ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Diplomová práce

Aplikace pro sběr a analýzu dat z VR tréninkových aplikací

Bc. Leoš Řeháček
Studijní program: Otevřená informatika
Specializace: Interakce člověka s počítačem

Leden 2024
Vedoucí práce: Ing. David Sedláček, Ph.D.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Řeháček** Jméno: **Leoš** Osobní číslo: **466869**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Interakce člověka s počítačem**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Aplikace pro sběr a analýzu dat z VR tréninkových aplikací

Název diplomové práce anglicky:

Application for data collection and analysis of VR training applications

Pokyny pro vypracování:

1) Seznamte se s VR (virtuální realita) tréninkovými aplikacemi vyvinutými na katedře počítačové grafiky a interakce - dodá vedoucí práce.
2) Navrhněte systém pro správu uživatelů a ukládání relevantních informací o pohybu uživatele ve VR světě. Např. sledování pozice displeje, rukou, IMU jednotky, fps, interakce s objekty, čas a dobu trvání cvičení. Uvažujte možnost "online" zaznamenávání dat, i dávkového záznamu (např. po skončení úrovně nebo až při dostupném wifi připojení).
3) Narhněte rozšiřitelné API pro propojení existujících a budoucích VR aplikací se systémem. Poskytněte důkladnou programátorskou dokumentaci pro API včetně příkladů použití.
4) Implementujte systém a upravte minimálně dvě existující VR aplikace aby využívaly vámi vytvořený systém. Výběr aplikací konzultujte s vedoucím práce.
5) Při návrhu a implementaci postupujte dle metodiky UCD (user center design) - cílovou skupinou jsou vědci, doktoři a terapeuti.

Seznam doporučené literatury:

1] Jason Jerald. 2015. The VR Book: Human-Centered Design for Virtual Reality. Association for Computing Machinery and Morgan & Claypool, New York, NY, USA.
2] T. Lowdermilk, User-Centered Design, O'Reilly Media, 2013

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. David Sedláček, Ph.D. katedra počítačové grafiky a interakce FEL

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

Datum zadání diplomové práce: **17.02.2023** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **22.09.2024**

Ing. David Sedláček, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. _____ Datum převzetí zadání

_____ Podpis studenta

Poděkování / Prohlášení

Rád bych tímto poděkoval Ing. Davidu Sedláčkovi, Ph.D. za odborné vedení, čas a vytrvalou pomoc, kterou mi v průběhu zpracovávání diplomové práce věnoval.

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

V Praze dne 8. 1. 2024

.....

Abstrakt / Abstract

Tato diplomová práce se zabývá návrhem a implementací systému určeného pro sběr a analýzu dat z VR tréninkových aplikací. Hlavním úkolem je seznámit se s již existujícími VR tréninkovými aplikacemi a na základě jejich požadavků navrhnout a implementovat nový systém, který bude sloužit pro ukládání a následnou analýzu relevantních dat. Systém byl navržen jako webová aplikace s vlastní serverovou aplikací, umožňující správu uživatelů, kategorizací dat a následnou vizualizací pomocí grafů. Aplikace umožňuje ukládání dat z VR tréninkových aplikací pomocí API za využití implementovaného Unity balíčku, který ukládá data v nově navrženém formátu a podporuje budoucí rozšiřitelnost systému a napojení nových VR tréninkových aplikací. Nakonec proběhlo testování systému, kde participanti testovali jak uživatelské rozhraní webové aplikace, tak napojení aplikace pomocí Unity balíčku na již existující VR tréninkové aplikace. Výsledkem testování je, že aplikace je uživatelsky přívětivá a aplikaci se podařilo jednoduše napojit na již existující VR tréninkové aplikace.

Klíčová slova: virtuální realita; sběr dat; analýza dat; uživatelské testování; VR tréninková aplikace; návrh aplikace; implementace aplikace;

This thesis is concerned with the design and implementation of a system to collect and analyze data from VR training applications. The main task is to explore the existing VR training applications and on the basis of their requirements to design and implement a new system that will be used for storing and subsequent analysis of relevant data. The system was designed as a web application with a custom server application, allowing user management, data categorization and subsequent visualization using graphs. The application allows storing data from VR training applications via an API using the implemented Unity package, which stores data in a newly designed format and supports future extensibility of the system and connection of new VR training applications. Finally, testing of the system took place, where participants tested the user interface of the web application as well as the connection of the application using the Unity package to existing VR training applications. As a result of the testing, the application is user-friendly and the application was easily connected to existing VR training applications.

Keywords: virtual reality; data collection; data analysis; user testing; VR training application; application design; application implementation

Title translation: Application for data collection and analysis of VR training applications

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

1.1 Motivace a cíl

VR tréninkové aplikace se stávají stále populárnějším nástrojem pro zlepšování schopností a znalostí v různých oborech lidského života, a to od průmyslu přes zdravotnictví až po školství.[1] Prostřednictvím těchto aplikací lze vytvořit prostředí, v němž si uživatelé mohou testovat a procvičovat své již stávající dovednosti, dokonce pak získávat dovednosti zcela nové, a to skrz interaktivní cvičení a úkoly. VR aplikace tak mají velmi široký potenciál využití, přičemž jako příklad lze uvést zlepšování pracovních zdatností, pomoc při rehabilitačních programech, či ověření stanovených tezí při výzkumech.

S rostoucí popularitou VR aplikací však přichází také potřeba důsledněji shromažďovat data z těchto aplikací a následně je podrobit analýze. Získaná data a jejich rozbor mohou být použita pro vyhodnocení efektivity samotné aplikace, přizpůsobení jejího obsahu konkrétním potřebám uživatelů, ba dokonce v případě lékařského zaměření pro vyhodnocení pokroku léčby pacienta.

A právě tato skutečnost je motivací pro určení cíle této práce, kterým je navržení a implementace systému pro sběr a analýzu dat z VR tréninkových aplikací. Uvedený systém by následně mohl představovat elementární nástroj schopný zjednodušeně uskutečňovat analýzu získaných dat, jež by mohli využívat nejenom například výzkumníci a lékaři, ale též další v úvahu přicházející subjekty, přičemž všechny tyto osoby by na základě daného mohly bez dalších úkonů snadněji dospívat k závěrům ohledně svých hypotéz, tedy k jejich potvrzení či vyvrácení.

1.2 Struktura práce

První kapitola obsahuje stručné shrnutí důvodů, které v souhrnu představují motivaci, na základě níž byly stanoveny cíle této práce.

Druhá kapitola představuje již existující VR tréninkové aplikace, které by mohly sloužit jako podklad pro zjištění funkčnosti navrhovaného systému a které by posléze mohly navrhovaný systém samy využívat. Také jsou v této kapitole uvedeny požadavky, které by v ideálním případě měl navrhovaný systém splňovat. Naproti tomu jsou také zmíněna omezení, která může navrhovaný systém obsahovat. Dále je zde představeno, jaké typy dat jednotlivá VR zařízení poskytují a shromažďují. V poslední řadě jsou v této kapitole zmíněny již existující formáty datových záznamů a existující řešení zabývající se stejnou či podobnou problematikou, jež je shromažďování dat a jejich následná vizualizace a statistika pro analýzu.

Ve třetí kapitole je obsažen individuální návrh nového řešení. Je zde nastíněna a popsána základní architektura aplikace, která definuje doporučené technologie a návrh databázového modelu aplikace. Také v této kapitole jsou představeny základní návrhy obrazovek klientské části aplikace. V poslední řadě je zde nadefinována podoba formátu záznamů a následná kategorizace záznamových dat. Na základě těchto poznatků jsou

v závěru kapitoly zmíněny vhodné statistiky a vizualizace, které můžeme pro záznamy dat použít.

Čtvrtá kapitola popisuje samotnou implementaci systému, která vychází z analýzy a individuálního návrhu nového řešení. Jsou zde popsány následující: specifika a unikátnosti řešení jednotlivých částí, jež jsou klientská a serverová aplikace, doplňkový Unity balíček a ukázkový projekt s WebGL modulem. Také jsou zde představeny a vysvětleny zvolené vizualizace záznamových dat a finální vzhled aplikace.

V páté kapitole je popsáno uživatelské testování implementovaného systému spolu s testovacími scénáři, představením participantů a závěrem. Dále tato kapitola popisuje zkušební propojení systému s již existujícími VR tréninkovými aplikacemi, jež je zakončeno závěrem a popisem následných úprav aplikace, které vycházejí z tohoto zkušebního propojení.

Poslední částí a pomyslnou poslední kapitolou je závěr této práce, v němž je zhodnoceno, zda se podařilo či nepodařilo naplnit cíl stanovený v kapitole 1.1, tj. zda byl navržen a implementován systém pro sběr a analýzu dat z VR tréninkových aplikací tak, aby odpovídal vzneseným požadavkům. Na závěr jsou zmíněny vhodné změny a úpravy, které by bylo dobré v budoucí verzi systému zohlednit.

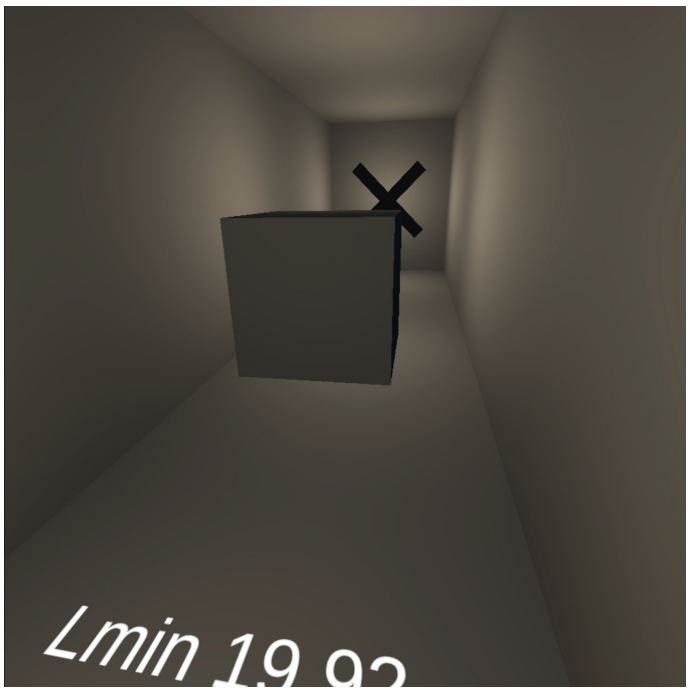
Kapitola 2
Analýza

2.1 Existující VR tréninkové aplikace

Tato kapitola se zabývá představením jednotlivých již existujících VR tréninkových aplikací, jejichž vztah s navrhovaným systémem lze rozčlenit do dvou rovin. Za prvé tyto aplikace mohou figurovat jako základní jednotky sloužící pro to, aby se prostřednictvím nich definovaly požadavky navrhovaného systému. Lze tak říct, že navrhovaný systém v tomto směru využije dané stávající aplikace. Na druhé straně pak mohou, a to zejména do budoucna, sloužit jako aplikace testovací. Data z níže uvedených aplikací by teoreticky v pilotní verzi prošla navrhovaným systémem, což samozřejmě opětovně pomůže přímo zjištění funkčnosti tohoto systému. Avšak nelze opomenout, že v tuto chvíli též uživatelé aplikací využijí navrhovaný systém, jelikož přímo tyto osoby následně získají informace o datech, která jsou v aplikacích generována a jež byla v systému zanalyzována.

2.1.1 Luna

Aplikace Luna testuje, zda-li se osobám po uskutečněné operaci očí zlepšuje zrak, anebo nikoliv. Participanti si nastavují intenzitu světla, při které vidí místnost s jednotlivými objekty. Vše je v barvě šedi. Pacient prochází místností tam a zpět, během čehož se musí vyhýbat ukotveným objektům, případně jiným překážkám. Pro participanta se prakticky jedná o průchod bludištěm, v rámci kterého má pouze headset bez ovladačů.



Obrázek 2.1. Ukázka z VR aplikace Luna.

Uživatelé a participanti

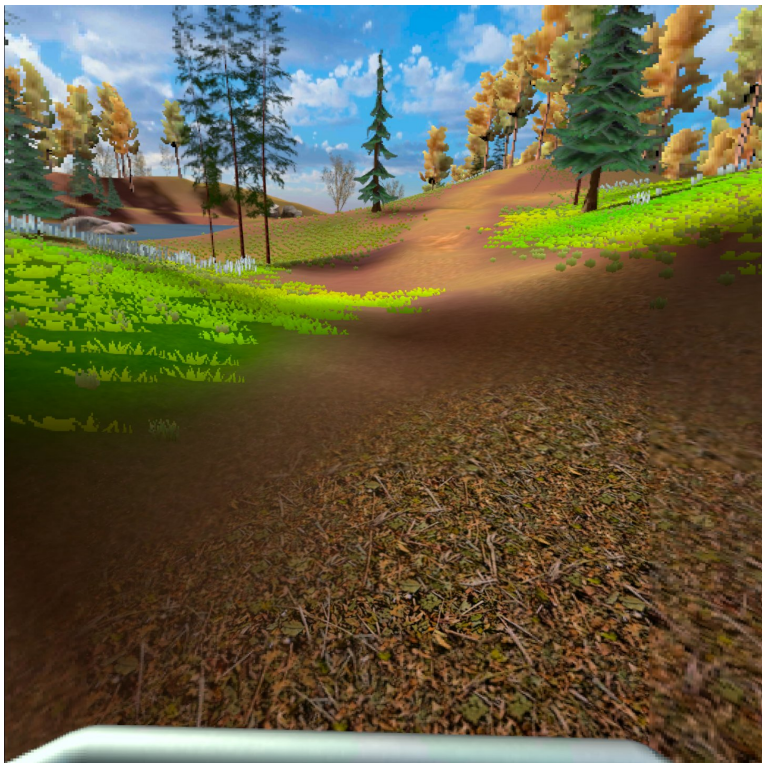
Uživatel této aplikace je výzkumník či lékař, který prostřednictvím ní zkoumá účinky léčby svých pacientů. Participant této aplikace je pacient, který trpí poruchou zraku a nevidí barevně jako běžný člověk, avšak vidí pouze v různých odstínech šedi.

Očekávaná data a analýzy dat

Uživatelé očekávají, že po provedených operacích participanti vidí lépe. To znamená, že při opakovaném průchodu aplikací by si pacient měl přirozeně nastavovat nižší hranici intenzity světla a dělat méně chyb spočívajících v narážení do jednotlivých objektů či překážek. Data z provedených zkoušek se zanesou do systému, v rámci nějž by bylo přínosné, aby se jednotlivé záznamy daly mezi sebou porovnávat, a to např. co do uplynulého času průchodu aplikací nebo co do počtu nárazů do objektů či překážek.

2.1.2 Physio Trails

Jedná se o aplikaci, která by měla pomoci seniorům získat jistotu v rovnováze, pakliže mají potíže s běžným pohybem ve smyslu chůze, ba dokonce dovednost chůze zcela ztratili. V aplikaci jsou 2 různé úrovně. Cílem pacienta je držet hlavu v jedné poloze a zároveň se při tom nepřidržovat madel, které má k dispozici před sebou, a to jak v realitě, tak v té virtuální, v níž by měla být madla synchronizována s těmi skutečnými. Pacient používá pouze headset bez ovladačů.



Obrázek 2.2. Ukázka z VR aplikace Physio trails.

Uživatelé a participanti

Uživatelé aplikace jsou pečovatelé a lékaři, kteří se dostávají do kontaktu se seniory, kteří mají specifické potíže. Participanti aplikace jsou senioři, kteří se snaží získat stabilitu při stání či chůzi, anebo lidé, kteří se například kvůli úrazu musí naučit znovu chodit, a to od úplného začátku. V tomto směru pak participantem nemusí být toliko senior, avšak i mladý člověk.

Očekávaná data a analýzy dat

Uživatelé by chtěli zaznamenávat nejenom to, kdy si jaký participant aplikaci vůbec spustil, ale zejména jeho výsledek, tj. jak byl pacient úspěšný. Pro uživatele by bylo užitečné, jestli se participant postupem času ve svém výkonu zlepšuje. Tato vyhodnocení se očekávají z analýzy pohybu hlavy participanta, resp. z analýzy toho, jak pacient hlavu naklání. Uživatelé by též chtěli být informováni o tom, jakou úroveň a rychlost si participant volí.

2.1.3 Rozpoznání chodeb

Jedná se o aplikaci, v níž má člověk určit, zda-li se jedná o chodbu, nebo o tvar, který chodbu jenom připomíná. Prakticky participant rozlišuje, zda má před sebou skutečnou chodbu, anebo toliko optický klam. V aplikaci je několik úrovní, které se liší určitými parametry jako je například délka chodby. Pro každou úroveň jsou nastaveny 2 obtížnosti.

Uživatelé a participanti

Uživatelé této aplikace jsou výzkumníci, kteří se tímto specifickým problémem zabývají. Participanti jsou předem vybraní lidé bez jakýchkoliv upřesnění.

Očekávaná data a analýzy dat

Uživatelé chtějí porovnávat, jak dlouho participantovi trvalo, než se dokázal rozhodnout, jestli se opravdu jedná o chodbu nebo o její napodobeninu, a případně jestli bylo učiněné rozhodnutí správné. Pro tyto účely by měla sloužit data obsahující to, jak dlouho se participant v dané úrovni rozmýšlí, jak daleko do stran směřuje a jestli při tom otáčí hlavou. Nadto se jednotlivé úrovně opakují, a tudíž uživatel očekává, že bude moct dané úrovně a rozhodnutí porovnávat.

2.2 Požadavky

Před samotným návrhem a vývojem systému je potřeba si definovat, co všechno má takový systém (aplikace) umět a jaká jsou jeho specifika. Níže jsou uvedeny základní požadavky, které by měl systém splňovat. Jedná se o podmínky, které vycházejí z nároků jednotlivých VR aplikací či jejich uživatelů.

Funkční požadavky

- Registrace a přihlášení uživatelů
- Existence kartotéky participantů
- Možnost uživatele zaregistrovat novou VR aplikaci
- Možnost uživatele zaregistrovat novou organizaci
- Možnost uživatele přizvat jiného uživatele do své organizace
- Omezení vhledu uživatele pouze na své VR aplikace a participanty
- Sběr jednotlivých aktivit participantů, které vykonávají v určitých VR aplikacích
- Možnost uživatele zobrazit si jednotlivé aktivity svých participantů
- Možnost uživatele upravit konkrétní aktivitu

Nefunkční požadavky

- Ideálně rozhraní webové aplikace
- Možnost jednoduše rozšířit aplikaci o novou VR aplikaci
- Rozhraní vhodné pro výzkumníky a zdravotníky

2.2.1 Specifika VR aplikací

Jelikož navrhovaný systém má za úkol zpracovávat data z VR aplikací, přináší to do vývoje a následné analýzy takového typu aplikace určité individualizace, na které je třeba brát ohled.

VR aplikace jsou od běžných aplikací odlišné primárně tím, že pracují v 3D prostoru. To znamená, že se uživatel VR aplikace pohybuje jak v reálném světě, tak v tom virtuálním. Každá VR aplikace proto dokáže poskytnout 3D pozici jejího uživatele v reálném světě a 3D pozici headsetu a ovladačů jejího uživatele ve virtuálním světě. Na základě těchto dat se dají provádět různé analýzy, a to včetně rekonstrukce pohybu uživatele ve VR aplikaci.

2.2.2 Omezení VR aplikací

S přihlédnutím k tomu, že každá VR aplikace má jiné zaměření, s čímž souvisí to, že může používat různá zařízení (např. headset, ovladače), očekává se, že v důsledku toho budou vycházet také odlišná data a jejich analýzy. To znamená, že pro každou VR aplikaci bude muset její autor vytvořit specifický modul, který tuto analýzu dat zpracuje. Vzhledem k tomu bude navrhovaný systém provádět pouze obecné analýzy, které jsou pro všechny VR aplikace společné, přičemž pro specifické úkony bude používat předpřipravené moduly od tvůrců VR aplikací.

2.3 Typy dat shromažďované VR zařízeními

Tato sekce se zabývá tím, jaké typy dat mohou VR zařízení o svém uživateli sbírat.[2] Díky tomu je možné získat širší přehled o tom, jaká data mohou být v rámci VR aplikací dostupná. To může pomoci definovat, která data by bylo vhodné sledovat a ukládat v rámci navrhovaného systému.

VR zařízení se spoléhají i na získávání informací z více zdrojů (třetích stran), aby poskytla optimální uživatelský zážitek. Obecně se shromažďované informace dají rozdělit do čtyř typů dat:

- pozorovatelné - informace, které mohou VR zařízení pozorovat. Jedná se například o digitální komunikaci.
- pozorované - informace, které uživatel poskytuje nebo vytváří a které mohou třetí strany pozorovat, ale nemohou je replikovat. Jedná se například o údaje o poloze.
- vypočítané - nové informace, které VR odvozují z pozorovaných a pozorovatelných informací, jako je například biometrická identifikace.
- přidružené - informace, které neposkytují popisné informace o daném uživateli, jako je například IP adresa.

V následující tabulce 2.1 jsou u každého typu dat popsány příklady ve VR a jejich užitečnost ve VR. Tato tabulka je vytvořena na základě Tabulky č.1 z článku *Balancing user privacy and innovation in Augmented and Virtual Reality* [2].

Typ dat	Příklady ve VR	Použitelnost ve VR
Pozorovatelný	Virtuální osoby (avatar), digitální komunikace, interakce v aplikacích, identifikace aktivit (snímky obrazovky, nahrávky)	Vytváří virtuální přítomnost, která je pro uživatele jedinečná, a umožňuje mu interakci s virtuálními prostory a objekty.
Pozorovaný	Polohová a prostorová data, sledování pohybu/rukou/očí, záznamy o aktivitách, údaje o chování uživatele	Vytváří a zlepšuje zážitek. Umisťuje uživatele do virtuálního prostoru. Umožňuje interakci s objekty.
Vypočítaný	Uživatelský profil, biometrická identifikace uživatele	Zlepšuje služby a umožňuje pokročilé funkce.
Přidružený	Přihlašovací údaje, kontaktní informace, IP zařízení, seznam přátel	Umožňuje přiřazení obsahu k uživateli, identifikaci zařízení, přístup k internetu.

Tabulka 2.1. Typy dat ve VR.

Pozorovatelný typ dat

Mezi pozorovatelná data patří například avatar uživatele. Avatar představuje virtuální reprezentaci uživatele. Na základě avatara můžeme odhadnout některé informace, jako je například fyzická podoba, rasa či pohlaví uživatele. Jako další do tohoto typu dat patří sociální interakce jako například komunikace s ostatními uživateli, snímky obrazovky a nahraná videa.

Pozorovaný typ dat

Do tohoto typu spadá značné množství dat vytvářené ve VR. Je to kvůli závislosti VR na různých senzorech jako například inerciální měřící jednotka, gyroskop, akcelerometr a kamery. Díky těmto senzorům VR aplikace dokáže uživatele umístit do virtuálního prostoru a upozornit ho, kdyby se například přibližoval k nějaké překážce, jako je stěna ve fyzickém prostoru.

Kromě shromažďování informací o poloze VR zařízení také sbírají pohyby a biometrické údaje, aby dokázaly replikovat tyto akce ve virtuálním prostoru. Tato forma je velice důležitá, jelikož uživatel je plně ponořen do virtuálního prostředí a VR zařízení musí stoprocentně rekonstruovat fyzický zážitek, aby uživatelé měli pocit, že je jako skutečný. Díky tomu můžeme mít pocit, že stojíme na zemi, dotýkáme se předmětů a další.

Kromě sledování pohybů a polohy jednotlivých částí těla dokážou VR zařízení sledovat i pohyby očí a mimické pohyby tváří. Díky tomu mohou tyto pohyby přenést na virtuálního avatara, což zdokonaluje komunikaci s ostatními uživateli ve VR aplikacích. Uživatelé tak mohou odhadnout pocity ostatních uživatelů. Všechna tato data slouží k tomu, aby VR aplikace a zařízení mohly přinést plně pohlcující zážitek.

Vypočítaný typ dat

Na rozdíl od pozorovatelného a pozorovaného typu vypočítaný typ není poskytovaný přímo uživatelem. Tato data jsou vytvořena manipulací pozorovatelných a pozorovaných dat. Díky těmto datům mohou VR aplikace odvozovat více informací o uživateli a přinášet mu lepší zážitek či doporučení. Může se jednat například o personalizovanou reklamu či doporučení na další aktivity. Zařízení mohou vypočítat i zdravotní data jako je počet spálených kalorií.

VR zařízení mohou pro generování vypočítaných dat používat i senzory. Díky tomu mohou například ze sledování ruky odhadnout její fyzickou velikost, tvar a polohu ruky a prstů.

Přidružený typ dat

Mezi přidružený typ patří data, která uživatel poskytl v rámci VR aplikaci. Jedná se například o přihlašovací údaje, kontaktní informace, platební informace a další. Dále sem může patřit seznam přátel, se kterými se uživatel seznámil, a díky tomu nabízet uživateli podobné aktivity jako jeho přátelům.

Také do tohoto typu patří informace o samotném VR zařízení jako je sériové číslo, dostupné senzory, název připojené Wi-Fi sítě, IP adresa a další.

2.4 Způsoby ukládání záznamových dat

V této sekci je popsáno, jakým způsobem je možné ukládat záznamová data z různých typů aplikací. Tento průzkum by měl pomoci s navržením formátu pro ukládání záznamových dat. Můžeme si totiž z těchto příkladů vzít inspiraci již používaných technik.

2.4.1 Ukládání záznamových dat z VR aplikací

Z průzkumu již existujících řešení, které měly za úkol zaznamenávat VR data a následně je porovnávat, bylo zjištěno, že existují různé formáty záznamu těchto VR dat. Jedno hledisko byl samotný formát, ve kterém jsou data uložena. Druhým hlediskem bylo, jaké jednotlivé vlastnosti a data se do těchto záznamů ukládají.

Co se týká formátu ve kterém jsou data uložena, při průzkumu se lze setkat například se soubory ve formátu `txt`, `csv` či vlastním běžně neznámým formátem.

Z hlediska obsahu záznamových dat například v řešení uvedeném v článku *Motion and Meaning: Sample-Level Nonlinear Analyses of Virtual Reality Tracking Data*[3] v rámci aplikace zaznamenávali pozici (osa x, y a z vůči globálním souřadnicím) a rotaci hlavy a rukou. Kromě toho si zapisovali čas pořízení, identifikátor snímku a seznam událostí, které se staly (například zmáčknutí tlačítka). Také si zaznamenávali vzorkovací frekvenci.

V jiném řešení uvedeném v článku *Human Movement Direction Classification using Virtual Reality and Eye Tracking*[4] si kromě předešle zmíněných vlastností navíc zaznamenávali informace o participantovi. Jednalo se o identifikátor, věk, pohlaví a jazyk participanta.

Co se týče obsahu jednotlivých záznamů, tak během průzkumu nebyl nalezen žádný formát, který by byl natolik obecný, aby podporoval více VR aplikací s různými potřebami. Z tohoto důvodu bude nutné navrhnout nový formát, který bude podporovat různé požadavky jednotlivých aplikací a navíc bude rozšiřitelný.

2.4.2 Ukládání záznamových dat z her

Ukládání záznamových dat ve hrách je v některých aspektech specifické. Hry jsou většinou komplexní, kdy každý uživatel může procházet určitou úrovní jiným způsobem.

Ve hrách se většinou ukládají záznamy, pokud dojde k chybě. V tomto případě záznam neobsahuje pouze jaká chyba nastala, ale je rozšířen o další informace. Jedná se například o operační systém a specifikaci počítače, jaký je aktuální stav okolí a postavy ve hře.

Aby bylo poté možné zreplikovat danou chybu, běžně se ukládají i důležité události a milníky, podle kterých se dá určit, co v jakou chvíli hráč prováděl.

Záznamy z her jsou ve většině případů uloženy lokálně na počítači ve formě textových souborů. Jen v některých případech mohou vývojáři požádat o sdílení logů, pokud dojde například k nějaké kritické chybě.

