Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Scenárom riadené výučbové prostredia vo virtuálnej realite

Bakalárska práca

2023 **Lukáš Pisarčík**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Scenárom riadené výučbové prostredia vo virtuálnej realite

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	Ing. Štefan Korečko, PhD.

Košice 2023 **Lukáš Pisarčík**

Abstrakt v SJ

Táto práca sa zaoberá rozšírením virtuálneho prostredia pomocou implementácie Petriho siete a ovládania s využitím raycastera pre VR prilby. Analyzovali sme existujúce riešenia a predchádzajúcu prácu Ing Adama Kašelu s cieľom identifikovať oblasti, v ktorých je možné vylepšiť súčasný systém. Navrhli sme a implementovali funkcionality, ktoré umožňujú výber Petriho siete pre virtuálnu scénu a znenie úloh vo forme samostatného textového súboru, oddeleného od scény. Výsledkom práce je univerzálny systém ovládania scény pomocou Petriho siete, ktorý je použiteľný na akúkoľvek scénu. Implementované funkcionality a efektívny mechanizmus zaznamenávania aktivity prispievajú k lepšej interakcii užívateľov s virtuálnym prostredím a umožňujú tvorbu vlastných úloh.

Kľúčové slová v SJ

Petriho siete, virtuálna realita, aframe, angular, záznam aktivity, rozšírená realita

Abstrakt v AJ

This work focuses on expanding the virtual environment through the implementation of Petri nets and control using a raycaster for VR headsets. We analyzed existing solutions and the previous work of Ing. Adam Kašela in order to identify areas where the current system can be improved. We proposed and implemented functionalities that allow the selection of Petri net for the virtual scene and the definition of tasks in a separate text file, detached from the scene. The outcome of this work is a universal scene control system using Petri nets, which can be applied to any scene. The implemented functionalities and efficient activity tracking mechanism contribute to better user interaction with the virtual environment and enable the creation of custom tasks.

Kľúčové slová v AJ

Petri nets, virtual reality, aframe, angular, activity recording, augmented reality

Bibliografická citácia

PISARČÍK, Lukáš. *Scenárom riadené výučbové prostredia vo virtuálnej realite*. Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2023. 47s. Vedúci práce: Ing. Štefan Korečko, PhD.

70475

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE BAKALÁRSKEJ PRÁCE

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

Scenárom riadené výučbové prostredia vo virtuálnej realite

Scenario-Driven Educational Environments in Virtual Reality

Študent: **Lukáš Pisarčík**
Školiteľ: **Ing. Štefan Korečko, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce:
Pracovisko konzultanta:

Pokyny na vypracovanie bakalárskej práce:

1. Oboznámiť sa s prototypom virtuálneho výučbového prostredia so scenárom definovaným Petriho sieťou, vyvinutým na školiacom pracovisku.
2. Analyzovať existujúce prístupy k využitiu Petriho sietí pri definovaní scenárov pre virtuálne prostredia.
3. Navrhnúť a implementovať časť systému pre virtuálne výučbové prostredia, zaoberajúcu sa samotnými virtuálnymi prostrediami a ich interakciou s Petriho sieťou.
4. Pri návrhu a implementácii postupovať na základe výsledkov analýzy a v súlade s konceptuálnym riešením, navrhnutým vedúcim práce.
5. Návrh a implementáciu koordinovať s ďalšími riešiteľmi systému.
6. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 26.05.2023
Dátum zadania bakalárskej práce: 31.10.2022

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -02-

.....
prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 26.5.2023 .....

Vlastnoručný podpis

Poďakovanie

Na tomto mieste by som rád poďakoval svojmu vedúcemu práce za jeho čas a odborné vedenie počas riešenia mojej záverečnej práce.

Rovnako by som sa rád poďakoval svojim rodičom a priateľom za ich podporu a povzbudzovanie počas celého môjho štúdia.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam zdrojových kódov

Úvod

Cieľom tejto práce je nadviazať na diplomovú prácu Ing. Adama Kašelu s názvom Výučbové scenáre v rozšírenej realite využívajúce procesné grafy. Výsledkom vyššie uvedenej práce je výučbové prostredie vo virtuálnej realite, ktoré je zamerané na výučbu dejepisu pre študentov základných a stredných škôl. Študenti si v takomto prostredí môžu otestovať svoje znalosti pomocou kvízu.

Vytvorená aplikácia však neobsahuje potrebné funkcionality, aby bola použiteľná v reálnej výučbe na stredných školách. Vytvorenie práve takýchto funkcionalít je hlavným cieľom bakalárskej práce. Možnosť úpravy scénara alebo tvorby nového je zo strany učiteľa veľmi podstatná, kedže si takto vie prispôsobiť výučbový proces.

Využitím webových aplikácií sa WebVR stal dostupnejším pre bežných používateľov. Súčasné webové technológie sú schopné spracovať senzorické dáta získané z užívateľských zariadení obsahujúcich kamery, mikrofóny, dotykové senzory, a mnohé iné. [1] Pomocou rozšírení webového ramca A-Frame a s využitím kamier a mikrofónov vieme výučbové prostredie spraviť ešte interaktívnejšie, zaujímavejšie a pripojiť viacero užívatelov súčasne.

Ďalším dôležitým aspektom tejto práce je zaznamenávanie aktivity užívateľa počas vykonávania úlohy. Záznam slúži na zber informácií o aktivitách užívateľa, ktoré sú následne odosielané na server pre ďalšie spracovanie. Jeho hlavným účelom je generovanie štatistík a obnova scény zo záznamu, čo umožňuje užívateľovi pokračovať v úlohe aj po obnovení stránky.

Spomínané vylepšenia tak pomôžu hlavne učitelom vytvárať, upravovať a kontrolovať výučbové scénare a tým aj zlepšiť a zefektívniť používanie aplikácie.

Formulácia úlohy

Cieľom tejto práce je dôkladne zaktualizovať a vylepšiť existujúcu aplikáciu pre virtuálnu realitu, ktorá slúži na výučbu dejepisu na stredných školách. Na dosiahnutie tohto cieľa budeme využívať Petriho siete a rámec A-Frame, ktorý bol

úspešne využitý pri vytváraní pôvodnej verzie aplikácie.

Prvým krokom v našej práci je dôkladná analýza podobných projektov, ktoré zahŕňajú využitie Petriho sietí na výučbu v školskom prostredí, ako aj dôkladné preskúmanie existujúceho riešenia. Táto analýza nám umožní získať užitočné poznatky, skúsenosti a najlepšie postupy, ktoré môžeme následne aplikovať pri aktualizácii aplikácie. Výsledky tejto analýzy a porovnania s existujúcimi riešeniami sú podrobne zdokumentované v Kapitole 1.

Po dokončení analýzy nasleduje samotný návrh rozšírenia existujúceho riešenia. V tejto fáze budeme navrhovať a detailne popisovať nové funkcionality, ktoré chceme implementovať v aplikácii. Taktiež budeme venovať pozornosť úprave architektúry riešenia, aby sme zabezpečili efektívne integrovanie nových prvkov. Okrem toho sa budeme venovať aj implementácii nových funkcií, vrátane navrhovania scenárov na riadenie virtuálneho prostredia. Všetky tieto aspekty návrhu, úpravy a implementácie sú detailne rozobraté v kapitolách 2, 3, 4.

Na záver našej práce budeme zhodnocovať výsledky dosiahnuté v rámci tejto aktualizácie aplikácie. V kapitole 5 analyzujeme a diskutujeme o dosiahnutých výsledkoch, zahŕňajúcich nové funkcionality, vylepšenú použiteľnosť aplikácie. Okrem toho vytvoríme prílohy, ktoré obsahujú podrobnú dokumentáciu.

1 Analytická časť

Táto kapitola slúži na vysvetlenie hlavných technológií, prostriedkov a modelov použitých v práci. Ďalej táto kapitola analyzuje existujúce riešenia prístupov k použitiu Petriho sietí pre scenáre virtuálnych prostredí. Vykonávaná bola aj analyza existujúcich riešení v oblasti výučby študentov stredných a základných škôl s využitím technologií rozšírenej reality. A v neposlednom rade budeme aj analyzovať samotné riešenie diplomovej práce Ing. Adama Kašelu, na ktorú v tejto bakalárskej práci nadväzujeme. Tento krok je veľmi dôležitý pri stanovení nedostatkov a ich následnom riešení, ktoré je predmetom bakalárskej práce.

1.1 Základné pojmy a technologické riešenia

Prvá časť tejto kapitoly sa zameriava na definovanie kľúčových pojmov a konceptov, ktoré sú nevyhnutné na pochopenie technologických riešení. Či už ste nováčik alebo skúsený profesionál, objasnenie týchto základných pojmov vám pomôže pochopiť nasledujúce kapitoly.

1.1.1 Rozšírená realita (XR)

Pojem virtuálna realita (VR) je pravdepodobne známy. VR označuje zariadenia, ktoré obmedzujú pohľad používateľov na fyzický svet, takže používateľ vidí iba digitálne vykreslené obrázky. Zariadenia VR dokážu napodobňovať 3D stereoskopické videnie tým, že každému oku prezentujú samostatné obrázky. Okrem stereoskopického 3D vykonávajú pokročilejšie zariadenia sledovanie náhlavnej súpravy VR v reálnom čase, takže keď používateľ pohybuje hlavou vo fyzickom svete, pohyb sa zhoduje aj v digitálnom svete.

Ďalším pojmom, ktorého význam rýchlo narastá, je rozšírená realita (AR). AR je pohľad na fyzický svet, ktorý je rozšírený o digitálne informácie. Toto zobrazenie môže byť priame (napríklad displej namontovaný na hlave) alebo nepriame (zobrazené na obrazovke telefónu alebo tabletu).

Pojem „zmiešaná realita“ (MR) bol vytvorený na opis spektra zariadení VR a AR, ktoré spájajú fyzický svet s digitálnym svetom. Termín MR sa niekedy používa odlišne a môže zahŕňať alebo vylúčiť určité aplikácie VR alebo AR v závislosti od presnej definície. Z tohto dôvodu si pojem „rozšírená realita“ (XR) nedávno získal priazeň ako zastrešujúci pojem, ktorý zahŕňa všetky AR, VR a MR[2].

Oblasti virtuálnej reality a rozšírenej reality rýchlo rastú a využívajú sa v širokej škále spôsobov, od zábavy, marketingu, nehnuteľností, školení a práce na diaľku[3].

Motiváciou aplikácie XR vo vzdelávaní je to, že študenti alebo pacienti môžu lepšie pochopiť anatómiu prostredníctvom 3D XR pohľadov než prostredníctvom tradičných materiálov[2].

1.1.2 A-frame

A-Frame je webový rámec na vytváranie prostredí a aplikácií virtuálnej reality pomocou HTML a JavaScriptu. Bol vyvinutý skupinou Mozilla VR na uľahčenie tvorby obsahu WebVR a umožňuje vývojárom vytvárať VR scény pomocou HTML a ďalších značiek A-Frame. A-Frame tiež poskytuje prístup k základnému JavaScriptu a existujúcim webovým rozhraniam API a používa systém entity-komponent na podporu zloženia a rozšíriteľnosti. Je to bezplatný produkt s otvoreným zdrojovým kódom s komunitou vývojárov, ktorí preň vytvárajú nástroje a komponenty.

A-Frame bol navrhnutý tak, aby bol vývoj VR prístupnejší pre vývojárov webu ako tradičné systémy herných motorov. Je ľahké začať s A-Frame a využiť existujúce skúsenosti s vývojom webu na vytváranie prostredí VR. A-Frame je ľahký systém, ktorý sa dobre hodí na použitie na smartfónoch a s lacnými náhlavnými súpravami VR, ako sú Google Cardboard alebo súpravy náhlavných súprav VR DIY od Vullo. To umožňuje vývojárom vytvárať obsah VR, ku ktorému je možné pristupovať na širokej škále zariadení bez potreby špeciálneho hardvéru VR[4].

1.1.3 Angular

Angular je komplexná platforma navrhnutá tak, aby pomohla vývojárom vytvárať dynamické, responzívne a na funkcie bohaté webové aplikácie. Angular, vyvinutý spoločnosťou Google, je rámec, ktorý používa TypeScript ako svoj základ. TypeScript je nadmnožina JavaScriptu, ktorá do jazyka zavádza typový systém a umožňuje vývojárom zachytiť potenciálne chyby pred spustením kódu. Výsledkom sú robustnejšie a spoľahlivejšie aplikácie [5].

Jadrom Angularu je architektúra založená na komponentoch, ktorá umožňuje vývojárom vytvárať modulárny a opakovane použiteľný kód. Komponenty sú stavebnými kameňmi aplikácií Angular a sú zodpovedné za vykresľovanie používateľského rozhrania. Možno ich považovať za vlastné prvky HTML, ktoré zapuzdrujú používateľské rozhranie aj logiku konkrétnej funkcie alebo časti aplikácie. Rozdelením aplikácií na menšie komponenty môžu vývojári písať kód, ktorý sa ľahšie testuje, udržiava a rozširuje [5].

Ďalšou kľúčovou vlastnosťou Angularu je jeho súbor tesne integrovaných knižníc, ktoré pokrývajú širokú škálu funkcií. Tieto knižnice zahŕňajú nástroje na smerovanie, správu formulárov, komunikáciu klient-server, animácie a ďalšie. Poskytnutím týchto vopred vytvorených nástrojov pomáha Angular vývojárom šetriť čas a vyhnúť sa bežným nástrahám, ktoré prichádzajú s vytváraním zložitých webových aplikácií [5].

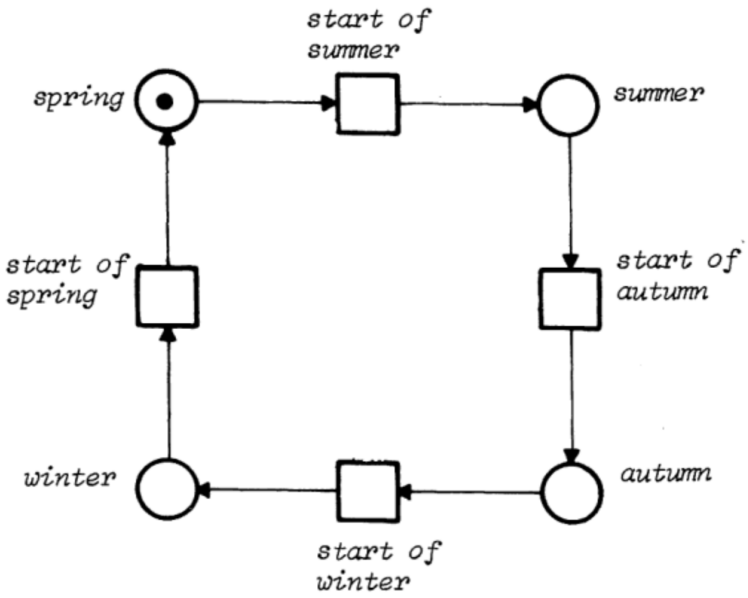
Jednou z najvýznamnejších výhod Angularu je jeho schopnosť škálovať od malých projektov jedného vývojára až po veľké aplikácie na podnikovej úrovni. Ako aplikácie rastú vo veľkosti a zložitosti, architektúra a sada nástrojov Angular umožňujú vývojárom udržiavať kvalitu kódu, spravovať závislosti a udržiavať výkon aplikácie [5].

1.1.4 Petriho siete

Petriho sieť je model, ktorý slúži na popis a analýzu toku dát. Využívajú sa najmä v systémoch, ktoré obsahujú súbežne bežiace aktivity. Pomocou Petriho sietí môžeme modelovať aktivity súbežne[6]. Správanie systémov môžeme riadiť pomocou stavových, algebraických rovníc alebo aj iných matematických modelov, ktoré pomocou Petriho sietí ako matematického nástroja vieme zostrojiť. Podobne ako vývojové diagramy, Petriho siete môžu byť použité ako vizuálna pomôcka pre opis informácií vo forme grafu. Petriho siete sú definované dvoma typmi uzlov. Ako kružnice sú zobrazované miesta a ako obdlžníku sú definované prechody. Prechody a miesta sú v grafe spojené šípkou, ktorá je ohodnotená kladným číslom, ktoré reprezentuje váhu[7]. Miesto môže mať niekoľko oblúkov do a z prechodu, taktiež aj prechod môže mať niekoľko oblúkov do a z miesta [8]. Vykonávanie Petriho sietí je kontrolované pozíciou značky na mieste v grafe. V grafickom znázornení je táto značka, nazývaná aj token označovaná bodkou na danom mieste v grafe. Petriho sieť, ktorá obsahuje takýto token je nazývaná označená Petriho sieť [6]. Prechod môže byť odpálení pokiaľ všetky miesta, ktoré sú s ním spojené oblúkom majú aspoň jeden token. Po odpálení prechodu sa tokeny z miest, ktoré prichádzali do prechodu odstránia a pridajú sa na miesta, ktoré z

prechodu vystupujú [8]. Prichádzajúce a odchádzajúce miesta pri prechode znázorňuje šípka.

Ako príklad grafického znázornenia Petriho siete použijeme príklad z knihy [9], znázornený na obrázku 1.1.



Obr. 1.1: Príklad Petriho siete v grafickom znázornení[9]

Na obrázku 1.1 je možné vidieť miesta, ktoré predstavujú ročné obdobia znázornené v kružniciach a prechody, ktoré predstavujú začiatok ročného obdobia v štvorcoch [9]. Z obrázku taktiež možno vidieť že prechod 'start of summer' môže byť odpálení, pretože všetky miesta, ktoré doňho prichádzajú majú aspoň jeden token. Po odpálení prechodu sa token na mieste 'spring' odstráni a zároveň sa pridá na miesto 'summer'.

1.2 Analýza existujúcich prístupov použitia Petriho sietí pre scénare virtuálnych prostredí

V tejto podkapitole sa venujeme vyhľadávaniu a naslednej analýze voľne dostupných prác, ktoré používajú Petriho siete ako prostriedok pre tvorbu scénarov. Petriho siete sú grafický a matematický nástroj, poskytujú jednotné prostredie pre modelovanie, formálnu analýzu a návrh systémov diskrétnych udalostí[10]. V našej práci pôsobia Petriho siete ako vhodný nástroj na vytváranie a kontrolu scénarov priebehu aktivity vo virtuálnej realite. Tento náš predpoklad chceme v tejto analýze s pomocou dôkazov aj potvrdiť.

V práci sme použili Kitchenhamovu metódu na vyhľadávanie relevantných prác a postupovali sme podľa postupov obsiahnutých v práci [11]. Táto metóda zahŕňa tri fázy:

- **plánovanie**
- **vykonanie**
- **zhodnotenie výsledkov**

V tomto kroku si overíme či sú Petriho siete vhodný nástroj pre vytváranie scénarov pre herné prostedie a to pomocou vyhľadania relevantných prác, ktoré nám môžu našu úvahu potvrdiť. Vyhľadanie publikácií sme vykonávali v troch dostupných databázových knižniciach ktorými boli **Google schollar, Scoptus a Web of science**. V nasledujúcich podkapitolách je podrobne opísaný postup pri vykonávaní analýzy a aj následné vyhodnotenie vedeckých otázok a zhodnotenie výsledkov hľadania.

1.2.1 Identifikácia nevyhnutnosti preskúmania

O použití Petriho sieti na vytváranie scénarov pre prostredia vo virtuálnej realite zatiaľ toho veľa nevieme. Preto je nevyhnutnosťou nájsť práce, ktoré uvedenú problematiku riešia, ak takéto práce existujú. Okrem toho naša analýza identifikuje povahu každého nájdeného riešenia, ako aj oblasti, v ktorých sa relevantné práce používajú.

