



## ZADÁNÍ BAKALÁ SKÉ PRÁCE

<b>Název:</b>	MobChar - desktopová aplikace pro Pána jeskyn pro Dra ī doup
<b>Student:</b>	Jan Horá ek
<b>Vedoucí:</b>	Ing. Zden k Rybola
<b>Studijní program:</b>	Informatika
<b>Studijní obor:</b>	Softwarové inženýrství
<b>Katedra:</b>	Katedra softwarového inženýrství
<b>Platnost zadání:</b>	Do konce letního semestru 2017/18

### Pokyny pro vypracování

Vytvo te desktopovou aplikaci pro Pána jeskyn ve h e Dra ī doup spolupracující s balí kem pro Dra ī doup v rámci mobilní aplikace pro správu postav v r zných hrách na hrdiny MobChar.

Hlavní funkce aplikace:

- p íprava dobrodružství - p íprava map, p edm t , cizích postav a nestv r,
- správa šablon pro p edm ty, kouzla, schopnosti, efekty a modifikátory,
- import a export šablon a dobrodružství pro balí ek Dra ī doup v aplikaci MobChar.

V souladu se standardy softwarového inženýrství:

- analyzujte požadavky na desktopovou aplikaci a stávající aplikaci MobChar v etn balí ku pro Dra ī doup ,
- vhodným zp sobem navrhn te architekturu aplikace a zp sob realizace požadavk ,
- implementujte v souladu s provedeným návrhem,
- implementaci pat i n zdokumentujte a otestujte,
- vytvo te uživatelskou p íru ku pro práci s touto aplikací.

### Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.  
vedoucí katedry

prof. Ing. Pavel Tvrďk, CSc.  
d kan



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



Bakalářská práce

**MobChar - desktopová aplikace pro Pána jeskyně pro Dračí doupě**

*Jan Horáček*

Vedoucí práce: Ing. Zdeněk Rybola

7. května 2017



---

## **Poděkování**

Rád bych poděkoval svému vedoucímu práce Ing. Zdeňkovi Rybolovi za odborný dohled a neustálé vedení mé práce správným směrem. Dále bych rád poděkoval Šárce Sochorové za vytvořených grafiky pro mojí aplikaci. V neposlední řadě bych rád poděkoval svým kolegům Matějovi Shánělovi a Šárce Weberové, za dobrou spolupráci a užitečné řady. Nakonec bych poděkoval své rodině a přátelům, že i v době vytváření práce mě stále podporovali a pomáhali.



---

## **Prohlášení**

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

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

V Praze dne 7. května 2017

.....

České vysoké učení technické v Praze  
Fakulta informačních technologií

© 2017 Jan Horáček. 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

Horáček, Jan. *MobChar - desktopová aplikace pro Pána jeskyně pro Dračí doupě*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

---

# Abstrakt

Cílem této práce je hráčům Dračího doupěte, a zejména Pánu jeskyně, usnadnit vytváření dobrodružství, postav a předmětů pomocí jednoduché desktopové aplikace. Hlavní výhodou aplikace je možnost exportování dobrodružství pro mobilní aplikaci MobChar pro Pána jeskyně a exportování do přehledného HTML formátu vhodné pro tisk a hraní Dračího doupěte bez mobilů a jiných zařízení. V práci byla vytvořena aplikace, ve které je možné pomocí grafického rozhraní vytvářet rozsáhlá dobrodružství plná lokací, map, předmětů, nestvůr a mnohého dalšího.

V první části je vypracovaná důkladná analýza problém a již existujících aplikací. Na základě analýzy je vytvořen návrh aplikace, který se převážně zaměřuje na nejdůležitější funkcionality aplikace, což je tvorba XML a HTML výstupních souborů.

Na základě analýzy a návrhu je vytvořena implementace aplikace. V poslední fázi je aplikace důkladně otestována, aby byla zaručena výsledná kvalita a splnění požadavků.

Všechny zdrojové kódy, dokumentace, spustitelný program a uživatelská příručka jsou umístěny na přiloženém CD.

**Klíčová slova** desktopová aplikace, hra na hrdiny, MobChar, Dračí doupě, rozšíření, herní scéna, tvorba dobrodružství, Python, SQLite, XML

---

# Abstract

**Keywords** desktop application, heroes games, MobChar, Dračí doupě, extension, desktop games, scenario creation, Python, SQLite, XML

---

# Obsah

<b>Úvod</b>	<b>1</b>
<b>1 Úvod do problematiky</b>	<b>3</b>
1.1 Cíl práce . . . . .	3
1.2 Dračí doupě . . . . .	3
<b>2 Současný stav</b>	<b>7</b>
2.1 Projekt MobChar . . . . .	7
2.2 Existující aplikace . . . . .	8
<b>3 Analýza</b>	<b>11</b>
3.1 Požadavky . . . . .	11
3.2 Business procesy . . . . .	14
3.3 Případy užití . . . . .	16
3.4 Doménový model . . . . .	21
3.5 Návrh řešení architektury . . . . .	23
<b>4 Návrh</b>	<b>27</b>
4.1 Model databáze . . . . .	27
4.2 Model XML souborů . . . . .	29
4.3 Model komunikace . . . . .	31
4.4 Model nasazení . . . . .	33
<b>5 Implementace</b>	<b>35</b>
5.1 Použitý programovací jazyk . . . . .	35
5.2 Využité knihovny . . . . .	35
5.3 Použité nástroje . . . . .	38
5.4 Adresářová struktura projektu . . . . .	39
5.5 Zajímavé části implementace . . . . .	40
5.6 Vzniklé problémy při implementaci . . . . .	42

<b>6 Testování</b>	<b>45</b>
6.1 Jednotkové testy . . . . .	45
6.2 Uživatelské testování . . . . .	47
<b>Závěr</b>	<b>53</b>
<b>Literatura</b>	<b>55</b>
<b>A Seznam použitých zkratek</b>	<b>57</b>
<b>B Obsah přiloženého CD/USB</b>	<b>59</b>

---

# **Seznam obrázků**

3.1	Business proces vytváření šablon . . . . .	14
3.2	Business proces vytváření dobrodružství . . . . .	15
3.3	Proces vytváření mapy . . . . .	16
3.4	Tabulka pokrytí požadavků případy užití . . . . .	17
3.5	Případy užití pro šablony . . . . .	17
3.6	Případy užití pro dobrodružstvím . . . . .	19
3.7	Případy užití pro mapu . . . . .	21
3.8	Analytický doménový model stromové struktury . . . . .	22
3.9	Analytický doménový model dobrodružství . . . . .	23
3.10	Model architektury . . . . .	24
4.1	Základní model databáze . . . . .	28
4.2	Model XML souhoru dobrodružství . . . . .	30
4.3	Model XML souhoru příšery . . . . .	31
4.4	Model komunikace pro hlavní obrazovku . . . . .	32
4.5	Model nasazení . . . . .	33
5.1	Adresářová struktura projektu . . . . .	39



---

# Úvod

Dračí doupě je populární hra na hrdiny, která vznikla jako česká verze hry Dungeons & dragons. V České Republice si získala velkou oblibu zvláště mezi hráči stolních her a počítačových RPG her.

Hráči hrají za své hrdiny, které si na začátku dobrodružství vytvořili, putují světem a zažívají rozličná dobrodružství, která pro ně Pán jeskyně připraví. Celá družina si na začátku dobrodružství vytvoří své postavy podle toho, do které postavy by se rádi vžili. Někteří si vytvoří postavy, které svým chováním či názory jsou podobné samotným hráčům, někteří zase Dračí doupě využijí jako únik do úplně jiného světa, kde nemusí být sami sebou a chovají se úplně jinak a nepředvídatelně. Nicméně během celého dobrodružství se všichni hráči vtělí do svých fiktivních postav a veškerá rozhodnutí rozhodují, jako by právě stáli v magickém světě s kouzelnou hůlkou v ruce a rozhodovali, jestli svého zajatce zabijí či ho nechají žít. Každý hráč si může vybrat z rozličného výběru ras, které se ve světě vyskytují, a zvolit si, jakému povolání se budou věnovat. Ale ať si hráč vybere mocného trpasličího válečníka, rychlého a úskočného hobitího zloděje nebo moudrého a mocného elfského kouzelníka, budou ho čekat složitá rozhodnutí, která určí, jakým směrem se dobrodružství bude dále ubírat.

Dračí doupě se lehce liší od klasických stolních her, kde plán hry máte pouze jeden a toho se musíte držet. Zde celý příběh vymýslí další hráč, takzvaný Pán jeskyně, který při hře celé dobrodružství řídí a popisuje hráčům. Veškerá rozhodnutí, která neučiní samotní hráči, rozhodne Pán jeskyně. Příprava propracovaných a zábavných dobrodružství je velice složitá a zabere mnoho hodin času. Je zapotřebí vytvořit přehledné poznámky a mapy, které při hře umožní všem hráčům dobře pochopit situaci a oblast, ve které se právě nachází jejich postavy. Cílem této bakalářské práce je tento proces usnadnit a zkrátit dobu přípravy, aby se Pán jeskyně mohl věnovat převážně hraní s ostatními spoluhráči.

Aplikace vznikala souběžně s mobilní aplikací MobChar pro Pány jeskyně[1],

ve které je možné veškeré informace o dobrodružství přehledně zobrazit a prezentovat hráčům. V balíčku aplikací již existuje mobilní aplikace MobChar pro hráče, která slouží pro zaznamenávání osobního deníku hráče[2]. Společně tyto aplikace mají vytvořit základnu, pro hraní Dračího doupěte.

## Struktura práce

V první kapitole je popsán úvod do problematiky. Jsou zde popsána základní pravidla Dračího doupěte a pravidla vytváření dobrodružství. Jsou zde také popsány základní mechanizmy hraní a interakce mezi hráči a Pánem jeskyně a základy tvoření dobrodružství.

Druhá kapitola se věnuje analýze současného stavu. Důležitou částí je analýza již existujících aplikací, ať už se jedná o aplikace, které by tato aplikace měla nahradit a nebo aplikace MobChar, se kterými bude aplikace spolupracovat.

Třetí kapitola je věnována analýze problematiky. Jsou zde popsány veškeré požadavky na aplikaci a zadefinované činnosti, které uživatelé s aplikací budou provozovat.

Na základě třetí kapitoly je ve čtvrté kapitole vytvořen návrh architektury a struktury tříd. Jsou zde detailně popsány vrstvy architektury a komunikace mezi nimi.

V páté kapitole je popsána problematika, týkající se samotné implementace. Jsou zde popsány problémy, které vznikly na základě návrhu architektury a jejich řešení. Nachází se zde zajímavé části implementace a také jsou zde dopodrobna popsány použité technologie, zvolený programovací jazyk a využité knihovny.

Poslední šestá kapitola se věnuje testování. Popis druhů testů, detailní popis průběhu testování a řešení některých nalezených problémů.

# Úvod do problematiky

Tato kapitola se zabývá základním úvodem do problematiky. Ve stručnosti představí hru Dračí doupě a její základní pravidla a principy hrání. Část této kapitoly je také věnována popisu průběhu vytváření dobrodružství.

## 1.1 Cíl práce

Cílem této práce je vytvořit desktopovou aplikaci, která rozšíří již existují balíček mobilních aplikací MobChar o nové funkcionality. Hlavním účelem aplikace je vytváření dobrodružství pro Dračí doupě a vytváření šablon pro mobilní aplikace MobChar. Pomocí strukturovaných XML souborů bude možné přenášet rozsáhlá dobrodružství do mobilní aplikace.

Někteří hráči Dračího doupěte si stále potrpí pouze na papírovou verzi dobrodružství a proto aplikace mimo jiné bude umožňovat exportovat veškeré šablony a dobrodružství do formátu HTML, jehož formát bude přizpůsoben pro případný tisk.

## 1.2 Dračí doupě

Dračí doupě vzniklo roku 1990 pod vedením nakladatelství Altar [3]. Hra se svými pravidly inspirovala ve velice populární americké hře Dungeons & Dragons, některé mechanismy si však upravila podle svého a mnoho principů zjednodušili. V roce 1990 vznikla oficiální pravidla verze 1.0, která se od té doby velice změnila a v nynější době již existuje verze pravidel 1.6. I když autoři hry se snažili pravidla udržovat aktuální a veškeré podmětné připomínky hráčů do nové verze zapracovat, mnoho hráčů si pravidla lehce upravili pro jejich potřebu, díky čemuž vzniklo velké množství herních mutací což mnoha hráčům ještě umocnilo zážitek ze hry.

### 1.2.1 Rozdělení hráčů

Při vytváření herní skupiny čeká hráče důležité rozhodnutí. Musí mezi sebou zvolit jednoho hráče, který bude ostatní celým dobrodružství provázet, takzvaného Pána jeskyně. Jeho úkolem je připravit celé dobrodružství do posledních detailů a následně ho převyprávět ostatním hráčům. Zpravidla Pánum jeskyně bývá kreativní člověk, který dokáže vymyslet zajímavé, někdy až bláznivé zápletky, díky čemu ostatní hráči budou zažívat stále nové a neohrané dobrodružství a neopakovatelný zážitek ze hry.

Ostatní hráči se stanou součástí dobrodružství jako hrdinové, kteří dobrodružství prožívají a společně jako družina prochází celým světem. Na začátku hry si vytvoří své postavy, které si až do konce hry nemůžou změnit. Tyto postavy můžou mít různé názory a chování, které by však hráč měl dodržovat během celé hry. Jak se však jeho postava bude rozhodovat, záleží už pouze na něm.

### 1.2.2 Zlaté pravidlo

Jedním ze základních a neměnných pravidel Dračího doupěte a obecně her na hrdiny je „Hráč není postava a postava není hráč“. Ačkoliv toto pravidlo může znít velice banálně a samozřejmě, mnoho hráčů s ním má veliký problém. Většina hráčů si své postavy vytvoří tak, že jejich běžné chování se velice liší od nich samotných, což znamená, že i ve všech situacích, ať se to zdá jakkoliv nepříjemné pro své spoluhráče, se musí zachovat jako postava. Spoustě hráčů také dělá problém, nevyužívat informace, které jeho postava nezná ale hráč ano.

Zlaté pravidlo se netýká pouze hráčů, ale je i velice důležité pro Pána jeskyně. I když ve svém rozhodování má velikou volnost, neměl by nijak upřednostňovat některé hráče, jen pro to, že je velmi dobrě zná. Vše by se mělo řídit pravidly a předem připraveným dobrodružstvím.

### 1.2.3 Příprava dobrodružství

Jedním z nejdůležitějších a taky časově nejnáročnějších povinností Pána jeskyně je příprava dobrodružství. Příprava dobrodružství vyžaduje dlouhé hodiny přípravy jednotlivých lokací, map, předmětů, ostatních postav a také celého příběhu. Tento proces je velice složitý a vyžaduje jistou míru zkušenosti a dobré představivosti. Čím poutavější a zábavnější dobrodružství Pán jeskyně vytvoří, o to lepší zážitky ostatní hráči budou při hraní zažívat a rádi si příště přijdou zahrát znovu.

Příprava jednotlivých lokací je velmi náročná věc, musí se připravit mapa celé oblasti, navrhnut správně silné neprátele, aby družina byla schopná je s trochou šikovnosti porazit a na druhou stranu aby nebyli příliš jednoduchým cílem pro jejich zbraně. Nutné je také vymyslet propracovaný příběh, proč že to družina vyrazila zrovna do té nejzapadlejší jeskyně v celé říši a snaží

se získat svitek s pár čarami a v neposlední řadě také veškeré nové vybavení, které hráči můžou objevit a vylepšit tak své postavy.

Samozřejmě Pán jeskyně nemůže mít předem rozmyšlené všechny možné scénáře, které dokáže družina vymyslet, ale je dobré být připraven na co nejvíce možností, aby Pán jeskyně nemusel během hraní příliš improvizovat, čímž zamezí chybám a zbrklým rozhodnutím.

Ve výsledku pak pro jednovečerní sezení má příběh spoustu popsaných papírů textem, mnoho nakreslených map a spoustu další poznámek. Navíc tyto poznámky se během hraní mění a rozšiřují, například když družina porazí zlobry, cí vybere jejich tajný poklad. Při velkém množství změn se poznámky stávají nepřehledné, což by přenos velká části dobrodružství do aplikace velice zjednodušil.

#### 1.2.4 Průběh hry

Celá hra probíhá formou dialogu mezi hráči a Pánem jeskyně. Každou situaci uvede Pán jeskyně. Popíše hráčům kde se nachází, co všechno vidí a případně co se právě stalo (např. „z tajných dveří v chodbě za vámi vyskákali dva obří skřetové a valí se na vás“). Hráči pak musí na tuto situaci reagovat, samozřejmě v omezeném čase, než se k nim skřeti dostanou. Poradí se spolu nebo některý z pohotových hráčů sám učiní rozhodnutí a podnikne například protiútok.