2.4.3 Ukládání záznamových dat z webových aplikací

Většina webových aplikací pro ukládání a zobrazení záznamových dat používá cloudové řešení. Jedná se buď o řešení dané platformy, na které je aplikace nasazena nebo o využití řešení třetích stran. Takové platformy se nazývají `Log Management Solution`.[5]

Mezi řešení dané platformy patří například Amazon CloudWatch, Google Cloud Logging a Azure Monitor. Mezi na platformě nezávislé řešení patří například Datadog a Better Stack. Výhoda nezávislých řešení je v tom, že do těchto řešení je možné ukládat záznamy, i když aplikace běží na vlastním serveru nebo kdekoliv jinde.

Všechna tato řešení jsou přizpůsobena pro ukládání záznamů v textové podobě. Nad danými záznamy pak lze provádět hledání či vytvářet vizualizace. Proto musí ale vývojář definovat určitou podobu textových záznamů, ze které pak dokáže poznat, jaká část textu odpovídá jaké vlastnosti.

Kromě běžných záznamů dat, které vývojáři implementovali, webové aplikace také většinou zaznamenávají data v `Common Log Format (CLF)` formátu. [6] Tento formát pro každý HTTP požadavek obsahuje důležité informace o požadavku jako je IP adresa zařízení, protokol, data požadavku, čas požadavku atd.

2.5 Existující řešení

V této sekci jsou popsána nalezená již existující řešení, která se zabývají problematikou ukládání záznamových dat nebo zobrazením statistik či vizualizací ze záznamových dat.

U každého řešení jsou popsány jeho výhody a nevýhody. Bohužel žádné z řešení plně neodpovídá našim požadavkům. Nicméně je dobré si z těchto výhod a nevýhod vzít inspiraci pro vytvoření vlastního řešení.

2.5.1 ManySense VR

ManySense VR je framework vyvinutý v Unity, který slouží pro sběr a ukládání kontextových dat.[7] Framework byl vytvořen jako kolekce Unity skript pro jednoduché použití vývojáři. Jeho implementace se opírá o využití rozhraní API jednotlivých zařízení, které slouží pro sledování participanta.

Framework dokáže sledovat různé typy senzorů, které dokážou pozorovat hlavu, ruce, ale dokonce i oči či čelisti. Navíc framework umožňuje získávat data i z jiných zdrojů, než jsou senzory, jako je externí API, například Open Weather Map.

ManySense VR nasbíraná data ukládá do souboru ve formátu `CSV`, který následně posílá na server pro další analýzu a zpracování dat.

Mezi výhody ManySenseVR patří to, že se dá snadno implementovat do Unity, je snadno rozšířitelný a podporuje získávání dat z různých typů zdrojů.

Nevýhody jsou například nepohodlné získávání uložených dat a nevhodná implementace identifikace klíčů. Hlavní nevýhodou je, že nikde nebyl nalezen zdrojový kód frameworku, aby se dal použít či rozšířit.

2.5.2 Better Stack

Better Stack je platforma, která se zaměřuje na Log Management.[8] Umožňuje shromažďovat datové záznamy z celé aplikace. To znamená, že můžeme na jednom místě ukládat data jak z aplikace, tak ze serveru, na kterém aplikace běží. Ukládání dat zde probíhá přes REST api, pro které jsou i vytvořené balíčky pro jednotlivé programovací jazyky. Kromě ukládání dat Better Stack poskytuje i vytváření statistik a vizualizací nad těmito daty.

Mezi hlavní výhody Better Stacku patří možnost se na data dotazovat pomocí SQL jazyka, centralizace všech dat na jednom místě a transformace prostých textových dat do strukturovaného JSON formátu. Vývojáři tak nemusí mapovat textové záznamy na určité proměnné, ale Better Stack se to pokusí udělat sám. Dále poskytuje ukládání dat a jejich vizualizaci na jednom místě.

Nevýhodou Better Stack je ten, že není zadarmo. Z toho důvodu, když chceme využívat pokročilejší funkce jako spolupráce více lidí atd., musíme za toto řešení platit. Další nevýhodou je, že nebyla nalezena možnost, že by se dala data nějak strukturovat a rozdělovat, například podle typu aplikace a pak umožnit určitým členům přístup pouze k těmto záznamům.

2.5.3 Grafana

Grafana je open-source cloudové řešení pro sledování záznamových dat, které umožňuje vyhledávání, vizualizaci, zkoumání metrik a další.[9] Grafana se dokáže připojit na různé zdroje dat jako jsou Prometheus, ElasticSearch, MongoDB, CloudWatch. Na základě těchto zdrojů se v Grafaně dají vytvářet komplexnější vizualizace a statistiky ve formě Dashboardů.

Výhodou Grafany je, že umožňuje na základě určitých parametrů vytvářet různé dashboardy. Tzn. že v našem případě by mohl být pro každou aplikaci vytvořen samostatný dashboard se specifickými vizualizacemi a statistikami.

Nevýhodou Grafany je to, že musíme mít nějaké úložiště, kde máme záznamová data uložena. To znamená, že pro naše potřeby bychom museli implementovat dvě řešení, jedno pro ukládání dat a druhé pro vizualizaci.

2.5.4 Kibana

Kibana je open-source nástroj pro vizualizaci a průzkum dat, která jsou uložena v Elasticsearch.[10] Kibana je podobné řešení jako Grafana s tím, že je omezena pouze na jeden zdroj. Kibana je oproti Grafaně více zaměřená na analýzu dat, oproti tomu Grafana se více zaměřuje na zpracování dat v reálném čase.

Kapitola 3
Návrh řešení

Tato kapitola se zabývá konceptem nové aplikace pro sběr a analýzu dat z tréninkových VR aplikací. Je zde popsáno, jak by měla aplikace fungovat jako taková, z jakých celků bude aplikace složena a jak tyto celky budou navzájem propojeny ve smyslu jejich společné komunikace. Jsou zde doporučeny a uvedeny technologie, ze kterých by měla být aplikace vytvořena. Na konci kapitoly jsou uvedeny ukázky wireframů důležitých obrazovek a navržené formáty záznamových dat. Ná základě navrženého formátu jsou uvedeny možné vizualizace a statistiky, které lze z těchto dat provést.

V návaznosti na vzniklý návrh je možné začít s implementací aplikace. Aplikace by poté měla splňovat základní funkcionalitu, která se může dále rozšiřovat. Při návrhu řešení byly použity znalosti získané při analýze tohoto problému, které jsou popsány v kapitole 2.

3.1 Architektura aplikace

Vzhledem k tomu, že jedním z požadavků je, aby se jednalo ideálně o webovou aplikaci, je potřeba tomu přizpůsobit i výběr technologií, na kterých bude aplikace vyvinuta. Jako výhodu webové aplikace lze zmínit především to, že je dostupná kdekoli, kde se nachází internetové připojení. Při výběru technologií jsem bral v potaz i mé zkušenosti.

Standardní webová aplikace se ve většině případů skládá se tří základních stavebních kamenů, odborně tzv. třívrstvá architektura.[11] Jednotlivé úrovně se nazývají datová, aplikační a prezenční. Do datové úrovně patří databáze, jejímž cílem je ukládání všech dat, která pro aplikaci potřebujeme. Do aplikační úrovně patří serverová aplikace, která zpracovává všechny dotazy a tvoří logiku celé aplikace. Do poslední, prezenční úrovně patří klientská aplikace, což je uživatelské rozhraní, které interaguje s uživatelem a jeho požadavky posílá do aplikační vrstvy.

3.1.1 Doporučené technologie

Předně nutno podotknout, že výběr níže uvedených technologií není náhodný, avšak jedná se o velmi populární stack s názvem MEAN.[12] Jedná se o JavaScriptově založený framework pro vývoj webových aplikací, kde jsou 4 hlavní technologie, a to: MongoDB, Express.js, Angular a Node.js. Výhodou tohoto stacku je to, že všechny čtyři technologie jsou založeny na JavaScriptu a JSON, a tedy jejich integrita je intuitivní a přímočará. Také je tento stack vhodný pro použití real-time aplikací, jako je například aplikace kalendáře, nástroje pro správu pracovních postupů, stránky pro agregaci zpráv a další.

3.1.1.1 Databáze

Z omezení, která jsou zmíněna v sekci 2.2.2, je vhodné použít jednu z No-SQL databází.[13] Tyto databáze se vyznačují tím, že nemusí mít předem danou strukturu dat, tato struktura je variabilní a může se tedy kdykoli změnit. Daná vlastnost nám může pomoci v tom, že data z VR aplikací nemusíme mít nestrukturovaná například v souboru, ale můžeme je alespoň částečně strukturovat pro lepší použití. Další výhodou je,

že daný atribut nemusí mít přesně určený datový typ. Tento charakter se dá využívat, používáme-li netypizovaný programovací jazyk.

Jeden z hlavních zástupců je MongoDB. [14] MongoDB je open-source dokumentově založená databáze, která data ukládá ve formátu JSON. Předností MongoDB je, že je jednoduše škálovatelná a flexibilní a také podporuje značné množství programovacích jazyků, jako je například JavaScript, Java, Python, C++ a další. MongoDB podporuje též ukládání velkých souborů, což by se mohlo do budoucna hodit.

Pro navrhovanou aplikaci bych použil No-SQL databázi, konkrétně právě zmiňovanou MongoDB. To zaručí jednoduchý vývoj a navíc nezavře možnosti při případné budoucí implementaci nových funkcionalit.

3.1.1.2 Serverová aplikace

Serverovou aplikaci bych vytvořil na běhovém prostředí Node.js s frameworkem express.js.

Node.js je asynchronní běhové prostředí založené na javascriptovém enginu chrome V8, který běží například ve webovém prohlížeči Chrome.[15] Node.js oproti Chromu umožňuje spouštět javascriptový kód bez webového prohlížeče. To umožňuje využívat javascript i v serverových aplikacích, což je jeho primárním účelem. Node.js tak například klade velký důraz na vysokou škálovatelnost a zpracování velkého množství klientských dotazů.

Express.js je webový framework založený na základním Nodej.js http modulu a Connect komponentě, které se říká middleware.[16] Hlavní výhodou Express.js je to, že nám pomáhá se základními úkony, které bychom jako programátoři museli řešit. Jedná se například o parsování HTTP request body, parsování cookies, obsluhu chyb a další. Express.js je také otevřen implementacím dalších knihoven a frameworků, jako je například socket.io, což nám zaručuje rozšiřitelnost aplikace do budoucna.

3.1.1.3 Klientská aplikace

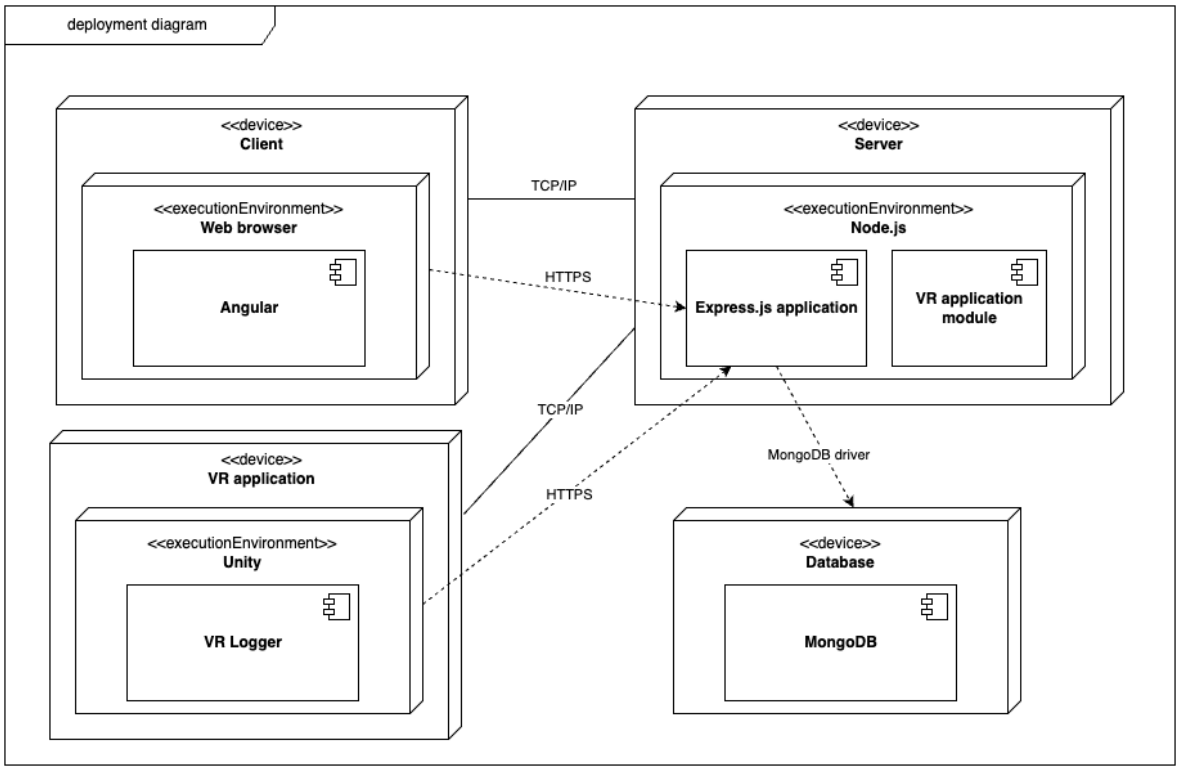
V posledních letech jsou nejvíce používané 3 frontend frameworky, a to Angular, React.js a Vue.js.[17] Na základě průzkumů vychází Angular jako nejrobustnější a dlouhodobě nejpoužívanější framework s neustálou podporou nových aktualizací.

Angular je někdy nazýván jako platforma, jelikož nabízí podporu velkého množství funkcí jako například kontrolování UI, validování uživatelských vstupů, posílání HTTP požadavků, atd.[18] Základní stavební kameny Angularu jsou moduly a komponenty. Komponenty jsou navrženy tak, aby se daly jednoduše přepoužívat v rámci celé aplikace a náš kód se nemusel opakovat. Díky tomu můžeme vytvářet složité klientské aplikace, které se dobře udržují.

Na základě těchto vlastností a i z toho důvodu, že již mám s Angularem mnoholeté zkušenosti, pro vývoj klientské aplikace volím právě Angular.

3.1.2 Diagram nasazení

V následující podsekci je na obrázku 3.1 grafické zobrazení jednotlivých celků/vrstev aplikace. Je zde uvedeno, z čeho se jednotlivé celky skládají a jak spolu komunikují. Proti technologiím zmíněným výše je v diagramu navíc komponenta VR application module. Tento modul představuje vlastní aplikaci, kterou vytvoří tvůrce VR aplikace, jež bude zpracovávat komplexnější statistiky. Kromě toho diagram také navíc obsahuje VR aplikaci, která představuje danou VR tréninkovou aplikaci. Tato aplikace využívá pro komunikaci se serverovou aplikací logovací balíček.



Obrázek 3.1. Diagram nasazení.

3.1.3 Doménový model

Doménový model slouží k definici základních entit a vztahů mezi nimi v rámci aplikace. Jinými slovy jsou to entity, které se můžou v rámci aplikace vyskytovat a které mají nějaké vlastnosti (atributy). Kladnou vlastností doménového modelu je to, že je na platformě nezávislý a zobrazuje nám strukturu aplikace ještě před samotnou implementací. Na základě doménového modelu se poté může navrhnout, jakým způsobem budou data ukládána do databáze a jak bude aplikace vyvíjena.

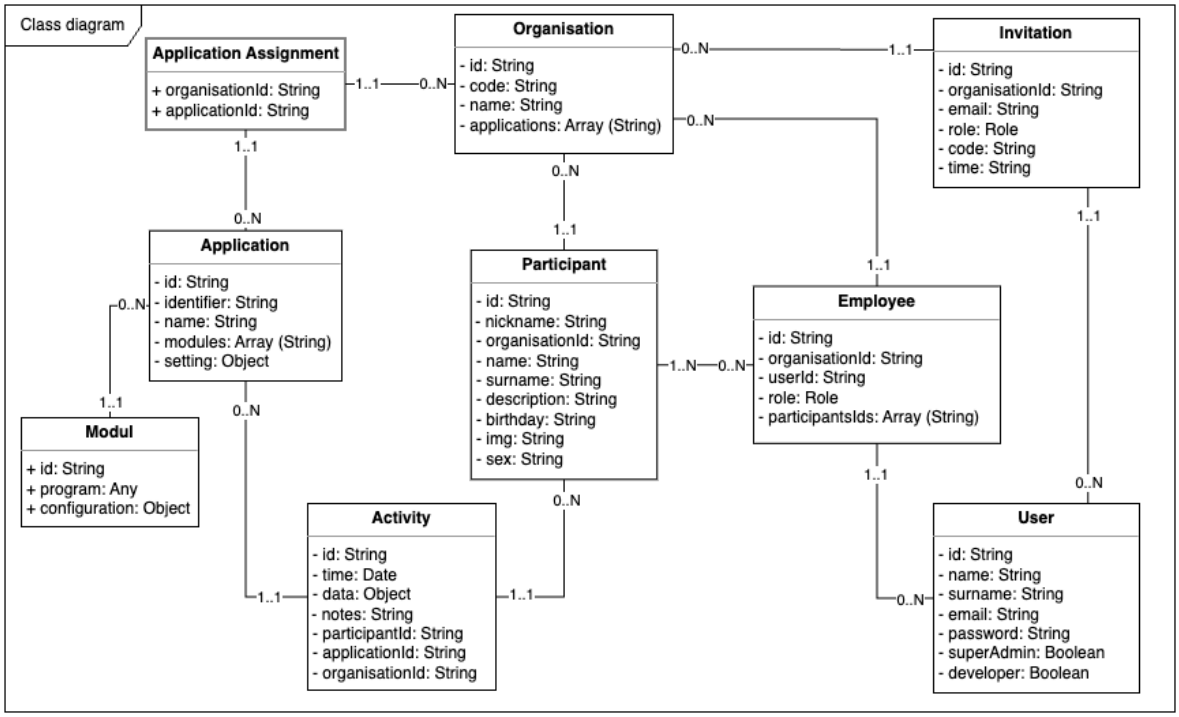
Konkrétní doménový model navrhované aplikace je na obrázku 3.2. Aplikace by měla obsahovat minimálně 8 entit. Hlavní z nich jsou:

- Uživatel
- Organizace
- Participant
- Aplikace
- Aktivita
- Modul
- Pozvánka
- Zaměstnanec

Uživatel bude moci zakládat libovolné množství Organizací a Participantů. Organizace bude muset mít minimálně jednoho Uživatele a pod Organizaci bude moci patřit libovolné množství Participantů. Participant bude muset být součástí jedné Organizace a o Participanta se bude muset starat minimálně jeden Uživatel, kterému v tomto vztahu říkáme Zaměstnanec. Zaměstnanec má v rámci Organizace specifikovanou roli. Uživatel je do Organizace zván pomocí Pozvánky. Organizace a Uživatel mohou mít libovolné množství pozvánek. Pozvánka je definována kódem a platností. Pod Organizací bude registrováno libovolné množství VR aplikací a VR aplikace bude moci být

u libovolného množství Organizací. Tato VR aplikace bude moci mít libovolné množství Modulů a Aktivit. Modul a Aktivita budou součástí pouze jedné VR aplikace. Aktivita bude volitelně součástí pouze jednoho Participanta a Participant bude moci vykonat libovolné množství Aktivit.

Každá entita je přesně identifikovatelná svým Id. Participant bude definovaný pomocí přezdívky. Pomocí přezdívky se Participant identifikuje v rámci VR aplikace. Uživatel oproti Participantovi bude definovaný skrz email a heslo, jimiž se bude do aplikace přihlašovat. Aktivita bude kromě Id definovaná i prostřednictvím času konání a bude dále obsahovat data pro statistiky a poznámky, které bude moci Uživatel zadávat.



Obrázek 3.2. Diagram tříd.

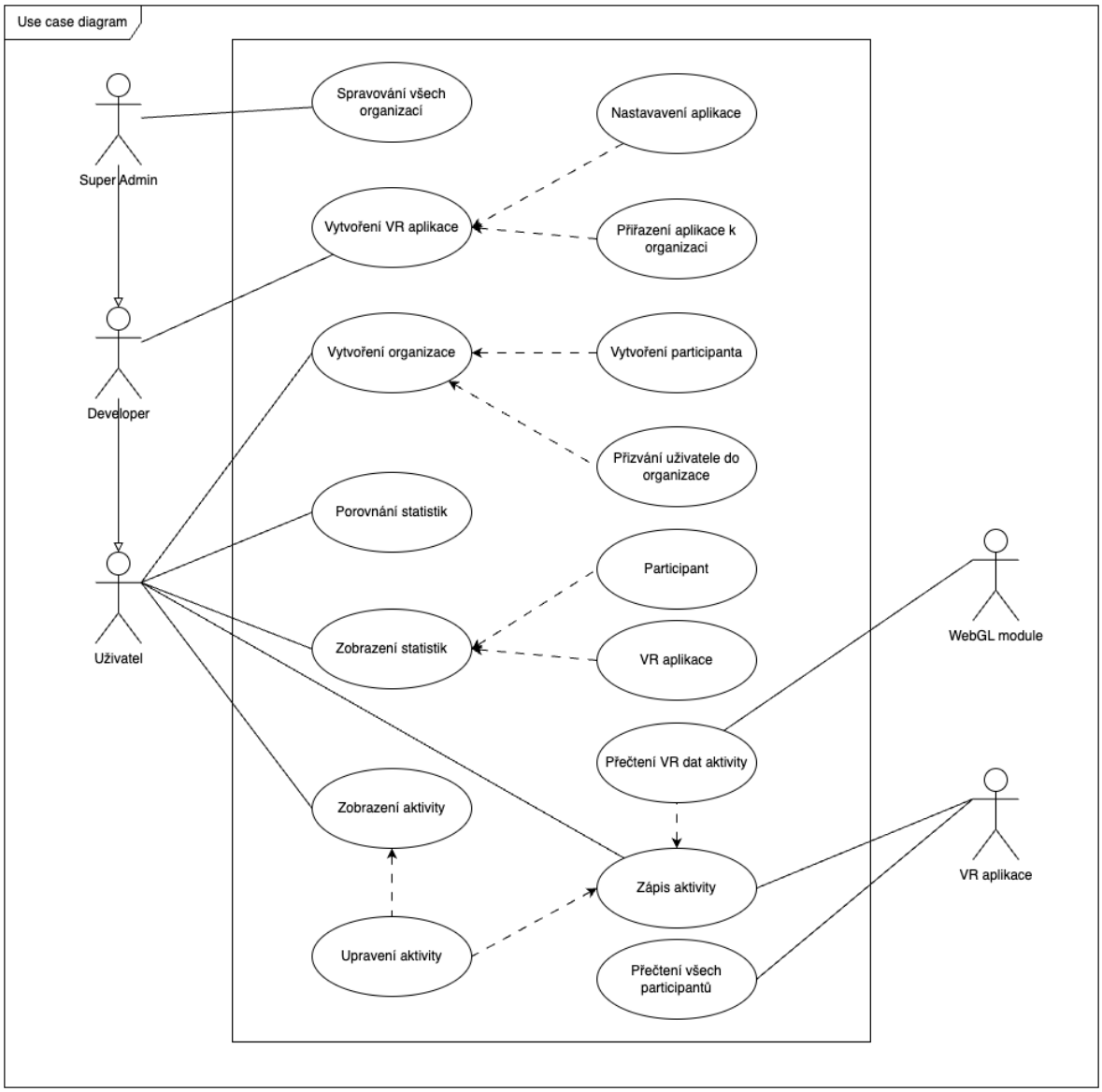
3.1.4 Diagram případů užití

Diagram případů užití slouží k definování a k lepší představě, co všechno má aplikace umět a kdo nebo co jí bude používat.[19] Diagram případů užití se skládá ze dvou komponent, a to za prvé aktérů a za druhé use case (případů užití). Aktéři jsou lidé, případně externí systémy, jež budou se systémem interagovat. Use casy jsou aktivity, které systém provádí pro aktéry. Tyto use casy jsou nedekomponovatelné.

Jak je vidět na diagramu 3.3 níže, s aplikací bude interagovat pět aktérů:

- Super admin
- Developer
- Uživatel
- VR aplikace
- WebGL modul

Na diagramu se vyjma klasických vazeb vyskytují i vazby «extend», které definují, že daný use case je závislý na předešlém use case. Jako příklad je use case `Vytvoření participanta`, kterou Uživatel může provést až poté, co provede `Vytvoření Organizace`.



Obrázek 3.3. Diagram případů užití.

3.2 Wireframy klientské aplikace

Wireframy slouží jako předběžný návrh designu aplikace. Hlavní funkcí wireframů je ukázat základní rozložení elementů, vytvořit základní strukturu aplikace, podklady pro grafika a designera, atd. Existují jak papírové, tak i digitální wireframy. V této podkapitole jsou představeny wireframy základních obrazovek, které by uživatelé měli využívat většinu času. V mém případě jsem vytvořil digitální wireframy pomocí aplikace Figma.

Na základě těchto wireframů je možné si zvalidovat návrh aplikace s jejími budoucími uživateli. Na základě vstupů od uživatelů se návrhy mohou upravit, ba dokonce kompletně předělat. Využití wireframů je velmi užitečné, jelikož to z velké části zabraňuje úpravě aplikace až ve fázi vývoje.

3.2.1 Participant

Na obrázku 3.4, je vidět návrh obrazovky s detailem participanta. Obrazovka se skládá ze 3 částí, a to `Základní informace`, `Statistiky` a `Aktivity`. Základní informace uvádějí elementární údaje o participantovi. V části Statistiky jsou souhrnné statistiky jako například počet vykonaných aktivit, čas poslední aktivity a průměrné skóre. V sekci Aktivity je seznam všech vykonaných aktivit participantem.

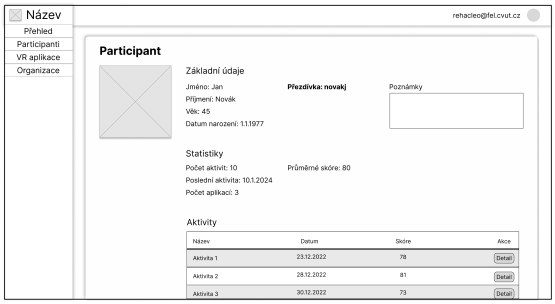
3.2.2 Aktivita

Wireframe zobrazující detail aktivity je ke shlédnutí na obrázku 3.5. Na obrazovce jsou uvedeny základní informace o aktivitě, jako je například délka trvání a v jaké VR aplikaci je aktivita zaznamenána. Také zde lze zjistit elementární údaje o participantovi, který aktivitu vykonal. Výzkumník k aktivitě může též přidat své i participantovy poznámky. Součástí jsou taktéž základní statistiky i pokročilé statistiky, jako je v tomto případě mapa průchodu úrovní VR aplikace.

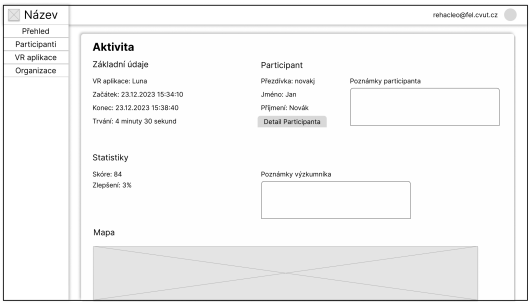
3.2.3 Porovnání aktivit

Poslední wireframe, který je zobrazen na obrázku 3.6, nám ukazuje, jak konkrétně může vypadat obrazovka, která slouží k porovnání aktivit. Je zde použité zobrazení dvou obrazovek vedle sebe, aby uživatel nemusel při porovnávání přecházet z jedné stránky na druhou. U každé aktivity jsou viditelné základní informace a statistiky. Posledně zde nechybí ani pokročilé statistiky, v tomto případě mapa.

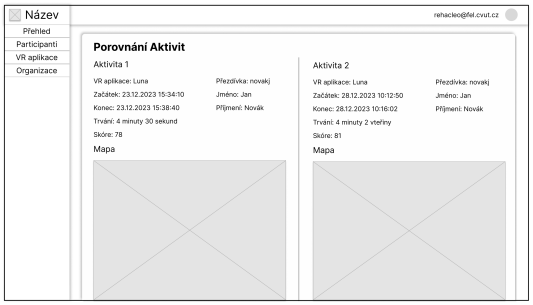
Následující obrázky 3.4, 3.5 a 3.6 jsou v plné velikosti zobrazeny v příloze A.



