s. Expert (Grada). ISBN 978-80-247-4275-5.

[7] PECINOVSKÝ, Rudolf. *Návrhové vzory*. Vyd. 1. Brno: Computer Press, 2007, 527 s. ISBN 978-80-251-1582-4.

[8] FOWLER, Martin. *Destilované UML*. 1. vyd. Praha: Grada, 2009, 173 s. Knihovna programátora (Grada). ISBN 978-80-247-2062-3.

[9] HOGAN, Brian P. *HTML5 a CSS3: výukový kurz webového vývojáře*. Vyd. 1. Brno: Computer Press, 2011, 272 s. ISBN 978-80-251-3576-1.

[10] KOTHAGAL, Koushik. Java Brains: Free Java training videos. [online]. Dostupné z: http://javabrains.koushik.org/

[11] SpringSource. [online]. Dostupné z: http://www.springsource.org/

[12] Hibernate - JBoss Community. [online]. Dostupné z: http://hibernate.org/

[13] Overview (Java EE 6). [online]. Dostupné z: http://docs.oracle.com/javaee/6/api/

[14] Overview (Java Platform SE 6). [online]. Dostupné z: http://docs.oracle.com/javase/6/docs/api/

[15] ORACLE CORPORATION. *Oracle | Hardware and Software, Engineered to Work Together* [online]. [cit. 2013-04-19]. Dostupné z: http://www.oracle.com/technetwork/java/javaee/overview/index.html

[16] WALLS, Craig. *Spring in action*. 3rd ed. Shelter Island: Manning, c2011, xxiii, 400 p. ISBN 19-351-8235-8.

9 Seznam obrázků a tabulek

10 Příloha

Případ užití:	Potvrdit členství ve filmovém projektu
Aktéři:	potenciální nový člen, systém
Vstupní podmínky:	Potenciální nový člen obdrží zprávu s pozváním do filmového projektu a odkazem na samotné připojení k projektu.
Výstupní podmínky:	
Hlavní scénář:	1. Potenciální nový člen si přečte zprávu a otevře stránku, na kterou zpráva odkazuje. 2. Systém zobrazí stránku s informacemi o uživatelových právech na filmovém projektu a ptá se ho na definitivní potvrzení. 3. Potencionální nový člen odsouhlasí svá práva. 4. Potencionální nový člen definitivně potvrdí svou účast na filmovém projektu. 5. Systém zařadí uživatele do filmového projektu.
Zvláštní případy:	2.a Nový člen není zaregistrován v systému. 1. Systém zobrazí registrační formulář. 2. Potencionální nový člen se zaregistruje a přihlásí.
Výskyt:	častý

