Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Framework na tvorbu sieťových hier so slovami

Bakalárska práca

2020
Andrea Smiešna

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Framework na tvorbu sieťových hier so slovami

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra didaktiky matematiky, fyziky a informatiky
Školiteľ:	PaedDr. Daniela Bezáková, PhD.

Bratislava, 2020
Andrea Smiešna

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Andrea Smiešna
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Framework na tvorbu sieťových hier so slovami
Framework for creating colaborative word games

Anotácia: Študent zanalyzuje niekoľko existujúcich sieťových (webových aj mobilných) hier zameraných na prácu so slovami pre žiakov ZŠ (napr. spoločná tvorba jednoduchých myšlienkových máp, tvorba slov z daných písmen, osemsmerovky, ...). Na základe analýzy navrhne a implementuje modul, resp. skupinu modulov, v ktorých budú definované základné triedy a nástroje, ktoré zjednodušia tvorbu takýchto hier. Funkčnosť modulu predvedie študent vytvorením niekoľkých ukážkových hier. Študent zvolí na tvorbu frameworku vhodné nástroje a technológie tak, aby výsledné hry boli buď responzívne webové aplikácie alebo mobilné aplikácie.

Cieľ: Navrhnúť a implementovať framework na tvorbu sieťových (kolaboratívnych) hier zameraných na hranie sa so slovami.

Kľúčové slová: framework, sieťová hra, kolaborácia

Vedúci: PaedDr. Daniela Bezáková, PhD.
Katedra: FMFI.KDMFI - Katedra didaktiky matematiky, fyziky a informatiky
Vedúci katedry: prof. RNDr. Ivan Kalaš, PhD.

Dátum zadania: 25.09.2019

Dátum schválenia: 07.10.2019 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie:

Čestné vyhlásenie:

Abstrakt

Kľúčové slová: framework, sieťová hra, kolaborácia

Abstract

Keywords: framework, network game, collaboration

Obsah

Zoznam obrázkov

Zoznam skratiek a značiek

CSS Cascading Style Sheets (Kaskádové štýly)

HTML HyperText MarkUp Language (Hypertextový značkový jazyk)

JSON JavaScript Object Notation (Označenie JavaScript objektu)

NPM Node.js Package Manager (Node.js správca balíkov)

Úvod

Online hry v súčasnosti tvoria bežnú súčasť života mnohých detí i dospelých. Hry sa dajú hrať prostredníctvom počítača, mobilného telefónu, či tabletu. Stačí sa pripojiť na internet, vybrať si hru, ktorá je pre daného človeka zaujímavá a hra sa môže začať. Online hier je nespočetne veľa, preto si každý môže vybrať podľa svojho vkusu, ale aj aktuálnej nálady.

Výhodou hier hraných cez sieť je fakt, že sa môže zapojiť každý - kamaráti aj rodina. Pri snahe o porazenie spoluhráča je podnecovaná súťaživosť, no niekedy hra vyžaduje opak a hráči si navzájom musia pomáhať a na dosiahnutie vytúženého cieľa je nutná spolupráca, čo zlepšuje schopnosť pracovania v tíme. Často sú však takéto hry odsudzované, pretože obsahujú veľa násilia a môžu vyvolať závislosť. Avšak edukačné hry sú opakom - obsahujú toho omnoho viac a môžu dieťa naučiť mnohým vlastnostiam, ktoré sú prospešné a dieťa môže vďaka nim byť úspešnejšie v neskoršom živote. V dnešnej dobe sú čoraz atraktívnejšie aj vo vyučovacom procese a učitelia využívajú online edukačné hry ako plnohodnotného pomocníka pri učení matematiky, či cudzieho jazyka.

Precvičovanie slovnej zásoby dostatočne nemotivuje, často je nezáživné a pre žiaka jednoznačne nepútavé. Učenie hrou je populárna metóda, ktorá zavádza herné prvky do vzdelávacích aktivít tak, aby boli pre žiakov zábavnejšie a zaujímavejšie.

Cieľom aplikácie „Framework na tvorbu sieťových hier so slovami“ je vytvoriť nástroj, ktorý uľahčí a urýchli vývoj jednoduchých sieťových hier so slovami.

Jedným z možných používateľov frameworku môže byť programátorsky zdatný učiteľ. Použitím predpripravenej šablóny hry vytvorí jednoduchým spôsobom nástroj, pomocou ktorého si žiaci hravo rozšíria slovnú zásobu, napríklad skladaním písmen do slov, pričom sú motivovaní ku učeniu pomocou získavaných bodov.

Práca je rozdelená do troch základných kapitol.
Pred samotnou implementáciou sa budeme venovať už existujúcemu systému, jeho nedostatkom a celkovej motivácii pre výber práce s danou témou. V analýze podobných bakalárskych prác sa zameriame na aspekty, ktoré boli podstatné pri návrhu riešenia. V ďalšom kroku analyzujeme technológie, ktoré budeme v práci využívať.

Vďaka výsledkom získame užitočné poznatky, ktoré budú využité v ďalšej fáze práce, teda pri návrhu. V tejto kapitole stanovíme cieľ a konkrétne požiadavky, ktoré má framework spĺňať tak, aby bol čo najväčším pomocníkom pri vývoji nenáročných hier so slovami. Implementácii bude predchádzať uvedenie požiadaviek pre programátora, opíšeme konkrétne znalosti, ktoré musí používateľ ovládať pre vytvorenie sieťovej hry, použitím frameworku.

V implementačnej časti, predstavíme framework a jeho funkcionalitu. Definujeme a podrobnejšie popíšeme základné funkcie, ktoré používateľ využije pri tvorbe webovej hry. Opíšeme zásadné problémy, ktoré nastali pri implementácii a zameriame sa aj na ich následné riešenia. Poslednou časťou tejto kapitoly bude predstavenie ukážkových hier, vytvorených pomocou frameworku. Postup inštalácie vysvetlíme v niekoľkých krokoch, po prečítaní bude používateľovi zrejmé, aké komponenty je potrebné stiahnuť pre úspešné používanie frameworku.

Poslednou časťou práce je testovanie frameworku. Po úspešnej implementácii ho otestujú študenti vysokej školy, ktorí majú potrebné programátorské zručnosti. Otestovaním frameworku vyplynú chyby, a tiež to, aké nutné opravy musíme urobiť na jej vylepšenie. Zároveň získame nový pohľad na výsledné riešenie, a ako následok sa vyvinú prípadné rozšírenia frameworku, ktoré môžu byť implementované v budúcnosti.

Kapitola 1

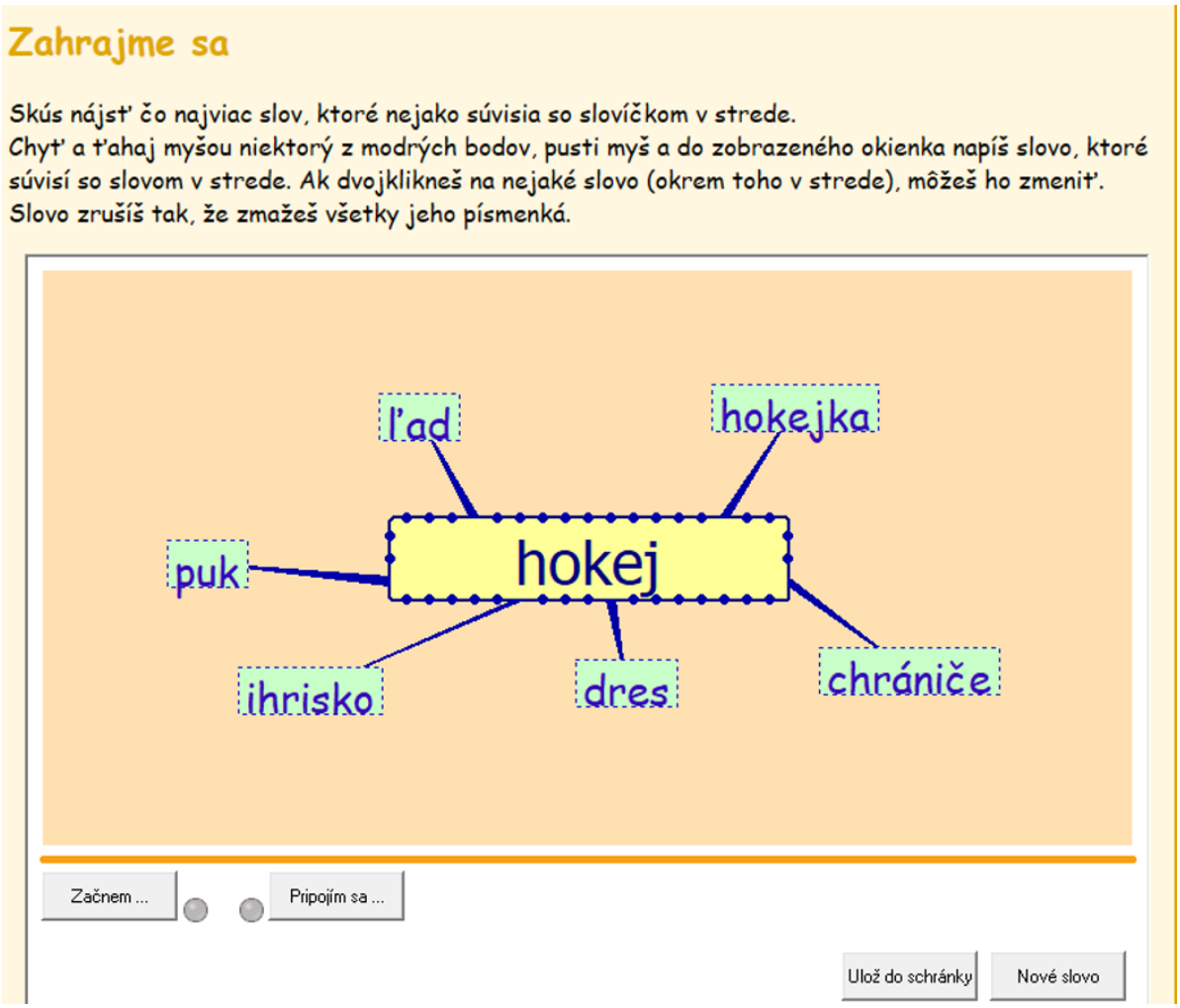
Teoretické východiská

1.1 Existujúci systém - Infovekáčik

Nápad vytvoriť nástroj na tvorbu jednoduchých hier so slovami vznikol netradične pri známej mobilnej hre Piknik. Cieľom hry je poskladať z písmen zmysluplné slová a samozrejme zabaviť sa. Po chvíľke sledovania päťročnej sesternice som bola prekvapená, ako jednoducho sa naučila pracovať s písmenkami a správne ich spojila do slov.Zábava a vzdelávanie v jednom, to je určite heslo dnešnej doby. Nápad bakalárskej práce sa zdokonalil zistením, že existuje systém, ktorý ponúka niekoľko zaujímavých hier so slovami a naprogramovali ho učitelia v jazyku Imagine. Infovekáčik je v skutočnosti online časopis, v ktorom si dieťa nájde omaľovánky, pesničky, rubriku o krásnych miestach Slovenska a v neposlednom rade sieťové hry pre viacero hráčov.

Obr. 1.1: Infovekáčik

Infovekáčik [12] vznikol v roku 2003 a technológie odvtedy pokročili. Žiak po spustení stránky s hrou zistí, že potrebuje nainštalovať špeciálny Imagine plugin, ktorý funguje iba v prehliadači Internet Explorer. Taktiež je nutné manuálne zadať IP adresu, pokiaľ sa chce spolu zahrať viacero hráčov. Ďalšia nevýhoda Infovekáčika, sa môže objaviť, pokiaľ ho chce používateľ používať na inom operačnom systéme ako Windows. Imagine, programovací jazyk, v ktorom je Infovekáčik vytvorený, nefunguje na operačných systémoch Linux a macOS. Nespadá preto medzi softvér, ktorý môže bežať na viacerých počítačových platformách, nie je multiplatformový.

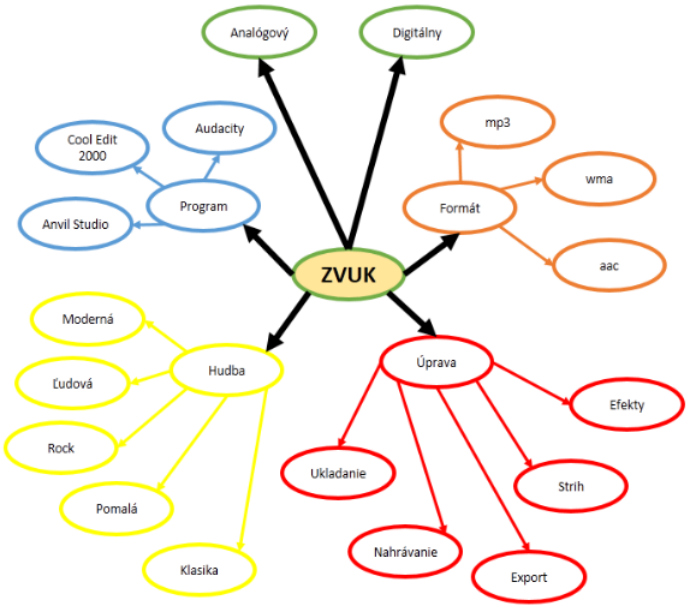


Obr. 1.2: Infovekáčik - myšlienková mapa

1.2 Existujúce hry

Inšpiráciou pre budovaciu časť aplikácie boli hry, ktoré som hrávala v minulosti ja, ale aj tie, ktoré spríjemnia napríklad dlhú cestu v aute. Vybrala som hry, ktoré umožnia hráčom prejaviť svoju kreativitu pri tvorení nových slovíčok a precvičia slovnú zásobu dieťaťa, či dospelého.

