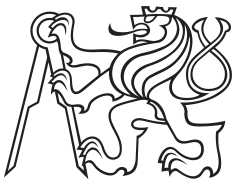
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Nástroj na vývoj a analýzu gamebooků

Jakub Hrdoun

Vedoucí: RNDr. Ladislav Serédi
Specializace: Základy umělé inteligence a počítačových věd
Studijní program: Otevřená informatika
Květen 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hrdoun** Jméno: **Jakub** Osobní číslo: **492295**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Otevřená informatika**

Specializace: **Základy umělé inteligence a počítačových věd**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Nástroj na vývoj a analýzu gamebooků

Název bakalářské práce anglicky:

Tool for Developing of Gamebooks

Pokyny pro vypracování:

Prostudujte dostupné nástroje pro vytváření a konzumace interaktivních herních příběhů (gamebooků). Diskutujte stávající řešení, jejich výhody a nevýhody. Implementujte nástroj obsahující:
• uživatelské rozhraní pro konzumaci příběhů,
• editor příběhu s možností ladění,
• analytické funkce poskytující zpětnou vazbu od uživatelů.
Při návrhu nástroje mějte na zřeteli následující aspekty:
• uživatelsky přívětivý editor příběhů, použitelné i bez znalostí programování,
• implementace komplexnějších možností nad rámec běžného rozvětveného příběhu, tj. (správa proměnných, podmíněnost větvení na kontextu, kontrola a vizualizace herního grafu),
• jednoduše dostupné a atraktivní rozhraní pro konzumaci příběhu.
Vaše řešení otestujte – vytvořte v něm interaktivní příběh adekvátně využívající Vámi implementované možnosti, poté získejte a zpracujte zpětnou vazbu od uživatelů (hráčů – čtenářů). Porovnejte možnosti vámi navrženého a stávajícího řešení z hlediska obecnosti, uživatelské přívětivosti a to jak z hlediska autora, tak z hlediska čtenáře příběhu. Analyzujte a diskutujte jednotlivé aspekty nástroje, zejména jednoduchost nasazení, spolehlivost, snadnost vytvoření obsahu, množství pokročilejších možností, srovnejte jejich výhody a nevýhody.

Seznam doporučené literatury:

[1] Mei Si, David Thue, Elisabeth André, James Lester, Theresa Jean Tanenbaum, Veronica Zammitto – Interactive Storytelling – Vancouver, Canada, 2011
[2] Open-source tool for telling interactive, stories [online]. @2021 Twinery [13.12.2021] https://twinery.org/
[3] Nick Montfort - Twisty Little Passages: an approach to interactive fiction – The MIT Press, 2005

Jméno a pracoviště vedoucí(ho) bakalářské práce:

RNDr. Ladislav Serédi kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **13.12.2021** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

RNDr. Ladislav Serédi podpis vedoucí(ho) práce	prof. Ing. Tomáš Svoboda, Ph.D. podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

 © ČVUT v Praze, Design: ČVUT v Praze, VIC

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

 © ČVUT v Praze, Design: ČVUT v Praze, VIC

Poděkování

Rád bych poděkoval RNDr. Ladislavu Serédimu za jeho čas a nejeden cenný poznatek, který tuto práci velmi obohatil.

Nesmím zapomenout ani na naší fakultu, jenž mi poskytla kvalitní vzdělání a prostor pro jeho rozvíjení.

A v neposlední řadě své rodině a přítelkyni, kteří mě během studia podporují a na něž se mohu v nelehkých chvílích obrátit.

Děkuji.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických postupů při přípravě vysokoškolských závěrečných prací.

V Praze, 10. května 2022

Abstrakt

Práce se zabývá nástrojem pro uživatelsky přívětivé vytváření interaktivních příběhů, jejich konzumaci a následnou podrobnou analýzu pro autory. Cílem je nejprve zmapovat existující nástroje, zjistit jejich výhody i nedostatky a na tomto základě navrhnout vlastní nástroj. Bude se jednat o webovou aplikaci pro konzumaci interaktivních příběhů a pro jejich vytváření. Během procesu vyváření interaktivního příběhu aplikace analyzuje herní graf, a poskytuje autorovi okamžitou zpětnou vazbu. Práce v teoretické části navrhne algoritmy pro analýzu herního grafu, pak přikročí k jejich implementaci. Aplikace bude též schopná vytvářet statistiku o pohybu a činnosti všech uživatelů-čtenářů v interaktivním příběhu, což autorovi může poskytnout cenné informace a posloužit ke zkvalitnění tvorby. Po implementaci nástroje proběhne uživatelské testování pomocí k tomu účelu vytvořeného přiměřeně komplexního interaktivního příběhu.

Klíčová slova: interaktivní příběh, gamebook, textová hra, herní graf

Vedoucí: RNDr. Ladislav Serédi

Abstract

The thesis deals with a tool for user-friendly creation of interactive stories, their consumption and subsequent detailed analysis for authors. The goal of this thesis is to map existing tools, identify their advantages and shortcomings and on this basis design and develop a new tool. The result is a web application for consuming and creating interactive stories. During the writing an interactive story the application analyzes the game graph and provides immediate feedback to the author. The theoretical part of the thesis includes proposals and implementation of algorithms for analyzing the game graph. The application generates statistics about the movement and activity of all user-readers in the interactive story, which can provide valuable information to the author and may serve to improve the design. Once the tool has been implemented, a reasonably complex interactive story, created for this purpose, will be tested by real users.

Keywords: interactive story, gamebook, text game, game graph

Obsah

Obrázky

Tabulky

Ukázky kódů

Kapitola 1

Úvod

1.1 Co je to gamebook?

Gamebook je literární útvar, který čtenář nekonzumuje lineárně od začátku do konce jako obyčejnou knihu, ale ovlivňuje její průběh svými volbami. Gamebook je tedy textovou hrou, interaktivním příběhem. Nejde ale jen o prosté volby, ale i o vnitřní mechanismy hry. Čtenář svým chováním během čtení nastavuje kontext hry, na jehož základě se hra dále vyvíjí.

1.2 Gamebooky dnes

V současné době část gamebooků vychází v papírové podobě jako obyčejná kniha, můžeme zmínit české nakladatelství Mytago, které se přímo na tuto literaturu zaměřuje. Velká část vycházejících gamebooků spadá do žánru fighting fantasy. V této práci se ale budu věnovat gamebooku jako konceptu, který je schopen poskytnout jakýkoli typ příběhu, nehledě na žánr. [1]

Příkladem takové knihy může být *Verax: Experiment*, kde se čtenář vžije do role člena posádky ztroskotalé vesmírné lodi na neznámé stanici. Prochází stanicí, komunikuje a bojuje s ostatními postavami, které potká, a jeho cílem je se z vesmírné stanice dostat pryč. [2]

Jako další lze zmínit *Zapadákov*, kde má čtenář na starosti malé město v prostředí westernu, které má za úkol obhospodařovat a rozvíjet. Jedná se o gamebook v podobě komiksu, to ale na principech interaktivního příběhu nic nemění. [3]

Kromě beletristických žánrů lze koncept interaktivního příběhu použít i v populárně naučné literatuře. Například textová hra *depression quest*, kde se čtenář vžije do role člověka trpícího depresí. Účelem je čtenáře vzdělat a názorně vysvětlit, co skutečná deprese znamená. Tento gamebook je jeden

z řady, kdy je hra vyvíjena od začátku jako samostatný software. Protože se ale v gameboocích podobné mechanismy opakují, má smysl uvažovat o nástroji, který poskytne možnost tvorby libovolného příběhu. [4]

1.3 Proč je třeba nástroj?

Jaké jsou důvody pro vytvoření nástroje, když lze gamebook jako obyčejnou knihu vytisknout?

1.3.1 Z pohledu čtenáře

Během čtení musí čtenář knihou tam a zpět listovat a poznamenávat si množství měnících se informací. Podívejme se blíže na dva výše uvedené příklady knižních gamebooků.

V případě knihy *Verax:Experiment* si během čtení čtenář musí do přiložených tabulek zapisovat hodnoty svého zdraví, náskoku před pronásledovateli, body uložení hry, předměty, jež vlastní, a další. [2]

Nejinak tomu je v komiksovém gamebooku *Zapadákov*. Tam je dokonce k dispozici podrobná mapa, kam čtenář zakresluje objevená území a postavené budovy, poznamenává si obnos peněz a aktuální měsíc v roce. [3]

1.3.2 Z pohledu autora

Jelikož je gamebook nelineární příběh, tvoří herní graf. S rostoucí velikostí díla je čím dál komplikovanější se v něm při tvorbě a korekturách orientovat, pokud ho tvoříme v obyčejném textovém editoru a mezi jeho částmi se pohybujeme pouze pomocí čísel stránek nebo odkazů v textu. Tato funkcionalita by se nicméně dala zařídit i bez čtenářské strany; autor by ve speciálním nástroji pro vývoj gamebooků příběh vytvořil a následně exportoval do textového formátu, jenž by putoval do tiskárny.

Autor, který se ve své práci chce zlepšovat, potřebuje podrobnou zpětnou vazbu. Může se orientovat podle počtu prodaných výtisků, podle odborných i čtenářských recenzí, ale na drobné niance, kterých je gamebook plný a z kterých se skládá, v případě knižního formátu nikdy nedostane dostatečně podrobnou zpětnou vazbu.

Nástroj tedy autorovi bude poskytovat kromě funkcionalit při prvotním psaní také podrobnou statistiku o tom, jak se čtenáři grafem jeho gamebooku pohybují. Pokud se tedy například dozví, že jednou rozsáhlou větví prošlo pouze 5% čtenářů, vidí jasný nevyužitý potenciál příběhu, s čímž může pracovat a gamebook postupně vylepšovat.

Kapitola 2

Funkcionality nástroje

Protože nástroj pro vývoj a analýzu gamebooků najde své využití, shrnu funkcionality, které musí obsahovat, aby mohl být oproti knižní variantě výhodou.

Nástroj by měl být rozdělen do dvou hlavních částí; čtenářského prostředí pro konzumaci interaktivních příběhů a editoru pro jejich tvorbu. Důležitým požadavkem je možnost vytvářet příběhy bez znalosti programování pomocí jednoduchých příkazů v textu hry. Důraz by měl být kladen na jednoduchost editoru, aby ho mohl uživatel neprogramátor využívat, aniž by přečetl několik stran rozsáhlé dokumentace (na rozdíl od již existujících projektů). Nesmíme opomenout ani jednoduchost a přímočarost příkazů, pro jejichž pochopení by měla stačit krátká nápověda, která bude po ruce přímo v editoru.

2.1 Základní funkcionality pro práci s kontextem hry

- **Výpis kontextu hry přímo do textu.** (Příklad: `V tuto chvíli máš v peněžence x Kč`).
- **Podmíněné vykreslení.** Vykreslení části textu nebo dokonce určitých voleb pouze pro čtenáře, kteří mají určitou hodnotu kontextu hry. (*Čtenář se v příběhu nachází v knihkupectví a spatří učebnici teorie grafů, která stojí 199 Kč. Volba "Koupit učebnici"se zobrazí jen těm čtenářům, kteří mají v aktuálním kontextu hry dostatek peněz.*)
- **Úprava kontextu hry.** Změna hodnot proměnných. *(Po koupi učebnice odečteme 199 Kč z čtenářova aktuálního kontextu hry).*
- **Textový vstup od čtenáře.** Čtenář zadá textový/číselný vstup nebo vybere položku ze seznamu, která se uloží do proměnné kontextu hry. Seznam může být vypočten na základě kontextu hry, nemusí být statický *(Čtenář může koupit pouze takové knihy, na které má dostatek financí.)*

2.2 Pokročilá analýza gamebooku

- **Konečnost hry.** Gamebook tvoří graf. Možná by se chtělo říci strom, ve skutečnosti to je ale skutečně cyklický graf, ale pouze pokud se průchodem cyklem změní kontext hry a není možné cyklem procházet donekonečna. Zda-li graf tuto podmínku splňuje, je třeba umět detekovat.
- **Dosažitelné kontexty hry**. Představme si gamebook jako herní graf, jehož uzly tvoří scény a hrany mezi nimi přechody mezi scénami. V herním grafu je jeden počáteční uzel, v kterém čtenář vždy začíná číst. Pokud autor pracuje na scénách vzdálených od startovního uzlu s kontextem hry, který už byl mnohokrát upraven, může jen těžko a zdlouhavě počítat, jakých hodnot může kontext nabývat. Tato funkcionalita za něj možné hodnoty jednotlivých stavů dovede pro každou scénu spočítat. *(Čtenář během hry mnohokrát o peníze přichází a nabývá jich. Pak se dostane do fáze, kdy se mu nabízí koupě učebnice. Autor v tuto chvíli musí učebnici "nacenit" tak, aby byla dosažitelná, ale (možná) zároveň ne pro ty kontexty hry, v kterých hráč hodně utrácel. Potřebuje tedy znát všechny možné hodnoty stavů v kontextu hry.)*
- **Nedokončené listy.** Během vývoje autorovi pod rukama přibývají scény bez pokračování, které nejsou zatím dokončené. Autor potřebuje znát jejich seznam, aby měl přehled o stavu práce.
- **Vizualizace.**
  - Vykreslení grafu příběhu.
  - Možnost přehledně zobrazit změny a použití jednotlivých stavů z kontextu hry.
  - Vykreslení četností průchodů do grafu gamebooku. Zvýraznění částí hry, kam se žádní čtenáři nedostali.

2.3 Další funkcionality

- **Zpožděný úsek.** Úsek textu a příkazů, který se vyhodnotí se zpožděním. Pokud autor chce, aby se text scény objevoval postupně, ne najednou, nebo pokud chce aby se čtenář rozhodl v časovém limitu. Pokud se nerozhodne, zpracuje se úsek textu, který například nastaví kontext hry tak, aby bylo jasné, že se čtenář nerozhodl.

Kapitola 3

Existující nástroje

Nabízí se řada mobilních i webových aplikací obsahujících interaktivní příběhy ale bez možnosti vytváření vlastních gamebooků. Například zmíněný *depression quest*, z mobilních aplikací *Mythion* nebo *Gamebook Adventures*. Tyto projekty, poskytující pouze čtenářské rozhraní ale zabírají jen malou část mého projektu, proto se zaměřím na následující nástroje obsahující zároveň i autorské prostředí. Nebudeme zde uvažovat placené platformy. [4, 5, 6]

3.1 Přehled nástrojů

3.1.1 FGBE (Fantasy GameBook Engine)

Obsahuje řadu pokročilých funkcí. Autor své dílo upravuje v textových souborech s kořenovou strukturou připomínajících XML. Chybí možnost vizualizace. Tento nástroj patří už spíše do historie. [7]

3.1.2 crumblyheadgames.co.uk

Jednoduchý editor, který neumí pracovat s kontextem hry. Ve verzi zdarma umožňuje jen omezené množství úseků textu. [8]

3.1.3 onlinegamebooks.com

Umí pracovat s předpřipravenými stavy v kontextu hry, bez možnosti vizualizace. [9]

3.1.4 Squiffy

Šikovný nástroj pro nelineární příběh, ovládá práci s kontextem hry. Bez možnosti vizualizace, zpětné čtenářské vazby. Editor příběhu zobrazuje celý

příběh najednou, což při větším počtu scén komplikuje orientaci. [10]

3.1.5 Quest

Velice rozsáhlý nástroj, který dovede pracovat s kontextem hry. Tento nástroj má největší množství pokročilých funkcí, požaduje ale také po autorovi významnou časovou investici pro pochopení celého systému. Jedná se o částečně odlišný formát narozdíl od ostatních nástrojů, kdy čtenář prochází různými místnostmi a sbírá předměty. Místnosti a předměty jsou spolu úzce spjaty a celý systém je na mapě místností založen. Desktop verze editoru dokonce obsahuje "Debugger", kdy může autor procházet příběhem a kontrolovat hodnoty kontextu hry, avšak pouze v aktuální hře. Zajímavé rozšíření je také možnost přidávání do hry vlastních skriptů v Javascriptu. To sice po autorovi požaduje znalost programování, otevírá mu to ale další možnosti. V dokumentaci rozlišují pojmy text adventure game (pro systém s místnostmi a předměty) a gamebook (zaměřený pouze na větvící příběh). Pro vytváření gamebooků pak doporučují vlastní nástroj Squiffy (viz odstavec výše). [11]

3.1.6 Twinery

Nástroj s velmi příjemným GUI. Autor gamebook vytváří v grafickém prostředí, kde je příběh rovnou vykreslen jako graf. Možnost práce s kontextem hry. Možnost exportu do více formátů. [12]

3.1.7 Ren'Py

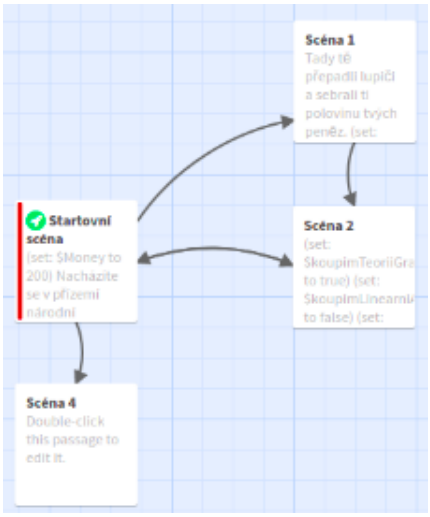
Nástroj umožňující kromě textu přidávat i grafiku a hudbu. Pokročilé funkce pracující s kontextem. Desktopová aplikace pro Windows, MacOS i Linux. Autor vyvíjí hru v několika textových souborech .rpy, ve vlastním skriptovacím jazyku, s možností základního použití skriptů v Pythonu. [13]

3.1.8 TADS

Nástroj z roku 2001. Autor gamebook vytváří ve vlastním skriptovacím jazyce připomínajícím C++ a Javascript. Pro plné využití všech funkcionalit musí být autor schopen jednoduchého programování. [14]

3.2 Twinery podrobně

Ze všech dohledaných nástrojů se právě Twinery nejvíce blíží mé představě ideálního nástroje na tvorbu interaktivních příběhů, zároveň je i jeden z nejpokročilejších. Proto jsem si na platformě Twinery vytvořil zkušební příběh a pokoušel se do něj aplikovat pokročilé funkce, které nástroj nabízí. Twinery



Obrázek 3.1: Ukázka grafu gamebooku na platformě Twinery.