Obrázek 3.4. Zmenšený wireframe obrazovky detailu participanta.



Obrázek 3.5. Zmenšený wireframe obrazovky detailu aktivity.



Obrázek 3.6. Zmenšený wireframe obrazovky porovnání 2 aktivit.

3.3 Formát záznamových dat z VR aplikací

Jak již bylo zmíněno v sekci Způsoby ukládání záznamových dat 2.4, existují různé formáty, do kterých je možné VR data ukládat. Jako nejvhodnější pro naše potřeby je využití formátu JSON, který přináší určité výhody.

Formát JSON je plně podporovaný jak Angularem, Nodej.js, tak i MongoDB databází, tzn. neměly by při implementaci nastat problémy s kompatibilitou. Díky této podpoře je možné využívat složitější operace nad daty, jako je například filtrování, mapování a další.

JSON je také rozšiřitelný a zpětně kompatibilní. To se může do budoucna hodit, pokud bychom kromě sledování pozice hlavy a rukou chtěli sledovat například i nohy. V tomto případě tento formát pouze rozšíříme o nové atributy.

Na základě analýzy existujících VR tréninkových aplikací byla navržena následující podoba JSON formátu, který pokrývá aktuální i budoucí potřeby. Logovací data obsahují tyto parametry: aplikační identifikátor, čas startu, čas konce, verze logu, jak často se dané logy zaznamenávají, volitelné informace a záznamy.

Každý záznam se skládá z časového razítka, pořadí záznamu, pole událostí, environmentu, volitelné informace a části těla, kterou sledujeme. Pole událostí představují události, které nastaly v časovém horizontu mezi dvěma záznamy. Environment určuje prostředí, ve kterém byl záznam pořízen. Může se jednat například o identifikátor úrovně. Každá sledovaná část obsahuje pozici a rotaci. Pozice i rotace jsou uvedeny v ose x, y a z.

Logovací data vždy patří jedné aktivitě. Aktivita navíc obsahuje identifikátor, datum vytvoření, logovací data, poznámky a volitelně identifikátor participanta.

Ukázka logovacích dat ve formátu JSON je dostupná v příloze C.1

3.4 Kategorizace záznamových dat

V této sekci se zaměříme na kategorizaci logovacích dat podle určitých aspektů. Při kategorizaci jsem vycházel z knihy *Visualization of Time-Oriented Data*[20], ze 3. kapitoly.

Jedná se o časově orientovaná data, jelikož je sledujeme po nějakou dobu vždy v uvedený moment. To znamená, že například u pozice hlavy pozorujeme v čase tři osy: x, y, z. Jinak řečeno, naším cílem je zobrazit v čase tři různé dimenze. Je to to samé, jako bychom chtěli například v čase zobrazit teplotu, vlhkost a tlak. Na obrázku 3.7 jsou zobrazeny všechny aspekty a jejich kategorie, které se u časově orientovaných dat uvažují.

3.4.1 Charakteristika času

Z hlediska typu času se jedná o diskrétní doménu. Diskrétní časové hodnoty lze mapovat na celá čísla a můžeme uvažovat o jejich časové vzdálenosti. V našem případě u každého záznamu známe časové razítko, které je možné převést na UNIXový čas a číslo záznamu. To znamená, že záznamy tuto podmínku splňují. Další varianty jsou ordinální a průběžný. V ordinální doméně známe pouze pořadí jednotlivých záznamů. U průběžné domény vždy mezi dvěma časovými záznamy existuje další. Toto v našem případě neplatí, jelikož zaznamenáváme data například jednou za 300 milisekund.

Dále z hlediska rozsahu se jedná o bodový typ, jelikož o oblasti mezi dvěma záznamy nemáme žádné informace. Jinak řečeno, mají časový rozsah rovný nule. Naproti tomu intervalový typ má časový rozsah větší než nula.

Z hlediska uspořádání se jedná o lineární data. To znamená že záznamy plynou v jednom směru, od minulosti do budoucnosti. Toto je náš případ, jelikož každý záznam má svého předchůdce a následovníka. Druhou variantou jsou cyklická data. V takovém případě se časové údaje opakují, například roční období, dny v týdnu a další.

Data patří mezi instantní kategorie z hlediska aspektu časového primitiva. Je to kvůli tomu, že data jsou zachycena ve specifický okamžik, nikoli v rozmezí. Dalšími variantami časového primitiva jsou intervalová kategorie a kategorie rozpětí.

3.4.2 Charakteristika časově orientovaných dat

Z hlediska měřítka rozdělujeme data do dvou skupin proměnných, a to jednak kvantitativních a jednak kvalitativních. V našem případě se jedná o kvantitativní proměnné, jelikož se dají číselně srovnávat.

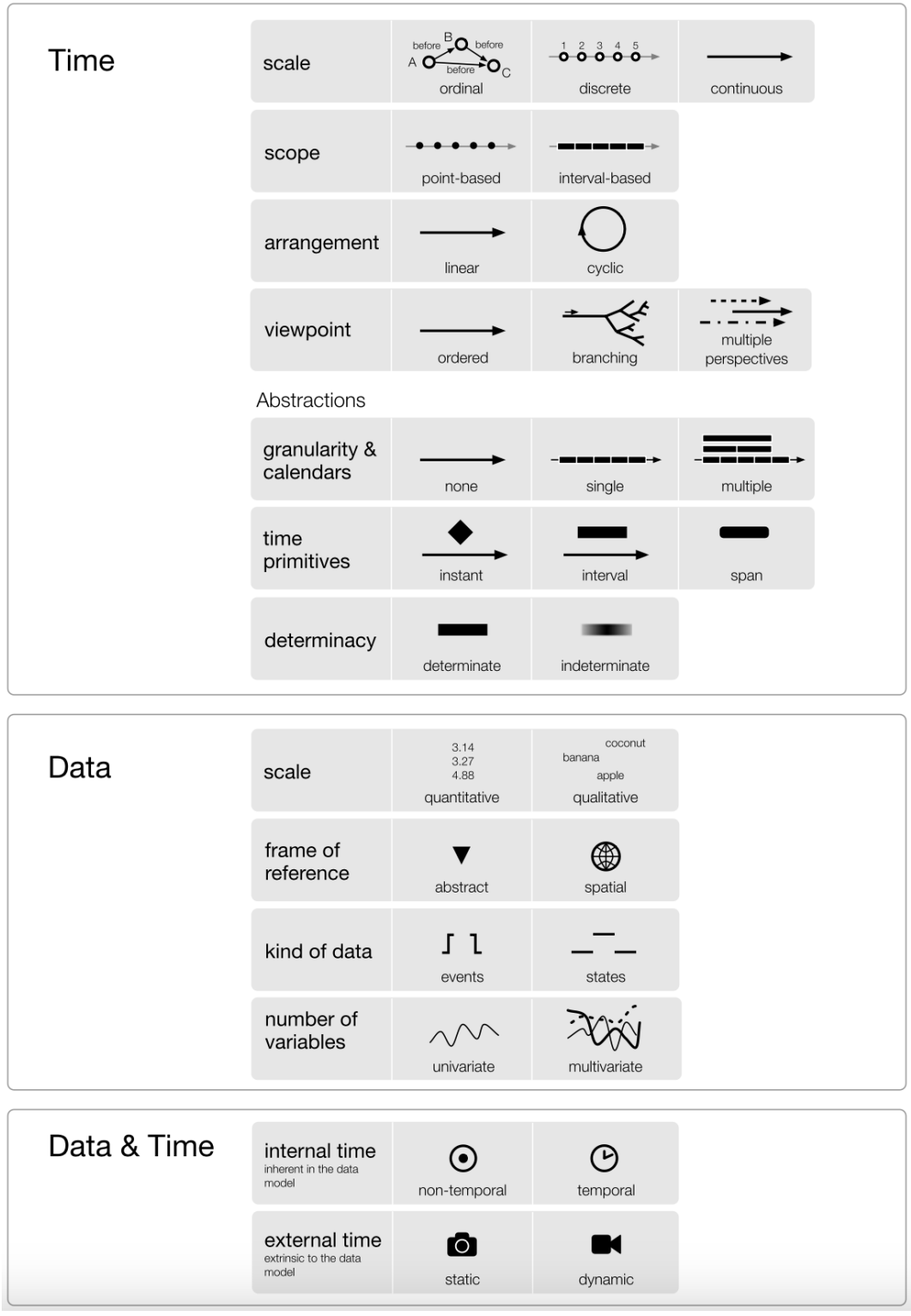
Podle referenčního rámce členíme data na abstraktní a prostorová. Abstraktní data nejsou sama o sobě spojena s prostorovým místem. Naproti tomu prostorová data obsahují dané místo v prostoru. Příkladem mohou být souřadnice na mapě. V našem případě data spadají mezi abstraktní. Abychom je mohli začlenit jako prostorová, museli bychom mít představu o tom, jak daná scéna vypadá.

Z hlediska počtu proměnných data spadají do obou kategorií. Mezi univariační patří v tu chvíli, pokud budeme chtít zobrazit pouze jednu osu. Naopak data mohou být multivariační v okamžiku, pokud budeme chtít zobrazit více os najednou.

3.4.3 Shrnutí

Pro výběr vhodných vizualizačních technik jsou nejdůležitější tyto následující aspekty: uspořádání, časová primitiva, referenční rámec a počet proměnných. U každého aspektu je zmíněno do jaké kategorie spadají záznamová data.

- uspořádání - lineární
- časová primitiva - instantní
- referenční rámec - abstraktní
- počet proměnných - univariační i multivariační podle typu použití



Obrázek 3.7. Návrhové aspekty časově orientovaných dat.

3.5 Statistiky a vizualizace aktivit

Aby se daly jednotlivé aktivity porovnávat a aby uživatelé aplikace mohli se záznamy pracovat, bylo by vhodné zobrazit na detailu aktivity základní statistiky a vizualizace. V této kapitole jsou popsány jednotlivé metody a vizualizace, které je možné získat z námi definovaného formátu záznamových dat, jež je uveden v sekci 3.3.

3.5.1 Základní statistiky aktivity

V této podsekci jsou popsány statistiky, které by bylo vhodné implementovat. Mohlo by se jednat o následující statistiky:

- doba trvání aktivity
- doba od konání aktivity
- počet záznamů
- počet událostí
- doba trvání jednotlivých environmentů

Kromě statistik lze ze záznamu také vyčíst volitelné informace. Pro volitelné informace jednotlivých záznamů, pokud budou mít číselnou povahu, by bylo vhodné vytvořit agregace, jako je například součet, průměr, maximální a minimální hodnota a další. Pro ostatní formáty jako text by bylo možné zobrazit počet výskytů dané textové hodnoty.

3.5.2 Vizualizace

Na základě kategorizace dat v předchozí sekci 3.4 jsou popsána vhodná zobrazení pro tento typ. Vhodnost typu grafů podle kategorie byla čerpána z knížky *Visualization of Time-Oriented Data*[20] z tabulky 7.1.

Jako druh mapování jsem vybral statické, jelikož oproti dynamickým se dají lépe porovnávat. Z hlediska dimenze jsem vybral 2D dimenzi pro jednodušší vyobrazení.

V následující tabulce 3.1 je popsána kategorizace doporučených vizualizací.

Vizualizace	lineární	instantní	abstraktní	univariační	multivariační	statické	2D
Point Plot	X	X	X	X		X	X
Line Plot	X	X	X	X		X	X
Bar Graph	X	X	X	X		X	X
Silhouette Graph	X	X	X	X		X	X
Layer Area Graph	X	X	X		X	X	X
ThemeRiver	X	X	X		X	X	X
Pixel-Oriented	X	X	X		X	X	X
CircleView	X	X	X		X	X	X

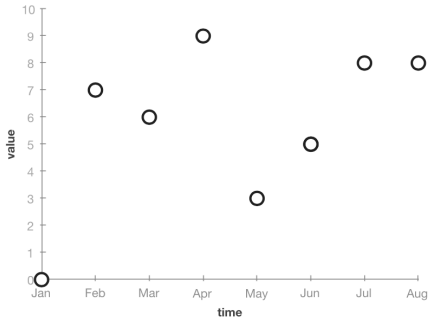
Tabulka 3.1. Kategorizace doporučených vizualizací.

Následuje popis doporučených vizualizací, které jsou rozděleny na univariační a multivariační.

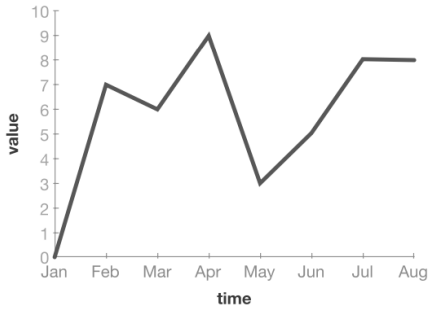
Univariační

- Point Plot (bodový graf) - Tento typ zobrazuje kartézský souřadnicový systém s časem na vodorovné ose a odpovídající hodnotou na svislé ose. Pro každou dvojici čas-hodnota je zanesen do systému bod. Tento typ je vhodný pro zdůraznění jednotlivých bodů.

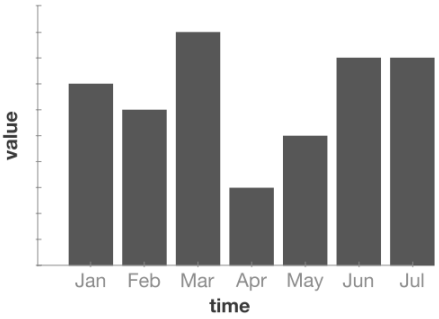
- Line Plot (spojnicový graf) - Jedná se o nejběžnější formu zobrazení časových dat. Rozšiřuje bodový graf tak, že jednotlivé body spojuje přímkami a tím zdůrazňuje jejich časovou souvislost. V důsledku toho se tento typ grafu zaměřuje na celkový tvar v čase.
- Bar Graph, Spike Graph (sloupcový graf) - Tento typ pro znázornění hodnot dat používá sloupce. Díky tomu se jednotlivé hodnoty porovnávají daleko lépe než například u bodového grafu. Protože se hodnoty dat znázorňují jako délky sloupce, lze zobrazit pouze proměnné se stupnicí obsahující nulu.
- Silhouette Graph (siluetový graf) - Tento typ rozšiřuje spojnicový graf. A to tak, že vyplňuje oblast pod spojnicí. To vede k výrazným siluetám, které zlepšují vnímání a také porovnání více časových řad mezi sebou.



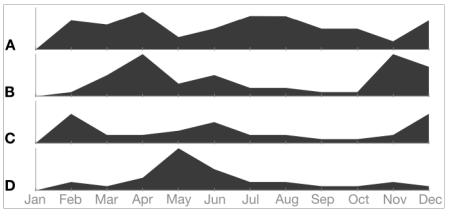
Obrázek 3.8. Ukázka point plot grafu.



Obrázek 3.9. Ukázka line plot grafu.



Obrázek 3.10. Ukázka sloupcového grafu.

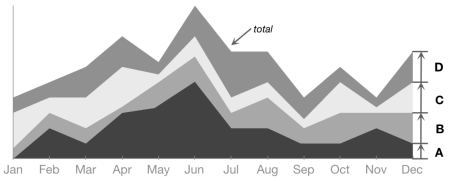


Obrázek 3.11. Ukázka silhouette grafu.

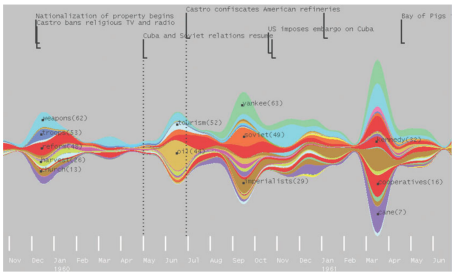
Multivariační

- Layer Area Graph (graf plošných vrstevnic) - Tento typ zobrazuje jednotlivé časové řady vrstvené na sobě. Hodnoty dat jsou zobrazeny jako výška vrstevnice. Tento typ je vhodný pouze pro časové řady, které mají stejnou jednotku hodnot a hodnoty se dají sčítat. Při implementaci vizualizace je potřeba si dávat pozor na pořadí časových řad, protože to ovlivňuje podobu grafu. Výhodou tohoto typu je, že znázorňuje celkový součet hodnot a zároveň poskytuje informaci o jednotlivých částech.
- ThemeRiver - Tento typ zobrazuje změny různých proměnných. Každá proměnná je zobrazena jako barevný pruh, jehož šířka se mění podle hodnoty v čase. Graf zobrazuje přehled proměnných, které byly v daném čase důležité.
- Pixel-Oriented Network Visualization - Tento typ vizualizuje matici příslušnosti jako maticovou tabulku, kdy sloupce a grafy znázorňují dvě různé proměnné. Tmavost a hodnota jednotlivých buněk znázorňuje příslušnost kategorií daných proměnných. Pokud je hodnota 0, tak je pozadí bílé.

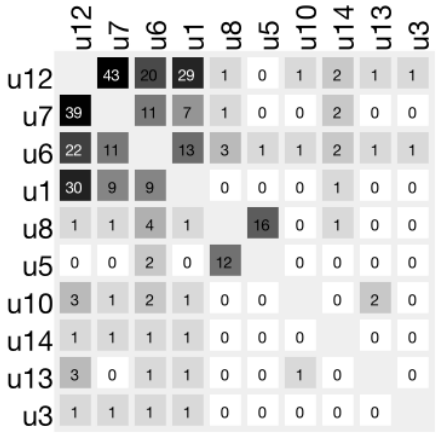
- CircleView - Tento typ zobrazuje proměnné jako výseče kruhu. Tyto výseče jsou dále rozděleny na sloty, které představují časové úseky. Barva slotu znázorňuje hodnotu dané proměnné v daném časovém úseku.



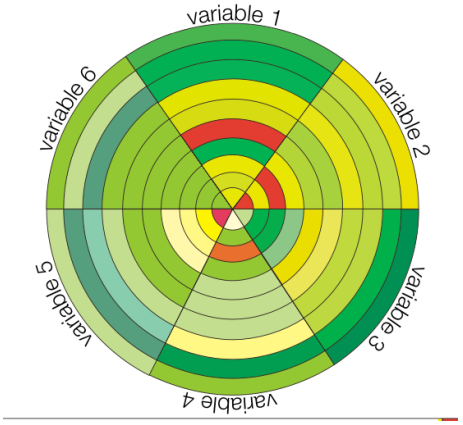
Obrázek 3.12. Ukázka layer area grafu.



Obrázek 3.13. Ukázka ThemeRiver grafu.



Obrázek 3.14. Ukázka Pixel-Oriented grafu.



Obrázek 3.15. Ukázka CircleView grafu.

Kapitola 4
Implementace

V následující kapitole jsou popsána specifika jednotlivých aplikací, které bylo nutno vyvinout v rámci diplomové práce. Jednotlivé bloky odráží poznatky popsané v kapitole Návrh řešení 3. V závěru kapitoly je sekce 4.5, která popisuje celkové propojení jednotlivých aplikací.

U každé aplikace jsou popsána její specifika, která bylo potřeba při implementaci vyřešit, či která jsou oproti běžným aplikacím zajímavá.

Jednotlivé aplikace jsem zahrnul pod jeden projekt, kterému jsem dal název VR dashboard. Pokud je v textu zmíněn VR dashboard nebo VR dashboard aplikace, je tím myšlena právě vyvinutá aplikace jako celek.

4.1 Klientská aplikace

Klientská aplikace je napsána ve frameworku Angular. Konkrétněji se jedná o verzi 16, která byla vydána v květnu 2023. V Angularu se aplikační celky dělí na moduly, které se skládají z komponent. Komponenta by měla být část kódu, která se dá opakovaně přepoužít a má jasně danou funkcionalitu. Komponenty se v Angularu skládají ze 3 samostatných souborů, které představují aplikační logiku, html šablonu a css styly.

Klientská aplikace je dostupná na adrese[1] uvedené v poznámce pod čarou. Zdrojový kód klientské aplikace a návod, jak spustit aplikaci lokálně, jsou dostupné na adrese[2] uvedené v poznámce pod čarou.

4.1.1 Použité balíčky a knihovny třetích stran

Při implementaci klientské aplikace byly využity následující balíčky a knihovny třetích stran, které usnadnily její vývoj.

- coreui[3] - kompletní Dashboard UI kit, který obsahuje již nadefinované komponenty a šablony stránek, ze kterých je možné vytvářet uživatelské rozhraní. Tento kit se používá jako hlavní designovací nástroj pro tvorbu stránek. Umožňuje používat předem definovaný styl html elementů přidáním css tříd k danému html elementu.
  - Je použita bezplatná Free Angular Admin Template šablona, která je poupravena tak, aby lépe odpovídala požadavkům.
- iconify[4] - framework, pomocí kterého se mohou jednoduše přidávat svg ikony do aplikace. Tento framework se používá pro získání ikon do aplikace, kdy se do html elementu přidá název ikony z knihovny a ikona se automaticky stáhne a zobrazí.
- ngx-translate[5] - balíček pro Angular, pomocí něhož lze vytvářet vícejazyčné aplikace (i18n). Tento balíček je použit pro implementaci multijazyčnosti aplikace. Přidává

[1] https://vr-dashboard.leosrehacek.com/
[2] https://gitlab.fel.cvut.cz/rehacleo/vr_dashboard_fe
[3] https://coreui.io/product/free-angular-admin-template/
[4] https://iconify.design/
[5] http://www.ngx-translate.com/

podporu i18n standardu a obsahuje metody pro snadnou implementaci do Angular aplikace.
- apexcharts[1] - balíček pro tvorbu grafů, který pomáhá vývojářům vytvářet uživatelsky přívětivé a interaktivní vizualizace pro webové stránky. Tento balíček je použit pro zobrazení vizualizací jednotlivých aktivit.
- dayjs[2]- balíček, který analyzuje, ověřuje, manipuluje a zobrazuje data a časy. Tento balíček je použit pro formátování data z ISO formátu a také například pro výpočet doby trvání.
- monaco-editor[3] - balíček, který obsahuje editor kódu. Na Monaco editoru je například vytvořený VS Code od společnosti Microsoft, který je celosvětově populární mezi vývojáři. Tento balíček je použit pro implementaci editoru pro úpravu nastavení jednotlivých aplikací. Takové nastavení je popsáno v podsekci 4.1.5.

4.1.2 Vizualizace aktivity

V následující sekci je popsáno, jak jsou jednotlivé aktivity zobrazeny v klientské aplikaci. Kromě klasického textového výpisu byly vytvořeny různé typy grafů, které si představíme. Pro vykreslení grafů byl použit balíček `apexcharts`. Návod, jak je možné typy grafů nastavit pro danou VR tréninkovou aplikaci, je dostupný v sekci E.2.

Jednotlivé typy grafů si budeme popisovat na základě sledování změny pozice hlavy a rotace hlavy, které máme dostupné ze záznamů aktivit více VR tréninkových aplikací.

U některých typů grafů jsou zobrazeny 2 různé vizualizace, pokaždé z jiné VR tréninkové aplikace. V rámci toho je popsáno i porovnání těchto vizualizací a vysvětleno, kdy je vhodné tento graf použít.

4.1.2.1 Rotace hlavy pomocí spojnicového grafu

Zobrazení rotace pomocí spojnicového grafu, který je nazván jako `rotation`, se vykresluje jako linka. Pro každou osu x, y a z je samostatná linka, pokaždé s jinou barvou. Barvy linek odpovídají barvám os, jak jsou nadefinované v unity.

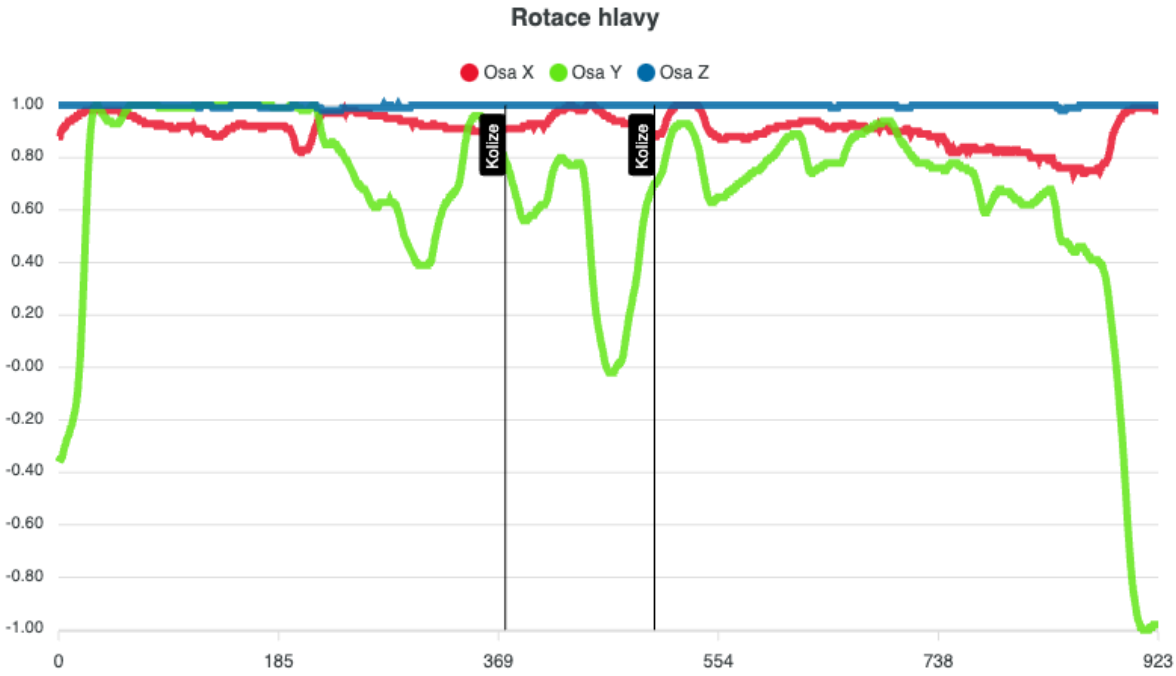
Uživatel si může vybrat, jaké osy chce zobrazit a jaké schovat po kliknutí na titulek osy. Navíc tento typ grafu podporuje i přiblížení a oddálení. To znamená, že uživatel si může vybrat menší časový úsek, který graf vykresluje, a tak zobrazit větší detail.

V Unity se rotace podle osy X označuje jako `pitch`, osa Y jako `yaw` a osa Z jako `roll`. Pokud toto převedeme na rotaci hlavy, tak rotace podle osy X znamená, že hlavu nakláníme nahoru a dolů. Rotace podle osy Y znamená, že hlavou otáčíme doprava a doleva. Rotace podle osy Z znamená, že hlavu nakláníme na pravou a levou stranu.

Osa na grafu nabývá na svislé ose hodnot od -1 do 1. Zobrazuje to přepočet rotace na cosinus daného úhlu. To znamená že pro úhel 0° je hodnota cosinu 1 a pro úhel 180° je hodnota cosinu -1. Jinak řečeno, pokud se uživatel koukal vpřed, hodnota linky osy Y na grafu je 1 a pokud se uživatel koukal vzad, hodnota osy Y na grafu je -1.

Tento typ grafu navíc podporuje zobrazení událostí. V tomto případě se pro daný záznam zobrazí svislá čára s typem události.