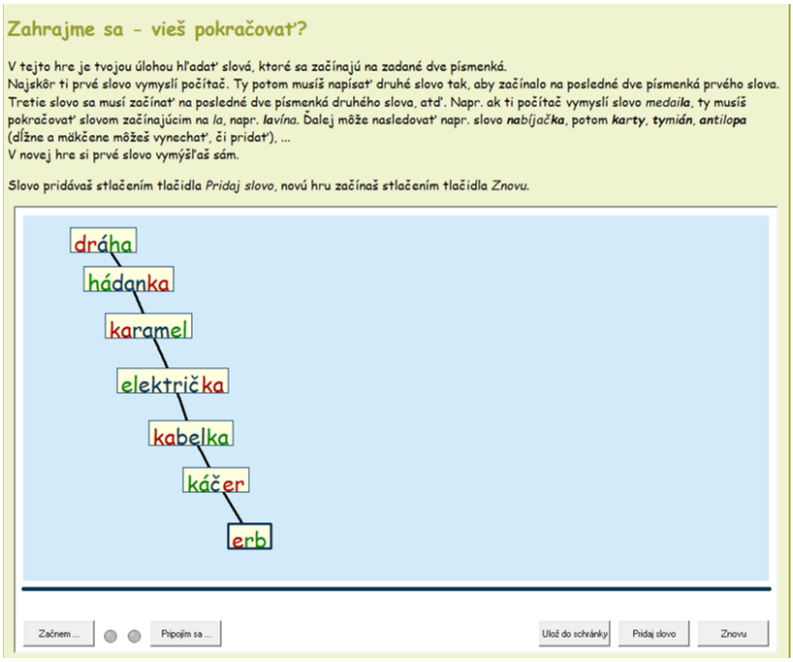
1.2.1 Myšlienková mapa



Obr. 1.3: Príklad myšlienkovej mapy

Myšlienkové mapy sú prirodzeným odzrkadlením toho, ako myseľ pracuje. Spája informácie okolo jednej hlavnej myšlienky a vytvára prepojenia. Vytvorenie mapy pomáha generovať viac nápadov a prekladá myšlienky v mysli do uceleného obrázku, teda diagramu. Zapamätávanie si dlhých, zložitých viet je náročnejšie ako pamätanie si kľúčových slov. Používanie farieb a obrázkov pôsobí pri pamätaní dôležitých informácií oveľa zaujímavejšie a zábavnejšie [24]. Myšlienková mapa sa zaradila medzi efektívne pomôcky ako u detí rozvíjať myslenie. Táto metóda deťom umožňuje zjednodušenie písania poznámok, učenia, skrátenie doby strávenej nad domácimi úlohami, rozvíja kreativitu, predstavivosť a zlepšuje pamäť. Hráč začína od hlavného slova daného na začiatku hry a hráč skúsi nájsť čo najviac slovíčok, ktoré s ním súvisia. Hru „Myšlienková mapa“ [10] je možné použiť ako pomocníka pri spracovaní poznámok na uľahčenie učenia.

1.2.2 Vieš pokračovať?



Obr. 1.4: Príklad hry „Vieš pokračovať?“

V hre „Vieš pokračovať?“ [11] je úlohou hráča hľadať slová, ktoré sa začínajú na zadané dve písmená. Prvé, hlavné slovo, je určené na začiatku hry. Druhé slovo sa musí začínať na posledné dve písmená prvého slova. Ďalšie slovo znova začína na posledné dve písmená predchádzajúceho slova a tak ďalej.

1.2.3 Piknik

Obr. 1.5: Hra „Piknik“

Piknik [9] je zaujímavá, návyková mobilná hra, ktorú si pre jej jednoduchosť obľúbili deti, ale i dospelí. Cieľom hry je spájaním písmen vytvoriť čo najviac zmysluplných

slov. Získavané body sú pre každého motiváciou a zvyšujúca náročnosť levelov podnecuje súťaživosť. V tejto graficky atraktívnej hre sa deti naučia logicky premýšľať a ďalšou výhodou je obohacovanie slovnej zásoby.

1.2.4 Word Search Pro

Obr. 1.6: Hra „Word Search Pro“

Word Search Pro [15] je hra, ktorú na Slovensku poznáme pod slovom Osemsmerovka. Táto populárna logická hra sa skladá z obrazca osemsmerovky - štvorca a zo zoznamu slov. V tele osemsmerovky sú náhodne zoradené písmená v stĺpcoch a riadkoch tak, že ju celú vyplnia. Cieľom hry je všetky slová zo zoznamu vyhľadať a vyškrtať [14].

1.3 Graf ako dátová štruktúra

Analýzou existujúcich hier môžme odpozorovať niekoľko znakov, ktoré majú hry spoločné. Jedným zo znakov je práca so slovami alebo písmenami. Je možné ich spájať, presúvať, dynamicky pridávať alebo odstraňovať. Takéto správanie sa podobá práci s dátovou štruktúrou *graf*. Graf slúži na reprezentovanie vzťahov medzi dvojicami objektov. Skladá sa z množiny vrcholov a hrán. Graf môže byť neorientovaný, čiže hrany nerozlišujú smer alebo orientovaný, potom sú spojovníkmi šípky.

Najčastejšie ho používame, keď potrebujeme vyjadriť rôzne vzťahy, napríklad mestá spojené cestami, či skupinu ľudí, ktorí sa navzájom poznajú.

V prípade hier so slovami, slová alebo písmená reprezentujú vrcholy, ktoré môžu byť spolu prepojené orientovanými alebo neorientovanými hranami, vyjadrujúcimi nejakú súvislosť medzi nimi.

1.4 Bakalárske práce

1.4.1 Detský mikrosvet ako motivačný nástroj

Cieľom tejto bakalárskej práce bolo vytvoriť prostredie, pomocou ktorého si žiaci môžu vytvoriť vlastný virtuálny mikrosvet. Tento edukačný softvér [23] podnecuje kreativitu a motiváciu ku učeniu, a to získavaním virtuálnych odmien. Vo svojej práci som sa inšpirovala práve serverovou časťou, ktorá je postavená na prostredí Node.js. Taktiež budem využívať komunikačný protokol WebSocket, ktorý sa bude používať na priebežnú komunikáciu medzi serverom a používateľmi. Pokiaľ žiak využije možnosť hrať sa sieťovú hru s viacerými hráčmi, protokol zabezpečí jej zosynchronizovanie a plynulé hranie pre všetkých pripojených hráčov. Cieľ sieťových hier so slovami je taktiež hrať sa hru okamžite, bez nutnosti prihlasovania, nebudem potrebovať perzistentnú vrstvu, ktorá by priebežne ukladala stav rozohratej hry. Hru, teda vytvorený graf, bude možné uložiť vo forme obrázku.

1.4.2 Pythonovský framework pre vytváranie hier

Bakalárska práca [20] bola venovaná vytvoreniu pythonovskej knižnice pre tvorbu hier pomocou modulu Tkinter, ktorej cieľovou skupinou sú programátori začiatočníci. Najdôležitejšími požiadavkami boli jednoduchosť a intuitívnosť, teda rýchle pochopenie princípov knižnice bez hlbšieho poznania zbytočne zložitých konceptov. Vo svojej bakalárskej práci využijem rozdelenie pri návrhu frameworku a taktiež, aký postup mám zvoliť pri požiadavkach na vytváraný framework. Súčasťou každého dobrého softvéru by mala byť dokumentácia vo forme príručky, videa alebo článku.

1.4.3 Webová aplikácia s využitím komunikačného protokolu WebSocket

Autor práce vytvoril jednoduchú online hru, pomocou ktorej ukázal, na akom princípe fungujú WebSockety a aké výhody má ich aplikovanie v praxi. Na vytvorenie ukážkovej aplikácie použil jazyk JavaScript a serverový programovací jazyk Node.js. Ten funguje na základe Single Event Loop a Non Blocking IO. Single Event Loop znamená to, že aplikácia využíva jedno vlákno, ktoré vykonáva úlohy a obsluhuje prichádzajúce spojenia. Non Blocking IO rieši problém, ktorý nastáva práve pri využívaní iba jedného vlákna. Aplikácia teda nie je spomalená, keďže operácie, ktoré trvajú dlhšie, sa vykonávajú asynchrónne. Teda napríklad operácie s databázou, či čítanie alebo písanie súborov sú vykonané osobitne. Pokým sa vykonáva nejaká úloha, voľné vlákno sa venuje iným úlohám [22].
Pri návrhu svojej knižnice som využila práve opis fungovania WebSocketov a Node.js.

1.5 Popis použitých technológií

V svojej práci používam štandardné technológie HTML, CSS, JavaScript a jQuery. Ďalej sú charakterizované neštandardné technológie a knižnice, ktoré aplikácia využíva.

1.5.1 TypeScript

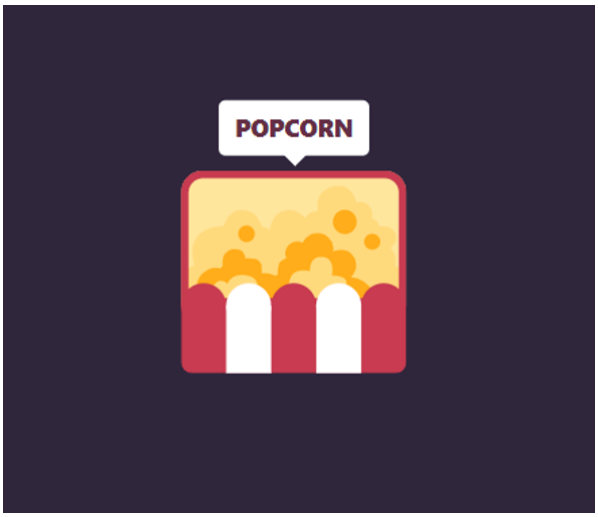
TypeScript [16] je open-source nadstavba JavaScript-u vyvíjaná spoločnosťou Microsoft. Využíva sa pri zložitejších aplikáciách a keďže zjednodušuje Javascript, kód napísaný v jazyku Typescript sa ľahšie udržiava. TypeScript je veľmi užitočný pri debuggovaní, pretože obsahuje transpilátor, ktorý automaticky kontroluje kód. Pri zistení syntaktickej chyby generuje kompilačné chyby. Vo svojej aplikácii využívam Typescript práve kvôli týmto výhodám. Prostredie, v ktorom aplikáciu vyvíjam, má priamu podporu pre TypeScript. Veľkým pozitívom je doplňovanie kódu pri jeho písaní. Všetky ostatné využívané technológie podporujú vyvíjanie v jazyku Typescript [21].

1.5.2 Cytoscape.js

Cytoscape.js [17] je grafická open-source knižnica napísaná v čistom jazyku Javascript. Umožňuje jednoducho vytvárať interaktívne grafy, využíva sa hlavne na ich vizualizáciu a analýzu. Knižnica obsahuje viaceré užitočné funkcie pre teóriu grafov. Je možné využiť ju na analýzu grafov v termináli alebo na webovom serveri. Cytoscape.js sa ľahko integruje do akejkoľvek aplikácie, keďže podporuje webové, ale aj mobilné prehliadače. Práve kvôli jednoduchej implementácii grafov a ich variabilite som sa sa rozhodla využiť Cytoscape.js vo svojej práci.

1.5.3 Popper.js

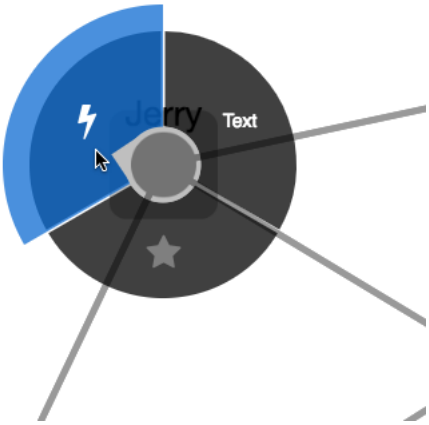
Popper.js [4] je knižnica, ktorá umožňuje umiestnenie vznášajúceho textu alebo HTML objektu blízko iného objektu, napríklad ikony alebo odkazu. Môže fungovať ako tooltip, teda predstavuje nápovedu, ktorá sa užívateľovi zobrazí, keď umiestni kurzor myši na špecifický objekt. Nástroj je modulárny, nezávislý a vhodný na súbežné použitie s grafickou knižnicou, napríklad Cytoscape.js. Túto knižnicu využívam pri editovaní dát, ktoré sa nachádzajú vo vrchole.



Obr. 1.7: Príklad využitia knižnice Popper.js

1.5.4 Cytoscape CxtMenu.js

Rozšírenie [2] knižnice Cytoscape.js vytvorí po podržaní vrchola grafický komponent - kontextové menu. Užívateľ si v kruhovom menu potiahnutím vyberie položku a následne sa vykoná určitý druh interakcie buď na vrchole alebo hrane grafu.



Obr. 1.8: Príklad využitia knižnice cytoscape-cxtmenu.js

1.5.5 Draggabilly

Knižnica [3], ktorá pomocou jQuery umožňuje transformovať statické HTML elementov na dynamické elementy. Používateľ môže pohybovať obrázkom alebo textom po osi x a y na obrazovke.
Hýbanie objektami grafu - vrcholov a hrán, rieši priamo knižnica Cytoscape.js, nie Draggabilly. Pomocou tejto knižnice je možné hýbať textom, čiže vytvoreným slovom.

1.5.6 Webpack

Webpack [5] pracuje s javascriptovskými modulmi a vytvára balíčky pre prehliadač. Ak máme kód napísaný modulárne, Webpack ho spracuje a vytvorí súbor, ktorý dokáže konkrétny prehliadač spracovať. Do jedného súboru môžme potom jednoducho importovať triedy a funkcie z iných súborov alebo z neho exportovať triedy a funkcie, ktoré potrebujeme použiť v inom súbore.