Obrázek 3.2: Dolní lišta na platformě Twinery.

obsahuje velmi rozsáhlou dokumentaci. Lze používat čtyři různé syntaxe příkazů v textu, tzv. *story formats*. Protože chci s vlastním nástrojem cílit právě na jednoduchost, pro testování jsem vybral story format Harlowe, který je podle dokumentace nejsnáze pochopitelný pro začínající autory.

3.2.1 Mapa příběhu

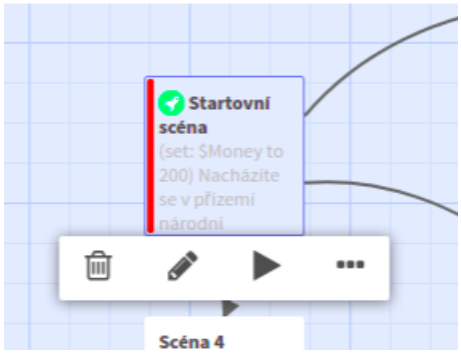
Po vytvoření gamebooku se na obrazovce objeví čisté plátno, na které lze umisťovat scény příběhu. Postupně při vývoji gamebooku tak vzniká orientovaný graf příběhu.

Scény lze přesouvat a poskládat si tak graf do podoby, která je pro autora nejpřehlednější. Přechody mezi scénami nelze vytvářet na mapě, ale až z detailu jednotlivých scén.

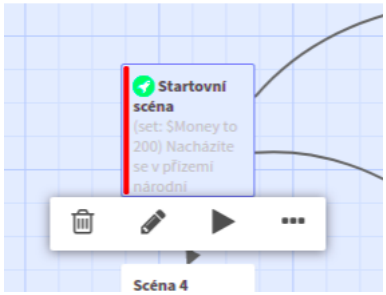
Nachází se zde dolní lišta (obrázek 3.2) pro změnu měřítka, přehrání příběhu (jako čtenář nebo v testovacím módu) a pro vytvoření nové scény (zde pojmenováno jako Místnost).

3.2.2 Úprava scény

Pokud chceme upravit scénu, je třeba umístit kurzor na čtverec se scénou, a kliknout na symbol tužky. Otevře se dialog pro úpravu scény. Z něj se bohužel



Obrázek 3.3: Detail možností úprav scény na platformě Twinery.



Obrázek 3.4: Detail možností úprav scény na platformě Twinery.

nedá na detail sousedních scén dostat pouhým kliknutím, ale je třeba dialog zavřít, najet na mapě scén na požadovaný čtverec a kliknout na symbol tužky.

Zde chci upozornit, že při vývoji gamebooku s velkým množstvím scén je právě přepínání mezi sousedními scénami to, co autor dělá nejčastěji. Na platformě Twinery je tento úkon relativně náročný na pozornost.

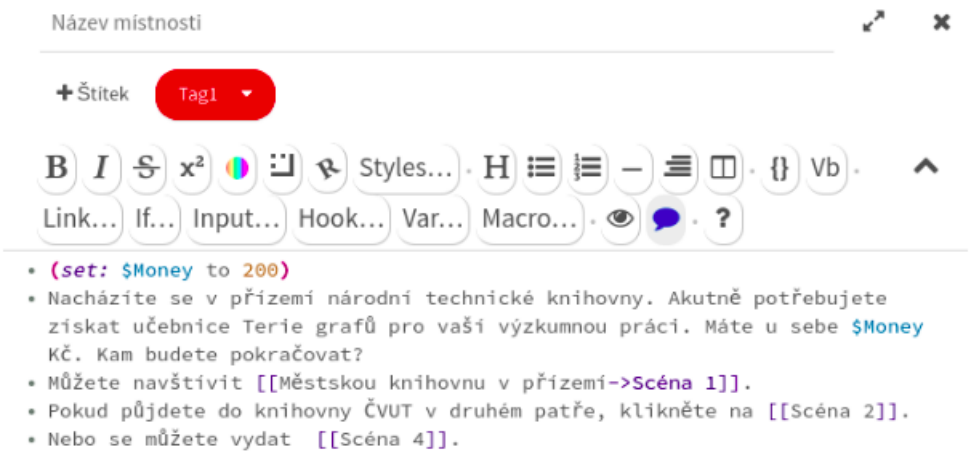
3.2.3 Příkazy v textu scény

V názvosloví Twinery jsou příkazy označovány jako makra. Po kliknutí na symbol tužky se zobrazí okno pro editaci scény. Zde lze nastavit název scény - v Twinery názvosloví *místnosti* - tagy a samotný text scény.

3.2.4 Přechod mezi scénami

```
[[Scéna 2]]
[[Městskou knihovnu v přízemí->Scéna 1]]
```

Kód 3.1: Ukázka kódu na platformě Twinery



Obrázek 3.5: Editace scény na platformě Twinery.

Do textu scény lze snadno přidat odkaz pro přechod na jinou scénu. Odkaz se může vypsat buď jako název scény nebo přepsat textem, který ideálně pasuje do věty v které se odkaz nachází.

3.2.5 Práce s proměnnými

Nastavení nebo změna proměnných se provádí v kulatých závorkách pomocí příkazu `set:` a `to`. Lze provádět operace mezi proměnnými.

```
(set: $Penize to 200)
(set: $Penize to $Penize−50)
(set: $Penize to $Penize−$CenaUcebnice)
```

Kód 3.2: Přiřazování do proměnných na platformě Twinery

3.2.6 Podmíněné úseky

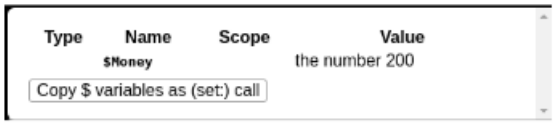
Použití podmínek demonstruji na příkladu. Čtenář prošel obchodem s učebnicemi, měl možnosti nabrat si některé do nákupního košíku. Tyto volby si pamatujeme nastavením pravdivostních proměnných.

```
(set: $koupitTeoriiGrafu to true)
```

Kód 3.3: Přiřazování do proměnných na platformě Twinery

```
• (set: $cenaNakupu to 0)
• (if: $koupitTeoriiGrafu)[(set: $cenaNakupu to $cenaNakupu+199)]
• (if: $koupitLinearniAlgebru)[(set: $cenaNakupu to $cenaNakupu+299)]
• (if: $cenaNakupu > $penize)[Nemůžete si koupit vše, co jste si naložil do
  košíku! Cena nákupu je $cenaNakupu Kč, a vy u sebe máte pouze $penize Kč!]
  (else:)[(set: $penize to $penize - $cenaNakupu)Slečna za pokladnou se na
  vás usměje a zdarma vám učebnice zabalí do dárkového papíru. Po opuštění
  obchodu vám zbylo $penize Kč.]
```

Obrázek 3.6: Ukázka práce s podmíněnými úseky na platformě Twinery.



Obrázek 3.7: Ladící prostředí na platformě Twinery.

Když se přiblíží k pokladně, text scény můžeme vidět na obrázku 3.6, přímo v Twinery editoru.

3.2.7 Ladění gamebooku

Při přehrání příběhu v Debug módu, lze vidět aktuální hodnoty proměnných.

Zároveň se při čtení příběhu vypisují chybové hlášky, pokud autor vytvořil nevalidní příkazy v textech scén. **Tato upozornění se nezobrazí při psaní v detailu scény, ani pokud se jedná o syntaktické chyby.**

3.2.8 Zveřejnění gamebooku

Na platformě Twinery nelze interaktivní příběhy publikovat a poskytnout čtenářům. Příběh je třeba exportovat do HTML formátu statické webové stránky. Na webové stránce itch.io pak lze příběh v tomto formátu nahrát a publikovat. Lze spekulovat o tom, zda-li je to pro autory a čtenáře pohodlné řešení, jasné však je, že **autoři nedostanou zpětnou vazbu v podobě analýzy čtenářských průchodů jejich gamebookem.**

3.2.9 Hodnocení

- Velmi rozsáhlá dokumentace.
- Snadné používání proměnných a podmínek.
- Přehledný a přizpůsobitelný graf scén.

- Scény lze tagovat a barevně rozlišovat.
- Možnost začít zkušebně číst příběh od kterékoli scény.
- Složité přepínání mezi scénami, sousední části textu nejsou v detailech scén propojeny.
- Odstavce v textu scény jsou v podobě odrážkového seznamu, což nemusí být vždy žádoucí.
- Scény nelze seskupovat do větších celků - kapitoly a části. Graf s několika sty scénami už ztrácí přehlednost a hodilo by se jej roztřídit do menšího počtu celků.
- Vzhledem k tomu že Twinery odkazy mezi scénami vykresluje jako odkazy, jsou zabarveny podle toho zda-li už byly navštíveny. Což dává čtenáři informaci, kterou by neměl mít.
- Nekontroluje syntaxi příkazů přímo v detailu scény, je třeba vždy přejít do přehrávání příběhu, na danou scénu, kde se vypíšou případné chyby.
- Chybové hlášky nejsou příliš výstižné. Například chybí informace o použití neexistující proměnné, což může při překlepech vyústit v dlouhotrvající hledání chyby.

3.2.10 Chybějící pokročilé funkce

Zásadním nedostatkem Twinery (stejně jako všech ostatních zmíněných nástrojů) je absence pokročilých funkcí pro vývoj gamebooku.

- Seznam nedokončených scén.
- Výpočet dosažitelného kontextu hry.
- Visualisace průchodu po zveřejnění.
- Korektura mezi autory v gamebooku, přidávání komentářů ke scénám.
- Detekce konečnosti hry. Lze chodit po kružnici stále dokola.

3.2.11 Twinery jako opensource

Protože systém práce s proměnnými a podmínkami hodnotím pozitivně, nabízí se myšlenka navázání na projekt Twinery. Je totiž k dispozici jako openSource. Bohužel obsahuje množství chyb. Stavět zmíněná rozšíření na dvanáct let starém nedoladěném projektu, kterému chybí zásadní funkcionality a který je i přes to velice rozsáhlý, hodnotím jako velice nesnadnou cestu.[15].

Hlavní překážkou zde je nemožnost sledování čtenářských průchodů gamebookem, které jsou pro pokročilé funkcionality nástroje nezbytné (Twinery totiž neposkytuje přehrávání příběhů, ale pouze export a pak případně přenos na další platformu). Jako možnost se nabízí do exportované statické HTML stránky přidat JavaScript, který bude sledovat čtenářské průchody a odesílat je na server, kde se budou data shromažďovat. Problém nastane při zobrazení dat v editoru; bylo by třeba na základě těchto dat vizuálně upravovat Twinery editor s mapou scén, aby mohl autor přehledně data o čtenářských průchodech konzumovat, což by nepochybně znamenalo zásah do jádra celého systému.

3.3 Závěr

Ani jeden ze zmíněných nástrojů autorovi neposkytne vizualizaci čtenářských průchodů gamebookem, detekci kontextových kružnic ani výpočet všech dosažitelných kontextů hry zároveň, což jsou hlavní funkcionality, kterými se můj projekt od již stávajících liší. Data o čtenářských průchodech mohou být pochopitelně pouze v systému, který obsahuje čtečku i editor. Velké množství existujících nástrojů umožňují čtení až po exportu (viz. Twinery) nebo nahrání na externí web, další překážkou je pak složitost některých systémů nebo nutnost znalosti programování.

Kapitola 4

Návrh formátu IF

Než se pustím do návrhu struktury aplikace, je třeba si ujasnit, v jakém formátu bude autor gamebook vyvíjet. Na základě průzkumu již existujících nástrojů jsem se rozhodl definovat vlastní formát, v kterém budou autoři gamebook vyvíjet. Nazvu jej **IF formát** (Interactive Fiction).

Tato zkratka dvojího významu je běžně používanou v různých pracích zabývajících se problematikou gamebooků. [16] Zároveň je používána i na mnohých webech, a to mnohdy i v souvislost s videohrami vypravějícími příběh. [17, 18]

Hlavní požadavkem formátu IF musí být **jednoduchost a snadné pochopení**. I když některé prvky formátu IF velmi připomínají práci v konvenčním programovacím jazyce (hlavně práce s proměnnými), autoři gamebooků programátoři nejsou a úkolem formátu IF je jim práci s kontextem hry co nejvíce usnadnit a přiblížit.

Protože v běžném textu nejsou používány hranaté závorky, všechny příkazy do nich uzavřeme, aby se od textu odlišily. Zároveň to poskytne editoru jednoduchou možnost okamžité nápovědy: jakmile autor napíše otevírací hranatou závorku, může okamžitě nabídnout možné příkazy, aby si je nemusel uživatel pamatovat (spolu s výstižným popisem).

```
Text scény.
[příkaz]
Další text [další příkaz] scény.
```

Kód 4.1: Návrh syntaxe formátu IF

Na druhou stranu zde máme funkční požadavky, které formát musí poskytnout.

4.1 Technické požadavky

1. **Náhodná hodnota.** Nastavení proměnné na náhodnou číselnou hodnotu z daného intervalu.
2. **Výpis proměnné do textu** (resp. její aktuální hodnoty).
3. **Podmíněný úsek.** Může obsahovat text, příkazy, kombinace obojího, další podmíněný úsek.
4. **Přechod na další scénu.** Bude obsahovat popis a scénu, na kterou lze přejít. Většinou bude umístěna na konci scény, ale může být také vypsána v podmíněném úseku.
5. **Výpis scény.** Pokud z jedné scény vede pouze jedna možnost a nechceme čtenáře nutit k stisknutí tlačítka s významem *Pokračovat*, jednoduše vytiskneme jedinou následující scénu za aktuální.
6. **Načtení vstupu od čtenáře.** Čtenář zadá textovou nebo číselnou hodnotu, která se uloží do proměnné, může také vybrat ze seznamu hodnot.
7. **Zpožděný úsek.** Stejně jako podmíněný úsek, může se daný úsek zobrazit až po uplynutí časového intervalu.

4.2 Příkazy formátu IF

Při definici budeme používat následující syntaxi:

1. `<var>` pro název proměnné
2. `<expr>` pro výraz jakéhokoli typu
3. `<list>` pro název proměnné typu list

S inspirací z dříve zmíněných nástrojů (3.2) a vědomím jasných požadavků na formát (4.1) jsem se pustil do definice jednotlivých příkazů.

Zároveň je třeba pomýšlet na to, že z těchto příkazů formátu IF bude generován javascriptový kód, jak si ukážeme v kapitole 5 o implementaci. Příkazy budou uvedeny velkými písmeny, ale bude možné je psát i malými.

4.3 Algoritmus parsování

K rozpoznání jednotlivých příkazů od sebe a příslušné vyhodnocení byly použity regulární výrazy. Ač mohou s Java implementací regulárních výrazů být problémy s rychlostí, pro účely této práce by neměly být omezující. V

testovacím příběhu, popsaném v kapitole 9, nebyl problém naparsovat scénu se zhruba tisíci znaky. [19]

Výhodou tohoto přístupu je přímočará implementace a snadná rozšiřitelnost o další příkazy nebo dokonce změna existující syntaxe.

