VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NÁSTROJ PRO SKRIPTOVÁNÍ SCÉNÁŘŮ EVALUACE LETECKÉHO KOKPITU

TOOL FOR SCRIPTING SCENARIOS FOR EVALUATION OF AERONAUTIC COCKPIT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **RICHARD GRANEC**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Granec Richard**
Obor: Informační technologie
Téma: **Nástroj pro skriptování scénářů evaluace leteckého kokpitu**
Tool for Scripting Scenarios for Evaluation of Aeronautic Cockpit
Kategorie: Uživatelská rozhraní

Pokyny:

1. Seznamte se s vybraným leteckým simulátorem, nástroji, které se ve spojitosti s ním používají, a s použitím skriptů při simulaci leteckých situací.
2. Navrhněte architekturu řešení pro správu a provádění skriptů řídících průběh letecké simulace.
3. Prototypujte základní verzi vytvářeného nástroje a jednotlivé prvky uživatelského rozhraní.
4. Na základě testů s uživateli iterativně vylepšujte vytvořený prototyp.
5. Sestavte s prototypovaných částí řešení produkčně použitelné pro správu a ovládání skriptů pro leteckou simulaci.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, rozpracování bodu 4.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Herout Adam, prof. Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cieľom tejto práce je vytvorenie nástroja pre správu scenárov bežiacich na pozadí leteckého simulátora. Úlohou týchto scenárov je prispôsobiť prostredie leteckého simulátora a v konečnom dôsledku aj celý let, pre jeho konkrétne využitie najmä pri testovaní zručností pilota alebo nového technického vybavenia. Riešenie pozostáva z komunikačnej aplikácie, ktorá komunikuje so simulátorom a získané dáta odosiela na NoSQL server a správcu scenárov, ktorý zvolený scenár spustí a zaznamenáva jeho činnosť. Pre výber konkrétneho scenára a jeho následnú kontrolu slúži webové grafické rozhranie. Takto vytvorený softvér bol testovaný a následne bude využívaný aj leteckým odvetvím firmy Honeywell.

Abstract

The main goal of this thesis is to develop a tool for managing scripts running at the background of a flight simulator. The task of these scripts is to adjust the environment of the flight simulator and in the final result the entire flight as well, in order to be used in particular situations for testing the skills of a pilot or at the implementation of a new technical device. The solution consists of a communicational application, which communicates with the simulator and sends the data to a NoSQL server and a script manager, which starts running the selected script and records its activity. Web graphic at user interface serves as a tool for selecting a particular script and its control. The software developed according to these criteria has been tested and will be used by the aviation branch of the Honeywell Company.

Kľúčové slová

Správca scenárov, Letecký simulátor, NoSQL, Redis, Scenáre riadiace let

Keywords

Scenario manager, Flight simulator, NoSQL, Redis, Scenario's control flight

Citácia

GRANEC, Richard. *Nástroj pro skriptování scénářů evaluace leteckého kokpitu.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Herout Adam.

Nástroj pro skriptování scénářů evaluace leteckého kokpitu

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána prof. Ing. Adama Herouta, Ph.D. Ďalšie informácie mi poskytol Ing. Lukáš Malý a Mgr. Tomáš Králiček. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Richard Granec
15. mája 2017

Poďakovanie

Moje poďakovanie patrí vedúcemu práce prof. Ing. Adamovi Heroutovi, Ph.D. a externým konzultantom firmy Honeywell Ing. Lukášovi Malému a Mgr. Tomášovi Králičkovi za ich odborné rady, pomoc a motiváciu.

Obsah

1 Úvod **2**

2 Evaluácia leteckého kokpitu **3**
2.1 Letecké simulátory 3
2.2 Scenáre riadiace a zaznamenávajúce priebeh letu 7
2.3 Úložisko dát, nerelačná databáza 8
2.4 Databáza Redis 9
2.5 Webová technológia Ajax 12

3 Nástroj pre správu scenárov **14**
3.1 Štruktúra výslednej aplikácie 14
3.2 Webový server 16
3.3 Webové užívateľské rozhranie 17
3.4 Komunikácia a sprístupnenie dát 19