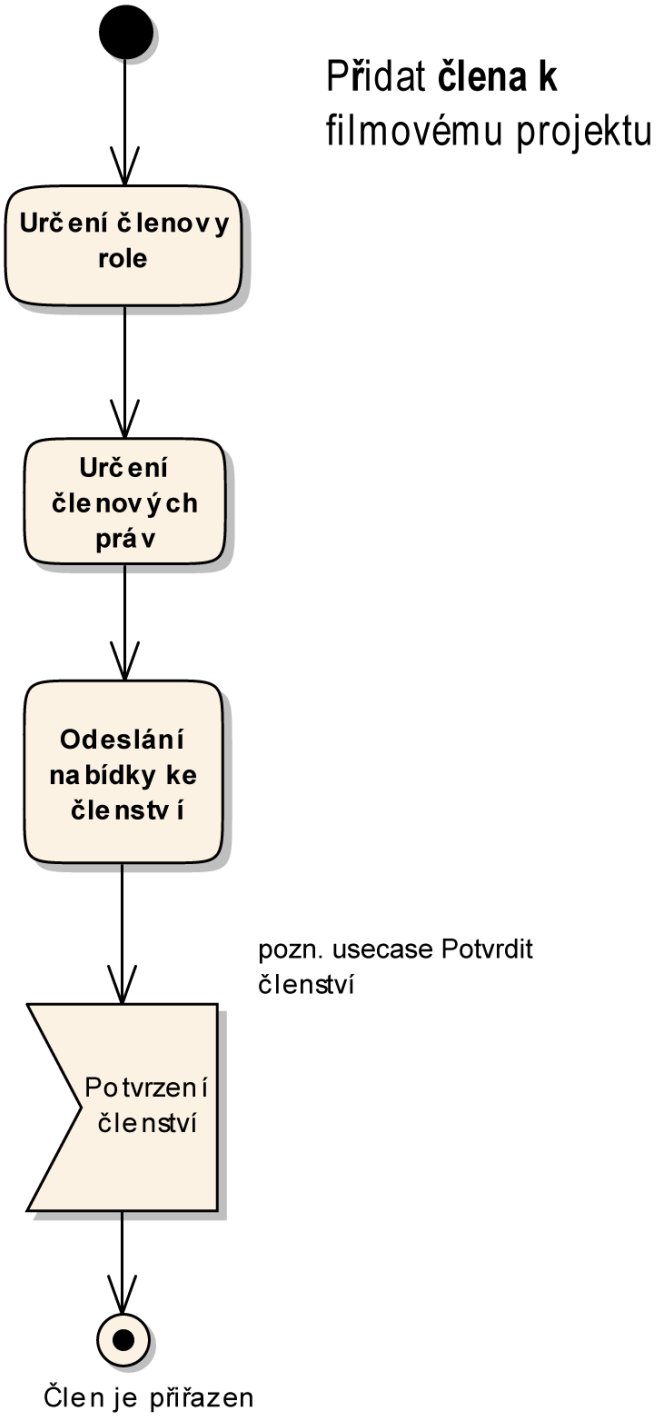
Tabulka 10.1. Scénář k případu užití „Potvrdit členství ve filmovém projektu“

Případ užití:	Přidat člena do filmového projektu
Aktéři:	člen, systém
Vstupní podmínky:	Člen je přihlášen a má práva k operaci přidání člena do filmového štábu
Výstupní podmínky:	
Hlavní scénář:	1. Člen otevře formulář pro přidání člena do filmového projektu 2. Systém zobrazí stránku s výběrem filmové role nového člena, výběrem jeho práv a také pole s kontaktem (e-mail) na danou osobu 3. Člen vyplní kontakt, na který se má odeslat nabídka k účasti na filmovém projektu 4. Člen vybere novému členovi jeho filmovou roli 5. Člen vybere práva, kterými bude nový člen disponovat 6. Člen potvrdí předchozí volby a vyšle požadavek k odeslání nabídky k účasti ve filmovém projektu novému členovi 7. Systém odešle nabídku na uvedenou adresu
Zvláštní případy:	
Výskyt:	častý

