Webová aplikace pro tvorbu role play her se zaměřením na výuku cizích jazyků

Bc. Miroslav Švarc

Diplomová práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Miroslav Švarc**
Osobní číslo:	**A21514**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Softwarové inženýrství**
Forma studia:	**Kombinovaná**
Téma práce:	**Webová aplikace pro tvorbu role play her se zaměřením na výuku cizích jazyků**
Téma práce anglicky:	**Web Application for the Creation of Role Playing Games with a Focus on Foreign Language Learning**

Zásady pro vypracování

1. Nastudujte a popište problematiku vývoje webových aplikací.
2. Rozepište možnosti vývoje příběhových výukových her cizího jazyka.
3. Navrhněte webovou aplikaci pro tvorbu role play her s cílem výuky cizího jazyka.
4. Zvolte vhodné technologie pro implementaci.
5. Implementujte navržené řešení.
6. Vaše výsledné řešení vhodně otestujte a porovnejte se stávajícími.
7. Věnujte se zabezpečení aplikace.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. Stauffer, Matt. Laravel: Up & Running. Sebastopol : O'Reilly Media, 2019. 9781492041214.
2. Jez Humble, David Farley. Modern Software Engineering. New York City : Pearson Education (US), 2022. 9780137314911.
3. Zichermann, Gabe. Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps. Sebastopol : O'Reilly Media, 2011. 9350234548.
4. Bill Scott, Theresa Neil. Designing web interfaces . Sebastopol : O'Reilly, 2009. 9780596516253.
5. Regina O. Obe, Leo S. Hsu. PostgreSQL : up and running. Sebastopol : O'Reilly, 2012. 9781449326333.
6. Marsh, Joel. UX pro začátečníky : (rychlokurz – 100 lekcí). Brno : Zoner Press, 2019. 9788074133978.

Vedoucí diplomové práce: **Ing. Petr Žáček, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **2. prosince 2022**
Termín odevzdání diplomové práce: **26. května 2023**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne Miroslav Švarc, v. r.

ABSTRAKT

Tato práce se zabývá návrhem a vývojem webové aplikace pro vytváření rolových a simulačních her za účelem využití ve výuce. V teoretické části je nejdříve popsána problematika vývoje webových aplikací a dále následuje popis rolových a simulačních her, jejich využití ve výuce a rozdělení na fyzické a digitální hry. Praktická část se zabývá návrhem této aplikace, který zahrnuje analýzu požadavků, výběr architektury, popis způsobu ukládání dat a návrh uživatelského rozhraní. Dále jsou popsány veškeré použité technologie. Následující část se věnuje implementaci navrženého řešení a jejímu zabezpečení. Na závěr je popsán průběh testování aplikace, které probíhalo s reálnými uživateli, kteří budou aplikaci v budoucnu používat.

Klíčová slova: webová aplikace, rolové hry, simulace, výuka, tvorba her, nuxt3, laravel, postgresql

ABSTRACT

This thesis deals with the design and development of a web application intended for creating educational role-playing and simulation games. In the theoretical part, the issues of web application development are first described, followed by a description of role-playing and simulation games, their use in education and the division into physical and digital games. The practical part deals with the design of this application, which includes requirements analysis, architecture selection, description of the data storage method and user interface design. Furthermore, all the technologies used are described. The following section deals with the implementation of the proposed solution and its security. Finally, the testing of the application is described, which was carried out with real users who will use the application in the future.

Keywords: web application, role-play games, simulation, teaching, game creation, nuxt3, laravel, postgresql

Chtěl bych poděkovat svému vedoucímu, Ing. Petru Žáčkovi, Ph.D., za ochotu a veškerou pomoc při vytváření této práce.

Dále bych rád poděkoval své rodině a přítelkyni, kteří mě po celou dobu studia podporovali a byli mi vždy oporou.

OBSAH

ÚVOD

V současné době se stále více přistupuje k novým metodám výuky, a to jak ve školství, tak i mimo něj. Učitelé se snaží své studenty motivovat a zaujmout, a tak vyhledávají nové metody a aktivity, které by studenty bavily. Jednou z těchto metod je výuka pomocí hraní rolových a simulačních her. Tato metoda zatím není v českém školství příliš rozšířená a většina učitelů se s ní nikdy nesetkala. Učitelé, kteří tuto metodu znají a používají, si většinou vytvářejí své vlastní hry, a to nejčastěji ve fyzické podobě, popřípadě pokud jsou technicky zdatní, v podobě jednoduchých webových stránek. V takovém případě je pak celý průběh hry napsán přímo v kódu.

Zatím neexistuje žádný univerzální nástroj, který by umožnil snadnou tvorbu rolových a simulačních her bez nutnosti pokročilých technických znalostí. Učitelé tak často využívají software, který není primárně určen pro tvorbu her, ale například pro vytváření prezentací. Existence nástroje, který by sloužil specificky k tomuto účelu, by umožnila větší rozšíření této metody mezi učitele a žáky, jelikož by bylo možné hry vytvářet jednoduše i bez technických znalostí a libovolně je sdílet mezi sebou. V dnešní době již velká většina škol má zkušenosti s využívání digitálních technologií k výuce a mezi studenty i učiteli se podobné nástroje stávají stále více oblíbené.

Tato práce se zaměřuje na návrh a vývoj aplikace, která by umožnila vytváření rolových a simulačních her pro využití ve výuce. Skládá se ze dvou částí, teoretické a praktické. Teoretická část se poté dále dělí na dvě samostatné hlavní kapitoly.

První kapitola teoretické části je zaměřena na problematiku webových aplikací a přístupy k jejich tvorbě. Obsahuje stručnou historii vývoje internetových stránek, vznik prvních webových aplikací a současný stav vývoje. Dále se věnuje rozdílným přístupům, architekturám a technologiím, které se k vývoji používají.

Druhá kapitola se zaměřuje na rolové a simulační hry. Obsahuje základní informace, definice a možnosti využití ve výuce. Také popisuje různé druhy rolových a simulačních her. Poslední část se poté věnuje existujícím řešením a jejich výhodám a nevýhodám.

Praktická část je také rozdělena na několik částí. První část se zabývá návrhem a vývojem aplikace, která umožní jednoduše vytvářet rolové a simulační hry bez pokročilých technických znalostí a s minimálním úsilím. Uživatele provede jednotlivými kroky sestavení hry, ve kterých tvoří jednotlivé části příběhu, a ty poté postupně navazuje na sebe podle předem určených pravidel. Vytvoření jednoduché hry tak nezabere více než několik desítek minut. Stačí jen vymyslet příběh, dodat libovolné multimediální materiály, to vše poskládat do jednotlivých obrazovek a pospojovat je tak, jak mají za sebou následovat.

Druhá část se poté věnuje testování implementovaného řešení, a to jak vlastními

testy, tak s pomocí reálných uživatelů. Testování se zaměřuje nejen na odhalení případných chyb, ale také na vylepšení funkcionalit tak, aby se uživatelům aplikace dobře používala. Kromě toho se také zaměřuje na porovnání implementovaného řešení s již existujícími aplikacemi.

Třetí a zároveň poslední část obsahuje shrnutí celé práce, zhodnocení výsledného řešení a popisuje také jeho další možné rozšíření.

I. TEORETICKÁ ČÁST

1 WEBOVÉ APLIKACE

V současné době je stále častější, že jsou klasické počítačové programy nahrazovány webovými aplikacemi. Jedná se komplexní a interaktivní internetové stránky, které je jsou schopny plnohodnotně nahradit. Pro uživatele to přináší značnou výhodu, jelikož si nemusí daný program instalovat lokálně na svém počítači, ale má ho přístupný odkudkoliv ve svém internetovém prohlížeči. [1] Stejně tak webová aplikace funguje nezávisle na operačním systému, který uživatel používá. U desktopových aplikací toto většinou neplatilo a pro různé operační systémy a platformy museli být vydávány rozdílné verze daného programu.

Díky rychlému pokroku na poli webových technologií je dnes již možné přenést téměř libovolný software do formátu webové aplikaci a používat jej online. Historii vzniku aplikací tohoto typu je věnována kapitola 1.1. Současný stav vývoje webových aplikací je popsán v kapitole 1.2.

1.1 Historie webových aplikací

Historie webových aplikací sahá až do roku 1995, kdy společnost Netscape představila klientský skriptovací jazyk, zvaný JavaScript[1)]. Ten uživatelům umožňoval přidávat do webových stránek různé dynamické elementy, které se spouštěli na straně klienta, tedy přímo v prohlížeči uživatele. Proces načtení stránky již nezávisel pouze na principu zaslání požadavku na serveru a zobrazení vrácené odpovědi v podobě html stránky. Místo toho umožňoval různé operace, jako například dynamické skrývání/zobrazování částí stránek, jejich načítání na vyžádání, nebo validaci formulářů a dalších vstupů, přímo na straně uživatele. [2]

V roce 2005 byl poprvé představen Ajax[2)], a tím byla odstartována éra webových aplikací. Jedním z nejaktivnějších průkopníků se stala například společnost Google a jejich aplikace pro elektronickou poštu Gmail. Ta se stávala čím dál tím více interaktivní, a to díky klientským skriptům, které byly schopné získávat ze serveru data bez načtení celé stránky (například umožňovala stahovat nové emaily a zobrazit je hned poté, co byly doručeny).[3]

1.2 Současný stav

V dnešní době již existuje mnoho aplikací, které byly transformovány do webové podoby, což jim umožňuje oslovit ještě více potencionálních zákazníků. Mezi nesporné

[1)] JavaScript je multiplatformní, objektově orientovaný, událostmi řízený skriptovací jazyk.

[2)] AJAX (Asynchronous JavaScript and XML) je v informatice obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich kompletního znovunačtení.

výhody webových aplikací patří především nízké nároky na hardware, jelikož aplikace je spuštěna na serveru a ne na zařízení uživatele. Díky tomu je kladen nárok pouze na stabilní internetové připojení. Dále umožňují jednoduché aktualizace, kdy uživatel nemusí nic instalovat, ale vše se provede automaticky na serveru.

Mezi aplikace, které byly převedeny do podoby webových aplikací patří například produkty společnosti Adobe, jako je Adobe Photoshop[3)], které lze díky jejich webové verzi spustit a používat na libovolném zařízení a uživatel nepotřebuje vlastnit drahý a výkonný počítač, který dnešní klasické verze této aplikace vyžadují. Dalším příkladem může být společnost Microsoft, která do své služby GitHub implementovala textový editor Visual Studio Code, který tak lze spustit v prohlížeči přímo z libovolného repozitáře kódu. Díky tomu může uživatel upravit část kódu přímo v prohlížeči a to se stejným uživatelským zážitkem, jako v desktopové aplikaci.

Zároveň existuje již celá řada technologií, které pohodlně umožňují vývoj náročných webových aplikací. Jedná se především o různé knihovny jazyka JavaScript pro tvorbu uživatelských rozhraní, které umožňují asynchronní komunikaci se serverem, dynamické načítání obsahu a responzivní chování. Velké množství těchto knihoven je vyvíjeno jako software s otevřeným zdrojovým kódem (pro takový software se používá anglické slovní spojení open-source, dále v textu bude používáno toto označení.) a jejich, často velmi rozsáhlá, uživatelská základna pravidelně přidává nové funkcionality a opravuje chyby. Mezi nejznámější patří například knihovna React[4)] od společnosti Meta Platforms, knihovna Vue.js[5)], nebo knihovna Angular[6)].

Podobných knihoven existuje nespočet a pravidelně se objevují nové. Tři výše zmíněné patří v dnešní době mezi nejčastěji používané, jak vyplynulo z každoročního průzkumu *The 2022 State of JS survey* [4]. Statistiku nejčastěji používaných knihoven a frameworků dle tohoto průzkumu lze vidět na obrázku 1.1, kde osa X představuje jednotlivé roky, kdy výzkum probíhal a osa Y procento uživatelů, kteří by danou knihovnu využili pro nový projekt. Z obrázku vyplývá, že výše zmíněné knihovny v posledních letech s náskokem předbíhají všechny ostatní.

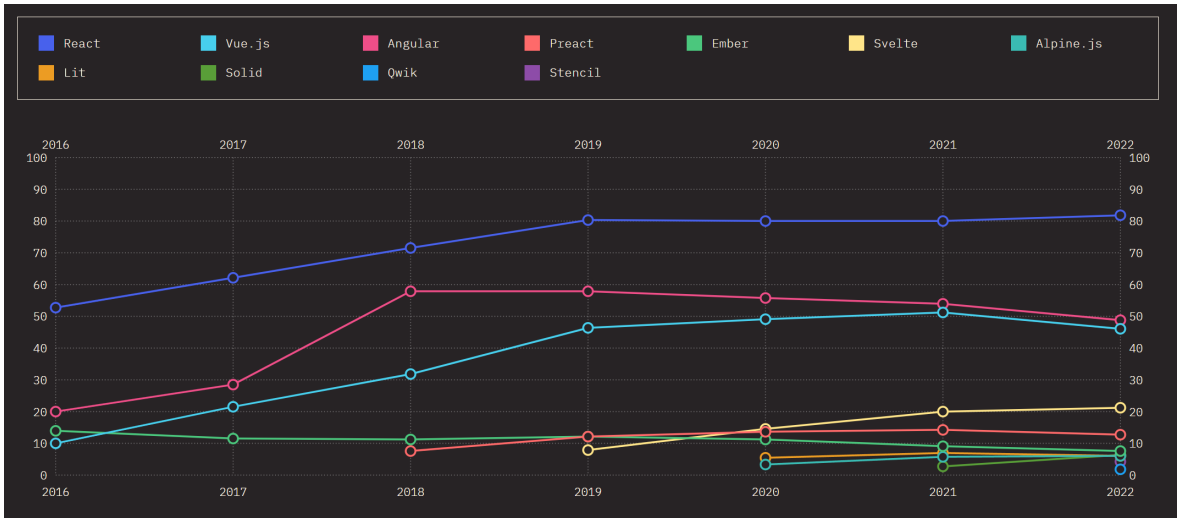
Tyto knihovny slouží především pro tvorbu uživatelských rozhraní a jednoduší logiky, jako je komunikace se serverem nebo vykreslování dat. Pro složitější operace a komunikaci s databází je nutné použít tzv. backend. Backend označuje kód, který je vykonáván na serveru a slouží především k provádění náročnějších výpočtů a ke komunikaci s databází. Pro tvorbu backendu je možné využít některý z mnoha programovacích

[3)] Adobe Photoshop je bitmapový grafický editor pro tvorbu a úpravy bitmapové grafiky vytvořený firmou Adobe Systems.

[4)] React je open-source knihovna vyvíjená společností Meta, dostupná na https://reactjs.org

[5)] Vue.js je open-source knihovna, dostupná na https://vuejs.org

[6)] Angular je bezplatný a open-source webový framework založený na TypeScriptu vedený týmem Angular ve společnosti Google



Obrázek 1.1 Statistika využívání nejznámějších Javascriptových frameworků a knihoven v %

jazyků. Mezi nejčastěji používané patří Node.js, PHP, .C# nebo Java. Většina těchto jazyků se používá v rámci některého z jejich frameworků. Mezi ty patří například Express.js pro Node.js, Laravel pro PHP, .NET pro C#, nebo Spring pro jazyk Java. Výběr vždy záleží pouze na vývojáři, pro koncového uživatele aplikace není podstatný.

1.3 Přístupy k tvorbě webových aplikací

Stejně jako existuje mnoho různých programovacích jazyků, knihoven a frameworků pro tvorbu webových aplikací, existují také různé typy architektur. Ty popisují, jak je software logicky rozdělen, jak jsou jednotlivé části navzájem propojené a jakým způsobem komunikují mezi sebou.

Mezi nejčastěji používané softwarové architektury, využívané pro vývoj webových aplikací, patří:

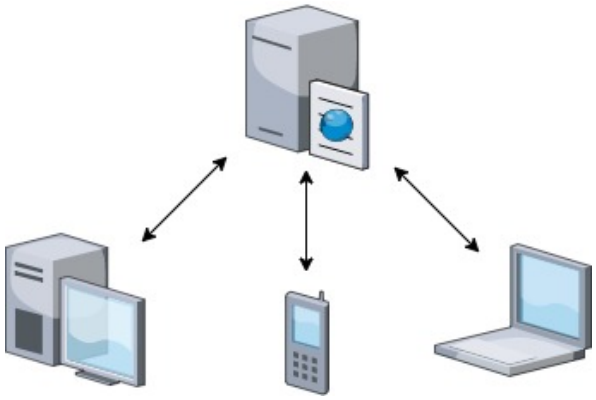
- Architektura klient-server
- Monolitová architektura
- Architektura orientovaná na služby
- Architektura mikroslužeb
- Architektura řízená událostmi

Následující kapitoly se věnují podrobnějšímu popisu těchto architektur, včetně jejich výhod a nevýhod.

1.3.1 Architektura klient-server

Při použití architektury klient-server je software rozdělen na dvě samostatné části, kde jedna část zastupuje roli klienta a druhá roli serveru. Klientskou část často představuje grafické rozhraní a jeho funkce. Serverová část poté obsahuje logiku aplikace a funkce pro komunikaci s databází. [5] Klient a server spolu navzájem komunikují a to v počítačové síti, nebo lokálně na stejném počítači.

Tato komunikace probíhá tím způsobem, že klient sestaví požadavek, zašle ho na server a poté čeká na odpověď. Server požadavek převezme, provede definované operace, sestaví odpověď a pošle ji zpět klientovi. Schéma této komunikace lze vidět na obrázku 1.2, kde zařízení v horní částí představuje server, se kterým komunikují ostatní zařízení ze spodní části obrázku.



Obrázek 1.2 Ukázka architektury Klient-server (vlastní tvorba, vytvořen službou `diagrams.net`)

Příkladem této architektury může být internetový prohlížeč, který zastupuje klienta a síťový server, na kterém je provozována webová stránka. Klient přistupuje k serveru, získá z něj data o webové stránce, a poté je zobrazí uživateli na jeho zařízení. V této komunikaci se samozřejmě nacházejí i další účastníci, ti jsou ale v tomto případě pro jednoduchost zanedbáni.

Podobnou architekturou je model peer-to-peer[7)], který se od architektury klient-server liší tím, že každá instance může fungovat zároveň jako klient i jako server. V tomto případě pak neexistuje žádný centrální bod, ale všichni účastníci komunikují spolu navzájem. [6]

1.3.2 Monolitová architektura

Monolitová architektura je tradiční model softwaru, který je tvořen jako jeden celek a je nezávislý na svém okolí. Slovo „monolit“ je často spojováno s něčím obrovským,

[7)] V doslovném překladu rovný s rovným. Často se používá zkratka P2P.

což v tomto případě není daleko od pravdy. Celý program je v jedné základně kódu, která zahrnuje veškerou logiku. Pokud je v takovém kódu potřeba provést jakékoliv změny, musí se často upravit a znovu sestavit celý program, vzhledem k tomu, jak na sebe jednotlivé části navazují. [7]

Výhodou monolitové architektury je rychlejší vývoj, jelikož se vše nachází na jednom místě, jednoduší testování a hledání chyb. Nevýhodami je náročná implementace větších změn, nízká flexibilita, co se týká použitých technologií, a v případě nárůstu velikosti aplikace následně zpomalení dalšího vývoje, jelikož, jak již bylo zmíněno výše, po implementaci každé změny je nutné znovu provést sestavení celé aplikace a provedení všech testů.

Monolitová architektura je vhodná především pro menší aplikace a také menší týmy vývojářů. Pro větší aplikace je vhodnější využít architekturu orientovanou na služby, popřípadě architekturu mikroslužeb.

1.3.3 Architektura orientovaná na služby

Architektura orientovaná na služby (často se používá zkratka SOA[8)]) definuje přístup k tvorbě softwaru, který se zaměřuje na znovupoužitelné a interoperabilní komponenty, které představují jednotlivé služby. Tyto služby využívají standardní rozhraní a architektonické vzory, aby je bylo možné jednoduše zakomponovat do nových aplikací. To značně ulehčuje práci vývojářům, kteří nemusí trávit mnoho času analýzou toho, jak implementovat připojení dané komponenty k ostatním. [8]

Každá služba zahrnuje veškerý kód a data, která jsou potřebné k úspěšnému spuštění kompletní funkce (takovou funkcí může být například zjištění stavu účtu klienta nebo výpočet měsíční splátky). Rozhraní služby poskytují jednoduché propojení, což znamená, že je lze volat jen s malou nebo téměř žádnou znalostí o tom, jak je služba uvnitř implementována. To značně snižuje závislosti mezi aplikacemi. [8]

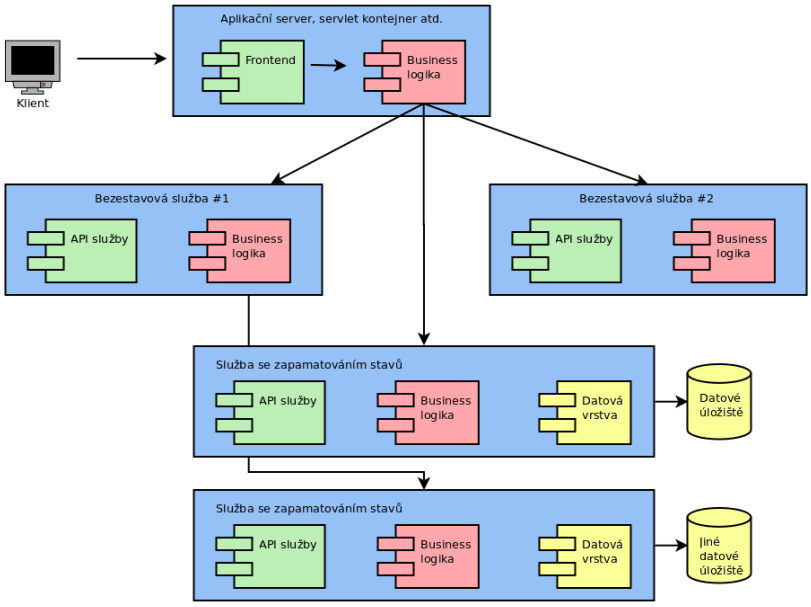
Na podobném principu stojí i architektura mikroslužeb, která je popsána v následující kapitole.