Musíte si však uvědomit, že i takto jednoduchá a častá situace sebou nese velké množství herních mechanismů, které se musí vykonat. Většina z nich se týká Pána jeskyně. Pán Jeskyně musí rozhodnout, zda skřeti trefili správný moment, aby zaskočili družinu a naopak nevyskočili v tu nejnehodnější dobu, nebo jestli se skřetům podařilo družinu překvapit natolik, že si útoků skřetů ani nevšimla.

Většina těchto událostí záleží na pravidlech, na schopnostech družiny a neprátele, na náhodě (v podobě hodu kostkami) a v neposlední řadě na rozhodnutí Pána jeskyně. PJ nahlédne do pravidel pro Pána jeskyně[4] a zjistí, jak hluční jsou skřeti a naopak jak pozorná je družina. Na základě nalezených údajů pak hodí kostkou a zjistí výsledek. Přeci jenom skřeti nepatrí mezi nejtěžší nestvůry a proto pravděpodobnost tichého přepadení je velice mizivá, ale přesto se vždy o tom musí rozhodnout.

Samozřejmě i hráči se musí držet pravidel [5], na základě kterých se dá určit, jak moc a zdali vůbec bude protiútok proti skřetům účinný. Hráč dohledá v pravidlech potřebné informace a zpravidla hodí kostkou, tolíkrát, kolikrát mu určí Pán jeskyně. Pán jeskyně si také hodí kostkou a na základě pravidel určí výsledek pokusu. Samozřejmě pokud se hráč chce pokusit o něco nesmyslného (např. rozeběhnout se po stěně a pokusit se dostat za skřety), Pán jeskyně má poslední slovo v tom, zdali je daná akce možná. Ve většině případů však Pán jeskyně určuje pouze o jaký druh akce se jedná a o výsledku rozhodnout pravidla a kostky.

## 1. ÚVOD DO PROBLEMATIKY

Po vyhodnocení těchto akcí přichází na řadu ostatní hráči nebo případně skřeti a pokračuje se stejným principem dále. Po dokončení boje si hráči rozdělí kořist, pokusí se vyléčit své zranění a pokračují dále k dalším překážkám a dalším úspěchům.

# KAPITOLA **2**

---

## **Současný stav**

V této kapitole se práce věnuje popisu současného stavu. Popisuje již existující aplikace, se kterými program bude spolupracovat. Dále popisuje souběžně vznikající balíček pro Pány jeskyně, se kterým úzce spolupracuje. V poslední části se věnuje již existující aplikacím, které by tato aplikace měla nahradit.

### **2.1 Projekt MobChar**

Aplikace MobChar vznikla jako samostatná aplikace, která měla sloužit jako osobní deník hráče pro hru Dračí doupě. Postupem času se rozširovala a měnila základní struktura. Nynější stav již umožňuje vytváření rozšíření pro další hry na hrdiny jako jsou například Dungeons & Dragons. V této chvíli jsou již plně funkční rozšíření pro Dračí doupě a Dungeons & Dragons a vznikají další.

#### **2.1.1 Balíček pro Dračí doupě**

Jedním z již vzniklých balíčků pro MobChar je balíček Dračí doupě. Jedná se o balíček pro správu osobního deníku. Mezi základní funkcionality balíčku patří zaznamenávání atributů, kouzel, schopností a efektů postavy, zaznamenávání veškerého vybavení postavy a zaznamenávání všech událostí, které se v aplikaci provedli. Všechny tyto entity se dají exportovat do XML souborů a také je možné tyto soubory do aplikace nainstalovat. Pokud se šablony připraví ještě před začátkem dobrodružství, velice to usnadní práci s aplikací. Přeci jenom mobilní telefon není nevhodnější zařízení, pro vypisování všech informací do jednotlivých šablon. Pokud jsou však šablony předpřipravené a importované do aplikace, práce je velice jednoduchá a intuitivní. Poté vám zážitek ze hry již žádné složité připravování rušit nebude. V nynější době je balíček rozširován v rámci bakalářské práce Šárky Weberové [2]. Mezi nové funkcionality například patří komunikace s aplikací pro Pány jeskyně.

### 2.1.2 Balíček pro pána jeskyně

V rámci aplikace MobChar vzniká další rozšíření, ne však pro osobní deník hráče, ale pro Pána jeskyně. Jednoduchá aplikace, ve které budou k dispozici údaje o všech hráčích a právě probíhajícím dobrodružství. Protože tvorba dobrodružství je poměrně náročná činnost, která vyžaduje napsání mnoha textů, připravení velkého množství šablon a další časově náročné činnosti, bylo by velice nepraktické toto dobrodružství vytvářet na mobilním zařízení. Proto vzniká desktopová aplikace DeskChar, jejíž hlavním cílem je vytváření dobrodružství a šablon pro mobilní aplikaci.

## 2.2 Existující aplikace

Díky rozsáhlé komunitě okolo HnH, jako jsou Dračí doupě a Dungeons & Dragons, vzniklo již mnoho aplikací, které mají hraní či přípravu dobrodružství usnadnit. Je možné najít desítky různých editorů map, které jsou více či méně povedené a propracované. Mezi nejzajímavější patří Dungeon Painter Studio [6], které se převážně zaměřuje na tvorbu rozsáhlých map, nebo například Donjon[7], který se zaměřuje na generování náhodných map a další objektů, jako jsou například obchody, truhly, počasí a další. Vybral jsem tyto dva systémy, protože z řady všech existujících aplikací jsou nejvíce propracované a hlavně stále živé a pravidelně aktualizované. Žádná z těchto aplikací nenabízí mobilní prohlížeč dobrodružství, nebo export do formátu, který by bylo možné na mobilních zařízeních jednoduše prohlížet. Nicméně PDF soubor formátu A4 s mnoha údaji není příliš přehledný, obzvlášt na malých obrazovkách.

**Dungeon Painter Studio** je aplikace, která se zaměřuje na tvorbu map a následné generování PDF souborů. Existuje online verze, dostupná je zdarma, která však nenabízí veškeré funkcionality plné verze. Ale i v této zjednodušené verzi dokážete vytvořit velmi propracované mapy. Plná verze je již desktopová aplikace, která je placená (stojí přibližně 400 korun). Je možné zde vytvořit opravdu propracované a poutavé mapy včetně stínování, světelných efektů a mnoha dalšího. Je však až zbytečně složitá a pro většinu Pánů jeskyně tvorba mapy bude příliš zdilouhavá. Vytvoření jedné mapy zabere mnoho času a ve většině případů takto propracované mapy nejsou pro hraní důležité. Bohužel v základní verzi není možné s mapou propojit předměty a příšery z dobrodružství, což by měla být jedna ze základních funkcionalit při tvorbě mapy.

**Donjon** je online aplikace, která se zaměřuje na generování náhodných lokací a událostí. Jedná se o poměrně unikátní aplikaci. Lze si zde navolit základní nastavení, jako je velikost mapy, velikost místností a další. Finální verzi mapy již upravit však nelze. K mapě se vygeneruje popis jednotlivých místností v poměrně nepřehledné tabulce. Tento formát však bez dodatečných úprav v obrázkovém editoru nelze rozumně vytisknout. Aplikace nabízí generování další

## 2.2. Existující aplikace

prvků, jako jsou například poklady, obchody a další, bohužel však tyto prvky se již nedají propojit s vygenerovanou mapou. Další nevýhodou je podpora pouze anglického jazyka textů, což mnoha Pánům jeskyně může působit problém. Jedná se o zajímavý nápad, který určitě usnadní vytváření dobrodružství, samostatný však nestačí.



# KAPITOLA 3

---

## Analýza

Třetí kapitola se věnuje analýze celé problematiky. Jsou zde analyzovány veškeré požadavky na aplikaci a také jednotlivé scénáře využití aplikace neboli business procesy. V poslední části je zde dopodrobna rozebrán datový model celé aplikace a navrhnut model architektury.

### 3.1 Požadavky

Na začátku každého projektu je jedním z prvních činností sběr požadavků na aplikaci. Je nutné přesně definovat veškeré funkční a nefunkční požadavky. Na základě těchto požadavků bude následně vznikat podrobná analýza a taká návrh aplikace. Na základě definování požadavků se dají vytvořit první odhady ceny celého projektu, ale tyto odhady jsou velice orientační a nepřesné. Je důležité požadavky správně zadefinovat, protože se jedná o první stavební kámen celé aplikace.

#### 3.1.1 Funkční požadavky

Funkční požadavky vymezují veškerou funkcionalitu, kterou by aplikace měla umožňovat. Funkční požadavky jsou rozděleny do dvou částí, funkcionalita pro hráče a funkcionalita pro pána jeskyně. Tyto sekce jsou rozdělené, protože popisují stejný program, ale z pohledu jiného uživatele.

##### 3.1.1.1 Funkcionalita pro hráče

Funkční požadavky, které program bude poskytovat zejména pro hráče pro práci se šablonami.

**Tvorba a editace šablon pro MobChar** Program bude umožňovat vytvářet nové šablony pro mobilní aplikaci MobChar. Bude také umožňovat editovat stávající šablony nebo případně i vybrané šablony smazat. Jednotlivé šablony lze přehledně uspořádat do stromové struktury.

### **3. ANALÝZA**

---

Šablony se kterými bude možné pracovat:

- Kouzla
- Předměty
- Schopnosti
- Kontext schopností
- Efekty
- Modifikátory
- Postavy

**Export šablon pro mobilní aplikaci MobChar** Program bude umožňovat jednoduchý a hromadný export vybraných šablon pro mobilní aplikaci MobChar ve formátu XML.

**Import šablon z mobilní aplikace MobChar** Program bude umožňovat import XML šablon, které se dají vytvořit v mobilní aplikaci MobChar. Šablony se přidají do databáze a následně je bude možné upravovat a rozšiřovat.

**Ukládání šablon do souboru** Program bude umožňovat ukládat veškeré vytvořené šablony do společného .drd souboru, který bude možné opět do programu načíst a nahradit veškeré šablony. Soubor bude sloužit pro jednoduché ukládání práce a sdílení šablon s ostatními uživateli.

#### **3.1.1.2 Funkcionalita pro Pána jeskyně**

Funkční požadavky, které program bude poskytovat zejména pro Pána jeskyně pro vytváření nových dobrodružství. Tyto funkce rozšiřují funkce pro hráče, které PJ musí také využívat.

**Tvorba a editace dobrodružství** Program bude umožňovat vytváření nových a editaci stávajících dobrodružství. K dobrodružství se dají přiřadit veškeré potřebné objekty, které se v aplikaci dají vytvořit. Celé dobrodružství lze členit do přehledné stromové struktury. V případě potřeby se jednotlivá dobrodružství nebo jejich části dají smazat.

**Tvorba a editace mapy pro dobrodružství** Program bude umožňovat vytvářet a editovat mapy pro dobrodružství. Na mapu bude možné umisťovat jednotlivé objekty, které jsou rozděleny do 4 kategorií:

- Monster - objekt, který reprezentuje příšeru nebo cizí postavu
- Room - objekt, který reprezentuje místo nebo oblast a její popis
- Item - objekt, který reprezentuje předmět nebo truhlu s předměty

- Object - objekt, který reprezentuje ostatní objekty na mapě (tajné dveře, páky atd.)

Jednotlivé objekty navíc obsahují informaci o názvu a jeho popis.

**Tvorba a editace příšer pro dobrodružství** Program bude umožňovat vytváření nových a editaci již existujících příšer pro dobrodružství nebo případné smazání některých nepovedených či nepotřebných příšer. Příšery půjdou seskupovat do stromové struktury pro větší přehlednost.

**Tvorba a editace postav pro dobrodružství** Program bude umožňovat vytváření nových postav a editaci již existujících. Postavy půjdou seskupovat do stromové struktury pro větší přehlednost. Postavám půjdou přiřazovat další objekty, jako jsou předměty, kouzla, schopnosti a další.

**Exportování dobrodružství pro mobilní aplikaci MobChar** Celé dobrodružství půjde exportovat ve formátu XML se strukturou, kterou podporuje mobilní aplikace MobChar pro Pána jeskyně.

**Importování a exportování dobrodružství pro sdílení** Celé dobrodružství včetně všech přiřazených objektů půjde exportovat do jednoho .drd souboru, který následně bude možné do aplikace znova nahrát. Soubor bude sloužit pro ukládání práce a případné sdílení s ostatními Pány jeskyně.

**Export dobrodružství pro tisk** Aplikace bude umožňovat exportování celého dobrodružství do přehledného formátu, který je vhodný pro tisk.

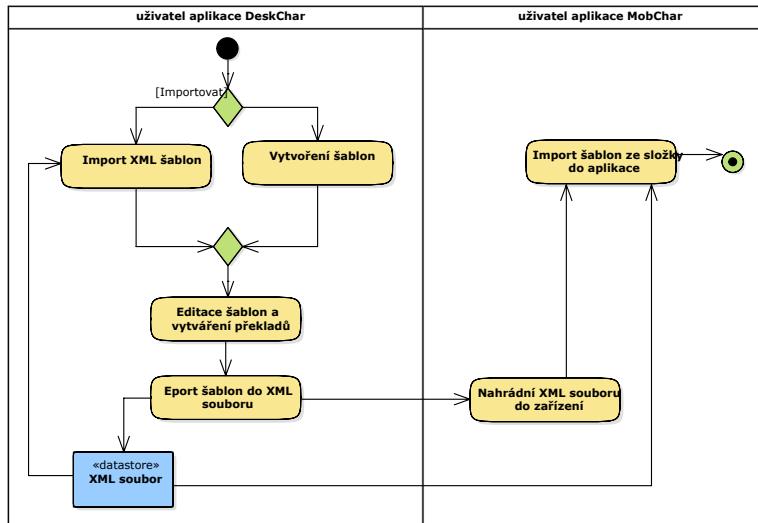
#### **3.1.2 Nefunkční požadavky**

Nefunkční požadavky vymezují převážně nároky na hardware, software nebo například nároky na výkonnost aplikace.

**Podpora operačních systémů Windows a Linux** Program půjde spustit pod operačními systémy Windows a Linux. U operačního systému Windows budou podporovány všechny verze, které jsou novější než Windows XP. Program bude spustitelný pomocí exe souboru. U operačního systému Linux budou podporovány standardní linuxové distribuce. Program bude spustitelný pomocí bash souboru.

**Jazyková podpora** Program bude umožňovat přepínání jazyků a případnou lokalizaci pro další jazyky. V základu bude podporován český jazyk a anglický jazyk.

### 3. ANALÝZA



Obrázek 3.1: Model bussines procesů vytváření šablon

## 3.2 Business procesy

Business proces (někdy též nazývaný podnikový nebo obchodní proces) popisuje tok práce nebo činnosti. Lze ho zaznamenávat pomocí textu nebo přehledných modelů. Modely business procesů se snaží přehledně do diagramu zanést jednotlivé procesy, které bude uživatel s danou aplikací nebo doménou provádět. V práci byl zvolen UML diagram aktivit pro zachycení business procesů.

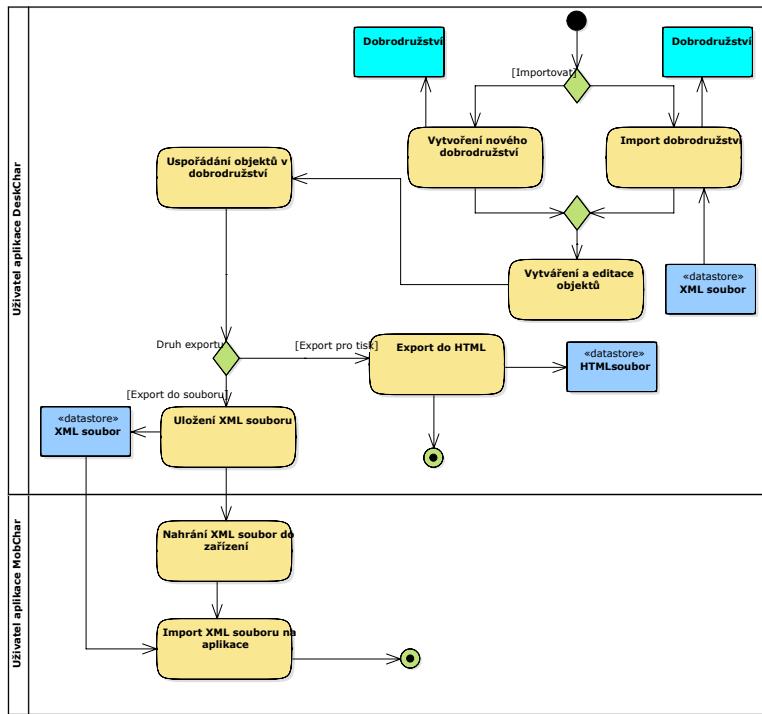
### 3.2.1 Vytváření šablon

Diagram na obrázku 3.1 popisuje základní proces práce se šablonami. Proces popisuje vytvoření šablon a následné nahrání do mobilní aplikace. Šablony vytvoříme v programu nebo případně importujeme z xml souboru vytvořeném buď aplikací nebo například aplikací MobChar. Samozřejmě import a vytvoření nových šablon lze kombinovat. Při tomto procesu nevzniká pouze jedna šablona, ale celý soubor šablon, obvykle jednoho typu. Následně jsou provedeny veškeré potřebné úpravy a vytvořené překlady (pokud jsou nějaké zapotřebí). Daná šablona je následně exportována do strukturovaného souboru XML ve formátu, který podporuje aplikace MobChar. Uživatel mobilní aplikace si vytvořený soubor nahraje do zařízení a následně provede import do aplikace. Nyní je již možné s novou šablonou pracovat.