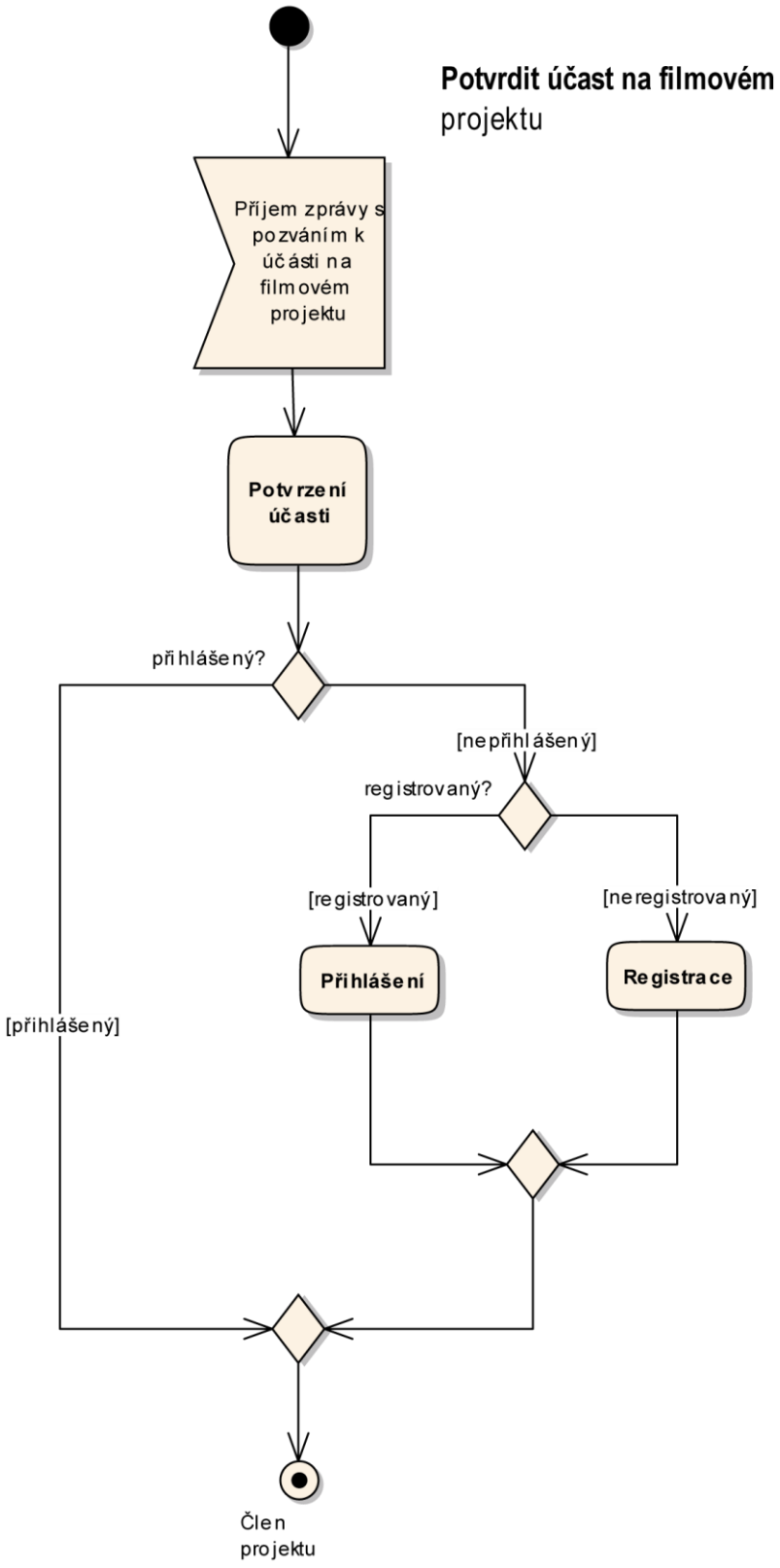
Tabulka 10.2. Scénář k případu užití „Přidat člena do filmového projektu“

Případ užití:	Naplánování filmového dne
Aktéři:	člen, systém
Vstupní podmínky:	Člen je přihlášen a má práva k operaci naplánování filmového dne
Výstupní podmínky:	
Hlavní scénář:	1. Člen otevře formulář pro vytvoření natáčecího dne 2. Systém zobrazí stránku pro vytvoření natáčecího dne 3. Člen vyplní datum natáčecího dne a čas začátku natáčení 4. Člen vyplní scény, které se budou v daný den natáčet 5. Systém na základě vybraných scén automaticky doplní ostatní informace k natáčecímu dni (zúčastnění herci, použité předměty, apod.) 6. Člen uloží natáčecí den
Zvláštní případy:	
Výskyt:	častý

Tabulka 10.3. Scénář k případu užití „Naplánování filmového dne“



Obrázek 10.1. Diagram aktivit „Přidat člena k filmovému projektu“



Obrázek 10.2. Diagram aktivit „Potvrdit účast na filmovém projektu“