Při parsování gamebooku zpracováváme scény z fronty. Začneme startovní scénou a vždy, když narazíme na příkaz přechodu na další scénu, přidáme si `ID scény` z příkazu do fronty. Chceme parsovat všechny přímé i nepřímé potomky startovní scény. Ostatní scény naparsujeme také, ale budeme o nich vědět, že nejsou zařazené v herním grafu a autora o tom informujeme.

4.3.1 Přiřazení do proměnné

Pokud `<expr>` obsahuje existující proměnné, převedeme je do tvaru `variables[<var>]`, kde `variables` bude proměnná frontendu mapující názvy proměnných na jejich aktuální hodnotu. Poté do `variables[<var>]` přiřadíme `<expr>`.

```
[SET <var> TO <expr>]
```

Kód 4.2: Formát IF: Přiřazení do proměnné

4.3.2 Operace se seznamy

Základní operace pro práci se seznamy.

```
[ADD <expr> TO <list>]
[REMOVE <expr> FROM <list>]
[REMOVE ALL FROM <list>] //vyprázdní seznam
```

Kód 4.3: Formát IF: Operace se seznamy

4.3.3 Výpis proměnné

Zkontrolujeme, jestli v dané scéně již proměnná kterou chceme vypsat existuje. Pokud ne, vypíšeme autorovi chybu.

```
[PRINT <var>]
```

Kód 4.4: Formát IF: Výpis proměnné

4.3.4 Podmíněný úsek

Úsek textu, pro který podmínka neplatí, se nejenže čtenáři nevypíše, ale ani se nevykonají příkazy v jeho těle. Při parsování si v případě otevření podmíněného úseku uložíme na zásobník daný podmíněný úsek. Při zavírání podmíněného úseku zkontrolujeme, že na vrcholu zásobníku se nachází podmíněný úsek (analogicky budeme postupovat při zpožděném úseku, se stejným zásobníkem).

[**IF** <expr>] *//Příkaz uvozující podmíněný úsek.*
[**ELSE**] *//Vykoná se, pokud <expr> vrátí False (volitelné).*
[**ELSE IF** <expr>] *//Větvení podle více podmínek (volitelné).*
[**END IF**] *//Ukončení podmíněného úseku.*

Kód 4.5: Formát IF: Podmíněné úseky

4.3.5 Výpis scény

Jedná se o nahrání textu jiné scény, který bude zpracován stejně jako jakýkoli jiný formát IF.

[**IMPORT SCENE** <ID scény>]

Kód 4.6: Formát IF: Výpis scény

4.3.6 Přechod na další scénu

Propojí mezi sebou aktuální scénu a scénu s `ID` z příkazu a vytvoří mezi nimi přechod se zadaným popiskem.

[**LINK TO SCENE** <ID scény> **WITH LABEL** <string>]

Kód 4.7: Formát IF: Přechod na další scénu

4.3.7 Načtení vstupu od čtenáře

Načte vstup od čtenáře se zadaným popiskem do proměnné daného typu. Typ je specifikován kvůli typové kontrole: pokud autor potřebuje od čtenáře číselnou hodnotu, nebude čtenáři v přehrávači dovoleno zadat nic jiného než

číslo a analogicky pro další typy. Prakticky řečeno, dle typu proměnné ve čtečce vykreslíme typ `input` elementu, do kterého bude uživatel hodnotu zadávat. Také lze hodnotu do proměnné vybrat z nabídky.

```
[SAVE TO <var> FROM INPUT AS REAL-NUMBER
WITH LABEL <label>]
[SAVE TO <var> FROM INPUT AS DECIMAL-NUMBER
WITH LABEL <label>]
[SAVE TO <var> FROM INPUT AS STRING
WITH LABEL <label>]
//zobrazí list s možností výběru, který uloží do dané proměnné
[SAVE TO <var> FROM SELECT <list> WITH LABEL <label>]
```

Kód 4.8: Formát IF: Načtení vstupu od čtenáře

4.3.8 Zpožděný úsek

Autor zadá zpoždění, s kterým bude úsek vyhodnocen. Stejně jako u podmíněného úseku zde pomocí zásobníku zkontrolujeme, jestli byl každý úsek zavřen právě jednou (viz 4.3.4).

```
[DELAY <expr> SECONDS]
[END DELAY]
```

Kód 4.9: Formát IF: Zpožděný úsek

4.3.9 Ukončení gamebooku

Interaktivní příběh na některých scénách musí skončit, v jazyce teorie grafů to budou listy herního grafu.

```
[END VICTORY] //vítězství, splnění zadaného úkolu
[END DEFEAT] //prohra, nedosažení cíle
```

Kód 4.10: Formát IF: Ukončení gamebooku

V každém příběhu může pochopitelně být více konců, úspěšných i špatných a mohou se od sebe lišit intenzitou úspěchu (příkladem může být příběh o vyjednávání rukojmích; čtenář jako vyjednavač může zachránit všechny/většinu/žádné rukojmí atp.). Tyto okolnosti, pro každý příběh specifické, má

možnost autor čtenáři v závěrečné scéně vysvětlit. Nicméně, musí pak být jasně řečeno, zda-li je to úspěch nebo ne. Důvodem je, aby mohla být obtížnost interaktivních příběhů vzájemně měřena, stejným měřítkem.

Ze scén s ukončujícími příkazy už pochopitelně nesmí vést žádné odchody.

4.4 Expression

Obsahem výrazu může být číslo, řetězec, definice listu nebo také některý z následujících příkazů. Tyto nejsou uzavírány do hranatých závorek, jsou součástí výrazu, který je součástí nějakého výše zmíněného příkazu.

4.4.1 Prvek náhody

Pokud autor bude chtít, aby průběh interaktivního příběhu závisel kromě důvtipu čtenáře také na náhodě, nabídneme mu možnost generování náhodného čísla.

RANDOM NUMBER FROM <number> **TO** <number>

Kód 4.11: Formát IF: Generování náhodného čísla

V kombinaci s podmínkami a dalšími příkazy lze vytvořit zcela náhodné situace ve hře.

4.4.2 Obsahuje list daný prvek?

Výraz vrací pravdivostní hodnotu `true` nebo `false`.

LIST <list> **CONTAINS** <expr>

Kód 4.12: Formát IF: Obsahuje list daný prvek?

4.4.3 Co obsahuje list na dané pozici?

Výraz nakonci musí být vyhodnocen na celočíselnou kladnou hodnotu v rozsahu aktuální velikosti pole, jinak příkaz skončí chybou. Nevýhodou tohoto příkazu je, že chybě lze těžko předejít již při vývoji a může se proto při neopatrném zacházení objevit při čtení příběhu.

```
GET ITEM FROM LIST <list> AT POSITION <expr>
```

Kód 4.13: Formát IF: Co obsahuje list na dané pozici?

4.5 Příklad transpilace formátu IF do JS

Ukažme si předešlou teorii této kapitoly na příkladu testovací úvodní scény gamebooku, který by se mohl nazývat například *Obchodník s jablky*. Popíšeme, z čeho se scéna skládá a jak bude transpilována do javascriptu.

Nejprve ukažme **formát IF**:

```
[SET n_apples TO 0]
[SET money TO 300]
[SET apple_price TO 12]

Jste obchodníkem na ovocném trhu, prodáváte jablka.

[SAVE TO name FROM INPUT AS STRING
WITH LABEL Zadejte své jméno]

Aktuální cena jablek je [PRINT apple_price] Kč, z předchozích obchodů
máte našetřeno [PRINT money] Kč.

Kolik jablek si na zítřejší trh koupíte?

[SAVE TO buy_apples FROM INPUT AS DECIMAL-NUMBER
WITH LABEL Počet jablek]

[LINK TO SCENE 1 WITH LABEL Pokračujme na trh!]
```

Kód 4.14: Příklad transpilace: formát IF

Krátce objasněme význam proměnných a co se s nimi děje:

- `n_apples` představuje počet jablek, která obchodník vlastní, na začátku hry žádné (řádek 1).
- `money` aktuální počet peněz našeho obchodníka (řádek 2).
- `apple_price` aktuální cena jablek, která se může během hry měnit (řádek 3).

- `name` Jméno obchodníka, které si hráč libovolně zvolí (řádek 7).
- `buy_apples` Počet jablek, která si chce obchodník koupit, tuto hodnotu musí čtenář zadat (řádek 14).

Text neuzavřený do `[hranatých závorek]` se vypíše jako normální text (řádky 5, 9, 10, 12). Na řádcích 9 a 10 zároveň čtenáře o aktuální ceně jablek a výši jeho peněz informujeme. Příkaz na poslední řádce vytvoří možnost přechodu na scénu 1 s popiskem `Pokračujme na trh!`.

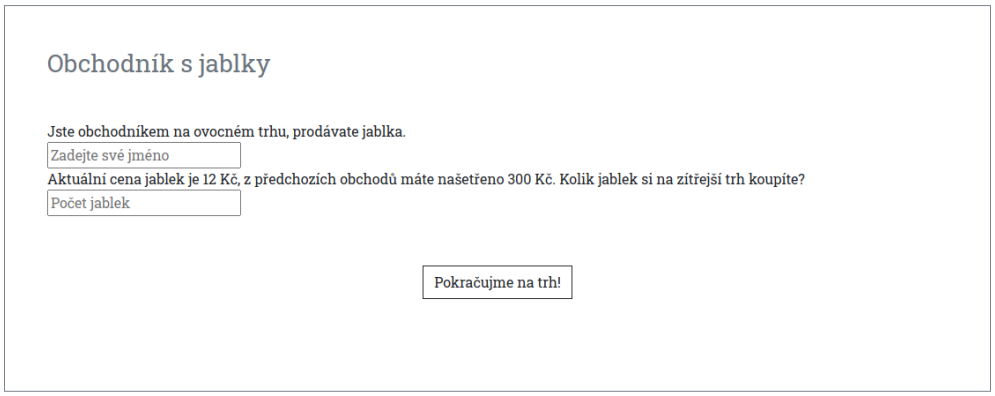
Teď si ukažme JS kód, který aplikace vygeneruje.

```
let ___tmp_idx;

variables["n_apples"] = 0;
variables["money"] = 300;
variables["apple_price"] = 12;

print("\n \nJste obchodníkem na ovocném trhu, prodávate jablka. \n");

addInput("name", "STRING", "Zadejte své jméno");

print("\nAktuální cena jablek je ");
print(variables["apple_price"]);
print(" Kč, z předchozích obchodů máte našetřeno ");
print(variables["money"]);
print(" Kč.\nKolik jablek si na zítřejší trh koupíte?\n");

addInput("buy_apples", "DECIMAL−NUMBER", "Počet jablek");

print("\n");

addLink(1, "Pokračujme na trh!");

setIsRendered(true);
```

Kód 4.15: Příklad transpilace: javascript

Co přesně kód dělá?

Protože je třeba v některých příkazech hledat v seznamu hodnot, na prvním řádku každé scény se pro tuto situaci vytvoří proměnná `__tmp_idx`, do které uložíme index nalezeného prvku (nebo -1 v případě nenalezení, JS funkce `indexOf`). Na dalších třech řádcích nastavujeme hodnoty proměnných do již dříve zmíněné mapy `variables`. Řádek 7 jednoduše vypíše text.



Obrázek 4.1: Čtecí prostředí ukázkového gamebooku *Obchodník s jablky*.

Na řádku 9 voláme JS funkci `addInput` s třemi argumenty, která je implementována na frontendu a doplní do textu patřičné vstupní pole a aktivuje listenery, které při odchodu ze scény vstupní hodnotu uloží do patřičných proměnných.

Řádky 11-15 kombinují výpis textu a hodnot definovaných proměnných. Za zmínku pak stojí funkce `addLink`, která vygeneruje funkční tlačítko pro přechod na další scénu. Úplně poslední řádek informuje prostředí frontendu, že již došlo k vyrenderování celé scény a není třeba jej už opakovat. Jedná se přímo o volání setteru `useState` hooku v knihovně React.

Jak se tato scéna nakonec vykreslí uživateli do čtecího prostředí můžeme vidět na obrázku 4.1.

Důležitá poznámka. Proces transpilace formátu IF přímo do JS zmíněný v této kapitole byl původním návrhem, ale během analýzy se ukázalo, že je třeba formát IF naparsovat do objektů, s kterými bude analýza schopná pracovat. Proces parsování formátu IF by se zbytečně opakoval. Z toho důvodu jsem v kapitole 7.2 popsal rozdělení parsování a transpilace, tak, aby naparsovaný formát IF mohl být použitý i pro analýzu. Na výsledku operace popisované v této kapitole se nic nezmění, pouze bude rozdělena do dvou logicky oddělených částí.

Kapitola 5

Vlastní nástroj

V předchozích kapitolách jsem nastínil, co musí vlastní nástroj na vývoj a analýzu gamebooků obsahovat, aby byl praktický a použitelný, a také aby mělo vůbec smysl jej vyvíjet, když už množství konkurenčních nástrojů existuje.

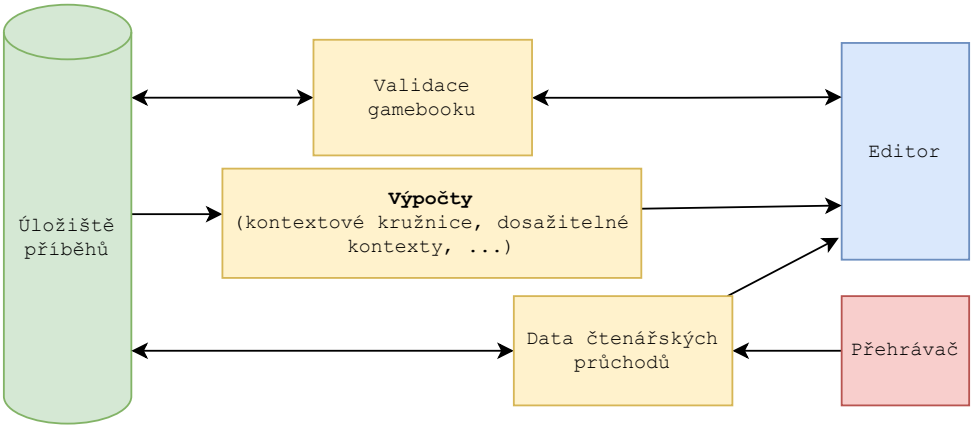
V této kapitole navážu návrhem aplikace.

Zásadní věcí, kterou se vlastní nástroj od již existujících bude lišit, jsou dříve zmíněné pokročilé funkcionality (3.2.9). Před prací na nich je ale třeba mít funkční jádro systému, které bude schopné gamebook uložit, upravovat a prezentovat čtenáři.

5.1 Struktura aplikace

Již máme definován formát IF, základní stavební kámen vlastního nástroje. V této části textu se zaměřím na návrh struktury celé aplikace. Jaké jsou na fungování aplikace požadavky?

1. **Editor.** Při vytváření příběhu je třeba poskytnout autorům přívětivé prostředí, v kterém lze gamebook vyvíjet.
   a. Protože cílíme na autory neprogramátory, nesmí chybět **živá nápověda** k jednotlivým příkazům a **validace správnosti syntaxe formátu IF**, případně **výpis chyb**.
   b. Během vývoje se autorovi zobrazí výpočty a informace o vývoji gamebooku (kontextové kružnice, dosažitelných kontextů, jak bylo zmíněno v sekci 2.2).
   c. Po zveřejnění gamebooku a prvních čteních bude umět prostředí editoru zobrazit **statistiky chování čtenářů v interaktivním příběhu**.



Obrázek 5.1: Schéma vlastního nástroje.

2. **Přehrávač.** Prostředí pro čtenáře musí zvládat zobrazit a smysluplně zpracovat všechny příkazy formátu IF. Kromě toho musí zaznamenávat čtenářovy akce, aby mohly být později analyzovány a zobrazeny autorovi.