[1] `https://apexcharts.com/`
[2] `https://day.js.org/en/`
[3] `https://microsoft.github.io/monaco-editor/`



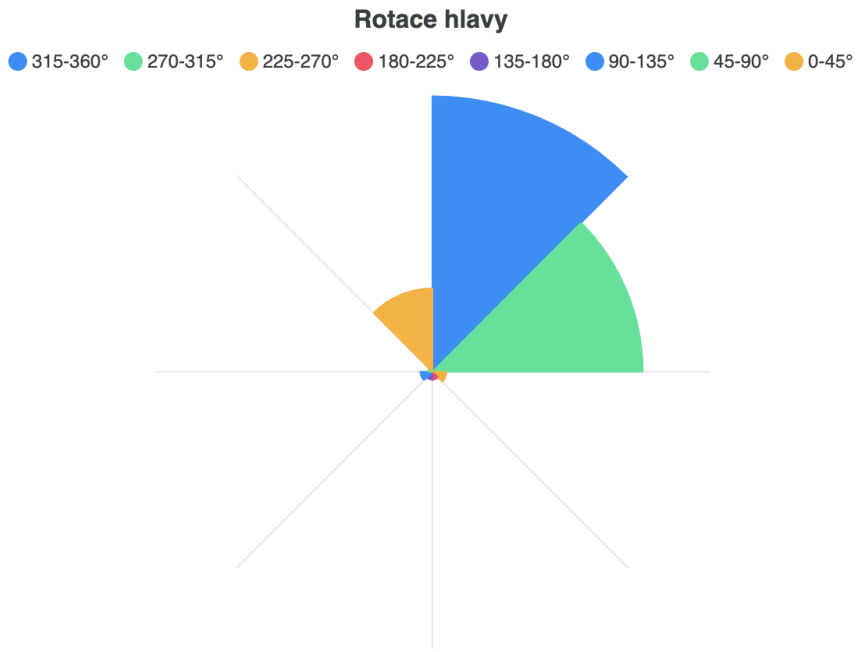
Obrázek 4.1. Graf zobrazující rotaci hlavy v aplikaci Luna.

Z přiloženého obrázku 4.1 můžeme vypozorovat, že participant skoro vůbec nenaklánět hlavu na strany, jelikož hodnota osy Z je 1. Dále můžeme vyčíst, že se participant občas podíval trochu nahoru nebo dolů, jelikož hodnota osy X je blízká 1. A v poslední řadě, pokud se podíváme na osu Y, tak se participant ze začátku díval vpřed, poté se občas podíval na strany a na konec se podíval vzad.

Dále můžeme vypozorovat, že v průběhu došlo ke dvěma událostem. V obou případech se jednalo o typ události kolize.

4.1.2.2 Rotace hlavy pomocí polar area grafu

Rotace pomocí tohoto typu grafu, který je nazván jako `rotation-polar`, je vykreslena jako jednotková kružnice. Daná kružnice je rozdělena na 8 dílů. Každý díl představuje interval s rozsahem 45°. Jednotková kružnice je oproti standardnímu zobrazení otočená o 90°, kdy 0° ukazuje směrem nahoru. Toto otočení je implementováno kvůli lepší představě, když se na graf díváme. Míra vyplnění jednotlivého dílku grafu představuje poměr výskytu v procentech jednotlivých záznamů v daném rozsahu rotace vůči ostatním dílkům.



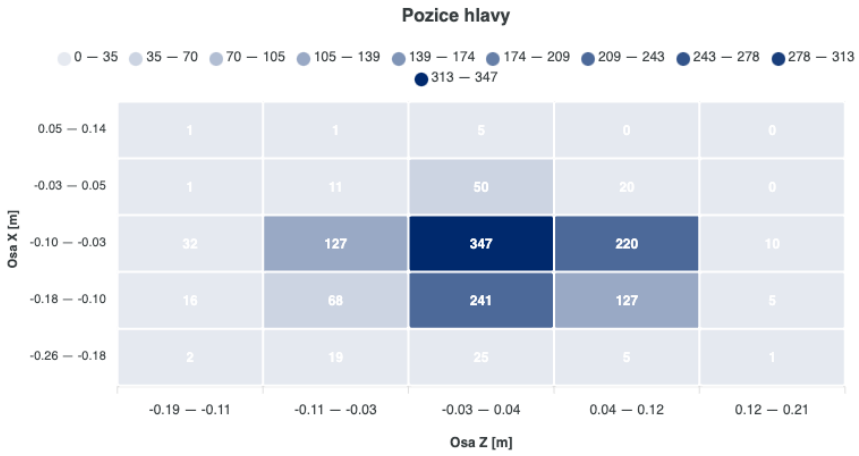
Obrázek 4.2. Graf zobrazující četnost úhlu rotace hlavy v aplikaci Luna.

Z přiloženého obrázku 4.2 můžeme vyčíst, že se participant většinu času koukal mírně doprava nebo doprava. Také se krátce podíval mírně doleva a na krátkou chvíli otočil hlavu dozadu.

4.1.2.3 Pozice hlavy pomocí heatmap grafu

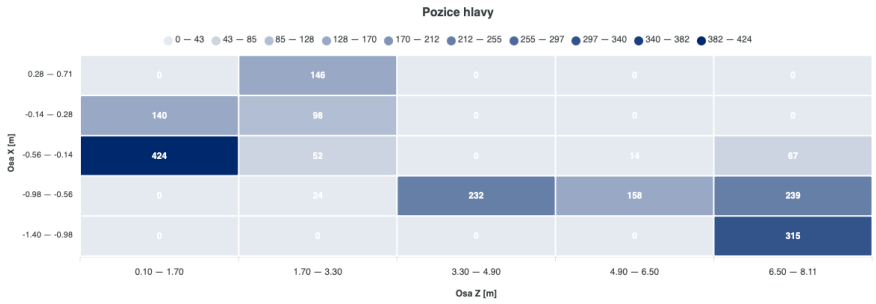
Graf zobrazující pozici hlavy pomocí heatmapy je nazván jako `position-heatmap` a zobrazuje počet záznamů v daném sektoru. V tomto případě graf zobrazuje pozici osy X vůči ose Z. Graf je rozdělen na 5 sloupců a 5 řádků, které dohromady tvoří matici s 25 sektory.

Záznamy jsou rozděleny do jednotlivých sektorů tak, že se nejdříve zjistí minimální a maximální hodnota v každé ose. Poté se tento interval pro každou osu rozdělí rovným dílem na 5 intervalů. Poté se prochází jednotlivé záznamy a u každého záznamu podle hodnoty osy X a Z přidělíme sektor podle toho, do jakého intervalu spadá osa X a osa Z.



Obrázek 4.3. Graf zobrazující četnost výskytu pozici hlavy v aplikaci PhysioTrails.

Na přiloženém obrázku 4.3 sledujeme pozici hlavy v rámci aplikace PhysioTrails. Pomocí tohoto grafu se snažíme zjistit, jak moc se participant vychyloval od ideální klidné polohy. Pokud by byl participant celou dobu naprosto klidný a nevychyloval by se, měl by všechny záznamy v prostředním sektoru, to znamená ve třetím sloupci a třetím řádku. Na obrázku můžeme vidět, že se uživatel i drobně vychýlil doprava, doleva, ale i dozadu.



Obrázek 4.4. Graf zobrazující četnost výskytu pozici hlavy v aplikaci Luna.

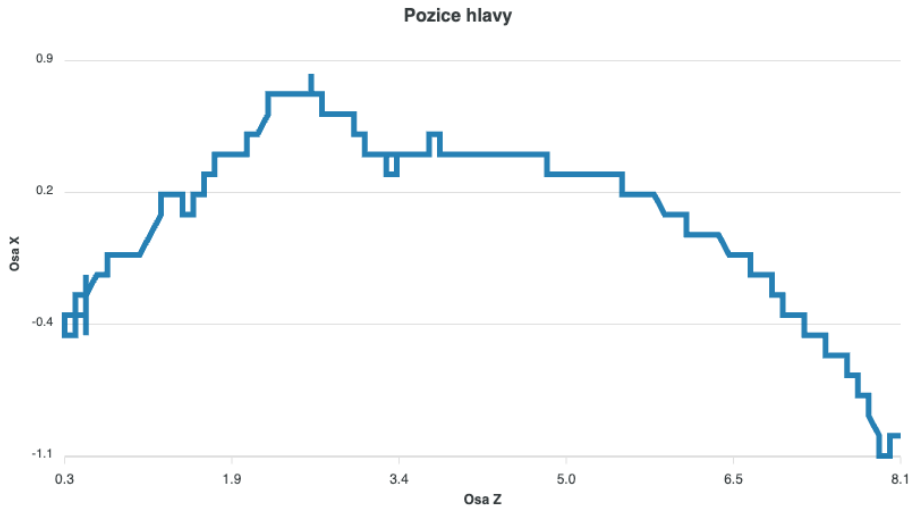
Na obrázku 4.4 je zobrazena pozice hlavy v rámci aplikace Luna. V tomto případě nám četnosti výskytů spíše ukazují, jak se participant fyzicky pohyboval, a nezobrazují nám vychýlení od ideální klidné pozice.

Tento typ grafu není vhodný na sledování pohybu participanta, jelikož je tento pohyb rozdělen pouze do 25 polí a můžeme tak přijít o větší detail. Z toho důvodu je tento graf vhodný pouze pro aplikace, kde jsou participanti stále na jednom místě. Příkladem může být například balanční aplikace, prostřednictvím níž chceme sledovat vychýlení participanta.

4.1.2.4 Pozice hlavy pomocí spojnicového grafu

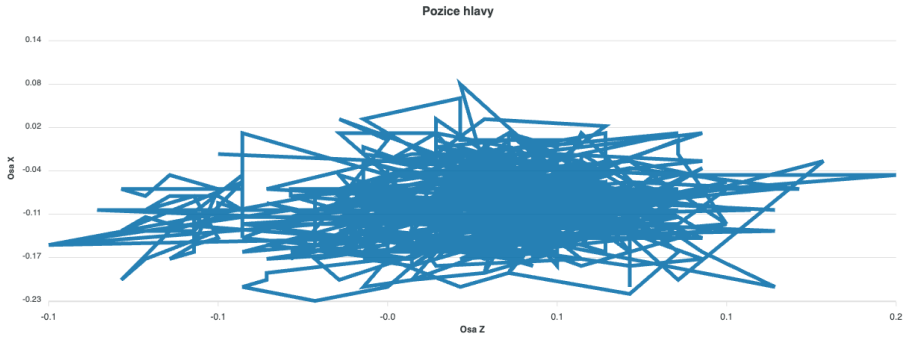
Graf zobrazující pozici hlavy pomocí spojnicového grafu, který je nazván jako `position`, zobrazuje průchod participanta prostředím. Na svislé ose je namapovaná osa X a na vodorovné osa Z. Jinak řečeno, graf zobrazuje cestu, jak se participant pohyboval v rámci prostoru.

Po najetí na jednotlivé body linky grafu je zobrazena nápověda s uvedeným číslem záznamu. Podle toho se dá poznat, jak jednotlivé body byly zaznamenány za sebou.



Obrázek 4.5. Graf zobrazující pozici hlavy v aplikaci Luna.

Na obrázku 4.5 je zobrazena pozice hlavy participanta v rámci aplikace Luna. Z obrázku můžeme vyčíst, jak se uživatel pohyboval v prostoru. Osa Z zde znázorňuje délku a osa X šířku určitého prostoru.



Obrázek 4.6. Graf zobrazující pozici hlavy v aplikaci PhysioTrails.

Na obrázku 4.6 je zobrazena pozice hlavy participanta v rámci aplikace PhysioTrails. Bohužel z tohoto grafu nelze odvodit pohyb uživatele v prostoru. Jediné, co dokážeme určit, je, že se uživatel pohyboval na velmi malém prostoru.

Rozdíl je dán tím, že participanti aplikace Luna se fyzicky pohybují v prostoru. Oproti tomu participanti aplikace PhysioTrails zůstávají fyzicky na jednom místě. Z toho důvodu je vhodné tento typ grafu použít v aplikacích, kde se participanti fyzicky pohybují v realitě.

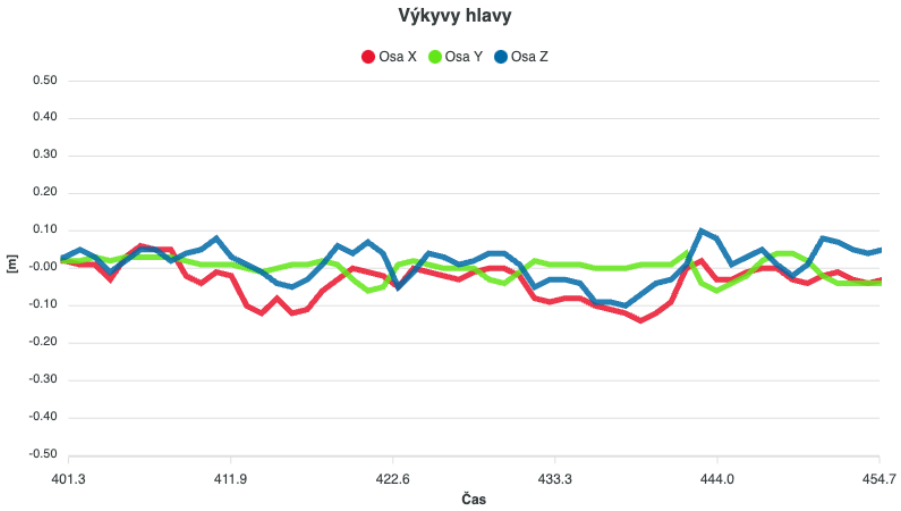
4.1.2.5 Výkyvy hlavy pozice pomocí spojnicového grafu

Tento typ grafu, který je nazván jako `position-differential`, zobrazuje výkyvy, jinak řečeno odchylky od námi daného bodu. Graf v tuto chvíli podporuje dva typy bodů vůči kterým můžeme porovnávat. Jedná se o absolutní bod s hodnotou 0 a o průměrný bod. Průměrný bod se vypočítá jako průměrná hodnota ze všech dostupných záznamů.

Uživatel si může zvolit, jaké osy si chce na grafu zobrazit. Navíc tento typ grafu podporuje i přiblížení a oddálení. Programátor pro tento typ grafů může nadefinovat, v jakých jednotkách jsou hodnoty na svislé ose. Také může nadefinovat doporučenou minimální a maximální hodnotu.

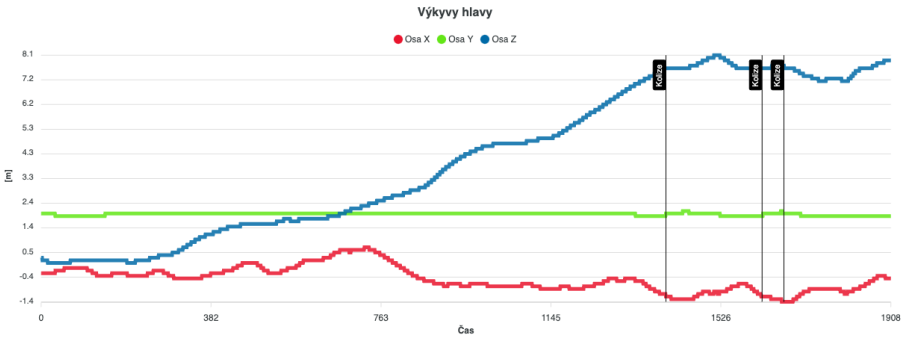
Pokud je maximální a minimální hodnota nadefinovaná, pak graf v případě minimální hodnoty zobrazí tu nejmenší hodnotu z doporučené hodnoty a z minimální hodnoty, která se vyskytuje ve všech záznamech.

Tento typ grafu navíc podporuje zobrazení událostí. V tomto případě se pro daný záznam zobrazí svislá čára s typem události.



Obrázek 4.7. Graf zobrazující výkyvy hlavy v aplikaci PhysioTrails.

Na obrázku 4.7 jsou zobrazeny výkyvy hlavy z aplikace PhysioTrails vůči absolutní hodnotě 0. Z grafů lze vyčíst, že participant mezi záznamy z číslem 401 až 454 měl maximální výkyvy ± 15 centimetrů.



Obrázek 4.8. Graf zobrazující výkyvy hlavy v aplikaci Luna.

Na obrázku 4.8 jsou zobrazeny výkyvy hlavy z aplikace Luna vůči absolutní hodnotě 0. Z daného grafu nelze poznat žádný trend či odvodit správný závěr. Pokud se podíváme na osu Z, tak nám může graf připomínat podobnost grafu `Pozice hlavy`.

Rozdíl mezi aplikacemi je dán tím, že participanti v aplikaci PhysioTrails jsou celou dobu fyzicky na jednom místě. Z toho důvodu nám tento graf opravdu zobrazuje výkyvy od ideální pozice, v tomto případě 0. Oproti tomu participanti v rámci aplikace Luna se pohybují. Vzhledem k tomu nám graf zobrazuje, jak se participanti oddalují či přibližují vůči dané pozici, v tomto případě 0.

S ohledem na to je vhodné tento typ grafu použít pro typ aplikací, kde participanti jsou fyzicky stále na jednom místě. Příkladem může být například balanční aplikace, kde chceme sledovat vychýlení participanta.

4.1.2.6 Přehled vlastností grafů

V následující tabulce 4.1 je u jednotlivých grafů znázorněno, které vlastnosti splňují. `Rotace` znamená, jestli daný graf umožňuje vizualizovat rotaci, a `Pozice` znamená, zda umožňuje vizualizovat pozici. `Volitelné osy` určují, jestli je možné nadefinovat, jaké osy se mohou zobrazit. Pokud tomu tak není, jsou zobrazené osy neměnné a předem určené. `Více os` znamená, že je možné nadefinovat a zobrazit více os než pouze jednu. `Události` určují, zda-li graf podporuje vykreslení událostí. `Jednotky` určují, jestli je

možné nastavit, v jakých jednotkách jsou dané hodnoty. `Min/Max` znázorňuje, zda-li je možné nadefinovat doporučené minimum a maximum hodnot.

Název	Rotace	Pozice	Volitelné osy	Více os	Události	Jednotky	Min/Max
rotation	X		X	X	X	X	
rotation-polar	X		X				
position-heatmap		X				X	
position		X				X	
position-differential		X	X	X	X	X	X

Tabulka 4.1. Vlastnosti implementovaných grafů.

4.1.3 Vícejazyčnost

Klientská aplikace v současné době podporuje český a anglický jazyk. Uživatel si může jazyk kdykoli přepnout. Automaticky se nastaví jazyk podle nastavení prohlížeče. Pokud není jazyk nalezen mezi dostupnými, použije se čeština.

Vícejazyčnost je implementována pomocí balíčku `ngx-translate`. Aplikace čte překlady z JSON souborů, pro každý jazyk jeden soubor, v němž jsou nadefinované klíče a k nim text v příslušném jazyce.

Pro pohodlné vyvíjení je dostupná proměnná `Translations`, jež je nadefinovaná v `TranslateComponent` třídě, která obsahuje všechny klíče zadefinované v JSON souboru hlavního jazyka. Vývojář tak nemusí vždy nahlížet do překladového souboru na dostupné klíče, ale může na ně přistoupit přes tuto proměnnou.

Návod, jak přidat nový jazyk do aplikace, je dostupný v podsekci G.7.1.

4.1.4 WebGL modul

Klientská aplikace podporuje zobrazení WebGL modulu. Funkcemi těchto modulů by měly být vlastní, tj. více specifické statistiky a vizualize, které VR dashboard nepodporuje. Příkladem WebGL modulu může být například vykreslení průchodu mapy tréninkové VR aplikace.

Každý vývojář testovací VR aplikace si tak může napsat vlastní WebGL modul a pokrýt další potřeby, které by uživatelé mohli využít. Vývojář pak vloží vytvořený modul do VR dashboard aplikace a nastaví, v jakých případech se má zobrazovat. Návod, jak je možné WebGL modul vložit do aplikace a nastavit ho, je možné nalézt v sekci E.3.

Zobrazení WebGL modulu v klientské aplikaci

WebGL modul je v klientské aplikaci zobrazen pomocí Iframu. Iframe umožňuje ve webové stránce vložení jiné webové stránky. Ukázka, jak iframe vypadá v aplikaci, je zobrazena na obrázku 4.12. Ukázka kódu, jak je možné v Angularu zobrazit WebGL modul, který je hostovaný na serveru, je zobrazena v příloze C.2

4.1.5 Nastavení VR tréninkových aplikací

VR tréninková aplikace se nastavuje pomocí formátu JSON. Toto řešení bylo vybráno, jelikož je takto nastavení volně rozšiřitelné. Navíc se počítá s tím, že aplikace budou nastavovat vývojáři, kteří by s tímto formátem neměli mít problémy. Navíc, pokud bychom chtěli implementovat formulářové řešení, které bude jednoduše rozšiřitelné i do budoucna, přinášelo by to zbytečnou komplexnost při vývoji.

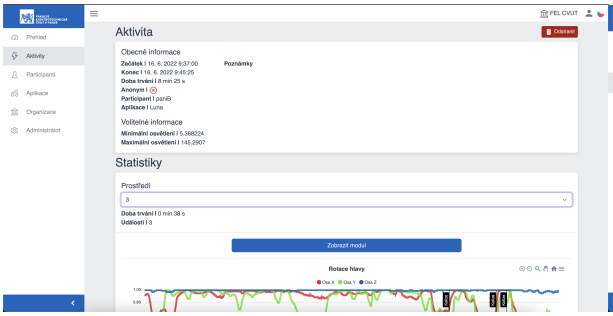
Aby bylo nastavení aplikace pohodlné, je implementován JSON editor pomocí balíčku `Monaco editor`. Tento editor umožňuje přidání validace JSON formátu a případně

vývojáře upozorní, pokud je něco zadané špatně. Návod, jak takové nastavení aplikace vypadá a jak nastavení napsat, je uveden v sekci E.2.

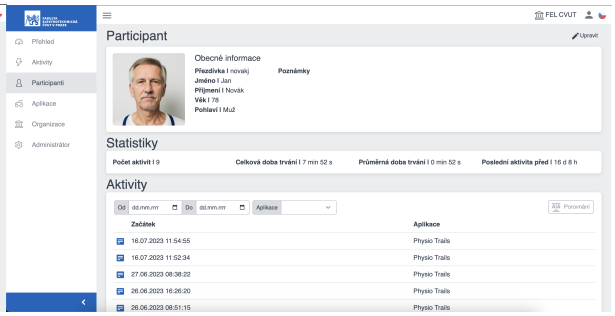
Pro generaci validace JSON formátu je napsán vlastní skript, který tuto validaci generuje z předem nadefinovaných typescript modelů. Návod, jak tuto validaci upravit a vygenerovat, je uveden v podsekci G.7.3.

4.1.6 Finální podoba klientské aplikace

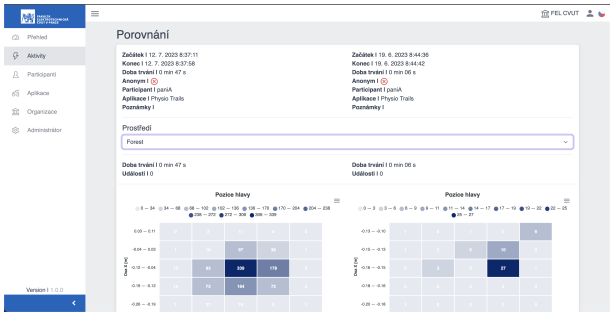
Finální podoba klientské aplikace vychází z vytvořených wireframů uvedených v sekci 3.2. Oproti wireframům některé obrazovky prošly změnou a některé se přidaly až při implementaci, jelikož z hlediska používání to bylo příjemnější. V této podsekci jsou tyto změny popsány. Na obrázku 4.9 jsou zobrazeny tři snímky obrazovek z klientské aplikace ukazující finální design aplikace. V příloze B jsou dostupné snímky všech obrazovek v plné velikosti.



a) Stránka detailu aktivity



b) Stránka detailu participanta



c) Stránka porovnání aktivit

Obrázek 4.9. Ukázka finálního vzhledu aplikace.

Obrazovka Přehled

První změnou je vytvoření nové obrazovky `Přehled`, která je znázorněna na obrázku B.4. Na přehledu se nachází základní statistiky dané organizace, jako je například počet aplikací, zaměstnanců, participantů a aktivit.

Obrazovka Organizace

Další změnou je přidání obrazovek `Organizace` a `Detail organizace`, které jsou zobrazeny na obrázku B.7 a B.8. Na této stránce vidíme seznam zaměstnanců a seznam pozvaných lidí, kteří ještě pozvánku nepřijali. Na této stránce můžeme zvát nové zaměstnance, odstraňovat je, nastavovat jim roli a přiřazovat participanty. Přiřazování participantů probíhá přes modálové okno, kde si vybereme, jaké participanty má daný zaměstnanec vidět.

Obrazovka Administrátor

Třetí změnou je přidání obrazovky `Administrátor`, jejíž snímek je na obrázku B.9. Na tuto obrazovku mají přístup pouze uživatelé s právy `superAdmin` a `developer`. Návod, jak uživateli nastavit tato práva, je uveden v podsekci G.6.1.

Developer zde vidí seznam všech dostupných VR tréninkových aplikací a může vytvořit novou VR tréninkovou aplikaci, která tím pádem bude v rámci celé aplikace dostupná. Super admin kromě seznamu aplikací vidí i seznam všech vytvořených organizací v aplikaci. Super admin tak má práva tyto organizace vytvářet a spravovat všechny její aspekty.

Obrazovka seznam aktivit

Na obrazovce seznamu aktivit bylo přidáno filtrování, pomocí kterého je možné zúžit výběr zobrazených aktivit. Aktuálně aplikace podporuje filtrování podle participanta, aplikace a data vytvoření. Ukázka použití filtru je dostupná na obrázku B.11. Návod, jak filtrování funguje a jak filtr použít, je dostupný v podsekci D.2.2.

4.2 Serverová aplikace

Serverová aplikace je napsána v jazyce typescript v běhovém prostředí Node.js s využitím frameworku express.js.[21] Aplikace podporuje spuštění na více vláknech, což může pomoct s výkonem, pokud máme k dispozici vícevláknové procesory. Tento režim se dá nastavit pomocí proměnné prostředí před spuštěním aplikace.

Aplikace je napsaná jako třívrstvá aplikace.[22] To znamená, že je rozdělena na kontrolery, servisy a DA. Kontrolery slouží na obsluhu HTTP požadavků a ověření práv, zda-li je možné na endpoint přistoupit. V servisách je všechna business logika, která má na starosti kontroly, úpravy dat a volání jednotlivých DA tříd. DA vrstva slouží pro komunikaci s databází. V DA vrstvě tak můžeme nalézt definice databázových dotazů.

Serverová aplikace je dostupná na adrese[1] uvedené v poznámce pod čarou. Zdrojový kód serverové aplikace a návod, jak spustit aplikaci lokálně, jsou dostupné na adrese[2] uvedené v poznámce pod čarou.

4.2.1 Použité balíčky a knihovny třetích stran

Při implementaci serverové aplikace byly využity následující balíčky a knihovny třetích stran. Byly využity z hlediska usnadnění vývoje a zabezpečení aplikace.

- adm-zip[3] - balíček, který usnadňuje rozbalování a zabalování souborů do zipu. Tento balíček je použit pro rozbalení zip souborů obsahující WebGL modul.
- bcrypt[4] - balíček, který slouží pro hashování hesel pomocí bcrypt algoritmu. Tento balíček je použit pro zahashování a ověření hash hesla uživatelů při registraci a přihlášení.
- cors[5] - balíček s express.js middleware, který umožňuje definovat CORS pravidla pro jednotlivé endpointy či celou aplikaci. Balíček je použit pro definování vlastních CORS pravidel.

[1] `https://api.vr-dashboard.leosrehacek.com/`
[2] `https://gitlab.fel.cvut.cz/rehacleo/vr_dashboard_be`
[3] `https://github.com/cthackers/adm-zip`
[4] `https://github.com/kelektiv/node.bcrypt.js`
[5] `https://github.com/expressjs/cors`

- dayjs[1] - balíček, který analyzuje, ověřuje, manipuluje a zobrazuje data a časy. Tento balíček je použit pro ověření data pro prací s datumy ve formátu ISO.
- helmet[2] - balíček, který usnadňuje zabezpečení express.js aplikace pomocí HTTP hlaviček.
- jsonwebtoken[3] - balíček, který implementuje JSON web tokeny. Tento balíček je použit pro vytváření a ověření JSON web tokenů, které se používají pro autorizaci HTTP requestů z klientské aplikace. Toto je popsáno v podsekci 4.2.2.
- lodash[4] - balíček usnadňující práci s objekty, poli a dalšími typy.
- mongodb[5] - oficiální ovladač databáze MongoDB pro Node.js. Tento balíček je použit pro komunikaci aplikace s databází. Komunikace je popsána v podsekci 4.2.4.
- nanoid[6] - balíček pro generování náhodných textových kódů. Tento balíček je používán pro generování kódů jednotlivých organizací a pozvánek.
- swagger-jsdoc[7] - balíček, pomocí kterého lze generovat OpenAPI (swagger) specifikaci z JSDoc anotace.
- swagger-ui-express[8] - balíček umožňující hostovat OpenAPI (swagger) dokumentaci na express.js aplikaci.
- uuid[9] - balíček přináší podporu pro tvorbu RFC4122 UUIDs. Tento balíček je používán pro generování interních Id jednotlivých entit.