1.2.2 Formulovanie výskumných otázok

Tento krok za zaoberá vytvorením si vhodných otázok, ktoré nám pomôžu vytvoriť systematický prehľad literatúry, a tým aj nájsť relevantné práce. Cieľom analýzy je nájsť publikácie obsahujúce aplikáciu Petriho sieti na vytvorenie scenára, pri formulovaní otázok je dostačujúce položiť jednu otázku, keďže hľadaný cieľ je veľmi špecifický. Otázka je znázornená v tabuľke 1.1.

Tabuľka 1.1: Tabuľka výskumných otázok

Otázka	**Motivácia**
O1. Boli Petriho siete použité pri vytváraní scénarov	Motiváciu otázky bolo nájsť príklady, kde sú Petriho siete na vytvorenie scenárov

1.2.3 Metodika získavania dát

Po identifikácii problému a formulovaní správnych vedeckých otázok je potrebné definovať metodiku získavania dát. Táto metodika obsahuje informácie o stratégiách vyhľadavania, kritéria výberu štúdií a meraní ich kvality, syntéze údajov a stratégií šírenia [11]. Vyhľadávanie sme vykonávali pomocou dopytov na databázy Google schollar, Scoptus a Web of science. Tabuľka 1.2 obsahuje vyhľadavacie dopyty pre našu problematiku.

Tabuľka 1.2: Tabuľka použitých dopytov na databázy

Databáza	**Dopyt**
Google schoolar	TITLE-ABS-KEY ('petri net' AND (game OR 'virtual environment' OR 'virtual scene') AND (story OR 'story management' OR scenario))
Scoptus	TITLE-ABS-KEY ('petri net' AND (game OR 'virtual environment' OR 'virtual scene') AND (story OR 'story management' OR scenario))
Web of science	TS=('petri net' AND (game OR 'virtual environment' OR 'virtual scene') AND (story OR 'story management' OR scenario))

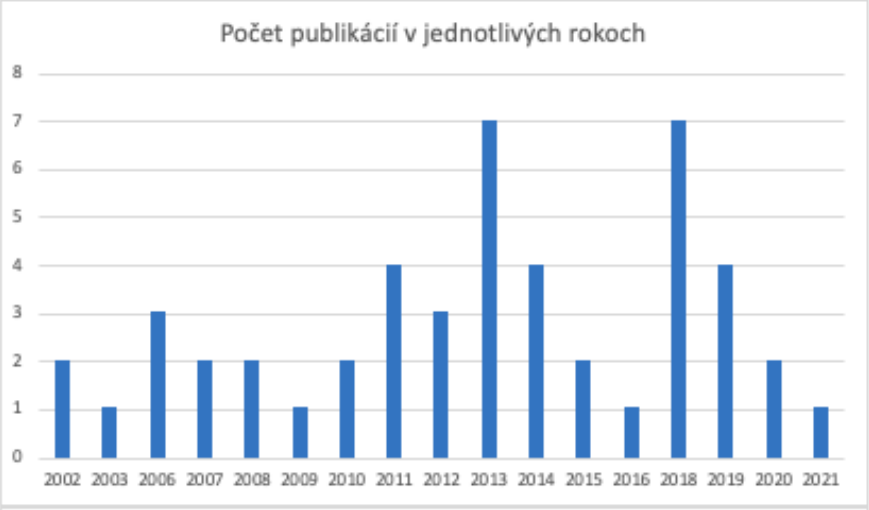
Po vykonaní dopytov nad databázamy bolo taktiež potrebné odstrániť publikácie, ktoré sa nachádzali vo viacerých databázach, súčasne a tým pádom sa vytvorili duplikáty.

1.2.4 Zhodnotenie výsledkov

V tejto podkapitole sa zaoberáme zhodnotením a analýzou výsledkov získaných pomocou vyššie uvedenej metodiky. Celkovo sa nám podarilo nazbierať 50 jedinečných publikácií v rámci troch databáz. Tieto publikácie si najprv kategoricky rozdelíme a neskôr si z nich vyberieme pár, ktoré sú pre našu prácu najrelevantnejšie a tie podrobnejšie opíšeme.

V nasledujúcom grafe 1.2 je možné vidieť počet publikácií za jednolivé roky. Z grafu je jasné vidieť že najviac publikácií bolo zverejnených v rokoch 2013 a 2018, v ostatných rokoch sa počet publikácií pohyboval v podobných hodnotách.

Vytvorili sme ešte jednu štatistiku, ktorá je zobrazená v grafe 1.3. V grafe je možné vidieť počet publikácií pre krajinu. Až 7 publikácií z celkového množstva



Obr. 1.2: Publikácie v jednotlivých rokoch

50 nemalo uvedenú krajinu. Najviac publikácií bolo vydaných vo Fracúzsku s celkovým počtom 9, na druhom mieste bola Brazília so 7 publikáciami. Nenašli sme však publikáciu vydanú na Slovensku, ale 4 práce boli publikované v Českej republike.

Obr. 1.3: Publikácie v jednotlivých krajinách

1.2.5 Najrelevantnejšie práce

V tejto podkapitole sa zaoberáme prácami, ktoré sme z nášho výskumu zhodnotili ako najrelevantnejšie pre našu prácu.

1.2.6 Petriho siete pre hernú zápletku

V práci [12], sa Petriho siete používajú na modelovanie nelineárneho deja príbehu, ako aj reaktívnych plánov, ktoré riadia správanie jednotlivých aktérov vo virtuálnom svete. To umožňuje iteratívny dizajn príbehu, ako aj schopnosť zvládnuť veľké virtuálne svety s mnohými hercami. Táto technika umožňuje vytvárať dynamické, interaktívne prostredia virtuálnej reality, ktoré sa dokážu prispôsobiť činnostiam používateľa.

Taktiež bola vyvinutá prototypová aplikácia na demonštráciu účinnosti techniky. Aplikácia umožňuje vytvárať virtuálne svety a príbehy pomocou Petriho sietí a umožňuje používateľom zažiť príbehy v prostredí virtuálnej reality. Tento prototyp môže slúžiť ako základ pre ďalší vývoj aplikácií interaktívnej virtuálnej reality pomocou Petriho sietí.

1.2.7 Model Petriho siete pre vážne hry na základe klasifikácie motivačného správania

Práca [13] uvažuje o prístupe k modelovaniu serióznych herných systémov pomocou Petriho sietí a pomocou LVQ (Learning Vector Quantization) na optimalizáciu klasifikácie motivačného správania hráčov. Navrhovaná hra motivačného správania (MBG) má za cieľ monitorovať, ako hráči interagujú s hrou a poskytovať vhodné úlohy na základe schopností hráča, s cieľom zachovať emocionálnu stabilitu hráčov a predchádzať nude a frustrácii. Autori tvrdia, že to môže pomôcť udržať záujem hráčov o hru a podporiť ich učenie.

1.2.8 Časovaný model Petriho siete na špecifikovanie scenárov videohier

Práca [14] predstavuje metódu navrhovania videohier pomocou Petriho sietí. Kľúčovou myšlienkou je použiť sieť WorkFlow na znázornenie činností, ktoré bude hráč v hre vykonávať a stavový graf na znázornenie oblastí virtuálneho sveta, s ktorými sa hráč stretne. Sieť WorkFlow a stavový graf sú potom spojené do jedného modelu, ktorý je možné simulovať pomocou softvéru CPN Tools. Výsledný model umožňuje dizajnérovi odhadnúť, koľko času bude hráč potrebovať na dokončenie konkrétnej úrovne hry.

Navrhovaný prístup je demonštrovaný na príklade videohry Silent Hill II. Sieť WorkFlow pre túto hru je vytvorená identifikáciou činností, ktoré bude hráč vykonávať, ako je skúmanie prostredia, riešenie hádaniek a boj s nepriateľmi. Štátny

graf je vytvorený modelovaním rôznych oblastí herného sveta, ako sú budovy, ulice a lesy.

Po spojení siete WorkFlow a stavu grafu do jedného modelu je možné použiť softvér CPN Tools na simuláciu modelu a určenie odhadovaného času dokončenia pre danú úroveň hry. To umožňuje dizajnérovi analyzovať vplyv rôznych faktorov na trvanie hry, ako je počet a náročnosť hádaniek alebo počet a sila nepriateľov.

1.2.9 Návrh príbehu pre hru Europe 2045

V kontexte hry Európa 2045 možno Petriho siete použiť na špecifikáciu rôznych scenárov rozvetvenia a kríz, s ktorými sa hráči môžu stretnúť. To umožňuje vysokú úroveň flexibility a prispôsobivosti v hre, pretože správca príbehu môže použiť Petriho siete na vytváranie zložitých, rozvetvených príbehov, ktoré sa môžu vyvíjať v reálnom čase na základe akcií hráčov.

Celkovo možno povedať, že používanie Petriho sietí v Európe 2045 ponúka sľubný prístup k vytváraniu dynamických, rozvetvených príbehov vo vzdelávacej hre. Využitím silných stránok Petriho sietí na modelovanie zložitých interakcií a scenárov môže hra poskytnúť hráčom pútavú a vzdelávaciu skúsenosť, ktorá ich naučí o ekonomike, politike a mediálnych štúdiách v kontexte súčasných európskych problémov[15] .

1.2.10 Návrh školiaceho scénara pre obsluhu strojov vstrekovania plastu

Práca [16] opisuje tréningový systém vstrekovania plastov založený na virtuálnej realite (VPIMTS). Tento systém je školiaci systém určený na zlepšenie pochopenia a účinnosti postupov vstrekovania plastov. Systém využíva modul plánovania úloh, modul inteligentných pokynov, modul simulácie a modul virtuálneho prostredia (VE) na vytvorenie integrovaného systému. Využitie modelovania Petriho siete umožňuje voľne plynúce prostredie, kde sa účastníci môžu učiť a skúmať, a tiež umožňuje systému poskytnúť účastníkovi pokyny k ďalšiemu kroku. Použitie metódy konečných prvkov (FEM) a techník modelovania postupu vstrekovania ďalej zvyšuje možnosti systému. Celkovo je VPIMTS výkonným nástrojom na zlepšenie pochopenia a efektívnosti školenia v postupoch vstrekovania.

1.2.11 Návrh tréningového scénara pre obsluhu CNC strojov

Architektúra tréningových systémov založených na VR prezentovaná v práci[17] zahŕňa použitie technológie virtuálnej reality na vytvorenie pohlcujúcich tréningových prostredí pre rôzne úlohy.

Článok tiež predstavuje praktický prístup k modelovaniu znalostí, ktorý využíva formalizmus Petriho sietí na modelovanie tréningových scenárov systémov založených na VR. Petriho siete sú matematický modelovací jazyk, ktorý možno použiť na znázornenie správania zložitých systémov, ako sú napríklad tie, ktoré sa podieľajú na CNC frézovacích operáciách.

1.2.12 Zhrnutie analýzy existujúcich prístupov použitia Petriho sieti pre scénare virtuálnych prostredí

V kontexte herného dizajnu by sa Petriho siete mohli potenciálne použiť na navrhovanie a analýzu scenárov pre hry alebo interaktívne zážitky, ako sú scenáre školení zamestnancov implementované pomocou technológií XR (rozšírená realita). XR technológie, ako je virtuálna realita (VR), rozšírená realita (AR) a zmiešaná realita (MR), sa dajú použiť na vytvorenie pohlcujúcich interaktívnych prostredí na školenia a vzdelávanie. Petriho siete by sa mohli použiť na modelovanie toku udalostí a interakcií v týchto prostrediach, čo umožňuje návrhárom analyzovať a optimalizovať scenáre z hľadiska účinnosti a efektívnosti.

S touto úvahou sme vstupovali do skúmania použitia Petriho sieti ako scénarov a naším cieľom analýzy bolo túto úvahu potvrdiť alebo vyvrátiť. Vykonali sme vyhľadávanie publikácií vo viacerých databázach, ktorým sme docielili nájdenie 50 publikácií, pomocou ktorých vieme odpovedať na vedeckú otázku definovanú pred začatím vyhľadávania.

O1. Boli Petriho siete použité pri vytváraní scénarov ?

Áno, vo viacerých publikáciach sme mohli nájsť použitie Petriho sieti na tvorbu scénarov, väčšinou sa jednalo o hry s dejom.

Vďaka týmto poznatkom zo skúmania danej problematiky môžeme potvrdiť že pouťitie Petriho sieti pre vytváranie scénarov je v praxi používané a aj je vhodné pre použitie v našej práci.

1.3 Analýza existujúceho riešenia

Riešenie, ktoré bolo prezentované v diplomovej práci Adama Kašela [18] sa zameriava na využitie Petriho sietí na vytvorenie virtuálneho prostredia s konkrétnym a nemenným scenárom. Cieľom bolo poskytnúť žiakom 7. ročníka základných škôl a 2. ročníka gymnázia s osemročným štúdiom interaktívnu formu výučby dejepisu.

Študenti mali za úlohu prezrieť si múzeum a splniť úlohy, ktoré mali overiť ich nadobudnuté poznatky ohľadom zadanej problematiky. Scéna bola rozdelená do 4 miestností, kde v 3 z nich boli pripravené úlohy zamerané na rôzne časti učiva. Riadenie logiky scény bolo implementované na základe Petriho siete, ktorá zabezpečovala vykonanie určitých činností v scéne v určenom poradí podľa návrhu siete.

Riešenie bolo overené prieskumom použiteľnosti pre používateľa, ktorý bol vykonaný s cieľom zistiť, či riešenie splnilo svoj účel a či bolo používateľné. Výsledky prieskumu ukázali, že riešenie bolo úspešne otestované a žiaci mali pozitívnu skúsenosť s využívaním virtuálneho prostredia.

Avšak, aj napriek tomu, že riešenie bolo úspešne otestované, má niekoľko obmedzení. Jedným z nich je chýbajúca flexibilita v možnosti upravovať scenár, čo by umožnilo prispôsobiť virtuálne prostredie rôznym skupinám žiakov s rôznymi úrovňami poznania. Architektúra riešenia tiež nie je dostatočne navrhnutá na podporu tejto možnosti. Ďalším obmedzením je chýbajúci systém registrácie a prihlasovania používateľov, čo by umožnilo sledovať progres jednotlivých žiakov v rámci vyučovania. Toto by bolo veľmi dôležité pre učiteľov, aby vedeli presne vidieť, ktorí žiaci sa s danou témou stretli už predtým a ktorí potrebujú viac pomoci.

Ďalším obmedzením je chýbajúca možnosť vytvárať vlastné scenáre pomocou úpravy textového súboru, kde si užívateľ môže zvoliť, ktoré otázky budú súčasťou testu a aké bude ich znenie. Táto možnosť by umožnila učiteľom prispôsobiť virtuálne prostredie konkrétnej skupine žiakov, čím by sa zlepšila efektívnosť vyučovania.

Ďalším krokom by bol refaktor kódu tak, aby užívateľ mal po obnovení stránky odpálené prechody, ktoré boli pred obnovením. Zaznamenávať sa bude každý odpálení prechod spolu s časom, kedy bol vykonaný. Tým by sme aktuálna prácu užívateľa uložili a tak mu umožnili pokračovať vo vykonávanej práci aj po obmovení stránky.

Rovnako by bolo potrebné pridať možnosť ovládania pomocou Oculusu, aby

bolo možné využiť virtuálne prostredie naplno. Táto možnosť by pridala ďalší rozmer do vyučovania dejepisu, pretože by žiaci mohli skutočne prežiť historické udalosti.

Posledným krokom by bolo rozšírenie scény o viacero exponátov, aby bolo možné poskytnúť žiakom bohatší interaktívny zážitok a rozšíriť tak možnosti vyučovania dejepisu. Tým by sa zvýšila atraktívnosť virtuálneho prostredia a žiaci by mali väčšiu motiváciu sa zaujímať o dejepis.

Všetky tieto kroky by mali výrazne zlepšiť užívateľskú skúsenosť a prispieť k úspešnejšiemu vyučovaniu dejepisu pre žiakov základných a stredných škôl. Učitelia by mali k dispozícii viac možností prispôsobiť virtuálne prostredie konkrétnej skupine žiakov, čím by sa zlepšila efektívnosť vyučovania. Žiaci by mali k dispozícii viac interaktívnych exponátov, čo by ich motivovalo k väčšiemu záujmu o dejepis. A celkový zážitok by bol ešte bohatší vďaka možnosti ovládania pomocou Oculusu.

1.4 Ovládanie pomocou VR headsetu

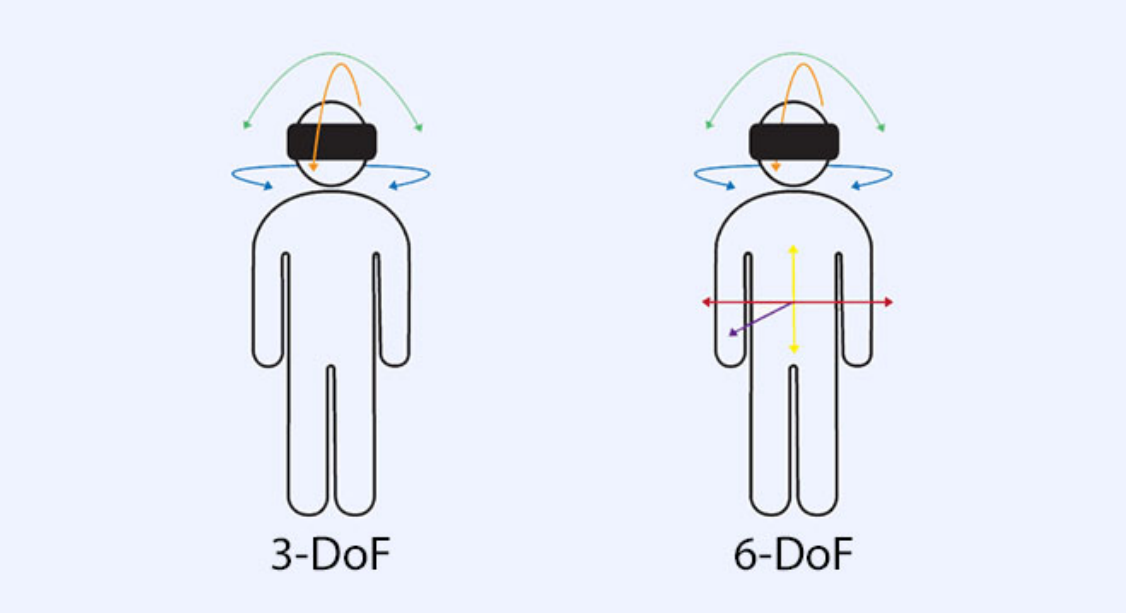
A-Frame je výkonný nástroj na vytváranie pohlcujúcich zážitkov z virtuálnej reality, ktorý poskytuje podporu pre širokú škálu platforiem, zariadení a metód vstupu. To znamená, že vývojári majú flexibilitu pri vytváraní skúseností, ktoré sú prispôsobené konkrétnym zariadeniam alebo platformám, pričom využívajú jedinečné funkcie a možnosti každého z nich [19].

1.4.1 Možnosti pohybu

Jednou z kľúčových výziev, pokiaľ ide o interakciu VR, je množstvo dostupných metód vstupu. Na rozdiel od 2D webu, kde sú hlavnými formami interakcie vstup myšou a dotykom. Pomocou VR môžu používatelia chytiť, hádzať, trieť, preklápať, štrngať, naťahovať, stláčať a vykonávať nespočetné množstvo ďalších akcií, ktoré sú na 2D webe nemožné.

Pokiaľ ide o ovládače VR, existujú dva hlavné typy. Ovládače s 3 stupňami voľnosti (3DoF) a ovládače so 6 stupňami voľnosti (6DoF). Ovládače 3DoF sú obmedzené na rotačné sledovanie, čo znamená, že môžu sledovať iba orientáciu ovládača v priestore. To znamená, že používatelia nemôžu pohybovať rukami dopredu a dozadu alebo hore a dole, čo môže obmedziť rozsah možných interakcií. Ovládače s 6DoF majú rotačné aj polohové sledovanie, čo umožňuje voľný pohyb v 3D priestore. To znamená, že používatelia sa môžu natiahnuť dopredu, za chrbát, pohybovať rukami po tele alebo blízko tváre. Mať 6DoF je ako realita, kde

máme ruky aj paže [19].