Obr. 1.9: Príklad využitia Webpacku

1.5.7 FileSaver.js

FileSaver.js [7] implementuje funkciu saveAs(), ktorú inak prehliadače nepodporujú. Rieši ukladanie súborov na strane klienta a je vhodný na ukladanie citlivých dát, ktoré nesmú byť poslané na externý server. Je možné zvoliť formát uloženia súboru. Z tohto dôvodu som ho využila vo svojej knižnici, práve na ukladanie grafu vo formáte PNG.

1.5.8 Node.js

Node.js [1] je Javascript interpreter na strane servera. Umožňuje programátorovi vytvárať vysoko škálovateľné aplikácie a písať optimálny kód, ktorý spracováva desiatky tisíc súčasných pripojení na jeden fyzický stroj. Pre spracovávanie údajov na strane servera sa v minulosti používali technológie ako napríklad PHP. Node.js je neustále rozširovaný o množstvo funkcií a vďaka jeho jednoduchému používaniu a modularite každým dňom rastie jeho popularita. Vo svojej aplikácii používam technológiu Node.js ako webový server hlavne kvôli tomu, ako jednoducho sa s ním pracuje a pre jeho rýchlosť. Nastavenie je nenáročné a umožňuje písanie kódu v Javascripte pre klienta ako aj server [25]. Taktiež som sa ho rozhodla používať kvôli jednoduchej implementácii WebSocket.

1.5.9 Node Package Manager

NPM (Node Package Manager) je správca javascriptovských balíčkov pre prostredie Node.js. Komponentami sú klient - príkazové riadky, nazývané *npm* a online databáza bezplatne publikovaných a platených balíčkov, ktoré sa nazývajú *npm registre*.

1.5.10 WebSocket

WebSocketová komunikácia je špeciálny typ sieťovej komunikácie, kde sa na komunikáciu používa vždy TCP protokol a používa sa event-driven programovanie, teda programovanie na základe udalostí. Takáto komunikácia povolí obojsmernú (full-duplex), asynchrónnu komunikáciu medzi klientom a serverom, pomocou ktorej je možné vytvoriť komunikáciu v reálnom čase. Komunikácia je vhodná napríklad

v multiplayer online hrách alebo pri online chatoch, kde je potrebná okamžitá obojsmerná interakcia medzi používateľmi a serverom.

1.5.11 Express

Express [6] je minimalistický, webový Node.js server. Rozšírenia, ktoré ponúka, sa ľahko nastavujú a používajú, preto serveru stačí nastaviť port, na ktorom bude počúvať, potrebné parametre a aplikácia je ihneď funkčná. Pre jeho nenáročnú inštaláciu a následné používanie je Express populárny a veľmi využívaný.
Využívam ho v serverovej časti aplikácie na komunikáciu medzi používateľmi a serverom.

Kapitola 2

Návrh frameworku

Bez do detailu prepracovaného návrhu by nemohla vzniknúť správne fungujúci framework. Táto kapitola bude venovaná práve tejto dôležitej časti celého frameworku. Jasne stanovené požiadavky a štruktúra tried sú kľúčom k úspešnej implementácii.

2.1 Hlavný cieľ frameworku

Framework by mal slúžiť na vytvorenie responzívnej, dizajnovo jednoduchej webovej aplikácie, ktorá bude pomocníkom pri práci s jazykom - spájanie písmen do slov, práca so slovami.

Vývoj sieťovej aplikácie môže byť zložitý proces, ktorý si vyžaduje prácu s webovým serverom. Cieľom frameworku je zjednodušiť tento proces, naprogramovanie jednoduchej aplikácie, teda hry, bude intuitívnejšie.

Hry so slovami sú určené pre každého, napríklad žiakov školského veku, preto sa predovšetkým dbá na nekomplikované, intuitívne ovládanie. Práca s aplikáciou by mala byť pre používateľa zábavou a jej ľahko zapamätateľné používanie umocňuje dobrý pocit z hry. Z tohto dôvodu sa v hrách využívajú na manipuláciu s grafom tlačidlá myši, respektíve dotyk na displeji mobilu, či tabletu.

Motiváciou pre hranie hry je možnosť voľby nepochybne obľúbeného režimu hry - multiplayer - hra určená pre viacerých hráčov. Hráči pripojení cez sieť môžu pracovať spolu v jednom tíme alebo sa môžu pri vykonávaní aktivity striedať.

Súčasťou knižnice budú ukážkové hry, ktoré sú zároveň návodom jej správneho používania. Používateľovi by z jednoduchých ukážok malo byť jasné, ako správne využívať funkcie a vytvárať jednotlivé objekty. Ukážky netvoria komplexnú hru, ale mali by slúšiť na to, aby vytvorenie podobných hier so slovami nebolo obtiažne. Vytvorením ukážkových hier môže užívateľ jednoduchou modifikáciou šablóny vytvoriť iné, podobné hry.

2.2 Základné charakteristiky frameworku

Samotnej implementácii by malo predchádzať jasné stanovenie charakteristík, ktoré budú zároveň požiadavkami na knižnicu. Tento krok povedie k úspešnému výsledku, pokiaľ budú požiadavky jednoznačné a zrozumiteľné. Následne budú aplikovateľné pri implementácii softvéru.

2.2.1 Trieda pre štruktúru *graf*

Základné metódy

Keďže hry, vytvorené pomocou nášho frameworku, sú založené na práci so slovami alebo písmenami, ktoré reprezentujú vrcholy prepojené hranami, potrebujeme vhodnú triedu pre prácu s elementami grafu. Pre programátora bude preto k dispozícii niekoľko elementárnych funkcií na manipuláciu so štruktúrou graf, ktoré tvoria framework:

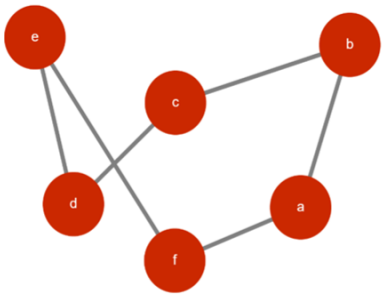
- Pridávanie vrcholov grafu
- Pomenovanie vrcholu a zmena názvu už existujúceho vrcholu
- Odstraňovanie vrcholov grafu
- Vytváranie orientovaných alebo neorientovaných hrán v grafe

2.2.2 Vizualizácia grafu

Na zobrazenie dátovej štruktúry graf, je nutné zvoliť vhodné grafické rozhranie tak, aby boli realizovateľné základné metódy z predchádzajúcej podčasti. Využívať budeme funkcionalitu grafickej knižnice Cytoscape.js a Popper.js, ktoré sú výbornou pomôckou pri tvorbe grafov a ich následnej manipulácii. Používateľ môže využívať všetky funkcie nášho frameworku na tvorbu sieťových hier, aj bez akýchkoľvek znalostí týchto knižníc, keďže práca s nimi bude prebiehať na pozadí.

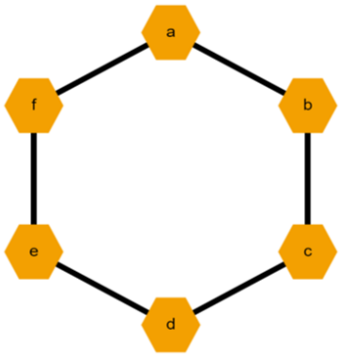
Štruktúra objektu

Objekt tejto knižnice je umiestnený v HTML elemente. Graf preto môže byť umiestnený buď na celej ploche okna, v ktorom bude bežať naprogramovaná hra alebo len na jeho časti. Používateľ si môže nastaviť vizuálne vlastnosti, je možný výber rozmiestnenia vrcholov, ich farba, tvar a farba textu. Prispôsobiť môže tiež hrúbku a farbu hrán a má možnosť zvoliť, či bude graf obsahovať orientované (so šípkou pri koncovom vrchole) alebo neorientované hrany (bez šípky).

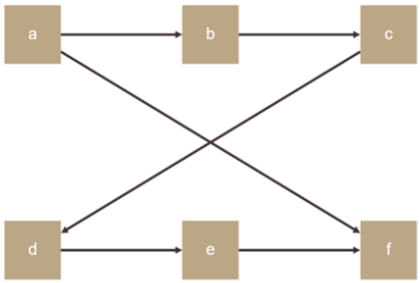


(a) Náhodné rozmiestnenie (random)

(b) Umiestnenie v strede (preset)



(c) Rozmiestnenie v kruhu (circle)



(d) Rozmiestnenie do mriežky (grid)

Obr. 2.1: Príklad rozmiestnenia vrcholov grafu

Odchytávanie udalostí

Vo frameworku na tvorbu slov je odchytávaných niekoľko typov udalostí, ktoré vznikajú v bežiacej grafickej aplikácii aktivitami používateľa, konkrétne klikanie myšou. Nasledujúce možnosti môže používateľ využiť, aby vrcholy grafu reagovali na aktivitu hráča.

- *tap*: klik myšou alebo krátky dotyk na obrazovke
- *taphold*: podržanie ľavého tlačidla myši alebo podržanie prsta na obrazovke

Medzi výhody knižnice Cytoscape.js patrí bezstarostnosť pri odchytávaní zmien súradníc vrcholov. Funcionalita posúvania vrcholov po x-ovej a y-ovej osi je zabudovaná priamo v danej knižnici a používateľ nemusí robiť dodatočné nastavenia, aby hráč mohol s vrcholmi ľubovoľne pohybovať. Táto funkcionalita je teda zabudovaná do nášho frameworku v rámci knižnice Cytoscape.js.

2.2.3 Sieťová aplikácia

Keďže primárnym cieľom frameworku je vytváranie sieťových responzívnych aplikácií, dôležitou časťou návrhu bude zvolenie vhodných prostriedkov na jeho realizovanie. Voľba webovej aplikácie, ako výsledného diela frameworku, bolo v dôsledku viacerých výhod, ktoré sa s ňou spájajú. Webová aplikácia je poskytovaná užívateľom z webového servera cez sieť Internet. Jej užívateľské rozhranie funguje rovnako kdekoľvek bez nutnosti inštalácie špeciálneho softvéru. Výhodou, ako pre užívateľa,

tak aj pre prevádzkovateľa aplikácie je jednoduchá aktualizácia. Tá sa vykonáva len na serveri, kde webová aplikácia beží.

V súčasnosti je najvyužívanejším nástrojom na tvorbu interaktívnych aplikácií programovací jazyk JavaScript. Jeho popularita medzi programátormi neustále narastá a medzi nesporné výhody patrí rýchlosť, výkon a funkčnosť, ktoré sú poskytované používateľovi na viacerých zariadeniach, prehliadačoch a operačných systémoch, pričom zdrojový kód ostáva rovnaký. JavaScript, spolu s kombináciou nespočetne veľa dostupných knižníc, umožňuje vývojárom vytvárať multiplatformový softvér, ktorý dokonale spĺňa všetky stanovené požiadavky responzívnej webovej aplikácie. Jednou z najpoužívajších knižníc je TypeScript, nadstavba zjednodušujúca syntax JavaScriptu. Pri vyvíjaní väčších systémov alebo štruktúr modulov je dobrým pomocníkom, ktorý pomôže zachytiť veľa chýb. TypeScript bude preto hlavný programovací jazyk, využívaný pri implementácii frameworku.

Webové hry, vytvorené pomocou nášho frameworku, budú ponúkať okrem základnej verzie hry *singleplayer* aj *multiplayer* verziu.

2.3 Požiadavky na používateľa frameworku

Hlavnou cieľovou skupinou „frameworku na tvorbu sieťových hier so slovami“ sú tvorcovia edukačného softvéru. Nemusí nutne ísť o profesionálov v tejto oblasti (tí pravdepodobne využijú profesionálne nástroje), ale napríklad o programátorsky zdatných učiteľov, či budúcich učiteľov, alebo iných ľudí s programátorskými zručnosťami a záujmom tvoriť jednoduché sieťové edukačné aplikácie pre deti.

Stanovením troch charakteristík frameworku nám vyplynulo niekoľko požiadaviek, ktoré má jeho používateľ spĺňať.

Používanie frameworku je možné iba so znalosťou objektovo orientovaného programovania v jazyku JavaScript alebo TypeScript. Používateľ vytvára vlastné triedy a využíva prácu s objektami. Je tiež potrebné, aby mal používateľ znalosť využívania asynchrónnych funkcií v jazyku JavaScript a nutné je aj orientovanie v terminológii z oblasti webových technológií - HTML, CSS.

2.4 Návrh tried

2.4.1 Hlavné triedy

1. **Graph**
   Hlavným objektom knižnice bude objekt Graph. Jedná sa o graf vykresľovaný knižnicou Cytoscape.js. Grafom je reprezentovaná najväčšia časť hracej plochy, ktorá sa hráčovi zobrazuje. Uchováva zoznam vrcholov a hrán, ktoré ho tvoria. Používateľ si môže zvoliť, aké bude mať graf rozloženie (kruh, mriežka, nádodné rozmiestnenie vrcholov). Celá funkcionalita aplikácie je postavená na práci s týmto objektom.

   Framework na tvorbu hier so slovami umožňuje do grafu pridávať nové vrcholy alebo odstraňovať vrcholy, taktiež je možné zamknúť vrchol, aby s ním používateľ nemohol pohybovať. Webová aplikácia je vymyslená tak, že trieda Graph je singleton. Na začiatku sa vytvorí jeden objekt, s ktorým bude programátor

pracovať podľa svojho uváženia. Graph reaguje na podnety, teda udalosti - kliknutie alebo dlhé podržanie myši, respektíve prsta na dotykovej obrazovke.