4.2.2 Zabezpečení komunikace mezi klientskou a serverovou aplikací

Zabezpečení komunikace probíhá pomocí JSON web tokenu.[23] Při přihlášení uživatele do aplikace se na serveru ověří email a heslo. Pokud je ověření platné, vytvoří se JSON web token s určitou platností, unikátním podpisem a payloadem. Payload jsou data, která můžeme v tokenu přenášet. V našem případě se jedná například o informace o uživateli. Poté, co je token vytvořen, pošle se jako odpověď přihlášení do klientské aplikace. Klientská aplikace si tento token uloží. Při každém HTTP requestu na endpoint, který je zabezpečený, klientská aplikace do hlavičky zabalí uložený token. Jakmile přijde HTTP request, na serveru se ověří, zda-li je token podepsaný stejným podpisem, jako byl vydán, jestli token již neexpiroval a zda data uvnitř odpovídají některému uživateli. Pokud jsou všechny tyto náležitosti splněny, server příjme HTTP požadavek.

4.2.3 Ukládání a hostování WebGL modulů

Při implementaci bylo potřeba vyřešit, jak ukládat WebGL moduly jednotlivých VR aplikací. Původní myšlenkou bylo ukládat tyto moduly do databáze. Bohužel toto řešení se nepodařilo implementovat tak, aby bylo 100% funkční. Konečné funkční řešení je ukládání WebGL modulu na server, na němž běží serverová aplikace. Aby tyto moduly byly paměťově optimální, serverová aplikace vyžaduje, aby WebGL moduly byly komprimované. Pokud nebude modul komprimovaný, tak se správně nenačte v klientské aplikaci.

[1] https://day.js.org/en/
[2] https://helmetjs.github.io/
[3] https://github.com/auth0/node-jsonwebtoken
[4] https://lodash.com/
[5] https://github.com/mongodb/node-mongodb-native
[6] https://github.com/ai/nanoid
[7] https://github.com/Surnet/swagger-jsdoc
[8] https://github.com/scottie1984/swagger-ui-express
[9] https://github.com/uuidjs/uuid

Hostování WebGL modulů je vyřešeno pomocí vestavěného express.js middlewaru pro obsluhu statických souborů. Statické soubory jsou dostupné na námi specifikované cestě. Jelikož aplikace umožňuje mít více VR aplikací a více verzí WebGL modulů, musí cesta obsahovat i tyto parametry. Následně se z parametrů složí cesta, kde jsou soubory uložené na serveru, která se následně předá express.js, aby věděl, kde tyto soubory najít. Ukázka, jak je toto hostování implementováno, je v příloze C.3.

4.2.4 Komunikace a správa databáze

Serverová aplikace pro komunikaci s MongoDB databází používá oficiální balíček mongodb. Pro připojení k databázi je potřeba definovat hodnotu `DB_URL` proměnné prostředí. Hodnota této proměnné je MongoDB URI.

Schéma databáze a jednotlivých kolekcí odpovídá doménovému modelu, který je popsán v podsekci 3.1.3. V rámci serverové aplikace se jednotlivé dokumenty definují podle modelu a podle insert DA metod.

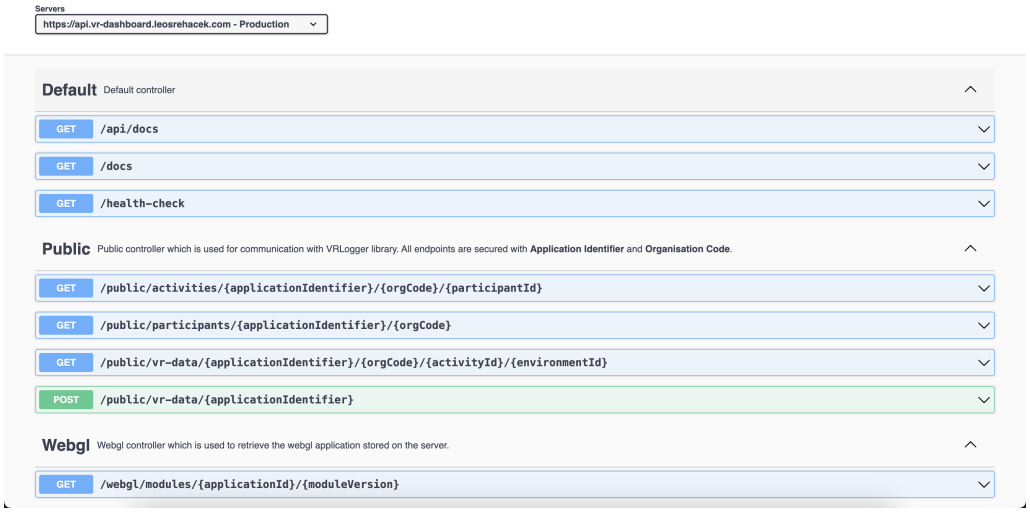
V rámci serverové aplikace jsou nadefinované i indexy jednotlivých kolekcí. Indexy určují podle čeho má databáze řadit jednotlivé záznamy. Pomocí indexů můžeme zrychlit jednotlivé databázové operace jako je například čtení.

4.2.5 Swagger dokumentace

Serverová aplikace obsahuje nadefinovaný swagger. Swagger slouží jako rozhraní pro provolání či otestování REST API endpointů. Navíc swagger ve většině případů slouží jako dokumentace těchto endpointů. Příklad, jak takové rozhraní swaggeru vypadá, je zobrazen na obrázku 4.10.

V rámci serverové aplikace jsou pomocí swaggeru nadefinovány takové endpointy, které slouží pro komunikaci se serverem bez použití JSON web tokenu. Jedná se o ty samé endpointy, které používá VR dashboard logger pro komunikaci se serverem.

Swagger je dostupný na odkazu[1] uvedeném v poznámce pod čarou. Návod, jak upravit swagger dokumentaci, je popsán v podsekci G.8.4.



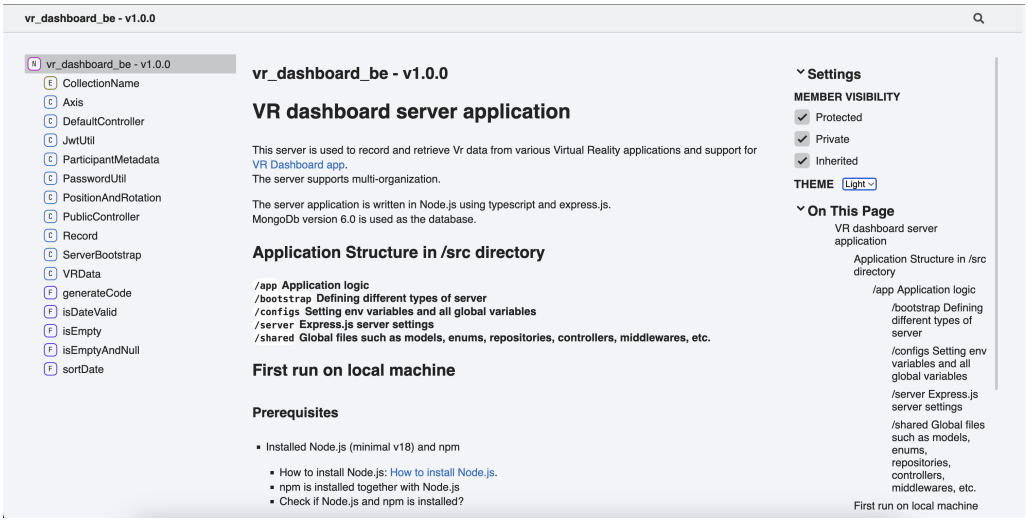
Obrázek 4.10. Rozhraní swaggeru aplikace.

[1] `https://api.vr-dashboard.leosrehacek.com/api/docs/`

4.2.6 JSDoc dokumentace

Serverová aplikace také obsahuje implementaci JSDoc dokumentace. Tato dokumentace slouží k popisu jednotlivých tříd a metod v rámci kódu aplikace. JSDoc navíc z běžných komentářů v kódu dokáže vygenerovat přehledné HTML stránky. Příklad, jak taková HTML stránka vypadá, je zobrazen na obrázku 4.11.

JSDoc dokumentace je dostupná na adrese[1] uvedené v poznámce pod čarou.



Obrázek 4.11. Příklad JSDoc HTML stránky.

4.3 Podpůrný Unity balíček (VR dashboard logger)

Podpůrný balíček je Unity balíček napsaný pro vývojáře VR testovacích aplikací. Tento balíček má usnadňovat vývojářům implementaci VR dashboardu do jejich aplikací.

V balíčku jsou metody pro inicializaci loggeru a správu zaznamenávání dat a aktivity. Jedná se například o nastavení organizace, participanta, volitelných informací, zaznamenání události a další. Poté, co je zaznamenávání dokončeno, je umožněno uložení aktivity na serveru či do lokálního souboru s převodem do správného JSON formátu.

Kromě zaznamenávání umožňuje balíček ze serveru získat všechny participanty dané organizace. Dále má balíček podporu pro WebGL modul, kdy pomáhá parsovat vstupní parametry modulu a na základě nich získat ze serveru data o požadované aktivitě.

Návod, jak tento balíček zapojit do VR tréninkové, je popsán v sekci G.4.

Zdrojový kód podpůrného unity balíčku je dostupný na adrese[2] uvedené v poznámce pod čarou. Pomocí tohoto odkazu je možné balíček přidat přes `Package manager` do Unity projektu.

4.3.1 Použité balíčky a knihovny třetích stran

- Newtonsoft.Json[3] - framework pro serializování a deserializování JSON objektů do .NET objektů. Tento balíček je používán pro deserializaci HTTP odpovědí ze serverové aplikace a pro serializaci .NET objektů do JSON stringu pro odeslání HTTP požadavku.

[1] `https://api.vr-dashboard.leosrehacek.com/docs/`
[2] `https://gitlab.fel.cvut.cz/rehacleo/vr_dashboard_logger`
[3] `https://www.newtonsoft.com/json`

4.3.2 Nahrání aktivity do aplikace

Poté, co je pomocí VR dashboard loggeru zaznamenána aktivita, jejíž zaznamenání je ukončeno zavoláním metody `StopLogging()`, je možné tuto aktivitu nahrát na server. Toto nahrání je možné provést pomocí zavolání metody `SendActivity()`.

Metoda `SendActivity()` má dva vstupní atributy. První je anonymní funkce, která slouží pro určení úspěšnosti nahrání aktivity na server. Druhá je volitelný parametr, zda-li chceme aktivitu také uložit lokálně ve formátu JSON.

Pokud je anonymní funkce zavolána s parametrem, který obsahuje hodnotu `true`, je aktivita úspěšně nahrána do VR dashboard aplikace. Pokud hodnota obsahuje `false`, pak je možné tuto aktivitu manuálně nahrát do aplikace, jestliže jsme si zvolili lokální uložení.

Návod, jak je možné manuálně importovat danou aktivitu uloženou ve formátu JSON, je popsán v podsekci D.2.1. Více informací, jaké metody balíček obsahuje a k čemu dané metody jsou, jsou popsány v `README.md` souboru, který je dostupný v repozitáři balíčku nebo na odkazu[1] uvedeném v poznámce pod čarou.

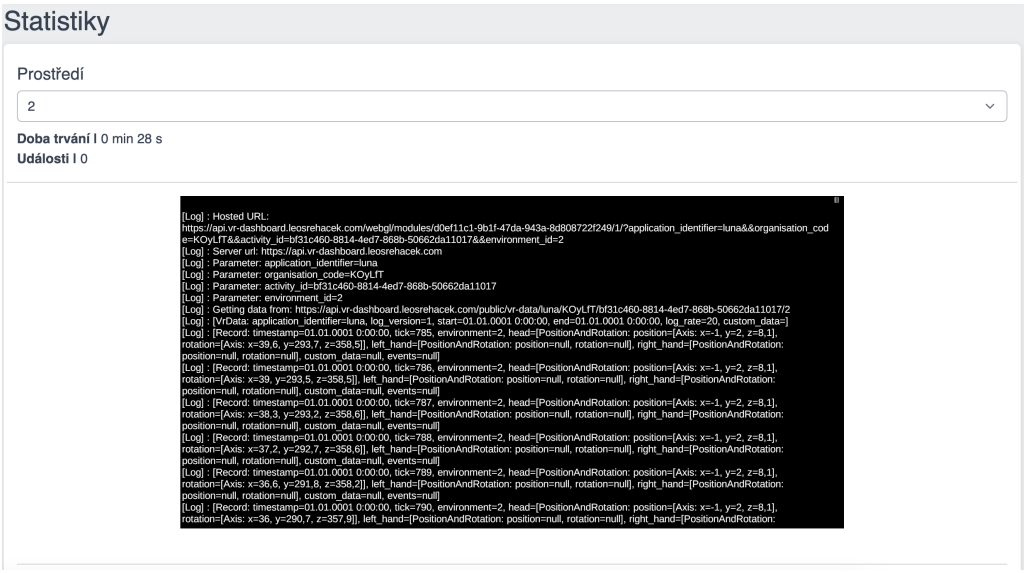
4.4 Ukázkový WebGL modul v Unity

Ukázkový WebGL modul využívá pro získávání VR dat výše zmíněný VR dashboard logger. Funkcí tohoto modulu je získaná data zobrazit na obrazovce jako konzolový výpis. Tento modul přijímá potřebná data pro získání VR dat pomocí URL query parametrů. Umožňuje nadefinování i určitého environmentu, aby nebylo nutné filtrovat data přímo v modulu, ale provádělo se toto filtrování na serveru.

Kromě ukázky použití VR dashboard loggeru pro získání dat má projekt nadefinované nastavení pro sestavení WebGL modulu, které podporuje klientská i serverová aplikace. To by mělo umožnit budoucím vývojářům snadnější implementaci řešení a nastavení WebGL modulu v aplikaci.

Návod, jak vytvořit takový WebGL modul, je popsán v sekci G.5.

Zdrojový kód ukázkového projektu WebGL modulu je dostupný na adrese[2] uvedené v poznámce pod čarou.



Obrázek 4.12. Ukázka WebGL modulu v klientské aplikaci.

[1] `https://gitlab.fel.cvut.cz/rehacleo/vr_dashboard_logger/-/blob/main/README.md`
[2] `https://gitlab.fel.cvut.cz/rehacleo/vr_dashboard_webgl_module_template`

4.5 Propojení jednotlivých částí VR dashboard aplikace

VR dashboard aplikace se skládá ze tří hlavních částí a dvou podpůrných částí. Mezi hlavní části patří klientská aplikace, serverová aplikace a MongoDB databáze. Mezi podpůrné části patří Unity balíček a ukázkový Unity projekt s WebGL modulem.

Klientská aplikace komunikuje se serverovou aplikací pomocí HTTP požadavků. Veškerá potřebná komunikace je zabezpečená pomocí JSON web tokenu. Podrobnější informace o zabezpečení komunikace mezi klientem se serverem jsou uvedeny v podsekci 4.2.2.

Serverová aplikace většinu dat ukládá do MongoDB databáze. Jediné, co je uloženo na lokálním stroji, kde serverová aplikace běží, jsou WebGL moduly. Serverová aplikace s databází komunikuje pomocí oficiálního mongodb balíčku.

VR tréninkové aplikace pro komunikaci se serverovou aplikací mohou využívat vlastní implementaci nebo použít vytvořený Unity balíček s názvem VR dashboard logger. VR dashboard logger se serverovou aplikací komunikuje pomocí HTTP požadavků. Jednotlivé endpointy, které logger využívá, jsou dostupné ve swaggeru, o kterém je zmínka v podkapitole 4.2.5. Tyto endpointy jsou zabezpečené pomocí identifikátoru aplikace a kódu organizace, pro které chceme získat nebo zapsat data.

Pro svoji funkci potřebuje WebGL modul získat ze serverové aplikace záznamy dané aktivity, pro kterou má zobrazit statistiky či vizualizace. Pro tyto potřeby je doporučeno využít VR dashboard logger, který má implementované podpůrné metody pro WebGL modul. Případně je možné vytvořit vlastní implementaci, která bude se serverovou aplikací komunikovat přes HTTP požadavky.

Kapitola 5
Testování

V této kapitole jsou popsané dva druhy testování, které byly provedeny s implementovanou aplikací. V první řadě proběhlo uživatelské testování klientské aplikace, která pro své fungování využívá i serverovou aplikaci. V druhé řadě byla provedena implementace VR dasboard loggeru, která pro své fungování využívá serverovou aplikaci.

Dá se tak říci, že v rámci testování byly otestovány všechny součásti VR dashboard aplikace. Ukázkový WebGL modul byl otestován při implementaci tohoto řešení.

5.1 Uživatelské testování

Cílem uživatelského testování je ověřit použitelnost a uživatelskou přívětivost klientské aplikace.[24] Testování použitelnosti se provádělo na kompletní aplikaci formou vzdáleného synchronního testování, které má srovnatelné výsledky jako běžné laboratorní testování.

Tato forma testování spočívá v tom, že participant a hodnotitel jsou každý na jiném místě. Při testování spolu komunikují přes webkameru a participant sdílí svojí obrazovku, aby hodnotitel viděl participantovo chování a úkony. Participant se snaží provést úkony zadané v testovacím scénáři a hodnotitel si zapisuje objevené problémy.

5.1.1 Participanti uživatelského testování

Uživatelské testování bylo provedeno se čtyřmi participanty. Tři participanti mají vysokoškolské vzdělání a jeden je aktuálně na bakalářském studiu. Polovina participantů jsou ženy a polovina muži. Participanti byli vybráni tak, aby pokryli všechny možné uživatelské role dostupné v aplikaci.

Participanti měli přiřazené různé role. Někteří byli běžní uživatelé, jiní byli developeři či super administrátoři. Podle role se odlišovaly scénáře, které participant plnil. Běžný uživatel plnil scénáře od *TS1* až do scénáře *TS7*. Developer navíc plnil scénář *TS8*. Super admin procházel stejné scénáře jako developer se scénářem *TS9* navíc.

Jeden participant byl otestován jako běžný uživatel. Dva participanti, kteří mají zkušenosti s vývojem VR tréninkových aplikací, byli otestováni jako developeři. Jeden participant, který má zkušenosti s vývojem a provozováním webových aplikací, byl otestován jako super admin.

5.1.2 Testovací scénáře uživatelského testování

V této sekci jsou uvedeny testovací scénáře, které byly použity při uživatelském testování. Jednotlivé scénáře se snaží zachytit, jak budou s aplikací uživatelé zacházet. Pořadí testovacích scénářů je záměrně seřazeno podle běžných procesů a podle uživatelských rolí.

TS1: Úvod do testování

Máte VR aplikaci, která má pomoci vašim zákazníkům. Abyste věděli, že je aplikace funkční, potřebujete své uživatele sledovat a vyhodnocovat, jak si vedou. Před Vámi je aplikace, která tyto požadavky splňuje.

TS2: Vytvoření účtu a organizace

Abyste mohli aplikaci používat, je potřeba vytvořit si v aplikaci účet. Po vytvoření účtu zaregistrujte svoji organizaci s libovolným názvem, která pomáhá Vašim zákazníkům.

TS3: Přidání participanta

Ve své organizaci si chcete přiřadit 2 zákazníky, kterým budeme říkat participanti. První participant se jmenuje Jan Novák, narozený 14.5.1972. Jan Novák je velice aktivní sportovec, který každý den jezdí na kole. Druhým participantem je Marie Dvořáková, která ale nikde nechce uvádět své jméno a ani další informace.

TS4: Nahrání aktivity

V přiloženém souboru pod názvem `activity.json` máte záznam aktivity, kterou prováděl participant Jan Novák. Tato konkrétní aktivita byla pořízena v aplikaci Luna. Jan Novák Vám po skončení aktivity sdělil své pocity: „Aktivita pro mě byla ze začátku příliš těžká, jelikož jsem se nemohl v prostoru orientovat. Ale myslím si, že úroveň po úrovni jsem se zlepšoval.“ Nahrajte předmětnou aktivitu do aplikace.

TS5: Zobrazení detailu aktivity

Poté, co jste nahráli aktivitu, Vás zajímají statistiky. Podívejte se na Vámi vytvořenou aktivitu a do poznámek si napište Váš názor, jak si participant vedl.

TS6: Porovnání aktivit

Kamarád Vás pozval do své organizace s cílem, abyste mu pomohli s porovnáním aktivit. Mělo by se jednat o poslední 3 aktivity s aplikací Physio Trails a participantem s přezdívkou paní A. Přijměte tedy pozvánku s kódem `XXXX` a porovnejte dané aktivity.

TS7: Úprava participanta

Marie Dvořáková se rozhodla, že Vám poskytne souhlas s uložením osobních údajů a dokonce Vám poskytla i svoji fotku uloženou v příloze pod názvem souboru `profile.jpeg`. Marie se narodila 27.10.1990. Upravte u daného participanta nyní poskytnuté údaje.

TS8: Přidání vlastní aplikace

Jelikož jste také vývojář své VR tréninkové aplikace, získali jste od správce práva pro vytvoření aplikace a nadefinování jejího nastavení. Pro VR aplikaci jste také vytvořili aplikační WebGL modul (dostupný v příloze pod názvem souboru `module.zip`), který má verzi `1.0.0` kompatibilní s verzí logů `1.0.0`. V aplikaci zaregistrujte svojí VR tréninkovou aplikaci, aby jí bylo možné v aplikaci používat. Aplikace loguje volitelná data v cestě `light` a eventu `collision`. Pro vykreslení záznamů je vhodné použít grafy `rotation`, `rotation-polar` a `position`. Ve všech třech případech chceme znázorňovat hlavu participanta.

TS9: Vytvoření a správa organizace

Od správce aplikace jste dostali práva super admin, která Vám umožňují vytvářet a spravovat nové organizace. Na základě toho Vás správce požádal, zda-li byste mohli vytvořit novou organizaci, jejíž jméno bude Vaše příjmení. Poté do nové organizace pozvěte nového zaměstnance s emailem pavel.novak@email.cz, kterému nastavíte práva `ADMIN`. Následně si uložte kód pozvánky, abyste ji mohli novému zaměstnanci poslat.

5.1.3 Závěr uživatelského testování

Celkově byly provedeny čtyři uživatelské testování, pokaždé s jedním participantem. Testování bylo provedeno online pomocí video schůzky za využití Google Meets a Discord, kdy každý participant sdílel svou obrazovku. Uživatelské testování bylo nahráváno pro podrobnější analýzu a upřesnění zápisků, které byly během testování zapisovány. Testování trvalo v rozmezí od 30 do 60 minut podle zkušeností a zadané role participanta. Na základě provedených testování vznikly následující závěry.

Všichni participanti splnili zadané testovací scénáře. Se základními testovacími scénáři (*TS1* - *TS7*) nikdo z participantů neměl problém. Participanti se bez problému orientovali v aplikaci a bez větších obtíží vždy dokázali nalézt to, co hledali. Participantům se aplikace líbí, podle jejich slov byla přehledná. Všichni participanti byli trochu zmatení, jelikož u několika tlačítek nebyl vhodně zvolený popisek. Jednalo se například o tlačítko s popiskem „Upravit“, které ve skutečnosti znamenalo uložit. Dále bylo z participantů poznat, že byly trochu zmatení, když klikli na tlačítko a nedostali žádnou zpětnou odpověď jako například přesměrování uživatele nebo zobrazení potvrzující hlášky. Všechny tyto neduhy byly po provedení testování upraveny.

Při testovacím scénáři *TS6* všem participantům nejvíce vadilo, že pokud při porovnání pro dané prostředí nejsou žádná data, tak tato záležitost není explicitně zmíněna a musí to poznat například z nulové délky trvání daného prostředí.

Největší problém měli 2 ze 3 participantů s testovacím scénářem *TS8*. Při tomto scénáři bylo vidět, že pokud mají participanti již zkušenosti s JSON formátem, napovídání v Monaco editoru jim stačilo, aby dokázali danou aplikaci nadefinovat. Ti dva participanti, kteří tuto zkušenost neměli, se museli plně spolehnout na přiložený manuál.

Při uživatelském testování se narazilo na několik drobností a chyb, které byly po testování opraveny či změněny. Zbylé změny a vylepšení, které se nestihly opravit nebo nejsou tak důležité, jsou uvedeny v sekci 6.1. Zde je uveden výčet chyb a změn, které byly po testování provedeny:

- Uživatelský manuál byl rozšířen a rozdělen podle typu role. Nyní je v manuálu daleko lépe popsáno nastavení VR aplikace.
- V detailu aplikací byl přidán přehled již přidaných aplikačních modulů, pokud je již nějaký přidaný.
- Bylo přidáno zablokování formuláře, pokud klientská aplikace nedostane odpověď ze serveru.
- Na detailu aktivit či v porovnání aktivit bylo přidáno upozornění, že pro dané prostředí neexistují žádná data.
- Na ikonu `detail` byla přidána nápověda, co daná ikona znamená nebo dělá za akci.
- Popisky u formulářových polí byly rozšířeny o příklady. Navíc byla pro kritická pole přidána striktní pravidla pro povolené hodnoty.
- Byly upraveny překlady tlačítek, aby lépe popisovaly, co daná tlačítka dělají.
- Byly přidány hlášky, které popisují, zda-li daná operace skončila úspěšně, či chybou.

5.2 Implementace aplikace do VR tréninkových aplikací

Aplikace byla implementována v rámci dvou VR tréninkových aplikací. Jednalo se o aplikace Luna a PhysioTrails, které byly představeny v sekci 2.1]. Implementaci prováděli správci jednotlivých VR aplikací s možností mé konzultace. Správci využili vytvořený VR dashboard logger Unity balíček, který je popsaný v sekci 4.3.

Při implementaci správci využívali své uživatelské účty a organizace v rámci VR dashboard aplikace. Provedli pár testovacích pokusů, kdy prošli svými VR aplikacemi a sledovali, zda-li se jednotlivé aktivity zobrazí ve VR dashboardu.

Kromě samotného ukládání aktivit implementovali i ostatní funkcionality, které Unity balíček umožňuje. Jedná se například o získání všech participantů, kteří jsou pro danou organizaci vytvořeni.

Každá VR tréninková aplikace má jiné požadavky na funkcionality. Jedna například obsahuje více prostředí a druhá pouze jedno. V rámci těchto dvou VR tréninkových aplikací tak bylo pokryto otestování všech metod, které VR dashboard logger umožňuje.

5.2.1 Závěr z implementace

V této sekci jsou popsány závěry ze zkušební instalace VR dashboardu do VR tréninkových aplikací. Při testování implementace se oběma vývojářům podařilo bez problému implementovat Unity balíček a následně nahrát aktivity do aplikace. Ukázka implementace, která je dostupná v repozitáři balíčku, a komentáře v kódu balíčku vývojářům plnohodnotně stačila a konzultace využili jen velmi zřídka. Participanti sdělili, že implementace Unity balíčku do VR tréninkové aplikace byla jednoduchá.

Při testování bylo posbíráno několik vylepšení, která byla následně do balíčku přidána. Ta vylepšení, která se nestihla implementovat nebo nejsou kritická, jsou popsána v sekci 6.1. Zde je výčet implementovaných změn po testování:

- VR dashboard logger byl změněn na singleton, aby ho bylo možné importovat kdekoli v rámci VR tréninkové aplikace.
- Byly upraveny a přidány konzolové záznamy, které nyní obsahují prefix, aby bylo jasné, že záznamy jsou z balíčku.
- Přidání možnosti, zda-li chceme sledovat globální, či lokální pozici a rotaci.

Kapitola 6
Závěr

V úvodu této práce jsem si stanovil cíl, a to navržení a implementaci systému pro sběr a analýzu dat z VR tréninkových aplikací.