Obr. 1.4: Ukážka rozdielu medzi 3DoF a 6DoF [20]

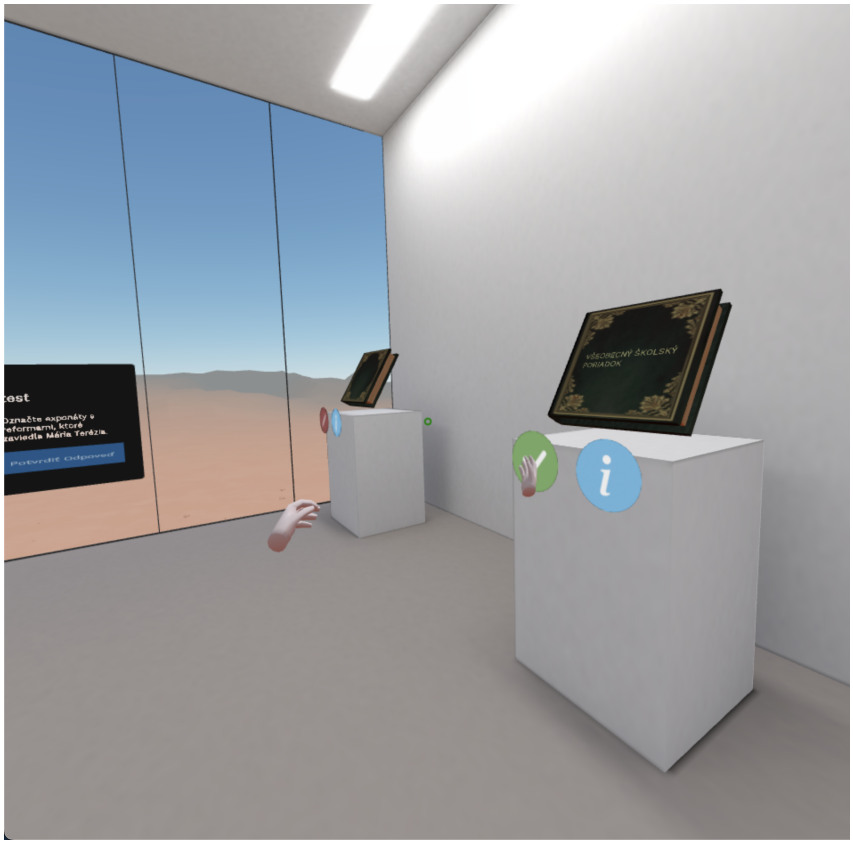
1.4.2 Komponent Super hands

Komponent super-hands od Williama Murphyho [1] poskytuje prirodzenú interakciu ručného ovládača. Komponent interpretuje vstup zo sledovaných ovládačov a komponentov detekcie kolízie do interakčných gest a tieto gestá komunikujú s cieľovými entitami, aby mohli reagovať [19].

Aktuálne implementované gestá sú Hover, Grab, Stretch a Drag-drop. Vznášanie zahŕňa držanie ovládača v kolíznom priestore entity, zatiaľ čo uchopenie zahŕňa stlačenie tlačidla počas vznášania sa entity, prípadne aj jej pohyb. Natiahnutie zahŕňa uchopenie entity dvoma rukami a zmenu jej veľkosti, zatiaľ čo pretiahnutie zahŕňa pretiahnutie entity na inú entitu [19].

Správnou konfiguráciou majú super-hands ponúkajú interaktívne ovládacie prvky pre rôzne typy zariadení vrátane stolných počítačov, mobilných zariadení, zariadení s 3DOF (napríklad Daydream a GearVR) a 6DOF ovládače ako Vive, Oculus Touch, Valve Index a ďalšie [21].

[1] Komponent super hands od Williama Murphyho dostupné na: https://github.com/c-frame/aframe-super-hands-component

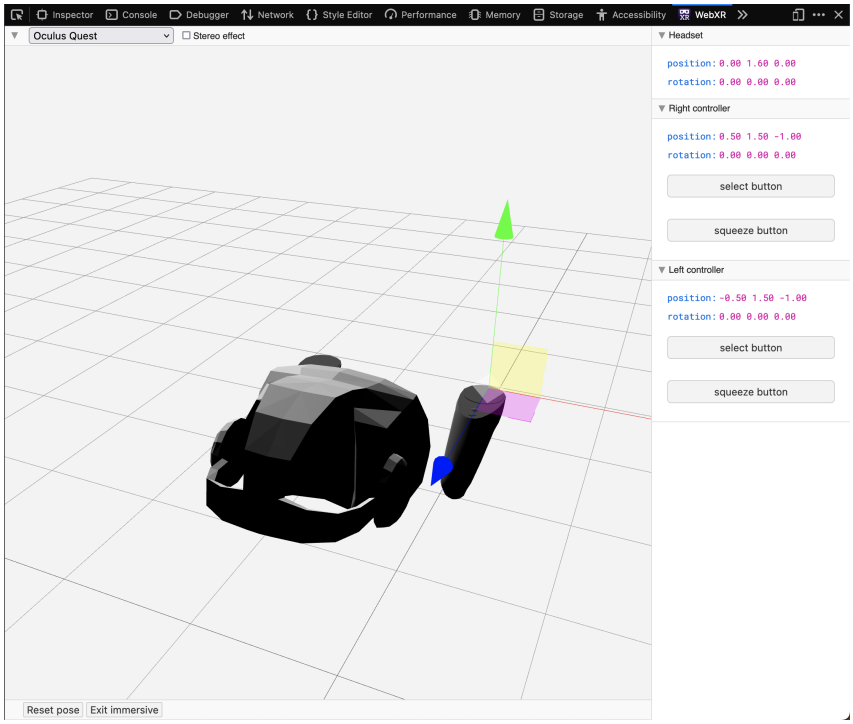


Obr. 1.5: Ukážka použitia VR ovládača v scéne

1.4.3 Testovanie ovládania

Na testovanie ovládania je možné použiť rozšírenie WebXR [2]. Rozšírenie poskytuje platformu pre vývojárov na testovanie ich WebXR aplikácií bez toho, aby museli používať fyzický headset VR. Namiesto toho rozšírenie emuluje VR zariadenia a ovládače, čo umožňuje testovať VR aplikácie. Rozšírenie emulátora WebXR poskytuje používateľské rozhranie, ktoré umožňuje simulovať rôzne typy ovládačov VR, ako sú ovládače napríklad Oculus Touch a HTC Vive. Toto rozhranie obsahuje aj tlačidlá na spustenie bežných akcií VR, ako je uchopenie a uvoľnenie predmetov, teleportovanie a otáčanie. Použitie Emulátor WebXR je možné vidieť na obrázku 1.6.

[2]Rozšírenie Emulátora WebXR, dostupné na: https://addons.mozilla.org/en-US/firefox/addon/webxr-api-emulator/



Obr. 1.6: Testovanie ovládania

1.4.4 Pohyb v scéne

Room-scale movement [3] vo VR umožňuje používateľom pohybovať sa vo virtuálnom prostredí pomocou fyzického pohybu svojho tela. Na umožnenie tohto typu pohybu môžete použiť rôzne vstupné zariadenia, ako napríklad Oculus Guardian a ovládače Oculus Touch. Jedným zo spôsobov, ako implementovať pohyb po miestnosti, je použitie komponentu *movement-controls* a jeho konfigurácie tak, aby obmedzila pohyb používateľa na nami definovanú plochu pohybu. Dá sa to dosiahnuť nastavením atribútu *constrainToNavMesh* na hodnotu true, čo obmedzí pohyb na oblasti, ktoré boli označené pomocou navigačnej siete.

[3] Room-scale movement, dostupné na: https://aframe.io/blog/aframe-v0.3.0/

2 Rozšírenia virtuálneho prostredia

Ako vyplýva z analýzy v predchádzajúcej kapitole, hlavným cieľom našej bakalárskej práce je upraviť existujúcu aplikáciu virtuálneho prostredia tak, aby používatelia mali možnosť výberu z niekoľkých scenárov a scén, ktoré budú dostupné vo virtuálnom prostredí. Na dosiahnutie tohto cieľa je potrebné vytvoriť nové scény alebo rozšíriť existujúcu scénu o ďalšie exponáty, aby sme mohli testovať viaceré scenáre definované Petriho sieťami.

Aby bolo virtuálne prostredie pre študentov pútavé, bude dôležité zvážiť, aké konkrétne historické artefakty alebo orientačné body by boli pre nich najzaujímavejšie a najrelevantnejšie. Okrem toho bude dôležité určiť, ktoré interaktívne funkcie by boli najúčinnejšie na zapojenie študentov a podporu vzdelávania. Mohli by sme napríklad zvážiť vytvorenie preťahovateľných objektov, ktoré umožnia študentom skúmať historické artefakty a interagovať s nimi praktickejším spôsobom. Môžeme tiež zahrnúť interaktívne mapy alebo iné vizuálne pomôcky, ktoré pomôžu študentom lepšie pochopiť historický kontext scény.

2.1 Vyhotovenie otázok

Na vytvorenie pútavejšieho a interaktívnejšieho vzdelávacieho zážitku pre študentov sme identifikovali päť potenciálnych otázok, ktoré by mohli byť začlenené do virtuálneho prostredia. Tieto otázky sú sústredené okolo kľúčových historických udalostí a postáv a sú navrhnuté tak, aby povzbudili študentov, aby skúmali virtuálne prostredie a interagovali s ním praktickejším spôsobom. Otázky sme čerpali z učebnice Dejepisu pre 7. ročník ZŠ a 2. ročník gymnázia s osemročným štúdiom. Aktuálna scéna obsahuje už 3 otázky týkajúce sa výučbového celku s názvom **Habsburská monarchia**. Pri výbere otázok sme sa zamerali práve na tento celok a doplnili sme však aj ďalšie otázky, ktoré sa síce nezapádajú do spomínaného celku, ale mohli by byť zaujímavejšie pre študentov.

Pri výbere otázok sme sa hlavne zamerali na dôležité osobnosti alebo udalosti z histórie. Pri jednej z otázok sme sa zamerali významné osobnosti slovenských dejín a kultúry, ktorými boli:

- **Peter Pazman**
- **Ján Kollár**
- **Ľudovit Štúr**
- **Matej Bel**

Ďalej sme sa zamerali na vplyvné európske kráľovské rodiny z rôznych historických období:

- **Habsburgovci**
- **Wittelsbachovci**
- **Valoisovci**
- **Plantageneti**

Rozhodli sme sa pridať aj otázku s obdobia renesancie, kde sme chceli poukázať na slávnych umelcov z tohto obdobia a taktiež študentom úkázať ich najvýznamnejšie diela:

- **Leonardo da Vinci**
- **Michelangelo Buonarroti**
- **Sandro Botticelli**
- **Raffael Santi**

Slovensko sa v súčastnosti pýši veľkým množstvom hradov, zámkov, pevností. Tie v minulosti zohrávali veľkú úlohu pri obrane územia, napríklad pred útokmi tureckých vojsk:

- **Oravský hrad**
- **Bojnický zámok**
- **Hrad Strečno**
- **Hrad Stará Ľubovňa**

- **Trenčiansky hrad**
- **Hrad Krásna Hôrka**

Ako vhodným zobrazeným pevností je aj mapa, kde si študenti možu precvičiť aj svoje vedomosti z oblasti geografie. Obrázok 2.1 zobrazuje spomínanú mapu.

Obr. 2.1: Mapa s pevnostami na Slovensku

V stredoveku a ranom novoveku bolo obchodovanie medzi Európou a ostatními krajinami veľmi dôležité a vzájomne prospešné. Dovážal sa tovar, ktorý je už dnes pomerne bežný. Niektoré z najdôležitejších druhov tovarov, ktoré sa dovážali boli napríklad:

- **Hodváb**
- **Zlato**
- **Korenie**
- **Káva**
- **Voňavky**

Zhrnuli sme všetky uvedené informácie, vypracovali sme otázky a zvolili možné odpovede na ne. Pre každú otázku sme tiež určili typ interakcie, ktorý môže používateľ vo virtuálnej scéne použiť (klikanie alebo prenášanie objektov). V tabuľke 2.1 je možné vidieť zvolené otázky a príslušné dodatočné informácie k nim, aby boli pre používateľa čo najzaujímavejšie.

Tabuľka 2.1: Tabuľka otázok na rozširenie scény

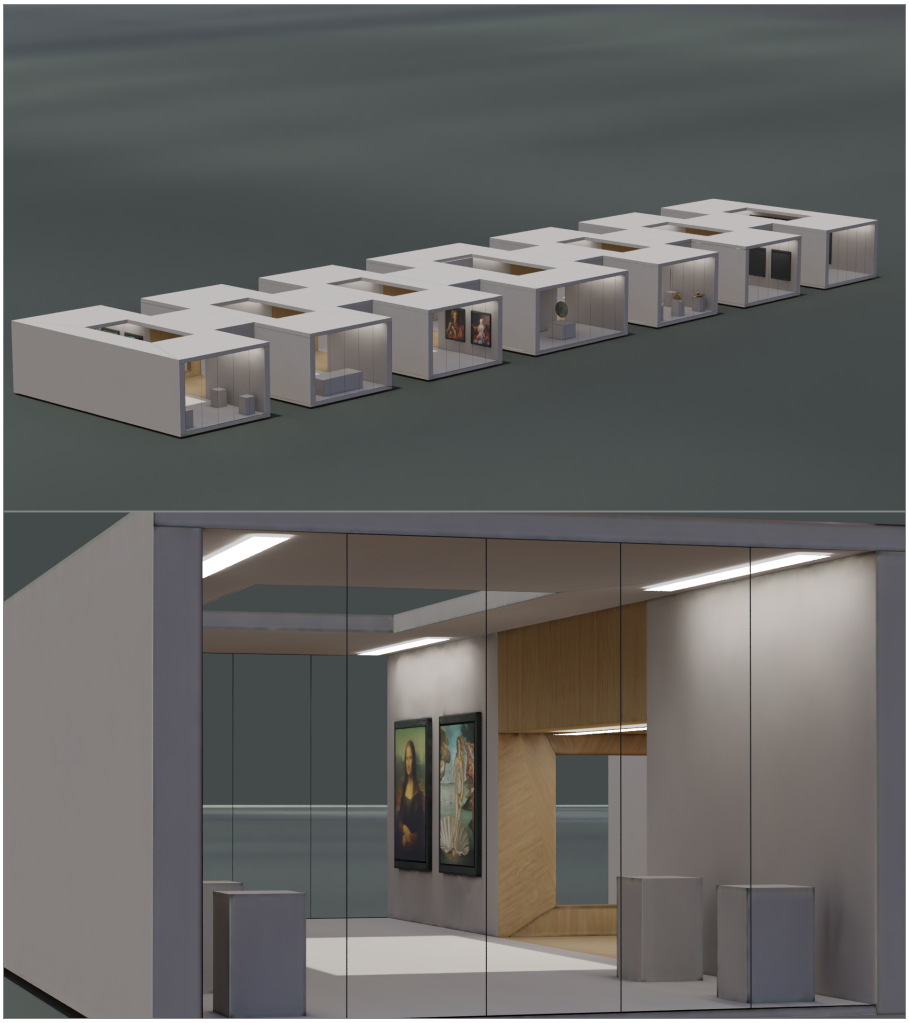
Otázka	**Správne odpovede**	**Neprávne odpovede**
O1. Aký tovar sa dovážal z krajín orientu do Európy	Korenie, prepychové latky, vonavky	Zlato, káva
O2. Ktorá panocnícka dynastia nastúpila v Uhorsku po moháčskej bitke	Habsburgovci	Wittelsbachovci, Valoisovci, Plantageneti
O3. Vyhľadajte na mape najznámejšie protiturecké pevnosti	Hrad Stará Ľubovňa, Trenčiansky hrad, Hrad Krásna Hôrka	Oravský hrad, Hrad Strečno, Bojnický zámok
O4. Ktoré osobnosti sa výrazne zaslúžili o rekatolizáciu na území Slovenska	Peter Pazman	Ján Kollár, Ľudovit Štúr, Matej Bel
O5. Priraďte mená autorov z ich slávnym dielam	Leonardo da Vinci -> Mona Lisa, Michelangelo Buonarroti -> socha Dávid, Sandro Botticelli -> Narození Venuše, Raffael Santi -> Sixtínska Madonna	-

2.2 Model múzea

Po dôkladnom zozbieraní všetkých potrebných údajov sme prešli k úprave existujúceho múzejného modelu. Naším cieľom bolo, aby bol model schopný zahrnúť všetky vyššie spomenuté otázky s požadovanými typmi interakcií a zároveň aby bol pre návštevníkov čo najzaujímavejší a najužitočnejší.

Pri návrhu modelu sme použili program Blender [1], ktorý nám umožnil nielen modelovať a vytvárať vizuálnu podobu múzea, ale aj zmenšiť jeho veľkosť a optimalizovať ho pre plynulejší zážitok v scéne. Na dosiahnutie tohto cieľa sme využili UV mapovanie a svetelné mapy, ktoré nám umožnili lepšie spracovanie a optimalizáciu modelu. Snažili sme sa tiež zabezpečiť, aby bol model čo najjednoduchší a zrozumiteľný pre návštevníkov. Výsledný model múzea, ako aj ukažky niektorých miestnosti je možné vidieť na koláži 2.2.

[1] Domovská stránka Blender, dostupné na: https://www.blender.org



Obr. 2.2: model rozšírenia scény

2.3 Návrh Petriho siete

Táto podkapitola sa zaoberá návrhom špecifickej Petriho siete pre implementáciu virtuálnej scény. Okrem finálnych stavov, naša metóda zahŕňa kontrolu každej jednej interakcie používateľa v scéne. Týmto spôsobom dosahujeme veľkú výhodu priamo v hernom zážitku, keďže nemusíme súčasne vyvíjať samostatný overovací model. Namiesto toho môžeme pomocou dobre navrhnutej Petriho siete zabezpečiť overenie priamo pri návrhu.

Na základe otázok, ktoré sme vytvorili, je potrebné vhodne rozšíriť Petriho sieť, ktorá bude použitá pre rozšírenú scénu. Vytvorili sme dva scénare. Jeden pre základnú scénu, kde sme len upravili pôvodnú sieť podľa novej architektúry, a druhý scénár, kde sme do základnej siete pridali spomínané otázky. V nasledujúcej podkapitole sa budeme zaoberať iba rozšíreným scénárom.

2.3.1 Navrhnutý scénar rozšírenia siete

V tejto podkapitole sa budeme venovať opisu navrhnutej Petriho siete, spomenieme hlavné časti siete ako aj samotnú logiku fungovania siete. Virtuálna scéna dokopy obasahuje 7 otázok, ktoré sú v sieti definované ako miesta s nasledujúcim označením:

- **Monarch**
- **Reforms**
- **Jewels**
- **Import**
- **Dynasty**
- **CounterReform**
- **TurkStrike**

Každá otázka obsahuje niekoľko exponátov. Exponáty sú rozdelené na správne a nesprávne. V Petriho sieti taktiež uchovávame informácie o počte správne a nesprávne označeních exponátov. Týmto spôsobom vieme vyhodnotiť správnosť zodpovedanej otázky. Napríklad otázka týkajúca sa dynastiám obsahuje miesta a prechody zobrazené v tabuľke 2.2.