5.2 Volba technologií

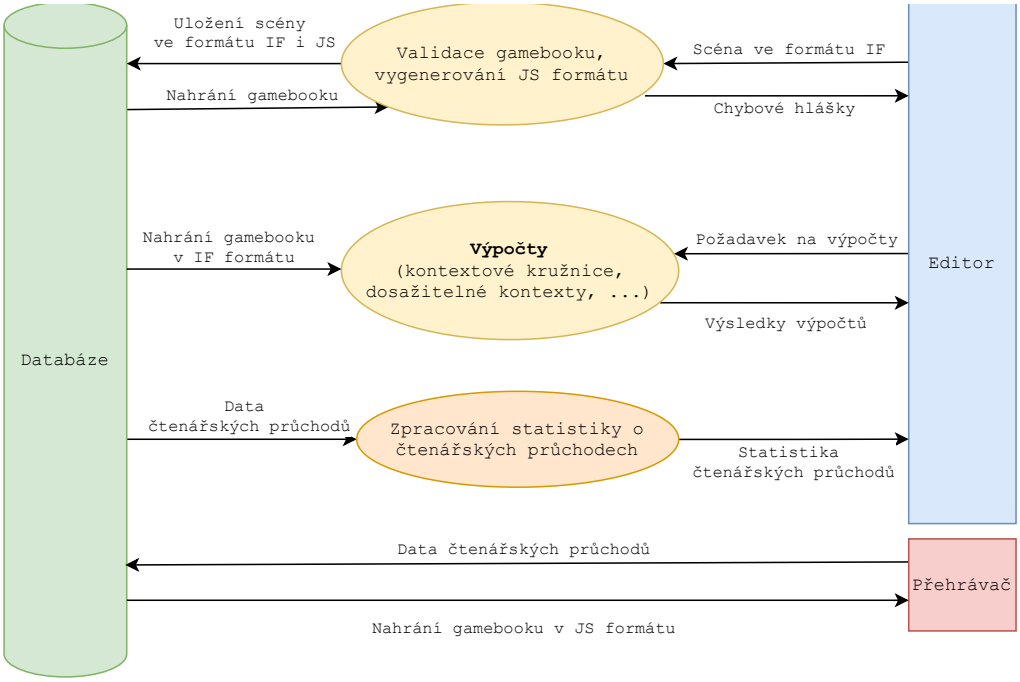
Více uživatelů, autorů i čtenářů, bude přistupovat ke stejným datům. Z tohoto důvodu se mi zdá jasnou volbou webová aplikace s daty uloženými v databázi. Pokud bych se uchýlil k desktopovému řešení, příběhová data by se musela exportovat a přeposílat mezi jednotlivými počítači ručně, což je značně nekomfortní a nešikovné řešení.

V následující části textu konkrétně nastíním způsob implementace.

5.2.1 Editor příběhu

Jak bude editor s interaktivním příběhem pracovat? Jediným rozumným řešením se mi zdá dělit text po scénách. Pokud by měl být zobrazitelný příběh celý najednou, v případě několika desítek nebo stovek scén by byla přehlednost nulová - to už bychom mohli gamebook psát do jednoho textového souboru.

Před uložením díla je třeba zvalidovat autorův text, jestli jsou příkazy v textu syntakticky správně nebo jestli gamebook neobsahuje kontextové kružnice (2.2). Pokud je s gamebookem něco v nepořádku, autor se o tom musí dozvědět v podobě přehledné chybové hlášky. Zároveň se musí pravidelně přepočítávat hodnoty dosažitelných kontextů a nedokončené scény (2.2). Editor



Obrázek 5.2: Struktura aplikace.

bude posílat pouze právě editovanou scénu, ostatní jsou totiž již uložené. Scéna se na backendu zvaliduje a pokud je v pořádku, uloží se do databáze (nákres 5.2).

5.3 Přehrávač příběhu

Před tím, než se zamyslím nad samotným návrhem přehrávače příběhu a jeho fungováním, je třeba si říct, jak se příběh vytvořený ve formátu IF vůbec spustí na frontendu, kde potřebujeme dynamické chování přímo v prohlížeči.

Přímo formát IF nemůže být spustitelný v prohlížeci bez jakékoli úpravy nebo konverze, protože by to znamenalo, že půjde o skripty v Javascriptu, které se sice jednoduše spustí, ale chtěli bychom po autorovi, aby se vyznal v programování, což je proti řečeným požadavkům (2). Prioritou je, aby formát IF byl pro autory jednoduchý a intuitivní.

Původní myšlenka byla taková, že bych formát IF převedl do objektového modelu, kde by každá funkcionalita a komponenta (text, příkazy v textu) tvořila vlastní objekt s vlastnostmi, které potřebuje pro přehrání. Tato složitá objektová struktura by se pak ve formátu JSON odeslala do frontendu, který by s ní dále pracoval. Jenže jak pracoval? Znamenalo by to složité rozbalování a parsování objektu do spustitelného formátu. Pravděpodobně by to bylo proveditelné, ale zbytečně komplikované. Výhodou tohoto přístupu se zdálo,

že po nahrání textu do objektů bude snadno zanalyzovatelný; pro výpočty dosažitelných kontextů nebo kontextových kružnic (2).

Jak již bylo řečeno na konci kapitoly 4, v první části práce jsem se přiklonil k řešení, které vypadalo snadně a elegantně. **Formát IF se přímo převede na javascriptové příkazy.** Text se převede na funkci tisknoucí text, příkaz nastavující proměnnou se převede do javascriptové syntaxe přiřazení hodnoty do proměnné a analogicky pro další příkazy. Podrobně jednotlivé příkazy a jejích konverzi do JS formátu rozeberu v následující kapitole (4.2). Problém nastal při implementaci analýzy, kdy bych musel parsování v celém procesu provádět dvakrát; nejdříve pro transpilaci do JS a pak pro analýzu. Proto jsem se v závěru přiklonil k řešení podobnému původní myšlence, které je popsáno v sekci 7.2.

Gamebook původně uložený ve formátu IF tedy převedu do javascriptových příkazů, které se následně odešlou do přehrávače, na frontend, kde budou spuštěny. Aktuální kontext hry na frontendu bude udržován v proměnné typu `Map`, kde bude jméno proměnné mapováno na její aktuální hodnotu.

Protože nechceme generování formátu JS zbytečně opakovat vždy při požadavku na čtení gamebooku, znamená to, že musíme do databáze uložit gamebook v obou formátech; IF i JS. Jedná se o částečnou duplicitu dat. Pokud bychom se jí chtěli vyhnout a uložit pouze formát JS, který spouštějí klienti-čtenáři, ztratili bychom možnost gamebook dále upravovat a zobrazovat ve formátu IF. Nebo pokud bychom uložili pouze formát IF, prodloužíme dobu odezvy při požadavku na čtení gamebooku.

Protože při přepínání mezi scénami nechceme, aby čtenář čekal, ideálním řešením je hned na začátku odeslat celý gamebook, neposílat jej po scénách. V prostředí frontendu se bude zároveň držet hodnota kontextu hry, s kterou se bude pracovat a která bude průběžně asynchronně odesílána na backend, aby se mohly sbírat statistiky o čtenářských průchodech.

Schéma aplikace je zobrazeno na obrázku 5.2.

5.4 Volba webového frameworku

Které webové technologie zvolit? Potřebujeme data ukládat do databáze, umět je zpracovat na backendu a odeslat na frontend. V něm musíme umět spustit javascriptové příkazy, udržovat kontext hry a odesílat na backend data o čtenářských průchodech, které se uloží do databáze, jak bylo řečeno dříve (5.3).

To jsou velmi obecné požadavky, které splňuje snad každá rozumná webová technologie. Vzhledem k tomu jsem si vybral řešení, které se zdá v dnešní době nejperspektivnější, práce s ním mě nejvíc obohatí a nabyté znalosti mi budou užitečné v budoucnu.

Po zralé úvaze jsem zvolil kombinaci Java Spring a javascriptové knihovy React. Java Spring podporuje velká vývojářská komunita, proto lze k tomuto frameworku najít mnoho materiálů. React umožňuje rychlé změny DOM struktury frontendu a staví na komponentové architektuře, což je zajímavý a užitečný koncept, s kterým jsem se doposud nesetkal. [20, 21, 22, 23]

Konvenční způsob uložení dat ve webových aplikacích psaných v Javě představuje namapování objektů do databázových tabulek, například pomocí Java Persistence API (JPA). Tuto technologii je možné využít i v Spring Boot aplikaci. [24, 25]

Po krátké práci s Reactem, používajíc JavaScript, jsem dospěl k rozhodnutí, že bude pro dlouhodobější vývoj aplikace přínosnější ji přepsat do TypeScriptu. TypeScript narozdíl od JavaScriptu pomáhá v práci s datovými typy, z toho důvodu lépe odchytí většinu chyb a nutí programátora psát kód čitelněji [26]. Celá práce tedy bude vyvíjená v TypeScriptu.

Ke stylování frontendu použiji *Bootstrap*, rozšířenou HTML, CSS a JS knihovnu a její implementaci *React Bootstrap* přizpůsobenou pro systém komponent, na kterém React staví. [27, 28]

Kapitola 6

Webová aplikace

V předchozích kapitolách jsme řešili základní fungování aplikace, jak bude interaktivní příběh tvořen a prezentován čtenáři. V této kapitole se posuneme dále, pokusíme se kostru aplikace zasadit do kontextu webové aplikace, analyzujeme potřebné uživatelské role, jejich oprávnění a implementaci celé správy uživatelů.

V první řadě se podíváme na požadavky, které na aplikaci a uživatelský systém máme. Z nich pak navrhneme uživatelské role a další potřebné atributy.

6.1 Požadavky

Co musí systém uživatelských rolí a celá webová aplikace poskytovat?

1. Uživatel musí být vždy přihlášen. Pokud interaktivní příběh tvoří, aby mohl být identifikován jako autor. Pokud příběhy čte, aby mohl být sledován jeho pohyb herním grafem pro tvorbu statistik. Nepřihlášení autoři budou mít možnosti prohlížet si seznam interaktivních příběhů.
2. Každý uživatel si může vytvořit vlastní gamebook.
3. Každý gamebook má právě jednoho autora. Kterýkoli čtenář může gamebook číst (ve zveřejněném stavu).
4. Čtenáři by ke čtení měli být připuštěni až ve chvíli, kdy je příběh hotov. Definujeme tedy různé stavy, ve kterých se může gamebook nacházet; minimálně *vytvářeno* a *zveřejněno*.
5. Interaktivní příběh by měl před zveřejněním projít schvalovacím procesem, aby nemohl kdokoli zveřejnit cokoli. Důvodem pro nezveřejnění by mohla být nemístná vulgarita, nesmyslný obsah atp. Nabízí se tedy

schvalovací stav gamebooku. Jemu pak musí být přiřazen uživatel s vyšší rolí, který má právo gamebook schválit. Nazvěme ho *redaktorem*.

6. Aby vůbec mohl být gamebook schválen, musí se dostat do stavu, kdy ho kromě autora uvidí také administrátoři, kteří ze svých řad určí jednoho, kdo se stane pro daný gamebook redaktorem. Nabízí se tedy další stav *výběr redaktora*.

7. Role čtenář/autor/redaktor se váže vždy ke konkrétnímu gamebooku. Tedy čtenáři jsou pro každý gamebook všichni uživatelé.

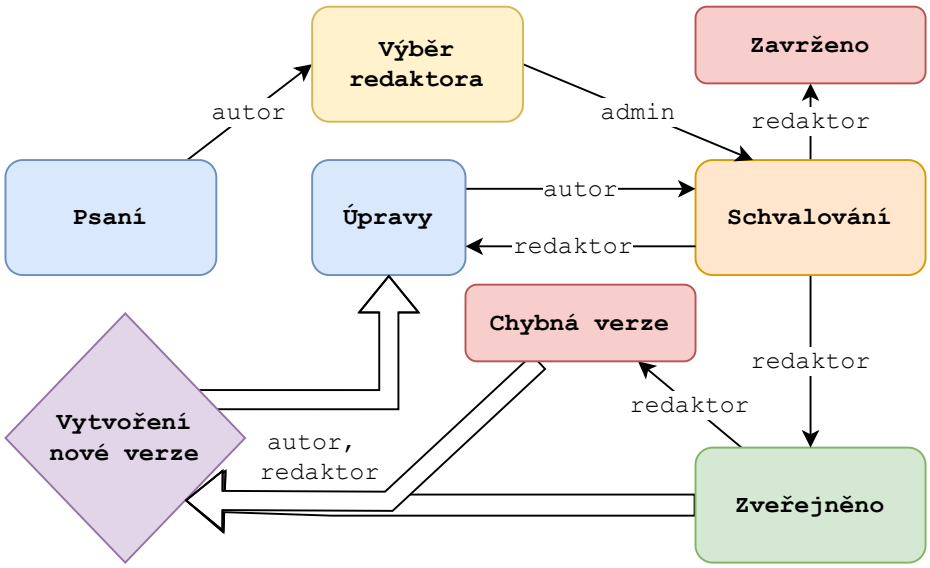
8. Ožehavou otázkou je editace gamebooku po zveřejnění. Pokud bychom dovolili měnit herní graf, těžko bychom mohli udržet statistiky o průchodech před a po změně. Ani změna textu scén se mi nejeví jako důvodná; také by mohla ovlivnit statistiky a čtenáři, kteří by měněnou scénu četli opakovaně, by byli akorát zmatení. Jediným důvodem pro změnu je chyba v gamebooku, která nebyla odhalena během testování ani aplikací. Ve fázi *schvalování* autor a redaktor odchytí většinu nepřesností a zveřejní gamebook jako hotový a odladěný, nikoli jeho testovací verzi, která se bude dále měnit. Původně jsem se přikláněl k přístupu, že změny ve zveřejněném gamebooku mohou být provedeny pouze v případě chyby a měly by být minimální, aby data o čtenářských průchodech neztratila význam pro statistiku. Ztratila by se tím ale možnost upravit gamebook na základě čtenářských statistik a nakonec by to mohlo vést i k problémům s chybami nebo rozhození statistických údajů (i přes to, že bychom se tomu chtěli vyhnout). Řešení těchto problémů by mohlo být ve verzování jednotlivých gamebooků; zveřejněný gamebook by byl již neměnný a pokud by autor nebo redaktor potřebovali provést změny, upravili by novou verzi gamebooku. Podrobně se o řešení zmiňuje v sekci 6.3.

9. Pokud čtenář příběh nedohraje do konce, budeme chtít, aby mohl pokračovat v místě, kde skončil. Pozici hráče v příběhu můžeme vyjádřit identifikátorem scény a hodnotou jeho kontextu hry. To jsou stejné informace, které budeme ukládat pro herní statistiky. Implementaci těchto dvou funkcionalit tedy provedu dohromady, viz kapitolu 8.

6.2 Stavy gamebooku

Poznámka: stavy gamebooku jsem naimplementoval z časových důvodů tím způsobem, že jejich stav může měnit autor v nastavení gamebooku. Analýza v této sekci je přípravou pro budoucí verzi.

Popišme si jednotlivé stavy, které může gamebook nabývat a přechody mezi nimi. V tabulce vidíme kdo má která práva v každém stavu.



Obrázek 6.1: Stavy gamebooku a přecházení mezi nimi.

	Právo editace	Právo čtení
Psaní	autor	
Výběr redaktora	*nikdo*	autor, administrátoři
Úpravy	autor	autor, redaktor
Schvalování	redaktor	autor, redaktor
Zveřejněno	*nikdo*	všichni uživatelé
Chybná verze	*nikdo*	autor, redaktor, administrátoři
Zavrženo	autor	

Tabulka 6.1: Stavy gamebooku

Graf 6.1 znázorňuje, jak gamebook může stavy měnit a kdo změnu iniciuje. Ze stavů **Zveřejněno** a **Chybná verze** lze vytvořit novou verzi gamebooku, která se ocitne ve stavu **Úpravy**. Původní ale zůstane beze změny.
Jednotlivé stavy si popíšeme.

6.2.1 Psaní

Výchozí stav gamebooku, autor píše příběh. Ve chvíli, kdy je s ním spokojen, změní stav na **Výběr redaktora**.

6.2.2 Výběr redaktora

V tomto stavu již autor nesmí gamebook upravovat. Některý z administrátorů příběhu přiřadí redaktora a tím posune gamebook do stavu **Schvalování**. Redaktorem může být libovolný zodpovědný uživatel, to záleží na administrátorově volbě.

6.2.3 Schvalování

Vybraný redaktor projde příběh. Pokud je spokojen, změní stav gamebooku na **Zveřejněno**. V opačném případě sepíše autorovi své poznatky a vrátí mu příběh k upravení, změnou stavu na **Úpravy**. Redaktor má v tomto stavu přístup k editoru gamebooku ze dvou důvodů; potřebuje vidět kromě výsledného textu také příkazy formátu IF, aby mohl pochopit fungování hry. Také pokud narazí na překlep, který autor přehlédl, aby ho mohl rovnou upravit. V zásadě by ale neměl toto právo využívat k přepisování příběhu.
Pokud je dodaný příběh absolutně nepřijatelný, redaktor má možnost nastavit stav na **Zavrženo**, čímž se gamebook vyřadí z dalšího schvalovacího procesu.

6.2.4 Úpravy

Autor zapracovává poznámky od redaktora, debatují o nich. Oba mohou gamebook číst, možnost upravovat má ale pouze autor. Ve chvíli, kdy je s výsledkem spokojen, nastaví stav na **Schvalování** a předá tak gamebook k opětovnému posouzení redaktorovi.
Do tohoto stavu se také dostane nová verze, zkopírovaná ze zveřejněného gamebooku.