1.3.4 Architektura mikroslužeb

Architektura mikroslužeb je velmi podobná architektuře orientované na služby. Hlavním rozdílem je, že mikroslužby jsou velmi malé části kódu a celý softwarový produkt pak obsahuje velké množství těchto částí. Jen pro vykonání jednoduchého uživatelského požadavku může aplikace zavolat mnoho interních mikroslužeb, aby mohla sestavit odpověď pro uživatele.

[8)] SOA je akronym pro anglický název Service-oriented architecture

Mikroslužby, stejně jaké služby, představují oddělenou část kódu, která obsahuje veškerý kód nutný pro svou funkcionalitu. Ostatní mikroslužby s ní poté komunikují pomocí definovaných rozhraní. [9] Schéma architektury mikroslužeb lze vidět na obrázku 1.3.



Obrázek 1.3 Ukázka architektury mikroslužeb (převzato z [10])

Oproti architektuře orientované na služby se liší také v protokolech využívaných pro komunikaci. Zatímco architektura orientovaná na služby využívá pro komunikaci mezi službami převážně protokoly SOAP[9)] a REST[10)], architektura mikroslužeb využívá méně náročné protokoly, jako je například HTTP[11)] nebo AMQP[12)]. [11]

1.3.5 Architektura řízená událostmi

Architektura řízená událostmi používá události ke spuštění a komunikaci mezi oddělenými službami. Jedná se o běžný přístup v moderních aplikacích. Událost je změna stavu (například přidání produktu do košíku). Tato událost může buď nést svůj stav (v případě produktu například jeho cenu a počet kusů), nebo může představovat nějaký identifikátor (například notifikace o odeslání objednávky). [12]

Tato architektura se skládá ze tří hlavních komponent. Těmi jsou zdroj události, cesta události a příjemce události. Zdroj vytvoří událost a pošle ji na směrovač, který události filtruje a posílá je podle definovaných cest příjemcům. [12]

[9)] SOAP je akronym pro Simple Object Access Protocol
[10)] REST je akronym pro REpresentational State Transfer and an architectural style for distributed hypermedia systems
[11)] HTTP je akronym pro The Hypertext Transfer Protocol
[12)] AMQP je akronym pro Advanced Message Queuing Protocol

Výhodami toho přístupu je především možnost škálovatelnosti a nízké náklady na provoz, jelikož veškeré služby se spouští na vyžádání, pouze pokud jsou potřeba. [12]

1.4 Frontend

Frontend označuje tu část aplikace, se kterou interaguje uživatel. Obsahuje design aplikace, textový a multimediální obsah a umožňuje uživateli navigaci mezi jednotlivými stránkami. [13]

Ve frontendové části jsou také sestavovány požadavky, které jsou poté posílány na část backendovou (backendová část je popsána v kapitole 1.5). Po odeslání požadavku následuje čekání na odpověď a následně její zpracování.

Jedním z hlavních cílů frontendové části je vytvořit pro uživatele příjemný uživatelský zážitek (používá se označení UX/UI[13)]). To znamená, že aplikace by měla být intuitivní a jednoduše použitelná. [13] Vytvořit uživatelsky přívětivou aplikaci může být poměrně náročný úkol, jelikož uživatelé budou k aplikaci přistupovat z různých zařízení, které mají jiný způsob používání a aplikace na nich vypadá jinak, například kvůli velikosti nebo rozlišení obrazovky.

Největším rozdílem je přístup pomocí počítače nebo notebooku a pomocí mobilního telefonu. Jelikož tato zařízení mají rozdílné velikosti obrazovky a poměry stran, musí na to být aplikace přizpůsobena. K tomu se využívá responzivní design.

Responzivní design je přístup, který slouží k tomu, že na na všech zařízeních se webová stránka zobrazí správně a bude připravena k používání. Značkovací jazyk HTML, který se používá k tvorbě webových stránek, je přirozeně responzivní. To znamená, že při změně velikosti okna se text automaticky zalomí, aby byl stále viditelný. [14]

Problém ovšem nastává u více komplikovaných webových stránek, které se skládají z velkého množství bloků a grafický prvků. U takových stránek je nutné, aby vývojář specifikoval, jak přesně se tyto prvky mají chovat při změně rozlišení.

Základním stavebním kamenem jsou technologie Html a CSS. Html je značkovací jazyk, pomocí kterého je sestavena struktura webové stránky. Jednotlivé části se označují pomocí definovaných značek, jak je vidět na obrázku 1.4, podle kterých poté internetový prohlížeč rozpozná, o jaký prvek se jedná a jakým způsobem ho má zobrazit.

Jazyk CSS oproti tomu slouží pouze k popisu toho, jak mají tyto prvky vypadat. Lze tak nastavit velké množství parametrů, jako například barva, velikost, odsazení, zarovnání a mnoho dalších. Kód na obrázku 1.5 ukazuje, jak lze pomocí CSS definovat barvu pozadí stránky a vzhled nadpisu a odstavce.

Tyto dvě technologie jsou dostačující pro jednoduché statické webové stránky. To

[13)] UI je zkratka pro User Interface, neboli Uživatelské rozhraní a UX pro User Experience, neboli Uživatelský zážitek

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>HTML 5 Boilerplate</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <script src="index.js"></script>
  </body>
</html>
```

Obrázek 1.4 Ukázka značkovacího jazyka HTML

znamená, že celý obsah stránky se načte, zobrazí uživateli najednou a nijak se již nemění. Pokud chceme zobrazovat prvky na stránce dynamicky, načítat data postupně nebo zobrazovat pro různé uživatele různé části, je nutné použít skriptovací jazyk JavaScript. Ten umožňuje provádět operace se strukturou webové stránky. Tyto tři technologie byly dlouhou dobu hlavními zástupci technologií pro vývoj webových stránek a aplikací.

```css
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
```

Obrázek 1.5 Ukázka stylovacího jazyka CSS

V dnešní době existuje celá řada knihoven a frameworků, které zjednodušují vývoj frontendové části. Mezi ty nejznámější patří React, VueJS, Angular[14] nebo Svelte[15]. Pro tyto knihovny existují další nadstavby, které je rozšiřují o nezbytné funkcionality pro kompletní vývoj webových aplikací. Mezi ty patří například NextJS pro React, Nuxt3 pro VueJS nebo SvelteKit pro Svelte.

[14] Angular je open-source framework od společnosti Google, dostupný na https://angular.io
[15] Svelte je open-source framework, dostupný na https://svelte.dev

1.4.1 Responzivní design

Nedílnou součástí moderních webových stránek a aplikací je také responzivní design. To je přístup, který doporučuje, aby se uživatelské rozhraní stránky nebo aplikace měnilo dle parametrů jednotlivých zařízení, jako je jejich velikost obrazovky, rozlišení a orientace. Hlavním důvodem tohoto přístupu je možnost zobrazení webové stránky na tabletu nebo mobilním telefonu, které mají mnohem menší obrazovku, než notebook nebo počítač. Stránka, která neobsahuje responzivní design, se na takovém zařízení většinou neukáže správně a různé prvky jsou buď zdeformované, nebo vykreslené mimo zorné pole.

V době, kdy se pro prohlížení webových stránek poprvé začali ve velkém používat mobilní telefony, bylo poměrně častou záležitostí vytvořit dvě totožné webové stránky, kde jedna sloužila pro zobrazení na počítači a druhá na mobilním telefonu. Některé webové stránky byly vytvořeny dokonce ve více verzích, například pro tablety a další zařízení. Tento přístup samozřejmě funguje, jelikož stránka je vždy perfektně vyladěná pro dané zařízení, ale má i svá úskalí. Největším problémem je, že jakékoliv úpravy, testování a údržba musí být prováděna na všech verzích zvlášť. To zvyšuje nejen časové, ale hlavně finanční náklady. [15]

Druhým přístupem jsou tzv. Media queries (mediální dotazy). Ty se poprvé objevily ve standardu W3 [16] již v roce 2001 a v dnešní době jsou již nedílnou součástí responzivního designu.

Media queries jsou součást stylovacího jazyka CSS. Umožňují definovat rozsah rozlišení obrazovky, pro které budou platit CSS styly zapsané uvnitř. Zápis se označuje klíčovým slovem *@media*, za kterým následuje orientace obrazovky, rozsah hodnot rozlišení a další volitelné parametry. Oba tyto parametry jsou nepovinné a mohou se různě kombinovat pomocí logických operátorů. Ukázku zápisu Media query lze vidět na obrázku 1.6, kde parametr *screen* označuje, že se jedná o klasické zobrazení stránky a parametr *max-device-width* určuje maximální šířku obrazovky v pixelech. Pokud jsou oba tyto parametry splněny, použije se CSS kód, který je zapsaný uvnitř závorek.

Díky Media queries je tedy možné definovat, jak bude stránka vypadat na různých zařízeních.

```
@media screen and (max-device-width: 768px) {
  p {
    font-size: 2rem;
  }
}
```

Obrázek 1.6 Ukázka definice Media query (vlastní tvorba)

V posledních letech se stále více používají moderní rozložení, která využívají flexibilní (CSS Flexible Box Layout, Flexbox) nebo mřížkové (CSS Grid layout) rozložení. Jedná se dva rozdílné přístupy, kde každý má svá pole využití.

Flexbox je koncipován jako jedno dimenzionální model rozložení a metoda, která umožňuje kombinovat distribuci prvků v prostoru s pokročilými možnostmi zarovnání. Flexbox může být nastaven do dvou módů – řádek (row) a sloupec (column). To značí, jakým směrem budou jednotlivé prvky řazeny. Další parametry poté určují, jak budou prvky rozmístěny v dané oblasti a jakým způsobem budou zarovnané. [17]

Oproti tomu rozložení grid pracuje s mřížkou, ve které jsou prvky umístěny. Definuje se počet sloupců a jejich parametry a prvky se do nich rozmístí. Tato metoda je ideální pro tvorbu základního rozložení webové stránky, jelikož ji umožňuje jednoduše rozdělit do řádků a sloupců. [18]

Oba tyto přístupy se používají i v responzivním designu. Díky jejich použití se v některých případech prvky sami přeskládají, pokud se změní velikost okna a není tak nutné definovat jejich rozložení pro různé druhy rozložení. I přes to je často lepší mít větší kontrolu nad chováním designu a tak se oba přístupy kombinují právě s Media queries. Tato kombinace bude využita také v této aplikaci.

1.5 Backend

Backend v kontextu vývoje webových aplikací popisuje tu část softwaru, která pracuje na serveru a obstarává komunikaci a veškeré operace, které probíhají mezi frontendovou částí a databází. [19] Dalo by se říci, že se jedná o vše, co uživatel na webové stránce nevidí.

V mnoha případech se backend využívá ve formě API. V takovém případě vývojář definuje rozhraní, které vede k určeným funkcím. Z frontendové části je pak možné s tímto rozhraním komunikovat a získávat z něj požadovaná data. API je detailněji popsáno v kapitole 1.6.

Pro programovaní backendu je možné využít velké množství programovacích jazyků. Mezi nejpoužívanější patří jazyky PHP, Java, C#, Javascript nebo Ruby. Podle průzkumu společnosti W3Techs je nejrozšířenější právě jazyk PHP. Z průzkumu vychází, že jej využívá až 79,2% webových aplikací. Oproti tomu, průzkum mezi vývojáři z roku 2020 na webové stránce Stack Overflow ukazuje, že největší oblíbenosti se těší jazyk Javascript s 69,7%, Python s 41,6% a PHP se umístilo až na 3. místě se ziskem 25,8% hlasů. Nutno dodat, že v tomto průzkumu se ovšem nejedná o procentuální zastoupení mezi webovými aplikacemi, ale o výsledky hlasování mezi profesionálními vývojáři. [20]

Existuje hned několik důvodů, proč je jazyk PHP nejrozšířenější mezi webovými

aplikacemi. Hlavním důvodem je především široká podpora na serverech, které nabízejí provozovatelé hostingů. Dalšími výhodami je nízká křivka učebního potenciálu a velmi obsáhlá dokumentace, která detailně popisuje všechny aspekty jazyka. PHP lze také jednoduše kombinovat se značkovacím jazykem HTML, který je základním stavebním kamenem většiny webových stránek. Stačí pouze místo souboru s příponou .html použít příponu .php a html kód lze psát úplně stejně, s tím že PHP kód se vkládá mezi značky *<?php* a *?>*, které označují blok tohoto kódu.

Jazyk PHP se stále vyvíjí, opravují se chyby a přidávají nové funkcionality. Mezi programátory nemá příliš dobrou pověst kvůli časté nízké kvalitě výsledného kódu, jelikož jazyk v minulosti nevedl k používání správných programovacích postupů, neuměl pracovat s objekty a některé jeho části byly značně zastaralé. V dnešní době to tak již není, jelikož v nových verzích jazyka bylo provedeno velké množství úprav a vylepšení, které jazyk PHP posunuly na vyšší úroveň. Špatná reputace mezi programátory sice stále zůstává, ovšem v poslední době se situace zlepšuje.

1.6 API

API, neboli Application Programming Interface, je soubor procedur, funkcí, protokolů a knihoven, sloužící pro vývoj a integraci softwarových aplikací. API má přesně určeno, jakým způsobem se budou jednotlivé části kódu volat a co bude jejich výstupem. Programátor pak může tyto části použít bez toho, aby je musel sám programovat. [21] V případě webové aplikace slouží k tomu, aby s daným softwarem mohli ostatní služby a aplikaci komunikovat a získávat z něj data bez toho, aby znali jeho vnitřní implementaci. Externí aplikace získávají data z API pomocí tzv. endpointů (koncových bodů), které jsou v API definovány.

Existují různé druhy API, které slouží ke svým specifickým účelům. Mezi ty nejznámější patří:

- **REST API**
- **SOAP API**
- **GraphQL**

REST API slouží pro manipulaci se zdroji, jako je například získávání dat ze serveru, vytváření nových dat, nebo jejich editace. Veškerá komunikace probíhá s využitím protokolu HTTP (Hypertext Transfer Protocol)[16)].

[16)] Protokol HTTP byl definován standardem RFC 2616, dostupným z `https://www.rfc-editor.org/rfc/rfc2616`

HTTP je protokol na aplikační vrstvě, který slouží k přenosu dokumentů (například HTML) nebo libovolných binárních a textových dat. Byl navržen primárně pro komunikaci mezi klientem a serverem. Komunikaci začíná klient, který otevře spojení, odešle požadavek a poté čeká na odpověď. HTTP je bezstavový protokol, takže server neukládá žádná data mezi dvěma žádostmi. [22]

Požadavky, které klient zasílá, mohou být různého typu. Mezi nejčastěji používané požadavky patří:

- **POST** – používá se pro předání komplexních dat, například při vytváření nového záznamu v databázi.
- **GET** – používá se pro získání dat z databáze, může také nést hodnoty v url adrese.
- **PUT** – používá se podobně jako POST pro vytváření nových záznamů, ale nevytvoří duplikaci, pokud již záznam existuje.
- **DELETE** – používá se pro odstranění záznamu. [23]

SOAP API je protokol pro výměnu dat ve formátu XML. Pro komunikaci používá, stejně jako REST API, protokol HTTP a navíc ještě protokoly SMTP[17)] a TCP[18)]. GraphQL je dotazovací jazyk, který umožňuje získávat data ze serveru a oproti REST API umožňuje větší flexibilitu.

V dnešní době existuje obrovské množství různých API, které slouží pro různé účely. Od propojení se sociálními sítěmi, jako jsou třeba Facebook nebo Twitter, přes získávání dat z veřejných databází, až například po integrace platebních bran.

Příkladem může být například *Distance Matrix API* od společnosti Google. To umožňuje programátorovi využívat funkcionalitu Google map ve vlastním softwaru. API například nabízí funkci, která jako parametry přijímá adresu dvou (nebo více) míst na mapě a poté vrací délku nejkratší možné cesty mezi těmito místy. Na obrázku 1.7 je ukázán příklad volání tohoto API, které se skládá z úvodní URL adresy a několika parametrů. Parametr *destinations* označuje cílovou destinaci, parametr *origins* původní destinaci, *units* jednotku vzdálenosti a *key* uživatelův vygenerovaný API klíč, který je nutný pro používání API.

API nemusí být jen externí, jako výše zmíněné Distance matrix API, ale programátor si může vytvořit své vlastní. Častým případem je implementace API, které zastupuje

[17)] SMTP, neboli Simple Mail Transfer Protocol, je protokol, který slouží pro přenos elektronické pošty.

[18)] TCP, neboli Transmission Control Protocol, je nejpoužívanější protokol transportní vrstvy, používaný pro komunikaci mezi zařízeními.