4 Program pre sprístupnenie dát a riadenie letu **21**
4.1 Správca scenárov 21
4.2 Komunikačná aplikácia 22
4.3 Spôsob konfigurácie správcu scenárov a komunikačnej aplikácie 25
4.4 Použitie výslednej aplikácie v praxi a jej vyhodnotenie 27

5 Záver **28**

Literatúra **29**

Prílohy **31**

A Kompletný vzhľad webového grafického užívateľského rozhrania **32**

B Plagát **33**

Kapitola 1

Úvod

Letecké simulátory sú často využívané nie len pre zábavu, ale v praxi sú nimi testované znalosti a zručnosti pilota, prípadne technického vybavenia leteckých dopravných prostriedkov. V týchto prípadoch použitia je mnohokrát požadovaná možnosť riadenia behu simulátora externými procesmi, čím je možné prispôsobiť prostredie simulátora a v konečnom dôsledku aj celý let konkrétnemu testovanému subjektu. Počas takto vykonávaných testov vzniká pre možnosť následnej analýzy a vyhodnotenia vykonávaných testov potreba zaznamenávať dáta a konfiguráciu technického vybavenia, priebeh letu a činnosť pilota. Všetky takto zadefinované požiadavky je možné napĺňať využitím scenárov, ktoré bežia na pozadí leteckého simulátora a komunikujú s ním.

Táto práca vznikla ako reakcia na potrebu vytvorenia správcu vyššie opísaných scenárov, ktorý by umožnil výber požadovaného scenára, zaistil by spoľahlivú komunikáciu scenárov a simulátora a následne by vhodnou metódou zobrazoval priebeh a stav, v ktorom sa spustený scenár nachádza. Pre možnosť výberu požadovaného scenára a sledovania jeho priebehu je využité webové grafické rozhranie, vďaka ktorému je možné obsluhovať vytvorenú aplikáciu z rôznych zariadení. Návrh na vypracovanie tejto témy, následné testovanie zhotovenej aplikácie a taktiež aj jej použitie, prebieha v spolupráci s leteckým odvetvím firmy Honeywell, ktoré sa zaoberá vývojom technického vybavenia leteckých dopravných prostriedkov.

Kapitola číslo dva opisuje technológie, využívané vytvorenou aplikáciou. V tretej kapitole sa nachádza štruktúra výslednej aplikácie. Štvrtá kapitola opisuje funkcionalitu správcu scenárov a komunikačnej aplikácie.

Kapitola 2

Evaluácia leteckého kokpitu

Proces vývoja technického vybavenia leteckých dopravných prostriedkov vyžaduje viacero foriem testovania daných komponentov. Testovanie zahŕňa aj použitie prvku počas simulácie letu, ktorá neprebieha vo vzduchu a tým pádom sú následky zlyhania testovaného objektu minimálne. Podobný princíp je využívaný aj pri testovaní schopností a znalostí pilota, kedy jeho chyby a nesprávne úsudky nikoho neohrozia. Aby boli výsledky testovania dôveryhodné, je potrebné, aby simulácia letu čo najviac odpovedala realite, teda reálnemu kokpitu. Jeho evaluácia zahŕňa hardvérové a softvérové vybavenie. V tejto práci je opísaná len softvérová stránka evaluácie. Tá zahŕňa letecký simulátor, ako hlavný prvok tejto časti a scenáre, ktoré jeho chod ovplyvňujú.

2.1 Letecké simulátory

Samotné lietanie v reálnom svete je pomerne finančne náročné a závisí na mnohých faktoroch. Ako príklad si môžme uviesť závislosti na poveternostných podmienkach, stavu lietadla prípadne jeho údržby, ktorá môže byť plánovaná v rámci pravidelného servisu, ale taktiež aj nutná v reakcii na novo vzniknutú závadu na používanom stroji. Letecké simulátory umožňujú napĺňať potreby lietať, v prípade že lietanie so skutočným strojom nie je z určitých príčin možné alebo vhodné [14].

V tejto časti sú opísané oba letecké simulátory, ktoré môžu spolupracovať s vytvorenou aplikáciou.

2.1.1 Microsoft Flight Simulator X

Tento letecký simulátor známy pod skratkou *FSX* vyšiel v roku 2006 a predstavuje desiate vydanie edície *Microsoft Flight Simulator*. Táto edícia začala v roku 1982 pod názvom *Flight Simulator 1.0*. Na Obrázku 2.1 je možné vidieť verziu z roku 1988.