Ak používateľ označí správny exponát v scény vyvolá vykonanie prechodu **DynastyCselect**. Po úspešnom odpálení prechodu sa do miesta **DynastyCselected** pridá jeden žetón a z miesta **DynastyCorrect** sa žetón odoberie. V prídade že užívateľ zruší označenie zvoleného exponátu, tak sa vykoná prechod **DynastyIselect**, a odoberie sa žetón z miesta **DynastyCselected** a do miesta **DynastyCorrect** sa žetón pridá.

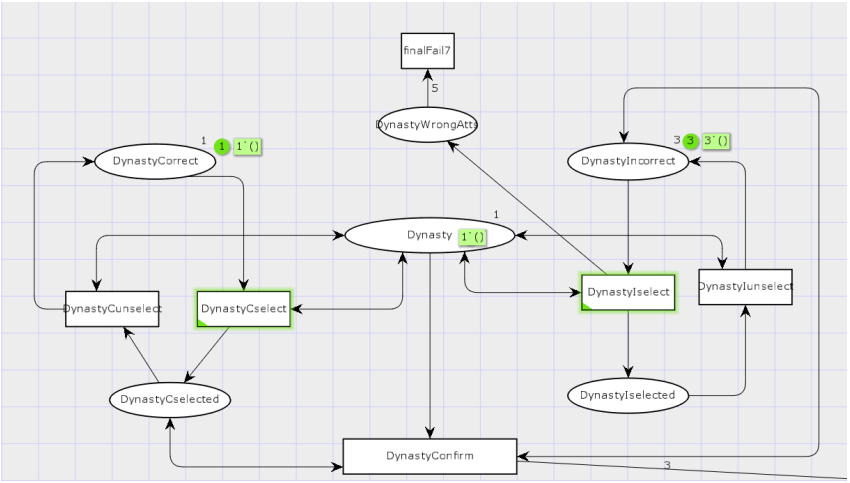
Rovnakým spôsobom funguje aj vyberanie nesprávnych exponátov pre miesta **DynastyIncorrect**, **DynastyIselected** a prechody **DynastyIselect**, **DynastyIunselect** s jediným rozdielom že pri každom označení nesprávneho exponátu sa pridá aj žetón do miesta **DynastyWrongAttempts**, čo pri postučujúcom počte žetónou môže odpáliť prechod **finFailDynasty** a tak aj ukončenie scenára.

Každé z miest obsahuje pri počiatočnom značení jeden token. Tento token sa môže presunúť do miesta **final**, pokial je **DynastyConfirm** prechod úspešne odpálený. Pokial je na mieste **final** dostatočný počet žetónou, tak je možné odpálenie prechodu **finalSucc**, ktorý ukončuje scénar správnym spôsobom.

Tabuľka 2.2: Miesta a prechody týkajúce sa otázky s dynastiami

Prechod	**Význam**
DynastyCselect	Označenie správneho exponátu
DynastyCunselect	Zrušenie označenia správneho exponátu
DynastyIselect	Označenie nesprávneho exponátu
DynastyIunselect	Zrušenie označenia nesprávneho exponátu
DynastyConfirm	Potvrdenie odpovede
finFailDynasty	Ukončenie scénara nesprávnym koncom
Miesto	**Význam**
Dynasty	Miesto úlohy
DynastyCorrect	Stav reprezentujúci neoznačené správne exponáty
DynastyIncorrect	Stav reprezentujúci neoznačené nesprávne exponáty
DynastyCselected	Stav reprezentujúci označené správne exponáty
DynastyIselected	Stav reprezentujúci označené nesprávne exponáty
DynastyWrongAttempts	Stav reprezentujúci množstvo nesprávnych opovedí

Opísaný postup tvorí základnú logiku vytvorenej Petriho siete. Postup neslúži len pre otázku týkajúcu sa dynastií, na ktorej sme to znázornili ale aj pre každú otázku v sieti, keďže postup je rovnaký. Grafické znáronenie uvedeného postupu v podobe časti Petriho siete je možné vidieť na obrázku 2.3.



Obr. 2.3: Ukončenie scénara v Petriho sieti so zlým koncom

3 Návrh architektúry riešenia

Naša práca spočíva v rozšírení existujúcej aplikácie, preto pri návrhu novej architektúry budeme vychádzať z už existujúcej. Tú však upravíme tak, aby vyhovovala novým požiadavkám, ktoré sú nevyhnutné pre implementáciu. V nasledujúcej kapitole podrobne opíšeme navrhovanú architektúru a jej úpravy oproti pôvodnej.

3.1 Funkčné požiadavky

Hlavným cieľom vylepšenia súčasného systému je umožniť koncovým užívateľom vytvárať vlastné úlohy. Úloha je činnosťou používateľa v rámci virtuálnej scény, ktorá je riadená príslušným scenárom. Virtuálna scéna je interaktívna 3D scéna vytvorená pomocou A-Frame, a scénar je Petriho sieť spolu s textovými popiskami v rôznych jazykoch. Tieto popisky sú textové informácie zobrazujúce sa v scéne. Je dôležité uchovávať ich oddelene, pretože rovnaká závislá udalosť môže mať rôzny význam v rôznych scenároch tej istej scény.

Existujúce riešenie umožňuje ovládanie scény prostredníctvom Petriho siete, avšak nie je univerzálne a nemožno ho použiť na akúkoľvek scénu. Preto je nevyhnutné prepracovať ovládanie Petriho siete tak, aby bolo univerzálne a použiteľné na akúkoľvek scénu.

Ďalšou požiadavkou je zaznamenávanie aktivity užívateľa počas vykonávania úlohy. Pomocou záznamu budeme mať presný prehľad o postupe, ktorý užívateľ nasledoval pri vykonávaní úlohy. Tento záznam nám bude slúžiť najmä na vytváranie štatistík a tiež na obnovu scény zo záznamu, aby užívateľ mohol pokračovať v úlohe aj po obnovení stránky.

3.2 Výber technológií a nástrojov

Keďže naša práca je pokračovaním už vytvorenej aplikácie, tak budeme používať aj technológie použité pri jej implementácií. Virtuálne prostredie bolo vytvorené

v rámci Aframe[1], ktorý je vykonávaný v samotnom priehladači. Aframe prináša množstvo výhod a umožňuje vytvárať interaktívne virtuálne scény, ktoré môžu byť použité v rôznych oblastiach.

Keďže sme na tomto projekte v tíme, musíme zvoliť riešenie, ktoré bude vhodné pre všetkých členov tímu a umožní nám ľahko pridávať ďalšie funkcionality do existujúceho riešenia. V tomto ohľade sme sa rozhodli použiť webový rámec Angular [2], ktorý nám poskytne potrebné nástroje a prostriedky na vytvorenie nového užívateľského rozhrania. Existuje množstvo iných webových rámcov, ktoré by sme mohli použiť, ale pre náš projekt nám prišiel najvhodnejší práve Angular, pretože je to robustný a veľmi populárny rámec, ktorý poskytuje veľa vynikajúcich funkcií a nástrojov.

3.3 Nová architektúra

Pri návrhu novej architektúry sme sa rozhodli zachovať základnú ECS (Entity-Component-System) architektúru, ktorú sme využívali v pôvodnej aplikácii. ECS umožňuje oddelenie dátových entít (entity) od ich správania (component) a systémov, ktoré tieto entity spracúvajú. Novú architektúru sme obohatili o webový rámec Angular, ktorý nám poskytuje robustné nástroje na tvorbu užívateľského rozhrania. Angular nám umožňuje efektívne spravovať komponenty a moduly, čo je veľmi dôležité pri rozsiahlom projekte. Celkové rozdelenie aplikácie sme realizovali na tri logické celky, kde moduly, komponenty a skripty súvisia:

- **so scénou**
- **s Petriho sietami**
- **s komunikáciou s užívateľom a serverom.**

Na obrázku 3.1 je zobrazená základná štruktúra projektu, kde sú zobrazené iba najpodstatnejšie súbory, ktorými sa budeme zaoberať v najsledujúcej kapitole týkajúcej sa implementácie. Medzi angulárové komponenty patria aj samotné scény, pričom každá scéna obsahuje scripty, komponenty, ktoré sú špecifické iba pre danú scénu. Komponenty, ktoré sú použiteľné pre viacoro scén, sú definované v priečinku *js*. V štruktúre je zobrazená iba časť celkovej štruktúry projektu, zvýšné súbory a ich implementácia je obsiahnutá v práci [18] od Ing. Adama Kašelu a [22] od Dmytra Demianenka.

[1]Domovská stránka Aframe, dostupné na: https://aframe.io
[2]Domovská stránka Angular, dostupné na:https://angular.io

Obr. 3.1: Súborová štruktúra projektu

4 Implementácia

V tejto kapitole sa venujeme implementácií vyššie spomenutých súborov, modulov, komponentov. Kapitola je rozdelená do podkapitol podľa logického rozdelenia aplikácie.

4.1 Petriho siete

Petriho siete sú využívané na modelovanie a riadenie procesov v aplikácii. V rámci celku, ktorý sa zaoberá Petriho sieťami sme implementovali nasledujúce súbory:

- **cpnLoader.mjs** - Skript pre načítanie a inicializáciu Petriho siete zo súboru vo formáte cpn.
- **petriNetSim.component.js** - Komponent zodpovedný za simuláciu a riadenie scény pomocou Petriho sieti.

Tieto súbory poskytujú funkcionalitu pre prácu s Petriho sieťami v rámci aplikácie.

4.1.1 Ovládanie scény pomocou Petriho sietí

Prechody v Petriho sieti sú stavové udalosti, ktoré sa môžu stať, keď sa dosiahne určitá kombinácia vstupných podmienok (označených ako tokeny) v sieti. Týmto prechodom môžu byť pridelené akcie, ktoré sa majú vykonať po jeho odpálení. Ak sa prechod podarí odpáliť, zmenia sa tokeny v sieti a môže sa vykonať ďalší prechod.

V prípade interaktívnej 3D scény môžu byť prechody v sieti naviazané na rôzne udalosti v scéne, ako napríklad kliknutie na objekt, jeho presunutie. Pri výskyte takejto udalosti v 3D scéne sa overí, či sa v sieti nachádza prechod, ktorý je naviazaný na túto udalosť. Ak áno, tak sa pokúsi odpáliť daný prechod a vykonať priradené akcie.

4.1.2 Skripty scény

Manipulácia s Petriho sieťami v kontexte dynamických scén môže byť zložitá úloha, ktorá si vyžaduje starostlivé zváženie toho, ako bude sieť interagovať s meniacim sa prostredím. V našej implementácii navrhujeme prístup, ktorý oddeľuje funkčnosť manipulácie s Petriho sieťami od následných zmien scény, ktoré sú jedinečné pre každú scénu.

Túto architektúru pôvodne vymyslel Ing. Štefan Korečko, PhD. počas konzultácie, kde navrhol vytvorenie poľa prechodov definovaných Petriho sieťou, kde každý prechod je reprezentovaný objektom obsahujúcim vlastnosti, ktoré definujú správanie prechodu. Vlastnosť **transitionName** špecifikuje názov prechodu. Vlastnosti **ifTransitionFound**, **ifTransitionEnabled**, **ifTransitionDisabled** a **ifTransitionNotFound** sú funkcie, ktoré definujú akcie vykonávané na základe stavu prechodu.

```
1 {
2   transitionName: String,
3   ifTransitionFound: function(targets, replay),
4   ifTransitionEnabled: function(targets, replay),
5   ifTransitionDisabled: function(targets, replay),
6   ifTransitionNotFound: function(targets, replay)
7 }
```

Zdrojový kód 4.1: Objekt prechodu

Parametre týchto funkcií sú nasledovné:

- **targets** - je pole reťazcov, ktoré obsahuje identifikátory prvkov v scéne, s ktorými bude manipulácia pracovať. Týmto umožňujeme priradenie rovnakého prechodu rôznym prvkom v scéne.
- **replay** - je boolean hodnota, ktorá označuje, či sa funkcie vykonávajú na základe používateľskej aktivity (hodnota true) alebo zo záznamu (hodnota false).

Tento prístup nám umožní ľahko upraviť správanie každého prechodu bez ovplyvnenia celkovej Petriho siete a zjednoduší proces implementácie zmien scény.

Podobne sme prispôsobili aj manipuláciu s miestami vymedzenými Petriho sieťou. Na rozdiel od prechodov sa miesta nedajú odpáliť a teda môžeme kontrolovať iba či sa dané miesto nachádza v Petriho sieti.

```
1 {
2     placeName: String,
3     ifPlaceFound: function(placeName),
```

```
4       ifPlaceFoundOnStart: function,
5       ifPlaceNotFoundOnStart: function
6   }
```

Zdrojový kód 4.2: Objekt miesta

Aby sme to dosiahli, vytvoríme pole miest definovaných Petriho sieťou, kde každé miesto je reprezentované objektom obsahujúcim vlastnosť **placeName**, ktorá špecifikuje názov miesta, a vlastnosť **ifPlaceFound**, ktorá definuje akcie, ktoré budú vždy vykonané, keď je na mieste prítomný žetón. Taktiež sme pridali vlastnosti **ifPlaceFoundOnStart** a **ifPlaceNotFoundOnStart**, ktoré sa vykonajú iba pri načítaní scény. Príklad využitia funkcií pri načítaní scény je možné vidieť na obrázku 4.1, kde sa na miesta, v scéne, ktoré však niesu v scénari sa umiestnia bariery, ktoré zabránia užívateľovi v interakciíí s danou časťou scény.

Obr. 4.1: Nedostupná časť scény

Polia prechodov a miest môžu byť pre každú scénu rozdielne, preto je ich potrebné oddeliť od samotného simulátora, ktorý realizuje riadenie scény, ten je pre všetky scény rovnaký. Spomínané polia tak vytvoríme pre samostatne pre každú scénu ako je možné vidieť v štruktúre projektu 3.1, pod názvom *sceneScript.ts*.

4.1.3 Realizácia obsluhy

Riadenie scény na základe interakcie užívateľa s Petriho sietou rieši komponent *petriNetSim*. Interakciu so scénou môže užívateľ vyvolať iba jediným spôsobom a to kolíziou objektov, a to buď kurzorom alebo samotným objektom hráča. Každá

takáto interakcia v scéne vyvolá udalosť v ktorej sú definované prvky scény, ktoré kolíziu vyvolali. Komponenty *clkMultiEventHandler*, *clkSingleEventHandler*, *collisionDetectorEventHandler* používame na poslúchanie takýchto udalosti. Tie po zachytení udalosti adekvátne upravia prvky, v našom prípade ich nazývame **targets** a tie následne odošlú spolu s typom akcie a názvom prechodu alebo miesta na komponent *petriNetSim*, ktorý už rieši samotnú obsluhu Petriho siete.

Realizácia obsluhy, vykonávaná v hlavnej slučke 3D scény (update) potom bude nasledovná:

1. Zisti, či Petriho sieť obsahuje prechod *transitionName*.Ak nie, choď na krok 5.
2. Vykonaj funkciu *ifTransitionFound*.
3. Ak je prechod *transitionName* odpáliteľný, odpáľ ho a vykonaj funkciu *ifTransitionEnabled*. Choď na krok 6.
4. Ak prechod *transitionName* nie je odpáliteľný, vykonaj funkciu *ifTransitionDisabled*. Choď na krok 9.
5. Vykonaj funkciu *ifTransitionNotFound*. Choď na krok 9.
6. Medzi finálnymi prechodmi hľadaj odpáliteľný prechod.
7. Ak nenájdeš odpáliteľný finálny prechod, choď na krok 9.
8. Ak nájdeš odpáliteľný finálny prechod, obslúž ho podľa tohto postupu, okrem krokov 6-8.
9. Koniec obsluhy.

4.1.4 Ukončovanie úlohy v scéne

Jedným z nevyriešených problémov pri ovládaní Petriho sietí v scéne bolo, ako ju ukončiť na základe určitých kritérií. Pôvodne navrhnutá sieť obsahovala logiku, ktorá umožňovala ukončiť Petriho sieť iba s úspešným výsledkom. Dosiahlo sa to zadaním názvu miesta v ovládacom prvku Petriho siete, ktoré signalizovalo koniec, ako aj počtu žetónov, ktoré musia byť na tomto mieste prítomné, aby nastal koniec.

Aby sa však scéna stala dynamickou a umožnila používateľom pridať do scény akýkoľvek scenár, vrátane tých, ktoré sa nemusia skončiť úspešne (napríklad nesprávne zodpovedaná otázka), je potrebné prehodnotiť logiku vytvárania a ukončovania Petriho sietí v scéne.

Jedným zo spôsobov, ako to dosiahnuť, je upraviť Petriho sieť tak, že namiesto toho, aby sme končili podľa miest s určitým počtom žetónov, vytvárame prechody s jednotným názvom začínajúcim na „final“. Tieto prechody sa budú líšiť od ostatných prechodov v scéne a vždy, keď sa spustí jeden z týchto prechodov, bude to znamenať koniec aktuálnej používateľskej úlohy. Prechody označujúce koniec scenára budú uložené v rovnakom poli ako ostatné prechody, ale pri spustení budú spúšťať rôzne funkcie.

Finálne prechody majú nasledujúce podmienky:

- **ich odpálenie znamená koniec scenára.**
- **od ostatných sa odlišujú začiatkom mena:**
  - ***finalSucc*** - úspešný koniec.
  - ***finalFail*** - neúspešný koniec.
- **nesmú byť priradené udalostiam v scéne.**
- **vždy by mal byť vykonateľný maximálne jeden z nich.**

Aby to fungovalo, budeme musieť neustále sledovať prechody spojené s ukončením scenára, aby sme skontrolovali, či sa dajú spustiť. To zaisťuje, že Petriho sieť bude správne ukončená bez ohľadu na výsledok užívateľskej úlohy.

4.1.5 Vykonávanie obsluhy zo záznamu

Okrem vvvolania vykonania obsluhy ativitou používateľa k nej môže dôjsť aj zo záznamu. K tejto situácii dochádza vtedy, keď je potrebné, napríklad z dôvodu výpadku spojenia, znovu načítať scénu, v ktorej má používateľ už rozpracované riešenie scenára. Ak by sme túto situáciu nijako neriešili, musel by používateľ po načítaní scény všetky kroky v nej vykonať znova. My však aktivitu používateľa v scéne zaznamenávame do databázy. O každom pokuse o odpálenie prechodu vieme:

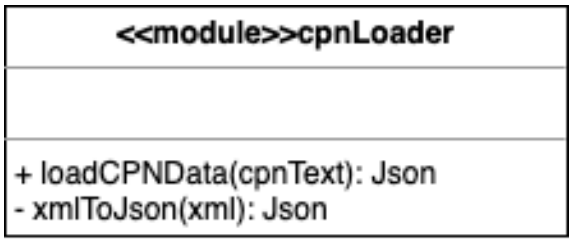
- **či bolo odpálenie úspešné**
- **aký prechod (*transitionName*) a na akých elementoch (*targets*) bol odpálený.**

Podľa tohto záznamu vieme replikovať celú doterajšiu činnosť používateľa. Sú však činnosti pri ktorých obsluha nemôže byť rovnaká, ako keď aktivitu robil

priamo používateľ. Napríklad keď pokus o odpálenie prechodu vyvolala aktivita spojená s prenášaním predmetu v scéne. Vtedy pri replikovaní zo záznamu musíme programovo vykonať aj samotné prenášanie, ktoré by normálne urobil používateľ. Preto majú funkcie vstupný parameter ***replay***, vďaka ktorému vieme špecifikovať ktorá časť kódu sa vykoná iba pri replikovaní činnosti zo záznamu.