2. **Node**
   Node, teda vrchol grafu, je elementárna jednotka, ktorá tvorí graf. Obsahuje informáciu o editovateľnosti, teda či používateľ môže meniť jeho názov. S vrcholom sa dá pohybovať, a je možné nastaviť farbu, tvar a veľkosť. Programátor má taktiež možnosť nastaviť atribúty názvu vrchola, teda veľkosť a farbu písma. Objekt Node uchováva zoznam objektov Node, s ktorými je prepojený.

3. **Edge**
   Hranu grafu budeme reprezentovať objektom typu Edge. Spája medzi sebou dva vrcholy, teda objekty Node. Podobne ako pre vrchol, aj hrane je možné nastaviť farbu a hrúbku. Uchováva dvojicu objektov typu Node, teda dva vrcholy, ktoré medzi sebou prepája.

4. **Socket**
   Trieda, ktorá zabezpečí komunikáciu medzi hráčmi pri prepojení dvoch hráčov. Uchováva konkrétny socket knižnice socket.io a počet aktuálne pripojených hráčov, pri čom sa toto číslo aktualizuje, pokiaľ sa pripojí nový hráč.

5. **Game**
   Game je abstraktná trieda, ktorej funkcie slúžia na jednoduché vytvorenie ľubovoľnej hry. Použitím už naprogramovaných funkcií tejto triedy má používateľ knižnice vopred danú šablónu, podľa ktorej sa môže riadiť. Podtriedou triedy Game je teda hra, ktorá využíva funkcionalitu nadtriedy Game.

6. **Player**
   Trieda Player reprezentuje aktuálne pripojeného hráča, ktorý obsluhuje aktivitu. Uchováva meno hráča a konkrétny objekt typu Socket, ktorý odpočúva potrebné udalosti a posiela ich na server. Pri type hry multiplayer nastáva komunikácia medzi serverom a všetkými pripojenými hráčmi.

7. **FileData**
   Na uchovávanie dát prečítaných zo súboru vo formáte .json, slúži trieda FileData. Dátami, ktoré obsahuje, je *vstup*, teda slová alebo písmená, ktoré vytvoria vrcholy grafu a*odpovede* potrebné pri vyhodnocovaní výsledkov hry. Pristupovať k nim môžeme pomocou funkcií *getInput()* a *getAnswers()*.

2.4.2 Vedľajšie triedy

1. **SessionStorage**
   Pravdepodobne najjednoduchšou a najrozšírenejšou technológiou lokálneho úložiska je WebStorage. WebStorage definuje dve úložiská, ktoré sa líšia perzistenciou dát. Prvé úložisko, LocalStorage, ukladá dáta dovtedy, pokiaľ nie sú vymazané skriptom. Vo triede svojej knižnice využívame druhý typ úložiska, a to SessionStorage, ktoré ukladá dáta len počas trvania jednej relácie (session). Dáta uchováva, až pokiaľ používateľ nezavrie webový prehliadač alebo okno s danou stránkou. Trieda umožňuje programátorovi uložiť dvojicu kľúč a hodnota a následne ich môže získať späť.

V úvode každej hry je potrebné zadať meno hráča a typ hry - singleplayer (jeden hráč) alebo multiplayer (viacero hráčov). Pre tento účel sme zvolili dialóg, v ktorom hráč zadá potrebné informácie. Keďže po potvrdení dialógu sa zobrazí nová HTML stránka a zvolené dáta sú stratené, využívam objekt tejto triedy práve pri ukladaní mena hráča a typu hry.

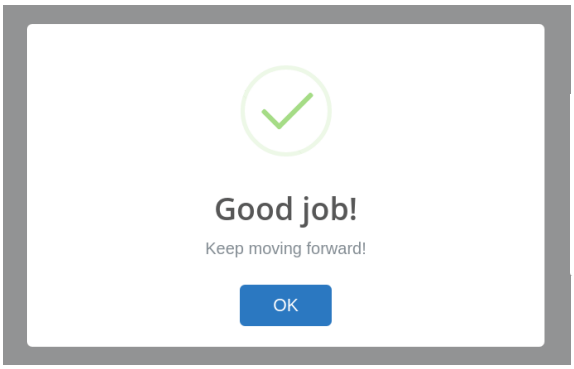
Tento problém by bolo možné vyriešiť aj ukladaním do databázy, ale pre účely jednoduchých ukážkových hier nám postačí úložisko SessionStorage.

2. **Dialog**
   Dialog je trieda, ktorá slúži používateľovi frameworku na zobrazenie spätnej väzby hráčovi. Ponúka tri metódy, ktoré môže programátor zmeniť podľa svojho uváženia.

   Metóda *callDialog* zobrazí dialógové okno, v ktorom sa nachádza text uvedený ako parameter a tlačidlo na zatvorenie okna.
   Parametrami druhej metódy *callDialogWithIcon*, je text zobrazený v dialógu a typ ikony, pri čom má používateľ na výber z troch typov, podľa správy, ktorú chce zobraziť hráčovi. Pokiaľ bola nejaká udalosť úspešná, programátor použije ikonu zobrazujúcu vybavenú položku ✔, parameter je typ ikony **„success“** Parameter **„warning“** , je možné využiť pri type udalosti, kde je potrebné zobraziť varovanie s ikonou **!** . Posledný typ spätnej väzby, ktorú je možné využiť, je chybné hlásenie s ikonou ✖, používateľ zadá ako parameter metódy **„error“**.

   Keďže hráč zadáva v úvode každej hry svoje meno a typ hry, singleplayer alebo multiplayer, vytvorili sme metódu, slúžiacu na zobrazenie dialógu, v ktorom hráč zadá potrebné informácie. Jej parametrom je názov HTML súboru, teda stránka s hrou, ktorú si hráč vybral.

   Pre účel zobrazovania dialógového okna, sme zvolili nenáročnú, graficky vkusnú knižnicu napísanú v jazyku JavaScript a metódy triedy Dialog sme v zdrojovom kóde okomentovali.



(a) Dialóg s ikonou

(b) Dialóg v úvode hry

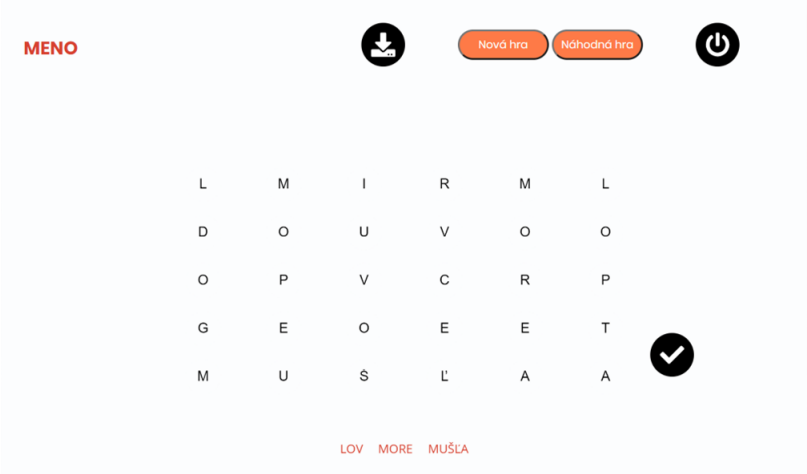
Obr. 2.2: Príklad dialógových okien

2.5 Práca so súborom

Užívateľ nastavuje obsah grafu pomocou jednoduchého súboru *.json*, ktorý bude vedieť framework prečítať, a následne riadne spracovať. Na korektné prečítanie súboru je nevyhnutné, aby boli splnené viaceré podmienky. Framework má predpísanú formu obsahu inicializačného súboru, v ktorom objekt typu JSON obsahuje zoznam objektov reprezentujúcich levely hry.

```
{
    "hra": [
        { "vstup":"L_M_I_R_C#D_O_I_V_A#O_E_V_C_T#G_M_O_E_E#U_E_K_V_A", "odpovede": ["MICE", "DOG", "CAT"] },
        { "vstup":"L_M_I_R_M_L#D_O_U_V_O_O#O_P_V_C_R_P#G_E_O_E_E_T#M_U_Š_Ľ_A_A", "odpovede": ["LOV", "MORE", "MUŠĽA"] }
    ]
}
```

Obr. 2.3: Príklad obsahu inicializačného súboru



Obr. 2.4: Graf vytvorený z inicializačného súboru

Používateľ zadá v časti *„vstup“* reťazec, v ktorom sú slová alebo písmená oddelené oddeľovačmi. Oddeľovač _ reprezentuje prirodzenú čiarku medzi znakmi v reťazci, každé slovo alebo písmeno, ktoré je rozdelené znakom _, tvorí v grafe jeden vrchol. Reťazec, v ktorom sa nachádza oddeľovač #, reprezentuje rozloženie výsledného grafu. Pokiaľ hráč zvolí rozloženie vrcholov do tvaru mriežky, oddeľovač # posunie nasledujúce slová alebo písmená na nový riadok.

Druhou podčasťou inicializačného súboru je zoznam odpovedí ku konkrétnemu levelu hry. V zozname sa slová alebo písmená oddeľujú čiarkou.

Pre korektné vykreslenie grafu odporúčame programátorovi využiť predpripravený .json súbor, nachádzajúci sa v priečinku priložených súborov k nášmu frameworku. Tento súbor slúži používateľovi ako šablóna, ktorej správny formát zabezpečuje korektné vytvorenie grafu.

2.6 Nastavenie udalostí po kliknutí na tlačidlo

Užívateľ frameworku má možnosť nastaviť metódy, ktoré sa vykonajú, keď hráč vytvorenej sieťovej hry klikne na HTML element umiestnený v grafickej ploche.

Naprogramovanie tlačidla je realizované pomocou dostupnej metódy, ktorej prvým parametrom je id alebo trieda HTML elementu (, <button>). Druhý parameter tvoria konkrétne metódy a potrebné parametre vo formáte dátovej štruktúry *Map (asociatívne pole)*, pričom kľúčom je metóda a hodnotou zoznam parametrov metódy. Ak je hodnotou prázdny zoznam, potom je metóda bez parametrov.

```
1     graph.setOnButtonClick("#add", { "add": ["", true],
2                                "connectFirst": [] });
3     game.setOnButtonClick("button#newGameBtn", { newGame: [] });
```

2.7 Návrh abstraktnej triedy *Game*

Programátor využíva potrebné metódy knižnice a je prakticky hlavným vývojárom celej responzívnej sieťovej aplikácie. Na to, aby bola webová hra plne funkčná, musí programátor správne využiť metódy našej triedy, ktorá pracuje s dátovou štruktúrou *graf*. Zjednodušením je predpripravená trieda, predstavujúca šablónu. Používateľ frameworku v nej môže pozmeniť alebo doplniť funkcionalitu niektorých metód, podľa potreby konkrétnej, vyvíjanej hry.

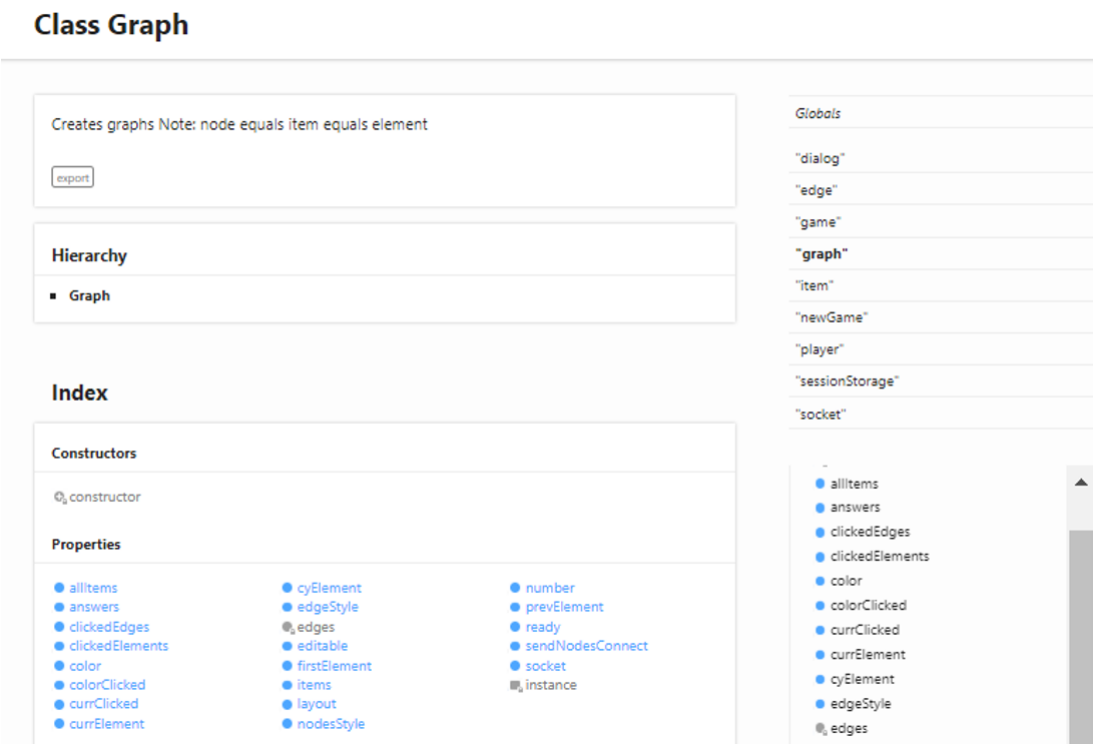
Základné metódy

- Vytvorenie grafu s ľubovoľnými atribútmi
- Nastavenie nového, pripojeného hráča
- Spracovanie kliknutých vrcholov, napríklad kontrola správnosti a následné vyhodnotenie hry
- Nastavenie slova alebo písmena, ako správne uhádnutého
- Nastavenie slova alebo písmena, ako nesprávne uhádnutého
- Naprogramovanie vlastného tlačidla, pomocou dostupných metód triedy

2.8 Dokumentácia zdrojového kódu

Neoddeliteľnou súčasťou frameworku je zrozumiteľná dokumentácia jednotlivých funkcií. Na tento účel sme zvolili TypeDoc, ktorý konvertuje komentáre v typescriptovom zdrojovom kóde. Funguje cez Node.js a je dostupný na inštaláciu ako NPM balíček (1.5.9).