6.2.5 Zveřejněno

Všichni přihlášení uživatelé mohou gamebook číst, editace není možná (důvody byly zmíněny výše). Pokud v gamebooku dojde k chybě nebo bude chtít autor na základě čtenářských statistik příběh upravit, má on nebo redaktor možnost vytvořit novou verzi a s ní dále pracovat. Viz 6.3. Pokud je verze chybná a už ji nadále nechceme zobrazovat uživatelům (například z toho důvodu, že jich byla vytvořena nová, funkční verze), redaktor ji může přesunout do stavu **Chybná verze**.

6.2.6 Zavrženo

Gamebook v tomto stavu již není možné dostat zpět do schvalovacího procesu, nikdy nebude zveřejněn. Nicméně autor k němu má nadále přístup.

6.2.7 Chybná verze

Vyřazené nefunkční nebo nekvalitní verze gamebooku, které již nahradily novější verze. Je možné z ní stále vytvořit novou verzi, ale tato již zůstane neměnná.

6.3 Verzování gamebooků

Abychom umožnili přehledně upravovat chyby v gamebooku a vylepšovat příběh na základě čtenářských statistik, zavedeme možnost práce s různými verzemi gamebooku. Ze stavu **Zveřejněno** bude mít autor nebo redaktor možnost vytvořit novou verzi. Jak budou různé verze uloženy, jak s nimi budeme pracovat?
Z předchozích kapitol máme funkční systém pro tvorbu a čtení jednotlivých gamebooků, tvorba nové verze bude tedy přesná kopie gamebooku, s kterou

budeme pracovat nezávisle na původní verzi. Na druhou stranu pro přehlednost ze strany čtenářů nebudeme různé verze gamebooků zobrazovat jako odlišné gamebooky, ale čtenáři budou mít možnost vybrat si, kterou verzi spustit (z těch, které jsou ve stavu **Zveřejněno**). Objekty gamebooku představující další verze v sobě tedy ponesou referenci na původní verzi gamebooku, aby bylo možné je sdružovat. Po vybrání konkrétní verze čtenářem spustíme čtečku s danou verzí, kde už s příběhem budeme pracovat jak známe z předchozích kapitol.
V nové verzi gamebooku už bude vybrán redaktor, shodný s původní verzí a gamebook se dostane do stavu **Úpravy**, v kterém může autor příběh na základě statistik vylepšovat. Dále budeme pokračovat shodně podle diagramu 6.1.

6.4 Uživatelské role

Z popisů stavů gamebooku a schvalovacího procesu jasně vyplývá, že v zásadě existují pouze dvě základní uživatelské role; uživatel a administrátor. Role *čtenář, autor, redaktor* se vážou přímo ke konkrétnímu gamebooku. V následující kapitole musíme tedy počítat s dvojím typem rolí;

1. **Základní role.** Uživatel, administrátor.
2. **Role vázající se ke gamebooku.** Čtenář, autor, redaktor.

Každý uživatel má základní roli a s každým existujícím gamebookem nějakou roli vázající se k němu.
V této sekci jsme si představili pravidla, podle kterých by webová aplikace s více autory i čtenáři mohla fungovat, jaké požadavky by to kladlo na administrátory. V rámci testování na závěr bakalářské práce bude schvalovací proces gamebooku vynechán, ale pro reálné použití s více autory je nezbytnou součástí aplikace.

6.5 Spring Security

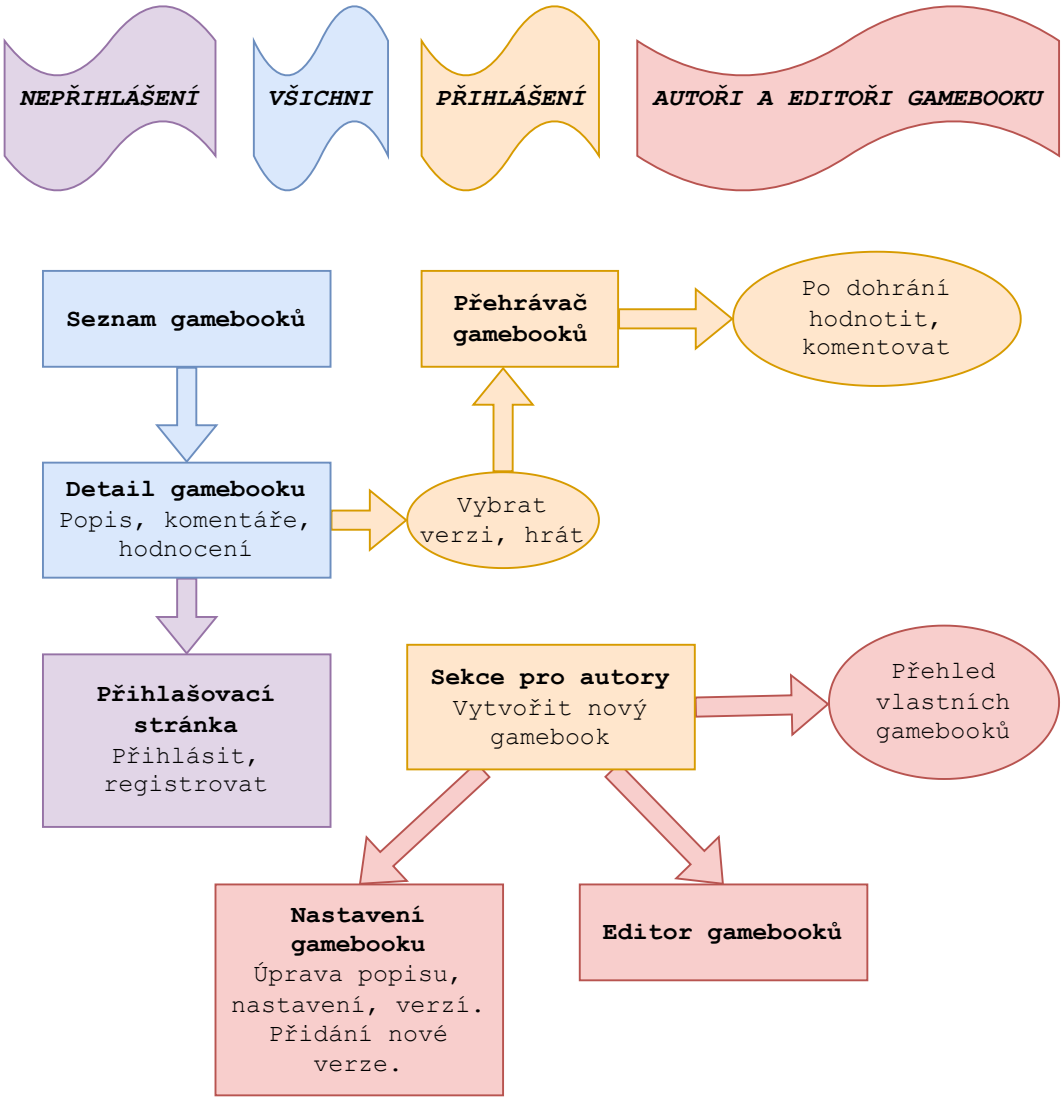
Vzhledem k volbě technologie Java Spring, se přirozeně nabízí používat již zabudované bezpečnostní mechanismy a použít framework Spring Security. Tento framework nabízí řešení autorizace i autentifikace uživatelů. Lze jej snadno přizpůsobit konkrétnímu řešení.
Vytvořil jsem si v databázi vlastní entitu uživatele namísto zabudovaného objektu `UserPrincipal`, abych mohl uživatele přidávat, odebírat, přiřazovat jako autory nebo redaktory ke gamebookům a spravovat uživatelské role, i s ohledem na budoucí rozšíření. Nějakou dobu jsem zápolil se správným nastavením Spring Security, připojením k mé databázi s uživateli a správným odesíláním požadavků z frontendu, nakonec jsem ale v rozsáhlé dokumentaci našel všechny potřebné informace k zprovoznění celého mechanismu od registrace přes přihlášení uživatele a následně ověřování jeho rolí i rolí vzhledem ke gamebookům. [29, 30, 31, 32, 33]

6.6 Frontend

Na závěr se podívejme, jaké možnosti uživatelům poskytuje uživatelské prostředí. Hlavními komponentami jsou stále editor a přehrávač gamebooků, přibyla k nim ale řada dalších stránek; přehledy příběhů a jejich verzí pro čtenáře, autory a editory, možnosti nastavení gamebooků a jejich verzí nebo přihlašovací stránka. Podrobně viz. diagram struktury webové aplikace 6.2.

6.7 Spojení backendu a frontendu

Komunikace mezi Spring aplikací běžící na backendu a React frontendem bude řešená konvenčně pomocí REST API rozhraní. Implementaci endpointů Spring framework nativně lehce umožňuje, stejně tak volání API z typescriptu na frontendu není problémem. Spring framework přijaté tělo požadavku naparsuje do Data Transfer Objectu, s kterým se dá v javě snadno manipulovat.



Obrázek 6.2: Struktura webové aplikace. Přístup na jednotlivé stránky a akce je znázorněn pomocí čtyř barev pro různé skupiny uživatelů.

Kapitola 7

Pokročilá analýza gamebooku

V této části textu se budu zabývat návrhem algoritmů pro analýzu interaktivního příběhu, představené v sekci 2.2.

7.1 Cíle pokročilé analýzy

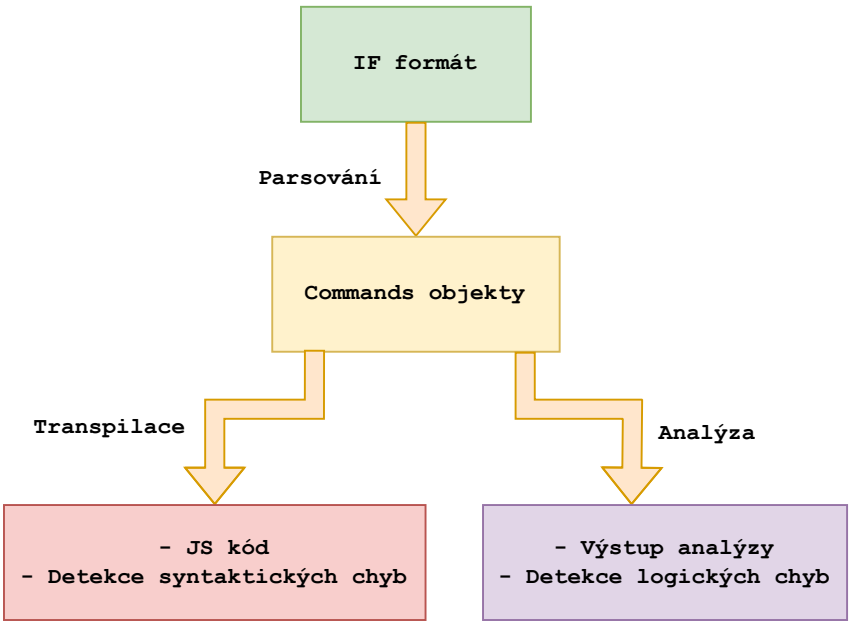
Herní graf je souvislý orientovaný graf s jedním počátečním uzlem (uzel bez vstupních hran) a nejméně jedním koncovým uzlem (uzel bez výstupních hran). V kterékoli části grafu může být definován nový stav kontextu hry, s výchozí hodnotou, v každém dalším uzlu může být měněn nebo používán. Tyto akce nemusí být vázány přímo ke scéně, ale pouze ke konkrétnímu bloku ve scéně (například bloku podmínky, nebo zpožděného bloku). Při analýze gamebooku bude třeba procházet příkazy IF ve scénách a vyhodnocovat je. Abychom pokaždé znovu nezpracovávali text, což může být časově náročné, pro potřeby algoritmu vypustíme z IF formátu prostý text (příkazy `PRINT` zanecháme, abychom mohli například zkontrolovat, zda-li byla používaná proměnná definována).
Zopakujme, že cílem analýzy gamebooku je poskytnout autorovi pomocné informace k vývoji příběhu a najít chybné konstrukce v herním grafu, tedy *zkontrolovat* po něm jeho práci. Shrňme si, co bude výsledkem algoritmu, který budeme navrhovat.

1. **Dosažitelné kontexty hry.** Spočítat pro každou scénu, jakých hodnot v ní může nabývat každý stav. Pro číselné stavy minimální a maximální hodnotu, pro ostatní stavy všechny možné hodnoty.

2. **Konečnost hry.** Graf gamebooku nesmí obsahovat kružnice, všechny listy grafu musí být scény s příkazem `[END VICTORY/DEFEAT]` (což pochopitelně nebude během vývoje splněno, viz další bod).

3. **Nedokončené scény.** Detekce zjevně rozpracovaných scén; těch, které nejsou konečné a nevedou z nich žádné další scény.

4. **Kontextově slepé scény.** Detekce kontextově slepých scén. Scénu nazveme kontextově slepou, pokud není konečná, vedou z ní odchody ale může nastat situace, že z ní za dosažitelného kontextu hry nelze odejít do další scény (a čtenář by v ní tak mohl uvíznout).



Obrázek 7.1: Spojení společné části parsování pro transpilaci a analýzu gamebooku.

5. **Nedosažitelné scény.** Detekce scén, do kterých nevede žádný příchod. Takové scény ale mají smysl pokud jsou do jiné scény importované pomocí příkazu `IMPORT SCENE`.

6. **Kontextově nedosažitelné scény.** Detekce scén, do kterých sice příchody vedou, ale za dosažitelných kontextů jimi nikdy nelze do scény přijít a jsou tak k ničemu.

7. **Nedefinované stavy.** V žádné scéně nelze používat hodnotu stavu, která zatím nebyla definována. Není třeba všechny stavy definovat ve startovní scéně, ale není možné pracovat se stavem, u něhož není jisté, že v některých předchozích scénách nebyla jeho hodnota definována.

Budeme rozlišovat chyby, se kterým gamebook není funkční a upozornění na možné problémy.

7.2 Transpilace a analýza

Po chvíli uvažování nad implementací algoritmu pro pokročilou analýzu gamebooků jsem si všiml zásadní podobnosti s algoritmem transpilace do javascriptu (4.5), který už jsem měl v tu chvíli naimplementován. V obou úlohách procházíme texty scén a pomocí regulárních výrazu detekujeme jednotlivé příkazy. Po detekci příkazů přichází v případě transpilace na řadu převod na javascriptový kód, v případě analýzy na potřebné výpočty. Proto jsem se rozhodl reimplementovat algoritmus transpilace tak, že oddělím parsování textu (tedy detekci příkazů) od překladu do javascriptového kódu,

abych mohl stejné parsování textu použít i pro analýzu gamebooku. Schéma je znázorněno na obrázku 7.1.
Během parsování textu budeme rozparsované informace ukládat do objektové struktury příkazů. Pro každý příkaz definujeme třídu, která bude dědit od `AbstractCommand` definující dvě hlavní funkce: `transpile` a `analyse`. V třídách příkazů bude naimplementována odpovídající logika pro transpilaci a analýzu.

7.2.1 Ukázka parsování do objektové struktury

Ukážeme si, jak se text scény v IF formátu naparsuje do struktury `Command` objektů na příkladu:

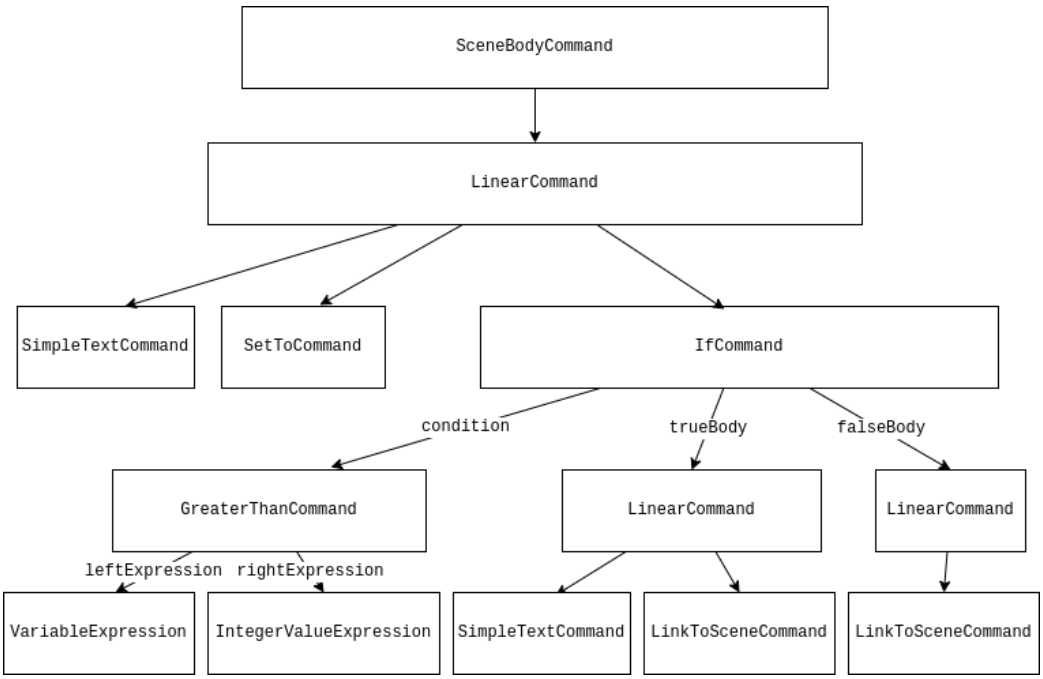
```
Vyjdeš ze sklepa a přivoláš si výtah, který tě odveze k recepci.

[SET lzeNavstivitLokaci TO lzeNavstivitLokaci − 1]

[IF lzeNavstivitLokaci < 1.0]
Už je příliš pozdě.
[LINK TO SCENE 8
WITH LABEL Vrať se na policejní stanici a uzavři případ.]
[ELSE]
[LINK TO SCENE 17 WITH LABEL Opustíš nemocnici. ]
[END IF]
```

Kód 7.1: Příkad scény z testovacího gamebooku.