4.1.6 Načítavanie Petriho siete

Pri implementácii zmien na Petriho sieti, ktoré zahŕňali definovanie nových ukončovacích podmienok a pridávanie ďalších otázok do siete, sme narazili na neočakávanú chybu pri používaní programu CPN Tools pre návrh Petriho siete. Konkrétne pri pokuse o uloženie siete v požadovanom formáte pnml program vygeneroval chybu počas exportu. Tento problém pravdepodobne súvisel s veľkosťou samotnej Petriho siete, ktorá mohla presiahnuť limity programu. Túto chybu sme nedodokázali opraviť a taktiež nebolo možné použiť už implementovaný modul *pnmlLoader* na načitávanie petriho sietí vo formáte pnml. Boli sme tak donútený spraviť nový modul na načítanie siete vo formáte cpn. Nasledujúci obrázok 4.2 znázorňuje modul, ktorý bol vytvorený na získanie potrebných údajov Petriho siete.



Obr. 4.2: Modul načítania Petriho siete

Na získanie údajov o Petriho siete v správnom formáte sme použili funkciu *loadCPNData*. Táto funkcia vezme súbor vo formáte cpn a vráti informácie o miestach, prechodoch a hrán ako objekt. Na čítanie informácií z cpn formátu sme použili funkciu *DOMParser*. Potom sme z toho zobrali relevantné informácie a vložili ich do polí. Vytvorili sme napríklad pole všetkých miest, prechodov a hrán, ako je možné v nasledujúcom výpise.

```
1     NetData = {
2         arcs: [
3             {
4                 id: "ID1436204442",
5                 markingWeight: 4,
6                 source: "ID1436203908",
```

```
                target: "ID1436203277"
            },
            ...
        ],
        places: [
            {
                id: 'ID1438140640',
                name: 'P7WrongAtts',
                marking: 0
            },
            ...
        ],
        transitions: [
            {
                id: 'ID1438093596',
                name: 'finSucc'
            },
            ...
        ]
    }
```

Zdrojový kód 4.3: Objekt Petriho siete

4.2 Scény

Scény predstavujú virtuálne prostredie, v ktorom sa odohráva interakcia medzi užívateľom a objektmi. Každá scéna je tvorená súbormi, ktoré ju definujú a ovládajú. Tieto súbory sú umiestnené v priečinku *scenes* a sú organizované podľa jednotlivých scén.

Každá scéna obsahuje nasledujúce súbory:

- **sceneScript.ts** - Skript definujúci správanie objektov v scéne. Obsahuje logiku pre ovládanie objektov, spracovanie udalostí a interakciu s užívateľom.
- **scene.component.css** - Štylový súbor pre vizuálnu úpravu scény.
- **scene.component.html** - HTML šablóna, ktorá definuje štruktúru a rozloženie prvkov scény.
- **scene.component.spec.ts** - Unit testy pre scénu.
- **scene.component.ts** - Komponent reprezentujúci scénu, ktorý zahŕňa logiku a správu scény.

Komponenty scény využívajú komponenty definované v priečinku *components*, ako napríklad *clkMultiEventHandler.component.js*, *collisionDetectorEventHandler.component.js*, *label.component.js* a ďalšie. Tieto komponenty sa starajú o konkrétne aspekty správania objektov v scéne, ako spracovanie udalostí, detekciu kolízií, zobrazenie textových popiskov a ďalšie.

4.2.1 Komponent jazykovej verzie scény

Na správne sprostredkovanie informácií používateľom v rôznych jazykoch je potrebné ukladať samostatné súbory obsahujúce popisy v každom jazyku. Oddelené uloženie týchto popisov pre rozdielne scénare je dôležité, pretože tá istá závislá udalosť môže mať rôzne významy pre rôzne scenáre v rámci tej istej scény.

Na uľahčenie tohto ukladania sa popisy ukladajú ako textové súbory vo formáte JSON. Každý popis bude zodpovedať aspoň jednému prechodu Petriho siete, hoci nie každý prechod bude vyžadovať popis.

```
1    {
2        "nazovPopisky1": "textPopisky1",
3        "nazovPopisky2": "textPopisky2",
4        ...
5    }
```

Zdrojový kód 4.4: Objekt popiskov

Zatiaľ čo popisy sa primárne používajú na sprostredkovanie informácií o scéne a jej relevantných objektoch, môžu existovať ďalšie popisy spojené s neúlohovými objektmi alebo exponátmi v rámci scény. Tieto popisy nie sú potrebné na dokončenie úlohy, ale poskytujú doplňujúce informácie, ktoré môžu byť pre používateľov užitočné alebo zaujímavé.

Schéma komponentu pre jazykové súbory je definovaná takto:

```
1    schema: {
2        languageFile: {type: 'string'},
3    }
```

Zdrojový kód 4.5: Schéma komponentu

Schéma poskytnutá v predchádzajúcom texte sa používa pre komponent AFRAME, ktorý je pripojený k scéne. Tento komponent je zodpovedný za zmenu textových hodnôt v scéne na základe kľúča poskytnutého v súbore jazyka JSON.

Je zodpovednosťou tvorcu úlohy alebo scenára priložiť jazykový súbor, aby ho používatelia mohli používať. Ak sú poskytnuté viaceré jazykové súbory, používatelia si môžu vybrať, ktorú jazykovú verziu chcú používať. To umožňuje používateľom interagovať s úlohou alebo scenárom v jazyku, ktorý im vyhovuje.

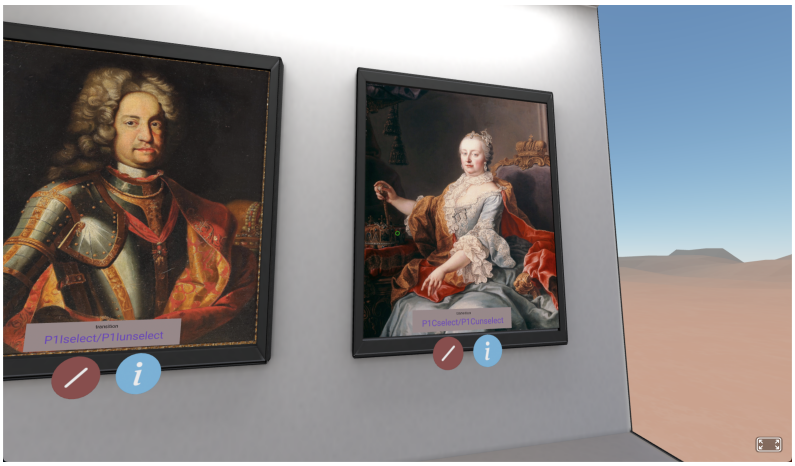
Keď si používateľ vyberie jazyk, z databázy sa príjíme príslušný jazykový súbor a komponentu nastavíme atribút podľa získaného súboru. To zabezpečí, že scéna zobrazí správnu jazykovú verziu pre používateľa. Dodržiavaním týchto postupov je možné efektívne ukladať a zobrazovať popisy vo viacerých jazykoch.

```
1    this.scene.nativeElement.setAttribute(
2        'language',
3        `languageFile: ${this.taskFiles.languageFile}
4    `)
```

Zdrojový kód 4.6: Pridanie jazykového súboru do komponentu

4.2.2 Zobrazovanie popiskov Petriho siete v scéne

Pre zlepšenie prehľadnosti v Petriho sietí sme pre tvorcov a editorov scenárov vytvorili komponent v rámci virtuálnej scény, ktorý slúži na zobrazovanie názvov prechodov a miest, ktoré ovplyvňujú konkrétne prvky v scéne.



Obr. 4.3: ukážka popisku v scéne

Použitie tohto komponentu je jednoduché, stačí ho pridať ako entitu v scéne s pozíciou, na ktorú chceme popisok umiestniť a nastaviť jeho typ a samotný text ako atribút v rámci schémy komponentu, podobne ako je to zobrazené na fragmente kódu. Komponent navyše zabezpečuje, že iba používatelia s príslušnými oprávneniami budú mať možnosť vidieť daný popisok, a to na základe ich roly. Rola používateľa je uložená v lokálnom úložisku pod kľúčom *user-profile*, pričom povolené role na prezeranie štítkov sú definované v poli v rámci samotného komponentu, v našom prípade sú to role *TEACHER* a *ADMIN*.

```
1    <a-entity
2        label="label: P1confirm"
3        position="0 0.576 0.01">
```

```
4     </a-entity>
```

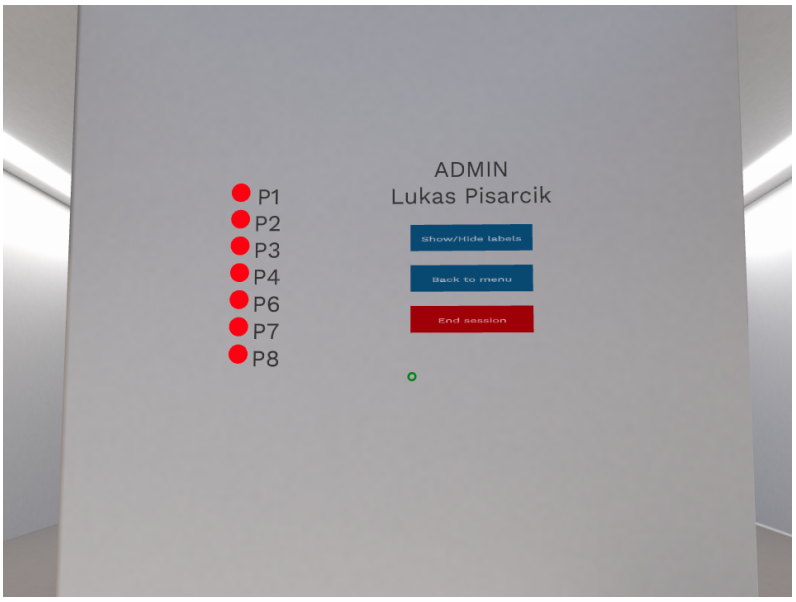
Zdrojový kód 4.7: Použitie komponentu label

Ak sa aktuálna rola používateľa nachádza medzi uvedenými rolami v poli, štítok sa zobrazí. V opačnom prípade bude štítok skrytý, čím sa zabezpečí, že obsah štítku bude viditeľný iba pre oprávnených používateľov a zachová sa tak bezpečnosť a súkromie.

Okrem toho, komponent obsahuje aj event listener, ktorý reaguje na udalosť *toggleLabelVisibility*. Týmto spôsobom sa viditeľnosť štítkov môže dynamicky meniť medzi viditeľným a skrytým stavom, čo pridáva komponentu ďalšiu vrstvu flexibility a umožňuje lepšiu kontrolu viditeľnosti štítkov v rámci celej scény.

4.2.3 Ovladacie centrum scény

Komponent *dashboard* slúži ako riadiace centrum pre používateľa v rámci aplikácie. Používatelia majú z ovládacieho centra prístup k rôznym funkciám vrátane možnosti ukončiť aktuálnu reláciu, vrátiť sa na hlavnú stránku a zobraziť informácie o používateľovi a stav jednotlivých úloch scénara. Komponent je integrovaný do aplikácie Angular a funkcia návratu na hlavnú stránku závisí od globálnej premennej router, ktorý zaisťuje navigáciu. V tomto prídade sme boli nútený použiť globálnu premennú na objekte window. Okrem týchto základných funkcií komponent dashboard tiež umožňuje používateľom prepínať viditeľnosť komponentu *label*. Túto možnosť majú iba užívatelia, ktorí majú na viditeľnosť komponentu oprávnenie.



Obr. 4.4: Ovladacie centrum scény

4.3 Komunikácia so serverom

Pre komunikáciu s užívateľom a serverom sme implementovali súbory:

- **userActivityLogger.js** - Tento súbor obsahuje implementáciu skriptu pre zaznamenávanie aktivity užívateľa počas vykonávania úlohy. Jeho hlavným účelom je zbierať informácie o aktivitách užívateľa a zasielať ich na server pre ďalšie spracovanie.
- **task-files.model.ts** - Tento súbor obsahuje definíciu modelu pre súbory úloh. Model popisuje štruktúru a atribúty súborov, ktoré sa používajú pri komunikácii so serverom a pri práci s úlohami v aplikácii.
- **backend.service.ts** - Tento súbor obsahuje implementáciu služby pre komunikáciu so serverom. Služba poskytuje metódy a funkcie pre odosielanie požiadaviek na server a spracovanie odpovedí.
- **scene.component.ts** - Tento súbor aj okrem iného zabezpečuje načítanie súborov potrebných na vykonanie scenára.

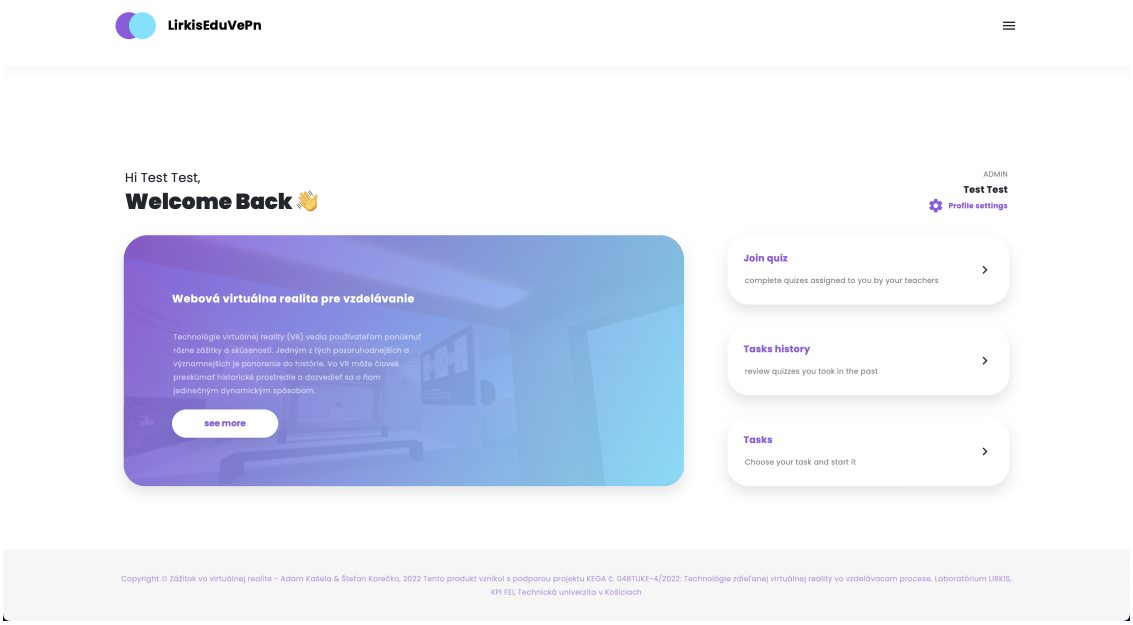
Tieto súbory spolu tvoria funkcionalitu pre komunikáciu so serverom v aplikácii. Sú zodpovedné za zasielanie a prijímanie dát, spracovanie odpovedí a zabezpečenie plynulej interakcie medzi klientom a serverom.

4.3.1 Načítavanie súborov na ovládanie scény

Aby sme diplomovú prácu Ing. Adama Kašelu zlepšili a urobili ju užívateľsky príjemnejšou pre koncových používateľov, musíme pridať funkcionalitu, ktorá dynamicky načítava scenáre, aby si používatelia mohli prispôsobiť scenár podľa vlastných potrieb. Okrem toho by používatelia mali byť schopní vytvoriť alebo vybrať jazykový súbor, ktorý obsahuje texty nachádzajúce sa v scéne (napr. znenie otázky).

Na implementáciu tejto funkcionality potrebujeme vytvoriť nové používateľské rozhranie, ktoré umožní používateľom vytvárať si vlastné scenáre, ktoré budú obsahovať súbor CPN s Petriho sieťami a súbor JSON s textovými popiskami. Keďže naše riešenie je univerzálne a nie je obmedzené na použitie iba jednej scény, používatelia si budú môcť vybrať aj z ponuky scén vytvorených v Aframe.

Vytvorenie takéhoto scenára spolu s príslušnou scénou sa bude označovať ako „úloha“. Úloha je činnosť vykonávaná používateľom (zvyčajne študentom) vo virtuálnej scéne, riadená zodpovedajúcim scenárom.



Obr. 4.5: návrh hlavnej obrazovky používateľského rozhrania

Používatelia nášho systému sa budú musieť pred jeho použitím prihlásiť. Registrácia a prihlásenie sú potrebné na to, aby používatelia mohli vytvárať, upravovať alebo odstraňovať úlohy, prideľovať úlohy iným používateľom a sledovať ich aktivitu pri plnení jednotlivých úloh. Títo používatelia budú mať tiež priradené roly, ktoré sú nasledovné:

- **Študent:** Táto rola umožňuje používateľom riešiť vybrané úlohy, ktoré im boli pridelené, ako aj prezerať a analyzovať ich výsledky. Môžu tiež upravovať informácie o svojom vlastnom používateľskom účte.
- **Učiteľ:** Táto rola umožňuje používateľom vytvárať, upravovať a odstraňovať úlohy, priraďovať alebo odvolávať úlohy študentom a prezerať a analyzovať výsledky študentov a ich vlastných riešení úloh. Môžu tiež upravovať informácie o svojom vlastnom používateľskom účte.
- **Administrátor:** Táto rola má všetky privilégiá učiteľa s ďalšou možnosťou pridávať, odstraňovať a upravovať používateľov, vrátane zmeny ich roly v systéme.

```
1 constructor(
2     private _route: ActivatedRoute,
3     private _client: BackendService) {
4
5         this._route.params.subscribe(params => {
6           this.taskId = params['taskId'];
```

```
7         })
8     this.task_files_subscription = this._client.getTaskFiles(this.
          taskId)
9         .subscribe(data => {
10          this.taskFiles = data as TaskFiles;
11          TaskFiles.decode(this.taskFiles);
12          if (this.taskFiles) {
13            const petriNetSimAttr = `pnmlFile: ${this.taskFiles.
                  pnmlFile}`;
14            this.scene.nativeElement
15            .setAttribute('petri-net-sim', petriNetSimAttr);
16          }
17        })
18  }
```

Zdrojový kód 4.8: Konštruktor scény

Tento úryvok kódu definuje funkciu konštruktora pre komponent Angular 'SceneComponent'. Konštruktor inicializuje závislosti 'ActivatedRoute' a 'BackendService' tak, že ich prijme ako parametre.

Vo vnútri konštruktora sa 'ActivatedRoute' používa na prihlásenie na odber pozorovateľných 'parametrov', čo poskytuje prístup k aktuálnym parametrom cesty. Konkrétne sa získa parameter 'taskId' a uloží sa do premennej 'taskId' komponentu.

Po získaní 'taskId' sa 'BackendService' použije na získanie príslušných údajov súborov úloh z backendu pomocou metódy 'getTaskFiles()'. Získané údaje sa potom prihlásia na odber pomocou metódy 'subscribe()' a uložia sa do premennej 'taskFiles' komponentu. Metóda 'TaskFiles.decode()' sa potom použije na dekódovanie údajov obnovených súborov úloh.

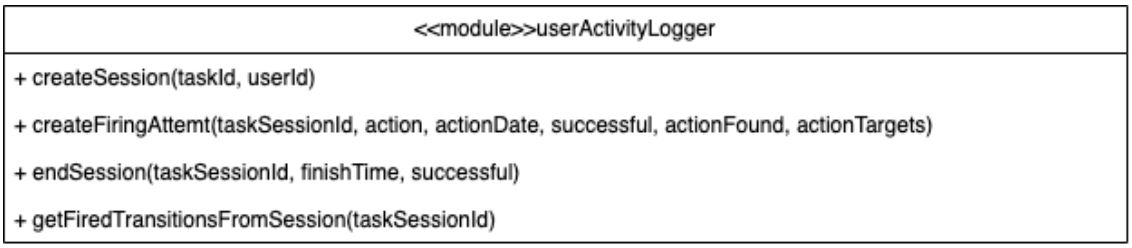
Ak boli údaje súborov úloh načítané úspešne, atribúty 'finalPlace', 'taskCount' a 'pnmlFile' prvku 'petri-net-sim' sa nastavia vhodne pomocou získaných údajov a metóda 'setAttribute()' používa sa na nastavenie atribútov prvku 'scene'.