Formát vygenerovaného súboru si používateľ môže zvoliť, výstupom je buď HTML dokumentácia, alebo model vo formáte JSON. Pre dokumentáciu funkcií frameworku, ktoré bude používateľ využívať, sme zvolili možnosť prehľadného HTML dokumentu.



Obr. 2.5: Ukážka triedy *Graph* vo vygenerovanom dokumente

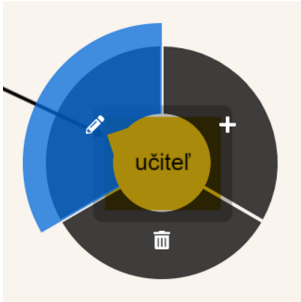
2.9 Ukážkové hry

V tejto časti si predstavíme niekoľko jednoduchých hier, ktoré vytvoríme pomocou nášho frameworku. Hráč má možnosť uložiť graf každej hry v akomkoľvek okamihu ako obrázok vo formáte PNG. Vytvorený graf sa stiahne do zariadenia stlačením ikony , v hornej časti grafickej plochy. Stlačením tlačidiel, nachádzajúcich sa v tej istej časti plochy, sa spustí nový alebo náhodný level hry.

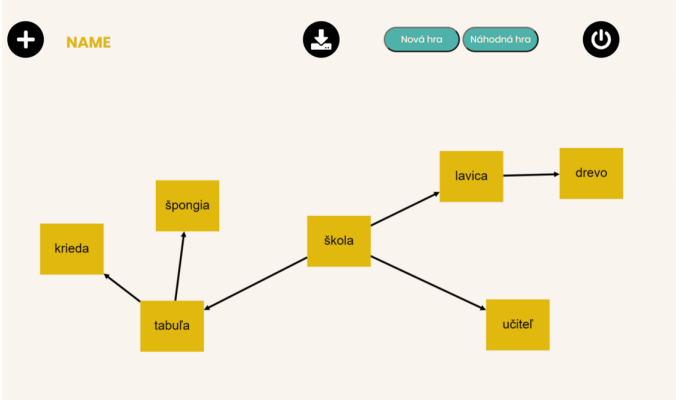
2.9.1 Myšlienková mapa

Hra je postavená na princípe vytvárania myšlienkovej mapy. Spája informácie okolo jedného hlavného slova a vytvára s ním súvisiace prepojenia. Na začiatku hry sa teda v strede grafickej plochy zobrazí slovo a úlohou hráča je pridávať nové vrcholy pomocou tlačidla tak, aby s ním významovo súviseli.
Pri dlhom kliku myšou sa zobrazí kruhové menu, ktoré ponúka 3 možnosti výberu:

- Vrchol je možné vymazať nadídením na ikonu .
- Vrcholu môžeme upraviť názov pomocou ikony .
- Vrcholu pridáme susedný vrchol po nadínení na ikonu .



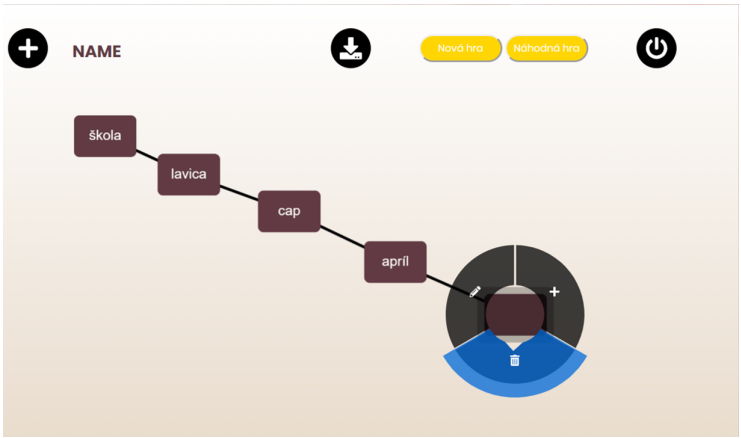
Obr. 2.6: Kontextové menu



Obr. 2.7: Hra „Myšlienková mapa“

2.9.2 „Vieš pokračovať?“

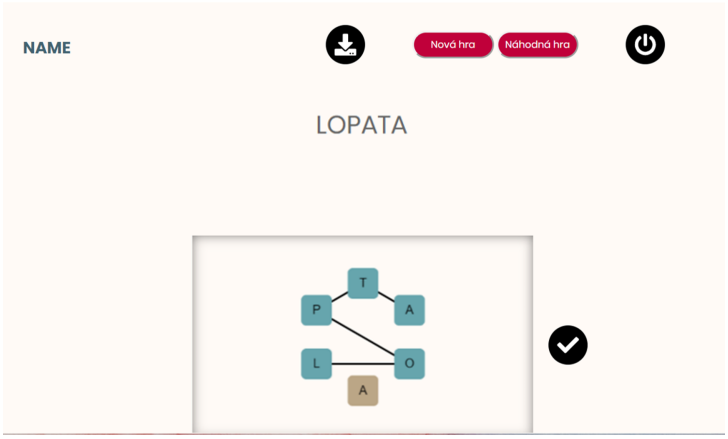
Cieľom hry je vytvárať reťaz slov podľa pravidla, že prvé dve písmená nasledujúceho slova sú totožné s poslednými dvomi písmenami predchádzajúceho slova. Po kliknutí na ikonu ⊕ , môže hráč pridať nový vrchol, spojený neorientovanou hranou s predošlým vrcholom. Rovnako ako v hre „Myšlienková mapa“, sa po dlhom kliku myši zobrazí kontextové menu, v ktorom si hráč vyberie vymazanie, zmenu názvu alebo pridanie nového vrchola. Hlavné slovo zobrazené na začiatku hry je prečítané z textového súboru.



Obr. 2.8: Hra „Vieš pokračovať?“

2.9.3 „Piknik“

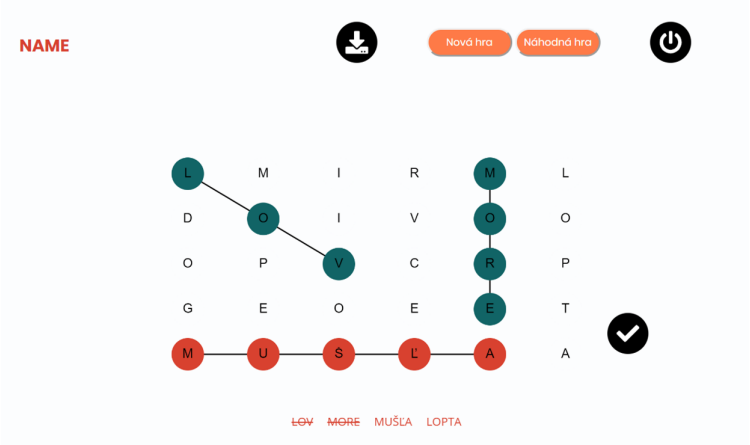
cieľom hry „Piknik“ je spojiť písmená do zmysluplného slova, ktorého správnosť sa vyhodnotí, ak hráč stlačí ✔. Slovo označené ako správne je zobrazené nad časťou, v ktorej sa nachádza graf, ktorého vrcholmi sú jednotlivé písmená. V textovom súbore ku každému levelu tejto hry musia byť zadané vstupné písmená, z ktorých sa majú slová skladať a všetky zmysluplné slová, ktoré sa dajú zložiť.



Obr. 2.9: Hra „Piknik“

2.9.4 Osemsmerovka

V poslednej ukážkovej hre sú písmená zobrazené v mriežke tak, ako boli prečítané z inicializačného súboru. Hráč skladá z písmen slová, ktoré sa nachádzajú v dolnej časti scény. Kliknutím na ikonu ✔ sa skontrolujú spojené písmená. Po úspešnom pokuse sa slovo vyškrtne, no ak slovo nie je správne uhádnuté, namiesto vyškrtnutia je zobrazené dialógové okno, oznamujúce nesprávny pokus.



Obr. 2.10: Hra „Osemsmerovka“

Kapitola 3

Implementácia

Nasledujúca kapitola bude venovaná implementácii niektorých častí frameworku. Detailnejšie je popísaná funkcionalita a vytvorenie ukážkových hier.

3.1 Funkcie triedy Graph

3.1.1 Funkcie na prácu s celým grafom

- getGraph() - vráti objekt grafu
- getGraphJSON() - vráti graf v JSON formáte
- recreateGraphJSON(graph: string) - vytvorí graf zo špecifického JSON formátu
- checkIfExists() - na pozadí priebežne zisťuje, či je graf vytvorený
- createGraphFromFile(file: string, nodeStyle: string[ ], edgeStyle: string[ ], lay: string, editable: boolean = false) - vytvorí graf zo súboru, parametrami sú štýl vrcholov, štýl hrán, rozloženie vrcholov (kruh, mriežka, náhodné rozmiestnenie, stred) a editovateľnosť vrcholov grafu
- clearGraph() - vymaže všetky vrcholy a hrany grafu

3.1.2 Funkcie na prácu s vrcholmi grafu

- addNode(id: number, str: string, editable: boolean = false) - pridá vrchol s daným id do grafu, určí sa aj jeho názov a to, či je vrchol upravovateľný
- removeNode(str: string) - odstráni vrchol s daným menom z grafu
- getNodes() - vráti elementy (vrcholy) grafu
- setNodes(nodes: Node[ ]) - nastaví elementy (vrcholy) grafu
- getClickedNodes() - vráti zoznam elementov, na ktoré užívateľ klikol
- setColorClickedNodes(color: string) - nastaví farbu aktuálne kliknutým vrcholom

- getClickedNodesName() - vráti zoznam názvov elementov, na ktoré užívateľ klikol
- clearClickedNodes() - odstráni elementy zo zoznamu kliknutých elementov
- unsetClickedNodes() - vráti stav kliknutých vrcholov na počiatočný, odstráni prepojenie hranou a zmení farbu vrchola
- lockNodes() - zamkne všetky elementy grafu, vrcholmi nie je možné naďalej hýbať
- setUnclickableNodes(elements: Node[ ]) - pre každý element z daného zoznamu, nastaví zákaz kliknutia, po kliknutí na jeden z týchto elementov sa nevykoná žiadna aktivita
- setAllNodesUneditable() - nastaví všetky vrcholy grafu ako needitovateľné, nie je možné upraviť ich názov, ani ich vymazať.
- getFirstNodeName() - vráti vrchol, ktorý do grafu pridaný ako prvý
- getPreviousNodeName() - vráti vrchol, ktorý bol pridaný pred naposledy pridaným vrcholom
- getCurrNodeName() - vráti naposledy pridaný vrchol grafu

3.1.3 Funkcie na prácu s hranami grafu

- addEdge(item1: string, item2: string) - spojí vrcholy, ktorých názvy sú *item1* a *item2*
- removeEdge(item1: string, item2: string) - odstráni hranu, ktorá spája vrcholy s názvom *item1* a *item2*

3.1.4 Funkcie na odchytávanie udalostí myši

- onNodeClick(color: string) - nastaví všetkým vrcholom grafu aktivitu, ktorá nastane po kliknutí na vrchol, vrchol sa zafarbí na zvolenú farbu
- onNodeClick_Connect(color: string) - nastaví všetkým vrcholom grafu aktivitu, ktorá nastane po kliknutí na vrchol, vrchol zafarbí na zvolenú farbu a spojí s posledným vrcholom, na ktorý hráč klikol
- setOnButtonClick(div: string, args:) - nastaví, aké funkcie sa vykonajú po kliknutí na tlačidlo, parametrami je *div* - HTML element, *args* - funkcie vo formáte *Map (asociatívne pole)* 2.6

3.1.5 Funkcie na spracovanie dát zo súboru

- readDataByLevel(level: number) - vráti všetky dáta prečítané zo súboru podľa levelu
- getAnswers() - vráti zoznam odpovedí (slov), ktoré boli prečítané zo súboru

3.1.6 Funkcie na komunikáciu so serverom

- send(evt: string, JSON: string) - odošle serveru správu, *evt* je názov udalosti, *JSON* sú dáta, ktoré sa odošlú

3.2 Import a export modulov

Pod pojmom modulácia, sa v programovaní rozumie rozdelenie zdrojového kódu do logických, nezávislých častí kódu, ktorý môžeme používať opakovane, kdekoľvek v projekte. Takéto časti sú stavebnými blokmi aplikácie a pomáhajú organizovať kód pre jeho lepšie porozumenie, udržiavanie, a samozrejme pre jednoduchšie odladovanie chýb.

Verzia JavaScriptu ES6 umožňuje importovať a exportovať funkcionalitu z modulov, ktorými môžu byť funkcie, triedy, konštanty, a v podstate čokoľvek, čo môžeme priradiť v JavaScripte do premennej. Keďže zo súboru nemusí byť exportovaná celá funkcionalita, môžeme si pojem modulácie predstaviť ako enkapsuláciu kódu. Exportované súbory sú akýmsi verejným programovacím rozhraním súboru, kde je dostupná len časť funkcionality, ktorá môže byť opakovane použitá kdekoľvek v inej časti projektu.