### 3.2.2 Vytváření dobrodružství

Diagram na obrázku 3.2 popisuje proces, při kterém vzniká nové dobrodružství. Objekt dobrodružství se také řadí mezi šablony, protože má velice podobnou

### 3.2. Business procesy



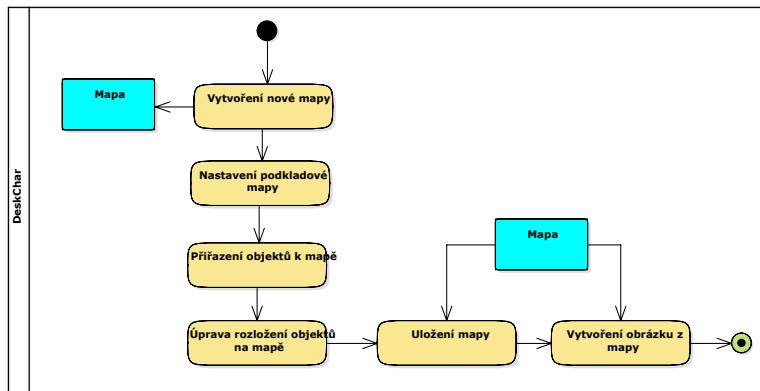
Obrázek 3.2: Model bussines procesů vytváření dobrodružství

funkcionalitu. Protože se jedná o hlavní šablonu, která v sobě obsahuje větší počet šablon různých druhů, je proces vytváření dobrodružství popsán více dopodrobna. Na začátku procesu se vytvoří nové dobrodružství nebo importuje již existující a dále se bude upravovat již stávající hodnoty. V bodě **Vytváření a editace objektů** se jedná o business proces popsáno v kapitole 3.2.1, ve kterém se vytvoří veškeré potřebné objekty, které budou součástí dobrodružství. Veškeré šablony se do dobrodružství přidají a rozřadí podle potřeby pro větší přehlednost (podle částí dobrodružství, podle typu šablony atd.). Když jsou veškeré úpravy hotové, lze výsledné dobrodružství exportovat. Zde si lze vybrat zda dobrodružství exportovat pro tisk do formátu HTML nebo pro mobilní aplikaci MobChar ve strukturovaném formátu XML.

**Formát pro tisk** Ze seznamu se vyberou objekty, které budou exportovány (nemusí být exportováno celé dobrodružství naráz) a následně se provede export. Vytvoří se nám přehledný HTML soubor, který je možné následně vytisknout nebo nahrát do zařízení, které umí pracovat s formátem HTML, na což je dostačující obyčejný webový prohlížeč.

**Formát pro MobChar** Při exportu do formátu XML se vždy exportuje celé dobrodružství. Nedají se vybrat pouze některé části. Aplikace vytvoří XML soubor. Uživatel následně získaný soubor nahraje do zařízení a

### 3. ANALÝZA



Obrázek 3.3: Proces vytváření mapy

importuje ho do aplikace. Takto vytvořené dobrodružství je určené pro mobilní aplikaci MobChar pro Pána jeskyně.

#### 3.2.3 Tvorba mapy

Diagram na obrázku 3.3 popisuje proces vytváření mapy pro Dobrodružství. Na začátku se vytvoří nová mapa a nastaví se její základní údaje, což je jméno a popis. V další části se do mapy vloží podkladový obrázek, který bude sloužit jako základ mapy. Tento obrázek se dá kdykoliv změnit, čímž se původní obrázek smaže a nahradí novým.

Následuje část, ve které se do mapy přidávají jednotlivé objekty. Vybere se požadovaný objekt a vyplní se u něj jméno a základní popis. Objekt se přidá na mapu. Následně je upravena jeho velikost a umístění na mapě pomocí klasické práce s obrázkem. Tuto činnost opakuje, dokud nejsou všechny potřebné objekty na svém místě. Následuje uložení mapy a rozložení předmětů. Po uložení rozložení automaticky následuje vygenerování nového obrázku z mapy, který se používá při exportu pro mobilní aplikaci.

## 3.3 Případy užití

Případy užití popisují jednotlivé scénáře využití aplikace, které uživatel může využívat. Diagram byl rozdělen na tři hlavní části pro větší přehlednost. Na obrázku 3.4 pak můžeme vidět tabulkou pokrytí požadavků případy užití.

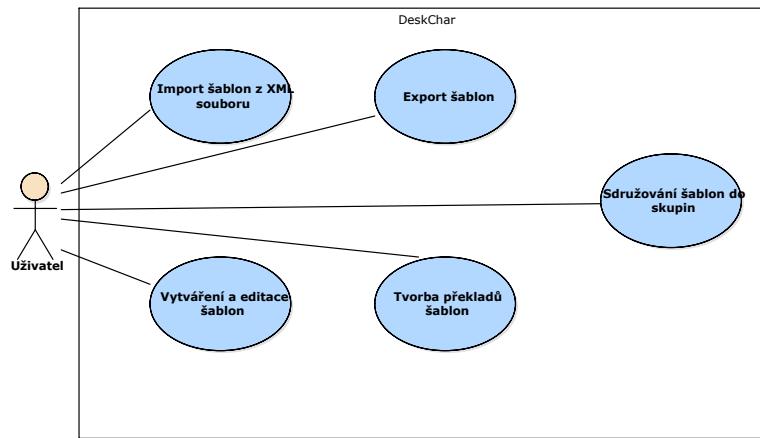
#### 3.3.1 Práce se šablonami

Množina případů užití týkající se práce se šablonami namodelovaná na obrázku 3.7, se týká veškeré práce se šablonami. Dobrodružství a postavy jsou složené z jednotlivých šablon, které se dají využít i samostatně. Samotné Dobrodružství

### 3.3. Případy užití

	Tvorba a editace šablon pro MobChar	Export šablon pro mobilní aplikaci MobChar	Import šablon z mobilní aplikace MobChar	Ukládání šablon do souboru	Tvorba a editace dobroružství	Tvorba a editace mapy pro dobroružství	Tvorba a editace příšer pro dobroružství	Tvorba a editace postav pro dobroružství	Exportování dobroružství pro mobilní aplikaci MobChar - Pj	Importování a exportování dobroružství pro sdílení	Export dobroružství pro tisk
Uložení dobroružství do souboru	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
Nahrání dobroružství ze souboru	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗
Rozdělení na části do stromové struktury	✗	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗
Export dobroružství	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗	✓
Tvorba a editace dobroružství	✗	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗
Editace objektů přímo pro dobroružství	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
Import dobroružství	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗
Vytvoření mapy	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Práce s objekty na mapě	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Přidání podkladové mapy	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Úprava mapy	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Export mapy	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗
Import šablon z XML souboru	✓	✗	✓	✗	✓	✗	✗	✗	✓	✓	✗
Export šablon	✗	✓	✗	✓	✗	✗	✗	✗	✓	✗	✗
Sdružování šablon do skupin	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Vytváření a editace šablon	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗
Tvorba překladů šablon	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Obrázek 3.4: Tabulka pokrytí požadavků případy užití



Obrázek 3.5: Případy užití pro práci se šablonami

### **3. ANALÝZA**

---

a postavy mají stejný formát jako šablony a navíc můžou obsahovat základní šablony (například postavy můžou mít u sebe zbraně).

**Vytváření a editace šablon** Uživatel si bude moci vytvořit nové šablony nebo editovat stávající. Veškeré parametry, které je možné do šablony zapsat, lze jednoduše upravovat v přehledném formuláři. Uživatel si může vytvořit libovolný počet šablon, které se na základě návaznosti sdružují do větších celků (dobrodružství, postava).

**Import šablon z XML souboru** Program bude umožňovat import šablon ze strukturovaných XML souborů, které vytváří mobilní aplikace MobChar a také samotný program DeskChar. Importované šablony se přidají do databáze a lze s nimi následně provádět stejné činnosti jako s nově vytvořenými. Import z formátu XML je možný pouze pro konkrétní objekt, kde ostatní šablony z XML souboru nejsou využity.

**Sdružování šablon do skupin** Veškeré šablony půjde rozřazovat do stromové struktury pro větší přehlednost a jednoduší práci s nimi. Pomocí vytváření složek a jednoduchého drag&drop systému bude rozřazování velice jednoduché a intuitivní. Díky rozřazení do složek je následná práce se šablonami velice usnadněná, ať už se jedná o hromadný export nebo přiřazování rodičovských šablon.

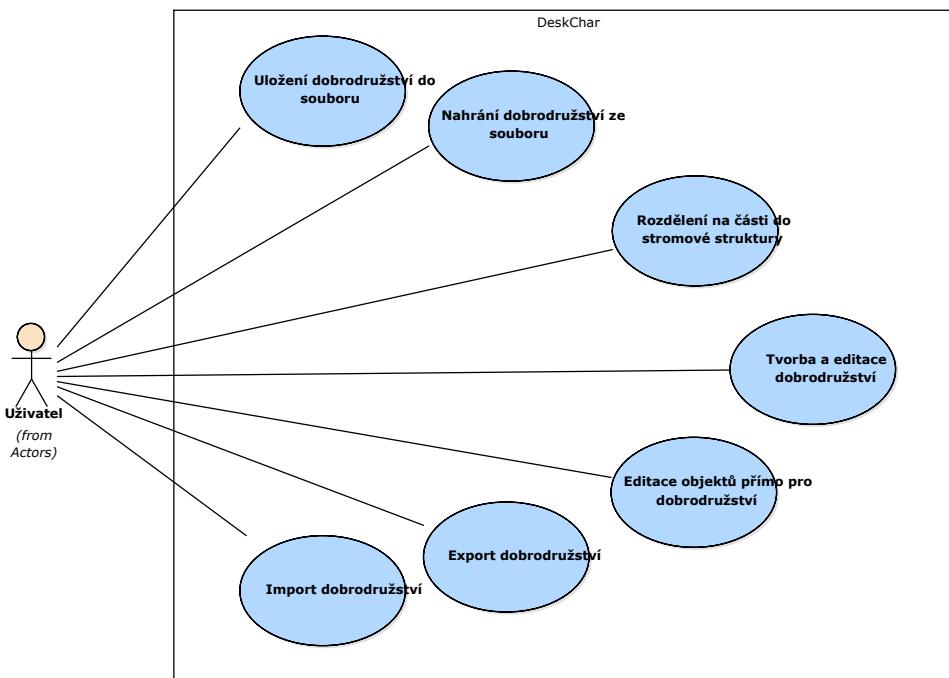
**Export šablon** Veškeré vytvořené šablony lze z aplikace exportovat. K dispozici jsou dva druhy exportu. První možnost je export pro mobilní aplikace MobChar. Aplikace vytváří strukturované XML soubory, které odpovídají formátu, který používá mobilní aplikace MobChar. Další možností exportu je HTML formát, který slouží pro tisknutí šablon do přehledného almanachu, který umožní využít šablony i hráčům, kteří mobilní aplikaci nemají nebo ji nechtějí použít.

**Tvorba překladů šablon** Ke každé šabloně vytvořené nebo importované do aplikace lze vytvořit neomezený počet překladů. Překlady se vytváří v rámci jedné šablony, pomocí přehledného přepínání mezi jazyky. Veškeré hodnoty, pro které překlad nemá smysl (číselné hodnoty, povolání, atd.) budou synchronizovány napříč všemi jazyky.

#### **3.3.2 Tvorba dobrodružství**

**Uložení dobrodružství do souboru** Celé dobrodružství, včetně veškerých šablon, které se v aplikaci nachází, je možné uložit do jednoho souboru s příponou .drd. Tento soubor obsahuje všechny vyplněné údaje šablon a také veškeré obrázky map. V tomto formátu se ukládají i mapy ve formátu, který umožňuje plné obnovení a následnou editaci.

### 3.3. Případy užití



Obrázek 3.6: Případy užití pro práci s dobrodružstvím

**Nahrání dobrodružství ze souboru** V aplikaci je možné otevřít soubor s příponou .drd, která obsahuje celé uložené dobrodružství včetně všech šablon. Tento formát také umožňuje plné obnovení map, včetně veškerých objektů na mapě. Mapy a celé dobrodružství je nadále možné upravovat. Do aplikace se dá nahrát pouze jeden soubor .drd, všechny ostatní data v aplikaci jsou smazána.

**Rozdělení na části do stromové struktury** Všechny části dobrodružství je možné rozdělovat do přehledné stromové struktury. Každý objekt má množinu objektů, které může obsahovat. Mimo to se dají vytvářet složky, které můžou obsahovat všechny objekty. Třídění objektů probíhá pomocí systému drag&drop. Při pokusu přesunutí objektu na nesprávné místo se struktura stromu nezmění.

**Tvorba a editace dobrodružství** V aplikaci je možné vytvořit neomezené množství dobrodružství. Dobrodružství se skládají z jednotlivých lokací, které dobrodružství rozdělují na logické celky, a předmětů, kouzel a schopností, které jsou společné pro celé dobrodružství. Do dobrodružství lze přidávat předpřipravené šablony ostatních objektů.

**Editace objektů přímo pro dobrodružství** Po přidání objektu ze šablony

### 3. ANALÝZA

---

do dobrodružství, se vytvoří nový objekt pomocí hluboké kopie, což znamená úplné zkopirování veškerých dat a vytvoření nového objektu. Tento objekt je pak možné přímo ze záložky dobrodružství editovat a ukládat. Podle druhu objektu se otevře příslušný formulář pro editaci dané šablony.

**Export dobrodružství** Celé dobrodružství je možné vyexportovat do strukturovaného formátu XML, který podporuje mobilní aplikace MobChar. K vyexportovanému dobrodružství se připojí složka resources, ve které se nachází veškeré mapy použité v dobrodružství, které mobilní aplikace umí zobrazit.

**Import dobrodružství** Dobrodružství je také možné ze strukturovaného XML souboru importovat. Z jednoho souboru lze nahrát pouze jedno dobrodružství. Při importu dobrodružství do aplikace se do dobrodružství nepřidají mapy, protože XML soubor v sobě neobsahuje údaje, které by umožňovali mapu editovat. Do aplikace je možné importovat libovolné množství dobrodružství.

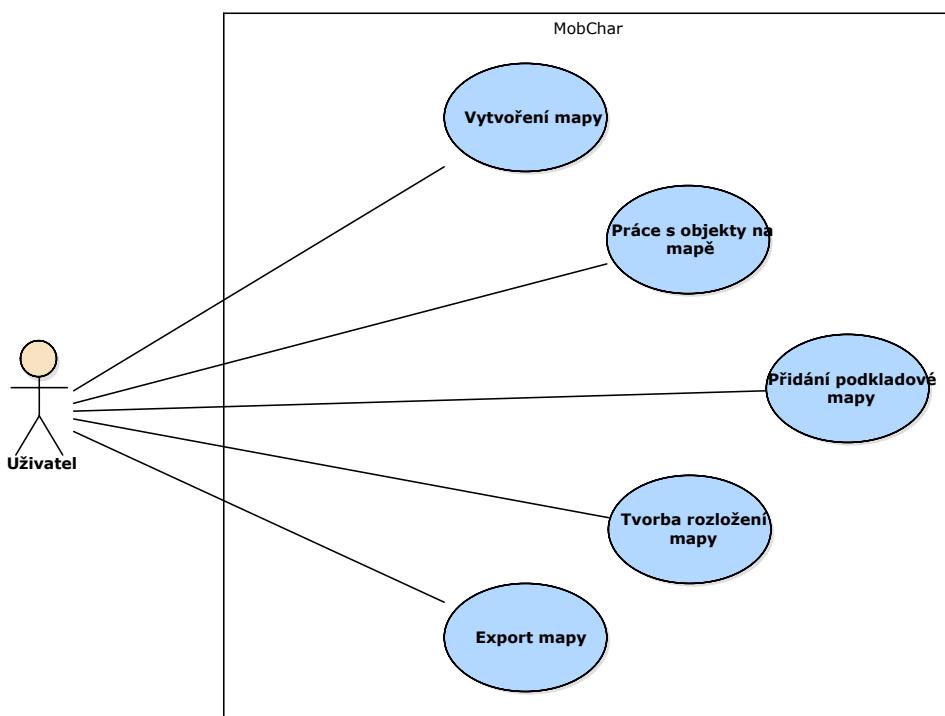
#### 3.3.3 Tvorba mapy

**Vytvoření mapy** V aplikaci je možné vytvořit libovolné množství map, které se následně dají přiřadit do jednotlivých lokací. Každá mapa má svůj název a základní popis. Mapy je možné jako veškeré objekty rozřazovat do stromové struktury.