Obr. 2.1: Flight Simulator 3.0 vydaný v roku 1988

Aktuálnu verziu je možné vidieť na Obrázku 2.2. Využité grafické jadro obsahuje mnoho nových 3-D modelov, ktoré výrazne napomáhajú v snahe o čo najrealistickejšie zobrazenie. To je doplnené aj o efekt vlnenia sa hladiny oceánov, pozemnú dopravu, pohybujúce sa zvieratá a zmenu osvetlenia kokpitu vzhľadom na pozíciu slnka a samotného stroja.

Zaujímavým prvkom je aj navigácia *Garmin*, ktorá je pilotmi často používaná a jej ovládanie v simulátore odpovedá skutočnosti. To rozširuje možnosť použitia tohto simulátora nie len na testovanie zručností pilota riadiť stroj, ale umožní aj otestovať pilotove schopnosti orientácie s využitím navigácie. Simulované je aj reálne fungovanie letiska zahrňujúce cisterny tankujúce palivo a vozidlá, ktoré zabezpečujú naloženie a vyloženie batožiny cestujúcich.

Obr. 2.2: Aktuálna verzia s názvom Microsoft Flight Simulator X

Dostupných je vyše 24 000 letísk pokrývajúcich každý kontinent. Poveternostné podmienky je možné nastaviť a prispôsobiť si podľa svojich predstáv. Na výber je napríklad možnosť nastavenia intenzity dažďa, hustoty oblačnosti alebo sily vetra. V prípade pripojenia počítača k sieti internet je možné nechať počasie automaticky nastaviť a aktualizovať

podľa skutočného počasia odpovedajúceho realite vo zvolenej oblasti. V rámci zobrazenia hviezd na oblohe, bola doplnená ich vizualizácia a rozloženie, ktoré odpovedá skutočnému rozloženiu hviezd v reálnom svete.

Jednou z poskytovaných možností využitia je prístup voľného letu, kedy je možné vzlietnuť kedykoľvek, bez nutnosti komunikácie s riadiacou vežou a potreby prejsť celým procesom korektného opustenia letiska. Avšak dostupné sú aj nástroje, umožňujúce vykonať všetky úkony odpovedajúce reálnemu vzlietnutiu, ako napríklad predletová kontrola stroja, kontrola poveternostných podmienok, vytvorenie plánu letu a komunikácia s vežou. Tá zahŕňa schválenie letového plánu, povolenie vstupu na dráhu a samotné povolenie k vzletu [6].

2.1.2 X-Plane

Ako tvrdia tvorcovia tohto simulátora nejedná sa o hru, ale o profesionálny nástroj, ktorý môže byť použitý na predpoveď vlastností leteckého dopravného prostriedku s neuveriteľnou presnosťou. X-plane disponuje podporou podzvukovej a nadzvukovej letovej dynamiky dovoľujúc užívateľovi simulovať letové vlastnosti najpomalších, ale aj najrýchlejších leteckých prostriedkov.

Základná inštalácia obsahuje tridsať lietadiel, stíhačiek a helikoptér pokrývajúce najnovšie modely, ale aj históriu letectva. Tento zoznam je možné rozšíriť o ďalších 2000 modelov, ktoré je možné stiahnuť z oficiálnych stránok výrobcu. Plná inštalácia scenéria pokrýva našu planétu v rozmedzí 74 stupňov severnej až 60 stupňov južnej zemepisnej šírky. To zahŕňa cez 33 000 letísk, lietadlové lode, ropné plošiny pohybujúce sa s hladinou vody a taktiež pristávacie miesta helikoptér na strechách budov. Podporované je aj lietanie viacerých užívateľov s využitím siete LAN. Takto pripojený užívateľ môže zastupovať aj rolu kopilota alebo inštruktora, a spolupracovať s užívateľom riadiacim let. Aj vďaka tejto možnosti sa rozširuje aplikovateľnosť leteckých simulátorov počas doby výcviku a preskúšania pilotov.