Tato krátká scéna se naparsuje do objektové struktury zobrazené na 7.2.

7.3 Analýza pro jednotlivé příkazy

Před započetím analýzy musí být gamebook úspěšně naparsován do `Commands` objektů, nad kterými budeme analýzu vykonávat. Tyto objekty tvoří stromové struktury, jednu pro každou scénu. Při analýze jednotlivých scén udržujeme a průběžně upravujeme a přidáváme dosažitelné kontexty a proměnné v nich podle příkazů, což bude podrobně popsáno v následujících podkapitolách.
Kdybychom v gamebooku zakázali kružnice, seřadili bychom scény do topologického uspořádání a v něm je postupně analyzovali, abychom měli jistotu, že všichni předchůdci právě analyzované scény již budou zpracováni. Jenže jak bylo řečeno v sekci 2.2, povolujeme kružnice, které **po průchodu mění kontext hry**, proto nelze topologického pořadí vždy dosáhnout. Proto algoritmus upravíme. Po zanalyzování scény `s` se podíváme na její potomky, a pokud už jsme některého z nich analyzovali, budeme ho analyzovat znovu s novým kontextem ze scény `s`. Pokud některou ze scén budeme analyzovat příliš mnohokrát (toto číslo je nastaveno na `100`), analýzu utneme a upozorníme autora, že daná scéna je pravděpodobně částí nekonečné kružnice. Do



Obrázek 7.2: Ukázka Command objektů.

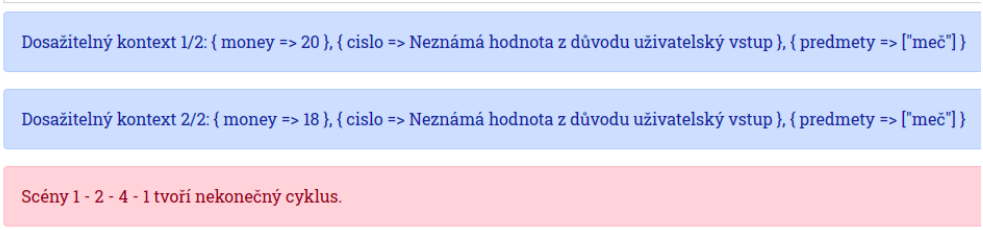
budoucna bych chtěl tuto magickou konstantu odstranit a detekovat kontextovou kružnici ve chvíli, kdy se **po jednom průchodu nezmění kontext hry**.
Dosažitelné kontexty ze všech předchůdců sjednotíme, vytvoříme nový stav analýzy, s kterým spustíme pro danou scénu analýzu.
V následujících podsekcích si projdeme způsob analýzy jednotlivých příkazů a výrazů.

7.3.1 Analýza výrazů

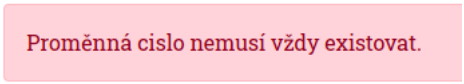
Po úspěšném parsování předpokládáme syntakticky správný výraz. V analýze budeme kontrolovat především existenci všech proměnných, ze kterých čteme. Výraz je vnitřně reprezentován obdobným způsobem jako příkazy v `Commands` objektech; tvoří strom do sebe zanořených objektů, které pro každý typ výrazu zajišťují jeho logiku.
Většina příkazů v sobě obsahuje výraz nebo výrazy, které musí být logicky validní, aby byl logicky validní i celý příkaz.
Stávající implementace si poradí se základními aritmetickými a logickými operacemi, ale neumí pracovat se složitějšími uzávorkovanými výrazy. Což lze vyřešit pomocí Shunting-Yard algoritmu, který nebyl implementován z časových důvodů. [34]

7.3.2 Přiřazení výrazu do proměnné

1. `expression` musí být logicky validní.



Obrázek 7.3: Ukázka výsledků analýzy v editoru s nekonečným cyklem a hodnotami dosažitelných kontextů.



Obrázek 7.4: Ukázka výsledků analýzy v editoru, pokud autor pracuje s proměnnou, kterou buď není definována vůbec, nebo pouze v části kontextů a během čtení tak může dojít k závažné chybě.

2. Pokud `variable` již existuje, pro všechny možné aktuální hodnoty `variable` vyhodnotit `expression` a uložit do možných hodnot (to pro případ, že by se v `expression` daná `variable` nacházela a jednalo by se o relativní změnu). Pokud neexistuje, pouze vyhodnotit `expression`.

[**SET** expression **TO** variable]

Kód 7.2: Analýza: Přiřazení výrazu do proměnné

7.3.3 Přidání výrazu do seznamu

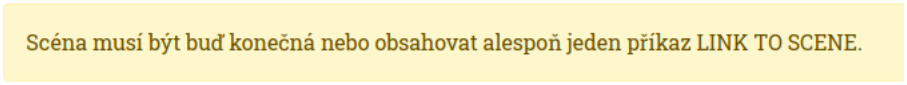
1. `expression` musí být logicky validní.
2. Seznam již musí být definován (pro inicializaci seznamu s výchozími hodnotami používáme příkaz `SET TO`).

[**ADD** expression **TO** list]

Kód 7.3: Analýza: Přidání výrazu do seznamu

7.3.4 Odebrání prvku ze seznamu

1. `expression` musí být logicky validní nebo se rovnat `ALL`.
2. Seznam již musí být definován.



Obrázek 7.5: Ukázka výsledků analýzy v editoru, pokud v IF formátu scény není nalezen END ani LINK TO SCENE příkaz.

3. Pokud prvek v nějakých kontextech nemusí existovat, vypíšeme varování.

[**REMOVE** expression **FROM** list]
[**REMOVE ALL FROM** list]

Kód 7.4: Analýza: Odebrání prvku ze seznamu

7.3.5 Výpis výrazu

1. Vypisovaný `expression` musí být logicky validní.

7.3.6 Podmíněný úsek

[**IF** expression] – [**END IF**]
[**IF** expression] – [**ELSE**] – [**END IF**]
[**IF** expression] – [**ELSE IF** innerExpression] – (vnořená podmínka)
– [**END IF**]

Kód 7.5: Analýza: Podmíněné úseky

1. `expression` musí být logicky validní.
2. Pokud pro všechny dosažitelné kontexty je `expression` pouze `true` nebo pouze `false`, podmínka nemá smysl, vypíšeme autorovi varování.
3. Dojde zde k rozdělení výpočtu. Do `true` části podmínky poputují ty hodnoty kontextu, které odpovídají splněné podmínce, obdobně do `false` části, pokud existuje. Obě pak pokračují analýzou příkazů *za* podmínkou.

7.3.7 Přechod na další scénu

Po té, co je tento příkaz správně naparsován, není již co analyzovat. Objekt představující tento příkaz v sobě udržuje stav, s kterým byl příkaz zavolán (pokud byl například volán v podmínce atp.), aby se scéna, na kterou tento přechod vede, analyzovala právě s tím stavem a nikoliv s tím, který bude nakonci vyhodnocení scény.

Obrázek 7.6: Ukázka výsledků analýzy v editoru, pokud autor nechá IF formát scény prázdný.

7.3.8 Načtení vstupu od čtenáře a náhodné hodnoty

Jelikož vstup od uživatele nebo náhodné hodnoty mají velké množství možností, v mnoha případech jsou dokonce nekonečné, nemůžeme pro všechny případy provádět analýzu. V prvním kroku analýzy proto *uznáme*, že hodnoty proměnných, do kterých se uživatelské vstupy nebo náhodné hodnoty uloží, neznáme.
V dalším kroku analýzy, který nebyl zatím implementován, bychom mohli s těmito neznámými hodnotami začít pracovat až ve chvíli, kdy budeme znát podmínky, v kterých se používají. Pokud například uživatel zadá množství jablek od 0 do 100, která si koupí, ale v celém příběhu bude záležet pouze na tom, jestli si koupil méně nebo více než 10, máme ze sta možností rázem dvě relevantní, s kterými potřebujeme dále počítat. Z obecného hlediska není ale tento přístup zdaleka tak jednoduchý jako na příkladu, hlavně pokud se bude neznámá proměnná nacházet v podmínce s dalšími proměnnými.
Z praktického hlediska pro velké množství případů autorovi postačí, když mu o *neznámé* hodnotě proměnné vypíšeme všechny informace, které prozatím máme; tedy odkud byla načtena, jakého je typu, případně rozsahu. Například: `Hodnota z čtenářského vstupu typu textový řetězec.`

7.3.9 Zpožděný úsek

Jelikož nevíme, zda-li bude čtenáři zpožděný úsek vyhodnocen nebo ne, musíme zvážit obě možnosti. Analyzujeme případ, kdy se tělo zpožděného úseku vykoná a pak výsledky analýzy spojíme s případem, kdy se nevykonalo.

7.3.10 Ukončení gamebooku

1. Za příkazem `END` již nesmí být další příkazy.

7.3.11 Pravidla na úrovni scény

1. Scéna musí obsahovat nejméně jeden příkaz nebo text.
2. Pokud je scéna konečná, nesmí obsahovat v žádném bloku `LINKED TO SCENE` příkazy.
3. Scéna musí být buď konečná nebo obsahovat nejméně jeden `LINKED TO SCENE` příkaz.

7.4 Analýza na úrovni gamebooku

Cíle analýzy na úrovni grafu interaktivního příběhu byly popsány v úvodu této kapitoly. Krátce popišme poznatky z implementace a dosažené výsledky.

1. **Dosažitelné kontexty hry.** Pro proměnné nastavené a upravované příkazy `SET`, `ADD, REMOVE` pro manipulaci s listy nebo uživatelským vstupem typu výběr (příkaz `SAVE TO FROM SELECT`) dovedeme v každé scéně přesně určit možné hodnoty a jejich kombinace. Při analyzování výběru ze seznamu počítáme se všemi možnými hodnotami (protože jich z podstaty bude "rozumné"množství). Problém nastane v případě, kdy nastavujeme proměnnou na náhodnou hodnotu příkazem `RANDOM` nebo obyčejným uživatelským vstupem (příkaz `SAVE TO FROM INPUT ...`). Detaily viz podsekci o načítání vstupů a náhodných hodnotách. 7.3.8

2. **Konečnost hry.** Pokud v grafu gamebooku budou nalezeny kružnice, pro všechny scény v kružnici vypíšeme chybovou hlášku `Scény 1 - 7 - 2 - 1 tvoří nekonečný cyklus`.

3. **Nedokončené scény.** Definujeme je jako scény bez konce a zároveň bez přechodů na další scény. V seznamu scén k nim vykreslíme ikonu kladiva jako informaci pro autora, že je třeba na této části textu ještě zapracovat.

4. **Kontextově slepé scény.** Implementace jde ještě o krok dále, detekuje jakýkoli nedosažitelný blok IF formátu (například v podmínce, která je vždy nepravdivá), což zahrnuje i příkazy `LINK TO SCENE`.

5. **Nedosažitelné scény.** Detekce scén, do kterých nevede žádný příchod. Příkaz `IMPORT`, pro nějž by scéna bez příchodů (která by se do jiné scény naimportovala) měla smysl, nebyl z časových důvodů naimplementován.

6. **Kontextově nedosažitelné scény.** Taková scéna nebude existovat, pokud autor odstraní všechny nedosažitelné bloky kódu. Implicitně je tedy zahrnuta v předchozích pravidlech.

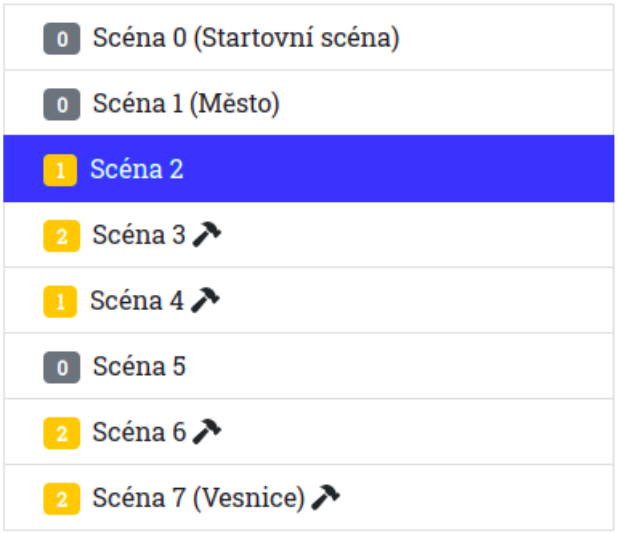
7. **Nedefinované stavy.** Práce s existujícími proměnnými je řešena na úrovni analýzy jednotlivých příkazů.

7.5 Debugger ve čtečce

Další ladící nástroj, který autorovi poskytneme, bude zobrazení aktuálních hodnot proměnných ve čtečce (*debugger*). Obdobný mechanismus pro ladění příběhu jsem při počáteční rešerši nástrojů nalezl i v systémech Quest a Twinery. [12, 11]
V budoucnu lze snadnou úpravou debugger rozšířit i pro editaci proměnných v konkrétním místě gamebooku. Ukázka debuggeru viz obrázek 7.8.

7.6 Visualizace interaktivního příběhu

Jak již bylo mnohokrát řečeno, gamebook je tvořen herním grafem. Pro usnadnění orientace v něm je žádoucí autorovi graf zobrazit. K tomu jsem použil nástroj *State Machine Cat*, který se mi jevil nejvýhodnější kombinací inverstované práce do generování grafu a přehlednosti výsledku. Narozdíl od



Obrázek 7.7: Seznam scén v editoru. Nedokončené scény jsou označeny ikonou kladiva. V závorkách lze vidět pracovní názvy scén, které se zobrazí jen autorovi.

velké části ostatních nástrojů pro vykreslení grafu nepotřebuje JSON nebo XML formát, ale prostý vlastní formát SMCAT, který lze velmi jednoduše vygenerovat (ukázka níže). [35]
Graf vykreslený pod editorem umí zobrazit:

1. Scény jako uzly a přechody jako orientované hrany mezi nimi.
2. Začátek scény je označen modrým kolečkem, konce zeleným nebo červeným podle toho jednáli si o úspěšné dohrání či prohru.
3. Aktuálně editovanou scénu modře zvýrazní.

Graf interaktivního příběhu na obrázku 7.9 je vykreslen pomocí následujícího SMCAT kódu:

```
initial [color="blue"],
final2 [color="red"],
"1 (Město)" [color="blue" active];

initial => "0 (Startovní scéna)";
"0 (Startovní scéna)" => "1 (Město)";
"1 (Město)" => "4";
"1 (Město)" => "2";
"4" => "7 (Vesnice)";
"2" => final2;
```

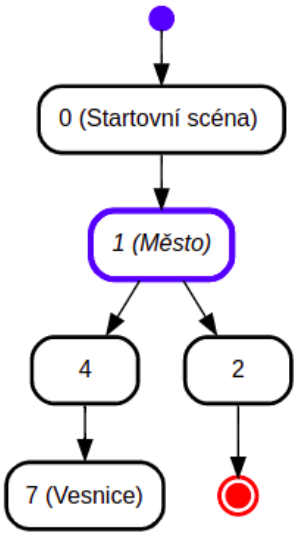
Kód 7.6: Ukázka SMCAT kódu z kterého se generuje graf statistik

Debugger (5)

Název proměnné	Hodnota
ovoce	hruška,jablko,pomeranč
vybraneOvoce	jablko
pocetOvoce	4
cenaZaKus	7
penize	43

Obrázek 7.8: Ukázka debuggeru v přehrávači interaktivního příběhu.

V první části popíšeme uzly grafu, které chceme zvýraznit, v druhé části definujeme hrany grafu. Formát SMCAT je z mého pohledu velmi přímočaře generovatelný a flexibilně se s ním pracuje. Další vykreslování grafů, barvení jeho částí a popisování budeme potřebovat v části vyhodnocování čtenářských statistik.



Obrázek 7.9: Ukázka grafu interaktivního příběhu vygenerovaného pomocí nástroje State Machine Cat.