JavaScript pred vydaním štandardu ES2015 nepodporoval organizovanie programového kódu. Skripty boli v prehliadači spracovávané v poradí, v akom boli rozdelené (parsované) z HTML dokumentu. Node.js ale ponúkal modul CommonJS, ktorý mal tento problém vyriešiť. Jeho cieľom bolo rozdeliť zdrojový kód do modulov závislostí, ktoré vývojár podľa potreby nahrával a používal.

CommonJS má však viacero nevýhod. V jednom súbore môžeme importovať alebo exportovať maximálne jeden modul, funkcie, ani konštanty, sa nedajú pretransformovať do osobitných modulov. Prehliadače nemôžu používať moduly priamo, bez predchádzajúcej transpilácie, prekladač musí najskôr preložiť zdrojový kód z jedného programovacieho jazyka do iného.

Webpack [5] je balíčkovač (module bundler), ktorý sa snaží vytvoriť podporu rozdelenia programového kódu v prehliadači tak, že dovolí písať moduly ako pre Node.js interpret. Spracuje modulárne napísaný kód a vytvorí z neho balíček. Taktiež umožňuje ako moduly použiť NPM balíčky ako napríklad jQuery. V takom prípade nie je nutné pridávať ho, ako ďalší skript do HTML dokumentu.

Vo všeobecnosti môžeme export modulov rozdeliť na dva typy.

- **Export podľa názvu (named)**
  Takýchto exportov môže byť v module viacero.

```
1         export var GAME;
```

- **Predvolený export (default)**
  Export je v module jediný, {} sa nepíšu.

```
1         export default class Node {...
2         }
```

Importovať moduly môžeme nasledovne.

- **Import podľa názvu**
  Názov funkcie, triedy, či konštanty, sa nachádza v {} (curly brackets).

```
1         import { GAME } from "./game";
```

- **Import predvoleného exportu**
  {}, sa narozdiel od importu podľa názvu, nepíšu.

```
1         import Node from "./item";
```

- **Import celého modulu (*)**
  Ak je tried, konštánt a funkcií, ktoré chceme importovať priveľa, potom použijeme znak *. Takto získame prístup ku všetkým komponentom exportovaných zo súboru.

```
1         import * as graph from "../objects/graph";
```

Inštalácia WebPacku nie je náročná, stačí stiahnuť inštalačný balíček pomocou NPM (1.5.9) a vytvoriť konfiguračný súbor s potrebnými nastaveniami.

3.3 Komunikácia cez sockety

Framework na tvorbu hier so slovami ponúka vytvorenie sieťovej hry pre jedného alebo viacerých hráčov. Pre tento účel obsahuje náš framework triedu Socket 4, ktorá zabezpečuje komunikáciu medzi hráčmi pomocou socketov.

Používateľ frameworku má možnosť využiť niekoľko udalostí, ktoré odchytáva pripojený socket a odosiela na server. Tie sú dostupné v metóde *set* triedy Socket. Programátor odošle udalosť pomocou funkcie **send**, ktorej parametrami je názov konkrétnej udalosti a dáta vo formáte JSON reťazca.

Príklad použitia funkcie *send*:

```
1     graph.send("correct", '{"name" : "' + word + '"}');
```

Udalosť *„correct“* reprezentuje, že jeden hráč správne uhádol slovo s menom *name*, zadaným v JSON stringu. Správa s dátami sa odošle na server, po prijatí správy server odpovedá na správu (požiadavku) a v odpovedi pošle všetkým hráčom (socketom) prijatý JSON string, ktorého obsahom je slovo uhádnuté prvým hráčom.

V súbore *server.js*, sa odchytávajú odoslané správy.

```
1     client.on("correct", function (data) {
2         var d = JSON.parse(data);
3         io.sockets.emit("correct", d.name);
4     });
```

Každý hráč, teda pripojený socket, reaguje na správu od servera. Prijaté slovo je označené ako uhádnuté, farba vrchola s daným menom sa u všetkých pripojených hráčov zmení napríklad na zelenú.

```
1     this.socket.on("correct", function (data) {
2         GAME.setCorrect(data);
3     });
```

Používateľ nášho frameworku môže odosielať nižšie uvedené udalosti:

- *addNode* - pridanie nového vrchola
- *editNode* - úprava názvu vrchola
- *removeNode* - odstránenie vrchola
- *addEdge* - pridanie hrany
- *addEdgeToPrevNode* - pridanie hrany k vrcholu, ktorý bol pridaný pred posledným vrcholom
- *addEdgeToFirstNode* - pridanie hrany k prvému vrcholu v grafe
- *newGame* - nová hra
- *rndGame* - náhodná hra
- *correct* - správne uhádnuté slovo alebo písmeno
- *incorrect* - nesprávne uhádnuté slovo alebo písmeno
- *clickedNode* - kliknutie na vrchol
- *clickedNodeAddEdge* - prepojenie aktuálne kliknutého vrchola hranou s naposledy kliknutým vrcholom

Pokiaľ si to hra vyžaduje, programátor môže vytvoriť novú udalosť, ktorá sa odošle na server. V tom prípade je potrebné ju doplniť do metódy **set** a taktiež do súboru **server.js**.

3.4 Čítanie zo súboru - problémové riešenie

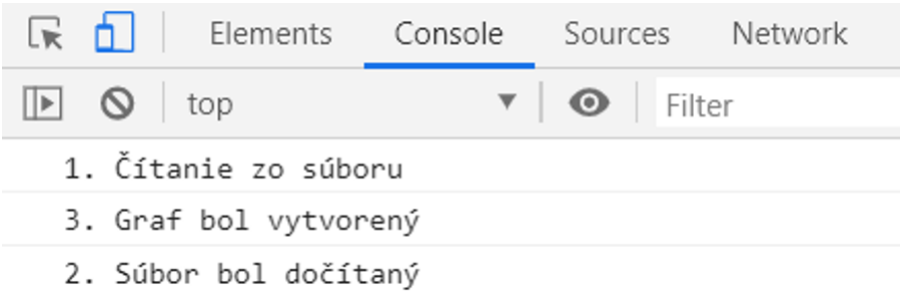
Najjednoduchší spôsob ako načítať súbory je použiť HTML dokument typu input. *<input type="file»* element podporuje každý prehliadač a používateľovi umožňuje načítať ľubovoľné súbory zo svojho zariadenia. Tento spôsob je však možné uskutočniť len pomocou prieskumníka súborov (file explorer), ktorý sa otvorí po udalosti vyvolanej užívateľom.

Čítanie súborov týmto spôsobom však nie je pre účel nášho frameworku vhodné, pretože prečítanie lokálneho .json súboru má fungovať na základe zadania jeho cesty, bez použitia externého užívateľského rozhrania.

JavaScriptovská knižnica jQuery obsahuje metódu na získavanie dát, ktorú využijeme.

```
1 $.get('file_to_read.txt', function(data) {
2     do_something_with(data)
3 });
```

Cieľom nášho frameworku na tvorbu sieťových hier so slovami je prečítať slová alebo písmená z textového súboru, spracovať ich a následne podľa nich vytvoriť graf. Očakávame, že nami implementovaná funkcia prečíta vstupný súbor pomocou jQuery metódy, iná funkcia dáta vhodne uloží do zoznamu objektov FileData a tretia funkcia z dát vytvorí a na grafickej ploche zobrazí graf. Pomôžeme si výpisom do konzoly.



Obr. 3.1: Výpis v konzole

Vidíme, že kontrolné výpisy nie sú zobrazené v očakávanom poradí. Program ako prvú funkciu vykonal vytvorenie grafu, bez predchádzajúceho prečítania súboru. Problémom tohto riešenia je v metóde *$.get(...)*, ktorá je asynchrónna V Node.js úlohy vykonáva jedno vlákno a operácie, ktoré môžu trvať dlhšie sa vykonávajú asynchrónne a spustené sú namiesto nej iné úlohy. Tento postup je oproti iným programovacím jazykom neštandardný, keďže operácie sa vykonávajú v poradí za sebou.

3.5 Čítanie zo súboru - výsledné riešenie

Predchádzajúci spôsob čítania súboru nefungoval z dôvodu Single Event Loop vykonávania úloh. Problém pri získavaní dát zo vstupného súboru nastáva, pretože táto úloha trvá dlho a začne sa vykonávať asynchrónne. Voľné vlákno sa zatiaľ venuje iným úlohám - vykonáva funkciu vytvorenia grafu.

Jedným z riešení tohto problému je využitie **Promise** spolu s **async a await** rozšíreniami JavaScriptu, čím sa zjednoduší spôsob, akým sa pracuje s asynchrónnymi volaniami.

Slovo async je označením pozastaviteľnej funkcie, v ktorej je následne možné použiť kľúčové slovo await. Automaticky sa potom do danej premennej priradí výsledok typu Promise.

Await prakticky znamená "počkaj, kým sa vykoná Promise a pokračuj". Async funkcia a await vracajú len typ Promise.

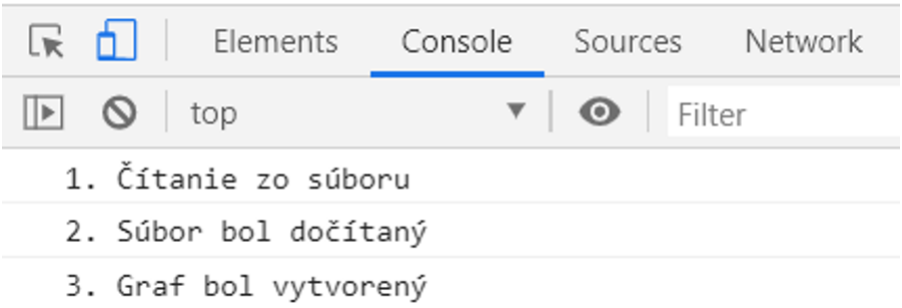
```
1 export async function checkIfExists() {
2   return new Promise((resolve) => {
3     var checkExist = setInterval(function () {
4       if (Graph.getInstance().ready == true) {
5         clearInterval(checkExist);
6         resolve("done!");
7       }
8     }, 200);
9   });
```

```
10 }
```

V našom prípade vytvoríme kontrolnú funkciu, ktorá zisťuje, či je súbor prečítaný, program funkciu na vytvorenie grafu ešte nevykonáva, ale čaká. Pre tento účel sme si vytvorili premennú *ready*, ktorá sa zmení na *true*, až keď sú zo súboru získané všetky dáta. Metóda *checkIfExists()* obsahuje časovač, ktorý túto premennú v pravidelnom intervale kontroluje.

```
1 async function createGraph() {
2   Graph.getInstance().setSocket();
3   await checkIfExists();
4   Graph.getInstance().createGraphCy();
5 }
```

Asynchrónna funkcia na vytvorenie grafu obsahuje kľúčové slovo await, program čaká, kým premenná *ready* nie je *true*, následne sa spustí funkcia na vytvorenie a zobrazenie grafu na ploche.



Obr. 3.2: Výpis v konzole

Poradie vykonávania funkcií teraz funguje tak, ako sme predpokladali.

3.6 Vyhodnocovanie funkcií zadaných formou reťazca

Medzi funkcionality našej knižnice patrí aj naprogramovanie vlastného tlačidla pomocou dostupných funkcií. Existuje niekoľko spôsobov, ako v JavaScripte môžeme vyhodnocovať funkcie typu string.

3.6.1 Eval

Ak je argumentom príkaz alebo postupnosť príkazov v JavaScripte, *eval()* ich vyhodnotí. Riziká však zvyčajne prevyšujú benefity jeho používania a vždy sa nájde iný, lepší spôsob, ako vyhodnotiť konkrétnu funkciu. Odporúča sa preto používať ho len v nevyhnutných prípadoch. Niekoľko rizík, ktoré môžu nastať používaním *eval()*:

- Bezpečnosť
  Používateľ môže ako dáta vstupu vložiť škodlivý kód, napríklad keď použije nekonečný cyklus ako prihlasovacie meno. Ak sa programátor rozhodne použiť funkciu *eval()*, je nutné zabezpečiť dôsledné ošetrenie všetkých používateľských vstupov.

- Hľadanie chýb
  Eval sa neodporúča používať aj kvôli náročnému odlaďovaniu kódu. Programátorovi nedovoľuje prechádzať ho postupne krok za krokom, preto to môže byť problematické pri bežnom vývoji aplikácie.

- Rýchlosť
  JavaScript je ako jazyk navrhnutý používať veľké množstvo rôznych dátových typov, čísla, funkcie, objekty, nielen typ string. Používanie *eval()* je preto omnoho pomalšie, ako vyhodnocovanie normálneho kódu v JavaScripte. Interpreter JavaScriptu funguje na princípe prekonvertovania kódu, ktorý predtým vyhodnotil do strojového jazyka. Pokiaľ sa vo funkcii *eval* zmení typ premennej, interpreter prehliadača je nútený opakovane spustiť kód, a to zapríčiní časovo náročnú kontrolu, či premenná existuje. Pri použití *eval* sa kód spúšťa viackrát a rovnaký kód musí byť preložený viackrát. Tento postup je preto veľmi pomalý a neúčinný.

Z vyššie uvedených dôvodov som sa rozhodla vo svojej práci nevyužiť tento typ vyhodnocovania funkcií.