Aby mohl být tento cíl splněn, bylo nutné vycházet z již existujících VR aplikací, které jsem za tímto účelem stručně popsal. Základní informace o třech konkrétních VR aplikacích byly nutným východiskem pro následná určení jednotlivých znaků a požadavků navrhovaného systému. Dále jsem se seznámil s různými způsoby ukládání záznamových dat a již existujícími řešeními, které se zabývají podobnou či stejnou problematikou. Tyto poznatky mi sloužily jako inspirace při následném navrhování systému.

Vlastní řešení navrhovaného systému je obsaženo ve třetí kapitole, která je důležitým stavebním kamenem pro následnou implementaci systému. V této kapitole jsem se zabýval výběrem technologií pro systém. Důvody pro zvolení jednotlivých technologií nespočívají toliko na jejich objektivných vlastnostech, ale též na mých vlastních zkušenostech z praxe. Jako databáze se pro tyto účely použila MongoDB, server aplikace běží v Node.js s frameworkem Express.js a klientská aplikace je založena na Angularu. Součástí návrhu systému pak bylo též navržení schématu databáze a určení základních use cases aplikace. Dále jsou v této kapitole znázorněny wireframy zobrazující možnou podobu částí klientské aplikace navrhovaného systému. Posledně je v této kapitole navrhnut a kategorizován formát záznamových dat na jehož základě jsou navrhnuty vhodné vizualizační a statistické metody.

Ve čtvrté kapitole je popsaná samotná implementace aplikace spolu s podpůrnými balíčky či ukázkovými projekty. Celému tomuto řešení jsem přiřadil a začal používat název VR dashboard. V implementaci jsou popsána specifika a unikátnosti jednotlivých částí aplikace. U každé aplikace jsou tak popsány její principy a problémy s odkazy do jednotlivých manuálů, které bylo potřeba při implementaci vyřešit. V závěru je popsáno propojení jednotlivých částí, které by mělo nastínit fungování systému jako celku.

Následuje kapitola popisující testování implementovaného systému. Sytém byl testován dvěma způsoby, a to pomocí uživatelského testování a začlenění systému do již existujících řešení. Součástí uživatelského testování je taktéž definice testovacích scénářů, které byly při testování použity. U každého typu testování jsou také představeni participanti testování a též uvedeny jejich závěry, ze kterých vzešlo, že systém je navržen a implementován úspěšně, tzn. že je možné ho používat bez větších problémů. Kromě toho ze závěrů testování vznikly podněty na drobná zlepšení, která byla buď opravena, nebo jsou uvedena v úplném závěru této práce pro budoucí rozvoj aplikace.

S ohledem na výše uvedené se domnívám, že návrh a implementace systému pro sběr a analýzu dat z VR tréninkových aplikací byl úspěšně uskutečněn, a tedy cíl práce byl splněn.

6.1 Vhodné změny a úpravy v budoucí verzi systému

V následující sekci jsou uvedeny nápady či zlepšení, které by bylo vhodné a přínosné implementovat v budoucích verzích VR dashboard aplikace.

Jedná se o zlepšení, která nebyla v rámci zadání požadována, ale která by mohla usnadnit či vylepšit uživatelský komfort při používání aplikace. Bodově se jedná o tyto návrhy:

- Možnost definovat uživatelům práva `superAdmin` a `developer` přímo v aplikaci.
- Implementace manuálu pro nastavení VR aplikace přímo v aplikaci. Napovídání v Monaco editoru není pro všechny uživatele ideální a pro uživatele je pohodlnější, pokud návod či nápovědu najdou přímo v aplikaci, ne v uživatelském manuálu.
- Přidat aktuální filtry aktivit do url, aby bylo možné sdílet nastavené filtry.
- Přidat možnost na odstranění verze WebGL modulu.
- Přidat možnost odstranit organizaci přímo v aplikaci.
- Přidat podporu nohou či jiných částí těla v rámci celé aplikace, jak klientské, serverové, tak loggeru.

Literatura

[1] Min-Chai Hsieh a Jia-Jin Lee. Preliminary Study of VR and AR Applications in Medical and Healthcare Education. *Journal of Nursing and Health*. 2018, 3 DOI 10.21767/2574-2825.100030.

[2] Ellysse Dick. *Balancing user privacy and innovation in Augmented and Virtual Reality*. Information Technology and Innovation Foundation. 2021. `https://itif.org/publications/2021/03/04/balancing-user-privacy-and-innovation-augmented-and-virtual-reality/`.

[3] Mark Roman Miller, Hanseul Jun a Jeremy N. Bailenson. *Motion and Meaning: Sample-Level Nonlinear Analyses of Virtual Reality Tracking Data.* In: *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. 2021. 147-152.

[4] Julius Pettersson a Petter Falkman. Human Movement Direction Classification using Virtual Reality and Eye Tracking. *Procedia Manufacturing*. 2020, 51 95-102. DOI https://doi.org/10.1016/j.promfg.2020.10.015. 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021).

[5] Arfan Sharif. *What is log management? The importance of logging and best practices.* 2022. `https://www.crowdstrike.com/cybersecurity-101/observability/log-management`. Navštíveno: prosinec 2023.

[6] Gonzalo Salgueiro, Vijay K. Gurbani a Adam Roach. *Format for the Session Initiation Protocol (SIP) Common Log Format (CLF)*. RFC 6873. Request for Comments. 2013. `https://www.rfc-editor.org/info/rfc6873`.

[7] Jiyoung Moon, Minho Jeong, Sangmin Oh, Teemu H. Laine a Jungryul Seo. Data Collection Framework for context-aware virtual reality application development in unity: Case of avatar embodiment. *Sensors*. 2022, 22 (12), 4623. DOI 10.3390/s22124623.

[8] Inc. Better Stack. *The fastest log search on the planet*. n.d. `https://betterstack.com/logs`. Navštíveno: prosinec 2023.

[9] Grafana Labs. *Grafana*. n.d. `https://grafana.com/grafana/`. Navštíveno: prosinec 2023.

[10] Elasticsearch B.V. *Kibana Guide*. n.d. `https://www.elastic.co/guide/en/kibana/current/index.html`. Navštíveno: prosinec 2023.

[11] IBM s.r.o. *What is three-tier architecture?* n.d. `https://www.ibm.com/topics/three-tier-architecture`. Navštíveno: leden 2023.

[12] MongoDB Inc. *What is MEAN Stack?* n.d. `https://www.mongodb.com/mean-stack`. Navštíveno: leden 2023.

[13] Dayne Hammes, Hiram Medero a Harrison Mitchell. *Comparison of NoSQL and SQL Databases in the Cloud.* In: *SAIS 2014 Proceedings*. 2014. https://aisel.aisnet.org/sais2014/12.

[14] S. Bradshaw, E. Brazil a K. Chodorow. *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. O'Reilly Media, 2019. ISBN 9781491954416.

[15] OpenJS Foundation. *About Node.js*. n.d. https://nodejs.org/en/about/. Navštíveno: leden 2023.

[16] A. Mardan. *Pro Express.js: Master Express.js: The Node.js Framework For Your Web Development*. Apress, 2014. ISBN 9781484200377.

[17] Rishi Vyas. Comparative Analysis on Front-End Frameworks for Web Applications. *International Journal for Research in Applied Science and Engineering Technology*. 2022, 298–307. DOI 10.22214/ijraset.2022.45260.

[18] D. Uluca. *Angular for Enterprise-Ready Web Applications: Build and deliver production-grade and cloud-scale evergreen web apps with Angular 9 and beyond, 2nd Edition*. Packt Publishing, 2020. ISBN 9781838646608.

[19] K. Bittner, I. Spence a I. Jacobson. *Use Case Modeling*. Addison Wesley, 2003. ISBN 9780201709131.

[20] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann a Christian Tominski. *Visualization of time-oriented data*. ISBN 9780857290793.

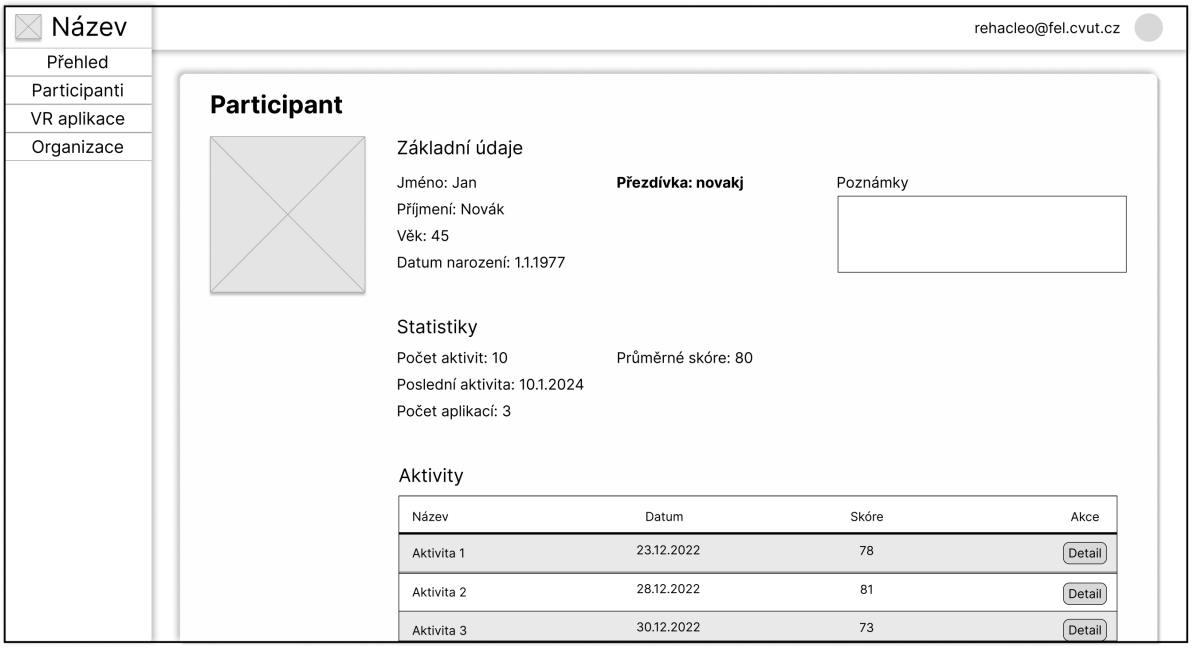
[21] J. Wexler. *Get Programming with Node.js*. Manning, 2019. ISBN 9781638352402.

[22] M. Richards. *Software Architecture Patterns: Understanding Common Architecture Patterns and when to Use Them*. O'Reilly Media, 2015. ISBN 9781491924242.

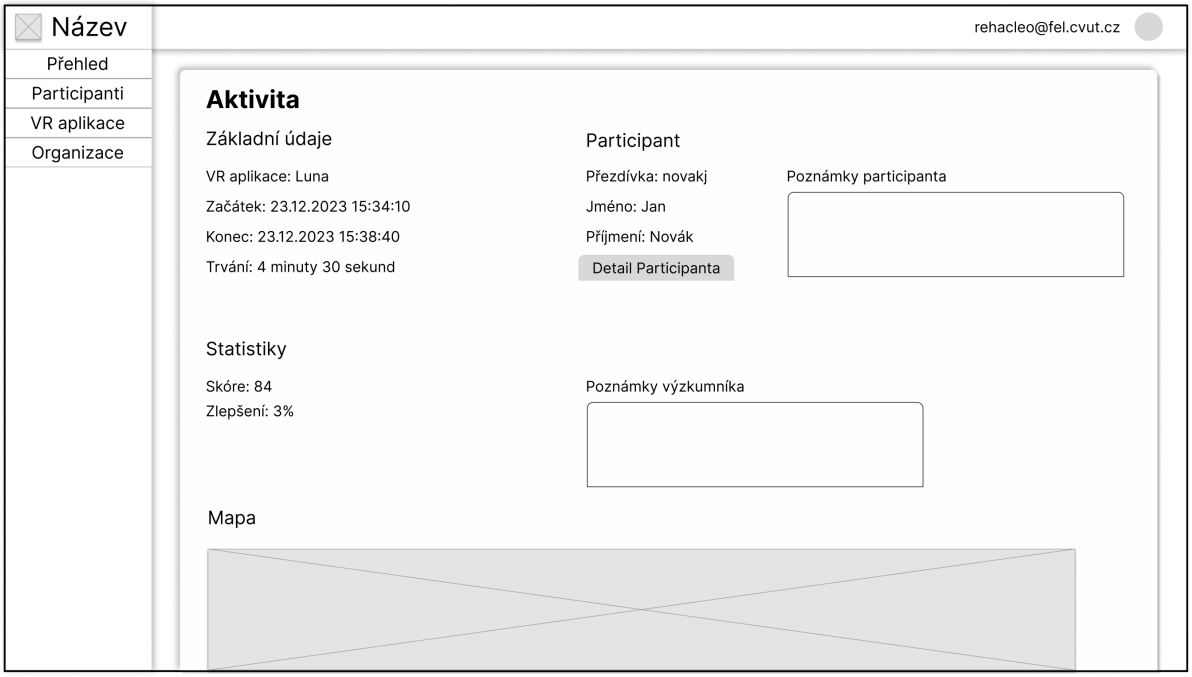
[23] Salman Ahmed a Qamar Mahmood. *An authentication based scheme for applications using JSON web token.* In: *2019 22nd International Multitopic Conference (INMIC)*. 2019. 1-6. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9022766.

[24] Morten Sieker Andreasen, Henrik Villemann Nielsen, Simon Ormholt Schrøder a Jan Stage. *What Happened to Remote Usability Testing? An Empirical Study of Three Methods.* In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2007. 1405–1414. ISBN 9781595935939. https://doi.org/10.1145/1240624.1240838.

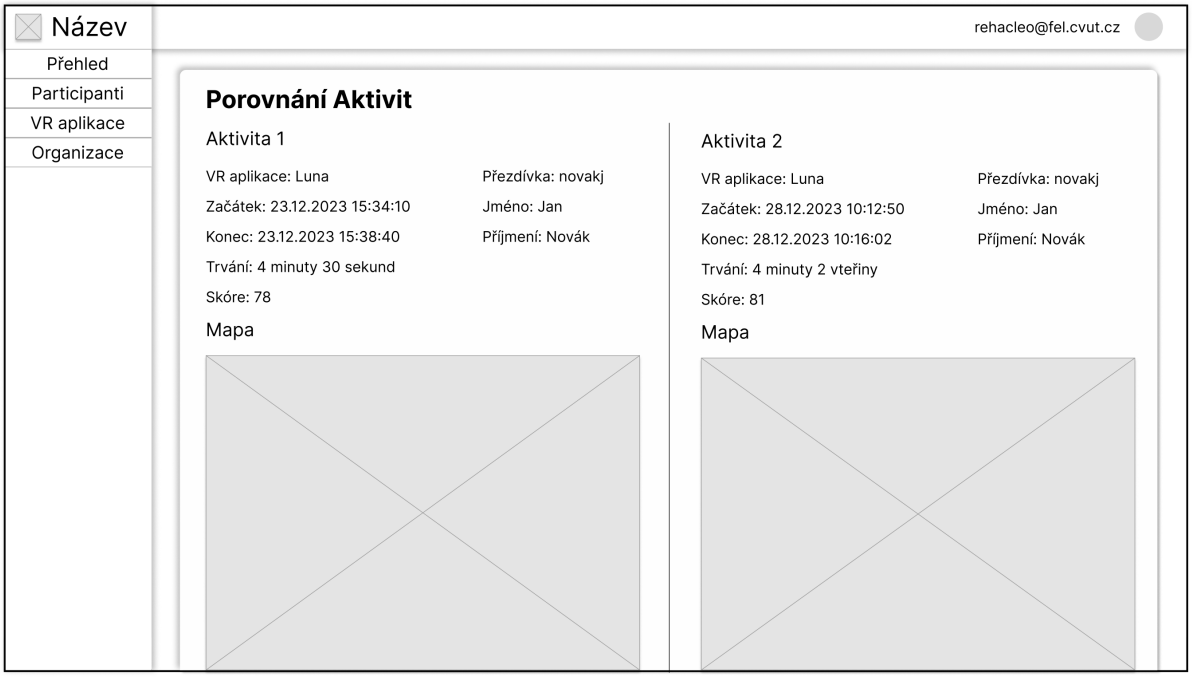
Příloha A
Wireframy klientské aplikace



Obrázek A.1. Wireframe obrazovky detailu participanta.

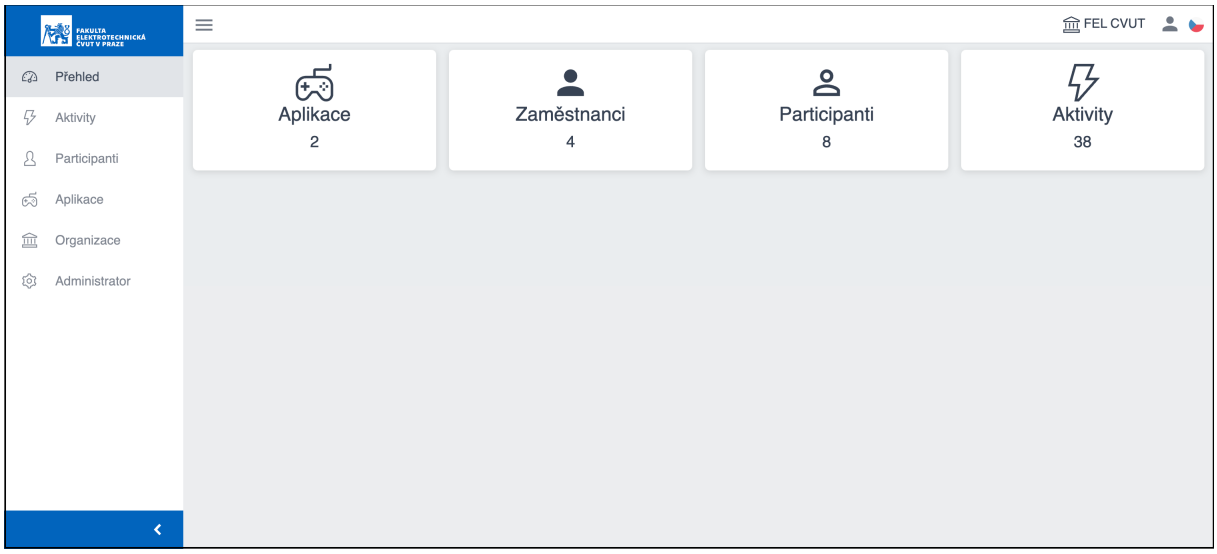


Obrázek A.2. Wireframe obrazovky detailu aktivity.

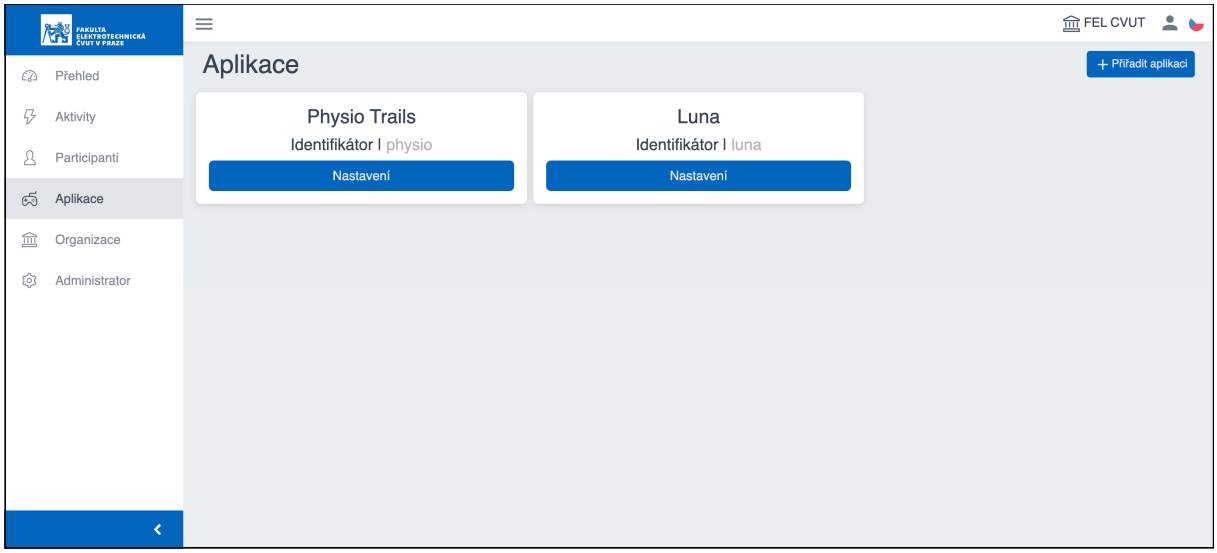


Obrázek A.3. Wireframe obrazovky porovnání 2 aktivit.

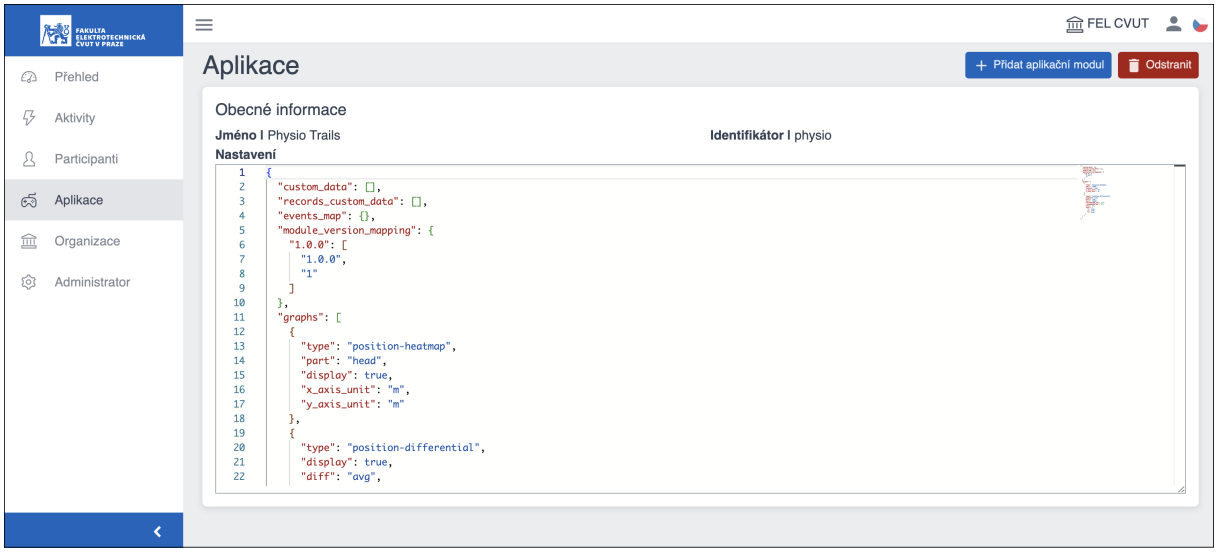
Příloha B
Snímky rozhraní klientské aplikace



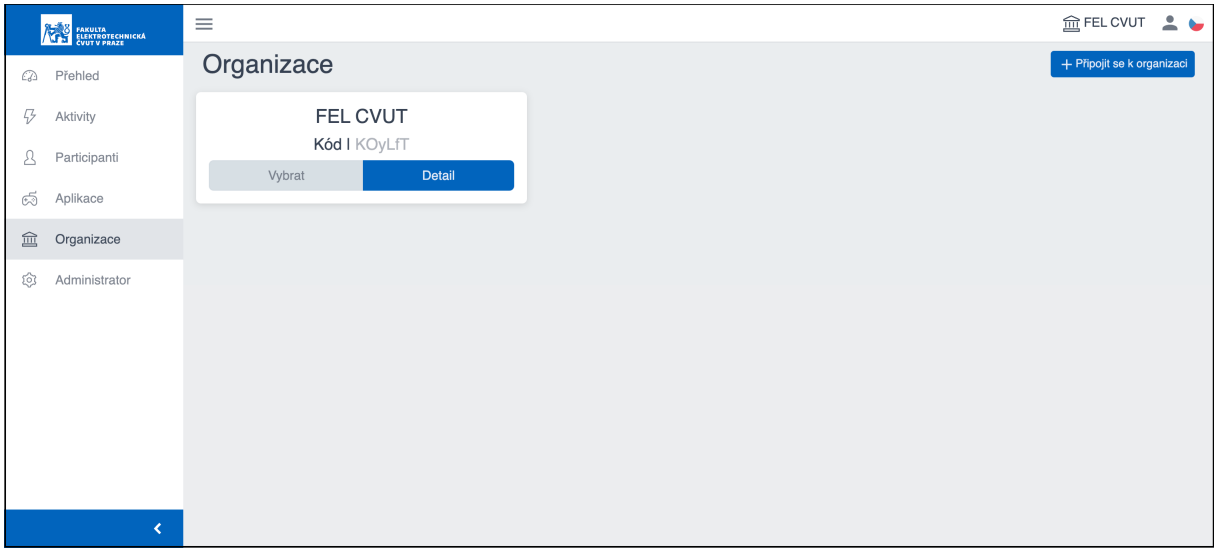
Obrázek B.4. Stránka přehledu.



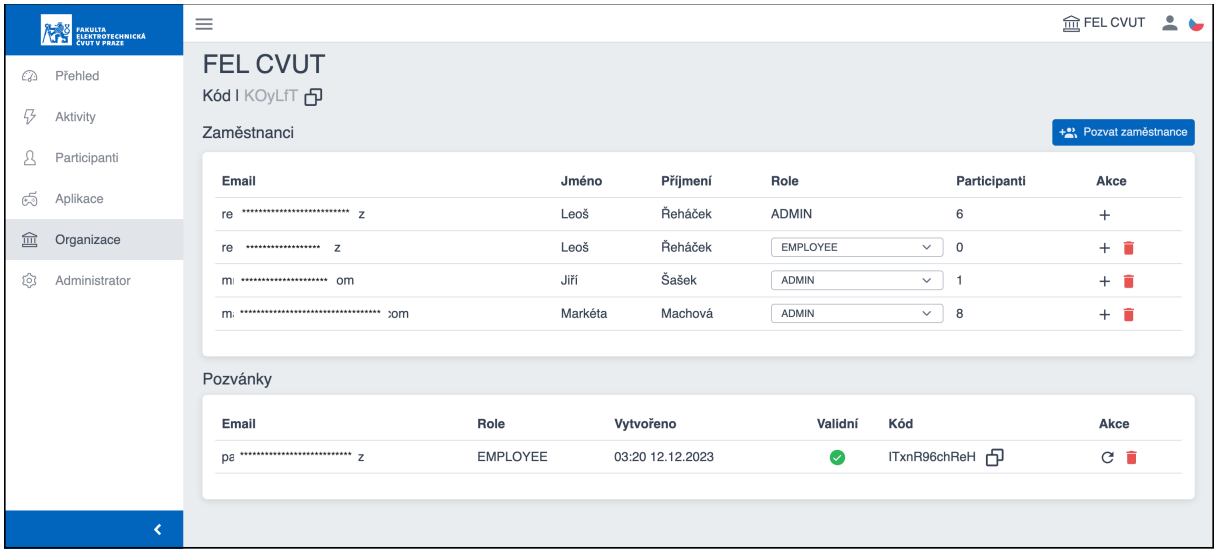
Obrázek B.5. Stránka seznamu aplikací organizace.



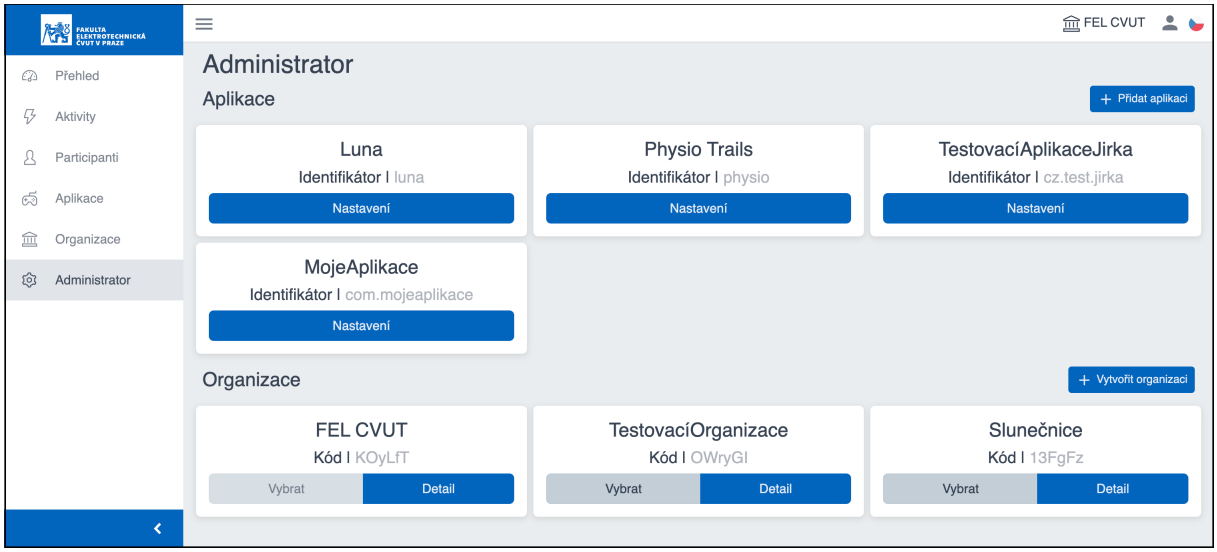
Obrázek B.6. Stránka detailu aplikace.