```
https://maps.googleapis.com/maps/api/distancematrix/json
  ?destinations=New%20York%20City%2C%20NY
  &origins=Washington%2C%20DC
  &units=imperial
  &key=YOUR_API_KEY
```

Obrázek 1.7 Ukázka volání Distance matrix API (vlastní tvorba)

backendovou část aplikace a obsahuje koncové body pro veškerou její funkcionalitu. Na tyto body jsou poté zasílány požadavky z frontendové části, která má za úkol zobrazovat data. V tomto případě jsou obě části navzájem nezávislé a pouze si mezi sebou navzájem vyměňují data.

Hlavním úkolem API je tedy práce s daty, která jsou uložena v databázi. K tomu slouží čtyři hlavní operace, a to vytvoření záznamu (Create), editace (Update), čtení (Read) a mazání (Delete). Pro tyto operace se používá označení CRUD[19]. Díky jejich definici může programátor jednoduše provádět operace s daty, aniž by musel znát detaily implementace dané databáze.

Využívání API tedy může výrazně zjednodušit a urychlit vývoj softwarových aplikací, zejména pokud jsou používány kvalitní a dobře dokumentované API. Je ovšem důležité mít na paměti bezpečnost a ochranu soukromí. Proto musí být zajištěno, aby komunikace s API probíhala bezpečným způsobem a nebyla ohrožena bezpečnost uživatelů.

[19] CRUD je anglická zkratka slov Create, Read, Update a Delete

2 ROLOVÉ A SIMULAČNÍ HRY

V následující kapitole je popsána problematika rolových a simulačních her a jsou v ní shrnuty základní informace o těchto hrách. Tato kapitola se dále se věnuje způsobům využití těchto her ve výuce a jejich výhodám a nevýhodám. V dalších podkapitolách jsou popsány různé druhy těchto her a jejich výhody a nevýhody. V poslední části jsou rozebrána existující řešení a motivace k vývoji nové aplikace pro jejich tvorbu.

2.1 Co jsou rolové a simulační hry

Rolové a simulační hry (více známé pod anglickým názvem Role Playing Games, zkráceně RPGs) jsou aktivity, ve kterých hráči převezmou roli definovaných postav a pomocí své představivosti se přenesou do vytvořeného příběhu. Tento příběh se poté na základě hráčských rozhodnutí ubírá daným směrem. Hráči jsou mezi sebou nuceni komunikovat a diskutovat, aby byli schopni provádět správná rozhodnutí a úspěšně příběh dokončit. Hra obvykle obsahuje příběhovou linii, která se v různých bodech dělí na další, které se pak následně mohu, nebo nemusí, vrátit zpět do původní linie. Díky tomuto větvení může příběh obsahovat více různých zakončení, které se odvíjejí od toho, jak se hráč v průběhu hraní rozhodoval. Tyto hry je možné hrát i vícekrát, jelikož hráč může zvolit alternativní rozhodnutí a příběh tak bude jiný, než při předchozím hraní.

Častým jevem je také použití určitého stupně náhody, například použitím klasické šestistěnné kostky. Některá rozhodnutí jsou pak ovlivněna výsledkem hodu jednou nebo více kostkami, na jehož základě poté musí hráč upravit své rozhodování.

Nejvíce jsou rolové a simulační hry zastoupeny ve světě počítačových her, kde se jedná o jeden z nejvíce populárních žánrů. [24]

2.2 Využití ve výuce

První, kdo popularizoval využívání prvků her ve výuce nebyl nikdo jiný, než jeden z nejznámějších českých pedagogů a spisovatelů Jan Ámos Komenský, se kterým je spojováno známé sousloví *Škola hrou*. V dnešní době je využívání her stále více populární, používají se nejen ve výuce, ale například i v personalistice, managementu, obchodu a mnoha dalších odvětvích. Pro tento přístup vznikl zcela nový termín, který se nazývá *gamifikace*.

Gamifikaci lze popsat jako využívání herních prvků a mechanik v neherním prostředí. Historie tohoto termínu sahá až do roku 2008, kdy se poprvé objevil v oblasti digitálních médií, ale do širšího podvědomí se začal dostávat až ve druhé polovině roku 2010. [25]

Základním kamenem je myšlenka, že lidé hrají hry primárně pro zábavu a jsou při hraní více motivováni překonávat překážky, jelikož se jim dostává rychlé odměny.

Pokud se tedy herní prvky a mechaniky přesunou do neherního prostředí, mohlo by to proměnit tyto aktivity ve více zábavné a poutavé. [25]

Důvod pro využívání rolových a simulačních her je tedy především ten, že studenti nemají primárně pocit, že se jedná o výuku, ale že hrají hru. To jim přináší více požitků a motivuje je se stále zlepšovat. Zároveň jsou, v případě hraní hry ve skupince, nuceni do komunikace mezi sebou a k obhajování vlastních názorů a rozhodnutí. To rozvíjí jejich komunikační schopnosti a kritické myšlení.

Rolové a simulační hry se dají implementovat do výuky téměř libovolného předmětu. Ve školství jsou populární především v zahraničí, a to nejčastěji ve výuce cizích jazyků. Své využití ale najdou například i v různých vědomostních předmětech, jako jsou historie nebo biologie. V případě využití pro výuku cizího jazyka jsou hráči během hraní motivováni komunikovat v daném jazyce, čímž nenásilně rozvíjejí své jazykové schopnosti a zvyšují si sebevědomí v užití jazyka v reálné konverzaci.

2.3 Fyzické hry

Rolové a simulační hry jsou nejvíce rozšířené ve fyzické podobě, kde jsou veškeré texty související s příběhem sepsané do jednoho dokumentu, tzv. gamebooku (česky Herní knihy). Každá část hry je označena jednoznačným identifikátorem (nejčastěji číslem), podle kterého je vždy možné danou část dohledat. Mezi jednotlivými částmi se hráč pohybuje pomocí odkazů s těmito identifikátory, kdy u jednotlivých rozhodnutí v rámci dané části je uvedeno číslo, které odkazuje na následující část.

Jednou z nejznámějších rolových her je hra Dračí doupě (v originále Dungeons & dragons), která se poprvé objevila již v roce 1974 a od té doby si získala miliony fanoušků po celém světě. [26]

V takovéto podobě vznikají často i únikové hry, které jsou známé spíše ve formě reálných místností, kde se hráč pomocí řešení hádanek snaží dostat ven. V deskové podobě fungují na stejném principu, jako výše popsané rolové a simulační hry, jen s tím rozdílem, že pro průchod příběhem musí hráč vyřešit různé šifry a úkoly.

2.4 Digitální hry

Digitální rolové a simulační hry jsou v principu velmi podobné hrám fyzickým. Hlavním rozdílem je, že veškeré materiály jsou k dispozici v elektronické podobě. Díky tomu jsou přístupné hráčům kdekoliv, stačí mít po ruce libovolné zařízení, které umožňuje hru spustit. Tyto hry jsou tvořeny několika způsoby, které jsou, i s jejich výhodami a nevýhodami, popsány v následujících podkapitolách.

2.4.1 Microsoft PowerPoint a Google Slides

Jednou z možností, jak tvořit jednoduché digitální hry, jsou aplikace pro tvorbu prezentací PowerPoint od společnosti Microsoft a Slides od společnosti Google. Tyto aplikace samozřejmě primárně neslouží k tvorbě her, ale obsahují různé nástroje, které to umožňují. Jsou využívány především z toho důvodu, že většina vyučujících má s těmito aplikacemi bohaté zkušenosti z vytváření prezentací pro jejich výuku.

Obvykle se jedná o jednoduché příběhové hry, které se skládají z jednotlivých stránek, kde na každé stránce může být výběr z několika rozhodnutí. Tato rozhodnutí následně odkazují na některý z následujících stránek. Takovéto hry nemohou obsahovat žádné pokročilé mechaniky, nabízí se pouze varianta spojení s různými fyzickými prvky (například hrací kostka, ruční počítání atributů, atd.).

2.4.2 Webové stránky

Digitální hry je také možné vytvářet ve formě webových stránek. Takovéto hry jsou tvořeny tím způsobem, že celý průběh je napsán ve zdrojovém kódu, který napsal autor hry. Tato metoda není příliš využívána, jelikož hry nejsou tvořeny v žádném nástroji, ale přímo v kódu a autor hry tedy musí mít potřebné technické znalosti.

Výhodou těchto her je, že při jejich tvorbě neexistují téměř žádná omezení a autor si hru může upravit libovolně dle vlastního uvážení. Při pokročilé znalosti programovaní může také do hry přidat libovolné herní mechaniky.

Několik takových her je popsáno v kapitole 2.5

2.5 Existující řešení

Existují různé platformy, které se věnují využití rolových a simulačních her ve výuce a slouží k jejich vytváření. V této kapitole jsou některá tato řešení popsána a analyzována.

2.5.1 Agama

Agama je projekt, který vznikl za podpory Masarykovy Univerzity v Brně. Jedná se o platformu, která obsahuje celkem pět komplexních simulačních her na různá témata, které všechny cílí na využití pro výuku angličtiny s úrovní B1 a vyšší. [27]

Celá platforma se skládá z úvodního rozcestníku, kde jsou popsány informace o projektu, a ze samotných her. Ty jsou vytvořeny jako samostatné webové aplikace.

Například hry People v. Johnson a GrimField byly vytvořeny pomocí programovacího jazyka PHP (konkrétně framework Nette), JavaScript, HTML a CSS, a to v rámci bakalářské práce [28], jejímž autorem je Jan Mottl.

Všechny hry jsou zdarma a byly pečlivě otestovány na mnoha skupinách studentů. Díky tomu jsou na velmi vysoké úrovni. Nevýhodou této platformy je, že se nejedná o univerzální nástroj, ale hry jsou vytvořeny na míru.

2.5.2 Cypher system

Cypher system je platforma, která slouží jako předloha pro vytváření rolových a simulačních her. [29]

Jedná se o herní knihu, kterou lze zakoupit ve formátu PDF, nebo ve formátu tištěné knihy. Kniha poté obsahuje podrobný návod a podklady pro tvorbu hry. Uživatele provede jednotlivými kroky, kde vždy dostane na výběr z určitého počtu možností. Kniha také popisuje velké množství detailních pravidel, které uživatel do hry může, ale nemusí, zakomponovat.

Kniha je velmi obsáhlá, skládá se celkem ze 448 stránek, které jsou plné informací. Výhodou je, že žádná vytvořená hra nebude úplně stejná, jako jiná hra a také opravdu rozsáhlý výběr možností, díky kterým si uživatel může hru velmi přizpůsobit. Tato obsáhlost je na druhou stranu i nevýhodou, jelikož příprava hry zabere opravdu spoustu času. Už jen přečtení všech 448 stránek zabere minimálně několik desítek hodin. Další nevýhodou je určitá míra podobnosti mezi hrami, jelikož uživatel vždy vybírá ze stejné nabídky možností. Poslední nevýhodou je samozřejmě cena, která je poměrně vysoká.

Pokud má někdo zájem o vytvoření velmi komplexní hry, nevadí mu investovat velké množství svého času a chce si trochu ulehčit vymýšlení složitých herních mechanik, může pro něj být Cypher system dobrá volba.

2.5.3 RPG Playground

RPG creator je webová aplikace, která slouží k vytváření 2D grafických rolových her, jejímž autorem je Koen Witters. [30]

Aplikace se zaměřuje především na tvorbu klasických příběhových her, které nejsou určeny k výuce, ale dá se využít i na tyto hry. Obsahuje grafické rozhraní, které se skládá z plochy hry a nabídky nástrojů. Plochu hry představuje pouze travnatá plocha. Nabídka je tvořena z různých objektů, které je možné přesunovat na plochu a tím sestavovat mapu. Stejně tak lze přidávat postavy, ke kterým lze doplnit dialogy.

Rozhraní aplikace není příliš přehledné a intuitivní, veškeré nástroje a možnosti je nutné dohledat v různých částech nabídky. Samotné přidávání objektů je také tvořeno netradičním způsobem, kdy se kliknutím a následným tahem kurzoru označuje část nabídky a objekty, které se v oblasti nacházejí, se následně kliknutím přidají na plochu.

Cílová skupina této aplikace jsou spíše hráči počítačových her, kteří mají pokročilé technické schopnosti. Pro využití ve školním prostředí je příliš složitá a neintuitivní.

Aplikace je dostupná zdarma s možností zakoupení prémiové verze.

2.5.4 Genially

Dalším zástupcem ze skupiny webových aplikací je Genially. Tato aplikace slouží k mnoha účelům, jako je například tvorba prezentací, interaktivních obrázků, infografik a nebo právě pro vytváření jednoduchých her.

Aplikace obsahuje velké množství šablon, které umožňují tvořit různé kvízy, testy, interaktivní příběhy a únikové místnosti. Uživatel si pouze jednoduše vybere šablonu a upraví si text podle sebe. Všechny šablony jsou hezky graficky zpracované a některé lze použít ihned, i bez vlastních úprav.

Tvorba her funguje podobně jako tvorba prezentací s možností přidat různé interaktivní prvky. Základní prvky jsou k dispozici zdarma, pro využití ostatních je nutné zaplatit prémiový účet.

II. PRAKTICKÁ ČÁST

3 NÁVRH APLIKACE

Návrh aplikace je proces, který definuje softwarové metody, funkce a celkovou strukturu aplikace tak, aby vyhověla všem uživatelským požadavkům. Měl by obsahovat seznam uživatelských požadavků, popis architektury, databáze, veškeré použité API třetích stran a návrhy uživatelského rozhraní. [31]

Jedná se o důležitou část, která ovlivní celý další vývoj aplikace. Pokud je návrh proveden pečlivě, může programátorům značně ulehčit práci.

3.1 Požadavky na aplikaci

Jako první krok návrhu je nutné si definovat, jaké jsou na aplikaci kladeny požadavky. Existují dvě hlavní skupiny požadavků. První skupinou jsou takzvané funkcionální požadavky.

Tyto požadavky definují, co vše musí aplikace umět a jaké služby má poskytovat. Jako příklad můžeme uvést například aplikaci pro správu objednávek. Taková aplikace by měla umět vytvořit, upravit a smazat objednávku a také jí odeslat. Toto vše patří mezi funkcionální požadavky.

Druhou skupinou jsou nefunkcionální požadavky. Ty nesouvisejí přímo s funkcionalitami aplikace, ale zahrnují její vlastnosti. Mezi nefunkcionální požadavky může patřit například rychlost odezvy systému, úroveň zabezpečení nebo výkon.

Primárním požadavkem na aplikace bude možnost vytvářet rolové a simulační hry a následně je hrát. Mezi další funkcionality bude patřit kompletní správa her, jejich prohlížení a správa uživatelských skupin (tříd). Požadavky na aplikaci jsou detailněji sepsány v následujících kapitolách.

3.1.1 Funkcionální požadavky

Funkcionální požadavky jsou sepsány ve formě seznamu, který je rozdělen do několika logických celků.

1. Uživatel
    - Vytvoření účtu
    - Přihlášení do vlastního účtu
    - Zobrazení informací o účtu
    - Editace osobních údajů
    - Změna hesla
    - Zobrazení vlastních her

- Zobrazení veřejně dostupných her
- Hodnocení hry

2. Tvorba hry
    - Možnost vytvořit hru, zvolit její název a popis
    - Možnost přidat ke hře heslo nutné pro hraní hry
    - Možnost přidat ke hře obrázek
    - Možnost přidělit hře atributy (např. zdraví, síla)
    - Možnost přidělit hře předměty (např. papír, provaz)
    - Možnost definovat postavy, které budou moci hráči zvolit
    - Možnost vytvářet, editovat a mazat jednotlivé scény hry
    - Možnost vložit obsah scény
    - Možnost vložení odkazu na další scény, formou:
        - výběru ze seznamu rozhodnutí
        - výběru místa na obrázku
        - pouze přechodu na další scénu
    - Možnost zvolení úvodní scény
    - Možnost přidání nápověd ke scéně
    - Možnost přidání doplňkových dokumentů ke hře
    - Možnost sdílet hru pomocím odkazu a QR kódu
    - Možnost smazat hru
3. Hra
    - Možnost spustit a hrát hru
    - Možnost výběru postavy před zahájením hry
4. Třídy
    - Možnost zobrazit seznam tříd
    - Možnost vytvořit třídu
    - Možnost editovat třídu
    - Možnost přidat uživatele do třídy
    - Možnost připojit se do třídy
    - Možnost přidat hru do třídy

3.1.2 Nefunkcionální požadavky

- Povolení přístupu k editaci her a scén pouze pokud je uživatel autorem hry
- Zamezení přístupu do aplikace bez přihlášení
- Pokud má třída nastavené heslo, lze se do ní přidat pouze po jeho zadání
- Povolení přístupu do administrátorské sekce pouze s rolí admin
- Dostatečný výkon serveru, aby aplikace běžela plynule

Na základě požadavků byly vytvořeny dva UML diagramy případů užití. Na obrázku 3.1 lze vidět diagram pro aplikaci, bez editoru příběhu. Pro ten byl z důvodu přehlednosti vytvořen druhý diagram, který lze vidět na obrázku 3.2. Tyto diagramy popisují interakce uživatelů se systémem. Skládají se z aktérů (používá se anglické označení Actors), kteří představují jednotlivé typy uživatelů a z akcí, které mohou vykonávat.

Na prvním diagramu lze vidět tři typy uživatelů. Nepřihlášený uživatel má nejvíce omezené možnosti. Kromě vytvoření účtu a přihlášení může pouze spustit a hrát hru. To je nastaveno z toho důvodu, aby si studenti kvůli jednomu zahrání hry nemuseli zbytečně vytvářet účet, což by zpomalovalo proces začátku hry. Pokud by chtěli získat vstup do zbylé části aplikace, vytvoří si účet a stane se z nich Přihlášený uživatel. Ten již získává přístup prakticky do všech částí systému. Je u umožněno vytvářet a spravovat hry a třídy, prohlížet si všechny veřejné hry a ovládat svůj uživatelský účet.

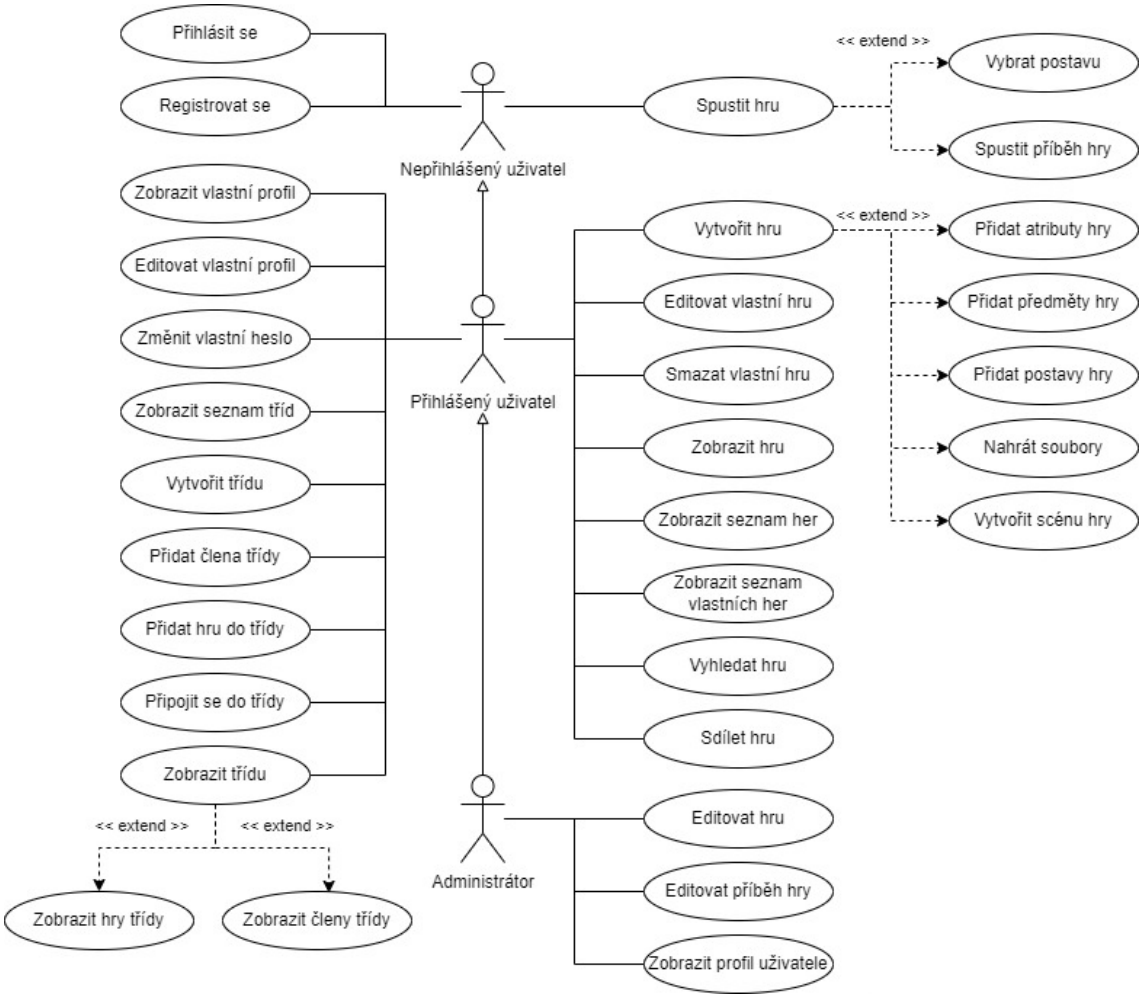
Nad těmito typy uživatelů stojí pouze administrátor aplikace. Ten, na rozdíl od běžných uživatelů, může upravovat veškeré hry, třídy a uživatele, bez ohledu na to, kdo je vytvořil.

Druhý diagram obsahuje případy užití pro editor scén. V tom mají všichni uživatelé stejné možnosti. Mohou přidávat scény, vyplňovat jejích obsah a provádět změny nastavení.

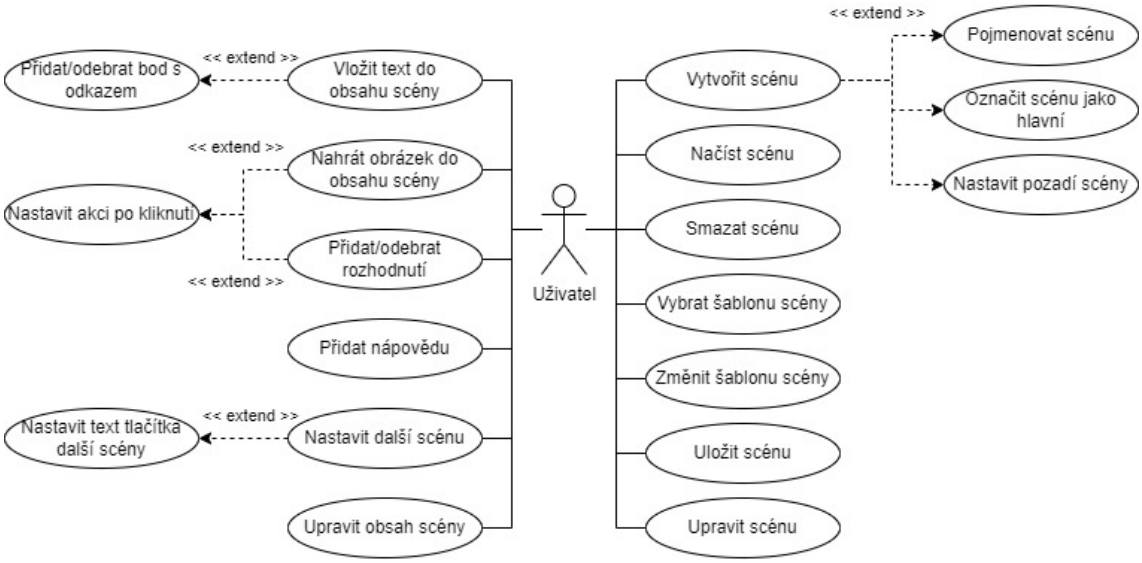
3.2 Architektura

Zvolení správné architektury je klíčovým prvkem pro efektivní vývoj aplikace. Na základě vybraných technologií byla pro tvorbu aplikace zvolena architektura klient-server, kde server představuje API a klient představuje frontend aplikace. Schéma této architektury lze vidět na obrázku 3.3.

Aplikace se tedy bude skládat ze tří hlavních částí. Těmi jsou klient (frontend), API (backend) a databáze. Tyto tři části mezi sebou navzájem komunikují. Klient sestaví požadavek a zašle ho na jeden z definovaných API endpointů. Ten požadavek zpracuje,

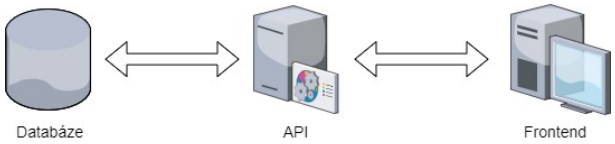


Obrázek 3.1 Diagram případů užití (vlastní tvorba, vytvořeno službou `diagrams.net`)



Obrázek 3.2 Diagram případů užití editoru příběhu (vlastní tvorba, vytvořeno službou `diagrams.net`)

získá požadovaná data z databáze a odešle je jako odpověď zpět klientovi, který je poté vhodně zobrazí uživateli.



Obrázek 3.3 Schéma architektury klient-server (vlastní tvorba, vytvořeno službou `diagrams.net`)

Způsob, jakým jsou data ukládána, je popsán v následující kapitole 3.3.

3.3 Ukládání dat

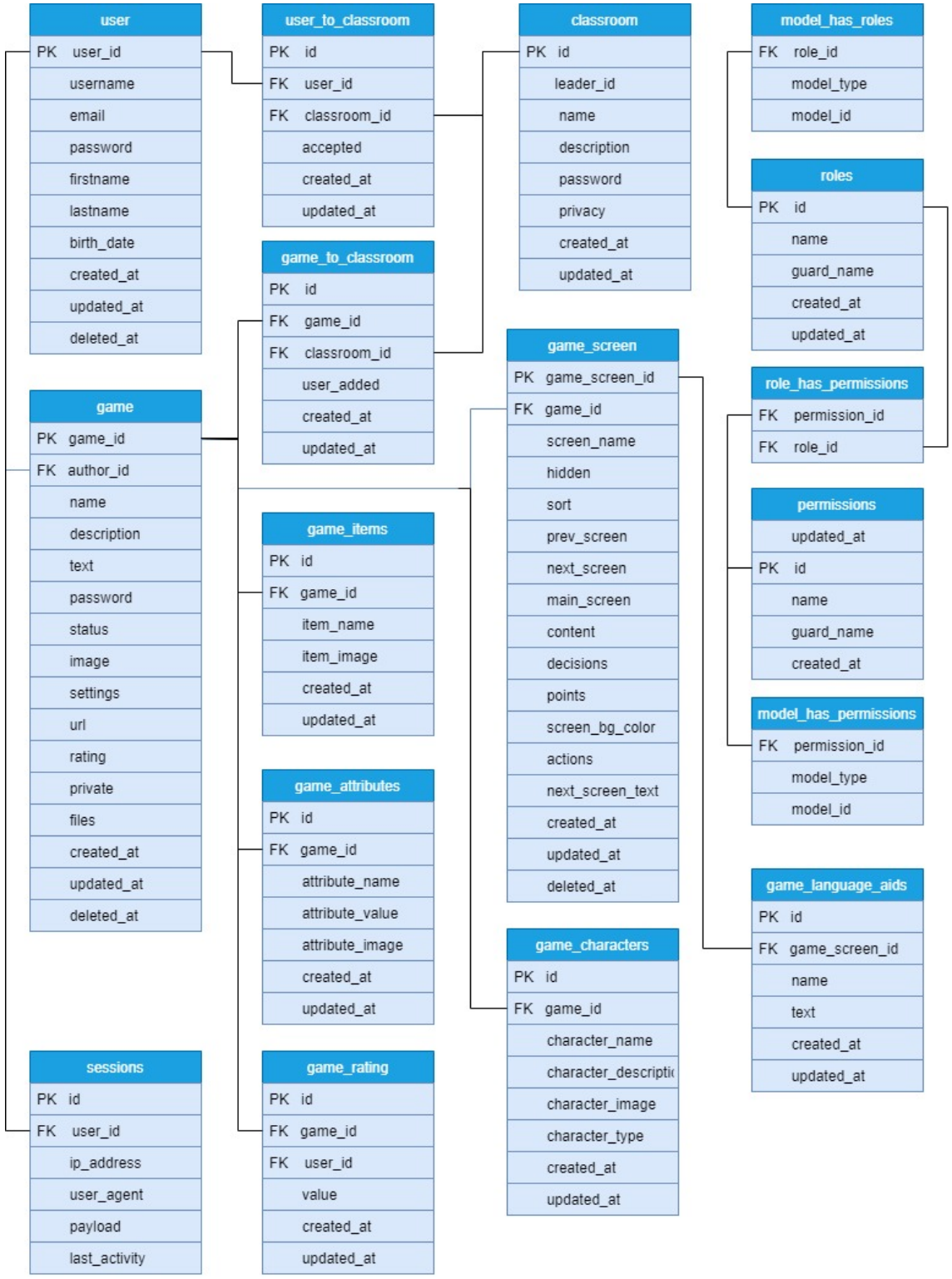
Pro ukládání dat byla zvolena objektově relační databáze PostgreSQL, která je detailněji popsána v kapitole 4.3. Pro každou entitu je vytvořena samostatná tabulka v databázi. Každá tabulka má svůj primární klíč, který slouží jako jednoznačný identifikátor jednotlivých záznamů v dané tabulce.