**Práce s objekty na mapě** Na mapu je možné přidávat jednotlivé objekty. Objekty jsou rozděleny na čtyři základní druhy. Objekt `monster`, který reprezentuje na mapě nestvůry a všechny postavy. Objekt `item` reprezentuje na mapě všechny předměty, ať už se jedná o truhlu s pokladem nebo tajné klíče, či nádoby. Další objekt je `room`, který slouží k popisu místností a lokací. Poslední objekt je `general`, který zastupuje všechny ostatní objekty na mapě, které je potřeba na mapě zaznamenat. Všem objektům je možné měnit velikost a umístit je na konkrétní místo na mapě.

**Přidání podkladové mapy** Základním prvkem mapy je podkladový obrázek, který tvoří základní rozložení mapy. Program podporuje základní druhy obrázků jpg, gif a png. Na mapě se může nacházet pouze jeden podkladový obrázek, pokud na mapě již nějaký se nachází a je přidán další, původní obrázek je smazán. Obrázek je zvětšen na velikost okna.

**Úprava mapy** Každou mapu je možné upravovat. Mapa se ukládá v editovatelném formátu. Všechny objekty se dají z mapy smazat a veškeré hodnoty objektů se dají upravit. Podkladová mapa se dá také kdykoliv upravit.



Obrázek 3.7: Případy užití pro práci s mapou

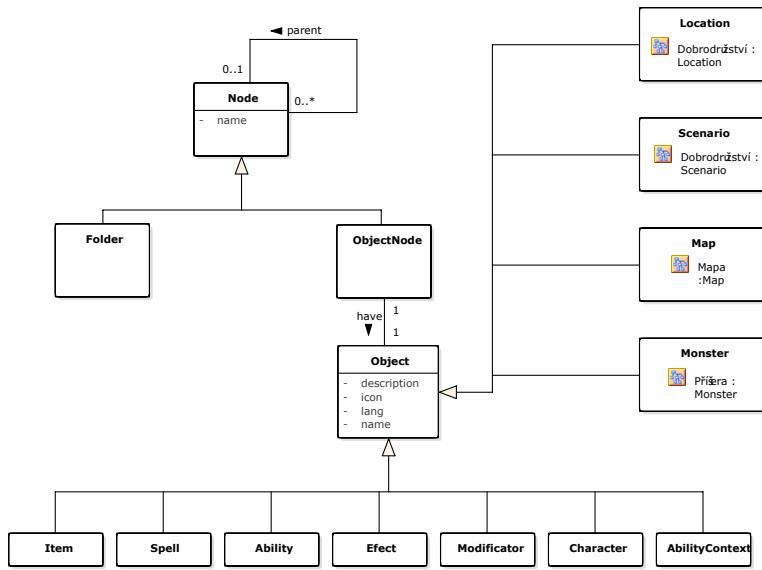
**Export mapy** Mapu je možné exportoval do formátu XML nebo do formátu HTML. Formát XML je převážně určen pro mobilní aplikaci MobChar. Soubor XML je vytvořen včetně přiložené složky **resources**, která obsahuje veškeré vytvořené mapy ve formátu png. V XML souboru se nachází seznam objektů, které jsou umístěny na mapě včetně jejich čísla, jména a popisu. Tyto údaje využívá mobilní aplikace. Při exportu do formátu HTML, se vytvoří stránka, jejíž hlavní část tvoří obrázek s mapou, pod kterou se nachází seznam všech objektů.

## 3.4 Doménový model

Doménový model má za úkol popsat strukturu tříd v aplikaci a jejich vazby mezi nimi. Pro zaznamenání se využívá UML diagram. Diagram nepopisuje jednotlivé funkce tříd ani atributy, které slouží k implementaci tříd, ale zachycuje pouze základní strukturu.

### 3. ANALÝZA

---

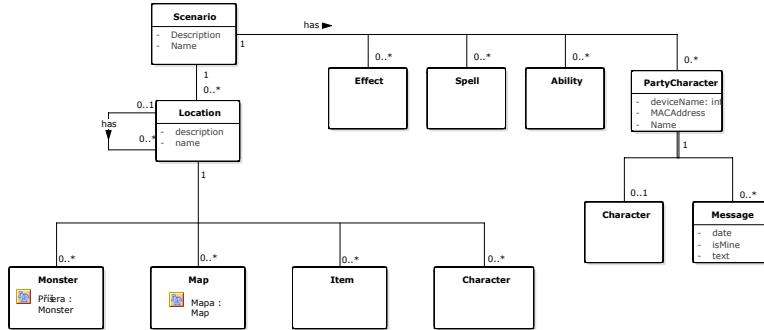


Obrázek 3.8: Analytický doménový model stromové struktury

#### 3.4.1 Stromová struktura

Diagram na obrázku 3.8 popisuje strukturu objektů, které slouží pro uchování všech objektů (šablon) a jejich rozřazení do stromové struktury. Veškeré šablony využívají tuto stromovou strukturu. Třída **Folder** (složka) slouží pouze k rozřazování jednotlivých objektů do skupin. Složka má pouze jméno a může obsahovat libovolný počet složek nebo objektů. **ObjektNode** pak slouží k uchování šablony samotné. Dolní část objektů na obrázku (**Item**, **Spell**, atd.) jsou objekty, které jsou totožné s aplikací MobChar a slouží k uchování šablon pro základní aplikaci a využívají se i v aplikaci pro Pána jeskyně. Pokud vás zajímá podrobnější popis těchto objektů, nahlédněte do bakalářské práce týkající se MobCharu [2]. Oproti tomu, objekty na pravé straně diagramu (**Map** item, **Scenario**, atd.) jsou objekty, které slouží převážně k vytvoření dobrodružství. Jejich podrobnější popis se nachází dále v textu.

Veškeré objekty můžou mít potomky. Systém složek slouží pouze pro přehledné uspořádání. Každý objekt má definované objekty, které může mít jako potomky. Tento systém slouží pro ukládání závislostí, například dobrodružství může obsahovat mnoho lokací, příšer, map a dalších. Některé objekty, jako například kouzla a schopnosti, žádné potomky mít nesmějí. Veškeré vazby mezi objekty jsou zachyceny pouze ve stromové struktuře, aby bylo zabráněno duplikování informací a nebylo nutné tyto údaje synchronizovat.



Obrázek 3.9: Analytický doménový model dobrodružství

### 3.4.2 Dobrodružství

Diagram na obrázku 3.9 popisuje strukturu pro ukládání a práci s dobrodružstvím. Hlavní třída **Scenario** je rozdělená do jednotlivých lokací. Pro celé dobrodružství jsou však společné předměty, kouzla, schopnosti a postavy. Předměty, kouzla a schopnosti jsou pro Pána jeskyně, který je nadále poskytuje hráčům, například pokud se hráč má možnost naučit nové kouzlo, nebo získal důležitý předmět pro dobrodružství (elixír, mapa, atd.). Třída **Character** zde zastupuje postavy hráčů, kteří dané dobrodružství hrají. Jedná se o stejné třídy jako v původní aplikaci MobChar. Pro přesnější popis těchto tříd, je možné nahlédnout do bakalářské práce týkající se aplikace MobChar[2].

Celé dobrodružství je členěné do lokací, které se navíc můžou do sebe zanořovat (lokace může mít pod sebou další lokace). Lokace obsahují příšery, mapy, předměty a postavy. Příšery mají velice podobnou strukturu jako postavy, můžou se naučit schopnosti a kouzla a vlastnit předměty, mají však odlišné atributy, proto jsou od postav odděleny. Jednotlivé mapy mají složitější strukturu, která je popsána níže. Předměty zde znamenají předměty důležité pro tuto lokaci (klíče, věci v bednách a další). Poslední částí jsou postavy, které zde znázorňují postavy, za které nehrají hráči, ale samotný PJ. Jsou to důležité postavy pro dobrodružství, které se nacházejí v dané lokaci. Objekt je stejný jako klasické postavy, ale význam je trochu odlišný.

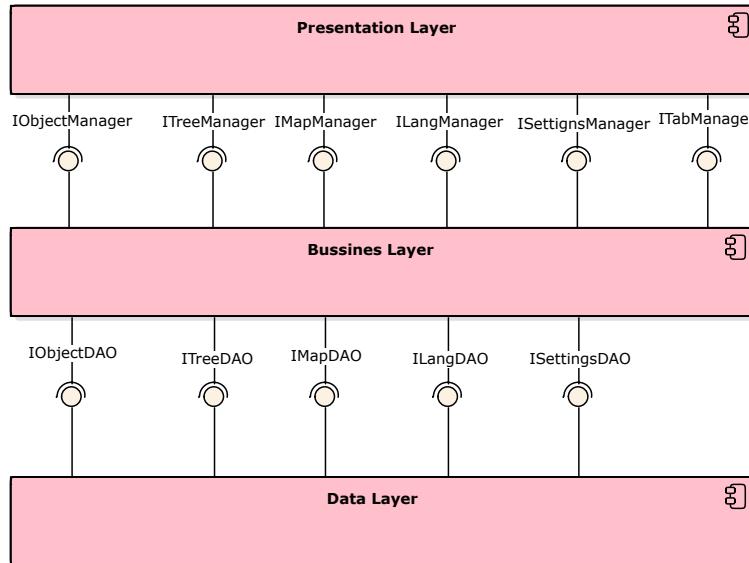
## 3.5 Návrh řešení architektury

Model architektury popisuje formální popis systému, případně jeho detailní plán na úrovni komponent vedoucí k jeho implementaci. Hlavním účelem modelu architektury je popsání hlavních vrstev programu a popisu způsobu komunikace mezi nimi.

Byla zvolena relaxovaná třívrstvá architektura. Relaxovaná třívrstvá architektura zajistuje dobrou přehlednost projektu a také snadnou nahraditelnost

### 3. ANALÝZA

---



Obrázek 3.10: Model architektury

celé vrstvy programu, například při přechodu na jinou databázi. Veškeré vrstvy mezi sebou komunikují na základě rozhraní definovaného interfaci. Interface je abstraktní třída, ze které musí implementační třída dědít, aby byla zajištěna implementace všech funkcí

#### 3.5.1 Vrstvy

**Prezentační vrstva** Prezentační vrstva slouží k zobrazení informací pro uživatele. Jedná se o grafickou část celé aplikace. Jejím hlavním úkolem je získávat od uživatele data a naopak uživateli data přehledně zobrazovat.

**Business vrstva** Business vrstva tvoří hlavní spojení mezi datovou a prezentační vrstvou. Business vrstva by se měla starat o veškeré logické operace s objekty, které by správně neměli provádět ani datová ani prezentační vrstva.

**Datová vrstva** Hlavním úkolem datové vrstvy je zajistit persistenci dat pro aplikace a také operace s externími soubory. Datová vrstva zajišťuje ukládání dat do databáze a také získávání potřebných dat. Mimo to datová vrstva zajišťuje práci s se soubory XML, HTML a drd.

#### 3.5.2 Rozhraní

Mezi vrstvami jsou definovaná rozhraní, která je nutné dodržet. Rozhraní začínají velkým písmenem I, konkrétní implementace daného rozhraní obvykle

má stejný název, pouze bez počátečního písmena I.

Pro datovou vrstvu se používají takzvané „data access objects“ (DAO). Jedná se o třídy, které zajišťují přístup k datům například z databáze a naopak i data ukládají. Pro business vrstvu se využívají takzvané „managery“.

### DAO třídy

**IObjectDAO** je skupina několika DAO tříd, které se starají o přístup k datům z databáze a z XML. Na diagramu 3.10 je zobrazen pouze zástupný IObjectDAO, který v implementaci neexistuje a je nahrazen jednotlivými DAO rozhraními (ISpellDAO, IEfectDAO atd.).

**ITreeDAO** se stará o veškeré ukládání stromové struktury do databáze.

**IMapDAO** se stará o ukládání veškerých dat, která se týkají map.

**ILangDAO** se stará o ukládání jazyků, které jsou v aplikaci použity. Nejedná se o překlady prezentační vrstvy, ale o jazyky vytvořené pro překlady šablon. Rozhraní se nestará o ukládání samotných překladů.

**ISettingsDAO** se stará o ukládání veškerého nastavení aplikace.

### Manager třídy

**IObjectManager** navazuje na IObjectDAO. Jedná se opět o skupinu tříd pro veškeré základní objekty (ISpellManager, IAbilityManager, atd.). Rozhraní definuje, jak přijímá data z prezentační vrstvy. Základní funkcionalita spočívá ve zpracování dat z prezentační vrstvy a vytvoření kompletního objektu, se kterým se dále pracuje nebo se pošle na datovou vrstvu pro uložení.

**ITreeManager** se stará u připravení stromové struktury pro zobrazení. Vytváří z objektů stromovou strukturu a přidává do struktury metadata, které slouží pro zobrazení a práci na prezentační vrstvě.

**IMapManager** se stará o zpracování dat ohledně mapy. Převádí zobrazenou mapu pro uživatele na formu, ve které se dá uložit do databáze.

**ILangManager** se stará o zpracování dat z prezentační vrstvy ohledně jazyků.

**ISettingsManager** se stará o zpracování dat získané z widgetu pro nastavení a předání správných dat datové vrstvě.

**ITabManager** se stará o vytváření překladových záložek pro prezentační vrstvu. Toto rozhraní nemá DAO rozhraní, protože veškeré informace které

### **3. ANALÝZA**

---

potřebuje, získá přímo z konkrétních IObjectDAO tříd. Jejím hlavním úkolem je zpracování všech překladů jedné šablony a připravení objektů pro uložení.

# KAPITOLA 4

## Návrh

Čtvrtá kapitola se věnuje návrhu vytvořené aplikace. Je zde popsán základní model databáze a také popsána struktura hlavních částí formátu XML. Následně je zde pomocí sekvenčního diagramu popsán základní princip zobrazení šablon. V poslední část této kapitoly je popsán diagram nasazení.

### 4.1 Model databáze

Pro realizaci aplikace DeskChar byla vybrána databázi **SQLite**, která je jednoduchá a nepotřebuje žádný složitý externí software pro běh. Na druhou stranu zvládá veškeré potřebné operace, jako jsou cizí klíče a kaskádové mazání záznamů. Pro vytváření a práci se stromovou strukturou je kaskádové mazání nedocenitelný pomocník.

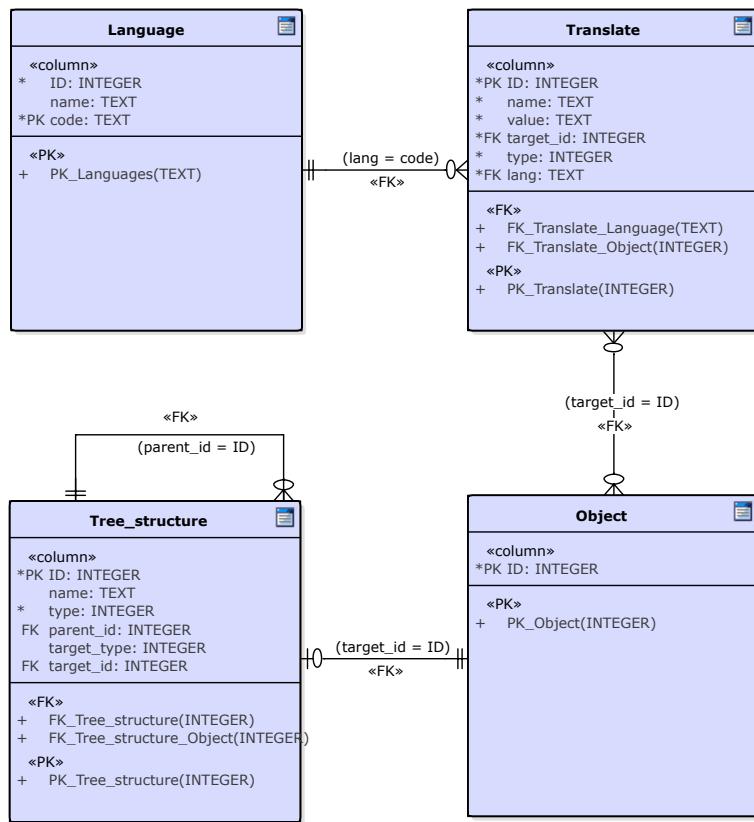
Diagram na obrázku 4.1 je zjednodušený pro větší přehlednost. Na diagramu je zachycena základní struktura a princip závislostí v databázi. Kompletní model databáze se nachází na přiloženém CD.