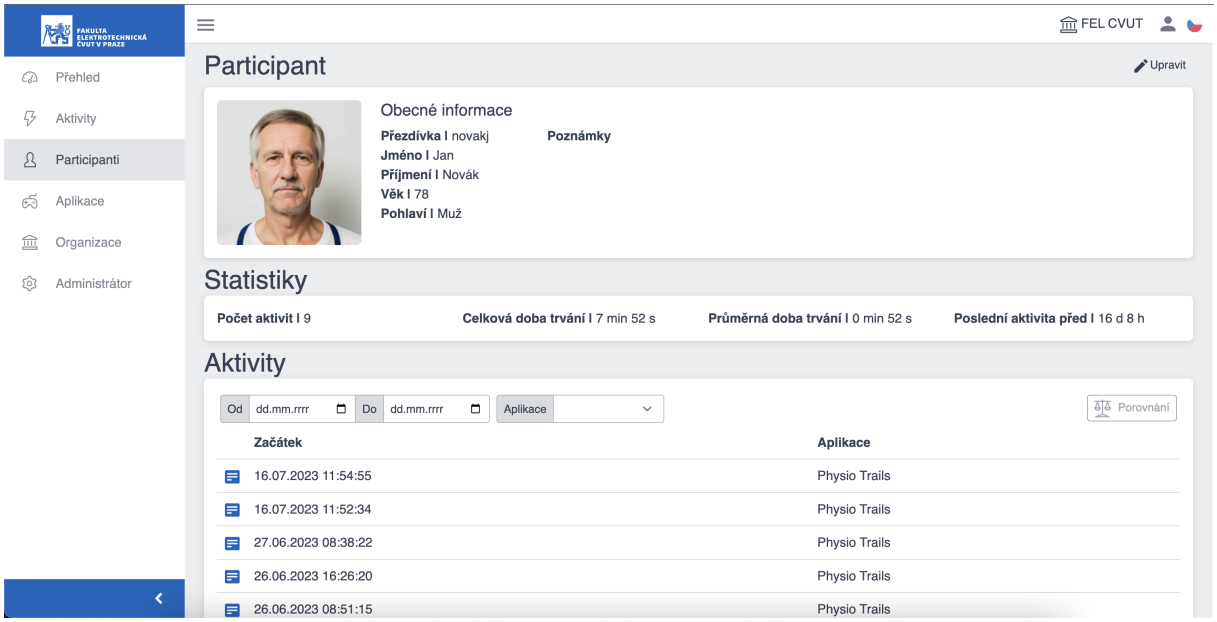
Obrázek B.7. Stránka seznamu organizací.



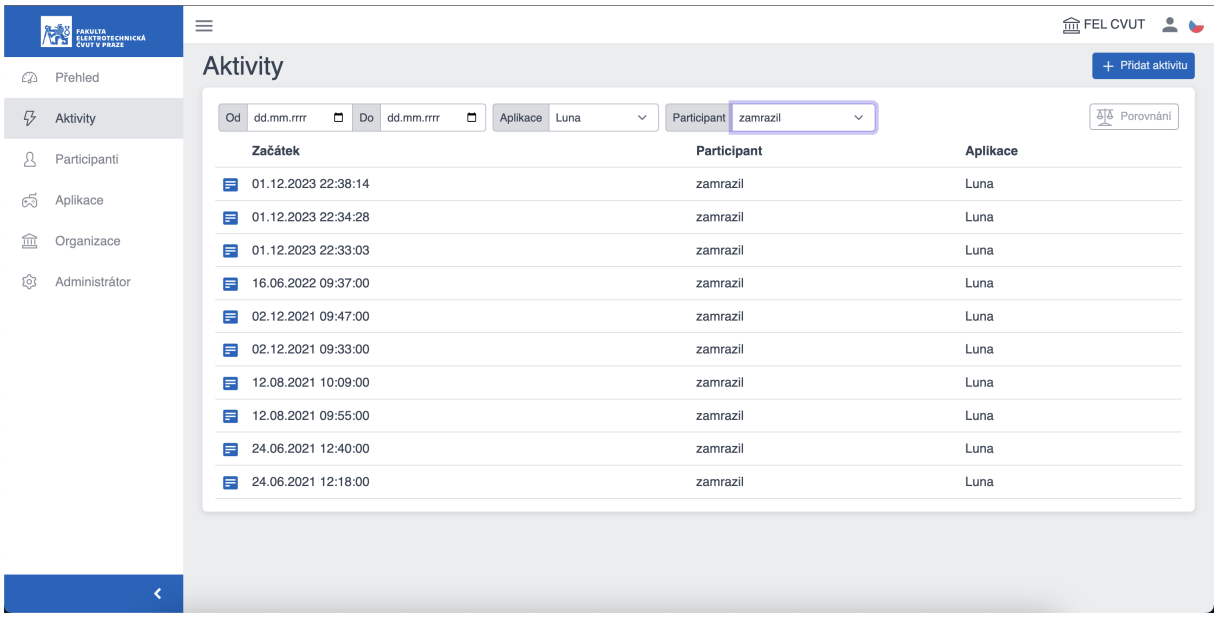
Obrázek B.8. Stránka detailu organizace.



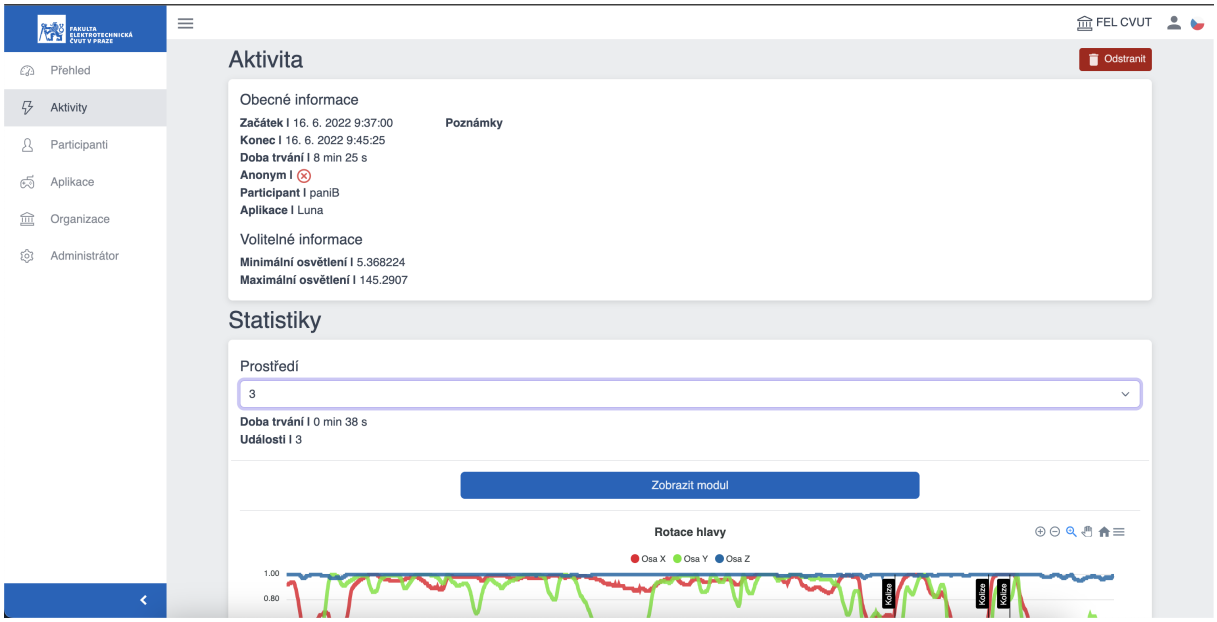
Obrázek B.9. Stránka administrátora.



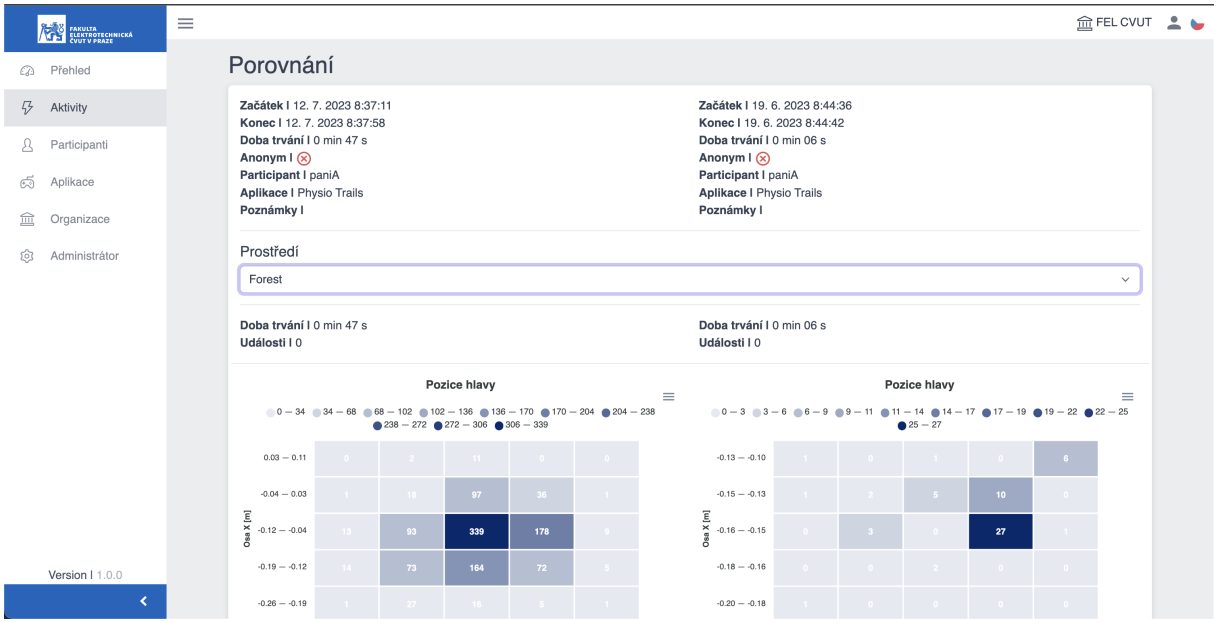
Obrázek B.10. Stránka detailu participanta.



Obrázek B.11. Stránka seznamu aktivit.



Obrázek B.12. Stránka detailu aktivity.



Obrázek B.13. Stránka porovnání aktivity.

Příloha C
Ukázky kódu

C.1 Formát logovacích dat ve formátu JSON

```
{
  "application_identifier": string, // identifikace aplikace
  "start": string, // čas začátku logování
  "log_version": string, // verze logu
  "log_rate": number, // doba, jak často se zaznamenává nový záznam
  "end": string, // čas konce logování
  "custom_data": object // objekt, pro vložení libovolných dat
  "records": [ // pole jednotlivých záznamů
    {
     "timestamp": string, // čas záznamu
     "tick": number, // pořadové číslo záznamu
     "events": []string, // pole událostí, které se udály v tomto záznamu
     "environment": string", // environment, ve kterém byl záznam zaznamenán
     "custom_data": {} // objekt, pro vložení libovolných dat záznamu
     "head": { // část těla, která se zaznamenává (head, left_hand, right_hand)
       "position": { // pozice
        "x": number,
        "y": number,
        "z": number
       },
       "rotation": { // rotace
        "x": number,
        "y": number,
        "z": number
       }
       ... // další části těla jako hlava
      }
    }
  ]
}
```

C.2 Ukázka kódu pro získání WebGL modulu v angularu

application-module.component.html

```
<iframe class="iframe" [src]="getApplicationModule()"></iframe>
```

application-module.component.ts

```
getApplicationModule() {
  // nadefinování potřebných query parametrů pro získání VR dat
  const params =
    `
    ?application_identifier=${this.application.identifier}&&
    organisation_code=${this.organisation.code}&&
    activity_id=${this.activityId}&&
    environment_id=${this.environmentId}
    `

  // Angular hlídá externí odkazy, které nejsou
  // hostované na stejném serveru jako Angular aplikace.
  // Z tohoto důvodu musíme Angularu říct,
  // že následující odkaz je bezpečný a může mu důvěřovat
  return this.sanitizer.bypassSecurityTrustResourceUrl(
    `
    ${environment.apiUrl}/webgl/modules/${this.application.id}/${this.moduleVersion}/${params}
    `
  );
}
```

C.3 Ukázka kódu pro hostování WebGL modulu na serveru

```
// hostování WebGL modulu je dostupná na cestě /webgl/modules/:applicationId/:moduleVersion/
this.app.use("/webgl/modules/:applicationId/:moduleVersion/*", (req, res, next) => {

  let applicationId = req.params.applicationId
  let moduleVersion = req.params.moduleVersion
  let assets = ""

  // pokud WebGL požádá o soubory, je jejich název přidán do cesty
  if (req.params[0]){
    assets = `/${req.params[0]}`;
  }

  // načtení požadovaných souborů ze složky /public/modules/${applicationId}/${moduleVersion}
  express.static(
  path.resolve(__dirname, `../../public/modules/${applicationId}/${moduleVersion}${assets}`),
  {
    setHeaders: (res, path) => {
      if (path.endsWith(".gz")) {
        // přidání http response hlavičky pro gzip
        res.set("Content-Encoding", "gzip");
      }
      if (path.includes("wasm")) {
        // přidání http response hlavičky pro wasm
        res.set("Content-Type", "application/wasm");
      }
    }
  }
  )(req, res, next);
});
```

Příloha D
Uživatelský manuál

Tato příloha obsahuje uživatelský manuál pro VR dashboard aplikaci. Manuál je určen pro běžné uživatele, kteří nemají práva `superAdmin` a `developer`. V manuálu jsou popsány jednotlivé úkony a operace, které lze v aplikaci provádět. VR dashboard aplikace je dostupná na adrese[1] uvedené v poznámce pod čarou.

V rámci manuálu je použito slovo `sekce` jako například „v sekci Participant“. Sekce v tomto případě znamená stránka, na kterou se dostaneme po kliknutí na odkaz v levém navigačním panelu se stejným názvem jako je název zmíněné sekce.

D.1 Participant

Tato sekce popisuje operace, které lze provádět s entitou participant.

D.1.1 Vytvoření participanta

Aby byli jednotliví participanti dostupní jak ve VR dahsboardu, tak i VR tréninkových aplikacích napojených na aplikaci, je potřeba nejdříve participanty vytvořit. Participanti se vytvářejí v sekci Participanti. Po přechodu do sekce v pravé horní části obrazovky se vyskytuje tlačítko `Přidat participanta`. Participanty může přidat pouze zaměstnanec organizace.

Po kliknutí na tlačítko se zobrazí modálové okno s formulářem, který je nutné vyplnit. Jediné povinné políčko je přezdívka, která by měla být unikátní, jelikož podle přezdívky musí být participant rozeznatelný od ostatních. Ostatní formulářová políčka jsou volitelná podle schválení participanta.

Po vyplnění formuláře a kliknutí na tlačítko `Přidat` je participant vytvořen a uživatel je přesměrován na detail participanta. Při vytvoření participanta je navíc tento participant přiřazen k zaměstnanci, který ho vytvořil. Chceme-li přiřadit participanta i ostatním zaměstnancům, postupujme podle návodu uvedeného v podsekci D.1.4.

D.1.2 Úprava participanta

Jakmile máme participanta vytvořeného a chceme upravit některé jeho vlastnosti jako například jméno, pohlaví, poznámku atd., můžeme toto provést na detailu participanta. Jsme-li na obrazovce s participantem, kterého chceme upravit, vlevo nahoře se nachází tlačítko `Upravit`.

Po kliknutí na tlačítko se přesměrujeme na obrazovku `Upravit participanta`. Zde můžeme upravit vlastnosti, přidat profilový obrázek či odstranit participanta. Po jakékoli změně je potřeba kliknout na tlačítko `Uložit`.

Chceme-li přidat profilový obrázek, klikneme na tlačítko `Změnit` pod profilovou fotkou a následně vybereme soubor ve formátu `jpeg`. Poté, co je soubor vybraný, klikneme na tlačítko `Uložit`.

[1] `https://vr-dashboard.leosrehacek.com/`

D.1.3 Odstranění participanta

Chceme-li participanta odstranit, přejdeme na detail daného participanta. Dále klikneme na tlačítko `Upravit`, což nás přesměruje na obrazovku `Upravit participanta`. Zde se v levé části nachází tlačítko `Odstranit participanta.` Po kliknutí na tlačítko se objeví modálové okno s hláškou a potvrzením, zda-li opravdu chceme participanta odstranit.

Po kliknutí na tlačítko `Odstranit` se participant odstraní a uživatel je přesměrován na seznam participantů. Také je zobrazena hláška, že participant byl odstraněn.

D.1.4 Přiřazení participanta k zaměstnanci

Přiřazovat participanty k zaměstnanci může pouze uživatel s rolí `ADMIN` nebo s právy `superAdmin`. Toto nastavení se provádí na detailu dané organizace.

Po přejití na obrazovku `Detail organizace` v seznamu zaměstnanců najdeme daného zaměstnance, kterému chceme přidat či odebrat přístup k participantovi. U zaměstnance klikneme na tlačítko `Přidat participanty` označeného symbolem +. Po kliknutí na tlačítko se zobrazí modálové okno se seznamem všech participantů.

K zaměstnanci přidáme daného participanta, když u něho zaškrtneme políčko ve sloupci `Přiřazen`. Poté, co máme participanty přiřazené, klikneme na tlačítko `Přiřadit`. V tuto chvíli se zaměstnanci přiřadí vybraní participanti a zobrazí se hláška: `Přiřazení participantů bylo aktualizováno.`

D.2 Aktivita

Tato sekce popisuje operace, které lze provádět s entitou aktivita.

D.2.1 Importování aktivity

Pokud nebyla aktivita automaticky nahrána nebo chceme-li nahrát předchozí aktivity, je toto možné v sekci `Aktivity`. Na obrazovce `Aktivity` se vpravo nahoře nachází tlačítko `Přidat aktivitu`. Po kliknutí na tlačítko se zobrazí modálové okno s formulářem.

Pro vytvářenou aktivitu je potřeba zvolit aplikaci, ve které byla aktivita vytvořena, vybrat soubor ve formátu `json` a vybrat participanta. Pokud byl participant anonymní, tak je nutné zakliknout přepínač `Anonym`. Volitelně je možné přidat k aktivitě poznámku. Poté, co máme všechna políčka vyplněna, aktivitu přidáme kliknutím na tlačítko `Přidat`.

Nyní musíme chvíli počkat, než se aktivita zpracuje. Po úspěšném zpracování je uživatel přesměrován na detail aktivity.

D.2.2 Filtrování aktivit

Na přehledu aktivit můžeme aktivity filtrovat pomocí aplikace, participanta nebo data pořízení. Standardně jsou zobrazeny všechny aktivity, které jsou seřazeny od nejnovějších.

Při filtrování pomocí data pořízení můžeme filtrovat pouze od, do nebo obojí od - do. Pokud vyplníme pouze `Od`, zobrazí se nám aktivity, které byly pořízeny později než vybrané datum. Pokud vyplníme pouze `Do`, zobrazí se nám aktivity pořízené dříve než vybrané datum. Pokud vyplníme `Od` i `Do`, zobrazí se nám aktivity, které byly pořízeny mezi těmito dvěma vybranými daty.

D.2.3 Zobrazení aktivity

Pokud chceme zobrazit detail dané aktivity, v sekci `Aktivity` najdeme příslušnou aktivitu a klikneme na ikonu `Detail`. Při výběru aktivity můžeme použít i filtrování, které je popsáno v podsekci D.2.2.

Po kliknutí na ikonu je uživatel přesměrován na detail aktivity. Na detailu aktivity se nachází dvě sekce, a to obecné informace a statistiky. Také je zde možné aktivitu odstranit.

Sekce obecné informace popisuje základní poznatky o dané aktivitě jako je například doba trvání, participant, aplikace či volitelné poznámky, které je možné průběžně dopisovat a upravovat.

Pokud chceme přidat poznámku, klikneme do vyznačené oblasti a v tu chvíli se nám poznámka přepne do editovacího režimu. Úpravy potvrdíme tlačítkem `Upravit`. Poté se poznámka přepne zpět do čtecího režimu a zobrazí se hláška: `Poznámka byla upravena.`

Sekce statistiky obsahuje `developerem` nadefinované grafy, aplikační modul a volitelné informace pro danou VR tréninkovou aplikaci. Statistiky jsou filtrované podle prostředí, které je možné vybrat pomocí rozevíracího pole.

Pokud pro danou aplikaci a verzi aktivity existuje aplikační modul, je možné si ho zobrazit pomocí tlačítka `Zobrazit modul`. V tuto chvíli se načte WebGL aplikační modul. Toto načtení může nějakou dobu trvat, jelikož to záleží na složitosti modulu a rychlosti internetového připojení.

D.2.4 Porovnání aktivit

Aktivity je možné porovnat na obrazovce Aktivity. Porovnat lze až 4 aktivity najednou, přičemž všechny 4 aktivity musí být z té samé VR aplikace. Pokud chce uživatel porovnat aktivity, klikne na tlačítko `Porovnání`, dále vybere až 4 aktivity, které chce porovnat, a klikne na tlačítko `Porovnat`. Při výběru aktivit pro porovnání můžeme použít i filtrování, které je popsáno v podsekci D.2.2.

Poté se uživateli zobrazí nová obrazovka, kde jsou vybrané aktivity zobrazeny vedle sebe se všemi informacemi a grafy, které daná VR aplikace podporuje. Dané porovnání se vždy vztahuje ke konkrétnímu Prostředí, které lze zvolit. Je možné, že pro dané prostředí nemá daná aktivita žádná data. V tuto chvíli je pro danou aktivitu zobrazena hláška: `Pro dané prostředí nejsou žádná data`.

D.2.5 Odstranění aktivity

Odstranění aktivity je možné na obrazovce `Detail aktivity`. Na této obrazovce se vpravo nahoře nachází tlačítko `Odstranit`. Po kliknutí na tlačítko se objeví modálové okno s hláškou a potvrzením, zda-li chceme aktivitu opravdu odstranit. Po kliknutí na `Odstranit` se aktivita odstraní, uživatel je přesměrován na seznam aktivit a zobrazí se hláška: `Aktivita byla odstraněna!`

D.3 Aplikace

Tato sekce popisuje operace, které lze provádět s entitou aplikace.

D.3.1 Zobrazení aplikace

V sekci `Aplikace` jsou zobrazeny všechny aplikace přiřazené dané organizaci. Uživatel si může zobrazit detail dané aplikace kliknutím na tlačítko `Nastavení`. Na detailu aplikace je zobrazeno aktuální nastavení aplikace a nadefinované aplikační moduly.

Návod, jak nastavit aplikaci, je uveden v sekci E.2. Návod, jak přidat aplikační modul k aplikaci, je v sekci E.3. Návod, jak přiřadit aplikaci k organizaci, je uveden v sekci E.4.

D.4 Organizace

Tato sekce popisuje operace, které lze provádět s entitou organizace.

D.4.1 Vytvoření organizace

Běžný uživatel může vytvořit pouze jednu organizaci, a to pouze v případě, že již není součástí jiné organizace. Možnost vytvořit organizaci se uživateli zobrazí hned po registraci, kdy má na výběr vytvoření organizace nebo připojení se k organizaci.

Pro vytvoření organizace stačí na kartě `Vytvořit organizaci` vyplnit jméno organizace a kliknout na tlačítko `Vytvořit`. Poté je organizace vytvořena a uživatel přesměrován na obrazovku `Přehled`.

D.4.2 Připojení se k organizaci

Uživatel má dvě možnosti, jak se připojit k organizaci. První způsob je hned po registraci a druhý je možný v sekci `Organizace`.

Po registraci do aplikace se uživateli zobrazí možnost vytvořit organizaci nebo připojit se k organizaci. Pokud se chce uživatel připojit k organizaci, je potřeba aby mu administrátor poslal kód pozvánky. Uživatel pak tento kód zadá v kartě `Připojit se k organizaci` do políčka kód organizace a následně klikne na tlačítko `Připojit se`. Poté je uživatel připojen k organizaci a přesměrován na obrazovku `Přehled`.

Druhá možnost připojení se k organizaci je v sekci `Organizace`. Na této obrazovce se vpravo nahoře nachází tlačítko `Připojit se k organizaci`. Po kliknutí na tlačítko se zobrazí modálové okno, kam je potřeba zadat kód pozvánky a kliknout na tlačítko `Připojit se`. Poté je uživatel přidán do organizace.

Návod, jak vytvořit pozvánku do organizace, je uveden v podsekci D.4.6.

D.4.3 Výběr organizace

Pokud je uživatel členem více organizací, má možnost mezi těmito organizacemi přepínat. Přepnutí na jinou organizaci je možné v sekci `Organizace`.

Na této obrazovce uživatel vidí seznam všech organizací, kterých je součástí. Pro výběr příslušné organizace je potřeba kliknout na tlačítko `Vybrat` u dané organizace. Po kliknutí na tlačítko se v horním panelu zobrazí název právě vybrané organizace a navíc se objeví hláška, že organizace byla vybrána.

D.4.4 Organizační role

Zaměstnanec v rámci organizace může mít 2 role, a to `ADMIN` a `EMPLOYEE`. Uživatel s právy `EMPLOYEE` může přidávat a upravovat své participanty, vytvářet, upravovat a porovnávat aktivity a zobrazovat si nastavení aplikace a detail své organizace.

Uživatel s právy `ADMIN` může navíc přidávat zaměstnance do organizace, nastavovat jejich role, přiřazovat participanty k jednotlivým zaměstnancům a přiřazovat již nadefinované aplikace ke své organizaci.

D.4.5 Pozvání zaměstnanců do organizace

Pozvat nové zaměstnance může do organizace pouze `ADMIN` nebo `superAdmin`. Při pozvání uživatel zadá email, pomocí kterého je zvaný zaměstnanec zaregistrovaný v aplikaci nebo pomocí kterého se bude do aplikace registrovat. Dále uživatel vybírá roli, jakou nový zaměstnanec může mít. Role se poté, co zaměstnanec přijme pozvánku, může změnit. Poté, co uživatel vytvoří pozvánku, je uživateli zobrazen kód pozvánky a její platnost.

D.4.6 Správa pozvánek

Spravovat pozvánky může pouze uživatel s rolí `ADMIN` nebo právy `superAdmin`. Pozvánky lze prodlužovat a odstraňovat.

Platnost pozvánky je možné prodloužit pomocí tlačítka `Obnovit`. Po kliknutí na tlačítko je pozvánka aktivní dalších 30 minut a je zobrazena hláška `Pozvánka byla obnovena`.

Pozvánku je taktéž možné odstranit a tím zabránit, aby se zaměstnanec připojil k organizaci, pokud tak ještě neučinil. Odstranění pozvánky je možné přes tlačítko `Odstranit` označené ikonou popelnice. Po kliknutí na tlačítko se objeví modální okno s hláškou a potvrzením, zda-li opravdu chceme pozvánku odstranit. Po kliknutí na tlačítko `Odstranit` je pozvánka odstraněna a je zobrazena hláška: `Pozvánka byla odstraněna`.