Nejdůležitější částí databáze jsou tabulky pro uložení dat o hře. Hlavním bodem je tabulka *game*. Ta obsahuje základní informace o hře, jako je název hry, popisek, úvodní obrázek, atd. Na tuto tabulku je navázáno několik dalších. Pro ukládání dat o atributech slouží tabulka *game_attributes*, pro předměty *game_items* a pro postavy *game_characters*. Hodnocení hry se nachází v tabulce *game_rating*. V tabulce *game_screen* se nachází jednotlivé scény hry, na které jsou dále napojeny jazykové nápovědy, které náleží tabulce *game_language_aids*. Tyto tabulky jsou navázané pouze pomocí cizího klíče, kterým je id hry, jelikož vždy patří pouze k jedné konkrétní hře.

Další částí jsou tabulky pro třídy. Hlavní tabulkou je v tomto případě tabulka *classroom*, která obsahuje základní informace o třídě. Ke třídě můžou náležet různé hry a uživatelé. Pro oba tyto případy jsou vytvořeny vazební tabulky, konkrétně *game_to_classroom* a *user_to_classroom*, které tvoří propojení mezi tabulkou pro třídu a tabulkami pro hry a uživatele. Vazební tabulky se v tomto případě používají z toho důvodu, že hra i uživatel mohou náležet do více tříd, takže již není možné tyto tabulky spojit pouze pomocí id.

Mezi zbylé tabulky patří tabulka *user*, která ukládá data o uživateli, tabulka *sessions*, která ukládá informace o aktuálním sezení uživatele a tabulky *roles*, *permissions*, *model_has_roles*, *model_has_permissions* a *role_has_permissions*, které byly vytvořeny modulem Laravel Fortify a slouží pro správu uživatelských rolí a oprávnění.

Schéma této navržené databáze lze vidět na obrázku 3.4.



Obrázek 3.4 Schéma navržené databáze (vlastní tvorba)

3.4 Uživatelské rozhraní

Tato kapitola se zaměřuje na návrh vhodného uživatelského rozhraní. Skládá se ze tří částí. V první části jsou popsány základní informace o způsobu návrhu rozhraní. Ve

druhé části jsou definované persóny, které reprezentují typické uživatele aplikace. Třetí část obsahuje konkrétní návrhy rozhraní a jejich popis.

3.4.1 Přístup k návrhu uživatelského rozhraní

U aplikace, která má sloužit primárně pro využití ve školství a výuce, je potřebné věnovat velkou pozornost návrhu uživatelského rozhraní. Je nutné brát v úvahu, že mezi cílovými uživateli se budou nacházet osoby s velkou škálou technických znalostí, od úplně minimálních, až po profesionální. Veškeré základní funkcionality aplikace je tedy nutné koncipovat tak, aby i uživatel s minimálními technickými znalostmi byl schopen aplikaci používat. Pro pokročilé uživatele pak aplikace může obsahovat další možnosti, které ale nejsou nezbytné pro základní používání.

Při vytváření uživatelského rozhraní je tedy dobré myslet na několik věcí. Mezi ty patří:

1. **Jednoduchost** – uživatelské prostředí by mělo být co nejjednodušší, bez zbytečných elementů, které uživatele ruší.
2. **Konzistentnost** – elementy uživatelského rozhraní by měli být konzistentní v rámci celé aplikace.
3. **Přehlednost** – prvky aplikace by měli být logicky rozmístěné a viditelné.
4. **Barevnost** – při výběru barev je důležité dbát na jejich vzájemné souznění a viditelnost na různých zařízeních.
5. **Typografie** – jednotlivé texty by měli být hierarchicky odděleny, aby vedli uživatele při procházení webu.
6. **Zpětná vazba** – aplikace by měla na veškeré uživatelské akce vrátit relevantní odpověď, například v podobě oznámení, aby uživatel vždy věděl, co se děje. [32]

Jak již bylo zmíněno v kapitole 1.4.1, moderní aplikace jsou obvykle založeny na responzivním designu. I přes to, že v dnešní době je stále častější využívat místo počítače mobilní telefon, v případě této aplikace je kladen větší důraz na design zobrazení na zařízeních s větším rozlišením, jako jsou právě notebooky nebo počítače, jelikož na těchto zařízeních bude aplikace primárně využívána. Veškerá funkcionalita půjde samozřejmě využívat i na mobilních telefonech, s výjimkou editoru scén hry, který má na malé obrazovce omezené možnosti.

3.4.2 Persóny

Pro návrh uživatelského rozhraní je důležité si definovat, kdo jsou uživatelé, kteří budou aplikaci používat. Pro vykreslení typů předpokládaných uživatelů se tvoří tzv. persóny (používá se anglické označení Personas). Mulder [33] popisuje persónu jako nákres realistické postavy, která reprezentuje jeden segment cílového publika webové stránky.

Využití persón přispívá k lepšímu návrhu, jelikož je kladen vyšší důraz na potřeby předpokládaných typů uživatelů.

Před definicí konkrétních persón je důležité analyzovat demografické rozložení možných uživatelů aplikace. Z tohoto rozdělení lze vyvodit následující závěry:

1. Uživatele lze rozdělit na dvě primární kategorie:
   - **Žáci a studenti** – tato kategorie bude využívat převážně tu část aplikace, která slouží k hraní hry.
   - **Učitelé** – tato kategorie bude využívat převážně tvorbu her
2. Tyto skupiny uživatelů lze rozdělit dále podle věku do několika kategorií, které představují jednotlivé generace.
   - **Baby boomers** (česky Generační populační exploze, narození v letech 1946-1964) – tuto generaci často charakterizuje konzervativnost a proto její zástupci mohou mít větší odpor k novým technologiím. Většinou nejsou příliš zkušení ve využívání počítačů a aplikací pro výuku. Pro tuto skupinu je důležité, aby aplikace měla jednoduché a intuitivní ovládání a bylo snadné naučit se ji používat.
   - **Generace X** (narození v letech 1965-1980) – aktuálně nejvíce zastoupená generace mezi učiteli. Tato generace se narodila v době, kdy se počítače začaly stávat běžně dostupnými a mají tedy o něco větší zkušenosti s technologiemi než Baby Boomers.
   - **Mileniálové** (narození v letech 1981-1996) – tato generace se narodila v době, kdy se internet stal široce dostupným. Díky tomu jsou nejvíce otevření novým technologiím a metodám. Aktuálně je mezi učiteli zastoupena nejvíce.
   - **Generace Z** (narození v letech 1997-2012) – tuto skupinu zastupují především žáci a studenti. Ti se narodili v plně digitální době a díky tomu mají zpravidla velké zkušenosti s využíváním počítačů, aplikací a také s hraním her.

Dle statistik společnosti OECD, která sestavila statistiku rozdělení učitelů dle věku, se nejvíce učitelů nachází ve věkové skupině 30 až 50 let, a to až 51,8%. Ve věkové skupině nad 50 let se nachází 44,9% učitelů a pouze 6,9% učitelů je mladší 30ti let. [34]

Při návrhu uživatelského rozhraní budeme předpokládat, že i když skupina učitelů pod 30 let má nejmenší zastoupení, bude tvořit většinu uživatelů aplikace, jelikož tato skupina je nejvíce otevřená zkoušení nových metod a technologií. Druhou nejvíce zastoupenou skupinou budou učitelé ve věku 30 až 50 let a nejméně poslední skupina, od 50 let výše.

Pro účely návrhu této aplikace budou tedy vytvořeny celkem čtyři persóny. První tři budou zástupci jednotlivých věkových skupin učitelů a čtvrtá bude představovat studenta. Tyto persóny jsou popsány v tabulkách 3.1 pro první dvě persóny a 3.2 pro zbylé dvě persóny.

Tabulka 3.1 Základní informace o persónách č.1 a 2

Jméno a příjmení	**Jana Nováková**	**Karel Novotný**
Věk	41	29
Povolání	Učitelka ZŠ	Učitel SŠ
Technické znalosti	Průměrné	Vysoké

Při návrhu prvků uživatelského rozhraní je stále nutné mít tyto persóny na mysli a pokusit se vžít do jejich role. U každého prvku je potřeba rozhodnout, zda je dostatečně intuitivní a slouží k tomu, co od něj uživatelé na první pohled očekávají. Nejdůležitější bude zaměřit se na editor her, jelikož ten bude obsahovat nejvíce funkcionalit.

V následujících kapitolách je popsáno uživatelské rozhraní jednotlivých částí aplikace s vysvětlením jeho vlastností a funkcionalit.

3.4.3 Hlavní prvky uživatelského rozhraní

Mezi základní operace v aplikaci patří vytvoření a obsluha uživatelského účtu. Ihned po první návštěvě úvodní stránky aplikace by měl být uživatel schopen identifikovat,

Tabulka 3.2 Základní informace o persónách č.3 a 4

Jméno a příjmení	**Josef Veselý**	**Anna Dvořáková**
Věk	52	16
Povolání	Učitel Gymnázia	Studentka SŠ
Technické znalosti	Nízké	Vysoké

jak přejít k používání aplikace. Tuto funkci zastává velké tlačítko s textem „Chci začít!“ (V anglické verzi aplikace „Start creating!“), které uživatele přesune na stránku s vytvořením účtu. Pokud už uživatel svůj účet má, může se jednoduše jedním kliknutím přesunout na stránku s přihlášením. Po přihlášení je uživatel automaticky přesunut přímo do aplikace na úvodní přehledovou stránku. Na té se nachází jednoduché přivítání a dva seznamy her. První seznam obsahuje nejnovější hry a druhý nejlépe hodnocené hry.

Druhou částí je samotná tvorba hry. Veškeré prvky uživatelského rozhraní v aplikaci by měli být koncipovány tak, aby nejvíce na očích byl prvek pro tvorbu nové hry. Tlačítko "Vytvořit hru"se tedy nachází hned na několika místech, a to v horní částí navigační nabídky a na stránce s přehledem her. Zároveň je kontrastně odlišeno od ostatních prvků, aby přitahovalo pozornost.

Začátek tvorby hry probíhá formou průvodce, který uživatele provede několika kroky. Prvním krokem je zvolení názvu hry, krátkého popisu, informačního textu, obrázku a případně hesla. Také dostane na výběr, zda má být hra veřejně dostupná, či nikoliv. Druhý krok obsahuje možnost nahrát ke hře pomocné a doplňkové soubory. Dalším krokem jen volba atributů hry (například zdraví, síla) a předmětů hry (například papír, provaz, meč). U obou je vždy možnost zvolit název a volitelně přidat obrázek. U atributů je navíc volba základní hodnoty. Posledním krokem je poté volba postav, za které bude možné v příběhu hrát.

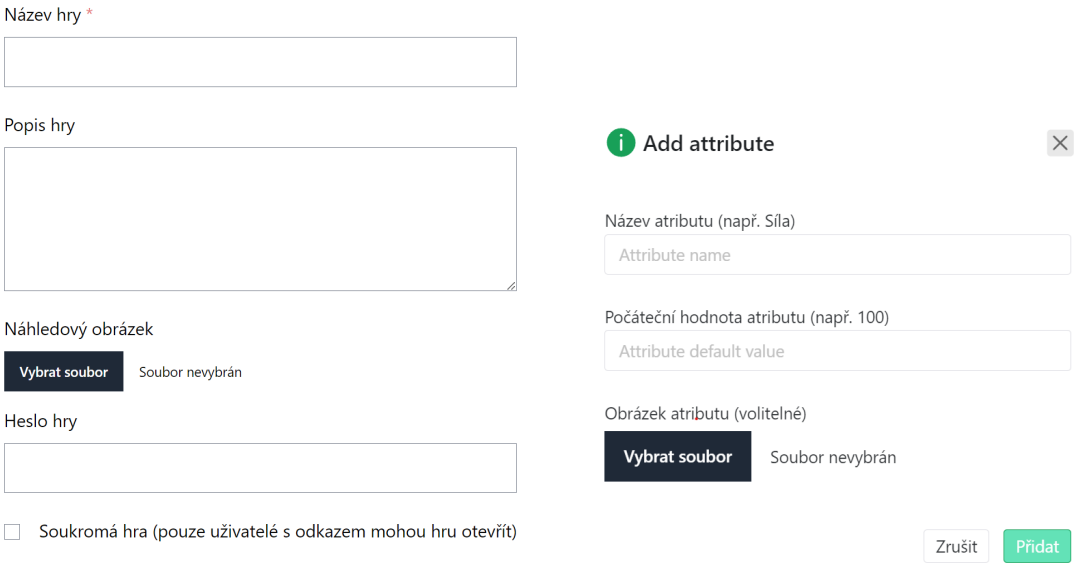
Všechna tato nastavení probíhají pomocí jednoduchých formulářů a dialogových oken, které obsahují minimum polí. Přílišná náročnost základního průvodce by mohla uživatelům znepříjemnit uživatelský zážitek a některé i odradit od dalšího používání aplikace. [35] Příklad formulářů pro definici základních informací o hře a přidání nového atributu lze vidět na obrázku 3.5

Po dokončení průvodce se uživateli otevře editor scén. Zde může vytvářet jednotlivé scény, tvořit jejich obsah a pomocí definic rozhodnutí, míst na obrázcích a odkazů tvořit postupně celý příběh.

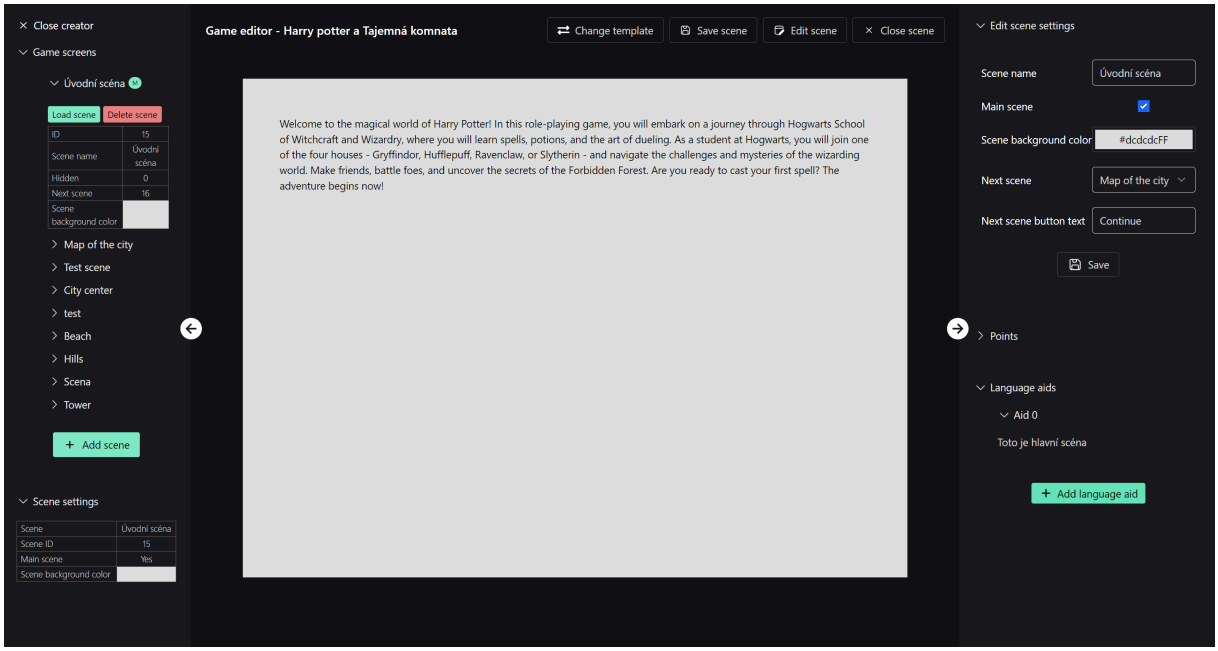
U všech prvků uživatelského rozhraní jsou doplněny popisky, které uživatele informují, co přesně který prvek dělá.

Vytvoření nové scény probíhá pomocí jednoduchého dialogového okna. Uživatel je vyzván, aby zvolil název scény, barvu pozadí scény a zda jde o hlavní scénu. Poté se nová scéna otevře a uživatel dostane na výběr z několika přednastavených šablon. Kliknutím na jednu z nich se daná šablona zvolí a spustí se možnost úpravy obsahu. Obsah se vkládá taktéž pomocí vhodný prvků. Pro text je přítomen jednoduchý textový editor typu WYSIWYG[1)], pro obrázky jednoduché okno, které po kliknutí vyzve

[1)] WYSIWYG (What you see is what you get, česky Co vidíš, to dostaneš) je typ textového editoru,



Obrázek 3.5 Ukázka formulářů pro základní nastavení hry a přidání atributu (vlastní obrázky)



Obrázek 3.6 Ukázka editoru příběhu hry (vlastní tvorba)

uživatele k výběru souboru. U šablony s výběrem možností je možné definovat pomocím dialogového okna libovolný počet rozhodnutí. Při tvorbě rozhodnutí je uživatel vyzván dialogovým oknem, aby zvolil text, který se u rozhodnutí objeví a také na jakou další scénu rozhodnutí vede.

U obrázku je možnost přidat neomezený počet bodů, které poté, podobně jako rozhodnutí, odkazují na další scény. Po kliknutí na libovolné místo na obrázku je uživatel

kde výsledný text vypadá přesně tak, jak ho viděl autor při jeho psaní v editoru.

vyzván k vyplnění textu, barvy bodu a následující scéně. Bod se poté objeví na místě, kde uživatel klikl. Takto je možné přidat neomezené množství bodů.

Informace o vytvořených scénách lze najít ve dvou postranních panelech. V levé části editoru se nachází panel, který obsahuje seznam scén a základní informace o všech scénách. V pravé části se poté nachází možnost editovat nastavení aktuálně zvolené scény a v případě obrázkové scény také seznam všech bodů, které se na obrázku nacházejí.

3.4.4 Seznam her

Seznam vytvořených her se zobrazuje také na několika místech v aplikaci. Hlavním místem je samostatná stránka, která zobrazuje veškeré hry, které se v aplikaci nachází, rozdělené do dvou sekcí. První sekce, "Mé hry", zobrazuje hry vytvořené aktuálně přihlášeným uživatelem. V druhé sekci, "Všechny hry", se zobrazují hry od všech uživatelů, které nebyly označeny jako soukromé. Hry jsou vypsány v podobě kartiček na řádcích vedle sebe. Každá kartička obsahuje jméno hry, autora a po najetí kurzorem myši i krátký popisek. Jako pozadí kartičky je použit obrázek dané hry.

3.4.5 Detail hry

Po kliknutí na hru ze seznamu se otevře její samostatná strana s detailními informacemi. Mezi ty patří název hry, její popis, text, autor, hodnocení a obrázek. V pravé části obrazovky se nachází tlačítka pro interakci se hrou. Hlavním tlačítkem je "Hrát hru", které vidí všichni uživatelé. Pokud je uživatel zároveň autorem hry, vidí navíc tlačítka "Editor hry", "Editor příběhu"a "Odstranit hru".

Ze stránky detailu hry je také možné hru sdílet. Pro sdílení slouží odkaz vedle jména hry, který otevře modální okno s možnostmi. Hlavní možností je zkopírovat url adresu hry. Druhou možností je QR kód, který se po otevření okna automaticky vygeneroval. Pro generování QR kódu bylo využito bezplatné API služby QR Code Generator. To umožňuje zaslat jednoduchý požadavek s parametrem, který obsahuje text pro zakódování a API vrátí vygenerovaný kód ve formátu obrázku.

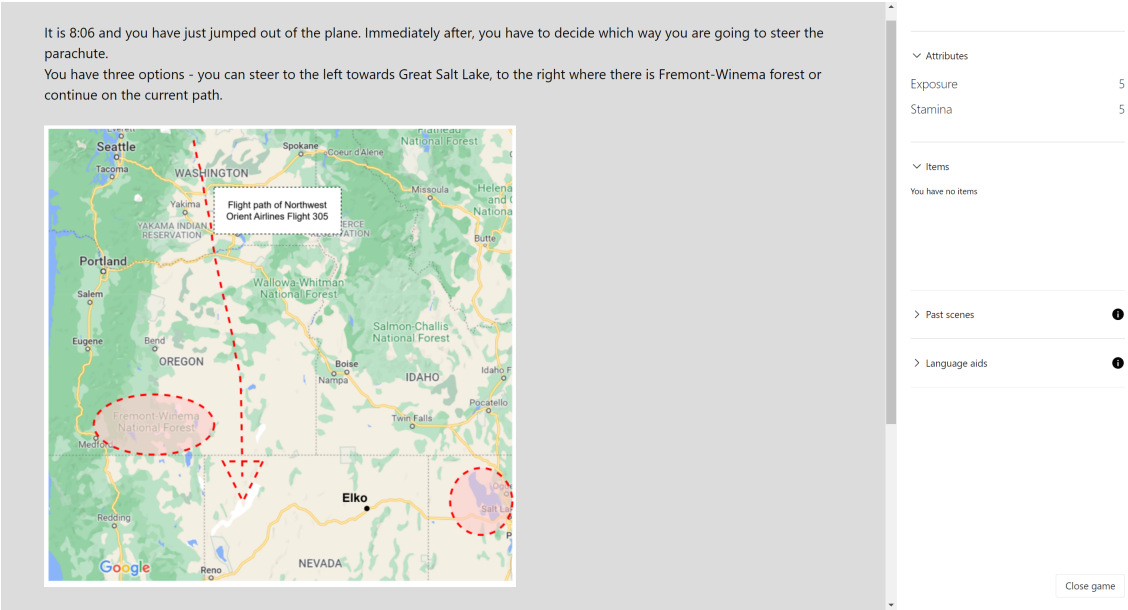
3.4.6 Hraní hry

Uživatelské rozhraní samotného hraní hry musí být navrženo především co nejpřehledněji, bez zbytečných rušivých prvků. Po spuštění hry se jako první objeví přehledová stránka s informacemi o hře a tlačítkem pro její spuštění.

V případě, že hra obsahuje nějaké postavy, dostane v dalším kroku uživatel možnost jejich výběru. Všechny postavy se objeví ve formě karet s obrázkem, názvem a popiskem.

Poté se spustí příběh hry. Okno je rozděleno na dvě části. V levé části se nachází

obsah scény, který lze vertikálně posouvat, pokud se obsah scény nevejde do zorného pole. Druhou částí je postranní panel. Ten obsahuje veškeré statistiky hry, jako jsou atributy, předměty, postava, atd. Ty jsou přehledně zobrazeny pod sebou. U jednotlivých částí panelu je možnost skrytí jejich obsahu pomocí rozbalovací roletky. Ukázku uživatelského rozhraní spuštěné hry lze vidět na obrázku 3.7.



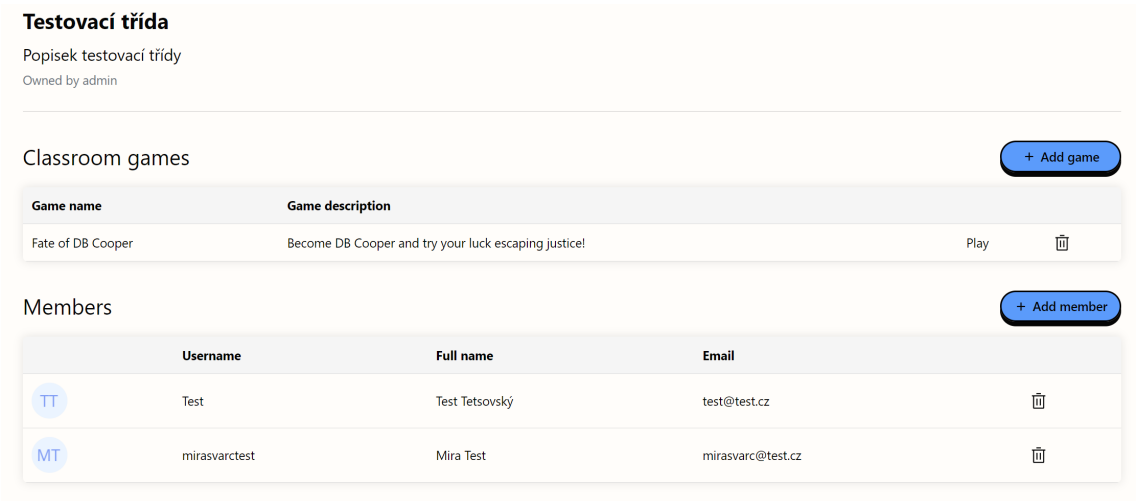
Obrázek 3.7 Ukázka uživatelského rozhraní spuštěné hry (vlastní tvorba)

3.4.7 Třídy

Přehled tříd tvoří dva seznamy. Jeden obsahuje všechny třídy, jejichž členem je aktuálně přihlášený uživatel a druhý všechny existující třídy. Oba seznamy jsou vykresleny formou jednoduchých tabulek.

Vytvoření třídy probíhá pomocí formuláře, kde uživatel vyplní jméno třídy, popisek a také má možnost zvolit, zda je třída veřejná, nebo soukromá a případně zvolit heslo pro připojení.

Detail třídy je tvořen také dvěma seznamy. První obsahuje seznam her, které byly ke třídě přiděleny. Druhý pak obsahuje informace o členech třídy. Ukázku detailu třídy s přidanou hrou a dvěma členy lze vidět na obrázku 3.8.



Obrázek 3.8 Ukázka stránky s detailem třídy (vlastní tvorba)

4 POUŽITÉ TECHNOLOGIE

V této kapitole jsou popsány veškeré technologie, které byly použity pro implementaci této aplikace. Ty byly voleny v závislosti na požadavcích na výslednou aplikaci s ohledem na možnosti jejich vzájemného propojení, rychlosti, stabilitě a bezpečnosti.

4.1 Laravel

Pro backendovou část byl zvolen jazyk PHP[1)] verze 7.4.29 a jeho framework Laravel[2)] verze 8. Laravel je open-source framework[3)], který slouží pro kompletní tvorbu webových aplikací. Dá se použít dvěma způsoby:

1. jako monolitová architektura zahrnující jak backend, tak frontend,
2. nebo jako backend API.

V rámci této práce byl využit pouze jako API rozhraní pro zpracovávání požadavků z frontendové části aplikace. Samostatná frontendová část je popsána v kapitole 4.2.

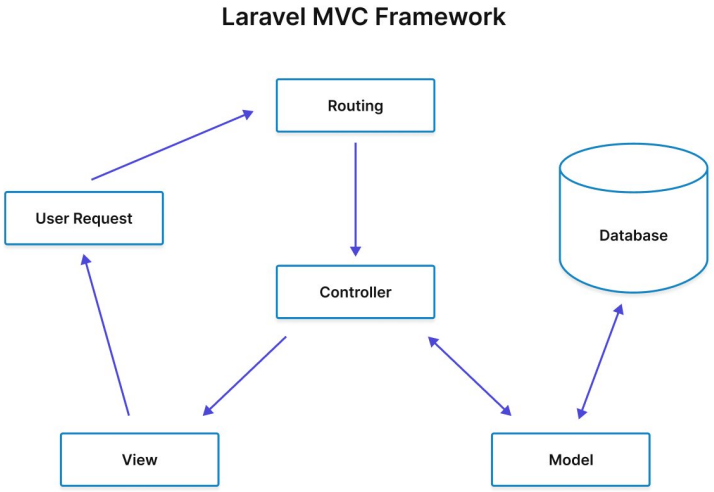
Laravel byl zvolen na základě svých vlastností, které zahrnují jednoduchou práci s daty díky vysokoúrovňovému databázovému mapování, pokročilému zabezpečení, stabilitě a výkonu. Také má poměrně širokou uživatelskou základnu a díky tomu je k dispozici velké množství materiálů a informací.

4.1.1 Architektura MVC

Laravel využívá architekturu MVC (Model-View-Controller). To znamená, že aplikace je rozdělena na 3 samostatné části, které spolu navzájem komunikují. Těmito částmi jsou:

1. **Modely** (Models) – představují jednotlivé entity a jejich logiku. Každý z nich je spojen s konkrétní tabulkou v databázi, se kterou komunikuje. Na druhé straně komunikuje s kontrolérem.
2. **Kontroléry** (Controllers) – zpracovávají požadavky přijaté z pohledů. Na základě nich získávají data z příslušných modelů a odesílají je zpět pohledům.
3. **Pohledy** (Views) – prezentují data uživateli. Jedná se o část, se kterou uživatel interaguje. [37]

[1)] https://www.php.net/
[2)] https://laravel.com/
[3)] Framework je sada nástrojů, knihoven, konvencí a osvědčených postupů, které vytváří abstrakci nad rutinními úkoly do obecných modulů, které mohou být jednoduše znovu využity [36]



Obrázek 4.1 Ukázka architektury MVC u frameworku Laravel (převzato z [38])

Důležitou součástí je také směrování. To funguje na principu definice jednotlivých cest. V Laravelu se tyto cesty nachází v souboru *api/routes.php* (v případě využití jako API), nebo *web/routes.php* (v případě využití jako samostatné aplikace). Pro každou cestu je definována url adresa, název cesty a funkce, která má zpracovat přijatý požadavek. Dále lze cesty spojovat do skupin podle libovolného prefixu, přiřazovat k nim pravidla nebo rovnou definovat tělo funkce, která se má provést. U každé cesty je také nutné zvolit, jaký typ HTTP požadavku má očekávat.

Na obrázku 4.2 lze vidět definici cest, které jsou spojeny prefixem *game* a mají nastavený middleware *auth:sanctum*, který kontroluje, zda požadavek zasílá uživatel, který je autentifikován. Uvnitř skupiny jsou definovány tři cesty. První cesta má nastavenou očekávanou metodu POST a slouží pro vytvoření hry. Druhá cesta slouží pro editaci hry a třetí, s nastaveným požadavkem GET, slouží pro zobrazení hry. V hranatých závorkách je vždy definována funkce, na kterou je přijatý požadavek přesměrován. V případě požadavku GET obsahuje adresa cesty parametr *id* v zapsaný v hranatých závorkách. Ten značí identifikátor hry, který je zaslán přímo v url adrese.