Z důvodu vícejazyčnosti šablon, bylo zapotřebí navrhnout strukturu databáze, která dokáže uložit neomezený počet jazykových překladů. Na obrázku 4.1 můžeme vidět základní strukturu databáze. Údaj o jazyku, ve kterém je daný překlad vytvořen, je uložen v tabulce **Languages**, který je navázán na tabulku **Translate** cizím klíčem **code**. Primárním klíče každého jazyka je textový kód, který je unikátní. V tabulce **Translate** pak nalezneme veškeré textové řetězce, které se nacházejí v objektech. Jeden záznam tabulky **Translate** je přiřazen ke konkrétnímu objektu pomocí dvojice hodnot **target\_type**, který určuje o jaký objekt (tabulku v databázi) se jedná, a **target\_id**, který určuje konkrétní záznam v tabulce (ukazuje na primární atribut ID v tabulce). Informace o překladu jsou uloženy ve dvojici hodnot **name** a **value**, které slouží jako slovník, jejich jazyk určuje hodnota **lang**. Takto zvolený návrh databáze umožňuje neomezený počet jazyků a jednoduchou rozšiřitelnost.

Druhá část diagramu se týká stromové struktury, která se používá pro všechny objekty v aplikaci. Tabulka **Tree\_structure** slouží pro zaznamenávání

#### 4. NÁVRH

---



Obrázek 4.1: Základní model databáze

stromové struktury pro veškeré objekty. Do tabulky se ukládají dva druhy uzlů, složky a objekty. Hodnota **type** určuje, o který druh uzlu se jedná, zda o složku nebo o objekt. Pomocí cizího klíče **parent\_id**, vzniká stromová struktura. Zde se využívá kaskádové mazání - pokud se smaže záznam, který má pod sebou navázané další záznamy pomocí cizího klíče, smažou se tyto záznamy také automaticky v databázi. Tento princip udržuje tabulkou konzistentní a nevznikají žádné záznamy, které se již nepoužívají. Každý uzel, který je typu object, ukazuje pomocí dvojice hodnot **target\_type**, která určuje tabulku v databázi, a **target\_id** na záznam v ostatních tabulkách. V diagramu je tento princip naznačen zjednodušeně, kde tabulka **Object** zastupuje tabulky všech objektů (**Spell**, **Item**, atd.).

Poslední problém návrhu databáze se skrýval v návrhu ukládání vazeb mezi jednotlivými objekty. V úvahu připadali dvě možnosti. První možnost byla zachytit veškeré vazby mezi objekty pomocí tabulek **relations**, které by obsahovaly pouze dvojici klíčů z obou tabulek. Toto řešení by bylo rychlé a jednoduché na používání, bohužel by se u některých vazeb nedalo použít samotně. Například u dobrodružství se podřazené objekty můžou sdružovat

dále do složek, čehož by se pouze pomocí těchto vazeb nedalo docílit. Musela by se zde navíc využít stromová struktura použitá pro všechny šablony a jejich základní rozřazení, čímž by vznikaly duplicitní data. Duplicít data sebou nesou dva základní problémy. První z nich je samozřejmě větší množství dat, které je potřebné uložit. V tomto případě by se však nejednalo o drastický nárůst, který by dělal problém. Druhý a závažnější problém se týká aktuálnosti dat. Bylo by zapotřebí udržovat veškeré údaje aktuální a stejně, což by mělo velké nároky na složitost ukládání a zpomalovalo by to některé operace, které je zapotřebí, aby aplikace prováděla okamžitě (například drag&drop rozřazování). Z těchto důvodů jsem se rozhodl zvolit druhou možnost, kde veškeré vazby mezi objekty jsou uloženy pouze ve stromové struktuře popsané výše. Problém bude vznikat při operacích exportu, kde bude nutné vazbu mezi objekty dohledávat ze stromové struktury. Tyto operace se však neprovádí tak často a není zde velký důraz na okamžitou odezvu.

## 4.2 Model XML souborů

S doménovým modelem úzce souvisí návrh struktury XML souborů. Struktura těchto souborů je velmi důležitá. Na základě této definované struktury bude probíhat komunikace s mobilní aplikací MobChar. Formát základních šablon předmětů, kouzel, schopností, efektů modifikátorů a postav, byl převzat z původní aplikace MobChar pro hráče, aby byla zachována konzistence mezi aplikacemi. Dále se zaměřuji pouze na nově navržené části, což jsou příšery, lokace, mapy a dobrodružství. Pokud vás zajímá struktura základních šablon, nahlédněte do bakalářky týkající se aplikace MobChar[2] nebo do přiložené kompletní dokumentace.

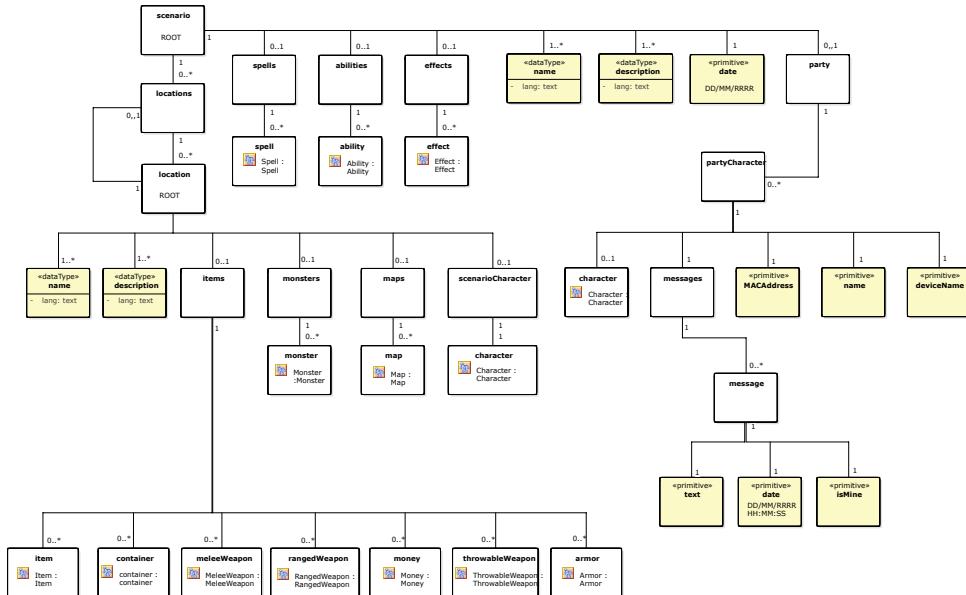
Strukturu XML byla namodelována diagramem, který se normálně využívá pro modelování tříd. Pro tento případ vypovídající hodnota diagramu je dostačující a jasně zadefinovává strukturu XML. Entity v diagramu které jsou znázorněny žlutou barvou znázorňují již konkrétní XML tag, který uvnitř obsahuje pouze hodnotu a žádné další vnořené tagy. Oproti tomu bílé entity znázorňují pouze obalující tag, který uvnitř obsahuje strukturu další tagů. Každá entita v diagramu znázorňuje jeden tag ve výsledném XML dokumentu. Parcialita a kardinalita v diagramu znázorňuje povinnost a případně množství tagů, které se na daném místě v XML souboru můžou objevit. Parcialita vztahu určuje, zda se jsou vnořené tagy povinné či nikoliv. V některých případech nemusí být všechny tagy přítomny. Kardinalita určuje, zda na stejně úrovni se těchto tagů může vyskytovat více. Kořenový tag je vždy označen textem „ROOT“ .

### 4.2.1 Dobrodružství

Struktura XML souboru pro dobrodružství je velice podobná doménovému modelu dobrodružství 3.9, jak můžeme vidět na obrázku 4.2. Model XML však

## 4. NÁVRH

---



Obrázek 4.2: Model XML souhoru dobrodružství

má dopodrobna namodelovány veškeré atributy objektů.

Obecný princip, který se používá ve všech navržených XML souborech je obalení tagů, které se na dané úrovni můžou vyskytovat vícekrát, rodičovským tagem, jehož jediným cílem je zpřehlednit XML soubor. Tento tag je zpravidla pojmenován jako množné číslo objektů, které obaluje (například pro skupinu tagů **Spell** se obalující tag jmeneje **Spells**).

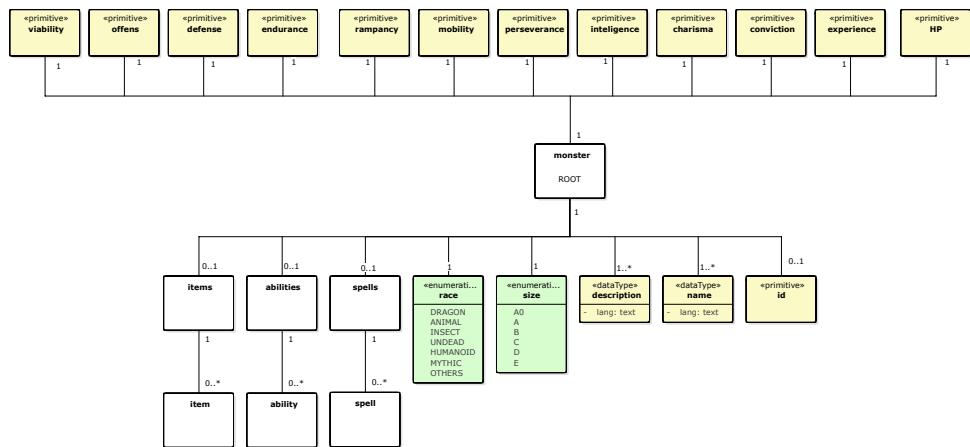
Pro potřeby správného rozzeznání předmětů při importu jsou předměty rozděleny do 7 kategorií, do stejných kategorií, které byly navrženy pro původní aplikaci MobChar. Zde se struktura od původního návrhu šablon lehce změnila. Každá kategorie předmětů již nemá vlastní obalující tag, ale všechny předměty jsou sloučené pod jedním tagem **items**.

Posledním výrazným rozdílem je jiný název pro postavy, které se nacházejí v lokacích. Jelikož se jedná o stejný objekt, se stejnou XML strukturou, ale v rámci celého dobrodružství mají tyto postavy lehce jiný význam, bylo zapotřebí tyto postavy oddělit, aby nebylo nutné se o druhu rozhodovat na základě rodičovského tagu. Proto obalující tag pro postavy, které se nachází v lokacích byl pojmenován **scenarioCharacter**.

### 4.2.2 Příšery

Druhá nově zadefinovaná struktura XML souborů se týká příšer. Příšery jsou samozřejmě součástí dobrodružství, jak jste mohli vidět na obrázku 4.2. Na obrázku 4.3 můžeme vidět detailní strukturu souboru. Struktura souboru je

### 4.3. Model komunikace



Obrázek 4.3: Model XML souhoru příšery

podobná struktura postav, také obsahuje seznam předmětů, schopností a kouzel. Obsahuje tag **race**, který určuje rasu příšery, ale jedná se o odlišné rasy než pro postavy. Pro určení rasy příšer jsme hledali kompromis, mezi dostatečnou vypovídající hodnotou a konzistencí oproti velkému množství ras, které by se velice rychle stalo nepřehledné. Uživatel si samozřejmě může příšery rozřadit podle libosti na základě stromové struktury nebo případně v popisku příšery. Proto jsme se rozhodli rasy příšer omezit pouze na základních sedm, které můžete vidět na obrázku. Jedním z hlavních důvodů, proč pro příšery nemohl být použit stejný model jako pro postavy je rozdílná množina základních atributů. Hlavní důvod rozdělení příšer od klasických postav byl v rozdílných attributech. Pro počítání výsledku soubojů se používají jiné atributy než u postav.

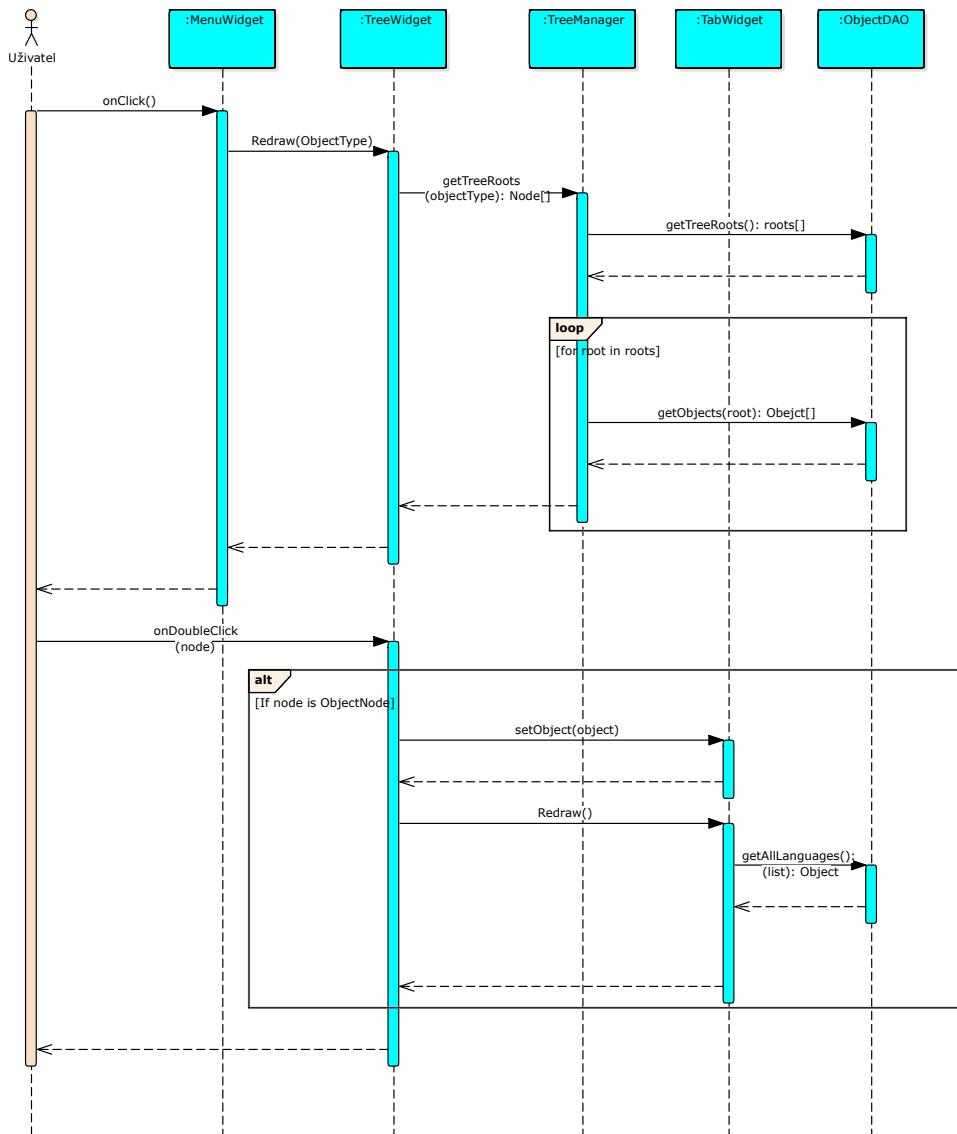
## 4.3 Model komunikace

Mezi nejčastější operaci prezentační vrstvy patří práce se stromovou strukturou. Na diagramu 4.4 můžeme vidět základní dvě činnosti. Vybrání konkrétního typu šablon, které zobrazí stromovou strukturu a zobrazení hlavního widgetu pro vytváření šablon po vybrání konkrétní položky ve stromové struktuře.