Samotná simulácia je obohatená o možnosť vypúšťania vody na lesné požiare, a taktiež je možné odskúšať si priblíženie k letisku v noci alebo za búrlivého počasia s poškodeným lietadlom.

Obr. 2.3: Aktuálna verzia leteckého simulátora X-Plane

Počasie je rovnako ako v simulátore *FSX* konfigurovateľné od slnečných lúčov a termických stúpavých prúdov, až po búrky plné dažďu, silného vetra, ktoré znižujú viditeľnosť

na minimum. Ani tu nechýba možnosť simulácie počasia stiahnutého cez internet, ktoré odpovedá aktuálnej poveternostnej situácii vo zvolenej oblasti.

Tento simulátor je možné využívať v širokom zábere záujmu počnúc domácim využitím a lietaním pre zábavu, až po komerčné testovanie. Tomuto rozdeleniu odpovedajú aj tri dostupné verzie. Prvou z nich je verzia *X-Plane Global*. Ako z jej názvu vyplýva, obsahuje všetky dostupné scenárie, pokrývajúce takmer celú planétu. Ďalšou ponúkanou možnosťou je verzia *X-Plane Regional*. Tá sa od jej nadmnožiny *X-Plane Global* líši len tým, že obsahuje nekompletnú scenáriu definovanú len pre určitú časť planéty, a v tom dôsledku aj nižšou cenou. Pre profesionálne využitie je určená verzia *X-Plane Professional*, ktorá je podobná verzii *Global*, no je rozšírená o možnosti pripojenia a využitia rôznych doplnkových funkcií a zariadení.

V poslednej opísanej verzii je sprístupnená aplikácia *EFIS*, reprezentujúca program bežiaci na samostatnom počítači ponúkajúcom veľmi reálny primárny letecký displej (PFD) značky *Avidyne*. Jeho podobu je možné vidieť na Obrázku 2.4. Patrí medzi moderné vybavenie leteckých dopravných prostriedkov. Samotný displej využíval pre svoje zobrazenie technológiu CRT, ktorá bola neskôr nahradená technológiou LCD. Toto zariadenie reprezentuje a nahrádza viacero jednotlivých prístrojov využívaných v minulosti. Zobrazuje údaje o aktuálnej výške, relatívnej rýchlosti k vzduchu, naklonení stroja, vertikálnej rýchlosti a kurze letu.

Obr. 2.4: Primárny letecký displej (PFD).

Obr. 2.5: Multifunkčný displej (MFD).

Aplikácia *EFIS* ponúka tiež možnosť zobrazenia multifunkčného displeja (MFD), ktorý je možné vidieť na Obrázku 2.5. Ako už z jeho názvu vyplýva, umožňuje zobraziť viacero informácii na rozličných stránkach, medzi ktorými je možné prepínať pomocou viacerých tlačidiel, ktoré daný displej obklopujú. To umožňuje šetrenie miesta v kokpite, keďže displej zobrazuje dáta, ktorých zobrazenie počas celej doby letu nie je nevyhnutné.

K celému systému je navyše možné pripojiť skutočnú navigáciu značky *Garmin*, ktorá bude v tomto prípade zobrazovať dáta získané zo simulátora. Ukážku takejto navigácie je možné vidieť na Obrázku 2.6. Podporovaná je aj sférická projekcia výsledného obrazu využívajúca plátno, ktoré je vhodne rozostavené, a na ktoré je pomocou kombinácie viacerých dataprojektorov premietaný obraz. Kombináciou vyššie opísaných doplnkov, samotného simulátora a v neposlednom rade aj modelu kokpitu lietadla odpovedajúcemu realite, je možné docieliť reálne a ideálne podmienky pre tréning pilotov, prípadne testovanie nových technológií a zariadení [15].

Obr. 2.6: Navigácia Garmin G430, ktorú je možné pripojiť k simulátoru X-plane

Aplikácia bola prispôsobená a spúšťaná spolu so simulátorom X-Plane vo verzii 10. Počas tvorby tejto práce bola vydaná jedenásta verzia tohto simulátora, avšak vytvorená aplikácia na nej zatiaľ nebola testovaná. Kľúčovým prvkom korektného fungovania výslednej aplikácie je podpora doplnku simulátora *XPUIPC* aj pre verziu 11.