```
Route::middleware('auth:sanctum')->prefix('game')->group(function () {
    Route::post('/create', [GameController::class, 'createGame']);
    Route::post('/update', [GameController::class, 'updateGame']);
    Route::get('/{id}', [GameController::class, 'getGame']);
});
```

Obrázek 4.2 Ukázka definice cest ve frameworku Laravel (vlastní tvorba)

4.1.2 Komunikace s databází

Pro komunikaci s databází je v Laravelu možné využít hned několik způsobů. V rámci této práce bylo pro operace s databází využito objektově-relační mapování Eloquent.

Při použití tohoto přístupu má každá tabulka v databázi přiřazen příslušný model, který ji zastupuje. Programátor pak interaguje pouze s tímto modelem, který umožňuje provádět veškeré operace, které by se jinak musely sestavovat pomocí jazyka SQL. Porovnání kódů pro získání seznamu všech her s databáze oběma způsoby můžeme vidět na obrázku 4.3

```
// ukázka získání seznamu her pomocí Eloquent
$games = Game::all();

// ukázka získání seznamu her pomocí SQL
"SELECT * FROM 'games' where 1"
```

Obrázek 4.3 Porovnání získání seznamu her z databáze pomocí Eloquent a SQL (vlastní tvorba)

Kromě získávání a zápisu dat do databáze umožňuje Eloquent také definovat relace. To znamená, že jednotlivé modely se pospojují podle svých vzájemných vztahů. Poté je možné jednoduše získávat data, která k sobě navzájem patří. Například pro získání všech scén, které obsahuje daná hra, stačí definovat, že hra a scéna hry mají vztah 1:N, tedy že k jedné hře náleží právě N scén. Příklad definice a získání dat lze vidět na obrázku 4.4, kde v horní části lze vidět definici modelu Game pro hru a následně vztahu se scénami pomocí funkce *hasMany*. Ta označuje, že k jedné hře náleží více scén. Zavoláním funkce *scenes()* lze poté jednoduše získat seznam scén pro danou hru.

4.2 Nuxt3

Pro frontendovou část byl zvolen framework Nuxt verze 3. Jedná se o framework jazyka JavaScript, který je postavený na knihovně Vue3 a umožňuje jednoduše tvořit webové aplikace. Oproti knihovně Vue je Nuxt rychlejší, efektivnější a umožňuje hybridní vykreslování, tedy zároveň vykreslování na straně serveru a na straně klienta.[39]

Další výhodou je funkcionalita přednačtení odkazů. To znamená že při načtení stránky Nuxt automaticky vyhledá všechny odkazy, které jsou v kódu vytvořeny pomocí značky *<NuxtLink>* (obrázek 4.5), místo klasické html značky *<a>*, a obsah cíle těchto odkazů si uloží do mezipaměti. Pokud poté uživatel klikne na nějaký takový odkaz, zobrazí se stránka okamžitě bez načítání, jelikož už byla načtená předem.

Oproti Vue3 také značně zjednodušuje směrování v aplikaci. Oproti klasickému směrovacímu souboru používá směrování na základě stromové struktury adresářů a zdrojových souborů. Veškeré cesty jsou tedy automaticky vygenerované na základě souborové

```php
// Definice vztahu mezi hrou a snímkami
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Game extends Model
{
    /**
     * Get the scenes for the game.
     */
    public function scenes(): HasMany
    {
        return $this->hasMany(GameScene::class);
    }
}


// získání všech snímků dané hry
$game_scenes = $game->scenes();
```

Obrázek 4.4 Ukázka definice vztahu mezi hrou a snímkami hry a následné získání snímků z konkrétní hry (vlastní tvorba)

struktury v adresáři Pages. Stejně tak lze vytvářet i dynamické cesty a to pomocí vložení hranatých závorek do názvu souboru.

Zjednodušené je i využívání dílčích souborů, komponent a knihoven, jelikož Nuxt používá automatické importování. Není tedy nutné soubory importovat ručně, ale jsou automaticky, dle potřeby, načteny při sestavování aplikace.

4.2.1 Composables

Pro často se opakující funkce, jako jsou například různá volání funkcí backendového API, se v aplikaci využívají tzv. Composables.

Composables pochází z knihovny Vue3, odkud je převzal framework Nuxt3. Jedná se o funkce, které jsou odděleny do samostatných souborů, aby mohly být jendoduše znovu využity. Díky tomu nemusí v kódu docházet ke zbytečným duplikacím.

Pro každou funkci je vytvořen jeden soubor. Ten je, jako ostatní prvky frameworku Nuxt3, automaticky importován a funkci tak lze použít kdekoliv v kódu. Všechny tyto soubory se nachází ve složce *composables*, která je v kořenovém adresáři aplikace.

Jednotlivé soubory je možné ukládat i do dalších podsložek. V takové případě se již neimportují automaticky, ale je nutné ve složce *composables* vytvořit soubor *index.ts* a v něm definovat export funkcí ze souborů, které nejsou v hlavní složce. Tato vlastnost má jediné využití, a to přehlednost, pokud se v projektu nachází velké množství

```
<template>
  <header>
    <nav>
      <ul>
        <li><NuxtLink to="/about">About</NuxtLink></li>
        <li><NuxtLink to="/posts/1">Post 1</NuxtLink></li>
        <li><NuxtLink to="/posts/2">Post 2</NuxtLink></li>
      </ul>
    </nav>
  </header>
</template>
```

Obrázek 4.5 Ukázka generování odkazů pro Nuxt (vlastní tvorba)

takových souborů.

Využívání Composable funkcí také přispívá k větší přehlednosti kódu a snižuje velikost jednotlivých komponent.

Příkladem využití může být například composable getGame.js. Ta obsahuje funkci *getGame()*, která má za úkol zaslat na API požadavek pro získání dat konkrétní hry na základě jejího identifikátoru. Tato funkce se používá v mnoha různých částech aplikace, jako jsou například zobrazení detailu hry, editace hry nebo samotné spuštění hry.

4.2.2 Composition vs Option API

Composition API a Option API jsou dva přístupy, které slouží k psaní Javascriptového kódu do komponent ve frameworku Nuxt3. Obě jsou ekvivalentní, je tedy pouze na programátorovy, který způsob zvolí. Je možné i oba způsoby kombinovat a to i v rámci jedné komponenty.

Ve starším Option API je daná struktura, kterou musí programátor využívat. Oproti tomu novější Composition API strukuturu nedodržuje a je tak možné psát kód stejně, jako v klasickém Javascriptu. Zároveň se snaží řešit některé nedostatky Option API, jako je například obtížná správa stavu a logika větších komponent. Composition API nabízí více flexibilní způsob práce s daty a funkcemi, díky čemuž lze snadněji psát čistý a srozumitelný kód. [40]

V Composition API lze také využít jazyk TypeScript. To je staticky typovaný jazyk, který umožňuje zlepšit kontrolu kódu a jeho bezpečnost. V kombinaci s Composition API lze vytvořit kód s vyšší úrovní abstrakce a přehlednosti.

Oba tyto přístupy jsou převzaty z knihovny Vue3. Nuxt směřuje spíše k využití Composition API, ale je možné využít obě dvě varianty.

Oba přístupy mají své výhody i nevýhody a volba tak závisí pouze na preferenci programátora.

4.3 Databáze

Pro ukládání dat byla zvolena databáze PostgreSQL. Jedná se o bezplatnou, otevřenou a výkonnou objektově-relační databázi, která je vyvíjena již od roku 1996. [41] Podpora PostgreSQL v Laravelu je nativní a není potřeba nic měnit v konfiguraci frameworku. Tato integrace umožňuje snadné vytváření, čtení, aktualizaci a mazání dat v PostgreSQL databázi.

Oproti MySQL má několik výhod. První z nich je nativní podpora bohatých datových formátů, jako jsou například JSON[4)] a XML. PostgreSQL je také rychlejší při provádění komplexních požadavků a práci s velkými objemy dat.

Hlavním důvodem jeho použití byla jednoduchá práce s datovým typem JSON, který je v aplikaci využíván například pro ukládání nastavení a obsahu scény hry.

4.4 JSON

Pro výměnu dat mezi frontendovou a backendovou částí a také pro ukládání některých dat v databázi slouží formát JSON. Jedná se o jednoduchý formát, který je lehce čitelný pro uživatele a zároveň i pro počítače. JSON byl definován v roce 1999 ve standardu ECMA-262 3. edice[5)] právě pro použití k výměně dat mezi platformami. [42] Na obrázku 4.6 lze vidět zkrácenou ukázku dat o hře ve formátu JSON.

```
{
  id: 1,
  name: "Testovací hra",
  author_id: 2,
  settings: {
    type: 1,
    private: 0,
  }
}
```

Obrázek 4.6 Ukázka dat ve formátu JSON
(vlastní tvorba)

[4)] JSON je podporován i v MySQL databázi, ale PostgreSQL navíc podporuje i binární verzi JSONB

[5)] Nejnovější vydání lze nalézt zde: `https://www.ecma-international.org/publications-and-standards/standards/ecma-404/`

4.5 Programy a služby použité pro vývoj

Pro vývoj aplikace bylo použito několik programů a služeb, které jsou popsány v následující kapitole.

4.5.1 Visual Studio Code

Visual Studio Code, zkráceně VS Code, je editor zdrojového kódu vyvíjený společností Microsoft. Jedná se o univerzální nástroj, který se dá použít pro vývoj v libovolném programovacím jazyce. Do editoru lze nainstalovat velké množství rozšíření, jak oficiálních tak komunitních, které jej rozšiřují o další funkcionality.

Umožňuje také vysokou míru personalizace, právě díky velkému množství rozšířeních, takže si každý může editor kompletně upravit podle sebe. Aplikace VS Code je naprogramována pomocí jazyka JavaScript s knihovnou Electron a je vyvíjena jako open-source.

4.5.2 PgAdmin4

PgAdmin4 je aplikace, která slouží pro připojení k databázovému serveru PostgreSQL. Obsahuje přehled všech vytvořených databází a umožňuje je jednoduše spravovat, bez nutnosti použití příkazové řádky.

Umožňuje kompletní obsluhu databází včetně pokročilých funkcionalit. Je možné jednoduše vytvářet schémata, tabulky, triggery, funkce a další.

Jedná se primárně o desktopovou aplikaci, ale je možné použít i verzi v internetovém prohlížeči. Ta se používá především na serverech, kde nelze spustit software s grafickým rozhraním.

4.5.3 Git a GitHub

Pro verzování zdrojového kódu aplikace byla využita služba Git. Ta uchovává jednotlivé verze kódu, nazývané Commits (zanesené změny), které jsou označeny unikátním identifikátorem. Do uložených verzí lze libovolně nahlížet a zobrazovat úpravy mezi jednotlivými změnami. Také umožňuje jednoduší spolupráci, v případě že na projektu pracuje v jednu chvíli více programátorů.

Pro uložení repozitáře byla použita služba GitHub, která slouží jako hosting pro git repozitáře. Kromě služby hostingu obsahuje mnoho dalších funkcionalit, jako jsou služby postupného nasazování, hostování statických stránek, správu projektů a další.

4.5.4 Windows Terminal

Windows terminal je příkazová řádka operačního systému Windows, která spojuje funkcionality původní příkazové řádky cmd a PowerShellu. Také podporuje instalaci Unixových systémů pomocí virtualizace.

Tato příkazová řádka byla využívána k vývoji aplikace, a to konkrétně ke spouštění vývojové verze aplikace a pro komunikaci se službou GitHub.

4.5.5 Postman

Pro testování API požadavků byla využita aplikace Postman. Ten slouží jako API klient, kde může uživatel zadat adresu svého API, sestavit HTTP požadavek a odeslat ho na danou adresu. Postman poté zobrazí odpověď, kterou server vrátil. Díky tomu je možné otestovat API ještě před implementací klientské části a odhalit tak zásadní chyby. U vrácené odpovědi Postman automaticky rozpozná její formát a korektně ji naformátuje. [43]

Všechny sestavené požadavky si aplikace pamatuje, takže je možné se k nim libovolně vracet a znovu je používat.

Kromě toho obsahuje mnoho dalších funkcionalit, jako například automatické testování, generování dokumentace, nebo monitorování API.

5 IMPLEMENTACE NAVRŽENÉHO ŘEŠENÍ

V této kapitole je popsána implementace navrženého řešení. Kapitola 5.1 popisuje implementaci API ve frameworku Laravel. Následující kapitola 5.2 se zabývá zabezpečením aplikace a to jak frontendové části, tak API. Je v ní popsána autentifikace uživatele, přístupy k zabezpečení komunikace a také nejčastější útoky a ochrana proti nim. Kapitola 5.3 popisuje implementaci frontendové části pomocí knihovny Nuxt3. Dále je popsáno využití knihovny Pinia pro uchovávání dat v rámci frontendové části, kterému se věnuje kapitola 5.4 a jazyková lokalizace, které se věnuje kapitola 5.5.

5.1 API

Pro implementaci API byl použit framework Laravel, který byl popsán v kapitole 4.1. API se skládá z několika dvojic modelů a kontrolérů. Každá z těchto dvojic obsluhuje jinou část aplikace a představuje konkrétní entitu. Kontroléry se skládají z funkcí, které přijímají požadavek z frontendové části, následně provedou zpracování požadavku a vracejí odpověď ve formátu JSON.

Hlavním kontrolérem je *GameController.php*, který obsahuje funkce pro správu hry a jejího obsahu. Pro atributy, předměty, postavy a jazykové nápovědy ve hře jsou použity vlastní kontroléry, *GameAttributeController.php*, *GameItemController.php*, *GameCharacterController.php* a *GameLanguageAidController.php*, a to převážně z důvodu přehlednosti a zachování čistotu kódu.

Pro správu uživatelů je používán *UserController.php* a pro autentifikaci *AuthController.php*.

Správa tříd je obsluhována kontrolérem *ClassroomController.php*, který obsahuje funkcionality nejen pro vytváření, editaci a mazání tříd, ale také pro přidávání uživatelů a her.

Všechny tyto kontroléry mají odpovídající modely, které představují dané entity. Modely odpovídají základem jménu, tedy například pro *GameController.php* existuje model *Game.php*.

V modelu je definován název tabulky v databázi a primární klíč. Pokud jsou tyto hodnoty identické, tedy pro model *Game* existuje tabulka *game* s primárním klíčem *id*, není potřeba je nastavovat. V aplikaci jsou nastaveny ve všech modelech z důvodu přehlednosti.

Dále se v modelech definují relace, tedy jejich vzájemné vztahy. Příkladem může být definice vztahu mezi třídou a uživatelem, která definuje, že uživatel může patřit do libovolného počtu tříd. Definici tohoto vztahu lze vidět na obrázku 5.1. Díky tomuto vztahu lze pak jednoduše zjistit, do jakých tříd uživatel patří a to voláním funkce

classrooms() na entitu *User*, jak lze vidět na obrázku 5.2.