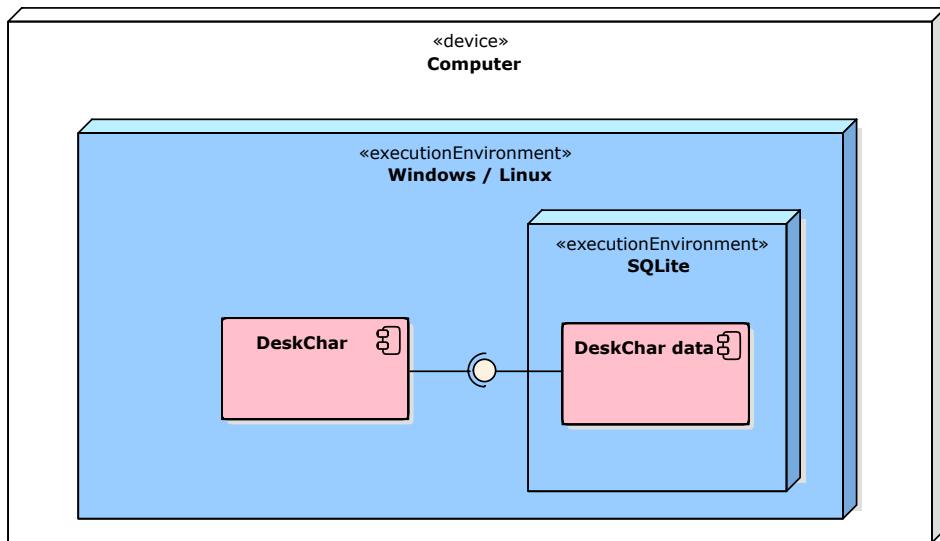
Uživatel v menu klikne na požadovaný druh šablon. Tím se zavolá funkce **Redraw**, která má na starosti vykreslení stromové struktury ve widgetu. Potřebná data získá z **TreeManageru** po zavolení funkce **getTreeRoots**, která má na starosti, vytvoření kompletního stromu uzlů, přičemž vrací seznam kořenových uzlů. **TreeWidget** z tohoto listu pomocí rekurence vykreslí celý strom. **TreeManager** získává data z konkrétní DAO třídy (v diagramu zjednodušena jako **ObjectDAO**). Objekty navěšené na uzly stromu můžou být různé, takže

#### 4. NÁVRH

---



Obrázek 4.4: Model komunikace pro hlavní obrazovku



Obrázek 4.5: Model nasazení

třída `TreeManager` volá více různých DAO tříd.

Druhá část diagramu se týká vykreslení hlavní části obrazovky, kde se vytváří samotné šablony a jejich překlady. Uživatel ve stromové struktuře dvakrát klikne na nějaký uzel. Pokud se jedná o složku, funkce zavolá pouze rodičovskou funkci `TreeWidgetu`, což má za následek rozbalení obsahu složky. Pokud se jedná o objekt, zavolá se nad třídou `TabWidget` funkce `setObject` s parametrem `object`. Hned poté se zavolá funkce, která vykreslí všechna data. Data získá na základě nastaveného objektu, který si pamatuje vlastní DAO třídu. Třída vrátí seznam objektů, přičemž se jedná o jednu šablonu, ale ve všech jazycích, ve kterých byla vytvořena. `TabWidget` následně pro každý jazyk vykreslí jednu záložku, kde se dají hodnoty šablony upravovat nebo případně vytvořit nový překlad pro nový jazyk.

## 4.4 Model nasazení

Diagram nasazení zobrazuje způsob rozdělení systému na samostatné části a komunikační vazby mezi nimi, čímž definuje architekturu systému. Nalezneme zde veškeré komponenty systému a všechny potřebné části pro běh aplikace.

Na obrázku 4.5 můžeme vidět, že software je určený pro počítače a funkční pod operačními systémy Windows a Linux (verze operačního systému jsou uvedeny v sekci nefunkční požadavky 3.1.2). Aplikace je napsaná v jazyce Python a zabalená do spustitelného balíčku, který nevyžaduje žádné speciální požadavky na systém. Veškeré potřebné knihovny a prostředí má uložené

#### 4. NÁVRH

v balíčku. Databáze SQLite běží v rámci tohoto balíčku také a nepotřebuje žádné dodatečné prostředí.

# Implementace

Pátá kapitola se věnuje samotné implementaci programu. Je zde popsáný použitý programovací jazyk a využité knihovny. Dále jsou zde popsány využité nástroje pro tvorbu celé bakalářské práce. V poslední části jsou popsány problémy, které vznikly při programování aplikace. Některé problémy byly zjištěny na základě testování, které je popsáno v další kapitole.

## 5.1 Použitý programovací jazyk

Jedním z nejdůležitějších rozhodnutí implementace je zvolení programovacího jazyku všech částí. V mnoha případech je programovací jazyk zvolen ještě před začátkem analýzy, což však není ideální. Analýza aplikace by měla být nezávislá na programovacím jazyku. Jelikož se jedná o desktopovou aplikaci, byla zvolena databáze typu sqlite, která je ideální pro práci na jedné stanici.

Jako hlavní programovací jazyk pro implementaci byl zvolen Python. Python je vysokoúrovňový skriptovací netypový programovací jazyk. I když se jedná o převážně scriptovací jazyk, díky velkému množství knihoven je velice snadné v něm psát i rozsáhlé grafické programy. Velké síla jazyka Python se skrývá v tom, že se jedná o netypový programovací jazyk, což v mnoha případech ulehčuje implementaci.

Programovací jazyk Python je momentálně vyvíjen ve dvou oddělených a nezávislých větví a to Python verze 2 a Python verze 3. Pro tuto práci byla zvolena verze 3, která je modernější, většina nových knihoven vzniká právě pro verzi 3 a navíc umožňuje v základu více funkcionalit.

## 5.2 Využité knihovny

Samotný programovací jazyk je omezen pouze na základní funkce programovacího jazyku. Aby nebylo nutné již existující funkcionality psát znova, využívají se dodatečné knihovny. Programovací jazyk Python je známý velkým množ-

stvím knihoven, které rozšiřují základní funkcionalitu například o programování grafického rozhraní, matematické výpočty a mnoho dalšího.

### 5.2.1 Knihovna pro práci s XML

Velká část programu se týká práce s XML šablonami. Bylo zapotřebí vybrat vhodnou knihovnu, která by veškerou práci co nejvíce usnadnila. Nejsnažší způsob vytváření rozsáhlých XML souborů přímo z objektů je pomocí mapování třídních atributů přímo na atributy v XML objektu. Bohužel taková knihovna, která by tuto funkcionalitu v základu podporovala pro Python neexistuje. Proto byla zvolena knihovna lxml[8], která umí základní parsování XML souborů a vytvoření stromového objektu ze získaných dat. Nad touto knihovnou bylo naprogramováno rozšíření pro mapování na objekty, které je blíže popsáné dále v textu v sekci implementace.

### 5.2.2 Grafická knihovna

Pro Python existuje velké množství grafických knihoven. Některé jsou zaměřené na mobilní aplikace, některé hlavně pro webové rozhraní a spoustu jich se také zaměřuje na desktop. Vybrat mezi takovým množstvím knihoven není lehké. Mezi nejznámější patří knihovny Kivy, PyGame, TkInter a PyQt. Knihoven pro tvorbu grafického rozhraní existují desítky. Pro rozebrání všech však není v této práci místo a proto jsou zmíněné pouze nejpoužívanější z nich.

Knihovna kivy je převážně zaměřené na dotykové displeje. Výsledný vzhled a veškeré widgety jsou pro to přizpůsobené a ovládání pomocí myši a klávesnice není tak intuitivní jako u ostatních knihoven.

Dalším důležitým kritériem pro výběr knihovny byla aktuálnost a vydávání nových aktualizací a samozřejmě podpora Python verze 3.0.\*. Pro knihovnu TkInter již dlouho nevychází nové aktualizace. Knihovna je velice jednoduchá a intuitivní, bohužel již několik let není aktuální, proto není příliš vhodná.

Knihovna PyGame, jak již napovídá její název, se specializuje převážně na tvorbu počítačových her. Má rozsáhlé možnosti animací a herních prvků, které jsou důležité převážně ve hrách. Knihovna neumí vytvářet přehledné grafické rozhraní aplikace, proto byla ze seznamu také vyřazena.

Poslední ze čtveřice knihoven je PyQt. Knihovna se zaměřuje převážně na tvorbu přehledných grafických rozhraní pro desktopové programy na různé operační systémy, což je přesně to co je zapotřebí. Navíc má velice moderní vzhled, což bylo jedno z hlavních kritérií při výběru. Aktualizace k této knihovně jsou vydávány pravidelně a nejnovější a stále aktualizovaná verze PyQt5 je určena pro Python 3. Proto byla tato knihovna zvolena jako nejlepší volba pro tvorbu uživatelského rozhraní. Přesněji byla využita verze knihovny 5.8.

### 5.2.3 Knihovna pro tvorbu HTML almanachů

```
{% extends "layout.html" %}

{% block body %}

<h1> {{ TR['Spell'] }} </h1>
{% for object in objects %}
    {% if object.object_type == ObjectType.SPELL %}
        {% include 'spell.html' %}
    {% endif %}
{% endfor %}

{% endblock %}
```

Algoritmus 1: Ukázka syntaxe šablonovacího systému Jinja

Jednou z dalších důležitých funkčností programu je generování HTML almanachů, které budou uzpůsobené pro tisk. Bylo zapotřebí zvolit vhodný šablonovací systém, kterým z předem připravených HTML šablon vytvoří přehledné almanachy, které bude možné prohlížet v libovolném webovém prohlížeči, kde lze využít krížových odkazů nebo je vytisknout a vzít kamkoliv sebou. Bylo zapotřebí zvolit vhodný šablonovací systém, který bude umožňovat nejen doplňování proměnných do textu, ale také podmínky a cykly.

Mezi nejpoužívanější šablonovací systémy pro Python patří například Mako [9], Django templates a Jinja [10]. Python podporuje i velice jednoduchý a základní šablonovací systém nativně. Tento systém je však opravdu velice jednoduchý, nepodporuje žádné podmínky ani cykly a hodí se spíše pro vytváření přehledných výstupů do konzole. Velice rozšířený šablonovací systém je Mako. Jedná se o výchozí šablonovací systém pro webové frameworky pro Python. Tento nástroj je velice mocný a zvládne velké množství různých funkcí. Bohužel je spíše zaměřený pro tvorbu webových aplikací a práce není příliš přizpůsobená pouze pro tvorbu statických stránek.

Jinja a Django templates jsou velice podobné šablonovací systémy, která mají velice podobnou syntaxi. Hlavní nevýhodou Django templates je nutnost instalace celého webového frameworku Django, aby bylo možné šablony zkompilovat a vytvořit výsledný HTML soubor. Proto byl vybrán šablonovací systém Jinja, který je velice populární a existuje pro něj přehledná dokumentace. Je také velice rychlý, takže i velké množství šablon, se vytvoří během chvilky. Na ukázce části šablon1 můžete vidět ukázku syntaxe. Jak je vidět, syntaxe je velice podobná syntaxi Pythonu, což velice zjednoduší práci. Mimo jiné Jinja také podporuje rozdělení šablon do jednotlivých souborů, což velice zpřehledňuje vytváření a editaci šablon. Díky takto vytvořeným šablonám si sami uživatelé výsledné šablony můžou upravit dle libosti a podle jejich potřeb.

### 5.2.4 Ostatní knihovny

Krom rozšiřujících knihoven byly také použity základní knihovny, které Python poskytuje. Mezi nejpoužívanější patří knihovna shutil, která umožňuje snadné kopírování souborů a vytváření složek. Dále knihovna datetime, která slouží pro uchování času a exportování do potřebných formátů a také základní knihovny os a sys, které slouží pro základní práci s operačním systémem, jako je například kontrola existence souborů, nebo správa návratových kódů aplikace.

## 5.3 Použité nástroje

V této sekci jsou popsány veškeré použité nástroje, které byly použity při vytváření bakalářské práce.

**PyCharm** Pro vývoj kódu aplikace byl využit program PyCharm od české firmy JetBrains. Hlavní výhodou tohoto vývojového prostředí je velké množství editovacích nástrojů, které velice urychlují práci a možnost rozšíření o další funkcionality. Mimo jiné také umožňuje snadné napojení na verzovací systémy jako jsou GIT nebo SVN. Během vývoje aplikace vzniklo několik nových verzí a byl vydán velký update pro rok 2017. Poslední použitá verze byla 2017.1.1.

**Enterprise architect** Tento program slouží pro tvorbu rozsáhlých a přehledných diagramů, které se používají při vytváření analytické dokumentace a při návrhu aplikace. Aplikace umožňuje modelování veškerých UML diagramů, které byly práci potřeba. Pro práci byla použita verze 12.0.1215

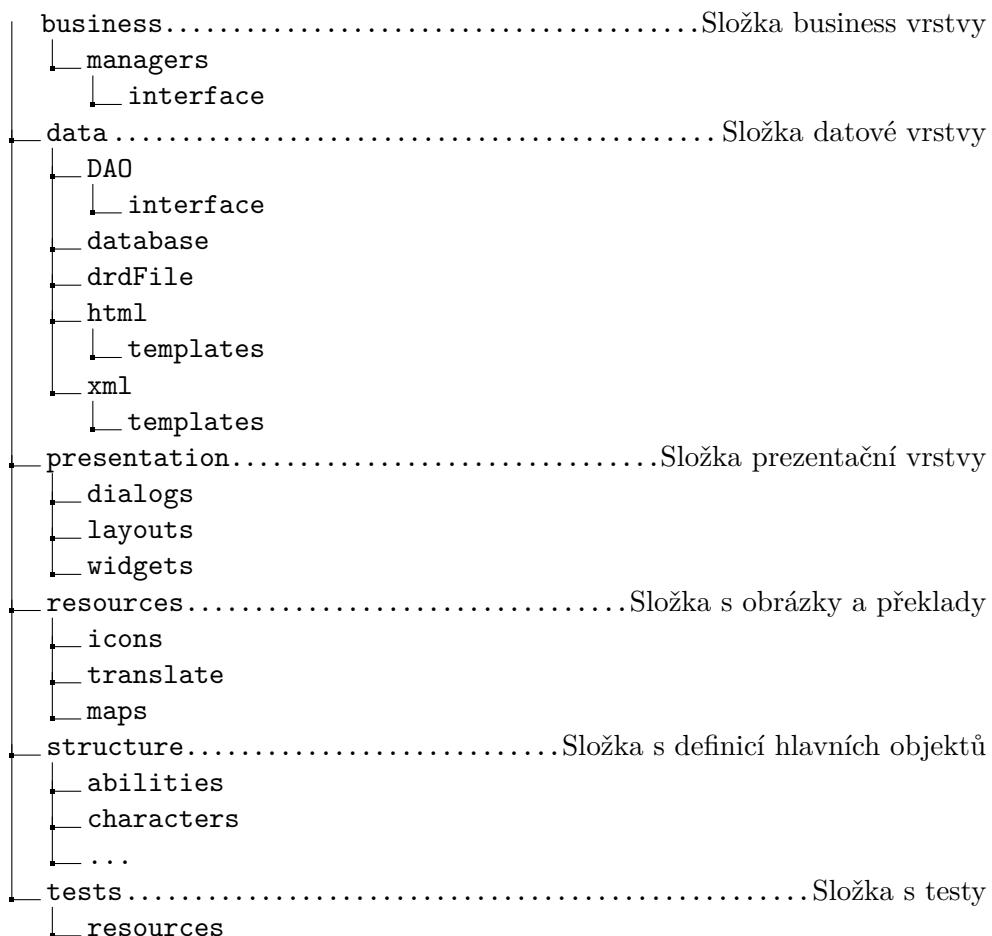
**GIT** Důležitou částí vývoje je také zálohování a verzování programu. Pro tyto účely byl využit verzovací systém GIT, který umožňuje jednoduchou a přehlednou správu verzí a možnost ukládat data na vzdálený server z důvodu zálohování. Pro verzování této práce byl využit GIT pro Windows verze 2.10.0

**MikTex** Celá textová část práce je psaná v jazyku L<sup>A</sup>T<sub>E</sub>X. Pro Windows existuje několik překladačů. Pro tuto práci byl zvolen program MikTex, který umožňuje jednoduchou a rychlou správu rozšiřujících balíčků, čím eliminuje problém s ručním stahováním potřebných balíčků, což na operačním systému windows bývá problém. MikTex byl použit ve verzi 2.9.6210

**TexMaker** Program TexMaker je velice přehledný a jednoduchý editor pro Latex. Umožňuje našeptávání syntaxe a také jednoduché zapnutí automatické opravy pro český jazyk, což nepodporují všechny editory. Pro práci byl využit TexMaker verze 4.5

## 5.4. Adresářová struktura projektu

---



Obrázek 5.1: Adresářová struktura projektu

## 5.4 Adresářová struktura projektu

Soubory v projektu jsou logicky rozdělené do složek pro snadnou orientaci v kódu. Na obrázku 5.1 je vyobrazena adresářová struktura celého projektu. Projekt je rozdělen do 6 hlavních složek.

**business** Složka `business` obsahuje všechny soubory týkající se business vrstvy. Soubory mají jméno složené z názvu objektu, kterého se třída týká, a slova manager. Složka obsahuje také podsložku `interface`, ve které se nacházejí veškeré interfaci pro business třídy.