4.3.2 Zaznamenavanie aktivity užívateľa

Zakaždým, keď používateľ interaguje so scénou a ovplyvní Petriho sieť, táto interakcia sa zaznamená. Tento proces zaznamenávania aktivity používateľov nám umožňuje monitorovať a analyzovať správanie používateľov a poskytuje cenné informácie o tom, ako aplikácia funguje v praxi. Tieto informácie môžu byť veľmi užitočné pri navrhovaní a zlepšovaní aplikácie.

Na uľahčenie procesu zaznamenávania aktivity používateľa sme implemento-

vali modul, ktorý zabezpečuje, že všetky interakcie používateľov sú zaznamenávané. Modul, ako je znázornené na obrázku 4.6, pozostáva zo štyroch funkcií: createSession, createFiringAttempt, endSession a getFiredTransitionsFromSession.



Obr. 4.6: modul zabezpečujúci zaznamenávanie používateľskej aktivity

Funkcia createSession odošle požiadavku POST do koncového bodu API /task-session/start na vytvorenie novej relácie pre úlohu a prihláseného používateľa. Ak je požiadavka úspešná, funkcia vráti ID relácie ako objekt JSON.

Funkcia createFiringAttempt odošle požiadavku POST do koncového bodu /firing-attempt na zaprotokolovanie akcie súvisiacej s reláciou so zadaným taskSessionId. Obsahuje podrobnosti o akcii, napríklad ako bola akcia vykonaná, dátum akcie, či bola akcia úspešná a informácie o zistenej akcii a cieľoch akcie. Táto funkcia nič nevracia.

Funkcia endSession odošle požiadavku POST do koncového bodu /task-session/finish na ukončenie relácie so zadaným taskSessionId. Obsahuje podrobnosti o čase ukončenia a o tom, či bola relácia úspešná. Ak je požiadavka úspešná, funkcia odstráni ID relácie z lokálneho úložného priestoru.

Funkcia getFiredTransitionsFromSession odošle požiadavku GET do koncového bodu /firing-attempt/session/taskSessionId na získanie všetkých akcií vykonaných v relácii so zadaným taskSessionId. Ak je požiadavka úspešná, funkcia vráti objekt JSON s podrobnosťami o každej akcii.

Zaznamenávanie interakcií používateľa je rozhodujúce pre zachytenie stavu scény a Petriho siete v prípade, že je činnosť používateľa prerušená, napríklad z dôvodu technických problémov. Aby sme to vyriešili, používame lokálne úložisko na uloženie identifikátora aktuálnej relácie. To umožňuje užívateľovi pokračovať v práci na úlohe tam, kde skončil, aj po prerušení.

5 Vyhodnotenie

V rámci tejto práce sme dosiahli niekoľko vylepšení aplikácie a zároveň identifikovali možnosti pre ďalší rozvoj. Nasledujúce podkapitoly zhrňujú splnené požiadavky a navrhované vylepšenia.

5.1 Zhodnotenie výsledkov práce

V tejto práci sme rozšírili funkcionality aplikácie. Nasledujúce body zhrňujú dosiahnuté vylepšenia:

- **Možnosť pridávania vlastných scénarov:** Pridali sme novú funkciu, ktorá umožňuje používateľom vytvárať vlastné scénare v aplikácii.
- **Komponent scény pre pridávanie popiskov:** Rozšírili sme možnosti scény tým, že sme pridali komponent umožňujúci priame pridávanie popiskov do scény.
- **Podpora viacerých jazykov v scéne:** Rozšírili sme podporu jazykov v aplikácii vrátane možnosti zobraziť texty a popisky v rôznych jazykoch v scéne.
- **Zaznamenávanie aktivity používateľov:** Implementovali sme funkcionalitu na zaznamenávanie aktivity používateľov v aplikácii. Týmto spôsobom môžeme sledovať a analyzovať interakcie používateľov s aplikáciou.
- **Rozšírenie existujúceho modelu o ďalšie exponáty:** Rozšírili sme existujúci model a pridali ďalšie exponáty. Tým sme rozšírili obsah aplikácie a poskytli používateľom viac možností na objavovanie nových informácií.
- **Podpora ovládania pre VR ovládače:** Pomocou výsledkov analýzy ovládania VR ovládačov vo virtuálnej scéne sme do práce zakomponovali komponent super-hands, ktorý nám po analýze prišiel najvhodnejší.

Poznatky získané z analýzy existujúcich riešení, nám pomohli potvrdiť našu úvahu že Petriho siete sa v praxi používajú na návrh scénarov a riadenie virtuálnich prostredí. Poznatky sme ďalej nepoužili pri návrhu a implementácií architektúry riešenia, tie sa však ešte môžu byť užitočné pri implementácií ďalších vylepšení.

5.2 Možné vylepšenia aplikácie

Aplikácia má veľký potenciál pre budúci rozvoj s niekoľkými inovatívnymi nápadmi. V nasledujúcom zozname sa nachádzajú možné vylepšenia aplikácie:

- **Pridanie networked Aframe:** Jednou z navrhovaných myšlienok je pridať do projektu networked Aframe, ktorý učiteľom umožní vidieť pokrok študentov a komunikovať s nimi v reálnom čase. Bola by to skvelá funkcia pre vzdelávanie na diaľku, pretože by to učiteľom umožnilo efektívnejšie monitorovať študentov a pomáhať im.
- **3D avatary pre užívateľov:** Vytvorenie plne prispôsobiteľných 3D avatarov pre študentov aj učiteľov, ktorí by reprezentovali užívateľa v scéne. Používatelia by si tak mohli prispôsobiť svoje avatary podľa svojich preferencií, vrátane farby pleti, vlasov, okuliarov a oblečenia.
- **Interaktívny formulár:** Interaktívny formulár na výber otázok zahrnutých v úlohe, správnych odpovedí a popisov namiesto importovania súborov. To by zjednodušilo proces vytvárania vlastných scenárov, ušetrilo čas pedagógom a zvýšilo dostupnosť aplikácie.
- **Prispôsobiteľné prostredia:** Používatelia by si tak mohli prispôsobiť prostredie podľa svojich preferencií, ako je napríklad výber vnútorných alebo vonkajších nastavení alebo nastavenie osvetlenia.
- **Špeciálnej učebne:** Vytvorenie špeciálnej učebne s tabuľami a možnosťami voľného pohybu, ktorí umožní učiteľom zdieľať alebo kresliť a prezentovať prednášky buď s voľným pohybom, alebo so statickou nepohyblivou polohou pre študentov. Učebňu je možné prispôsobiť tak, aby obsahovala funkcie, ako sú vonkajšie alebo vnútorné nastavenia, možnosti chatu a mikrofónu a emotikony.

6 Záver

V rámci našej práce sme pokračovali v práci Ing. Adama Kašelu a rozšírili virtuálne prostredie o nové exponáty a úlohy. Vypracovali sme návrh a implementáciu rozšírenia využitia Petriho siete, ktoré umožňuje výber siete pre prostredie a znenie úloh vo forme samostatného textového súboru, oddeleného od scény. Zároveň sme navrhli a implementovali ovládanie pomocou raycastera pre VR prilby.

V rámci našej práce sme analyzovali existujúce riešenia využívajúce Petriho siete a tiež samotnú prácu Ing Adama Kašelu. Na základe tejto analýzy sme identifikovali oblasti, v ktorých je možné vylepšiť súčasný systém. Navrhli sme a implementovali funkcionality umožňujúce zobraziť mená prechodov priradených k príslušným ovládacím prvkom v scéne. Toto je dôležité pre tvorcov scenárov, aby mohli lepšie porozumieť, na čo sa prechody siete viažu.

Ďalším dôležitým aspektom našej práce bolo zaznamenávanie aktivity užívateľa počas vykonávania úloh. Tieto záznamy nám poskytujú presný prehľad o postupe, ktorý užívateľ nasledoval pri vykonávaní úlohy. Tieto záznamy slúžia nielen na vytváranie štatistík, ale aj na obnovu scény zo záznamu, čo umožňuje užívateľom pokračovať v úlohe aj po obnovení stránky.

Naša práca prináša prínos v oblasti rozšírenia virtuálneho prostredia a vylepšenia existujúceho systému. Výsledkom našej práce je univerzálny systém ovládania scény pomocou Petriho siete, ktorý je použiteľný na akúkoľvek scénu. Taktiež sme vytvorili efektívny mechanizmus zaznamenávania aktivity užívateľa, čo nám poskytuje cenné informácie pre ďalšie spracovanie a vylepšenie systému.

Celkovo je naša práca v oblasti rozšírenia virtuálneho prostredia o ďalšie exponáty a úlohy veľmi úspešná. Naša implementácia navrhnutých funkcií bola úspešná a preukázala svoju hodnotu v praxi. Veríme, že naše vylepšenia prispievajú k lepšej interakcii užívateľov s virtuálnym prostredím a umožňujú tvorbu vlastných úloh pre koncových užívateľov

Literatúra

1. SOBOTA, Branislav; KOREČKO, Štefan; HUDÁK, Marián; SIVÝ, M. Collaborative virtual reality usage in educa-tional and training process. In: *XI International Conference of Information Technology and Development af Education, Itro 2020. Proceedings of Papers*. 2020, s. 242–247.
2. ANDREWS, Christopher; SOUTHWORTH, Michael K; SILVA, Jennifer NA; SILVA, Jonathan R. Extended reality in medical practice. *Current treatment options in cardiovascular medicine*. 2019, roč. 21, č. 4, s. 1–12.
3. CHUAH, Stephanie Hui-Wen. Why and who will adopt extended reality technology? Literature review, synthesis, and future research agenda. *Literature Review, Synthesis, and Future Research Agenda* (*December 13, 2018*). 2018.
4. CATALFAMO, Michelle A. Dynamically Generating Virtual Reality Scenes Using Molly and A-Frame. In: *Proceedings on the International Conference on Internet Computing* (*ICOMP*). 2017, s. 21–24.
5. ANGULAR. *What is Angular?* n.d. Dostupné tiež z: `https://angular.io/guide/what-is-angular`. Accessed May 12, 2023.
6. PETERSON, James L. Petri nets. *ACM Computing Surveys* (*CSUR*). 1977, roč. 9, č. 3, s. 223–252.
7. MURATA, Tadao. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*. 1989, roč. 77, č. 4, s. 541–580.
8. ARAÚJO, Manuel; ROQUE, Licínio. Modeling Games with Petri Nets. In: *Digra conference*. 2009.
9. REISIG, Wolfgang. *Petri nets: an introduction*. Springer Science & Business Media, 2012.
10. ZURAWSKI, Richard; ZHOU, MengChu. Petri nets and industrial applications: A tutorial. *IEEE Transactions on industrial electronics*. 1994, roč. 41, č. 6, s. 567–583.

11. GARCÍA-GARCÍA, Julián Alberto; ENRÍQUEZ, José Gonzalez; RUIZ, Mercedes; ARÉVALO, Carlos; JIMÉNEZ-RAMÍREZ, Andrés. Software process simulation modeling: systematic literature review. *Computer Standards & Interfaces*. 2020, roč. 70, s. 103425.

12. BROM, Cyril; ABONYI, Adam. Petri-nets for game plot. In: *Proceedings of AISB artificial intelligence and simulation behaviour convention, Bristol*. 2006, zv. 3, s. 6–13.

13. SYUFAGI, Moh; HARIADI, Mochamad; PURNOMO, Mauridhi Hery et al. Petri net model for serious games based on motivation behavior classification. *International Journal of Computer Games Technology*. 2013, roč. 2013.

14. BARRETO, Franciny M; FREITAS, Joslaine Cristina Jeske de; JULIA, Stéphane. A timed petri net model to specify scenarios of video games. In: *Information Technology-New Generations*. Springer, 2018, s. 467–473.

15. BROM, Cyril; ŠISLER, Vít; HOLAN, Tomáš. Story Manager in 'Europe 2045 'Uses Petri Nets. In: *International Conference on Virtual Storytelling*. 2007, s. 38–50.

16. MA, Bin; GUO, Zhi-ying; ZHOU, Hua-min. Development of a plastic injection molding training system using Petri nets and virtual reality. *Journal of Zhejiang University-SCIENCE A*. 2006, roč. 7, č. 3, s. 302–308.

17. LIN, Fuhua; YE, Lan; DUFFY, Vincent G; SU, Chuan-Jun. Developing virtual environments for industrial training. *Information sciences*. 2002, roč. 140, č. 1-2, s. 153–170.

18. KAŠELA, Adam. *Výučbové scenáre v rozšírenej realite využívajúce procesné grafy*. Košice, 2022. Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky. Vedúci práce: Ing. Štefan Korečko, PhD.

19. A-FRAME. *VR Controllers*. n.d. Dostupné tiež z: `https://aframe.io/docs/1.4.0/introduction/interactions-andcontrollers.html#vr-controllers`. Accessed May 11, 2023.

20. BARNARD, Dom. Degrees of Freedom (DoF): 3-DoF vs 6-DoF for VR Headset Selection. *virtualspeech*. 2019. Dostupné tiež z: `https://virtualspeech.com/blog/degrees-of-freedom-vr`.

21. C-FRAME. *aframe-super-hands-component: Compatibility*. n.d. Dostupné tiež z: `https://github.com/c-frame/aframe-super-hands-component#compatibility`. Accessed May 11, 2023.

22. DEMIANENKO, Dmytro. *Spracovanie údajov z virtuálnych výučbových prostredí*. Košice, 2023. Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky. Vedúci práce: Ing. Štefan Korečko, PhD.

Zoznam príloh

A Používateľská príručka

Používateľská príručka je vypracovaná spoločne s Dmytrom Demianenko a jeho prácou [22], kedže sme práce vykonávali súbežne a pracovali sme na rovnakej aplikácií. Taktiež naša práca nadväzuje na prácu [18] a niektoré aspekty používateľskej príručky, ktoré sa našou prácou nijak nemenili sme ponechali v pôvodnom stave.

A.1 Funkcia programu

Táto práca sa zameriava na vývoj systému analýzy údajov v rozšírenej realite (XR), ktorý uľahčuje žiakom plnenie úloh, poskytuje učiteľom dohľad nad úlohami a ponúka administratívne funkcie. Cieľom systému je zhromažďovať údaje o používateľoch počas plnenia úloh a poskytovať podrobné štatistiky o splnených úlohách. Integrácia možností analýzy údajov umožňuje získať prehľad v reálnom čase a personalizovanú spätnú väzbu pre používateľov, čo zlepšuje výsledky vzdelávania.

A.2 Inštalácia programu

V tejto kapitole sa rozoberá obsah produktu, po ktorom nasleduje špecifikácia technických a softvérových požiadaviek. Postup inštalácie produktu je opísaný z pohľadu používateľa (administrátora aj bežného používateľa).

A.2.1 Požiadavky na technické prostriedky

Na spustenie projektu musia byť splnené tieto minimálne hardvérové požiadavky pre WebGL, a to:

- 64-bitový operačný systém
- 4 GB pamäte RAM

- Integrovaná grafika s podporou WebGL 2.0 OpenGL ES 3.0
- Dvojjadrový procesor Intel/AMD
- Najnovšia verzia prehliadača Chrome, Firefox alebo Opera

A.2.2 Spustenie projektu

Táto časť opisuje kroky potrebné na otvorenie projektu vo webovom prehliadači pre bežných používateľov alebo na nasadenie projektu na server pre administrátorov.

Otvorenie projektu pre používateľov

Každý používateľ, ktorý má prístup k sieti VPN TUKE alebo k sieti Wi-Fi TUKE, môže tento projekt otvoriť v ľubovoľnom webovom prehliadači na adrese http://147.232.205.222:4200, kde sa potom môže zaregistrovať alebo prihlásiť.

Nasadenie aplikácie pre administrátorov

Ak administrátor potrebuje nasadiť projekt na server, môže toto urobiť pomocou nástroja Docker[1].

Po nainštalovaní nástroja Docker stačí na úspešné spustenie projektu len niekoľko krokov. Najprv je potrebné vytvoriť prázdny priečinok na serveri 147.232.205.222 s dvoma súbormi: docker-compose.yaml a servers.json.

Súbor docker-compose.yaml slúži na definovanie služieb, sietí a zväzkov potrebných pre nastavenie nástroja Docker. Umožňuje konfigurovať a spravovať viacero kontajnerov ako jednu aplikáciu. Tu je príklad základného súboru docker-compose.yaml:

```
1   version: '3.8'
2
3   services:
4
5   db-postgres:
6       image: "postgres:15.1"
7       container_name: lirkis-database
8       volumes:
```

[1]Docker je open-source platforma, ktorá umožňuje kontajnerizáciu aplikácií a poskytuje izolované a prenosné prostredie. Viac informácií nájdete na adrese: `https://www.docker.com/`

```
9         - lirkis-data:/var/lib/postgresql/data
10        ports:
11        - "5432:5432"
12        environment:
13        - POSTGRES_DB=postgres
14        - POSTGRES_USER=postgres
15        - POSTGRES_PASSWORD=postgres
16
17    pgadmin:
18        image: dpage/pgadmin4:7.1
19        container_name: lirkis-pgadmin
20        depends_on:
21        - db-postgres
22        ports:
23        - "3000:80"
24        environment:
25        - PGADMIN_DEFAULT_EMAIL=admin@admin.com
26        - PGADMIN_DEFAULT_PASSWORD=admin
27        volumes:
28        - ./servers.json:/pgadmin4/servers.json
29
30    spring:
31        image: aftermath1235/lirkiseduvepn-spring
32        container_name: lirkis-service
33        ports:
34        - "8080:8080"
35        links:
36        - db-postgres
37
38    angular:
39        image: aftermath1235/lirkiseduvepn-angular
40        container_name: lirkis-ui
41        ports:
42        - "4200:80"
43        links:
44        - spring
45
46    volumes:
47        lirkis-data:
```

V tomto súbore máme definované štyri služby: db-postgres, pgadmin, spring a angular.

Služba "db-postgres" používa obraz "postgres", nastavuje premenné prostredia pre predvoleného používateľa, heslo a názov databázy a pripája adresár ./data

ako zväzok na uchovávanie údajov databázy. Databáza bude prístupná na porte 5432.

Služba "spring" je prevzatá z obrazu "aftermath1235/lirkiseduvepn-spring", ktorý bol zostavený počas písania systému a vystavuje port 8080 na hostiteľovi, pričom ho mapuje na port 8080 v kontajneri. Táto služba funguje ako server.

Služba "angular" je prevzatá z obrazu "aftermath1235/lirkiseduvepn-angular", ktorý bol vytvorený počas písania systému a vystavuje port 4200 na hostiteľovi a mapuje ho na port 80 v rámci kontajnera. Táto služba funguje ako klient.

Služba "pgadmin" používa obraz "dpage/pgadmin4", nastavuje premenné prostredia pre predvolený e-mail a heslo používateľa admin a vystavuje port 3000 na hostiteľovi, pričom ho mapuje na port 80 v rámci kontajnera. Okrem toho pripája súbor ./servers.json ako zväzok, ktorý tiež musí byť nakonfigurovaný.