```php
public function classrooms() {
    return $this->belongsToMany(Classroom::class, 'user_to_classroom', 'user_id', 'classroom_id');
}
```

Obrázek 5.1 Ukázka definice relace mezi uživatelem a třídou (vlastní tvorba)

```php
$user = User::find($id);
$user_classrooms = $user->classrooms();
```

Obrázek 5.2 Ukázka získání seznamu tříd, do který patří uživatel, pomocí Eloquent relací (vlastní tvorba)

Důležitou částí API jsou cesty (pozn. používá se označení *routy* z anglického překladu *routes*). Ty určují, které kontroléry a funkce budou obsluhovat přijaté požadavky na zadaných adresách.

Celý průběh funkcionality API je tedy následující: API přijme požadavek a podívá se, na jakou url adresu byl zaslán. Podle této adresy vyhledá v souboru *api/routes.php* příslušný záznam a podívá, která funkce je pro tuto adresu nastavena a v jakém se nachází kontroléru. Data požadavku jsou poté zaslány na danou funkci, která provede definované operace. Pokud je nutné komunikovat s databází, odkazuje se dále na definovaný model. Poté sestaví odpověď ve formátu JSON a odešle ji zpět do klientské části.

Důležitou funkcionalitou, kterou zajišťuje backendová část, je autentifikace uživatele, která je popsána v samostatné kapitole 5.2.1.

5.2 Zabezpečení aplikace

Při implementaci webové aplikace, která bude využívána veřejností, je nutné dbát na to, aby byla co nejlépe zabezpečená. To lze řešit mnoha způsoby, od řízení přístupů k jednotlivým částem, které je důležité pro běžné uživatele, až po obranu proti známým typům útoků, jako jsou SQL injekce, nebo XSS útoky. Ty jsou popsány v následujících kapitolách.

V kapitole 5.2.1 je popsán přístup k autentifikaci uživatele a následné ověřování zasílaných API požadavků. V následující kapitole 5.2.2 jsou popsány ochrany přístupů podle typu uživatele. V kapitolách 5.2.3 a 5.2.4 jsou popsány nejběžnější útoky na webové stránky, a to SQL injekce a XSS. Poslední kapitola 5.2.5 se věnuje popisu knihovny Nuxt/Security, která slouží k zabezpečení frontendové části aplikace.

5.2.1 Autentifikace uživatele

Hlavní část autentifikace se odehrává na straně serveru. Zde je použit modul Laravel Sanctum, který poskytuje funkcionality nutné pro autentifikaci uživatele. Pro každého uživatele je vygenerován jednoznačný identifikátor (tzv. token), který poté uživatel používá pro ověření požadavků na server. Ke každému požadavku, který je zaslán z klienta, je tento token přiložen. Po přijetí požadavku server token porovná s tokenem uloženým na serveru a pokud se oba tokeny shodují, vrátí server odpověď.

Knihovna Sanctum je navíc rozšířena o startovací balíček Laravel Jetstream, který obsahuje mnoho dalších funkcionalit. Mezi ty hlavní patří pomocné služby, jako je registrace účtu, aktivace účtu po registraci pomocí emailu, vygenerování nového hesla a podpora implementace dvou fázového ověření. Doplňkovou funkcí je také například automatické vygenerování obrázku profilu z prvních písmen jména a příjmení uživatele, které jsou napsány v barevném poli. Zároveň obsahuje uživatelské rozhraní pro správu účtu, to je ovšem dostupné pouze při použití Laravelu jako samostatného frameworku a ne pouze ve formě API, jako v případě této aplikace.

Na straně klienta je pro autentifikaci použito vlastní řešení jelikož v době implementace této aplikace neexistuje oficiální modul pro autentifikace pro framework Nuxt verze 3[1)].

Po vyplnění přihlašovacího formuláře uživatelem je sestaven požadavek, který je následně zaslán do API. Tam je klient autentifikován pomocí knihovny Sanctum, jak je popsáno výše. Token, který API vrátí v odpovědi klientovy, je uložen do cookie a uživatel je přesměrován do aplikace.

Veškeré požadavky, které poté klient posílá do API, musí obsahovat tento token. Ten je vždy přiložen do hlavičky požadavku. Ukázku takového požadavku lze vidět na obrázku 5.3, kde proměnná *api_url.value* obsahuje url adresu API a *user_token* obsahuje vygenerovaný token, který je přiložen jako hlavička Authorization s klíčovým slovem Bearer, které označuje typ autorizace.

Pokud by token v požadavku nebyl zaslán, vrátil by server odpověď 401 Unauthorized, která označuje požadavek jako neověřený.

Pro ověření, zda je uživatel autentifikován, slouží v klientské části funkce *getAuthUser*. Ta zašle požadavek na cestu */api/me*, která jako odpověď vrací informace o aktuálně přihlášeném uživateli. Pokud odpověď tohoto požadavku obsahuje objekt s daty daného uživatele, je korektně autentifikován. V opačném případě je vrácena odpověď 401 Unauthorized.

[1)] Ke dni psaní této práce zatím není oficiální modul ve stádiu vývoje, plán je dostupný na adrese `https://nuxt.com/docs/community/roadmap#core-modules`

```
const response = await $fetch(api_url.value+'/api/me', {
    method: 'GET',
    headers: {
        'Content-Type': 'application/json',
        'Accept': 'application/json',
        'Authorization': `Bearer ${user_token}`,
    }
})
```

Obrázek 5.3 Ukázka odeslání požadavku na API (vlastní tvorba)

5.2.2 Middlewares

Hlavní částí zabezpečení aplikace je autentifikace uživatele, která byla popsána v kapitole 5.2.1. Díky té je do aplikace povolen přístup pouze přihlášeným uživatelům. Zároveň je díky ní zabezpečeno posílání požadavků na backend API, jelikož všechny zaslané požadavky musí obsahovat token přihlášeného uživatele. Bez tohoto tokenu se požadavek neprovede a API vrátí uživateli zprávu s chybovým kódem. Kotrolu tokenu provádí autentifikační knihovna Sanctum. Ta obsahuje mezivrstvy (používá se anglický název middlewares), které obsahují definovaná pravidla, která určují, co se má stát v případě přijetí různých požadavků. Těchto mezivrstev může programátor definovat libovolné množství. Ty se poté nastaví k jednotlivým cestám ve směrovači. Pokud je z klienta zaslán požadavek na některou z definovaných cest, Laravel se podívá, zda má cesta definovanou nějakou mezivrstvu a pokud ano, provede její kód.

Hlavní mezivrstva, kterou knihovna Sanctum používá, je *sanctum:auth*. Ta kontroluje, zda byl požadavek zaslán přihlášeným uživatelem a tedy obsahuje jeho token. Pokud ano a token je správný, požadavek je proveden.

Mezivrstva *sanctum:auth* je definována pro všechny cesty s výjímkou cest pro registraci, přihlášení a zobrazení úvodní stránky a jejích podstran. To je z důvodu, že uživatel, který na stránku přijde, není přihlášen, nebo ještě nemá vytvořený účet a tak mu nelze na základě toho omezovat přístup k těmto cestám.

Další mezivrstvy lze použít například pro ošetření přístupů podle uživatelských rolí a jejich pravomocí.

Své mezivrstvy obsahuje i knihovna Nuxt3, která je použita pro implementaci frontendové části aplikace. Ty fungují úplně stejně, jako v případě Laravelu, jen nekontrolují přístupy ke koncovým bodům API, ale hlídají cesty k jednotlivým stránkám aplikace.

5.2.3 SQL injections

SQL injection (v češtině SQL injekce) je druh útoku, kdy útočník vloží do vstupu na webové stránce kód v jazyce SQL, který provede často škodlivé změny v databázi.

Ochranu proti tomuto útoku zajišťuje objektově-relační mapování Eloquent (detailněji popsáno v kapitole 4.1.2), které Laravel používá pro sestavování databázových dotazů. Ten všechny dotazy parametrizuje a používá SQL vazby. Díky tomu nejsou prováděny nezpracované dotazy, ale jsou sestaveny pomocí funkcí, které očistí vstupy uživatele od všech částí, které by v nich být neměli.

5.2.4 XSS útoky

XSS, neboli Cross Site Scripting, je útok, který spočívá ve vložení škodlivého skriptu (například v jazyce JavaScript) do webové stránky. To se může stát například pokud by uživatel takový kód vložil do pole formuláře, který se poté někde na webu vypisuje.

Pro ochranu před XSS útoky je použita knihovna Nuxt/Security, která je popsána v následující kapitole 5.2.5. Tak provede kontrolu každého HTTP požadavku typu GET nebo POST. Pokud obsah nebo parametry takového požadavku obsahuje nebezpečný kód, je navrácena odpověď *400 Bad Request*.

5.2.5 Knihovna Nuxt/Security

Pro zabezpečení frontendové části aplikaci, která je implementována pomocí frameworku Nuxt3, byla využita knihovna Nuxt/Security [44]. Ta byla vytvořena pro jednoduché nastavení a správu všech běžných bezpečnostních prvků.

Knihovna zahrnuje tyto funkcionality:

- Nastavení bezpečnostních hlaviček
- Zásady zabezpečení obsahu (Content security policy)
- Omezení velikosti zasílaných požadavků a nastavení limitů
- Validace XSS
- Podpora CORS[2)]
- Správu povolených HTTP metod
- Základní autentifikaci a CSRF[3)] [45]

Po instalaci knihovny je použito výchozí nastavení. To je možné změnit definicí vlastních pravidel v konfiguračním souboru *nuxt.config.ts*.

[2)] Cross-origin resource sharing, lze přeložit jako sdílení zdrojů mezi různými doménami

[3)] CSRF (Cross site request forgery) je typ útoku, který nutí uživatele odeslat nechtěnou žádost do webové aplikace, ve které je aktuálně přihlášen

Po instalaci knihovny je výchozí nastavení obvykle dostačující. Nicméně, v některých situacích může být vhodné upravit parametry knihovny, například pokud je nutné explicitně povolit nebo zakázat určité zdroje nebo HTTP metody. V případě této aplikace byl upraven parametr *contentSecurityPolicy*, který specifikuje, jaké zdroje jsou povoleny pro načítání souborů se skripty a CSS styly.

Bezpečnostní hlavičky a další bezpečnostní prvky jsou nezbytnou součástí moderních webových aplikací a jsou klíčové pro zajištění bezpečnosti uživatelů. Proto je důležité mít tyto prvky správně nakonfigurované a udržovat je aktuální s nejnovějšími bezpečnostními standardy a praktikami.

5.3 Frontend

Frontendová část je implementována pomocí frameworku Nuxt3, jak již bylo zmíněno v kapitole 1.4. Jednotlivé části aplikace jsou definovány ve složce *Pages* v souborech s příponou *.vue*. Ty představují stránky, ze který Nuxt automaticky tvoří cesty. Stránky jsou dále tvořeny z komponent, které se nacházejí ve složce *components*. Ty se používají z důvodu znovupoužitelnosti a přehlednosti.

Další důležitou částí je složka *composables*. Ta obsahuje soubory s příponami *.js* a *.ts*, které představují jednotlivé funkce, které se v kódu používají vícekrát. Tyto funkce, stejně jako i komponenty, si Nuxt importuje automaticky všude tam, kde jsou potřeba a vývojář se tím nemusí zabývat.

Frontendová část je složena ze dvou částí. První částí je Landing page, neboli úvodní stránka. Druhou částí je samotná aplikace, která se skládá z editoru hry, editoru příběhu hry, přehledu aplikace a samotné hry.

5.3.1 Úvodní stránka

Úvodní stránka, mezi vývojáři známá jako landing page, je stránka, na kterou se uživatel dostane při prvním zadání url adresy aplikace, nebo pokud ji vyhledá přes internetový vyhledávač. Obsahuje informace o aplikaci, jako je název a stručný popis. Slouží k tomu, aby uživatele nalákala k použití aplikace a aby mu dala přehled o tom, o jakou aplikaci jde a k čemu se dá použít.

Landing page je koncipována jako jednoduchá přehledová stránka, která uživatele navede k vytvoření účtu a přihlášení do aplikace. Ukázku této stránky lze vidět na obrázku 5.4.

Kromě úvodní stránky je vytvořeno ještě několik podstran. Podstrana *O aplikaci* obsahuje podrobnější informace o tom, k čemu aplikace slouží. Na podstraně *Jak na to* lze nalézt návod na její používání. Poslední podstranou je *Kontakt*, kde se nachází kontaktní údaje na autora.

Obrázek 5.4 Vzhled úvodní stránky webové aplikace (vlastní tvorba)

5.3.2 Editor hry

Editor hry je jednoduchý průvodce vytvořením hry, na který se uživatel dostane po kliknutí na tlačítko *Vytvořit hru* v aplikaci. Skládá se z několika kroků, které byly sestaveny pomocí komponenty *n-steps* z knihovny NaiveUI[4]. Ty postupně provedou uživatele základním nastavením hry. První krok obsahuje informace o hře, tedy název, popisek, obrázek, zda je hra veřejná nebo soukromá a případně heslo pro přístup.

V dalším kroku je uživatel vyzván k definici atributů, které budou ve hře použity. Může jich vybrat libovolné množství pomocí dynamického formuláře a u každého zvolit jeho název, obrázek a počáteční hodnotu. Atributy určují libovolné hodnoty hry, lze je například využít pro počítání životů, času nebo síly. Ihned po přidání nového atributu je uložen v databázi do samostatné tabulky *game_ attributes*.

Následující krok obsahuje definici předmětů do hry. Tento krok je téměř identický s předchozím krokem, pouze se u předmětů neuvádí jejich počáteční hodnota. Stejně jako atributy jsou i předměty uloženy do databáze do samostatné tabulky *game_ items*.

Posledním krokem je výběr postav, které budou ve hře k dispozici. Při výběru každé postavy dostane uživatel na výběr ze dvou možností. První možnost je vytvořit vlastní postavu, tedy vymyslet pro ni jméno, popis a nahrát obrázek. Druhou možností je vybrat jednu z předem připravených postav. Takto může uživatel přidat do hry libovolný počet postav, z kterých si pak hráč na začátku hry může vybrat.

Po dokončení průvodce je uživatel vyzván k přechodu do další části aplikace, kterou je editor příběhu.

[4] NaiveUI je knihovna komponent pro Vue3, dostupná na `https://www.naiveui.com/`

5.3.3 Editor příběhu

Editor příběhu slouží k vytváření jednotlivých scén hry, ze kterých je poté tvořen celý příběh. Uživatel může vytvořit libovolný počet scén, které se navzájem propojí a tím vznikne stromová struktura scén.

Po kliknutí na tlačítko Vytvořit scénu se uživateli otevře modální okno, pro které je využita komponenta *n-modal* z knihovny NaiveUI. V tomto okně je uživatel vyzván k zadání základních informací o scéně. Těmi je název scény, barva pozadí scény a zda se jedná o hlavní scénu. Po vytvoření scény se uživateli zobrazí editace jejího obsahu. Zde má možnost vybrat z několika šablon scény a to ze šablon Text, Obrázek a Text + rozhodnutí.

Šablona Text obsahuje textový editor. Pro ten byla využita knihovna Quill[5)], která podporuje jednoduché formátování textu a vkládání multimedálních prvků.

Kromě toho obsahuje možnost definovat, jaká scéna bude následovat. Ta je potom napojena na tlačítko, které se zobrazí pod textem. U tohoto tlačítka může uživatel v nastavení scény zvolit, jaký text bude obsahovat.

Quill používá vlastní datový formát Delta. Jedná se o podmnožinu formátu JSON, který je jednoduše čitelný jak pro člověka, tak pro počítač. Pro jednoduchost zobrazení ve scéně se obsah textového pole ukládá do databáze nejen ve formátu Delta, ale i jako jednoduchý textový řetězec.

Další šablonou je obrázek. Tato šablona umožňuje uživateli nahrát libovolný obrázek, který se poté zobrazí ve scéně v plné velikosti. Obrázek se z frontendové části po nahrání přenáší zakódovaný ve formátu Base64[6)] a poté je přenesen pomocí HTTP POST požadavku. Na serveru je obrázek umístěn do úložiště a relativní cesta k jeho umístění je uložena do databáze.

Po nahrání obrázku má uživatel možnost přidat do obrázku libovolný počet bodů, které jsou reprezentovány malými kruhy a které odkazují na další scény. Tyto body se přidávají kliknutím na libovolné místo na obrázku, kdy se uživateli zobrazí modální okno s nastavením daného bodu. To obsahuje možnost zadat text bodu a vybrat scénu, na kterou bod povede. Takto přidaný bod se pak ve hře objeví na stejné pozici. Tuto funkcionalitu lze použít například pro prozkoumávání mapy, kdy jednotlivé body označují místa na mapě, které může hráč navštívit.

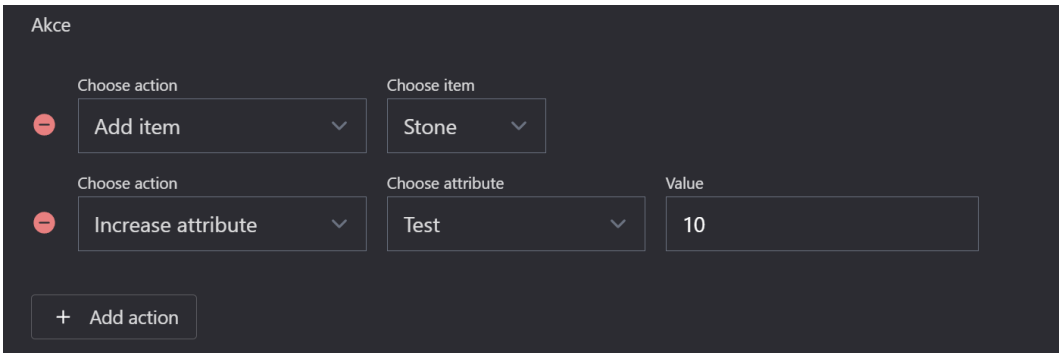
Také uživatel může definovat libovolný počet akcí, které se provedou poté, co hráč na bod klikne. Při definici akce je možné vybrat ze čtyř následujících možností:

[5)] Quill je knihovna bohatého textového editoru, volně dostupná z `https://quilljs.com/`

[6)] Base64 je formát, který slouží pro převod binárních dat do posloupnosti tisknutelných znaků, specifikovaný standardem RFC 4648, dostupný na `https://www.rfc-editor.org/rfc/rfc4648`

1. přidat předmět,
2. odebrat předmět,
3. zvýšit hodnotu atributu,
4. snížit hodnotu atributu

U akcí Přidat předmět a Odebrat předmět uživatel dostane na výběr ze seznamu předmětů, které byly ke hře přidané. U akcí Zvýšit hodnotu atributu a Snížit hodnotu atributu dostane na výběr ze seznamu přidaných atributů a ke zvolenému atributu poté specifikuje hodnotu, o kterou se změní. Ke každému přechodu, nebo rozhodnutí, lze přidat libovolný počet akcí. Ukázku definice akcí lze vidět na obrázku 5.5.



Obrázek 5.5 Ukázka definice akcí, které se provedou při přechodu na další scénu (vlastní tvorba)

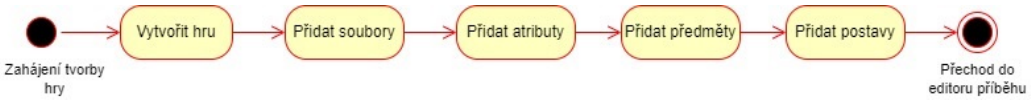
Šablona Text + rozhodnutí umožňuje uživateli vložit text, stejně jako v šabloně Text a definovat libovolný počet rozhodnutí. Rozhodnutí představují výběr z několika možností, která odkazují na další scény. Lze jej použít ve hře k vytvoření rozcestí z příběhu, kdy hráč svým výběrem může ovlivnit pokračování hry. U každého rozhodnutí uživatel definuje text, který se u rozhodnutí objeví a scénu, na kterou vede. Stejně jako u bodů v šabloně Obrázek, může uživatel pro každé rozhodnutí definovat libovolný počet akcí.

Další částí editoru příběhu jsou Jazykové nápovědy. Ty je možné přidat k jednotlivým scénám a slouží jako pomoc pro hráče, pokud například nerozumí textu, který se ve scéně nachází. Jejich seznam se nachází ve spodní části pravé nabídky. Uživatel může ke každé scéně přidat libovolný počet nápověd, které se skládají z názvu a textu. Tyto nápovědy se pak objeví ve hře u příslušné scény.

Veškeré nastavení a obsah, který uživatel vyplní, je uložen do sdíleného úložiště, pro které je využita knihovna Pinia. Díky tomu je možné předávat jednotlivé hodnoty mezi různými komponenty aplikace. O této knihovně detailněji pojednává kapitola 5.4. Zároveň se vše ihned odesílá do API a následně ukládá do databáze.

5.3.4 Kompletní tvorba hry

Vytvoření kompletní hry se skládá ze dvou kroků. První krok je vyplnění průvodce vytvořením hry, kde se definují základní informace o hře, dokumenty, atributy, předměty a postavy. Průchod tímto průvodcem je naznačen digramem aktivit, který lze vidět na obrázku 5.6.



Obrázek 5.6 Diagram aktivit pro vytvoření hry (vlastní tvorba, vytvořeno službou `diagrams.net`)

Druhým krokem je poté vytvoření obsahu hry. K tomu slouží editor příběhu, který byl popsán v kapitole 5.3.3. V tomto editoru se vytváří jednotlivé scény hry, které se poté spojují za sebe, podle toho, jak příběh probíhá. Postup je následující:

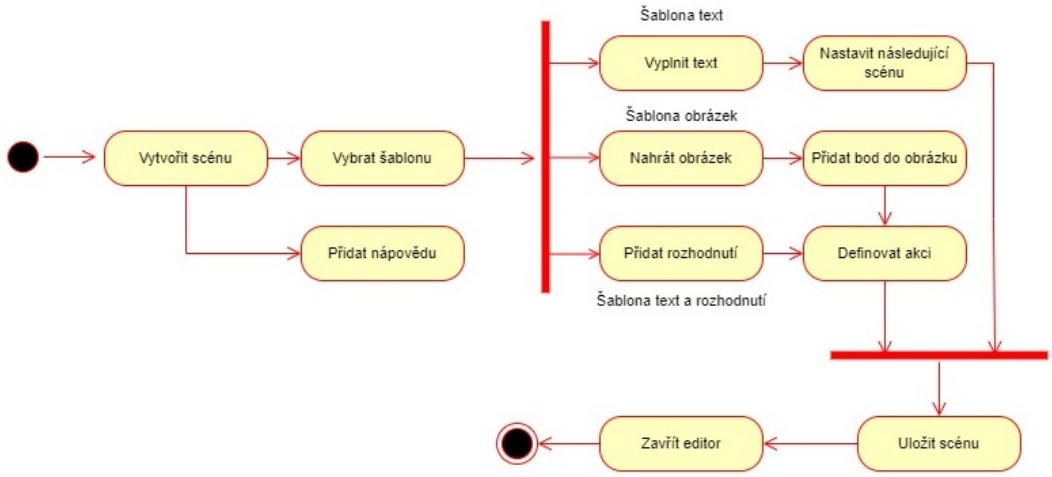
1. Uživatel vytvoří scénu, zvolí její název, barvu pozadí a zda je scéna hlavní
2. Uživatel vybere šablonu scény
3. Šablona scény určí způsob vyplnění obsahu:
    - Text – uživatel vloží text do editoru
    - Obrázek – uživatel nahraje obrázek, do kterého následně může přidávat body. Každá bod je napojen na další scénu a lze k němu definovat akce.
    - Text a rozhodnutí – uživatel vyplní text a přidá rozhodnutí, které vedou na další scény a lze k nim definovat akce.
4. Uživatel uloží scénu
5. Ke scéně lze přidat jazykovou nápovědu ve formě textu
6. Po vyplnění všech scén uživatel zavírá editor a hra je uložena

Ukázku průběhu tvorby obsahu hry lze vidět na obrázku 5.7.

5.3.5 Seznam her a vyhledávání

Seznam her se nachází na samostatné stránce, kde je rozdělen na dvě části. První část jsou hry, které vytvořil aktuálně přihlášený uživatel a druhá část jsou všechny hry, které nejsou nastaveny jako soukromé.

Oba tyto seznamy jsou vygenerovány pomocí vlastní composable funkce. Ta volá api, v prvním případě s parametrem id uživatele a vrací seznam her.



Obrázek 5.7 Diagram aktivit pro vytvoření hry (vlastní tvorba, vytvořeno službou `diagrams.net`)

V seznamu her lze také vyhledávat. Toto vyhledávání je tvořeno pomocí javascriptové funkce *filter()*, která vyfiltruje z objektu záznamy, jejichž parametr *name* se shoduje se zadaným výrazem. K tomuto vyhledávání bylo přistoupeno z důvodu rychlosti, jelikož hledání se provede po každé změně ve vyhledávacím poli. Varianta zasílání nového požadavku na API je v tomto případě pomalejší, pokud se nejedná o vyhledávání ve velkém počtu záznamů (řádově desítky tisíc a víc). Takový počet her v aplikaci není předpokládán a v případě, že by v budoucnu nastal, lze tuto funkcionalitu jednoduše předělat. Pro tento případ API obsahuje funkci pro vyhledávání her na základě jména.

Vyhledávání je implementováno pro oba seznamy her zvlášť, je tedy na uživateli, ve kterém seznamu chce zrovna vyhledávat.

U seznamu her lze přepínat mezi dvěma formáty. První možností je zobrazení v podobě dlaždic s obrázkem hry na pozadí. Druhou možností je jednoduchá tabulka, kde se na každém řádku nachází jedna hra.

5.3.6 Hra

Po vytvoření hry je pro ní vygenerována unikátní URL adresa, která slouží k jejímu spuštění. Ta je složena ze jména hry a případně doplněna čísly, pokud už hra se stejnou adresou existuje. Pokud se adresa nevygeneruje (například dojde k nějaké chybě), může administrátor hromadně vygenerovat adresy pro všechny hry, tlačítkem z administrátorského panelu.

Po přechodu na tuto URL adresu se hráči zobrazí úvodní stránka této hry. Na té se nacházejí základní informace o hře a tlačítko pro její spuštění. Veškeré informace o hře se automaticky načtou do sdílené paměti Pinia, která je popsána v kapitole 5.4. Do této paměti se následně ukládají všechna průběžná data o hře, jako je například

aktuální scéna, hodnoty atributů, předmětů, nebo informace o vybrané postavě.

Pokud má hra definované nějaké postavy, spustí se po zahájení hry jejich výběr. Uživateli se zobrazí přehled všech postav s přehledem informací o nich. Požadovanou postavu si vybere jednoduše kliknutím a tato postava se pak také uloží do sdílené paměti. Poté je uživatel přenesen na úvodní scénu hry. Pokud hra neobsahuje žádné postavy, je hráč přenesen do příběhu ihned.

Celá hra se odehrává na jedné obrazovce, kde se zobrazují jednotlivé komponenty, podle aktuální scény. Informace o hře obsahuje pravý panel, který zůstává stále na svém místě, jen se do něj načítá vždy aktuální obsah.

Po začátku hry se ze sdílené paměti vybere hlavní scéna, která je označena příznakem *main_scene*, a zobrazí se její obsah. Způsob zobrazení je určen příznakem *type*, který označuje šablonu scény. Podle tohoto příznaku se vybere správný způsob naformátování obsahu scény a její zobrazení uživateli.

Přechod na další scény pak probíhá identicky. Po kliknutí na prvek, který odkazuje na další scénu, si aplikace uloží identifikátor této scény a podle něj ve sdílené paměti vyhledá správnou následující scénu. Novou scénou je poté nahrazena scéna aktuální.

Zároveň aplikace obsahuje funkci pro načtení libovolné předchozí scény a to kvůli možnosti vracení se zpět mezi scénami. Tato funkce se spustí výběrem požadované scény ze seznamu předešlých scén.

Pokud jsou u přechodu na další scénu definovány nějaké operace, aktivují se funkce pro jejich provedení. Jedná se o dvě funkce, jedna pro operace s atributy a druhá pro operace s předměty. Každá operace se skládá ze tří částí. První část je identifikátor operace, který určuje, co se má stát. Jedná se o čtyři varianty a to zvýšení nebo snížení hodnoty atributu a přidání nebo odebrání předmětu. Druhá část představuje atribut nebo předmět, který bude ovlivněn. Třetí část je poté přítomna pouze u atributů a značí hodnotu, o kterou se má daný atribut snížit nebo zvýšit.

Každá scéna také může obsahovat jazykové nápovědy. Ty jsou vypsány v podobě seznamu v nabídce. Po kliku na konkrétní nápovědu se její obsah zobrazí v modálním okně.

5.4 Pinia

Pro uchovávání stavů v rámci sezení uživatele je použita knihovna Pinia. V aplikaci je definováno několik úložišť, které uchovávají data v různých částech aplikace. Tato úložiště se využívají z toho důvodu, aby bylo možné k datům přistupovat navzájem mezi jednotlivými komponenty. Pinia má také podporu přímo v Vývojářských nástrojích v internetovém prohlížeči a lze se tak jednoduše kdykoliv podívat, jaká data se ve kterém úložišti právě nachází.

Každé úložiště se skládá zpravidla ze tří částí. První část je definice jednotlivých proměnných (*state*) v úložišti, druhá část je definice funkcí pro získání dat (*getters*) a třetí část je definice funkcí pro uložení dat do úložiště a další operace s daty (*actions*). Příklad úložiště pro uchování přihlášeného uživatele a jeho rolí lze vidět na obrázku 5.8