D.4.7 Odstranění zaměstnance

Odstranit zaměstnance může pouze uživatel s rolí `ADMIN` nebo právy `superAdmin`. Odstranění je možné na obrazovce `Detail organizace` dané organizace. Odstranit nelze zaměstnance, který je autorem dané organizace.

Odstranění je možné přes tlačítko `Odstranit zaměstnance` označené ikonou popelnice, která se nachází na řádku příslušného zaměstnance, kterého chceme odstranit. Po kliknutí na tlačítko se zobrazí modálové okno s hláškou a potvrzením, zda-li chceme uživatele odstranit. Po kliknutí na tlačítko `Odstranit` je daný zaměstnanec odstraněn a zobrazen hláška: `Zaměstnanec byl odstraněn!`

Příloha E
Manuál vývojáře

V této příloze jsou popsány úkony, které může provádět uživatel s právy `developer`. Developer kromě těchto úkonů může provádět to samé jako běžný uživatel. Manuál pro běžného uživatele je dostupný v příloze D.

E.1 Vytvoření VR aplikace

Aplikaci je možné vytvořit v sekci `Administrátor`. Zde u seznamu aplikací je vpravo nahoře tlačítko `Vytvořit aplikaci`. Po kliknutí na tlačítko se zobrazí modálové okno s formulářem.

Zde je potřeba zadat jméno aplikace a identifikátor aplikace. Identifikátor aplikace je unikátní a používá konvekci reverzní notace doménových jmen. Příklad takové konvekce je `com.example.application`. Poté, co jsou tyto informace vyplněny, aplikace se vytvoří kliknutím na tlačítko `Přidat` a zobrazí se hláška: `Aplikace byla vytořena`.

E.2 Nastavení VR aplikace

Po vytvoření VR aplikace je možné tuto aplikaci nastavit. V nastavení se definuje, jaké volitelné informace ze záznamů se mají zobrazovat, jaké se v záznamech vyskytují události, jaké máme WebGL moduly a v poslední řadě, jaké se budou zobrazovat grafy. Nastavení probíhá pomocí JSON textového editoru, které má definované validační schéma a upozorňuje na nesprávnost vyplněného nastavení.

Při psaní nastavení ve formátu JSON je možné v Monaco editoru kliknout pravým tlačítkem myši a vybrat možnost `Format document`. Po provedení se JSON naformátuje do lepšího formátu pro čtení.

Do nastavení aplikace se dostaneme přes sekci `Administrátor`, najdeme příslušnou aplikaci a klikneme na tlačítko `Nastavení`. Poté se zobrazí detail aplikace.

E.2.1 Volitelné parametry aktivity

Atribut *custom_data* definuje volitelné parametry aktivity. Jedná se o pole objektů, které mají atributy *path* a *languages*. *Path* určuje klíč, pod kterým je volitelný parametr uložen. *Languages* určuje překlady významu volitelného parametru.

Příklad nastavení:

```
"custom_data": [
  {
    "path": "lights_min",
    "languages": {
      "cs": "Minimální osvětlení",
      "en": "Minimal lights"
    }
  }
]
```

V tomto případě volitelnému parametru aktivity s identifikátorem `lights_min` nastavujeme český překlad na `Minimální osvětlení` a anglický na `Minimal lights`.

E.2.2 Volitelné parametry jednotlivých záznamů

Atribut *records_custom_data* definuje volitelné parametry jednotlivých záznamů aktivity. Je podobný jako atribut *custom_data*, ale navíc obsahuje atribut *type*, který určuje matematické funkce. V tento moment jsou dostupné tyto matematické funkce: sum (sčítání), avg (průměr), max (maximální hodnota) a min (minimální hodnota). Volitelné parametry záznamů v tuto chvíli podporují pouze číselné hodnoty.

Příklad nastavení:

```
"records_custom_data": [
  {
    "path": "lights_min",
    "type": "sum",
    "languages": {
      "cs": "Minimální osvětlení",
      "en": "Minimal lights"
    }
  }
]
```

V tomto případě volitelnému parametru jednotlivých záznamů s identifikátorem `lights_min` nastavujeme český překlad na `Minimální osvětlení` a anglický na `Minimal lights`. Typ matematické funkce jsme nastavili na `sum (sčítání)`.

E.2.3 Události

Atribut *events_map* definuje události, které se nacházejí v záznamech aktivity. Pro danou událost se uvádí klíč a uživatelské překlady.

Příklad nastavení:

```
"events_map": {
  "collision": {
    "cs": "Kolize",
    "en": "Collision"
  }
}
```

V tomto případě události s klíčem `collision` nastavujeme český překlad `Kolize` a anglický překlad `Collision`.

E.2.4 Mapování WebGL modulu na verzi záznamu

Attribut *module_version_mapping* mapuje verzi WebGL modulu na verzi záznamu aktivity. Klíč určuje verzi WebGL modulu a pole určuje kompatibilní verze záznamů aktivity.

Příklad nastavení:

```
"module_version_mapping": {
  "1.0.0": [
    "1.0.0",
    "1.0.1"
  ]
}
```

V tomto případě mapujeme verzi modulu `1.0.0` na verzi záznamu aktivity `1.0.0` a `1.0.1`.

E.2.5 Grafy

Atribut *graphs* určuje, jaké grafy se mají na přehledu aktivity zobrazit. V současné době aplikace podporuje 5 typů grafů, a to: `position`, `position-differential`, `position-heatmap`, `rotation` a `rotation-polar`. Každý graf má své specifické atributy až na *type*, *display* a *part*. *Type* určuje typ grafu. *Display* určuje, zda-li se má graf zobrazit. A *part* definuje, jakou trackovací část těla chceme ze záznamu grafu použít pro vykreslení.

Příklad nastavení grafu typu `rotation-polar`:

```
"graphs": [
  {
    "type": "rotation-polar",
    "display": true,
    "part": "head",
    "axis": "y"
  }
]
```

V tomto případě daný typ grafu bude vykreslovat rotaci hlavy v ose Y.

Příklad nastavení grafu typu `rotation`:

```
"graphs": [
  {
    "type": "rotation",
    "display": true,
    "part": "head",
    "axis": {
      "x": true,
      "y": true,
      "z": true
    }
  }
]
```

V tomto případe bude graf vykreslovat rotaci hlavy ve všech osách (X,Y,Z).

Příklad nastavení grafu typu `position`:

```
"graphs": [
  {
    "type": "position",
    "part": "head",
    "display": true
  }
]
```

V tomto případě bude graf vykreslovat pozici hlavy v ose X a Z.

Příklad nastavení grafu typu `position-heatmap`:

```
"graphs": [
  {
    "type": "position-heatmap",
    "part": "head",
    "display": true,
    "x_axis_unit": "m",
    "y_axis_unit": "m"
  }
]
```

V tomto případě bude graf vykreslovat pozici hlavy, kdy jednotka horizontální a vertikální osy je metr (m).

Příklad nastavení grafu typu `position-differential`:

```
"graphs": [
  {
    "type": "position-differential",
    "display": true,
    "diff": "avg",
    "part": "head",
    "recommended_min": -0.5,
    "recommended_max": 0.5,
    "unit": "m",
    "axis": {
      "x": true,
      "y": true,
      "z": true
    }
  }
]
```

V tomto případě bude graf vykreslovat rozdíl pozice hlavy ve všech osách (X, Y, Z), kdy rozdíl bude počítán od průměru všech hodnot dané osy. Jednotka rozdílu je v metrech (m) a doporučené minimum a maximum je -0.5 metru a 0.5 metru.

E.2.6 Příklad kompletního nastavení aplikace

V této podsekci je zobrazeno kompletní nastavení aplikace, jak by mělo vypadat v JSON editoru. Jednotlivé sekce nastavení odpovídají předchozím ukázkám ze sekce E.2.

```
{
  "custom_data": [
     {
       "path": "lights_min",
       "languages": {
         "cs": "Minimální osvětlení",
         "en": "Minimal lights"
       }
     },
     {
       "path": "lights_max",
       "languages": {
        "cs": "Maximální osvětlení",
        "en": "Maximal lights"
      }
    }
  ],
  "module_version_mapping": {
    "1.0.0": ["1.0.0", "1.0.1]
  },
  "events_map": {
    "collision": {
      "cs": "Kolize",
      "en": "Collision"
    }
  },
  "records_custom_data": [
    {
      "path": "lights_min",
      "type": "sum",
      "languages": {
        "cs": "Minimální osvětlení",
        "en": "Minimal lights"
      }
    }
  ],
  "graphs": [
    {
      "type": "position",
      "part": "head",
      "display": true
    }
  ]
}
```

E.3 Přiřazení WebGL modulu k aplikaci

K VR tréninkové aplikaci lze přiřadit více verzí aplikačních (WebGL) modulů. Moduly přidáváme na obrazovce `Detail aplikace` přes tlačítko `Přidat aplikační modul`, který se nachází vpravo nahoře.

Po kliknutí na tlačítko se zobrazí modálové okno s formulářem. Zde je potřeba vybrat soubor ve formátu `zip` a vyplnit verzi aplikačního modulu. Přidání potvrdíme kliknutím na tlačítko `Přidat`. Poté musíme chvíli počkat na zpracování modulu, dokud se modálové okno nezavře a nezobrazí se hláška: `Modul byl přiřazen k aplikaci.`

Poté, co byl modul úspěšně přiřazen k aplikaci, je potřeba ještě namapovat verzi modulu na verzi záznamů aktivity. Toto nastavení je uvedeno v podsekci E.2.4.

E.4 Přiřazení aplikace k organizaci

Po vytvoření a nastavení aplikace je potřeba aplikaci přiřadit k jednotlivým organizacím, aby byla v dané organizaci dostupná. Přiřazení lze nastavit v sekci `Aplikace` dané organizace. V této sekci se vpravo nahoře nachází tlačítko s označením `Přidat aplikaci k organizaci`. Po kliknutí na tlačítko se objeví modálové okno, kde můžeme vybrat jednu z dosud nepřiřazených aplikací. Přiřazení potvrdíme tlačítkem `Přiřadit`. Poté se zobrazí hláška: `Aplikace byla přiřazena` a aplikace se zobrazí v seznamu přiřazených aplikací.

Chceme-li přiřadit aplikaci více organizacím, musíme v tuto chvíli vždy přepnout do dané organizace a provést přiřazení.

E.5 Odstranění aplikace

Aplikaci je možné odstranit na detailu aplikace pomocí tlačítka `Odstranit`, které se nachází vpravo nahoře. Po kliknutí na tlačítko se zobrazí modálové okno s hláškou a potvrzením, zda-li opravdu chceme aplikaci odstranit. Odstranění potvrdíme tlačítkem `Odstranit`. Poté se zobrazí hláška `Aplikace byla odstraněna` a uživatel je přesměrován do sekce `Administrátor`.

Bereme v potaz, že poté, co je aplikace odstraněna, VR tréninkové aplikace přestanou komunikovat s aplikací a nebude možné ukládat aktivity či získávat participanty. Aby v tuto chvíli vše fungovalo, je pouze možné znovu vytvořit aplikaci se stejným identifikátorem, se kterým jsme aplikaci odstranili.

Příloha F
Manuál správce

V této příloze jsou popsány návody pro úkony, které může provádět uživatel aplikace s právy `superAdmin`. `Super admin` může provádět mimo jiné to samé, co běžný uživatel a uživatel s právy `developer`. Návody těchto typů uživatelů jsou uvedeny v příloze D a příloze E.

F.1 Vytvoření organizace

Vytvořit novou organizaci může pouze uživatel, který dosud není přiřazen v žádné organizaci nebo super admin. Uživateli se nabídne tato možnost po registraci nebo po opětovném přihlášení. Super admin má tuto možnost dostupnou v administrátorské sekci.

V administrátorské sekci u seznamu organizací je tlačítko `Vytvořit organizaci`. Po kliknutí na tlačítko se objeví modální okno, kde je potřeba zadat jméno nové organizace. Po kliknutí na tlačítko `Vytvořit` se vytvoří nová organizace s náhodně generovaným kódem.

F.2 Správa organizací

Uživatel s právy `superAdmin` může spravovat a zobrazovat si všechny informace a entity jednotlivých organizací, které jsou v aplikaci vytvořeny. Přepnutí do náhledu jednotlivé organizace je možné přes administrátorskou sekci.

V administrátorské sekci v seznamu organizací uživatel klikne na tlačítko `Vybrat` u příslušné organizace, kterou chce zobrazit či spravovat. Po kliknutí na tlačítko se v horním panelu zobrazí název právě vybrané organizace a navíc se objeví hláška, že organizace byla vybrána.

Příloha G
Příručka pro programátory a správce aplikace

V této příloze jsou uvedeny návody a jednotlivé popisy architektury VR dashboard aplikace. Tyto návody by měly sloužit budoucím správcům aplikace či budoucím vývojářům VR dashboard aplikace a VR tréninkových aplikací. Účelem těchto návodů je lepší orientace v rámci aplikace a jejích specifik.

VR dashboard aplikace se skládá ze 3 hlavních repozitářů a 1 podpůrného repozitáře. Před spuštěním nebo nasazením aplikace na server je potřeba, aby pro tento projekt byla zřízena MongoDB databáze, do které aplikace ukládá data.

MongoDB databáze musí mít minimálně verzi 6 a aplikace k ní musí mít plný přístup. Navíc je žádoucí, aby správce aplikace měl taktéž k databázi přístup a mohl v databázi případně upravovat data. Při vývoji byla použita databáze s typem `Shared` a úrovní klastru `M0 Sandbox`, která plně dostačovala.

G.1 Struktura jednotlivých projektů

Struktura jednotlivých projektů je uvedena vždy repozitáři projektu v `README.md` souboru. V tomto souboru je popsána struktura, která je odlišná od běžně zavedených standardů dané platformy. To znamená, že například pro Angular zde není uvedena lokace proměnných jednotlivých prostředí, jelikož nejsou oproti standardu přesunuty.

G.2 Spuštění aplikace na lokálním počítači

Pokud chceme spustit aplikaci na svém počítači, musíme si stáhnout klientskou a serverou aplikaci z příslušných git repozitářů uvedených na adrese[1] a adrese[2], které jsou v poznámkách pod čarou, nebo je rozbalit ze souborů 1 a 2, uvedených v příloze I.

V každé složce jednotlivého projektu se nachází soubor README.md, kde je bližší návod, jak danou aplikaci spustit.

G.3 Nahrání aplikace na server

Pokud máme již aplikaci spuštěnou na lokálním počítači, můžeme sestavit aplikaci pro nasazení. V opačném případě je nutné nejdříve postupovat podle kapitoly G.2.

Sestavení klientské aplikace se provádí pomocí příkazu *npm run build*. Poté, co se příkaz provede, je ve složce projektu vytvořena složka *public*, ve které jsou soubory, které stačí nahrát na hostovací server.

Sestavení serverové aplikace se provádí příkazem *npm run zip:deploy* nebo *npm run build*. První z příkazů ve složce projektu vytvoří zip soubor, který obsahuje sestavenou aplikaci. Druhý příkaz vytvoří pouze soubory do složky *build* bez zabalení. Poté, co máme aplikaci sestavenou, stačí zip soubor nebo soubory ze složky *build* nahrát na hostovací server.

[1] `https://gitlab.fel.cvut.cz/rehacleo/vr_dashboard_be`
[2] `https://gitlab.fel.cvut.cz/rehacleo/vr_dashboard_fe`

G.3.1 Nahrání serverové aplikace na AWS Elastic Beanstalk

V rámci testování byla serverová aplikace nasazena na AWS Elastic Beanstalk. Pro toto nasazení je v aplikaci nadefinovaný speciální příkaz *npm run zip:deploy-aws*. Tento příkaz sestaví aplikaci a zabalí jí do zip souboru, který lze na AWS Elastic Benastalku nahrát. Kromě sestavené aplikace, zip soubor obsahuje i definici AWS Elastic Beanstalk platformy, jako je například zvýšení request body dotazů.

G.4 Zapojení VR dashboard Unity balíčku do VR tréninkové aplikace

Pro zapojení VR dashboard logger balíčku do projektu v Unity je nejdříve nutné tento balíček přidat do projektu. Toto lze provést přes Package Manager. Zde klikneme na `add` a následně na `Add package from git URL`. Jako url použijeme adresu našeho VR dashboard logger repozitáře, jehož hodnota[1] je uložena v poznámce pod čarou. Kdyby url nefungovala, je možné zkusit náhradní url[2] uvedené v poznámce pod čarou.

Po přidání balíčku do Unity můžeme v projektu začít používat metody, které balíček podporuje. Ukázka, jak se balíček používá, je uvedena v `README.md` souboru v git repozitáři. Balíček navíc obsahuje komentáře, které vývojové prostředí bez problému zobrazuje.

G.5 Vytvoření aplikačního WebGL modulu

Aplikační WebGL modul slouží pro zobrazení složitějších statistik či vizualizací, které aplikace nepodporuje. Ná následujícím odkazu[3] uvedeném v poznámce pod čarou je ukázkový Unity projekt s implementací jednoduchého WebGL modulu. Tento modul získá z VR dashboard aplikace záznamy aktivity, které vypíše na obrazovku. Pro komunikaci s VR dashboard aplikací používá VR dashboard logger. Při vlastní implementaci se tak lze inspirovat, či začít podle ukázkového projektu.

Projekt navíc obsahuje potřebné nastavení pro postavení WebGL aplikace, jako je například komprimace, kterou VR dashboard vyžaduje. Po vytvoření a postavení WebGL modulu, je potřeba postavenou WebGL aplikaci zabalit do formátu a nahrát do VR dashboard aplikace. Návod, jak tento WebGL modul nahrát, je uveden v sekci E.3.

G.6 Databáze

V této sekci jsou uvedeny návody, které je v tuto chvíli potřeba provádět úpravou přímo v databázi. Databázi je možné upravovat ve webové aplikaci MongoDB nebo při použití oficiální aplikace MongoDB Compass.

Návod, jak prohlížet data přes webovou aplikaci, je uveden na odkazu[4] uvedeném v poznámce pod čarou. Návod na instalaci a použití MongoDB Compass je uveden na odkazu[5] uvedeném v poznámce pod čarou.

[1] `https://gitlab.fel.cvut.cz/rehacleo/vr_dashboard_logger`
[2] `https://github.com/rehacekleos/vr_dashboard_logger`
[3] `https://gitlab.fel.cvut.cz/rehacleo/vr_dashboard_webgl_module_template`
[4] `https://www.mongodb.com/docs/atlas/atlas-ui/#interact-with-your-data`
[5] `https://mongodb.com/docs/compass/current/#what-is-mongodb-compass-`

G.6.1 Nastavení práv superAdmin nebo developer uživateli

V tuto chvíli je možné uživateli aplikace nastavit práva `superAdmin` nebo `developer` pouze přes úpravu v databázi.

Pro přidání práv je potřeba upravit dokument v kolekci s názvem `Users`. V této kolekci vyhledáme daného uživatele s daným emailem a tento dokument upravíme. Pokud chceme přidat práva `superAdmin`, přidáme do dokumentu field s klíčem `superAdmin` a hodnotou `true`, kdy jako typ fieldu vybereme `Boolean`. Obdobně můžeme nastavit práva `developer`, kdy místo klíče `superAdmin` použijeme klíč `developer`. Po uložení upraveného dokumentu se uživateli nastaví daná práva a uživateli se stačí odhlásit a opět přihlásit do aplikace.

G.6.2 Odstranění organizace

V tuto chvíli je možné organizaci odstranit pouze přes úpravu přímo v databázi.

Organizace se odstraňuje tak, že z databáze odstraníme příslušný dokument, který obsahuje onu organizaci. Berme v potaz, že pokud odstraníme organizaci, tak v aplikaci ztratíme přístup ke všem datům dané organizace. Jediná možnost, jak tato data v aplikaci obnovit, je v databázi vytvořit organizaci se stejným Id a kódem jako právě odstraněná organizace.

G.7 Klientská aplikace

V následující sekci jsou popsány návody, které se týkají klientské aplikace.

G.7.1 Přidání nového jazyka do aplikace

Přidání nového jazyka do VR dashboardu je jednoduché. Stačí vytvořit nový JSON soubor se všemi klíči, které obsahuje již existující jazyk a doplnit odpovídající texty v příslušném jazyce. Tento nový soubor následně uložit v projektu `vr_dashboard_fe` do složky `src/assets/i18n` a v environment souborech nadefinovat nový jazyk jako dostupný.

Pro kontrolu všech vyplněných klíčů v jednotlivých jazycích oproti hlavnímu jazyku je v projektu implementován příkaz `npm run check:translation`, který do konzole vypíše, jaké klíče chybí, jaké klíče přebývají a jaké klíče se v aplikaci nepoužívají.

G.7.2 Definování a úprava css stylů

Jelikož aplikace pro design používá CoreUI kit, jsou v aplikaci nadefinované css proměné, které je možné libovolně měnit. Definice těchto proměnných je možné změnit v souboru `/src/scss/_variables.scss`.

Pokud chceme nadefinovat globální vlastní styly a css třídy, můžeme tak učinit v souboru `/src/scss/_custom.scss`. Jestliže chceme vytvořit vlastní styly a třídy pouze pro specifickou komponentu, můžeme je vytvořit vždy ve specifickém scss souboru dané komponenty.

CoreUI kit navíc používá Bootstrap verze 5. To znamená, že v aplikaci můžeme používat i třídy známé z Bootstrapu.

G.7.3 Úprava a vygenerování nového validačního schématu pro Monaco editor

Validační schéma pro Monaco editor se odvozuje z typescript typů, které definují podobu jednotlivých grafů. Toto zaručuje, že v rámci kódu programátor přistupuje na stejné atributy, které se poté zobrazí uživatelům při nastavení aplikací.

Hlavní typescript typ, na základě kterého se generuje validační schéma, je uveden v souboru `/src/app/models/application.model.ts` a má název `ApplicationSetting`. Typy, které definují nastavení jednotlivých grafů, jsou uvedeny v souboru `/src/app/models/graph.model.ts`.

Poté, co jsme provedli změny některého z typů, je potřeba přegenerovat validační schéma. Pro toto existuje příkaz `npm run generate:app-schema`, který z typů vygeneruje schéma a uloží ho do souboru `/src/assets/monaco-editor/settings-type.json`. Tento soubor následně používá Monaco editor.

G.7.4 Úprava či přidání nového typu grafu

Grafy jsou v klientské aplikaci implementovány jako jednotlivé komponenty. Tyto komponenty se nachází ve složce `/src/app/components/charts`. Název komponenty je stejný jako typ grafu použitý v nastavení aplikace.

Pokud chceme přidat nový typ grafu, stačí v této složce vytvořit novou komponentu a implementovat jí. Poté je komponentu potřeba přidat do 2 stránek, které tyto komponenty využívají. Jedná se o stránky `activity-detail` a `activity-compare`, které najdeme ve složce `/src/app/pages/activity`, a to konkrétně `/activity-detail/activity-detail.component.html` a `/activity-compare/activity-compare.component.html`.

G.8 Serverová aplikace

V následující sekci jsou uvedeny návody, které se týkají serverové aplikace.

G.8.1 Přidání nové proměnné prostředí

Pokud chceme upravit či přidat novou proměnnou prostředí, musíme upravit model konfigurace a poté nastavit danou proměnnou prostředí. Model konfigurace se nachází v souboru `/src/configs/config.model.ts`. V tomto souboru se nachází 2 třídy, a to `BaseConfigModel` a `ConfigModel`.

`BaseConfigModel` popisuje společnou konfiguraci pro jednotlivé prostředí. `ConfigModel` popisuje konfiguraci, která může být odlišná pro každé prostředí a není například nastavená pomocí proměnné prostředí.

Aktuálně se používá pouze společná konfigurace. Aktuální hodnoty dané konfigurace a definice proměnných prostředí se nachází v souboru `/src/configs/base.config.ts`. Zde se nachází konstanta `baseConfig`, která má právě typ `BaseConfigModel`.

Pokud chceme použít hodnoty konfigurace v rámci kódu, měli bychom tak učinit pomocí `ConfigFactory.getConfig()`, což vrátí konfiguraci s aktuálními hodnotami.

G.8.2 Přidání nové entity v rámci serverové aplikace

Pokud chceme přidat novou entitu do aplikace, je vhodné pro ni vytvořit model, kontrolér, servisu a DA. Nejlepší je ve složce `/src/app` vytvořit novou složku s názvem entity, kde tyto soubory vytvoříme.

V souboru model by měla být definována typescript třída, která popisuje atributy nové entity. Kontrolér by měl rozšiřovat třídu `BaseController` a implementovat REST endpointy. Servisa by měla rozšiřovat třídu `BaseService`. V servise by se měla implementovat business logika. Navíc by se v service měla vytvořit instance DA. DA by mělo rozšiřovat třídu `BaseDataAccess`. V DA by měla být napsána logika pro komunikaci s databází.

Aby nový kontrolér a servisa byly dostupné, je potřeba je přidat do bootsrapu aplikace. V bootsrapu aplikace se definují servisy, kontroléry a jejich závislosti. Bootstrap se nachází v souboru `/src/bootstrap/server/serverBase.bootstrap.ts`.

G.8.3 Správa databáze

Nastavení databáze jako názvy kolekcí, definice indexů, připojení k databázi, atd. je umístěno ve složce `/src/shared/repositories/mongoDb`. Zde se nachází 3 hlavní soubory, a to za prvé `collectionName.enum.ts`, za druhé `mongoDb.connection.ts` a za třetí `mongoDb.indexes.ts`.

V souboru `collectionName.enum.ts` je definovaný enum jednotlivých kolekcí. Tento enum se používá jak pro programátory, tak i pro vytváření kolekcí v připojené databázi. Při startu aplikace a připojení se k databázi server prověří, jestli jsou vytvořeny všechny kolekce uvedené v enumu a pokud některá kolekce chybí, automaticky ji vytvoří.

V souboru `mongoDb.connection.ts` je implementována logika pro připojení aplikace k databázi.

V souboru `mongoDb.indexes.ts` jsou definovány indexy jednotlivých kolekcí. Indexy se definují pro optimálnější hledání podle specifikovaného atributu, který je v indexu nastaven. Aplikace po spuštění a připojení se k databázi ověří, jestli jsou všechny nadefinované indexy vytvořeny v databázi. Pokud tomu tak není, tak se chybějící indexy automaticky vytvoří.

G.8.4 Specifikace swagger dokumentace

Swagger dokumentace je v aplikaci definována pomocí komentářů, které jsou nadefinovány v jednotlivých kontrolerech. V tuto chvíli se jedná o kontrolery `public` a `default`. Dále je swagger nadefinový pro express middlewary v souboru `server.ts`.

Pro dokumentaci pomocí komentářů se používá balíček `swagger-jsdoc`, který je dostupný na adrese[1] uvedené v poznámce pod čarou.

Kromě specifikace pomocí komentářů je v souboru `/src/swagger/definition.ts` uvedena základní konfigurace swaggeru. Kromě toho se zde nachází i definice modelů, parametrů a tagů.

[1] `https://github.com/Surnet/swagger-jsdoc#readme`

Příloha H
Slovníček

Aktivita	Jedno testování participanta v dané VR tréninkové aplikaci. Jinak řečeno je to proces od spuštění aktivity po ukončení aktivity v rámci VR tréninkové aplikaci.
DA	Data Access (Přístup k datů).
headset	VR brýle
Participant	Člověk, který je testován v dané VR tréninkové aplikaci.
UI	Uživatelské rozhraní.
Uživatel	Správce nebo terapeut, který má na starosti používání určité VR tréninkové aplikace. Navíc je to uživatel, který bude používat cílovou aplikaci pro analýzu dat.
VR	Virtuální realita.
VR dashboard	Pojmenování implementované aplikace.

Příloha I
Seznam přiložených souborů

`server_application.zip`	Soubor 1 - Projekt Node.js serverové aplikace.
`client_application.zip`	Soubor 2 - Projekt Angular klientské aplikace.
`vr_dashboard_logger.zip`	Soubor 3 - Projekt s unity balíčkem obsahující VR dashboard logger.
`webgl_module_template.zip`	Soubor 4 - Unity projekt s ukázkovým WebGL modulem za využití VR dashboard loggeru.
`text.zip`	Soubor 5 - Text diplomové práce v latexu.
`usability_testing.zip`	Soubor 6 - Zip souborů, které byly použity při uživatelském testování.