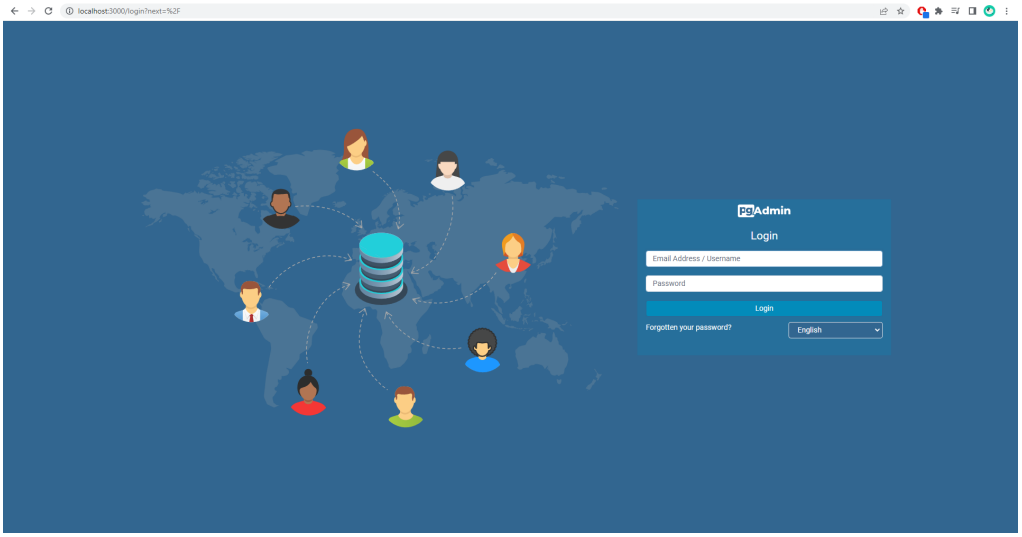
Nižšie je uvedený súbor "servers.json", ktorý používa služba "pgadmin" na konfiguráciu pripojenia k serveru PostgreSQL:

```
{
  "Servers": {
    "1": {
      "Name": "Local",
      "Group": "Servers",
      "Host": "db-postgres",
      "Port": 5432,
      "MaintenanceDB": "postgres",
      "Username": "postgres",
      "SSLMode": "prefer",
      "SSLCert": "<STORAGE_DIR>/.postgresql/postgresql.crt",
      "SSLKey": "<STORAGE_DIR>/.postgresql/postgresql.key",
      "SSLCompression": 0,
      "Timeout": 10,
      "UseSSHTunnel": 0,
      "TunnelPort": "22",
      "TunnelAuthentication": 0
    }
  }
}
```

Po vytvorení dvoch vyššie uvedených súborov stačí na spustenie programu napísať príkaz "docker compose up", ak používate systém Windows, alebo "docker-compose up", ak používate systém Linux.

Po spustení programu môžete prejsť na stránku http://147.232.205.222:4200, ktorá bude mať všetky funkcie rovnaké ako v používateľskom režime.

Veľmi dôležitým aspektom je aj pridanie správcu do systému. Keďže pri písaní práce nebolo možné zabezpečiť, aby sa používateľ s rolou administrátora pri vytváraní databázy pomocou skriptu SQL pridával automaticky, je potrebné pridať tohto používateľa ručne. Ak to chcete urobiť, musíte najprv prejsť na stránku http://147.232.205.222:3000, ktorá je zodpovedná za grafický panel databázy. Na obrázku A.1 je možné vidieť prihlasovacie okno.



Obr. A.1: Prihlasovacia obrazovka do pgAdmin

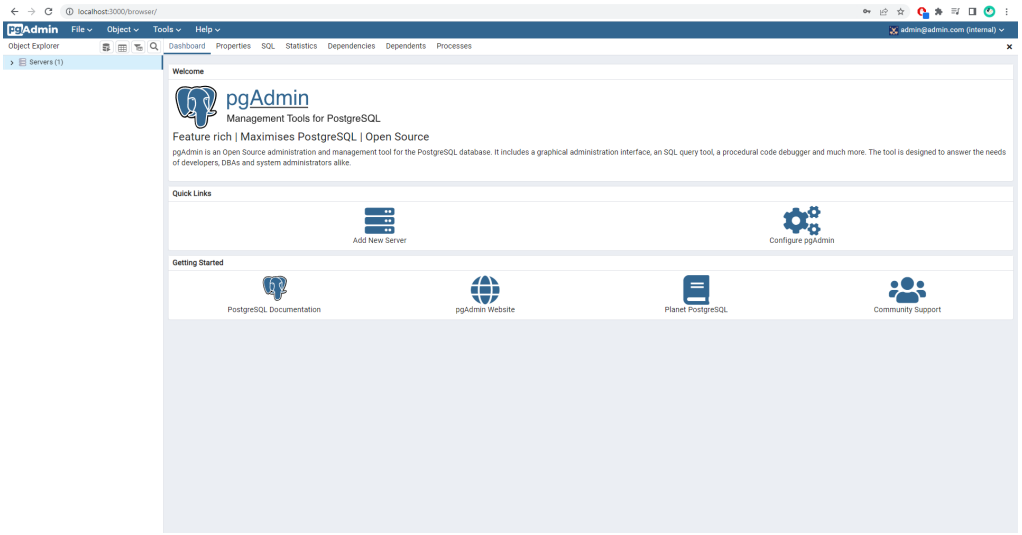
Ak sa chcete prihlásiť, musíte zadať údaje uvedené v súbore "docker-compose.yaml" v službe "pgAdmin". V našom prípade je to e-mail "admin@admin.com" a heslo "admin". Po overení sa otvorí ovládací panel. Na obrázku A.2 vľavo hore v rohu vidíte počet registrovaných serverov, v ktorých bude naša databáza. Ak sa pokúsite otvoriť našu databázu, zobrazí sa okno s požiadavkou na zadanie hesla na prístup do databázy. Heslo je "postgres". Po zadaní hesla sa otvorí naša databáza, ktorú môžete vidieť na obrázku A.3.

Po otvorení databázy kliknite na tlačidlo "Query Tool" alebo "Alt + Shift + Q", čím otvoríte editor dotazov SQL. Po otvorení editora zadajte nasledujúci dotaz, ktorý pridá nového používateľa s administrátorskými právami:

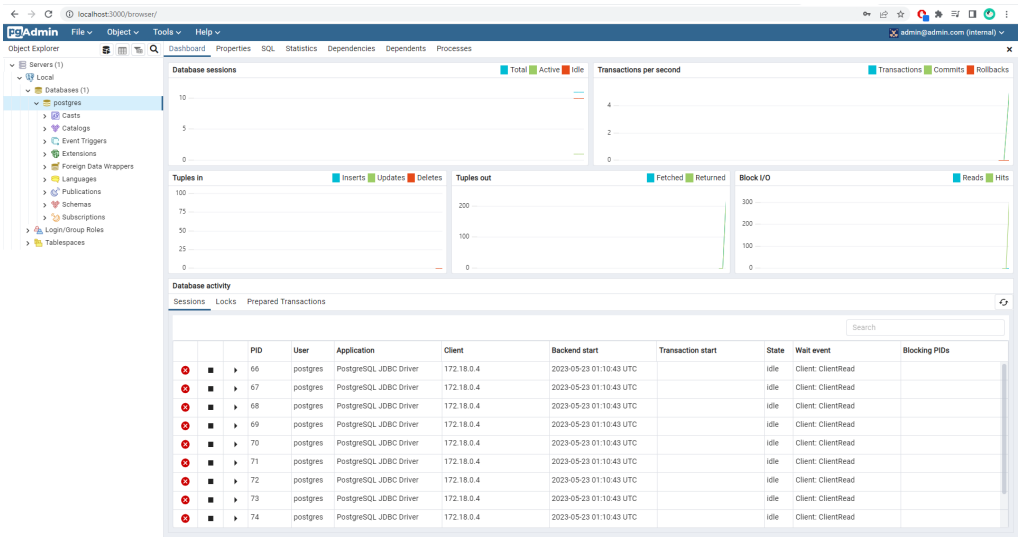
```
1    INSERT INTO users(
2      email, firstname, is_enabled,
3      lastname, username, password, role
```

```
4     )
5     VALUES (
6       'admin@admin', 'admin', true, 'admin', 'admin',
7       '$2a$12$3HMoKSwt7wbRayzLmoApieJTfvP0oqtc5ZTYmsPyo4PSJoy9TeDmW',
8       'ADMIN'
9     );
```

Po pridaní používateľa s administrátorskými právami sa môžete autentifikovať na serveri pomocou používateľského mena "admin@admin" a hesla "admin", ktoré potom môžete zmeniť. Po autentifikácii bude môcť administrátor využívať všetky funkcie systému.



Obr. A.2: Okno so servermi v pgAdmin



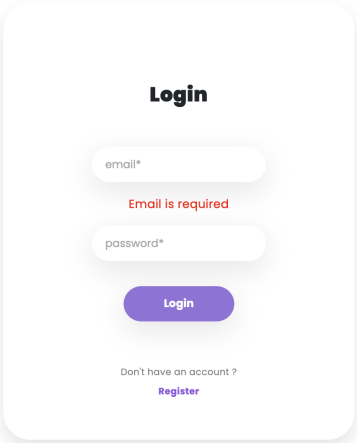
Obr. A.3: Databáza v pgAdmin

A.3 Použitie programu

Tento program ponúka užívateľom možnosť vykonávať úlohy v virtuálnom prostredí na základe predom vytvorených scénárov. Okrem toho užívateľ môže vytvárať a upravovať vlastné scénáre a má prístup k histórii svojich predchádzajúcich pokusov. V tejto kapitole nájdete podrobný návod, ako používať aplikáciu.

A.3.1 Prihlasovacia obrazovka

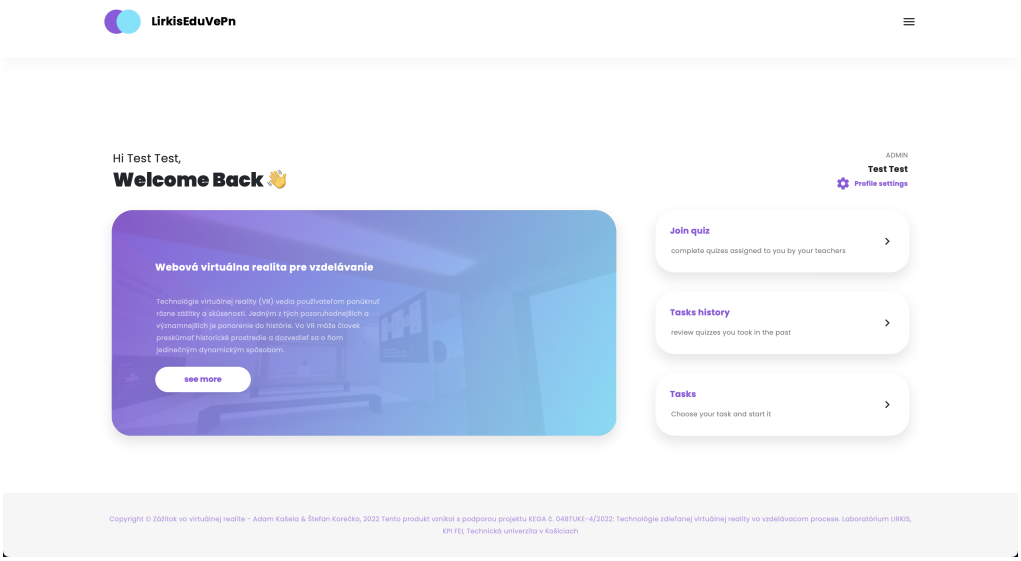
Pri spustení aplikácie sa zobrazí prihlasovacia obrazovka, kde môžete zadať svoje prihlasovacie údaje, ako je používateľské meno a heslo. Ak ste ešte nevytvorili účet, máte možnosť sa zaregistrovať prostredníctvom príslušného odkazu. Po zadaní správnych prihlasovacích údajov a kliknutí na tlačidlo **Prihlásiť sa**, budete presmerovaní do hlavného rozhrania aplikácie.



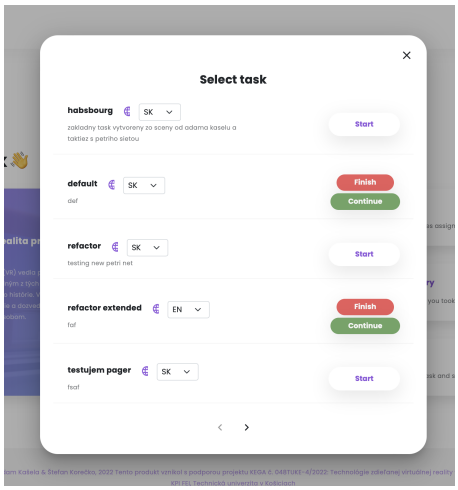
Obr. A.4: Prihlasovacie okno

A.3.2 Hlavná obrazovka

Hlavné menu pre žiakov bolo navrhnuté tak, aby poskytovalo prehľad o všetkých úlohách, ktoré im učiteľ pridelil, a zároveň im umožňovalo zobraziť si výsledky. Pri výbere niektorej z úloh má užívateľ na výber z niekoľkých jazykov, taktiež má možnosť pokračovať v úlohe ak ju už začali alebo danú úlohu ukončiť, vidno na obrázku A.6. Okrem toho obsahuje aj možnosť prístupu k nastaveniam profilu a zmeny informácií o účte. Hlavnú obrazovku je možné vidieť na obrázku A.5.



Obr. A.5: Hlavná obrazovka

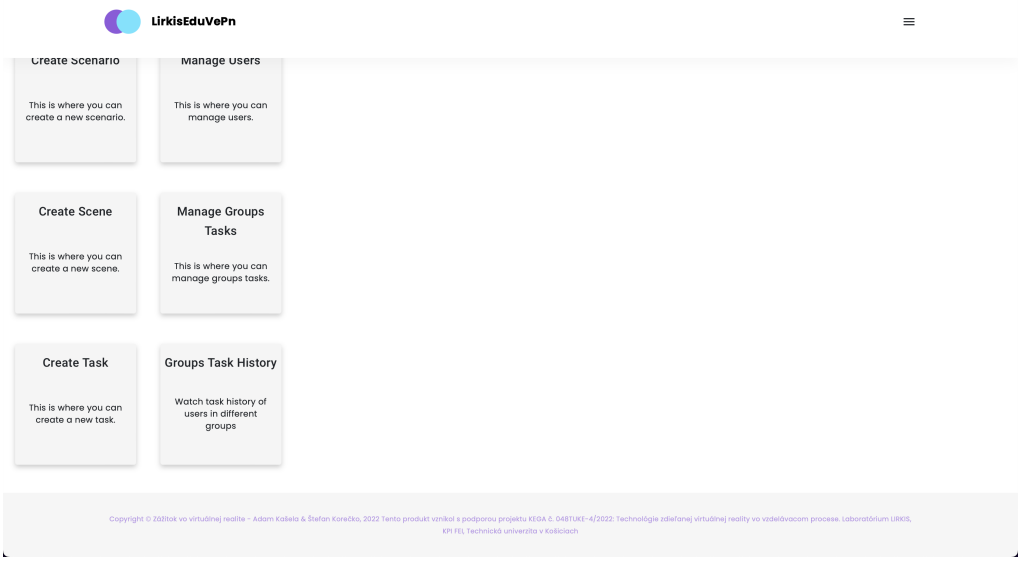


Obr. A.6: Výber úlohy

A.3.3 Dashboard pre učiteľov a adminov

Dashboard pre učiteľov a adminov je navrhnutý tak, aby im poskytoval efektívne nástroje na správu a organizáciu vzdelávacieho obsahu a používateľov. Nasledujúce sú niektoré z hlavných funkcií, ktoré tento dashboard ponúka:

- **Vytvoriť scénar** Učitelia a admini majú možnosť vytvoriť nový scénar, kde pridajú scénar v podobe Petriho siete a jazykový súbor s popiskami.
- **Vytvoriť scénu** Pri vytváraní scénarov učitelia a admini môžu vytvárať jednotlivé scény, ktoré sú už predpripravnené.
- **Vytvoriť úlohu** Učitelia a admini majú možnosť vytvoriť úlohy pre študentov, kde zadafinujú názov, popi, scénar a scénu použitú pre úlohu.

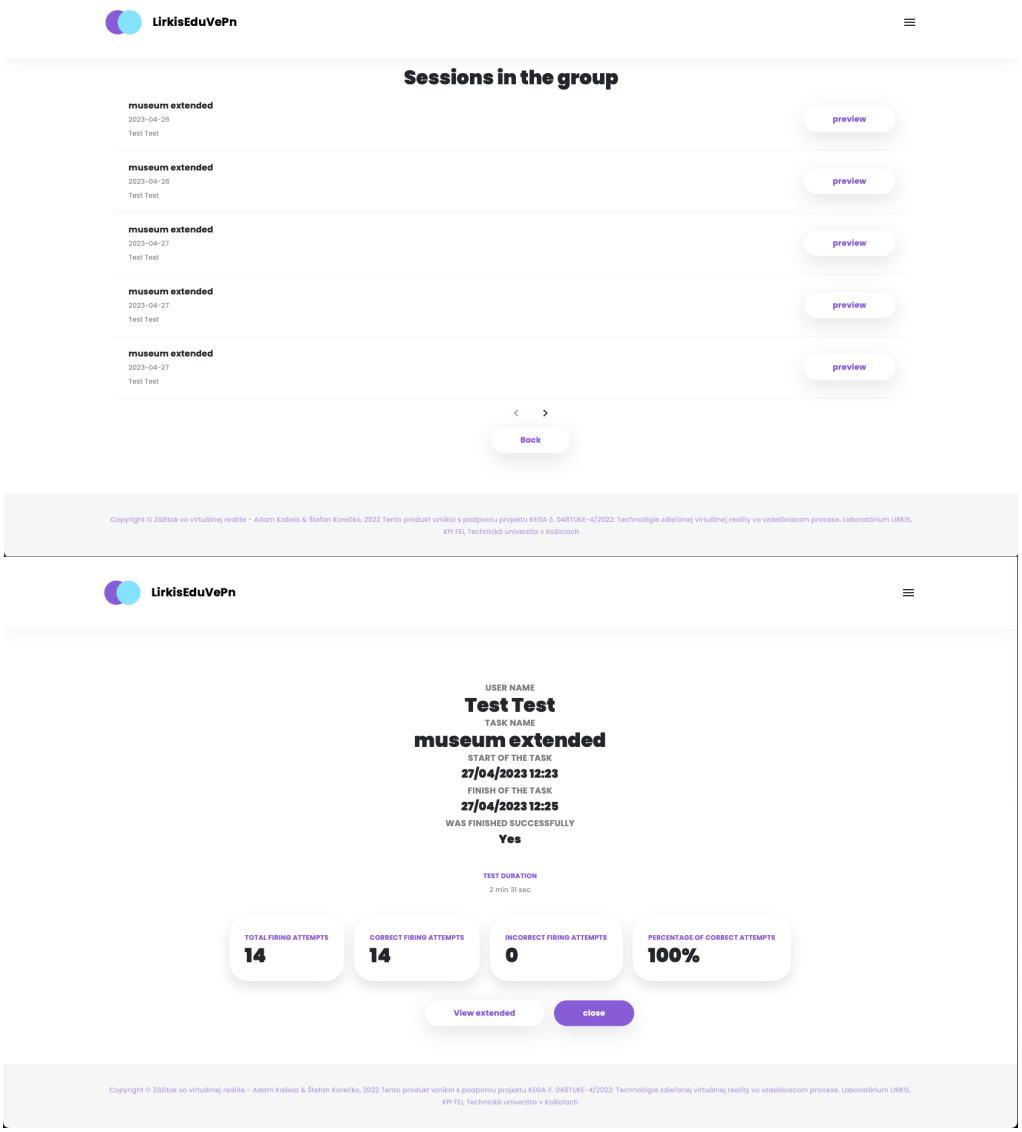


Obr. A.7: Používateľský dashobard

- **Spravovať používateľov** Pre administrátorov je dostupná možnosť spravovať používateľov systému. Môžu vytvárať, upravovať a odstraňovať používateľské účty, priradiť ich do skupín, a spravovať ich oprávnenia a prístup k obsahu.