**data** Složka `data` obsahuje veškeré soubory týkající se datové vrstvy. Nachází se zde třídy DAO, které se nacházejí ve složce `DAO` a jejich název končí stejným klíčovým slovem. Dále jsou zde třídy, které se starají o import a export do formátů XML, HTML a .drd, které jsou rozřazeny do

stejnojmenných složek. Ve složce `database` se nachází třídy pro základní práci s databází.

**presentation** Složka `presenation` obsahuje veškeré soubory, které definují prezentaci vrstvy. Soubory jsou logicky rozděleny do složek a hlavní soubory prezentace vrstvy se nachází přímo ve složce `presentation`. Ve složce `dialogs` se nachází definice veškerých dialogů a vyskakovacích oken aplikace. Ve složce `layouts` se nachází definice rozložení formulářů pro editaci veškerých šablon. Složka `widget` obsahuje definice hlavních částí uživatelského rozhraní programu.

**resources** Ve složce `resources` se nachází veškeré ikony a překlady pro aplikaci. Také se zde ukládají vygenerované výsledné mapy a zdrojové podkladové mapy.

**structure** Ve složce `structure` se nachází veškeré definice objektových tříd. Soubory jsou rozděleny do logických celků po složkách.

**tests** Složka `tests` obsahuje definice veškerých testů aplikace. Jméno souborů v této složce musí začínat prefixem `test`, aby bylo možné testy jednoduše hromadně pouštět. V podsložce `resources` se nachází soubory, které jsou potřeba pro testování, například vzorový XML soubor se šablonami pro porovnání výstupu testů.

## 5.5 Zajímavé části implementace

### 5.5.1 Nadstavba knihovny `lxml` pro mapování objektů

V ukázce kódu 2 můžeme vidět namapování objektů `Effect` a `Modifier` a jejich vzájemné provázání pomocí objektu `XInstance`.

Rozšíření obsahuje tři základní třídy, `XElement`, `XAttribElement` a `XInstance`. Každá z těchto tříd reprezentuje jiný typ atributu v objektu. Kombinací těchto tří základních objektů, můžeme vytvořit velice komplexní XML soubor.

Třída `XElement` reprezentuje základní atribut, který nemá překlady. V XML se jedná o tag bez atributů a bez vnořených tagů. Jako parametry přijímá název tagu a volitelný atribut `enum`, díky kterému se hodnota tagu upraví na základě zvolené enum třídy. Posledním volitelným parametrem je `date`, o jaký typ hodnoty tagu se jedná, například pokud je zapotřebí výstup naformátovat jako datum je třídě předán parametr `date`.

Třída `XAttribElement` reprezentuje tagy, které mohou navíc obsahovat atributy. Převážně se využívá atribut `lang`, který slouží pro definici jazyku překladu. Při importu šablony se tyto tagy vyhodnotí jako překladové a vytvoří se slovník hodnot, kde klíče slovníku určují jazyk a jeho hodnoty pak samotný objekt.

```

class XMLModifier(XMLTemplate):
    ROOT_NAME = 'modifier'
    OBJECT_TYPE = ObjectType.MODIFIER%

    def __init__(self):
        self.id = XElement('id')
        self.valueType = XElement('valueType', ModifierValueTypes)
        self.value = XElement('value')
        self.targetType = XElement('targetType')
        self.valueTargetAttribute = XElement('valueTargetAttribute')

class XMLEffect(XMLTemplate):
    ROOT_NAME = 'effect'
    OBJECT_TYPE = ObjectType.EFFECT

    def __init__(self):
        self.id = XElement('id')
        self.name = XAttribElement('name', 'lang')
        self.description = XAttribElement('description', 'lang')
        self.targetType = XElement('targetType')
        self.modifiers = XInstance('modifiers', XMLModifier)

```

Algoritmus 2: Mapování objektů na XML strukturu

Poslední třída XInstance reprezentuje vazbu na další mapovací třídu. Jako parametr komě jména, přijímá instanci cílové mapovací třídy. Objekt se při importu a exportu postará o rekursivní provolání do dalších tříd.

Třída XMLTemplate, ze které všechny mapovací třídy dědí, obsahuje dvě základní funkce, `import_xml` a `create_xml`. Funkce `import_xml` načte XML soubor a vrátí seznam objektů, které se dále načítají do databáze. Jednotlivé objekty jsou navíc rozděleny do slovníku podle jazyků, aby bylo možné importovat vícejazyčné překlady. O proti tomu třída `create_xml` přijímá seznam objektů, ze kterých vytvoří strukturovaný XML soubor.

Každá definice XML objektu musí dělit ze třídy `XMLTemplate` a navíc mít definované třídní atributy `ROOT_NAME`, který určuje název kořenového tagu a atribut `OBJECT_TYPE`, který určuje, o který objekt aplikace se jedná. Dále musí obsahovat třídní funkci `__init__`, která určí mapování atributů objektu na XML objekty. Princip mapování funguje na základě přiřazení třídního atributu, který definuje název atributu v objektu k mapovací XML třídě.

### 5.5.2 Soubor pro ukládání dat v aplikaci

Jelikož bylo nutné přesunout databázi ze souborového uložení pouze do paměti, z důvodu zrychlení aplikace, bylo zapotřebí navrhnout strukturu souboru pro uložení stavu celé aplikace, ze kterého se veškeré data dají snadno a rychle obnovit. Tento soubor neslouží pouze pro uložení rozpracované práce, ale také například pro sdílení dobrodružství pro ostatní uživatele.

Důležitá data aplikace se skládají ze dvou částí. První část je obsah databáze, kde jsou uloženy veškeré hodnoty, překlady a struktura všech šablon a objektů. Druhá část tvoří importované podklady pod mapu a uložené hotové mapy. Všechny tyto data a soubory je zapotřebí uložit do jednoho souboru, aby se s těmito daty dalo jednoduše pracovat. Jelikož soubor musí obsahovat i potřebné obrázky, není možné využít pouze textový formát ukládání.

Pro potřeby aplikace byl navržen nový soubor s tématickou koncovkou `.drd`. Struktura tohoto souboru je velice jednoduchá. Jedná se o obyčejný zip archiv, kterému je změněna koncovka. V tomto archivu se nachází složka s veškerými obrázky a textový soubor, který v sobě obsahuje veškerá data z databáze.

Při vytváření tohoto souboru se vytvoří kopie celé databáze, včetně struktury tabulek. Dále se zkopiují veškeré soubory ze složky `maps`, kde se nachází pouze používané obrázky map. Z těchto souborů se vytvoří zip archiv, kterému se následně změní koncovka na `.drd`. Při otevření tohoto souboru se naopak archiv rozbalí do dočasné složky, nakopírují se veškeré mapy na příslušné místo, spustí se SQL script, který vytvoří databázi a dočasné soubory se smaží.

Tento systém podporuje pouze ukládání a nahrávání všech dat v aplikaci a neumožňuje částečné spojení dat z více souborů. Pro tyto účely se dá využít export do formátu XML a následný import těchto souborů.

## 5.6 Vzniklé problémy při implementaci

Při implementaci vzniklo několik problémů, které se v návrhu nedali odhalit a bylo zapotřebí, některé části návrhu upravit nebo naopak rozšířit. Některé problémy se týkali samotné implementace, což analýza ani návrh nijak nezachycuje.

Na základě uživatelských testů, které jsou popsány dále, vyšlo na jeho, že největší problém aplikace se týkal rychlosti některých operací v aplikaci. Problém rychlosti se týkal převážně importu XML šablon do aplikace a překreslování stromové struktury šablon.

### 5.6.1 Problém s rychlostí importu z XML souborů

Na základě následně provedených měření bylo zjištěno, že import velkých dobrodružství mohlo trvat až několik desítek vteřin, což pro uživatele může být velmi otravné. Pro odhalení problému bylo provedeno podrobné měření

částí kódu, aby byla nalezena problematická část kódu. Z měření bylo jasné, že problém způsobuje rychlosť zápisu do databáze, který z důvodu získávání hodnot přiřazené hodnoty `id` zd databáze, nemohl být celý transakční, ale musel být rozložen na menší transakce. Hodnota `ID` je důležitá z důvodu mapování překladů na objekt. Tento problém byl vyřešen změnou umístění databáze. Z původního souboru, který se nacházel na pevném disku, byla celá databáze přesunuta do paměti. Tento přesun bohužel způsobil ztrátu veškerých neuložených dat po ukončení aplikace.

Aby byly ztráty dat eliminovány na minimum bylo současně do aplikace přidáno množství kontrolních dotazů v podobě kontrolních oken, které informují uživatele o případně ztrátě dat. Dále byl také zjednodušen a zrychlen systém ukládání a nahrávání `.drd` souborů, díky kterým se při znovaotevření aplikace dají data nahrát velice rychle zpátky.

### 5.6.2 Problém s rychlostí překreslování stromové struktury

Jelikož aplikace umožňuje třídění veškerých šablon do přehledné stromové struktury pomocí jednoduchého drag&drop systému, je zapotřebí, aby tato akce netrvala příliš dlouho. Jelikož při změně struktury stromu, je zapotřebí překreslit celý obsah widgetu, který zobrazuje stromovou strukturu, je zapotřebí tuto akci velice optimalizovat, protože se velmi často provádí.

Při základní implementaci se časová náročnost pohybovala v řádu sekund, při velkém množství uzlů ve stromové struktuře, což pro systém drag&drop je naprostě nepřijatelné. Problém zde byl analyzován do nejmenších detailů, aby bylo dosaženo přijatelné časové náročnosti. Po důkladné analýze vyšlo najevo, že jeden z největších problémů je velikost zobrazovaných ikon u jednotlivých položek.

Po zmenšení ikon více jak na jednu pětinu, byla doba překreslování snížena na méně než vteřinu, což již je přijatelná hodnota, ne však ideální. Pro ještě větší urychlení bylo zapotřebí upravit některé principy překreslování. Pokud by bylo nutné dosáhnout ještě lepšího výsledku, bylo by zapotřebí upravit některé principy překreslování stromové struktury a při změně dat nepřekreslovat celý strom, ale pouze jeho změněnou část. Jelikož však časová náročnost vzniká až při velice vysokém počtu elementů ve stromové struktuře, bylo toto řešení problému odsunuto do další verze aplikace.

### 5.6.3 Problém s křížovými závislosti

Další problém který vznikl, se týká problému křížových závislostí modulů nebo tříd. Jazyk Python je především používaný jako skriptovací jazyk, při kterém se obvykle pracuje pouze v jednom souboru, kde importování jiných souborů není potřeba. Při tvoření velkých projektů, které jsou přehledně rozděleny na velké množství modulů může vznikat problém s křížovými závislosti. Python pro importování jiných tříd neboli modulů používá dvě syntaxe.

## 5. IMPLEMENTACE

---

```
import data.database.Database  
  
data.database.Database().select(...)
```

Při tomto druhu importu nevznikají problémy s křížovými závislostmi, bohužel se při každém použití daného modulu nebo dané třídy musí pro její definici použít celá cesta. Při poměrně rozsáhlé adresářové struktuře, která byla využita v tomto projektu, se psaní celých cest stane nepříjemné a také velice nepřehledné. Proto pro takto velké projekty je tento způsob nepoužitelný.

```
from data.database.Database import Database  
  
Database().select(...)
```

O proti tomu pokud použijeme tento druhý způsob, již nemusíme psát celé cesty k třídám, což velice zkrátí zápis. Bohužel pokud nám zde vzniknou křížové závislosti, Python si s tím již neumí poradit. Jediná možnost, jak se tomuto problému vyvarovat, je použití lokálních importů. Jedná se o stejné importy jako v případě jedna nebo dvě, ale importy se nenachází na začátku souboru, jak jsme běžně zvyklí, ale přímo v konkrétní funkci, kde daný modul potřebujeme. V takovém případě Python provede import požadovaného modulu až ve chvíli zavolání funkce a nevznikají problémy s křížovou závislostí. Nicméně z hlediska přehlednosti kódu, by se veškeré importy, pokud je to možné, měli psát na začátek souboru.

# KAPITOLA **6**

---

## **Testování**

Poslední kapitola se věnuje testování vyvinuté aplikace. Testování aplikace bylo provedeno pomocí jednotkových testů a uživatelských testů. V kapitole je dopodrobna popsán způsob testování a rozebrány testované části. U obou skupin testů je popsán jejich účel, průběh testování a výsledky.

### **6.1 Jednotkové testy**

Jednotkové testy se zaměřují na testování jedné konkrétní třídy a její funkcionalit. Testy by měly fungovat nezávisle na ostatních třídách. Pokud se například testuje funkcionalita databáze, neměly by testy nijak ovlivnit produkční databázi. Unit testy byly rozděleny do 4 skupin po logických cílech. K tvorbě jednotkových testů pro Python se používá knihovna `unittest`.

#### **6.1.1 Testování databáze**

První skupina testů se věnuje přímému testování operací s databází.

##### **Účel**

Tato skupina testů má za úkol otestovat třídu `Database`, která má na starosti základní operace s databází. Testují se základní operace jako jsou `SELECT`, `DROP`, `DELETE` nebo `INSERT`. Testování probíhá na oddělené databázi, aby nedošlo k poškození aplikační databáze.

##### **Výsledky**

Tato skupina testů byla velice nápomocná při odhalování rychlostních problémů celé aplikace. Při spuštění testů je možné vidět i dobu, za kterou se jednotlivé testy provedly, což napomohlo k nalezení nejpomalejší části práce s databází a následné přesunutí databáze do paměti.

## 6. TESTOVÁNÍ

---

### 6.1.2 Testování DAO tříd

Další skupina testů se zabývá testováním důležitých DAO tříd. Některé třídy nejsou testované, protože je nelze testovat samostatně.

#### Účel

Hlavním účelem této skupiny testů je odhalení chyb při vytváření objektů z databáze a ukládání objektů do databáze. V testech se testují základní operace všech DAO tříd což jsou `create`, `insert`, `update` a `delete`. Pro každou DAO třídu se provede několik operací a výsledné objekty se porovnávají, zda jsou správně vytvořené, případně data v databázi upravené. Testy se provádí na stejně struktuře databáze na jaké běží aplikace, ale z důvodu neovlivnění dat aplikace běží v jiné databázi.

#### Výsledky

Na základě testování DAO tříd bylo odhaleno mnoho chyb při mapování dat na objekty. Převážně se jednalo o špatné přiřazování hodnot do objektu. Takto odhalené chyby by se bez těchto testů velmi špatně odhalovali.

### 6.1.3 Testování importu a exportu XML

Třetí skupina testů se zabývá kontrolou správné práce s XML soubory.

#### Účel

Hlavním účelem těchto testů je kontrola správné práce s XML soubory jak při importu do aplikace tak při exportu z aplikace.

Při testování XML souborů do aplikace se využívá připravené šablony. V aplikaci se pustí import dat pro danou šablonu a kontrolují se výsledné objekty, které při importu vzniknout. Data se kontrolují pouze na základě objektu, neukládají se do databáze. Kontrolují se veškeré hodnoty všech objektů, kromě hodnoty `id`, která se může měnit. Samozřejmě se také kontroluje správné zanořování objektů.

Při exportu objektů do XML souboru se kontroluje struktura výsledného XML souboru. Aby bylo možné výsledné soubory porovnávat, vzorový soubor a výstupní soubor jsou patřičně seřazeny podle abecedy. Kontroluje se přítomnost všech potřebných tagů, které aplikace MobChar vyžaduje a také jejich hodnot.

#### Výsledky

Na základě testování práce s XML soubory bylo odhaleno množství chyb, které by se dali snadno odhalit při uživatelských testech, ale při tomto testování je snadnější odhalit konkrétní chybu a není potřeba zdlouhavě komunikovat

s mobilní aplikací. Díky těmto testům bylo výrazně sníženo množství chyb, které se museli opravovat na základě uživatelských testů.

### 6.1.4 Testování .drd souboru

Poslední skupina testů testuje vytváření .drd souboru a jeho správný obsah.

#### Účel

Hlavním účelem těchto testů je kontrola vytváření a správného obsahu souboru .drd. Testy využívají připravené databáze, ze které soubor vytvoří. Soubor se vytvoří do dočasné složky, kde se také rozbalí. Zkontrolují se veškerá přítomná data která jsou tvořena z databázového souboru a složky s mapami. Po skončení testů se veškeré dočasné soubory smažou, aby neovlivnili další puštění testů.

#### Výsledek

Na základě těchto testů máme zajištěno, snadné odhalení potencionální chyby, pokud bychom změnily strukturu adresářů nebo jiný větší zásah do aplikace.

## 6.2 Uživatelské testování

Další možností testování aplikace jsou uživatelské testy. Uživatelské testy mají simulovat praktické využití aplikace. Na rozdíl od jednotkových testů, kde veškerou funkcionality testuje počítač, uživatelské testy provádějí lidé. Uživatelské testy se provádějí na základě připravených scénářů.