2.2 Scenáre riadiace a zaznamenávajúce priebeh letu

Letecké simulátory *Microsoft flight simulator X* a *X-Plane* poskytujú svojím užívateľom možnosť výberu dopravného prostriedku, štartovacieho letiska, prípadne úpravy času letu a poveternostných podmienok, avšak po takto spustenom lete už ho nie je možné ovplyvniť, respektíve prispôsobiť. Pri využívaní týchto simulátorov pre potreby testovania zručností pilota alebo nového elektrotechnického vybavenia, ktoré podlieha testovaniu z dôvodu jeho možného využívania v kokpite lietadla, je často potrebné vedieť ovplyvniť a zmeniť spustenú simuláciu. Vyžadované sú často úpravy stavu paliva, pozastavenie simulácie alebo úprava stavu určitých komponentov. Ako príklad je možné uviesť odstavenie motorov lietadla, čo je možné využiť pri testovaní krízových situácií. Práve scenáre prinášajú možnosť ako externými procesmi ovplyvniť chod simulácie a prispôsobiť ju tak požadovaným podmienkam. Takýmito úpravami a zásahmi do chodu spustenej simulácie je možné prispôsobiť ju tak, aby bolo možné testovaný subjekt otestovať čo najlepšie. Vďaka tomu je možné navodiť konkrétnu a špecifickú situáciu, v ktorej chceme sledovať správanie testovaného subjektu.

Po úspešnej konfigurácii letu a počas celej doby simulácie vzniká pre možnosť následnej analýzy letu, správania sa a reakcii pilota potreba zaznamenávania jeho priebehu, stavu ovládacích prvkov dopravného prostriedku a konfigurácii sledovaného technického vybavenia. Tieto potreby nám opäť umožňujú napĺňať scenáre, ktoré okrem riadenia letu umožňujú zaznamenávať všetky potrebné premenné a ukladať ich do zvolených logovacích súborov. Často dochádza ku kombinácii oboch týchto vlastností scenárov. Spustený scenár číta a kontroluje určité hodnoty a následne, po splnení danej podmienky, vykoná zápis dát a tým upraví simuláciu.

Ako príklad je možné uviesť situáciu, kedy scenár čaká, kým lietadlo naberie určitú výšku alebo kým sa priblíži k určitému bodu. Po jej dosiahnutí scenár zasiahne do spustenej simulácie napríklad odstavením palivovej pumpy motora, čím je simulovaná jej porucha a tým aj porucha celého motora. Následne je možné sledovať reakciu pilota na vzniknutý problém a periodickým čítaním dôležitých hodnôt a ich následným uložením získať dáta,

ktoré je možné po ukončení simulácie analyzovať. To umožňuje posúdiť spôsobilosť pilota reagovať na vzniknuté problémy, bez nutnosti ohrozenia jeho života. Ako príklad bol uvedený pomerne jednoduchý scenár, no možnosti jeho rozšírenia a využitia sú omnoho väčšie.

Scenáre predstavujú programy spustené na pozadí leteckého simulátora, ktoré s ním komunikujú. Môžu byť implementované pomocou rozličných programovacích jazykov, ale podmienkou je podpora daného jazyka doplnkom simulátora, teda existencia komunikačnej knižnice.

2.3 Úložisko dát, nerelačná databáza

Relačné databázové systémy, v anglickom jazyku známe pod názvom *Relational database management systems (RDMBS)*, sa v dnešnej dobe považujú za dominantnú technológiu, využívajúcu sa pre uloženie štruktúrovaných dát webových, ale aj iných aplikácií. V minulosti boli SQL databázy celosvetovo adaptované a často považované za jedinú alternatívu uloženia dát s možnosťou konzistentného pripojenia viacerých klientov. Popri ich používaní vznikli aj objektové databázy alebo XML sklady, avšak tieto technológie nikdy nezažili rovnakú adaptáciu ako RDMBS. Spomínané technológie boli v konečnom dôsledku zahrnuté relačno databázovými systémami, ktoré dovoľujú uložiť priamo dátový typ XML a využiť ho za účelom indexovania textu. V posledných rokoch, *"one size fits all"* zmýšľanie týkajúce sa dátových úložísk požadované najmä spoločnosťami pracujúcimi s webovými technológiami viedlo k vzniku najrôznejších databáz. Nový spôsob úschovy dát je pomenovaný termínom NoSQL [12].