3.6.2 Window

Bezpečnejším riešením na vyhodnocovanie funkcií typu string ako *eval* je použiť window objekt. Ten sa odkazuje na aktuálne HTML okno a všetky elementy v ňom. Objekt môže byť akéhokoľvek typu, aký si zvolíme, napríklad objektom triedy. Postup vyhodnocovania funkcií je jednoduchý. Ak je string funkcia, skontrolujeme, či je existuje funkcia s daným názvom, ktorá je dostupná z daného objektu a spustíme ju. Ak funkcia očakáva nejaké parametre, použijeme metódu *apply* a zoznam parametrov aplikujeme. V našom prípade voláme konkrétne funkcie typu string na objekt triedy Graph.

```
1     export function setOnButtonClick(div: string, args: {}) {
2       var thiss = Graph.getInstance();
3       $(document).ready(function () {
4         $(div).on("click", function () {
5           for (var key in args) {
6             var value = args[key];
7             var execute = thiss[key];
8             if (typeof execute === "function") {
9               execute.apply(thiss, value);
10            }
11          }
12        });
13      });
14    }
```

3.7 Abstraktná trieda *Game* ako šablóna

Implementované ukážkové hry, ktoré sú založené na manipulácii s dátovou štruktúrou graf, obsahujú metódy s podobnou, v určitých volaniach aj identickou funkcionalitou. Aby sme predišli vysokému výskytu duplicitného kódu, využili sme v našom

frameworku abstraktnú triedu. Tá slúži ako šablóna, vďaka ktorej má používateľ možnosť vytvoriť ľubovoľnú sieťovú hru.

3.7.1 Funkcie

- async setNewGame(file: string) - prečíta súbor a vytvorí nový graf s danými nastaveniami pre štýl vrcholov a hrán, používa sa v singleplayer verzii hry
- async recreateGame() - zreplikuje už vytvorený graf, používa sa vo verzii hry multiplayer
- setCorrect(word: string, color: string) - nastaví slovo ako správne, farba slova, sa nastaví podľa parametra *color*
- check() - programátor si túto funkciu prispôsobí vzhľadom na logiku hry, je možné ju použiť na kontrolu správnosti riešenia, ak to daná úloha vyžaduje
- newGameS() - vyberie náhodné slovo zo zoznamu objektov FileData, ktoré boli prečítané zo vstupného súboru a spustí novú hru
- randomGameS() - vyberie ďalšie slovo v poradí zo zoznamu objektov FileData a spustí novú hru
- async text(div: string) - ak je potrebné v hre využiť dodatočnú textovú informáciu, pomocou tejto funkcie nastavíme obsah HTML elementu, ktorý je vložený v HTML dokumente

3.7.2 Vytvorenie ukážkových hier

Formát HTML súboru hry

Na spustenie hry v prehliadači je potrebný HTML dokument, do ktorého pripojíme skript s vytvorenou hrou. Používateľ nášho frameworku má preto k dispozícii vytvorenú šablónu HTML súboru s názvom *template.html*, ktorú odporúčame využiť pri vytváraní novej hry. Súbor obsahuje element obsahujúci objekt grafu a taktiež tlačidlá na pridanie vrchola, uloženie grafu a voľbu novej hry. Predpokladáme, že práve predpripravené elementy <button> sú vhodné pri funkcionalite akejkoľvek hry a programátor ich môže využiť.

```
<div id="canvas" class='nameOfGame'>
    <div id="pane" class="nameOfGame">
      <div id="add" class="check">
        <i id="checkIcon" class="fas fa-plus"></i><span></span>
      </div>
      <h1 id="player"></h1>
      <div id="paneGame">
          <div id="save" class="check">
            <i id="iconCheck" class="fas fa-download"></i><span></span>
          </div>

        <div id="gameButtons" class='nameOfGame'>
          <button type="button" id="newGameBtn">Nová hra</button>
          <button type="button" id="rndGameBtn">Náhodná hra</button>
        </div>
        <div id="off" class="check">
          <a href="gamesPage.html">
            <i id="iconCheck" class="fas fa-power-off"></i><span></span>
          </a>
        </div>
      </div>

    </div>
    <div id="graph" class='nameOfGame'>
    </div>
```

Obr. 3.3: Šablóna HTML súboru

Vytvorenie hry „Myšlienková mapa“

Vytvoríme si nový súbor .ts s názvom hry - **mapa.ts** a vložíme obsah šablóny *template.ts*. Meno triedy zmeníme podľa práve vytváranej hry.

```
1     export class Mapa extends Game {...
2     }
```

V ďalšom kroku máme možnosť upraviť funkcionalitu dostupných metód triedy Game (3.7.1).

setNewGame(file: string)

Parametrom funkcie je názov .json súboru, z ktorého chceme prečítať slová a jeho obsah vyzerá nasledovne. Vytvorená hra bude pozostávať z troch levelov a v každom sa na začiatku zobrazí jedno zadané slovo. Odpovede ku každému levelu tejto hry nie sú potrebné, kontrolu pri uhádnutí slova preto nemusíme implementovať.

```
{
    "hra": [
        { "vstup":"škola", "odpovede": [] },
        { "vstup":"leto", "odpovede": [] },
        { "vstup":"Vianoce", "odpovede": [] }
      ]
}
```

Obr. 3.4: Textový súbor so slovami

Vo funkcii, ktorá nastavuje graf, zadáme, aký štýl majú vrcholy a hrany grafu. Vrcholu tvaru štvorca nastavíme šírku 100px, výšku 80px, konkrétnu farbu, ktorú zapíšeme v hexadecimálnej sústave. Písmo vo vnútri vrchola je biele, s veľkosťou 20px.

```
1     async setNewGame(file: string) {
2         var nodeStyle =
3         ["#e1b80d", "white",  "20px", "rectangle",
4             "100px", "80px"];
5         var edgeStyle = ["black", "3px", "arrow"];
6     }
```

Všetky dostupné funkcie na manipuláciu s grafom sú importované ako modul pod názvom **graph** a pristupujeme k nim cez tzv. bodkovú notáciu. Za meno modulu uvedieme prvok (v tomto prípade funkciu) z daného modulu.

Na vytvorenie grafu slúži funkcia *createGraphFromFile* a jej parametrami sú názov súboru, štýl vrcholov, štýl hrán, rozloženie grafu (mriežka, kruh, náhodné rozmiestnenie, stred) a atribút editovateľnosti vrcholov. Keďže je funkcia čítania zo súboru asynchrónna (3.5), musíme pridaním kľúčového slova await skontrolovať, v akom okamihu je graf pripravený na použitie. Funkcia nastavenia grafu teraz vyzerá nasledovne.

```
1     async setNewGame(file: string) {
2         var nodeStyle =
3         ["#e1b80d", "white",  "20px", "rectangle",
4             "100px", "80px"];
5         var edgeStyle = ["black", "3px", "arrow"];
6         graph.createGraphFromFile(file, nodeStyle, edgeStyle, "
              preset", true);
7         await graph.checkIfExists();
8     }
```

V strede grafickej plochy sa nachádza jeden vrchol s názvom „škola“. Funkcia grafu setAllNodesUneditable(), nastaví všetky vrcholy ako needitovateľné, pretože nechceme, aby hráč vymazal hlavný vrchol grafu alebo upravil jeho názov.
Pre účely ďalšieho a náhodného levelu hry ešte vrchol s názvom „škola“ nastavíme ako aktuálne slovo. Všetky prečítané slová z textového súboru uložíme do premennej typu pole *this.readData* a kedykoľvek k nej môžeme pristupovať pomocou funkcie *readData* alebo podľa aktuálneho levelu hry funkciou *readDataByLevel(level)*.

```
1     async setNewGame(file: string) {
2         var nodeStyle =
3         ["#e1b80d", "white",  "20px", "rectangle",
4             "100px", "80px"];
5         var edgeStyle = ["black", "3px", "arrow"];
6         graph.createGraphFromFile(nodeStyle, edgeStyle, "preset",
              true);
7         await graph.checkIfExists();
8         graph.setAllNodesUneditable();
9         this.word = graph.getNodes()[0];
10        this.readData = graph.readData();
11    }
```

Hra „Myšlienková mapa“ pozostáva z troch levelov, musíme preto zabezpečiť, aby sa hráč po skončení jedného levelu mohol preklikom dostať na nasledujúci.
Na tento účel slúžia metódy *newGame* a *randomGame* v triede Game, ktoré odstránia všetky elementy grafu. K dispozícii máme prázdny graf, do ktorého doplníme

vrcholy. Metódy triedy našej hry *newThisGame* a *randomThisGame* zmeníme tak, aby sa v ďalšom leveli zobrazilo slovo z prečítaného súboru. Získame vstup (input) z prečítaných dát podľa levelu a funkcia *setNodes* nastaví zoznam vrcholov grafu. V našom prípade má zoznam jeden prvok, vrchol s názvom slova. Nakoniec obnovíme elementy grafu a nastavíme slovo ako needitovateľné.

```
1       newThisGame() {
2         ...
3         this.level++;
4         var w = graph.readDataByLevel(this.level);
5         this.word = w.getInput()[0];
6         graph.setFirstNode(this.word);
7         graph.setNodes([this.word]);
8         graph.recreateGraph();
9         graph.setAllNodesUneditable();
10      }
11    }
```

Poslednou časťou vytvárania hry je vytvorenie objektu novej hry a nastavenie tlačidiel, ktoré sme V HTML dokumente vytvorili ako elementy.

Obr. 3.5: Tlačidlá hry

Ak hráč klikne na tlačidlo (#add), do grafu sa pridá nový editovateľný vrchol a vytvorí sa hrana, ktorý ho spojí s prvým vrcholom, ktorý sme pridali.

Funkcie prepínania nasledujúceho levelu, sa zavolajú po stlačení tlačidiel (#newGameBtn) a (#rndGameBtn) a tlačidlo (#save) slúži na uloženie grafu vo formáte PNG.

```
1     var game = new Mapa("client/files/fileMapa.json");
2     game.setNameOfPlayer("player");
3     graph.setOnButtonClick("#add", { "addNode": ["", true],
4         "addEdgeToFirstNode": [] });
5     game.setOnButtonClick("button#newGameBtn", { "newGame": [] });
6     game.setOnButtonClick("button#rndGameBtn",
7         { "randomGame": [] });
8     graph.setOnButtonClick("#save", { "saveImage": [] });
```

Keďže našu hru vytvárame v TypeScripte, pre správne fungovanie je nutné pridať jej názov, ako vstupný súbor do WebPack konfiguračného súboru.

Hru „Myšlienková Mapa“ sme úspešne vytvorili a nastaveniami zabezpečili jej správne fungovanie.

Vytvorenie hry „Vieš pokračovať?“

Hru „Vieš pokračovať?“ vytvoríme podobným spôsobom, ako sme opísali v predchádzajúcej podčasti. Zmenou oproti hre „Myšlienková Mapa“ je kontrola pri vytváraní

vrchola. Túto funkcionalitu môžeme nastaviť pomocou dostupnej, asynchrónnej metódy v triede Game, *checkNode*. Tá zabezpečí skontrolovanie obsahu pridávaného vrchola, ktorý dostala funkcia ako parameter a hráčovi zobrazí dialóg so spätnou väzbou, ak obsah nemá očakávaný tvar.

V prípade hry „Vieš pokračovať?“, nastane kontrola pridaného slova, ktorého prvé dve písmená musia byť rovnaké ako posledné dve písmená slova pridaného naposledy. Meno tohto vrchola zistíme pomocou metódy *getNodes*. Funkcia *checkNode* je asynchrónna, pretože najskôr skontrolujeme názov pridávaného vrchola a podľa výsledku tejto funkcie buď zobrazíme dialógové okno s chybou alebo pridáme nový vrchol s daným menom. Ak je výsledkom *resolve(„OK“)*, pridávané slovo je správne, v prípade, že je vo výsledku *resolve* iný reťazec, ten sa zobrazí v zobrazenom chybovom dialógu.

```
1     async checkNode(node: string) {
2         var promise = new Promise(function (resolve, reject) {
3           var words = graph.getNodes();
4           var last = words[words.length - 2].name;
5           if (last.length < 2) {
6             resolve('Slovo musi obsahovat aspon 2 pismena.');
7           } else {
8             var lastSub = last.substring(last.length - 2);
9             var lastW = node.substring(0, 2);
10            if (lastSub == lastW) {
11              resolve("OK");
12            }
13          }
14          resolve('Slovo musi zacinat na rovnake pismena
                predchadzajuceho slova.');
15        });
16        return promise;
17    }
```

3.8 Inštalácia

Inštalácia je prispôsobená používaniu nadstavby jazyka JavaScript, TypeScriptu. Pred samotnou inštaláciou knižnice predpokladáme, že na PC zariadení je už nainštalovaný program na editovanie súborov, platforma Node.js a TypeScript v rámci NPM. Nutným krokom k úspešnému používaniu nášho frameworku je stiahnutie priečinku so zdrojovými súbormi. V editore spustíme terminál a v priečinku nastavíme Node Package Manager.
Ďalej po jednom vložíme do terminálu nasledovné inštalačné príkazy:

- *npm install –save-dev express*
- *npm install –save-dev webpack*
- *npm install –save-dev webpack-hot-middleware*
- *npm install –save-dev draggabilly*

Inštalácia knižnice je dokončená. Pokiaľ aplikáciu skúšame na localhoste, pre spustenie stačí v termináli zadať príkaz *node server.js*.

Pre korektné fungovanie novovytvorenej hry je potrebné pridať do konfiguračného súboru *webpack.config.js* vstupný súbor .ts, ktorý WebPack spracuje a výstupný súbor pripojíme do HTML dokumentu ako script.

```
1         module.exports = {
2             entry: {
3                 main: path.join(__dirname,
4                         '/client/js/ts_main/main.ts'),
5                 pokracuj: path.join(__dirname,
6                         '/client/js/ts_main/pokracuj.ts'),
7                 osemsmerovka: path.join(__dirname,
8                         '/client/js/ts_main/osemsmerovka.ts'),
9                 slova: path.join(__dirname,
10                        '/client/js/ts_main/slova.ts'),
11                mapa: path.join(__dirname,
12                        '/client/js/ts_main/mapa.ts')
13             },
14            output: {
15                path: __dirname,
16                filename: "[name].js"
17             },
18             ...
19    }
```

Kapitola 4

Testovanie

Kapitola je venovaná testovaniu frameworku na tvorbu hier so slovami. Podrobnejšie opíšeme jeho priebeh a v závere odprezentujeme výsledky a taktiež niektoré nedostatky, ktoré sa pri testovaní ukázali.