Pro případy testování se zpravidla připravuje speciální verze programu, které veškeré chování zaznamenává, aby bylo možné snadněji odhalit chyby. Testovací uživatel dostane scénáře, které mu říkají, jaké činnosti má v aplikaci vyzkoušet. Pokud vše funguje jak je uvedeno ve scénáři, uživatel ohodnotí test jako splněný, v opačném případě test označí jako nesplněný a do komentáře napíše problém, který vznikl. Uživatel také může do komentáře napsat i připomínky v případě, že scénář proběhl v pořádku, ale některé věci se nechovají jak by očekával.

### 6.2.1 Účel

Hlavním účelem uživatelských testů, které byly vytvořeny pro aplikaci, bylo otestování veškerých scénářů, které jsou popsány v analýze 3.3. Pokud veškeré scénáře uživatelkým testováním projdou, můžeme na základě tabulky, která popisuje pokrytí požadavky, rozhodnout, že aplikace splňuje veškeré požadavky, které byly na začátku stanoveny.

## 6. TESTOVÁNÍ

---

### 6.2.2 Průběh

Jelikož aplikace DeskChar nevzniká jako samostatná aplikace, mělo by uživatelské testování probíhat v rámci celého balíčku. Velká část funkcionality aplikace se týká exportování XML souborů pro mobilní aplikace, které bez této aplikací není možné vyzkoušet. Pro testovacího uživatele byl vytvořen testovací balíček, který obsahuje veškeré potřebné instalacní soubory pro nainstalování celého balíčku aplikací, krátký popis instalace a veškeré testovací scénáře.

Z důvodu velkého množství testovacích scénářů pro všechny aplikace, jsou v této práci uvedeny pouze scénáře, týkající se aplikace DeskChar. Ostatní scénáře jsou podrobněji popsány v pracích Matěje Sháněla [1] a Šárky Weberové [2].

#### 6.2.2.1 Práce se šablonami

První z testovacích scénářů se týká vytváření šablon pro aplikaci MobChar. Tyto šablony se pak dále využívají také při tvorbě dobrodružství. Uživatel testuje základní práci se šablonami v aplikaci. Postup je totožný pro všechny druhy šablon, proto je popsán pouze scénář pro jeden druh šablon.

Uživatel si vytvoří libovolné množství předmětů. U každého předmětu vyplnění všechny potřebné údaje a šablony uloží. K některým šablonám vytvoří alespoň další dva překlady. Následně některé šablony smaže. Zkontroluje zda všechny údaje jsou uložené správně, překlady jsou správně uložené a všechny číselné hodnoty se mění naráz na všech záložkách.

Pokud vše funguje jak má, testovací uživatel předmětů přidá další objekty, které je možné pod šablony přidat. V případě předmětů se jedná o efekty nebo jiné předměty. Šablony efektů si uživatel vytvoří v záložce efekty stejným postupem, jako vytvářel předměty. Předmětů přidá libovolné množství efektů nebo předmětů. Následně vyzkouší vytvořené efekty přesunout k jiným předmětům pomocí drag&drop.

Poslední část scénáře se týká vytvoření adresářové struktury. Testovací uživatel si vytvoří alespoň 3 adresáře, které do sebe vnoří a následně do této složek přesune předměty a v některých složkách vytvoří nové předměty. Nakonec uživatel přesune složku včetně obsažených předmětů do kořene stromové struktury. Testovací uživatel zkонтroluje, zda se veškeré operace se stromovou strukturou provedli správně. Pokud ano, scénář testující práci se šablonami zde končí úspěchem.

#### 6.2.2.2 Vytváření mapy

Další scénář se týká vytváření mapy a operace s mapou. Testovací uživatel se přepne do záložky Mapa. Ve adresářové struktuře vytvoří novou mapu, vyplní název mapy a dvojklikem se přepne na její editaci. Pomocí ikony Přidat podkladovou mapu, vybere soubor, který obsahuje libovolný obrázek. Tento

obrázek by měl mít rysy mapy, aby přidávání objektů na mapu mělo smysl. Následně zkusí přidat nový podkladový obrázek, přičemž starý obrázek je tímto nahrazen.

Testovací uživatel dále na mapu přidá libovolné množství dalších objektů. Pro každý objekt vybere jeho typ a vyplní základní údaje objektu. Objekt se vytvoří na mapě. Uživatel následně objekt zvětší nebo zmenší podle potřeby a přesune na potřebné místo. Pokud se jedná o větší mapu, uživatel může využít funkcí na přiblížení a oddálení pro lepší práci s mapou. Po přidání několika objektů také některé předměty smaže. Následně mapu uloží a zkонтroluje, zda celá mapa zůstala ve stejném stavu a všechny údaje jsou správně uloženy. Pokud vše souhlasí, testovací scénář zde končí úspěchem.

### 6.2.2.3 Vytváření dobrodružství

Další scénář navazuje na předchozí scénáře, protože využívá vytvořené šablony a mapy v předchozích scénářích. Pokud je testování provedeno samostatně je zapotřebí vytvořit alespoň nějaké šablony a mapy podle předchozích scénářů. Testovací uživatel vytvoří nové dobrodružství. Do dobrodružství přidá libovolné množství předmětů, lokací, kouzel, schopností a postav. Všechny tyto objekty může libovolně roztrídit do složek pro větší přehlednost. Některé objekty zedituje přímo v záložce dobrodružství pomocí dvojkliku na daný objekt ve stromové struktuře. Pokud všechny možné operace se provedou správně, testovací scénář zde úspěšně končí.

### 6.2.2.4 Práce s formátem XML

Tento scénář převážně testuje funkčnost komunikace s mobilními aplikacemi MobChar, jak s balíčkem pro hráče tak s balíčkem pro Pána jeskyně. Pro potřeby testování se opět využívají vytvořené šablony a objekty z předešlých scénářů. Pokud testování probíhá samostatně, je nejdříve zapotřebí vytvořit dostatečné množství šablon a objektů, nejlépe alespoň dvě od každého druhu, podle předchozích scénářů.

První část scénáře se zaměřuje na komunikaci s hráčskou aplikací. Uživatel postupně pro každý druh šablon provede stejnou operaci a také vyexportuje alespoň jednu postavu. Přepne se do sekce pro danou šablonu a zvolí možnost exportovat. Ve stromové struktuře se objeví zaškrťávací políčka, pomocí kterých vybere požadované šablony pro export. Po dokončení výběru uživatel vybere možnost **Export XML**, kde v otevřeném okně vybere cílový soubor, do kterého se šablony vyexportují.

Výsledné soubory nahraje do zařízení, ve kterém se nachází nainstalovaná aplikace MobChar, na patřičná místa. Následuje testovací scénář, který se týká aplikace MobChar, který je detailně popsán v práci Šárky Weberová [2]. Po provedení příslušného testování mobilní aplikace vytvoří soubory XML, které je možné zpět do naší aplikace nahrát.

## 6. TESTOVÁNÍ

---

Uživatel se přepne do záložky, která se týká šablon, které chce importovat. Pokud se v XML souboru nachází i šablony jiného druhu, tyto šablony jsou vynechány. Uživatel vybere tlačítko **Importovat** a v novém okně vybere zdrojový soubor. Aplikace provede import všech šablon správného typu a zobrazí nově vytvořené šablony. Uživatel zkонтroluje, zda obsahují všechny údaje, které aplikaci byly předány pomocí XML souboru. Uživatel provede tuto operaci pro všechny druhy šablon a také pro postavu. V poslední části vyzkouší importovat poškozený XML soubor, přičemž by aplikace měla vytvořit chybovou hlášku, která informuje o nezdařeném importu.

Velice podobný průběh má scénář, který testuje komunikaci s aplikací pro Pány jeskyně. Testovací uživatel se přepne do záložky **Dobrodružství** a vybere tlačítko export. Zvolí požadované dobrodružství a vybere možnost **Export XML**. V novém okně vybere cílový soubor, kam se výsledek exportu uloží. Výsledný XML soubor nahraje do mobilního zařízení, kde se nachází nainstalovaná aplikace MobChar.

Dále probíhá testování na základě scénářů definované v práci Matěje Sháněla [1]. Po provedení veškerých testů týkajících se dobrodružství, aplikace vytvoří nový XML soubor, který je možné zpět nahrát do aplikace DeskChar. Testovací uživatel výsledný soubor nakopíruje zpět na testovací zařízení.

V aplikaci v záložce **Dobrodružství** vybere možnost **Importovat** a zvolí zdrojový soubor. Po provedení úspěšného importu, se v záložce zobrazí nové dobrodružství. Veškeré části dobrodružství by měli být na správncích místech a obsahovat všechny předané údaje. Import dobrodružství neumožňuje vytvoření map, proto se mapy do dobrodružství nevytvorí. Uživatel zkонтroluje správnost všech údajů a celé struktury. Pokud vše souhlasí testovací scénář zde končí úspěchem.

### 6.2.2.5 Práce s formátem .drd

Krátký testovací scénář se týká testování práce se soubory s koncovkou .drd. Uživatel si v aplikaci vytvoří libovolné množství šablon a objektů. V menu zvolí možnost **Uložit**. Pokud aplikace nebyla ze souboru .drd otevřena, objeví se nové okno pro zadání cíle pro uložení. Po zvolení cílového souboru a potvrzení jsou data do souboru uloženy. Následně uživatel upraví některé šablony v aplikaci. Následně pokud opět zvolí možnost **Uložit**, aplikace by se již neměla na umístění zeptat. Je možné však data uložit na jiné umístnění pomocí možnosti **Uložit jako**.

Uživatel aplikaci zavře a znova otevře. Všechny data z aplikace jsou smazána. Uživatel zvolí možnost **Otevřít** a v novém okně vybere dříve vytvořený soubor. Aplikace data načte a znova vytvoří všechny uložené šablony a objekty. Při ukládání do formátu .drd je možné zachovat i mapy, tudíž při otevření souboru se obnoví i veškeré uložené mapy. Testovací uživatel zkонтroluje přítomnost a správnost všech dat. Pokud vše souhlasí, končí zde testovací scénář úspěchem.

### 6.2.2.6 Vytváření HTML almanachů

Poslední testovací scénář se týká vytváření HTML almanachů. Uživatel opět potřebuje mít vytvořené šablony a objekty z předchozích scénářů. Testování je provádí na všech šablonách a objektech. Scénáře jsou totožné pro všechny záložky.

Testovací uživatel se přepne do konkrétní záložky a vybere možnost **Exportovat**. V zobrazených zaškrťvacích políčkách vybere objekty, které chce exportovat. Následně vybere možnost **Export HTML** a v novém okně vybere cíl, kam se výsledné soubory uloží. Aplikace na dané místo vygeneruje potřebné soubory.

Uživatel otevře soubor `index.html`. Soubor zkонтroluje, že obsahuje všechny zvolené šablony a u každé všechny vyplněné parametry. Hlavní cílem tohoto scénáře je zjistit, zda jsou výsledné almanachy přehledné a dobře uzpůsobené pro tisk. Pokud uživatel uzná, že všechny údaje jsou uvedené a almanach je dostatečně přehledný, testovací scénář zde končí úspěchem.

### 6.2.3 Výsledky

Pro uživatelské testy bylo domluveno několik Pánů jeskyně a také dvě kompletní skupiny hráčů. Všem testovacím uživatelům byl předán stejný testovací balíček, včetně veškerých potřebných scénářů. Testovací uživatelé si scénáře prošli a poslali zpět odezvu. Odezva se týkala všech testovaných aplikací. V této práci jsou uvedeny pouze odezvy, týkající se aplikace MobChar. Ostatní jsou možné k nahlédnutí v práci Matěje Sháněla[1] a v práci Šárky Weberové [2].

Uživatelé, především Páni jeskyně, si aplikaci DeskChar vyzkoušeli a dopodrobna vyzkoušeli všechny funkcionality a prošli všechny vytvořené scénáře.

Dle testovacích uživatelů byl největší problém aplikace její rychlosť. Některé činnosti trvali příliš dlouho, což velice zdržovalo práci s aplikací. Z tohoto důvodu byl věnován čas na vylepšení časové náročnosti některých operací, který je popsán v kapitole implementace ??.

Dále uživatelé narazili na drobné chyby, týkající se správného nastavování hodnot u šablon modifikátorů. Tyto problémy byly na základě odezvy opraveny.

Bez ohledu na nalezené drobné chyby byli všichni testovací uživatelé s aplikací spokojeni. Některí uživatelé poslali návrh na nové funkce, které by aplikace mohla umožňovat. Tyto funkcionality byly zaevidovány, avšak v této práci již nebudou realizovány.



---

# Závěr

Cílem této práce bylo vytvořit desktopovou aplikaci DeskChar pro všechny hráče Dračího doupěte v souladu se standardy softwarového inženýrství.

V první fázi vývoje aplikace byla vytvořena analýza již existujících aplikací, se kterými výsledná aplikace spolupracuje. Byly analyzovány všechny požadavky na aplikaci a vytvořeny scénáře užívání. Na základě této analýzy vznikl podrobný návrh realizace. Dopodrobna byl popsán návrh tříd, databáze a architektury celého programu.

Na základě analýzy a návrhu byl program implementován, včetně veškeré dokumentace a popisu tříd. Následně pomocí jednotkových a uživatelských testů byla ověřena funkcionality celé aplikace a splnění všech analyzovaných požadavků. Po dokončení celé implementace vznikla k aplikaci uživatelská příručka, která budoucím uživatelům usnadní práci s aplikací. Uživatelská příručka je k nalezení na přiloženém CD.

Vytvořením aplikace podle standardů softwarového inženýrství a dodáním plné dokumentace a uživatelské příručky bylo splněno zadání bakalářské práce.

Vzniklá aplikace je pouze ve verzi 1.0 a má velký potenciál pro zlepšování. Hlavní směr vývoje pro verzi 2.0 by se měl týkat mapy. Ve verzi 1.0 je editor mapy pouze základní a aplikace zde nabízí velké množství pro zlepšení.



---

## Literatura

- [1] Sháněl, M.: *MobChar - balíček pro Pána jeskyně v Dračím doupěti*. Bakalářská práce, České vysoké učení technické, Fakulta informačních technologií, Praha, 2017.
- [2] Weberová, r.: *MobChar - balíček pro Dračí doupě 3.0*. Bakalářská práce, České vysoké učení technické, Fakulta informačních technologií, Praha, 2017.
- [3] Klíma, M.: Nakladatelství Altar. 2017. Dostupné z: <http://www.altar.cz/>
- [4] *Pravidla pro Dračí doupě pro Pána jeskyně, verze 1.5*. Altar, první vydání, 1994.
- [5] Bronec, M.; Vorlová, M., Jiřina a Klíma: *Pravidla pro Dračí doupě, verze 1.5*. Altar, první vydání, 1994.
- [6] Sovko, A.: Dungeon maps for RPG. Create maps online, download as PDF and PNG. 2017. Dostupné z: <http://pyromancers.com/>
- [7] Arneson, D.; Stucki, D.; Gygax, G.; aj.: Donjon, RPG Tools. 2017. Dostupné z: <https://donjon.bin.sh/>
- [8] Behnel, S.: lxml - Processing XML and HTML with Python. 2017. Dostupné z: <http://lxml.de/>
- [9] Bayer, M.: Mako, Hyperfast and lightweight templating for Python. 2017. Dostupné z: <http://www.makotemplates.org/>
- [10] RONACHER, A.: Jinja2, The Python Template Engine. 2017. Dostupné z: <http://jinja.pocoo.org/>



## Seznam použitých zkratek

**bash** Born again shell

**DAO** Data access object

**DeskChar** Desktop character

**DrD** Dračí doupě

**enum** Enumeration

**exe** Executable

**HnH** Hra na hrdiny

**HTML** Hypertext Markup Language

**MobChar** Mobile character

**PJ** Pán jeskyně

**UML** Unified Modeling Language

**RPG** Role-playing game

**XML** Extensible Markup Language



## Obsah přiloženého CD/USB

```
readme.txt.....stručný popis obsahu CD
├── doc.....dokumentace
│   ├── model.eap ..... Soubor veškeré modelové dokumentace
│   └── UserGuide.pdf ..... Uživatelská příručka
├── exe ..... adresář se spustitelnou formou implementace
│   ├── Windows ..... Spustitelná verze pro Windows
│   └── Linux ..... Spustitelná verze pro Linux
└── src
    ├── impl ..... zdrojové kódy implementace
    └── thesis ..... zdrojová forma práce ve formátu LATEX
└── text ..... text práce
    ├── thesis.pdf ..... text práce ve formátu PDF
    └── thesis.ps ..... text práce ve formátu PS
```