- **Spravovať skupiny** Učitelia a admini majú možnosť vytvárať a spravovať skupiny študentov. Tieto skupiny môžu slúžiť na organizáciu študentov podľa tried, kurzu alebo iných kritérií. Učitelia môžu priradiť úlohy skupinám a sledovať ich pokrok a výsledky.

- **História úloh skupín** Tento dashboard tiež umožňuje prehľadávať históriu úloh skupín. Učitelia a admini majú prístup k predošlým výkonom a výsledkom študentov v rámci jednotlivých úloh. Môžu si pozrieť štatistiky a pokrok skupín v čase. Ako je možné vidieť na obrázku A.8, po vybratí skupiny má učiteľ k dispozícii všetky vykonané úlohy študentov v rámci skupiny. Pomocou tlačidla ***preview*** si vie zobraziť štatistiky pre danú úlohu. Následne pomocou tlačidla ***View extended*** môže zobrať odpalenia jednotlyvých prechodov spolu s informáciami či bol prechod odpálený úspešne, kedy sa odpálil a samozrejme aj názov prechodu.



Obr. A.8: Prehlad historie úloh

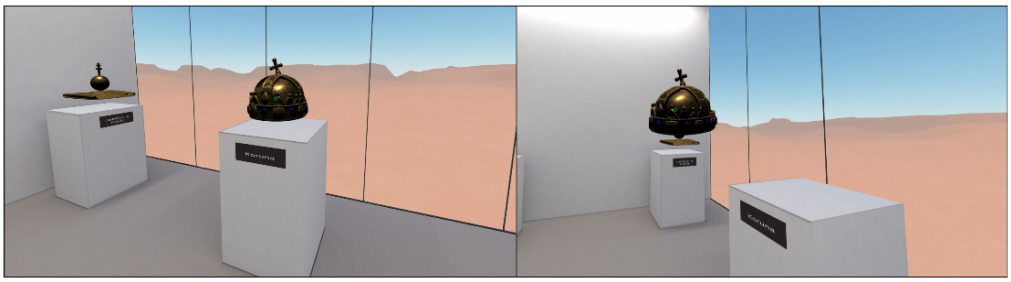
A.3.4 Virtuálna scéna

Virtuálna scéna je prostredie vytvorené v rámci programu, ktoré umožňuje interakciu a simuláciu rôznych situácií. Je to priestor, kde sa odohrávajú úlohy a aktivity pre používateľov. Scéna obsahuje rôzne typy interakcií, ktoré majú používateľom napomôcť alebo umožniť dokončiť stanovené úlohy v scéne. Používateľ môže v scéne uskutočňovať tieto interakcie:

- **Prenášanie exponátov**
- **Označovanie exponátov**
- **Zobrazovane informácií k exponátom**

A.3.5 Prenášanie exponátov

V jednej z vytvorených miestností používateľ má za úlohu prenášať exponáty na vopred určené miesto. Táto interakcia je zobrazená na obrázku A.9. Používateľ musí umiestniť zelený kurzor na objekt a následne kliknutím a držaním ľavého tlačidla počítačovej myši môže následným pohybom prenášať 3D objekt. V nasledujúcom obrázku je akcia ukázaná pred uchopením a po uchopení objektu zľava doprava.



Obr. A.9: Interakcie prenášania exponátov

A.3.6 Označovanie exponátov

Táto interakcia v scéne je znázornená na obrázku A.10. Slúži na vyberanie správnych alebo nesprávnych exponátov v jednotlivých miestnostiach s úlohami. Ľavý obrá- zok znázorňuje neoznačený exponát a pravý obrázok znázorňuje označený expo- nát. Interakcia sa vykonáva klikom počítačovej myši po umiestnení zeleného kur- zora na 3D objekt v scéne.

Obr. A.10: Interakcie označovania exponátov

A.3.7 Zobrazovane informácií k exponátom

Používateľ pri preskúmavaní virtuálnej scény môže zobrazovať informácie k jednotlivým exponátom. Tento typ interakcie je znázornený na obrázku A.11. Po umiestení kurzora na modrú ikonu pred exponátom a nasledovaním kliknutím

ľavého tla- čidla myši vie používateľ,buď zobraziť alebo skryť informácie k exponátu.

Obr. A.11: Interakcie zobrazenia informácie

A.3.8 Zvukové znamenia

Pri uskutočňovaní jednotlivých interakcií je používateľ sprevádzaný zvukovými znameniami. Jednotlivé zvuky a ich význam:

- **Úspešná odpoveď** - Zvukový signál, ktorý sa prehraje pri vykonaní po- tvrdenia na paneli úlohy v prípade keď odpoveď je správna.
- **Neúspešná odpoveď** - Zvukový signál, ktorý sa prehraje pri vykonaní potvrdenia na paneli úlohy v prípade keď odpoveď je nesprávna.
- **Bežná akcia** - Zvuky použité pri bežnej interakcii s expozíciami.
- **Úspešné vykonanie všetkých úloh** - Po dokončení všetkých úloh zaznie tento zvuk.

B Systémová príručka

B.1 Funkcia programu

Táto práca sa zameriava na vývoj systému analýzy údajov v rozšírenej realite (XR), ktorý uľahčuje žiakom plnenie úloh, poskytuje učiteľom dohľad nad úlohami a ponúka administratívne funkcie. Cieľom systému je zhromažďovať údaje o používateľoch počas plnenia úloh a poskytovať podrobné štatistiky o splnených úlohách. Integrácia možností analýzy údajov umožňuje získať prehľad v reálnom čase a personalizovanú spätnú väzbu pre používateľov, čo zlepšuje výsledky vzdelávania.

B.2 Popis programu

V tejto časti opíšeme základnú architektúru riešenia, taktiež opíšeme všetky implementované modely, moduly a componenty, opíšeme dôležité algoritmy a premenené.

B.2.1 Popis riešenia

Riešenie je založené na ECS (Entity-Component-System) architektúre. ECS umožňuje oddelenie dátových entít (entity) od ich správania (component) a systémov, ktoré tieto entity spracúvajú. ECS architektúru sme obohatili o webový rámec Angular, ktorý poskytuje robustné nástroje na tvorbu užívateľského rozhrania. Angular nám umožňuje efektívne spravovať komponenty a moduly, čo je veľmi dôležité pri rozsiahlejších projektoch.

B.2.2 Štruktúra projektu

Na obrázku B.1 je zobrazená základná štruktúra projektu, kde sú zobrazené iba najpodstatnejšie súbory, ktorými sa budeme zaoberať v najsledujúcich kapi- tolách týkajúcich sa implementácie. Medzi angulárové komponenty patria aj sa-

motné scény, pričom každá scéna obsahuje scripty, komponenty, ktoré sú špecifické iba pre danú scénu. Komponenty, ktoré sú použitelné pre viacoro scén, sú definované v priečinku js.

Obr. B.1: Súborová štruktúra projektu

B.2.3 Popis implementovaných súborov

B.2.4 Aframe komponenty

dashboard.component.js - slúži ako riadiace centrum pre používateľa v rámci aplikácie. Používatelia majú z ovládacieho centra prístup k rôznym funkciám vrátane možnosti ukončiť aktuálnu reláciu, vrátiť sa na hlavnú stránku a zobraziť informácie o používateľovi a stav jednotlivých úloch scénara.

label.component.js - slúži na zobrazovanie názvov prechodov a miest, ktoré ovplyvňujú konkrétne prvky v scéne.

petriNetSim.component.js - najdôležitejší komponent aplikácie, zodpovedný za simuláciu a riadenie scény pomocou petriho sieti.

sceneLanguage.component.js - Umožňuje zobrazovať informácie v scéne v jazyku, ktorý si užívateľ zvolí

toggleLabelVisibility.component.js - komponent zodpovedný ze zmenu viditeľnosti popiskov v scéne.

B.2.5 Angular komponenty

muzeum_extended - komponent, ktorý obshuje scénu so základným modelom múzea, slúži na obsluhu scény a poskytovanie parametrov scény od užívateľa (scénar)

muzeum_habsbourg - komponent, ktorý obshuje scénu so rozšíreným modelom múzea a doplnenými otázkamy, slúži na obsluhu scény a poskytovanie parametrov scény od užívateľa (scénar)

B.2.6 Moduly

cpnLoader.mjs - modul pre načítanie a inicializáciu Petriho siete zo súboru vo formáte cpn. Obsahuje metódy, ktorá spracujú vstup z formatu cpn:

- **loadCPNData(cpnText)** - načítanie petriho siete z parametra **cpnText**, ktorý je *string* a vracia Petriho sieť vo forme objektu.
- **xmlToJson(xml)** - vracia parsovaný objekt z dokumentu XML.

userActivityLogger.mjs - tento súbor obsahuje implementáciu skriptu pre zaznamenávanie aktivity užívateľa počas vykonávania úlohy. Obsahuje metódy:

- **createSession(taskId: number, userId: number)** - vytvára novú reláciu na základe parametou:

  - *taskId* - identifikátor úlohy
  - *userId* - identifikátor užívateľa

- **createFiringAttemt(taskSessionId: number, action: string, actionDate: Date, successful: boolean, actionFound: boolean, actionTargets: string[])** - zaznamenáva pokus o odpalenie prechodu v Petriho sieti, na základe interakcie užívateľa so scénou. Má nasledujúce parametre:
  - *taskSessionId* - identifikátor úlohy
  - *action* - názov prechodu
  - *actionDate* - čas v ktorom, bol pokus o odpálenie vykonaný
  - *successful* - značí či bol prechod odpálený úspešne
  - *actionFound* - značí či bol prechod nájdený
  - *actionTargets* - zoznam identifikátorov prvkov scény, ktoré prechod ovplyvňuje
- **endSession(taskSessionId: number, finishTime: Date, successful: boolean)** - funkcia, ktorá ukončuje reláciu užívateľa. Obsahuje nasledujúce parametre:
  - *taskSessionId* - identifikátor úlohy
  - *finishTime* - čas ukončenia úlohy
  - *successful* - značí či bola úloha ukončná úspešne
- **getFiredTransitionsFromSession(taskSessionId: number)** - funkcia, ktorá je dôležitá pri vykonávaní obsluhy zo záznamu, vracia pole všetkych prechodov, ktoré užívateľ vykonal. Obsahuje parametre:
  - *taskSessionId* - identifikátor relácie

B.2.7 Dôležité premenné

transitions - poľa prechodov definovaných Petriho sieťou, kde každý prechod je reprezentovaný objektom obsahujúcim vlastnosti, ktoré definujú správanie prechodu:

- ***transitionName: String*** - názov prechodu
- ***ifTransitionFound: function*** (***targets ,replay***) - funkcia definujúca správanie prechodu ak je nájdený

- ***ifTransitionEnabled: function*** (***targets ,replay***) - funkcia definujúca správanie prechodu ak je odpálený
- ***ifTransitionDisabled: function*** (***targets ,replay***) - funkcia definujúca správanie prechodu ak nemôže byť odpálený
- ***ifTransitionNotFound: function*** (***targets ,replay***) - funkcia definujúca správanie prechodu ak není nájdený

places - poľe miest definovaných Petriho sieťou, kde každé je reprezentované objektom obsahujúcim vlastnosti, ktoré definujú správanie miesta:

- ***placeName: String*** - názov miesta
- ***ifPlaceFound: function*** (***placeName***) - funkcia definujúca správanie miesta ak je nájdené
- ***ifPlaceFoundOnStart: function*** - funkcia definujúca správanie miesta ak je nájdené pri načítaní scény
- ***ifPlaceNotFoundOnStart: function*** - funkcia definujúca správanie miesta ak nieje nájdené pri načítaní scény

B.3 Využité technológie

Technológie, ktoré sme použili na vytvorenie aplikácie. V zozname je uvedená aj minimálna verzia technológie, ktorá je potrebná na úspešné spustenie programu:

- ***Aframe*** (***1.4.1***)
- ***Angular*** (***15.0.4***)
- ***Bootstrap*** (***5.2.3***)
- ***Java*** (***17***)
- ***PostgreSQL*** (***15.3***)
- ***Docker***

B.4 Spustenie programu

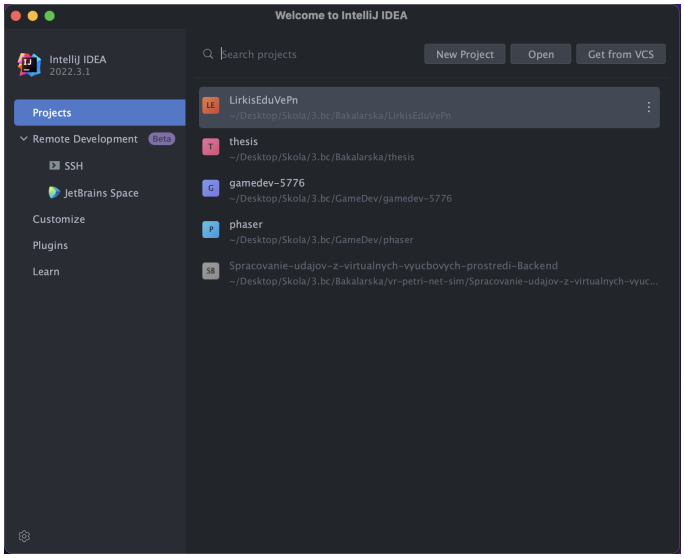
Na lokálny vývoj a spustenie aplikácie lokálne je potrebné postupovať podľa nasledujúceho postupu:

1. **Stiahnutie zdrojových kódov**

   Medzi prílohami práce sa nachádzajú aj zdrojové kódy aplikácie, je potrebné si ich stiahnúť a následne odbaliť komprimovaný balíček

2. **Otvorenie v IDE**

   Po stiahnutí potrebných súborov si projekt otvorte v ľubovolnom IDE. Odporúčame IntelliJ IDEA od spoločnosti JetBrains, ktorý aj v návode budeme používať. Po otvorení IntelliJ sa vám zobrazí úvodné okno ako je možné vidieť na obrázku B.2, následne je potrebné pomocou tlačidla *Open* vyhladať rozbalený priečinok so zdrojovými kódmi aplikácie.



Obr. B.2: Otvorenie projektu

3. **Príprava projektu na spustenie**

   Pred spustením aplikácie je najprv potrebné stiahnuť všetky potrebné závislosti. Je potrebne mať nainštalovanú minimálne Javu 17, PostgreSQL 15.3 a NodeJS (verziu 8 alebo novšiu) a NPM.

   Nastavenie databázy
   Ďalej je potrebné mať správne nakonfigurovanú databázu. Po nainštalovaní PostgreSQL vytvorte lokálnu prázdnu databázu v termináli (psql) alebo desktopovej aplikácii (pgAdmin), ktorú poskytuje PostgreSQL počas inštalácie. Následne je potrebné nastaviť vytvorenú databázu pre SpringBoot a to

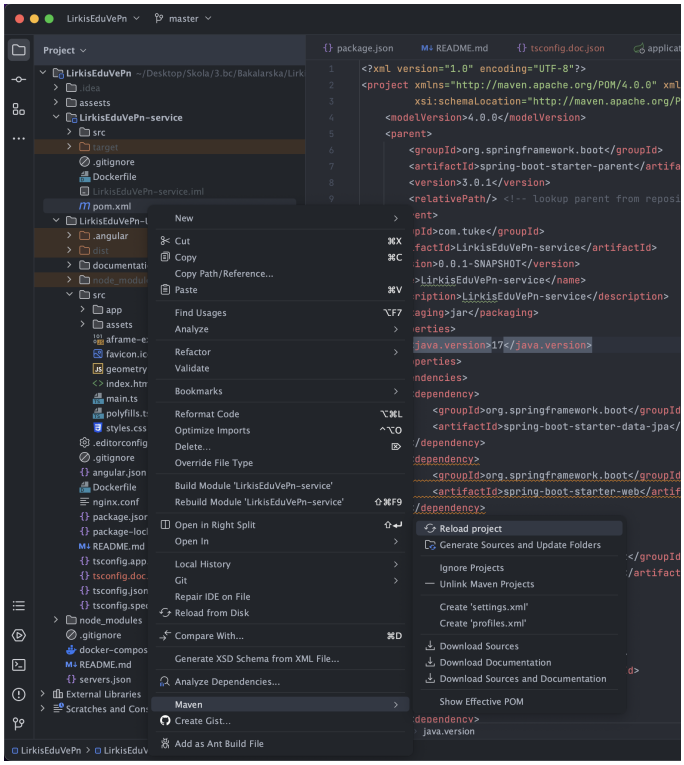
v súbore ktorý nájdete v *LirkisEduVePn-service/src/main/resources/application.properties*. V súbore *application.properties* je potrebné upraviť *url, username a password* podľa nastavení PostgreSQL. Príklad súboru je vidno na nasledujúcom kóde.

```
spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.jpa.hibernate.ddl-auto=update
jwt.secret=655368566D597133743677397924422645294840 4D635166546A576E5A723475

jwt.expiration=120
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB
```

Inštalácia závislosti

Na spustenie serverovej časti je potrebné mať stiahnuté všetky potrebné závislosti, v našom prípade používame na inštaláciu *Maven*, vďaka ktorému je inštalácia veľmi jednoduchá. Stači niekde v priečinku *LirkisEduVePn-service* stlačiť pravým tlačidlom a následne vybrať možnosť *Maven* a ***Reload project***. Výber by mal vyzerať podobne ako na obrázku B.3.



Obr. B.3: Maven závislosti

Rovnako aj na správne fungovanie klientskej časti je potrebné nainštalovať potrebné knižnice, ktoré sú definované v súbore ***package.json***. Je potrebné sa premiestniť do priečinka *LirkisEduVePn-UI* a do terminálu zadať nasledujúci príkaz:

```
1         npm install
```

4. **Spustenie projektu**

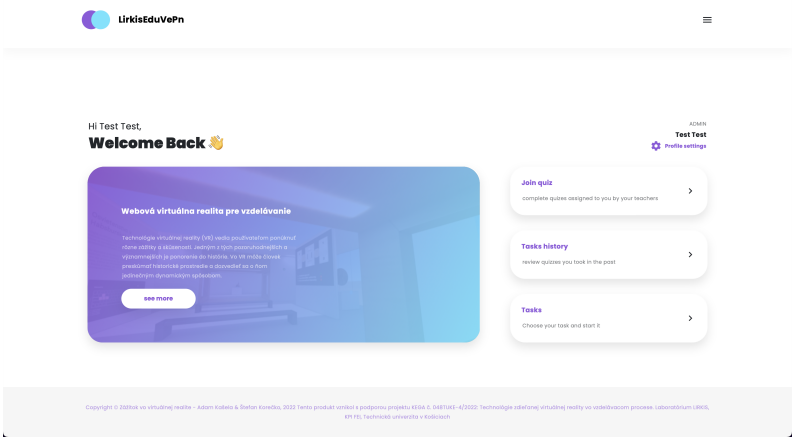
   Na fungovanie aplikácie je potrebné zvlášť spustiť klientsku a serverovú časť. Klientsku časť spustíme v termináli v priečinku *LirkisEduVePn-UI*, pomocou príkazu:

```
1         ng serve
```

   Na spustenie serverovej časti je potrebné otvoriť súbor ***LirkisEduVePnServiceApplication.java***, ktorý sa nachádza na ceste *LirkisEduVePn-service/src/main/java/com/tuke/lirkiseduvepnservice/LirkisEduVePnServiceApplication.java*. Následne je potrebne spustiť server, pomocou zelenej šípky ako je vidno na obrázku B.4.

Obr. B.4: Spustenie servera

Následne je aplikácia dostupná na adrese ***http://localhost:4200***, ktorú treba zadať do prehliadača. Následne je potrebné sa do systému prihlásiť alebo zaregistrovať ak nemáte vytvorený účet. Po úspešnom prihlásení sa zobrazí úvodná obrazovka ako na obrázku B.5.



Obr. B.5: Úvodná obrazovka aplikácie