4.1 Testovanie frameworku

Cieľom testovania bolo zistiť, či využívanie nášho frameworku zvládnu zdatnejší programátori. Zúčastnili sa ho dvaja študenti, ktorí spĺňajú požiadavky, vytýčené v časti 2.3. Na účely testovania sme vytvorili webovú stránku, na ktorej sú k dispozícii dostupné ukážkové hry. Stránka nie je optimalizovaná vzhľadom na detského používateľa, keďže cieľom práce nebolo vytvoriť kompletnú aplikáciu s hrami. Po načítaní stránky má hráč na výber z niekoľkých hier a po zvolení konkrétnej hry, sa mu zobrazí dialóg, v ktorom zadá svoje meno a typ hry.

4.1.1 Priebeh testovania

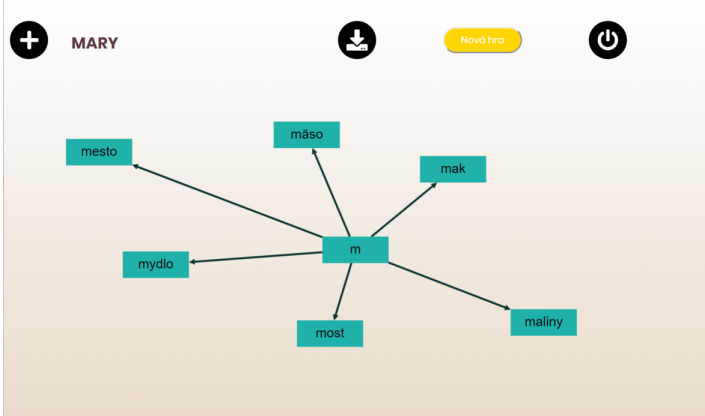
Framework sme testovali s každou osobou zvlášť, kvôli času potrebnému na dostatočné vysvetlenie jednotlivých funkcií šablóny, ktorú používateľ využije pri tvorbe sieťovej hry so slovami. Na začiatku sme tiež poskytli výklad ku korektnému obsahu inicializačného súboru, z ktorého je vytvorený graf a ku zdrojovému kódu ukážkových hier, aby mal tester predstavu o implementovaní konkrétnej hry.

Ďalšiu časť testovania tvorilo zadanie úloh. Jeho cieľom bolo vytvoriť dve funkčné webové hry, v ktorých hráč pracuje so slovami alebo písmenami. Vopred pripravené úlohy boli rôznej obtiažnosti a zadanie obsahovalo konkrétne požiadavky, ktoré má každá hra spĺňať.

Počas testovacej fázy som sa snažila programátorom nepomáhať s implementáciou, usmerňovala som ich len pri zadaniach a inštalácii frameworku do vytvoreného projektu. Obaja testeri pracovali vo voľne dostupnom editore *Visual Studio Code* a na svojom zariadení mali nainštalovaný Node.js a TypeScript.

Hra „Doplň slovo začínajúce na písmeno“

Zadanie hry
Na začiatku hry sa v strede hracej plochy zobrazí písmeno a úlohou hráča je dopĺňať slová, ktoré začínajú daným písmenom.



Obr. 4.1: Ukážka vytvorenej hry

Požiadavky:

- Hra má 3 levely, bez dopredu daných správnych odpovedí. V prvom leveli hráč tvorí slová začínajúce na písmeno **m**, v druhom na písmeno **a** a v treťom leveli na písmeno **p**.
- V hracej ploche sa nachádzajú tlačidlá na pridanie nového vrchola, na uloženie grafu a na návrat do menu hier. Ďalším tlačidlom sa hráč prepne na ďalší level hry.
- Vrcholy tvaru obdĺžnika majú farbu #**20B2AA**, písmo je **čierne**, veľkosti **20px**. Šírka každého vrchola je **100px** a výška **40px**. Hráč nemôže vrcholy vymazať, zmeniť už zadaný názov alebo pridávať susedný k vrcholu susedný vrchol.
- Hrana spájajúca vrcholy je **orientovaná** s hrúbkou **3px** a farbou #**0B332D**.

Vytváranie hry „Doplň slovo začínajúce na písmeno“

Cieľom implementácie prvej hry bolo overiť správne pochopenie funkcionality šablóny a na jej vytvorenie stačila nekomplikovaná obmena už vytvorenej hry a inicializačného súboru.

Testeri postupovali, ako sme predpokladali a našli podobnosť medzi vytváranou sieťovou hrou a ukážkovými hrami „Vieš pokračovať?“ a „Myšlienková mapa“. Jeden z nich správnou úvahou zistil, že je potrebné pozmeniť štýl vrcholov a hrán vo funkcii **setNewGame** podľa zadania a zmeniť funkciu **checkNode**, ktorá kontroluje názov pridaného vrchola. Prepísať bolo potrebné aj vstupný súbor tak, aby sa pre každý level hry zobrazilo dané písmeno.

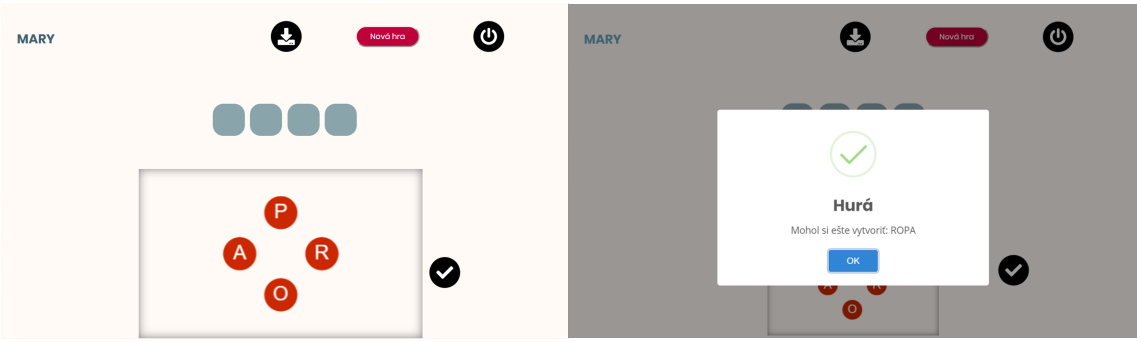
Posledným krokom bolo vytvorenie HTML dokumentu a pripojenie scriptu s hrou. Keďže jeden z testerov nemal skúsenosti s prácou s balíčkovačom WebPack, musela som znovu vysvetliť, ako správne pripojiť súbor tak, aby ho bolo možné spustiť v prehliadači.

Vytvorenie hry splňajúcej požiadavky zo zadania nezabralo študentom, ktorí testovali náš framework, veľa času.

Hra „Štyri písmená“

Zadanie hry

V úvode každého levelu hry sa hráčovi na grafickej ploche zobrazia štyri prázdne políčka, pod ktorými sú zoradené štyri písmená v kruhu. Cieľom hry je správne spojiť písmená do zmysluplného slova. Ak hráč uhádne slovo, objaví sa v prázdnych políčkach a zobrazí sa mu dialóg oznamujúci úspech a zvyšné slová, ktoré mohol spojením písmen vytvoriť.



(a) Úvod hry (b) Zobrazený dialóg

Obr. 4.2: Ukážka vytvorenej hry

Požiadavky:

- Hra má 5 levelov a ku každému je zadaný zoznam správnych odpovedí.
  V prvom leveli hráč tvorí slová z písmen **P, R, O, A**.
  Správne odpovede sú **ROPA, OPAR**.
  V druhom leveli tvorí slová z písmen **S, K, O, A**.
  Správne odpovede sú **KOSA, SAKO**.
  V treťom leveli hráč tvorí slová z písmen **L, P, I, A**.
  Správne odpovede sú **LIPA, PILA**.
  V štvrtom leveli hráč tvorí slová z písmen **K, K, S, O**.
  Správne odpovede sú **SKOK, KOKS**.
  V piatom leveli hráč tvorí slová z písmen **P, L, O, E**.
  Správne odpovede sú **POLE, OLEP**.
- V hracej ploche sa nachádzajú tlačidlá na uloženie grafu a na návrat do menu hier. Ďalším tlačidlom sa hráč prepne na ďalší level hry.
- Vrcholy v tvare kruhu majú farbu #**CB2800**, písmo je **biele**, veľkosti **30px**. Šírka každého vrchola je **60px** a výška **60px**.
- Hrana spájajúca vrcholy je **neorientovaná** s hrúbkou **3px** a farbou #**E88200**.

Vytváranie hry „Štyri písmená“

Najväčším problémom pri tvorbe hry bolo vykreslenie prázdnych políčok a zobrazenie dialógu so správnymi odpoveďami. Naopak, vytvorenie .json inicializačného súboru a nastavenie štýlu vrcholov a hrán trvalo len pár minút. Implementácia tejto hry bola pre zúčastnených programátorov výzvou a strávili nad ňou viac času, ako sa predpokladalo, ale obaja úlohu zvládli.

4.2 Záver testovania

Obaja študenti, ktorí sa zúčastnili testovania, vyjadrili spokojnosť s používaním frameworku a uviedli, že jednoduchým spôsobom mohli vytvoriť webovú hru so slovami s ľubovoľnou funkcionalitou. Ďalším pozitívom je podľa testerov tvorba sieťovej aplikácie, bez predchádzajúcich znalostí princípu komunikácie cez sockety.

Kapitola 5

Záver

Literatúra

[1] *Node.js dokumentácia. [online].*, [cit 01.01.2020].
Dostupné na internete: `https://nodejs.org/api/`.

[2] *Cytoscape-cxtmenu.js dokumentácia. [online].*, [cit 05.01.2020].
Dostupné na internete:
`https://github.com/cytoscape/cytoscape.js-cxtmenu`.

[3] *Draggabilly. [online].*, [cit 05.01.2020].
Dostupné na internete: `https://draggabilly.desandro.com/`.

[4] *Popper.js dokumentácia. [online].*, [cit 05.01.2020].
Dostupné na internete: `https://popper.js.org/docs`.

[5] *Webpack. [online].*, [cit 05.01.2020].
Dostupné na internete: `https://webpack.js.org/concepts/`.

[6] *Express.js. [online].*, [cit 07.01.2020].
Dostupné na internete: `https://expressjs.com/en/guide/routing.html`.

[7] *FileSaver.js. [online].*, [cit 07.01.2020].
Dostupné na internete:
`https://www.npmjs.com/package/file-saver/v/1.3.2`.

[8] *Socket.io dokumentácia. [online].*, [cit 07.01.2020].
Dostupné na internete: `https://socket.io/docs/`.

[9] *Hra piknik. [online].*, [cit 10.01.2020].
Dostupné na internete: `https://play.google.com/store/apps/details?id=com.apnax.wordsnack.csk&hl=sk`.

[10] *Infovekáčik - Myšlienková mapa. [online].*, [cit 10.01.2020].
Dostupné na internete: `http://infovekacik.edu.fmph.uniba.sk/2006-september/zahrajme.php`.

[11] *Infovekáčik - Vieš pokračovať?. [online].*, [cit 10.01.2020].
Dostupné na internete: `http://infovekacik.edu.fmph.uniba.sk/2012-september/zahrajme.php`.

[12] *Infovekáčik. [online].*, [cit 10.01.2020].
Dostupné na internete: `http://infovekacik.edu.fmph.uniba.sk/`.

[13] *Sweetalert2.js. [online].*, [cit 10.01.2020].
Dostupné na internete:
`https://sweetalert2.github.io/`.

[14] *Osemsmerovka. [online].*, [cit 15.01.2020].
Dostupné na internete: `https://www.osemsmerovky.sk/`.

[15] *Word Search Pro. [online].*, [cit 15.01.2020].
Dostupné na internete:
`https://apps.apple.com/us/app/word-search-pro/id1123930673`.

[16] *TypeScript dokumentácia. [online].*, [cit 15.12.2019].
Dostupné na internete: `https://www.typescriptlang.org/docs/home.html`.

[17] *Cytoscape.js dokumentácia. [online].*, [cit 18.12.2019].
Dostupné na internete: `https://js.cytoscape.org/`.

[18] *Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika. [online].*, [cit 20.01.2020].
Dostupné na internete: `http://dvui.ccv.upjs.sk/`.

[19] Ďurisová N. 2015. *Návrh a implementácia systému na tvorbu webových edukačných aplikácií.* Univerzita Komenského - Fakulta matematiky, fyziky a informatiky, Bratislava, 2015.

[20] Bohunická I. 2018. *Pythonovský framework pre vytváranie hier.* Univerzita Komenského - Fakulta matematiky, fyziky a informatiky, Bratislava, 2018.

[21] Cherny B. 2018. *Programming TypeScript - Making Your JavaScript Applications Scale.* O'Reilly Media, 2018. ISBN 978-1492037651.

[22] Nagy B. 2018. *Webová aplikácia s využitím komunikačného protokolu WebSocket.* Slovenská technická univerzita v Bratislave - Fakulta elektroniky a informatiky, 2018.

[23] Balog R. 2019. *Detský mikrosvet ako motivačný nástroj.* Univerzita Komenského - Fakulta matematiky, fyziky a informatiky, Bratislava, 2019.

[24] Neznámy autor. Benefit from the mind mapping concept. [online]., [cit 20.01.2020].
Dostupné na internete:
`https://mindmapsunleashed.com/the-mind-mapping-concept`.

[25] Flavio Copes. The definitive node.js handbook. [online]., [cit 08.01.2020].
Dostupné na internete: `https://www.freecodecamp.org/news/the-definitive-node-js-handbook-6912378afc6e/`.