Kapitola 8

Statistiky čtenářských dat

Poslední částí aplikace, kterou bylo třeba implementovat, je sběr dat o čtenářských průchodech interaktivním příběhem, jejich zpracování a zobrazení autorovi. Připomenu, že tyto funkcionality jsou pro tuto práci klíčové, protože jsem během rešerše nenarazil na žádný systém nebo aplikaci, která by něco podobného do takové hloubky poskytovala.
Spolu s těmito daty nás bude zajímat přímá zpětná vazba od čtenáře v podobně slovního hodnocení (komentáře) a hodnocení na číselné škále.

8.1 Datový model pro uložení statistik

Datový model aplikace jsem rozšířil o 3 datové třídy:

1. `ReadPath` představuje jednu *cestu* interaktivním příběhem. Tato cesta nemusí být dokončená. Skládá se z `ReadEntry`'s.
2. `ReadEntry` jsou části cesty interaktivním příběhem `ReadPath`, představující vždy spojení mezi scénami, tj. hranu v grafu gamebooku.
3. `Rating` představuje hodnocení, číselné nebo i slovní.

8.2 Sběr informací o čtenářských průchodech

8.2.1 Průchod gamebookem

Při nové hře (čtení gamebooku od začátku) se vytvoří `ReadPath` s informacemi:

1. vazba na čtenáře
2. vazba na gamebook
3. pořadové číslo průchodu daného čtenáře v daném gamebooku
4. datum a čas vytvoření (pro vypsání seznamu hraných her čtenáři i autorovi)
5. stav (výhra/prohra/nedokončeno)
6. vazbu na všechny odpovídající `ReadEntry`

Statistiky

Čtenáři

Počet čtenářů, kteří ...	
... nedohráli ani jednou	6
... dohráli právě jednou	8
... dohráli právě dvakrát	2
... dohráli alespoň třikrát	4
Celkový počet čtenářů	**20**

Výhry

Počet čtenářů, kteří alespoň jednou dočetli a ...	
... výhry nikdy nedosáhli	7
... dosáhli výhry poprvé při prvním průchodu	2
... dosáhli výhry poprvé při druhém průchodu	3
... dosáhli výhry nejdříve při třetím průchodu	8

Průchody jednoho čtenáře

Počet čtenářů, kteří ...	
Průměrný počet průchodů na jednoho čtenáře	3
Nejvyšší počet průchodů na jednoho čtenáře	7
Průměrné číslo průchodu, v kterém čtenáři, kteří alespoň jednou vyhráli, dosáhli výhry	1

Obrázek 8.1: Ukázka obecných statistik testovacího příběhu po uživatelském testování, viz kapitola 10

8.2.2 Záznam o přechodu

Při každém kliknutí na přechod ve čtečce se na backend aplikace odešle požadavek, který uloží nový `ReadEntry`:

1. vazbu na odpovídající `ReadPath`
2. vazba na obě scény, mezi kterými se přechází; `ReadEntry` představuje v grafu gamebooku hranu
3. hodnoty proměnných (a to včetně právě přečtených inputů)
4. datum a čas uložení (při hraní rozehrané hry budeme hledat nejnovější záznam)

Tento záznam se uloží do databáze a bude použit pro statistiku a jako uložení hry.

8.2.3 Hodnocení a komentáře

Po dohrání gamebooku bude mít čtenář možnost hodnotit gamebook od 0 do 5, případně napsat komentář. Bude to možné vždy po dohrání gamebooku. Do objektu `Rating` se uloží:

1. vazba na čtenáře
2. vazba na gamebook
3. hodnocení od 0 do 5
4. komentář (volitelně)
5. `isSpoiler`; pokud čtenář nastaví na true, komentář se zobrazí pouze těm čtenářům, kteří mají úspěšně dohráno

Hodnocení a komentáře se zobrazí v detailu gamebooku.

8.3 Zpracování a zobrazení dat

Statistiky rozdělíme do tří částí, podle toho, kde je budeme zobrazovat a k čemu se logicky váží.

1. Obecné statistiky verze gamebooku, počty průchodů, informace o dosažení úspěšných konců.
2. Statistiky průchodů vazáné k verzi gamebooku zobrazené v grafu.
3. Statistiky počtu využití přechodů mezi scénami.

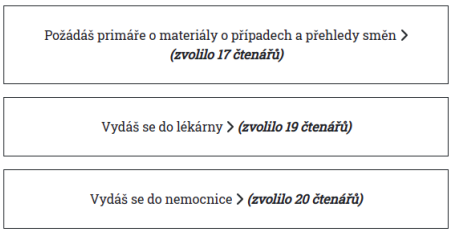
8.3.1 Obecné statistiky

Obecné statistiky představují shrnutí počtu čtenářů, jejich průchodů a výher, jsou vyjádřené vždy jedním číselným údajem:

- celkový počet čtenářů (kteří začali číst)
- počet čtenářů, kteří gamebook nedohráli ani jednou
- počet čtenářů, kteří právě jednou/dvakrát/třikrát a více dočetli
- průměrný počet průchodů na jednoho čtenáře
- nejvyšší počet průchodů na jednoho čtenáře
- počet všech průchodů všech čtenářů
- průměrný počet dokončených průchodů na jednoho čtenáře
- počet čtenářů, kteří alespoň jednou dočetli, ale výhry nikdy nedosáhli
- počet čtenářů, kteří dosáhli výhry poprvé při prvním/druhém/třetím a dalším průchodu
- průměrné číslo průchodu, v kterém čtenáři, kteří alespoň jednou vyhráli, dosáhli výhry

Jak vypadají tato data vykreslená ve frontendu lze vidět na obrázku 8.1.

Obrázek 8.2: Tyto statistiky se zobrazí pod čtečkou pouze autorovi, ukázka na gamebooku po uživatelském testování, viz kapitola 10



Obrázek 8.3: Autor může ve čtečce u přechodů vidět četnost jejich použití čtenáři, ukázka na gamebooku po uživatelském testování, viz kapitola 10

8.3.2 Grafové statistiky

Pro každou verzi gamebooku zobrazíme 4 různé grafy, zabarvené podle toho, jak byly jednotlivé uzly grafu navštěvovány pro tyto případy:

- souhrn všech průchodů
- pouze první průchody (tento graf je zvlášť důležitý, protože je zde vidět přirozené chování čtenářů bez znalosti příběhu a některých chybných konců)
- pouze druhé/třetí a další průchody

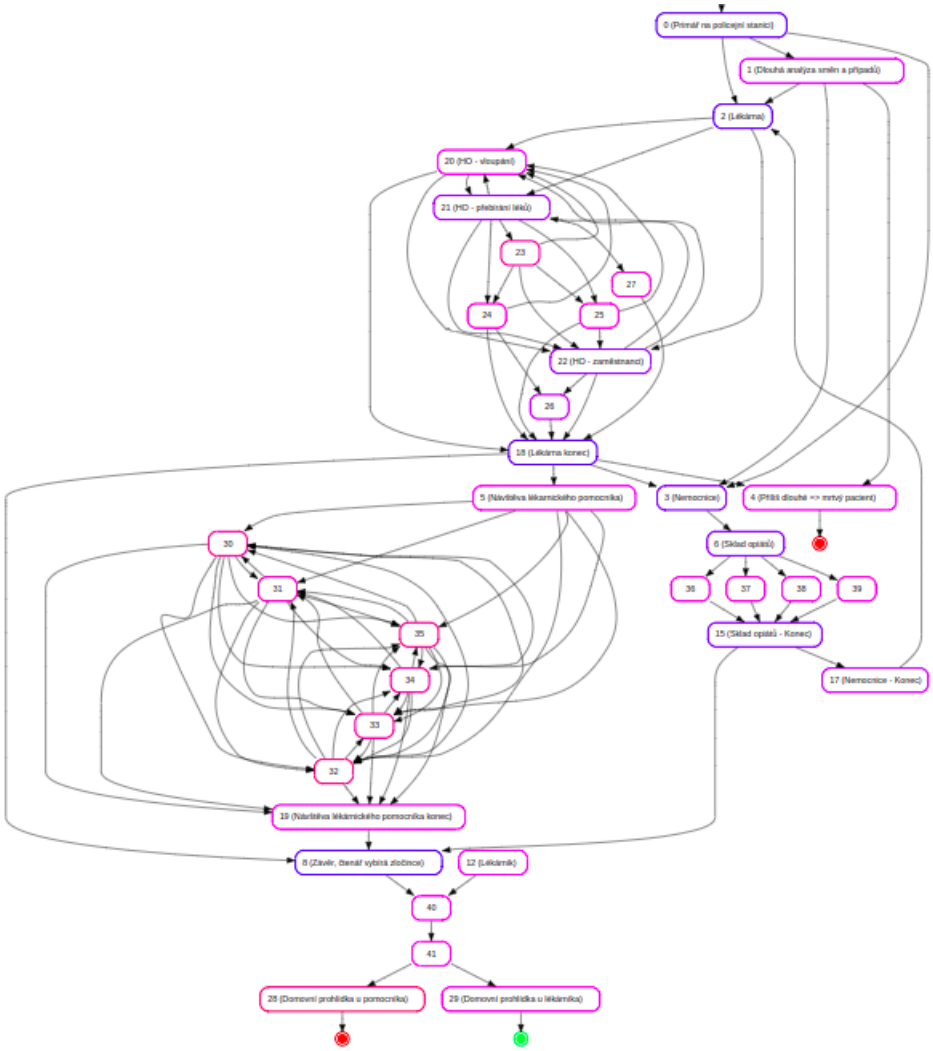
Grafové statistiky po testování příběhu jsou zobrazené na obrázku 8.4.

8.3.3 Statistiky přechodů

Tyto statistiky nevypíšeme k celému gamebooku, ale do čtečky přímo k možným volbám, aby autor viděl, kudy se z jednotlivých scén čtenáři vydávali a v jakých počtech.
Zároveň ke každé scéně vypíšeme:

- kolik čtenářů scénu celkem navštívilo
- kolik čtenářů přestalo na této scéně hrát (tento údaj může autora upozornit na zásadní nesrozumitelnost či jinou nesrovnalost, pokud na jedné scéně přestane hrát velké množství čtenářů).

Ukázky lze vidět na obrázcích 8.3 a 8.2.



Obrázek 8.4: Ukázka grafových průchodů gamebookem od červené (málo navštívené scény) po modrou (hustě navštívené) po uživatelském testování, viz kapitola 10

Kapitola 9

Testování editoru

Pro demonstraci možností vytvořeného systému jsem se rozhodl vytvořit ukázku interaktivního příběhu. Na tvorbě si mohu otestovat v praxi funkčnost editoru gamebooků a zároveň mohu příběh nechat přečíst několik uživatelů v rámci testování. Inspiroval jsem se hrou Českého rozhlasu a vytvořil jsem detektivní interaktivní příběh, v kterém je čtenářovým úkolem chytit zločince. Gamebook má celkem 40 scén. Příběh jsem nazval *Nemocnice Na Hůrce*.
V testovacím příběhu jsem používal různé složitější i jednodušší konstrukce formátu IF. Například jsem počítal, kolik lokací čtenář navštívil a toto číslo uchovával v proměnné a pokud vyčerpal maximální počet navštívení lokace, do další již nemohl (celkem bylo ze tří lokací možno navštívit dvě).

9.1 Vlastní hodnocení editoru

Poprvé jsem se přenesl z role vývojáře systému do uživatelské role autora a měl jsem možnost zhodnotit přívětivost a použitelnost editoru a především jeho hlavním funkcionalit na nichž stojí celá tato práce. Řadu z nich jsem ocenil a uvědomil jsem si množství dalších nutných pomůcek a úprav, které jsou třeba pro usnadnění autorovy práce provést. Zároveň jsem zjistil, že orientace v grafu gamebooku není pro autora skutečně jednoduchá, protože musí narozdíl od obyčejné knihy domýšlet důsledky množství různých variant příběhu a jejich kombinací, uvažovat nad jejich propojováním. Vypočtené dosažitelné kontexty mi přitom sloužily jako kontrola, zda-li jsem neudělal chybu, což jsem shledal užitečným. Na druhou stranu pokud počet kontextů začne růst, jednotlivé zprávy se stanou značně nepřehlednými. Bylo by třeba zobrazení hodnot v tabulce s vyhledáváním, nebo minimálně možnost zprávy o dosažitelných kontextem filtrovat a slučovat podle hodnot jednotlivých proměnných.

9.1.1 Návrhy na vylepšení

1. **Krátký přehled příkazů.** I přes to, že jsem formát IF vymýšlel a implementoval jsem si přesnou syntaxi všech příkazů nepamatoval, proto jsem do přílohy A sepsal "formát IF cheatsheet", který jsem při psaní používal.

2. **Příchody scén.** Jakmile počet scén přerostl 15 až 20, začal jsem ztrácet

kontext příběhu, který předcházel právě otevřené scéně a podle vykresleného grafu jsem předchozí scény hledal. Ke každé scéně jsem přidal zprávy o příchodech a jejich popisech.

3. **Nedokončené scény.** V analýze gamebooku sice identifikujeme, zdali scéna obsahuje přechody nebo konce a pokud ano, označíme ji jako dokončenou. Při vývoji gamebooku dává ale smysl nejdříve načrtnout kostru příběhu bez textu, která splňuje podmínky analýzy pro dokončenou scénu, ale chybí v ní text. Proto jsem do editoru implementoval jednoduchou pomůcku, kdy má autor možnost do textu scény přidat řádek `/TODO`, který symbol kladiva ke scéně vykreslí a autor si tak může poznámenávat scény, kam chce text dopsat nebo ho přepsat. Snadno lze tento systém rozšířit o další ikony s různým smyslem, které by mohl autor scénám přidělovat (například scény k revizi) nebo oddělit význam analyticky dokončené scény a skutečně dokončené scény dle mínění autora.

4. **Kapitoly.** Protože s přibývajícím počtem scén graf začíná být více a více nepřehledný, příběh by se dal členit do větších celků, říkejme jim kapitoly. Autor by si pak mohl zobrazit graf kapitol, z čehož by získal představu o kostře příběhu. A při psaní jednotlivých scén by se mu mohl zobrazovat graf pouze aktuální kapitoly.

9.1.2 Nalezené chyby

1. **Pomalá zpětná vazba editoru.** Z důvodu nesprávně naimplementované správy editoru v Reactu docházelo k zasekávání uživatelského prostředí, protože se po každém stisknutí klávesy v editoru překreslovala část stromu React komponent. Přeimplementoval jsem proto obsluhu editoru z řešení přes `useState` hook na `useRef`, jehož změna nevyvolá překreslení stromu komponent.

2. **Kružnice a kontextové kružnice.** Z původní analýzy vyplynula představa, že v interaktivním příběhu zakážeme kružnice, jejichž detekce je poměrně jednoduchá. Což se při vytváření příběhu ukázalo jako velmi těžko překonatelné omezení, ve chvíli, kdy sice nechceme čtenáři dovolit chodit v kruhu, nicméně vzájemné přechody mezi scénami kružnici tvoří. Bylo tedy třeba změnit posloupnost analýzy scén z topologického na poněkud složitější. V prvním průchodu graf gamebooku, i přes to že tvoří kružnice, seřadíme algoritmem pro získání topologického uspořádání, dostaneme tedy *pseudotopologické* uspořádání. Pokud narazíme na hranu vedoucí do již zanalyzovaných scén, přidáme ji mezi scény, které je třeba analyzovat a zároveň uložíme aktuální kontext hry. Ve výsledku průchod grafem vypadá tak, že kontextové kružnice *rozbalíme* duplikováním scén do stromu, který jsme schopni zanalyzovat. Protože povolujeme pouze takové kontextové kružnice, které nejsou nekonečné, pokud algoritmus navštíví stejnou scénu příliš mnohokrát, vypíšeme autorovi hlášku, že s vysokou pravděpodobností vytvořil nekonečnou kružnici. Toto číslo je prozatím nastaveno na hodnotu `100`, v budoucnu by se jeho hodnota

dala relativně určovat podle počtu scén, případně by jej mohl nastavovat autor. V této chvíli je ale algoritmus svižný a objeví kružnice v interaktivním příběhu, z praktického hlediska je tedy plně funkční.

Hlavním rozdílem mezi původní implementací je rozdělení analýzy do více kroků, z nichž první se vykonává vícekrát, jak bylo zmíněno výše a při detekci kružnic pracujeme s kontextem hry.

9.1.3 Upravená analýza

Z důvodů zmíněných v předchozí sekci byla analýza rozdělena do tří kroků. V prvních dvou částech vždy příkazy volají metody pro odpovídající částí analýzy svých potomků.

První část analýzy

Spouští se pro každou scénu vícekrát, pokud gamebook obsahuje kružnice. Upravují se zde kontexty a vyhodnocují podmínky, podle příkazů v jednotlivých scénách. Ve chvílí volání příkazů `LINK TO SCENE` ukládáme aktuální kontexty, které v následujících krocích analýzy používáme.