```
import { defineStore } from 'pinia'

export const useLoggedUser = defineStore('loggedUser',{
    state: () => ({
        user: [],
        user_roles: [],
        userIsAdmin: false,
    }),
    getters: {
        getUser: (state) => state.user,
        getUserRoles: (state) => state.user_roles,
        getUserIsAdmin: (state) => state.userIsAdmin,
    },
    actions: {
        setUser(user: any) {
            this.user = user;
        },
        setUserRoles(user_roles: any) {
            this.user_roles = user_roles;
        },
        setUserIsAdmin(userIsAdmin: boolean) {
            this.userIsAdmin = userIsAdmin;
        }
    },
});
```

Obrázek 5.8 Ukázka definice úložiště knihovny Pinia (vlastní tvorba)

V aplikaci je definováno celkem 6 úložišť. Mezi ně patří úložiště *loggedUserStore* pro uchování dat o přihlášeném uživateli, *storyCreatorLeftMenuStore* a *storyCreatorRightMenuStore* pro uchování stavu ovládacích nabídek v editoru příběhu, *storyCreatorCurrentSceneStore* pro uchování dat aktuálně editované scény, *gameCreatorStore* pro uchování dat o aktuálně editované hře a *gamePlayStore* pro uchování dat aktuálně spuštěné hry.

Pinia umožňuje využít pro všechna data pouze jedno úložiště, ale z důvodu přehlednosti je dobrou praktikou oddělit jednotlivé logické celky. Rozdělení na více úložišť zároveň snižuje nároky na operační paměť, jelikož se jednotlivá úložiště načítají pouze pokud jsou potřeba.

5.5 Lokalizace

Veškeré texty v aplikaci jsou převážně v anglickém jazyce. Aby mohla být aplikace využita i uživateli, kteří anglický jazyk neovládají, bylo přistoupeno k její lokalizaci do českého jazyka.

Pro tuto lokalizaci je využita knihovna i18n[7]. Ta umožňuje definovat libovolný počet jazykových mutací pouze pomocí souborů ve formátu JSON. Pro každou mutaci je vytvořen soubor, pojmenovaný podle kódu daného jazyka. V tomto souboru jsou poté definovány veškeré texty, které mají být v aplikaci lokalizované. Ukázku definice textů lze vidět na obrázku 5.9. Ve všech těchto souborech jsou tyto texty uloženy pod stejným klíčem a v kódu se pak automaticky zobrazí podle uživatelem vybrané varianty jazyka.

Díky použité knihovně je v případě potřeby jednoduché přeložit aplikaci do dalšího libovolného jazyka.