Druhá část analýzy

Pro každou scénu se zavolá pouze jednou. Vytvoří zprávy o všech dosažitelných kontextech, které byly vypočítány v předchozí části. Pro podmínky zjišťujeme, zda-li mohou být v dosažitelných kontextech vyhodnoceny jako pravdivé i nepravdivé, v opačném případě vypíšeme upozornění o stále (ne)platné podmínce.

Třetí část analýzy

Skládá se ze dvou částí.

1. Vygenerování zpráv pro každý přechod mezi scénami do koncové scény ve tvaru `Příchod ze scény XX (Popis přechodu)`.
2. Hledání nedosažitelných scén a vygenerování zpráv pro ně.

Kapitola 10
Uživatelské testování přehrávače

Jak bylo zmíněno v kapitole 9, vytvořil jsem testovací příběh. V poslední části práce jsem aplikaci zpřístupnil testovacím uživatelům, aby otestovali funkčnost čtečky příběhu. K testování editoru jsem se bohužel nedostal, protože jsem v časovém horizontu práce nestihl zpracovat podrobnější nápovědu k IF formátu a vysvětlení funkcí editoru, proto jsem zpětnou vazbu k této důležité části aplikace shrnul v kapitole 9.

10.1 Průběh testování

Uživatelé dostanou k dispozici aplikaci s testovacím příběhem. Pro účely sběru dat čtenářských statistik je třeba se před čtením nejprve registrovat a přihlásit. Testovací uživatelé poté čtou příběh, jednou nebo opakovaně a na jeho konci mohou zanechat komentář nebo hodnocení.
Po dokončení celého testovacího procesu je respondentovi předložen dotazník k ohodnocení celého zážitku, identifikace chyb, nepřehledností a případně předností.
Testování se zúčastnilo **23 uživatelů**, kteří celkem **prošli interaktivní příběh 81-krát**. Celkem testovací uživatelé **v součtu přečetli 784 scén**.

10.2 Výsledky testování

Celkový dojem testovacích uživatelů byl vesměs pozitivní, na škále od 1 do 5 jej uživatelé v průměru shrnuli na **4,55**.

10.2.1 Funkčnost

- Funkčnost na škále od 1 do 5 uživatelé hodnotili na **4,81**.
- Jeden uživatel zmínil, že mu vyskakovalo chybové upozornění: `Uložení postupu ve hře selhalo`. Po přihlášení ale čtečka začala opět postup ukládat a fungovala. Z logů na serveru jsem si potvrdil, že klientský počítač odesílal požadavek aniž by byl přihlášen, který z toho důvodu nebyl správně vyhodnocen. Řešením této chyby bude ujistit se, že není možné, aby gamebook četl nepřihlášený uživatel, což je řešeno na úrovni frontendu.

10.2.2 Přehlednost, vzhled

- V souhrnu byla přehlednost hodnocena všemi uživateli plným počtem bodů.
- V otázce, kde měli hodnotit úroveň pochopení konceptu interaktivního příběhu, hodnotili na **4,70**.
- Jeden z uživatelů nepostřehl, že je přihlášen. Tento proces by mohl být vylepšen přidáním upozorněním na přihlášení, v této fázi okno s přihlašovacím formulářem po přihlášení zmizí a je nahrazeno tlačítkem pro spuštění hry (které se nepřihlášeným nezobrazí).
- Jeden uživatel zmiňoval přespříliš světlé pozadí.

10.2.3 Problémy a chyby

- Větší množství čtenářů se zmiňovalo o chybějícím tlačítku *Hrát znovu* na konci gamebooku, což se mi zdá jako naprosto relevantní připomínka. Doposud to bylo řešeno tak, že pod ukončením gamebooku byla možnost hodnotit a komentovat a tři tlačítka; *Hodnotit a komentovat*, *Pouze hodnotit*, *Ukončit hru bez uložení*. Po stisknutí tlačítek byli čtenáři shodně přesměrováni do detailu gamebooku, kde museli znovu kliknout na tlačítko *Hrát*.

10.2.4 Pozitivní zpětná vazba

- Všichni uživatelé shodně vyplnili, že by si s chutí přečetli další interaktivní příběh.
- Pouze jeden z nich zaškrtl, že by podobný interaktivní příběh chtěl sám vytvořit.
- Čtyři uživatelé výslovně chválili kvalitu a zábavnost příběhu.
- Dva z nich vyzdvihli přehlednost a jednoduchost ovládání, mimo jiné na zařízeních s menší obrazovkou.

10.2.5 Zpětná vazba k příběhu

Přestože jsem se v práci soustředil na aplikaci, nikoliv na příběh, krátce zde uvedu ohlasy čtenářů, které byly soustředěné spíše na literární část testovacího příběhu z kapitoly 9.

- Více uživatelů se zmínilo, že jeden z konců hry byl příliš krátký; pouze na tři přechody. S tím nelze než souhlasit.
- Jeden uživatel vytýkal gramatické nedostatky napříč příběhem.
- Zpětná vazba také narážela na to, zda-li je neúspěšný konec ve hře v příběhu odhalitelný předem nebo zda-li je v zásadě náhodný. Má reakce je na to taková, že logická vodítka k odhalení skutečného zločince a dosažení úspěšného konce v testovacím příběhu jsou, ale uznávám, že nejsou příliš viditelná a vyhrát gamebook napoprvé je těžké (což se ostatně podařilo pouze 2 čtenářům, méně než 10%).

10.3 Závěr testování

Zpětná vazba od uživatelů byla vesměs pozitivní. Odnesl jsem si z ní poznatky k vylepšení aplikace i dalších interaktivních příběhů.

Kapitola 11

Závěr práce

11.1 Rekapitulace cílů

Podívejme se do kapitoly 2, ve které jsem popisoval funkcionality, které bude můj nástroj obsahovat, kterými se bude lišit od již stávajících nástrojů, zmíněné v rešerši v kapitole 3, a zároveň do sekce 7.1, kde zmiňuji cíle pokročilé analýzy a zrekapitulujme, co se podařilo splnit.

11.1.1 Splněné cíle

- **Výpis kontextu hry do textu.**
- **Podmíněné vykreslení.**
- **Úprava kontextu hry.**
- **Textový vstup od čtenáře.** S jistými omezeními v analýze, ale pro většinu použití funkční.
- **Vizualizace statistik.**
- **Detekce nedokončených listů.**
- **Detekce konečnosti hry.**

11.1.2 Částečně splněné cíle

- **Zpožděný úsek.** Lze jej sice používat, ale zpožděná část IF formátu se vždy vykreslí až nakonec scény, což může způsobovat chyby, pokud za zpožděným úsekem následují jiné části IF formátu, které do čtečky vypisují nějaký obsah.
- **Pokročilá analýza.** Pro většinu případů je analýza funkční a při vývoji interaktivního příběhu je autorovi skutečně prakticky nápomocná, viz sekce 7.4 a kapitola 9.
- **Vizualizace vyvíjeného gamebooku.** Je pouze statická, ve vykresleném grafu pod editorem nelze přecházet mezi scénami pomocí odkazů v grafu.

11.1.3 Nesplněné cíle

- **Nápověda v editoru.** Tento bod jsem nestihl implementovat z časových důvodů. Editor je proto hůře použitelný pro nepoučeného autora, který bude mít k dispozici pouze syntax příkazů, viz Příloha A.

11.2 Osobní přínos

Kromě práce na aplikaci se zajímavým tématem, které se chci dále věnovat, jsem se naučil prakticky používat aplikaci založenou na komunikaci Spring Bootu na backendu a Reactu na frontendu, což hodnotím jako skvělý benefit této závěrečné práce. Nabyl jsem nových prakticky použitelných znalostí.

11.3 Současný stav

Aplikace je v tomto okamžiku nasazená na platformách Heroku (Java Spring, backend) a Vercel (React, frontend). Je volně dostupná na adrese gejmbukov.vercel.app, je možné číst gamebook zmíněný v kapitole 9, vytvářet a zveřejňovat vlastní příběhy. [36, 37]

11.4 Výhled do budoucna

Aplikaci se chci nadále věnovat a rozvíjet jí. Během vývoje jsem narazil na řadu překážek, z nichž ne všechny se podařilo překonat, a vznikly z nich podněty k dalšímu vývoji.

1. Implementace schvalovacího procesu gamebooku a jeho verzí pro uživatele s patřičnými oprávněními.
2. Rozdělení interaktivního příběhu do kapitol; přidání příkazu `CHAPTER`, pomocí kterého budeme kapitoly definovat; přidání vykreslení grafu kapitol, který bude užitečný zejména pro první plánovací část vývoje interaktivního příběhu.
3. Automaticky v editoru zvýraznit detekované příkazy formátu IF změnou typu písma.
4. Přidat do Debuggeru ve čtečce možnost měnit za běhu hodnoty proměnných.
5. Přidat možnost parsovat a nezobrazovat ve čtečce řádky začínající `//`, představující autorovy komentáře.
6. Přidat funkcionalitu, umožňující provést akci kdykoli, kdy daná proměnná dosáhne konstantní hodnoty (tento požadavek vzešel při vývoji testovacího příběhu, kdy jsem musel na více místech opakovat podobnou podmínku při omezení počtu navštívených lokací).
7. Implementovat příkaz `ONLY ONCE`, při kterém bude možné navštívit scénu pouze jednou a čtečka nebude zobrazovat přechody na navštívené *only-once* scény.

8. Optimalizovat volání ukládacích requestů v editoru; jejich počet je možné snížit.
9. Implementovat příkazy `LIST LENGTH` a negace.
10. Přidat možnost odstranit proměnnou, která je potřebná pouze v určité části příběhu aby dále nekomplikovala přehlednost výpisu dosažitelných kontextů.
11. Přehlednější zobrazení dosažitelných kontextů v tabulce s možností filtrování a zobrazení pouze části informací.
12. Vytvoření nápovědy v editoru, která bude živě zobrazovat popis příkazů, které právě autor píše. Zároveň bude našeptávat a doplňovat za autora.
13. Živý graf vyvíjeného gamebooku s odkazy na scény.
14. Vykreslovat do zvláštního grafu definice a použití jednotlivých stavů kontextu hry.

Literatura

[1] Nakladatelství mytago. URL: https://mytago.cz/, navštíveno 25.11.2021.

[2] Jörg Benne. *Verax: Experiment, ISBN: 978-80-87761-63-2*. Mytago, 2020.

[3] Shuky. *Zapadákov, ISBN: 978-80-270-6400-7*. Rexhry, 2019.

[4] Isaac Schankler Zoe Quinn, Patrick Lindsey. Depression quest. 2013. URL: http://www.depressionquest.com/, navštíveno 25.11.2021.

[5] Mythion games. Gamebook, text adventure. 2021. URL: https://play.google.com/store/apps/details?id=com.mythiongames.awf&hl=cs&gl=US, navštíveno 25.11.2021.

[6] Mythion games. Gamebook adventures. 2014. URL: http://gamebookadventures.com/, navštíveno 25.11.2021.

[7] Fantasy gamebook engine. 2008. URL: https://web.archive.org/web/20100228061211/http://gamebooky.syslik.net/, navštíveno 25.11.2021.

[8] Gamebook authoring tool. 2021. URL: https://www.crumblyheadgames.co.uk/the-gamebook-authoring-tool/, navštíveno 25.11.2021.

[9] Online gamebooks. URL: https://www.onlinegamebooks.com/, navštíveno 25.11.2021.

[10] Squiffy, tool for creating interactive fiction. URL: https://textadventures.co.uk/squiffy/editor, navštíveno 25.11.2021.

[11] Quest, software for creating text adventure games. URL: http://docs.textadventures.co.uk/quest/, navštíveno 25.11.2021.

[12] Twine, tool for telling nonlinear stories. URL: https://twinery.org/, navštíveno 25.11.2021.

[13] Ren'py, visual novel engine. URL: https://www.renpy.org/doc/html/quickstart.html, navštíveno 25.11.2021.

[14] Tads, authoring system for writing interactive fiction. 2013. URL: http://www.tads.org/index.htm, navštíveno 25.11.2021.

[15] Github repository twinery. URL: https://github.com/klembot/twinejs, navštíveno 27.11.2021.

[16] J. Robinson Wheeler Kevin Jackson-Mead. If theory reader. URL: https://www.inform-fiction.org/manual/if_theory_reader.pdf, navštíveno 6.12.2021.

[17] Interactive fiction on steam. URL: https://store.steampowered.com/tags/en/Interactive+Fiction, navštíveno 9.12.2021.

[18] Interactive fiction on itch.io. URL: https://itch.io/games/tag-interactive-fiction, navštíveno 9.12.2021.

[19] Java regex performance. URL: https://www.baeldung.com/java-regex-performance, navštíveno 5.5.2022.

[20] 10 popular web frameworks for web app development in 2021. URL: https://www.monocubed.com/10-most-popular-web-frameworks/, navštíveno 3.12.2021.

[21] What are the best web frameworks to create a web rest api? URL: https://www.slant.co/topics/1397/, navštíveno 3.12.2021.

[22] Best web development frameworks. URL: https://hackr.io/blog/web-development-frameworks, navštíveno 3.12.2021.

[23] React js. URL: https://reactjs.org/, navštíveno 3.12.2021.

[24] Oracle. Java persistence api. URL: https://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html, navštíveno 3.12.2021.

[25] Accesing data with jpa in spring. URL: https://spring.io/guides/gs/accessing-data-jpa/, navštíveno 3.12.2021.

[26] Typescript. URL: https://www.typescriptlang.org/why-create-typescript, navštíveno 5.5.2022.

[27] Bootstrap: Html, css a js knihovna. URL: https://getbootstrap.com/, navštíveno 26.3.2022.

[28] React-bootstrap. URL: https://react-bootstrap.github.io/, navštíveno 26.3.2022.

[29] Spring security - authentication with database. URL: https://www.baeldung.com/spring-security-authentication-with-a-database, navštíveno 20.2.2022.

[30] Spring security. URL: https://spring.io/projects/spring-security, navštíveno 20.2.2022.

[31] Spring security - user login in react. URL: https://www.baeldung.com/spring-security-login-react, navštíveno 20.2.2022.

[32] Spring security - user login. URL: https://www.baeldung.com/spring-security-login, navštíveno 20.2.2022.

[33] Spring security documentation. URL: https://docs.spring.io/spring-security/reference/servlet/authentication/index.html, navštíveno 20.2.2022.

[34] E. W. Dijkstra. Algol-60 translation. 1961. URL: https://www.cs.utexas.edu/~EWD/MCReps/MR35.PDF, navštíveno 26.3.2022.

[35] State machine cat. URL: https://state-machine-cat.js.org/, navštíveno 26.3.2022.

[36] Heroku. URL: https://www.heroku.com/, navštíveno 10.5.2022.

[37] Vercel. URL: https://vercel.com/, navštíveno 10.5.2022.

Příloha A

Příkazy IF formátu

```
[LINK TO SCENE 2 WITH LABEL Pokračuj do lesa] //odkaz na další scénu

[END VICTORY] //výherní ukončení gamebooku
[END DEFEAT] //neúspěšné ukončení gamebooku

[SET nazevPromenne TO 5 ] //přiřazení do proměnné
[SET textovyRetezec TO "jablko" ]
[SET seznam TO {1, 2, 5} ] //seznam definujeme ve složených závorkách

[PRINT nazevPromenne] //výpis proměnné

[IF nazevPromenne = 5]
...
[ELSE] / [ELSE IF nazevPromenne < 5]
...
[END IF]

[DELAY 5.0 SECONDS]
...
[END DELAY]

[ADD 7 TO seznam] //přidání prvku do seznamu
[REMOVE 2 FROM seznam ] //odebrání prvku se seznamu
[REMOVE ALL FROM seznam ] //vyprázdnění seznamu

//Uživatelské vstupy
[SAVE TO promenna FROM INPUT AS REAL-NUMBER WITH LABEL ...]
[SAVE TO promenna FROM INPUT AS DECIMAL-NUMBER WITH LABEL ...]
[SAVE TO promenna FROM INPUT AS STRING WITH LABEL ... ]

[SAVE TO promenna FROM SELECT seznamPostav WITH LABEL ... ]
[SAVE TO obor FROM SELECT {"Matematika", "Informatika" } WITH LABEL ...]

//Výrazy, mohou být součástí podmínek a přiřazení proměnných
RANDOM NUMBER FROM 1 TO 12
LIST seznam CONTAINS 5
GET ITEM FROM LIST seznam AT POSITION 1 //indexováno od 0
```

Příloha B

GitLab repozitáře

Zdrojový kód práce je uložen ve dvou GitLab repozitářích dostupných na adrese `https://gitlab.com/gejmbukov`. Je rozdělen na backendovou a frontendovou část.