```
{
    "hello": "Ahoj, {name}!",
    "language": "Jazyk",
    "home": "Domů",
    "pricing": "Ceník",
    "contact": "Kontakt",
    "show_more": "Zobrazit více",
    "login" : "Přihlásit se",
    "logout" : "Odhlásit se",
    "account" : "Účet",
}
```

Obrázek 5.9 Ukázka definice překladů
(vlastní tvorba)

[7] Lokalizační knihovna, verze pro Nuxt3 dostupná z `https://v8.i18n.nuxtjs.org/`

6 TESTOVÁNÍ

Důležitou součástí vývoje softwaru je jeho testování. To musí probíhat nejen po dokončení implementace, ale i během celého průběhu. V této kapitole jsou popsány způsoby testování aplikace. V kapitole 6.1 je popsáno vlastní testování programátorem. To probíhalo průběžně, vždy po implementaci nové funkcionality. V kapitole 6.2 je popsáno testování s reálnými uživateli.

6.1 Vlastní testování

Vlastní testování probíhalo během celého vývoje aplikace. Pokaždé, když byla implementována nová funkcionalita, byla nejprve samostatně otestována. Tím byly zjištěny hlavní nedostatky, které byly okamžitě opraveny. Po úspěšném otestování funkcionality byla dále otestována celá aplikace, aby bylo ověřeno, že nová funkcionalita nezpůsobila chybu v jiné části. Testování bylo vždy prováděno dle předem připraveného plánu, který sloužil jako kontrolní seznam. Tento plán je popsán v následující kapitole 6.1.1

6.1.1 Plán vlastního testování

Každé vlastní testování zahrnovalo kontrolu následujících funkcionalit:

- Vytvoření účtu
- Přihlášení do aplikace
- Korektní zobrazení přehledové stránky aplikace
- Zobrazení seznamu her
- Funkčnost vyhledávání her a přepínání mezi dvěma pohledy
- Zobrazení detailu hry
- Ohodnocení hry
- Vygenerování odkazu pro sdílení hry
- Vytvoření hry (s různými kombinacemi atributů, předmětů a postav)
- Editace hry
- Vytváření a správa tříd
- Responzivita uživatelského rozhraní

6.2 Testování s reálnými uživateli

Po dokončení implementace byla aplikace otestována s reálnými uživateli. Bylo vybráno několik uživatelů, kterým byla aplikace představena a poté dostali předem připravené zadání k testování, které bylo vytvořeno formou tabulky v aplikaci Google Sheets. Toto zadání obsahovalo seznam akcí, které si měli uživatelé vyzkoušet a poté zhodnotit, zda se jim akci podařilo úspěšně provést a zda se jim zdála dostatečné intuitivní. Zadání obsahovalo celkem 26 kroků, které pokryly veškeré hlavní funkcionality aplikace. U každého kroku uživatel zvolil, zda se mu akci povedlo úspěšně provést. Intuitivnost jednotlivých akcí byla určena škálou od 1 do 10, kde 1 je nejméně intuitivní a 10 naopak nejvíce. Ke každému kroku také bylo možné napsat textový komentář.

Prvním účastníkem testování byla Aneta T., studentka Pedagogické fakulty Masarykovy univerzity a učitelka anglického jazyka, která se zabývá metodou výuky pomocí rolových a simulačních her. Té se podařilo úspěšně provést téměř všechny operace. Problém nastal u nahrávání souboru ke hře, jelikož na serveru byl nastaven nízký limit pro maximální velikost odeslané žádosti a nahrávaný soubor ji přesáhl. Dále nastal problém při zpětné úpravě hry, kdy nebylo možné přejít z editoru hry přímo do editoru příběhu. V tomto případě došlo k chybě v implementaci, jelikož nebyl uložen identifikátor hry, podle kterého se editor příběhu načítá. Tato chyba byla následně opravena.

Poslední neúspěch nastal u pokusu o sdílení hry, kde se objevily hned dva problémy. Prvním problémem byla nefunkčnost tlačítka pro zkopírování adresy hry. Tato chyba nastala z toho důvodu, že javascriptová funkce, která slouží pro tuto operaci, požaduje bezpečný zdroj. To znamená, že doména webové aplikace musí být chráněna certifikátem, který v době testování ještě nebyl vystaven. Výjimku má aplikace běžící na lokálním počítači a z toho důvodu chyba unikla při vlastním testování programátorem.

Druhým problémem byl QR kód, který se vůbec nezobrazil. Tato chyba se také projevila pouze v ostrém provozu na serveru a došlo k ní také z důvodu nedůvěryhodného zdroje. Pro adresu služby, která poskytuje generování QR kódů, byla tedy udělena výjimka a QR kód se již korektně zobrazuje.

Intuitivnost provedení akcí byla hodnocena z většiny pozitivně, a to nejvyšší hodnotou 10. Za zmínku stojí například funkce hodnocení hry, která byla ohodnocena číslem 8, kdy bylo vytknuto, že pro ohodnocení hry musí být kliknuto nejdříve na text vedle hvězdiček, který následně zobrazí možnost hodnocení. Nejnižší hodnotu získala akce otevření editoru příběhu. Toto hodnocení souviselo s chybou, která byla popsána výše, kdy při editaci hry nebylo možné přejít do editoru příběhu rovnou, ale uživatel se musel nejdříve vrátit na detail hry a až pak editor otevřít.

Aneta taktéž navrhla možné vylepšení pro správu tříd, kde by se jí líbila možnost přidat i žáky, kteří nemají vytvořený účet, čímž by byly do aplikace pozváni.

Druhým účastníkem testování byl Ondřej K., učitel druhého stupně základní školy a autor projektu Agama, který byl zmíněn v kapitole 2.5.1.

Ondřejovi výsledky jsou téměř identické, jako výsledky Anety. Většina akcí proběhla úspěšně, problém nastal pouze u přidávání členů a her do třídy. Tam Ondřej zmiňuje, že mu obě tyto funkcionality nefungovali. V tomto případě se jednalo o problém s nejednoznačným uživatelským rozhraním, jelikož do třídy je možné přidat pouze uživatele a hru, které již existují a lze je vyhledat pomocí textového pole. Pokud se ovšem uživatel pokusí přidat uživatele nebo hru, která neexistuje, neukáže se mu žádná zpětná vazba o tom, že se přidání nezdařilo. Do uživatelského rozhraní byly tedy přidány doplňující informace o funkčnosti. Stejně jako Aneta také zmínil chybu s kopírováním odkazu na hru do schránky, která byla popsána výše.

Několik poznámek napsal k uživatelskému rozhraní. Stejně jako Aneta narazil při hodnocení hry na to, že hodnocení se neaktualizovalo okamžitě a tudíž nabyl dojmu, že není funkční. Další připomínka byla k nahrávání obrázku do scény, kdy při pokus o změnu obrázku není jasné, že už byl dříve nahraný jiný obrázek. Také méně body hodnotil přidávání nápověd ke scéně a to konkrétně pozici seznamu nápověd a tlačítka pro přidání v editoru scén.

Ondřej také navrhl několik funkcionalit, které by aplikace mohla v budoucnu obsahovat. Příkladem může být například možnost přidat do hry skrytý atribut. Ten by fungoval stejně jako aktuální atributy, ale hráč by ho při hraní neviděl a pouze by ovlivňoval průběh hry na pozadí. Dále zmínil možnost přidávat podmínky při přechodech na další scénu, kdy by následující scéna závisela na hodnotě určitého atributu.

Kromě chyb a připomínek také zmínil, že editor se mu líbí a jeho ovládání mu přijde intuitivní. Také pochválil moderní design aplikace.

Detailní zpětnou vazbu účastníků testování lze vidět v příloze 1

Na základě zpětné vazby bylo opraveno několik drobných chyb a byly mírně upraveny některé funkcionality, a to tak, aby co nejvíce vyhovovaly uživatelům, kteří budou aplikaci v budoucnu používat.

Obecně lze testování považovat za úspěšné, jelikož veškeré kritické chyby byly odhaleny a opraveny.

6.3 Porovnání se stávajícími řešeními

Po úspěšném otestování aplikace bylo provedeno porovnání se stávajícími řešeními. Vzhledem k povaze aplikace byly pro porovnání zvoleny ta řešení, která umožňují tvorbu rolových a simulačních her v elektronické podobě a jsou k tomuto účelu nejčastěji používány.

První řešení, které bylo zvoleno k porovnání, je aplikace pro tvorbu prezentací Microsoft PowerPoint. Tu mnozí učitelé využívají právě pro tvorbu příběhových her, i když k tomuto účelu primárně neslouží. Vytvářením jednotlivých stránek prezentace lze složit dohromady celý příběh a pomocí interaktivních prvků lze mezi jednotlivými stránkami libovolně přecházet. Pro vytváření jednoduchých her to samozřejmě stačí, ale pokud chce uživatel vytvořit více propracovanou hru, musí sáhnout po jiném řešení.

Na podobném principu funguje i služba Genially. Ta, podobně jako aplikace PowerPoint, slouží také k vytváření prezentací. Oproti té ale obsahuje další interaktivní prvky, které výsledek o trochu více přiblíží požadované hře.

Oproti tomu, řešení navržené v této práci slouží specificky pro tvorbu příběhových her a umožňuje tedy větší variabilitu při vytváření obsahu jednotlivých scén hry. Plocha scény není prostorově omezena a pokud do ní chce uživatel vložit například dlouhý text, je mu to umožněno, aniž by přesahoval přes okraj stránky. Při hraní hry v takovém případě může uživatel stránku vertikálně posouvat.

Kromě vkládání jednoduchých textů a obrázků obsahuje navíc interaktivní šablony. Jedna z šablon umožňuje vložit obrázek a do něj poté přidat pojmenované body. K těmto bodům lze poté nastavit, na kterou další scénu uživatele přenesou a jaké operace se při tom mají provést. Další šablona umožňuje k obsahu scény přidat výběr z možností, které, stejně jako body na obrázku, přenesou hráče na definovanou scénu a provedou příslušné operace. Operace slouží ke změně atributů a předmětů hry. U každého přechodu na další scénu jich lze definovat libovolný počet.

Ve výše zmíněných řešeních toto nelze jednoduše udělat. Jedinou možností je mít veškeré změny atributů a předmětů napsané přímo v obsahu stránky a při hraní si je zapisovat, nebo mít jejich hodnoty pevně zapsané přímo na stránce.

Odlišný způsob vytváření her poté nabízí aplikace RPG Creator. V této aplikaci je možné vytvářet jednoduché 2D hry, kde se hráč příběhem nepohybuje pomocí přechodu mezi jednotlivými scénami, ale místo toho svou postavu ovládá na mapě pomocí klávesnice. Tyto hry se pak spíše blíží klasickým počítačovým hrám, místo výukových her, o kterých pojednává tato práce. Zároveň také ovládání aplikace je poměrně náročné a pokud uživatel nemá příliš zkušeností ze světa počítačových her, nebude pro něj tato volba ideální.

Vzhledem k tomu, že řešení z této práce slouží primárně pro využití ve výuce, obsahuje i další podpůrné funkcionality. Hlavní z nich jsou takzvané *Language Aids*, neboli jazykové nápovědy. Ty je možné přidat ke všem scénám hry a slouží jako jazyková podpora pro hráče, kteří jsou v daném jazyce slabší a potřebují pomoc. Uživatel do těchto nápověd může vložit libovolný text, který lze poté u každé scény jednoduše zobrazit. Nápovědy lze samozřejmě využít i pro jinou pomoc, pokud například hra není určena

pro výuku cizího jazyka, ale jiného tématu.

Další výhodou implementovaného řešení je možnost jednoduchého sdílení a procházení her. Uživatel si může prohlížet seznam veškerých volně přístupných her a kteroukoliv z nich si může zahrát. Zároveň lze uživatele seskupovat do tříd, do kterých lze přidat libovolný počet her, aby je vždy jednoduše nalezli. Možnost hraní her, které vytvořili ostatní uživatelé, může sloužit nejenom k usnadnění práce, ale například i pro inspiraci při následném vytváření her vlastních.

7 NASAZENÍ APLIKACE

Aplikaci je možné nasadit na libovolný server, který podporuje přístup pomocí SSH a splňuje minimální požadavky.

7.1 Požadavky

Pro nasazení aplikace je nutné, aby server splňoval určité požadavky. V této kapitole jsou veškeré tyto programy a služby popsány.

7.1.1 Seznam požadavků

- Nginx / Apache
- PHP >= 7.4
  - BCMath PHP Extension
  - Ctype PHP Extension
  - Fileinfo PHP extension
  - JSON PHP Extension
  - Mbstring PHP Extension
  - OpenSSL PHP Extension
  - PDO PHP Extension
  - Tokenizer PHP Extension
  - XML PHP Extension
  - Postgresql Extension
- Node >= 16.15.0
- NPM >= 8.5.5
- Composer
- Postgresql

7.1.2 Nastavení webového serveru

Jako webový server lze použít libovolný webový server, který podporuje aplikace Node a PHP. Pro nasazení této aplikace byl použit server Nginx, další text se tedy bude věnovat konfiguraci tohoto serveru.

V základním nastavení načítá Nginx soubory, které se nacházejí ve složce varwwwhtml. Pokud chceme tuto cestu změnit, nebo provozovat na serveru více služeb, je nutné definovat vlastní konfigurační soubory. Pro každou službu (klient, api a databáze) vytvoříme vlastní konfigurační soubor, ve kterém se daná služba nasměruje na doménu, na které má být dostupná.

V případě této aplikace je tedy nutné provést konfiguraci zvlášť pro API část, frontendovou část a pro databázi.

API je po nastavení nginx serveru automaticky dostupné na vybrané adrese. Oproti tomu u frontendové části je nutné nejdříve provést sestavení aplikace a poté ji spustit. Pro sestavení aplikace slouží příkaz *npm run build*, který porvede veškeré operace a výstup uloží do složky *.output*. Pro spuštěné sestavené aplikace lze použít například službu *pm2*, která aplikaci spustí na pozadí. Stačí ji spustit příkazem *pm2 start /var/www/frontend/run.sh*, kde soubor *run.sh* obsahuje příkazy pro změnu umístění do složky frontend a spuštění příkazu *npm run start*. Poté je aplikace spuštěna na doméně, která byla nastavena v konfiguračním souboru.

ZÁVĚR

Cílem této práce bylo navrhnout a implementovat aplikaci, která bude sloužit jako platforma pro vytváření a hraní rolových a simulačních her s využitím ve výuce. Tato aplikace má sloužit pro učitele, kteří tuto výukovou metodu využívají, aby je mohli jednoduše vytvářet a sdílet mezi sebou. Kromě samotné tvorby her aplikace obsahuje další podpůrné funkcionality, jako je vytváření a správa tříd, sdílení her, nebo jejich hodnocení.

V teoretické části se práce zaměřuje na historii a současný vývoj webových aplikací. Popisuje různé možnosti vývoje, rozdílné architektury, přístupy a technologie. Dále se věnuje rolovým a simulačním hrám. Tyto hry jsou stručně charakterizovány a jsou vysvětleny možnosti jejich použití ve výuce. Následuje rozdělení těchto her na fyzické a digitální a představení již existujících řešení z obou kategorií.

Praktická část se věnuje samotnému návrhu a vývoji aplikace. Prvním krokem bylo sepsání veškerých požadavků na aplikaci. Byla zvolena architektura, která bude využita pro implementaci, a bylo vytvořeno schéma databáze, které popisuje způsob ukládání dat. V dalším kroku byla provedena analýza demografického rozdělení cílového publika a dle ní byl poté proveden návrh uživatelského rozhraní aplikace.

Na základě tohoto návrhu byly vybrány vhodné technologie, pomocí kterých byla provedena implementace výsledného řešení.

Výsledkem práce je funkční aplikace, která umožňuje jednoduché a intuitivní vytváření rolových a simulačních her pro využití ve výuce. Aplikace splňuje veškeré stanovené požadavky a poskytuje uživatelům jednoduché a intuitivní rozhraní pro tvorbu, sdílení, správu a hraní her. Dále umožňuje vytvářet třídy pro seskupování uživatelů a přidávání her, které tak budou jednoduše přístupné pro členy dané třídy.

Pro tvorbu obsahu hry byl implementován interaktivní editor, který umožňuje vytvářet jednotlivé části příběhu, vkládat do nich textový a multimediální obsah a pomocí různých prvků je spojovat do finálního celku. Mezi interaktivní prvky patří například možnost definice klikatelných bodů v obrázku, které mohou sloužit pro prohledávání prostoru, nebo výběr z více rozhodnutí, které následně ovlivní děj příběhu.

Aplikace byla otestována s reálnými uživateli, kteří poskytli cennou zpětnou vazbu a několik návrhů na budoucí rozšíření. Na základě této zpětné vazby byly některé funkcionality poupraveny tak, aby vyhovovaly uživatelům, kteří budou výslednou aplikaci používat.

V budoucnu se nabízí různé možnosti rozšíření aplikace, které by umožnily další využití ve výuce. Jako příklad lze uvést přidání podrobných statistik her, díky kterým by hráči mohli mezi sebou soutěžit. Další možností by mohlo být přidání nových funk-

cionalit do editoru příběhu hry, jako například více interaktivních prvků pro tvorbu scén, nebo možnost nastavit atribut hry jako skrytý, který by poté ovlivňoval průběh hry na pozadí.

SEZNAM POUŽITÉ LITERATURY

[1] CZ.NIC: Webové aplikace [online]. https://www.jaknainternet.cz/page/1262/webove-aplikace/, 2013, [cit. 2023-01-16].

[2] Tung, L.: JavaScript creator Eich: My take on 20 years of the world's top programming language [online]. https://www.zdnet.com/article/javascript-creator-eich-my-take-on-20-years-of-the-worlds-top-programming-language/, 2017, [cit. 2023-01-16].

[3] Jaz Hoffmann: What Does AJAX Even Stand For? [online]. https://thehistoryoftheweb.com/blog/2005/02/17/what-does-ajax-even-stand-for/, 2019, [cit. 2023-01-16].

[4] Greif, S.; Burel, E.: The 2022 State of JS survey [online]. https://2022.stateofjs.com, 2022, [cit. 2023-04-15].

[5] The Editors of Encyclopaedia Britannica: Client-server architecture. https://www.britannica.com/technology/client-server-architecture, 2023, [cit. 2023-04-16].

[6] Alegsa, L.: Klient-server [online]. https://cs.alegsaonline.com/art/20972, 2021, [cit. 2023-04-16].

[7] Harris, C.: Microservices vs. monolithic architecture [online]. https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith, 2022, [cit. 2023-01-21].

[8] IBM: What is service-oriented architecture (SOA)? [online]. https://www.ibm.com/topics/soa, 2022, [cit. 2023-04-16].

[9] Google: What is Microservices Architecture? [online]. https://cloud.google.com/learn/what-is-microservices-architecture, 2022, [cit. 2023-04-16].

[10] Tišnovský, P.: Mikroslužby: moderní aplikace využívající známých konceptů [online]. https://www.root.cz/clanky/mikrosluzby-moderni-aplikace-vyuzivajici-znamych-konceptu/, 2019, [cit. 2023-04-16].

[11] Longbottom, C.: Styles, protocols and methods of microservices communication [online]. https://www.techtarget.com/searchapparchitecture/tip/Styles-protocols-and-methods-of-microservices-communication, 2020, [cit. 2023-04-16].

[12] Amazon AWS: What is an Event-Driven Architecture? [online]. https://aws.amazon.com/event-driven-architecture/, 2023, [cit. 2023-04-16].

[13] Per Christensson: Frontend Definition [online]. https://techterms.com/definition/frontend, 2020, [cit. 2023-03-15].

[14] MDN contributors: Responsive design [online]. https://developer.mozilla.org/en-US/docs/learn/css/css_layout/responsive_design, 2023, [cit. 2023-03-15].

[15] Kadlec, T.: *Implementing Responsive Design: Building Sites for an Anywhere, Everywhere Web (Voices That Matter)*. New Riders, první vydání, 2012, ISBN 978-0321821683.

[16] Lie, H. W.; Çelik, T.: Media queries [online]. https://www.w3.org/TR/2001/WD-css3-mediaqueries-20010517/, 2001, [cit. 2023-04-27].

[17] MDN contributors: Basic concepts of flexbox [online]. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox, 2023, [cit. 2023-04-28].

[18] MDN contributors: Basic concepts of grid layout [online]. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Basic_Concepts_of_Grid_Layout, 2023, [cit. 2023-04-28].

[19] Eseme, S.: Backend Development: Beginners Guide to Backend Development [online]. https://masteringbackend.com/posts/getting-started-with-backend-development/, 2021, [cit. 2023-03-15].

[20] Clark, J.: The Best 10 Backend Programming Languages [online]. https://blog.back4app.com/backend-programming-languages-list/, 2020, [cit. 2023-03-15].

[21] Red Hat, Inc.: What is an API? [online]. https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces, 2022, [cit. 2023-03-13].

[22] MDN contributors: HTTP [online]. https://developer.mozilla.org/en-US/docs/Web/HTTP, 2023, [cit. 2023-04-28].

[23] MDN contributors: HTTP request methods [online]. https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods, 2023, [cit. 2023-03-31].

[24] Hadji-Vasilev, A.: 23 Video Game and Online Gaming Statistics, Facts Trends for 2023 [online]. https://www.cloudwards.net/online-gaming-statistics/, 2022, [cit. 2023-04-17].

[25] Deterding, S.; Dixon, D.; Khaled, R.; aj.: From game design elements to gamefulness: defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, ročník 11, Září 2011, ISBN 9781450308168, s. 9–15, doi:10.1145/2181037.2181040.

[26] Hosch, W. L.: Dungeons Dragons. https://www.britannica.com/biography/Ernest-Gary-Gygax, 2023, [cit. 2023-03-25].

[27] Krahulec, O.: Agama. http://agama.ped.muni.cz/, n.d., [cit. 2023-03-28].

[28] Mottl, J.: *Interaktivní herní prostředí založené na dynamickém webu jako prostředek gamifikace výuky*. Bakalářská práce, Univerzita Hradec Králové, Duben 2019. URL `https://theses.cz/id/esj845/32255643`

[29] Monte Cook Games: Cypher System. http://cypher-system.com/, n.d., [cit. 2023-03-28].

[30] Witters, K.: RPG Playground. https://rpgplayground.com/, n.d., [cit. 2023-03-28].

[31] UCAR Software Engineering Assembly: Software design and modeling [online]. https://sea.ucar.edu/best-practices/design, 2023, [cit. 2023-04-03].

[32] Garrett, J. J.: *The Elements of User Experience: User-Centered Design for the Web and Beyond*. New Riders, druhé vydání, 2011, ISBN 978-0-321-68368-7.

[33] Mulder, S.; Yaar, Z.: *The User Is Always Right: A Practical Guide to Creating and Using Personas for the Web*. New Riders, první vydání, 2006, ISBN 0321449924.

[34] OECD Data: Teachers by age [online]. https://data.oecd.org/teachers/teachers-by-age.htm, 2020, [cit. 2023-04-23].

[35] MDN contributors: Your first form [online]. https://developer.mozilla.org/en-US/docs/Learn/Forms/You_first_form, 2023, [cit. 2023-04-03].

[36] Croft, J.: Frameworks for Designers [online]. https://alistapart.com/article/frameworksfordesigners/, 2007, [cit. 2023-03-06].

[37] Subecz, Z.: Web-development with Laravel framework. *Gradus*, ročník 8, č. 1, duben 2021: s. 211–218, ISSN 2064-8014.

[38] Tijūnaitis, T.: How to send emails with Laravel using MailerSend [online]. https://www.mailersend.com/blog/send-email-in-laravel, 2021, [cit. 2023-03-31].

[39] Thiessen, M.: Best Features in Nuxt 3 [online]. https://masteringnuxt.com/blog/best-features-in-nuxt-3, 2023, [cit. 2023-03-13].

[40] Vue.js: Introduction [online]. https://vuejs.org/guide/introduction.html, 2022, [cit. 2023-04-28].

[41] The PostgreSQL Global Development Group: About [online]. https://www.postgresql.org/about/, 2023, [cit. 2023-03-13].

[42] Bassett, L.: *Introduction to JavaScript Object Notation*. O'Reilly, první vydání, 2015, ISBN 978-1-491-92948-3.

[43] Postman, Inc.: What is Postman? [online]. https://www.postman.com/product/what-is-postman/, 2023, [cit. 2023-04-03].

[44] Andrzejewski, J.: Nuxt Security [online]. https://github.com/Baroshem/nuxt-security, 2021, [cit. 2023-04-23].

[45] Andrzejewski, J.: Nuxt Security [online]. https://nuxt-security.vercel.app/, 2021, [cit. 2023-04-23].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AMQP	Advanced Message Queuing Protocol
AJAX	synchronous JavaScript and XML
API	Application Programming Interface
CORS	Cross-origin resource sharing
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
HTML	Hypertext Transfer Protocol
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
QR	Quick Response
REST	Representational state transfer
RPG	Role-playing game
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured query language
TCP	Transmission Control Protocol
UI/UX	User interface / User experience
URL	Uniform Resource Locator
WYSIWYG	What you see is what you get
XML	Extensible Markup Language
XSS	Cross-site scripting

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I. VÝSLEDKY UŽIVATELSKÉHO TESTOVÁNÍ

Akce	Úspěch provedení	Intuitivnost	Komentář
Otevřít aplikaci	ANO	10	
Vytvořit si účet	ANO	10	
Přihlásit se do účtu	ANO	10	
Zobrazit seznam her	ANO	10	
Zobrazit detail libovolné hry	ANO	10	
Spustit hru	ANO	10	
Zkusit hru dohrát	ANO	10	
Ohodnotit zahranou hru	ANO	8	Původně jsem klikala na hvězdičky, ne na text, doporučila bych ho buď podtrhnout nebo rozlišit barevně
Zobrazit svůj profil	ANO	10	
Vytvořit vlastní hru	ANO	10	
Nahrát ke hře soubor	NE	10	Soubor (video ke hře) se bohužel nenačetl
Přidat do hry atributy a předměty	ANO	10	
Přidat do hry postavu	ANO	10	Přidat ano, změnit ne
Otevřít editor příběhu	ANO/NE	5	Samostatně jde otevřít, ale když jsem na něj pokračovala z "Editoru hry", nejdřív se nenačetl a potom zobrazil prázdné scény (přestože už hra byla vytvořena)
Přidat scénu hry a vyplnit ji	ANO	9	
Vytvořit scénu s šablonou "Obrázek"	ANO	10	
Přidat do obrázku bod s odkazem na další scénu	ANO	10	
Přidat ke scéně nápovědu	ANO	10	
Zavřít editor příběhu a zobrazit detail vytvořené hry	ANO	10	
Sdílet odkaz na hru	ANO/NE	8	Nefunguje tlačítko "Copy" ani QR kód, pouze manuální zkopírování odkazu
Editovat hru	ANO	9	Nejde se vracet mezi jednotlivými kroky a editovat už vytvořené, např. postavu
Zobrazit seznam tříd	ANO	10	
Vytvořit vlastní třídu	ANO	10	Třída potom nejde vymazat/upravovat
Přidat člena do třídy	ANO	9	Jdou přidat pouze registrovaní žáci, do budoucna bych zařadila možnost "pozvat do třídy"
Přidat hru do třídy	ANO	10	
Odhlásit se	ANO	10	

Obrázek 1.1 Výsledky testování uživatelky Anety T. (vlastní tvorba)

Akce	Úspěch provedení	Intuitivnost	Komentář
Otevřít aplikaci	Ano	10	
Vytvořit si účet	Ano	10	
Přihlásit se do účtu	Ano	10	
Zobrazit seznam her	Ano	10	
Zobrazit detail libovolné hry	Ano	10	
Spustit hru	Ano	10	
Zkusit hru dohrát	Ano	10	
Ohodnotit zahranou hru	Ano	8	Zkusil jsme hodnotit svou hru, nešlo to - je to bug nebo fičura? :-) Nereagovalo to, ale když jsme odešel a přišel, už byla hra napsaná, že je zhodnocená ode mne - takže asi problém při tom aktu přímo?
Zobrazit svůj profil	Ano	10	
Vytvořit vlastní hru	Ano	10	
Nahrát ke hře soubor	Ano	10	V jednu chvíli upload button nereagoval, pak už to šlo a soubor se mi objevil jako nahraný - až na několikátý pokus
Přidat do hry atributy a předměty	Ano	10	Ocenil bych možnost mít skrytý atribut, se kterým hra hraje, ale není veřejný - vlastní proměnné, co se mění, ale jsou pro hráče neviditelné jen důležité pro hru - tzn. např. viditelnost atributů nastavit
Přidat do hry postavu	Ano	10	Nejde mi odstranit postava, která je z Vaší nabídky, jen jsem zkusil a křížek mi nefunguje - začalo to fungovat až jsme odešel a přišel zpět do editace hry po jejím vytvoření
Otevřít editor příběhu	Ano	10	
Přidat scénu hry a vyplnit ji	Ano	7	Na začátku jsem nenašel ty typy scén, tak jsme dal text a pak tam hledal možnosti, na druhý pokus už dobré, jak jsou karty za sebou tak to trošku mátlo při výběru, co chci
Vytvořit scénu s šablonou "Obrázek"	Ano	10	Nenašel jsme možnost editovat později bod (!), při kliknutí na editovat scénu se mi nenahrál obrázek a vypadalo to jako když se vše ztratilo, když jsme nahrál obrázek, tak se objevil i bod
Přidat do obrázku bod s odkazem na další scénu	Ano	8	
Přidat ke scéně nápovědu	Ano	6	šipka napravo mi dala zabrat, abych našel meníčko i tam
Zavřít editor příběhu a zobrazit detail vytvořené hry	Ano	10	
Sdílet odkaz na hru	Ano	10	Tlačítko copy mi nehodilo do schránky link, musel jsem ručně
Editovat hru	Ano	9	Vytvořil jsem veřejnou hru, po vytvoření jsme se rozhodl ji změnit na neveřejnou - stále ji vidím zalistovanou mezi all public games, i když v nastavení už je zatrženo, že ji lze hrát jen s odkazem, chyběla mi možnost jít přímo do určitého kroku v editoru hry, jinak jsme to musle projít vše až do konce. Trochu mne mátlo, že tam není save button - nebo jsem ho neviděl
Zobrazit seznam tříd	Ano	10	
Vytvořit vlastní třídu	Ano	10	
Přidat člena do třídy	Ne	10	V okýnku přidat člena mi nefungovalo tlačítko add, takže jsme Jirku nepřidal
Přidat hru do třídy	Ne	10	V okýnku přidat hru mi nefungovalo tlačítko add
Odhlásit se	Ano	10	

Obrázek 1.2 Výsledky testování uživatele Ondřeje K. (vlastní tvorba)