2.3.1 Databáza NoSQL

Tento pojem bol prvý krát použitý v roku 1998 pre relačnú databázu nevyužívajúcu SQL. V roku 2009 bol znova použitý na konferencii v San Francisku, avšak populárny sa stal najmä vďaka blogerovi menom Eric Evans, ktorý neskôr opísal ambície týchto databáz slovami „Celý zmysel hľadania alternatív je ten, že potrebujeme vyriešiť problém, na ktorý sa relačné databázy nehodia" [5]. Potrebu využitia nového spôsobu uloženia dát priniesol aj projekt Web 2.0, ktorý začal svoj vývoj bez využitia produktov Oracle alebo MySQL, ktoré boli v danú dobu využívané mnohými začínajúcimi projektami. Namiesto nich použil svoje vlastné dátové úložiská využívajúce technológie Amazon's dynamo [3] a Google's Bigtabe [2] pre uloženie a spracovanie obrovského množstva dát, podobné množstvu využívanému v sociálnych komunitách alebo pri aplikáciach cloud computing.

NoSQL systémy sú taktiež často nazývané "Not only SQL" pre poukázanie podpory dotazovacích jazykov podobných SQL. Poskytujú mechanizmy pre uloženie a načítanie dát, ktoré sú modelované spôsobom odlišným od relačných databáz. Podporujú nerelačný dátový model a distribuovanú architektúru. Sú často používané pre spracovanie veľkého množstva dát a pri real-timeových webových aplikáciách.

Základné vlastnosti týchto databáz sú: jednoduchosť dizajnu, nenáročné horizontálne škálovanie clustermi počítačov (čo je problém pri relačných databázach), lepšia kontrola nad ich dostupnosťou a možnosťou pripojenia k nim. Dátové štruktúry využívané týmito databázami sú predovšetkým typu *kľúč-hodnota*, *graf*, *wide column* a *document store*. Práve tieto typy umožňujú vykonávať mnohé operácie nad uloženými dátami rýchlejšie, než tomu tak je v relačných databázach [7, 10].

2.4 Databáza Redis

Redis je nerelačná databáza, často opisovaná ako pamäťovo perzistentné uloženie typu kľúč-hodnota, avšak nejedná sa o presnú definíciu, keďže Redis je oveľa viac, než len jednoduché uloženie typu kľúč-hodnota [11]. Realita je taká, že Redis ponúka oficiálne päť dátových štruktúr:

- string,
- hash,
- lists,
- sets,
- sorted sets.

Pochopenie týchto metód uloženia dát, ako pracujú, aké operácie nad dátami ponúkajú a čo je možné pomocou nich modelovať, je kľúčom k pochopeniu databázy Redis. Redis využíva rovnaký koncept databázy, ktorý je nám už známy. Databáza obsahuje sadu dát. Typické využitie databázy je zoskupenie všetkých dát aplikácie dokopy a ponechanie ich separátne od ostatných aplikácií. Jednotlivé databázy sú identifikovateľné pomocou čísla, respektíve ich indexu. Prednastavená databáza začína číslom 0. Ak užívateľ chce zmeniť využívanú databázu, môže tak učiniť zadaním príkazu *select 1* pre zmenu na databázu s indexom 1. Redis by mal odpovedať správou OK a začiatok príkazového riadka by mal nadobudnúť podobu *redis 127.0.0.1:6379[1].*

Kým Redis je viac než úložisko typu kľúč-hodnota, v jeho jadre obsahuje každá z dátových štruktúr prinajmenšom kľúč a k nemu prináležiacu hodnotu. Je potrebné pochopiť princíp kľúčov a hodnôt pred tým, než budu uvedené nasledujúce dostupné možnosti. Kľúče tvoria spôsob, ktorým sú dáta identifikované. Kľúče môžu mať viacero podôb, ale ako príklad je možné uviesť *users:forest*. Dá sa očakávať, že tento kľúč bude obsahovať informácie o užívateľovi *forest*. Dvojbodka v tomto prípade nehrá žiadnu rolu a je to len spôsob akým si užívateľ môže organizovať svoje kľúče.

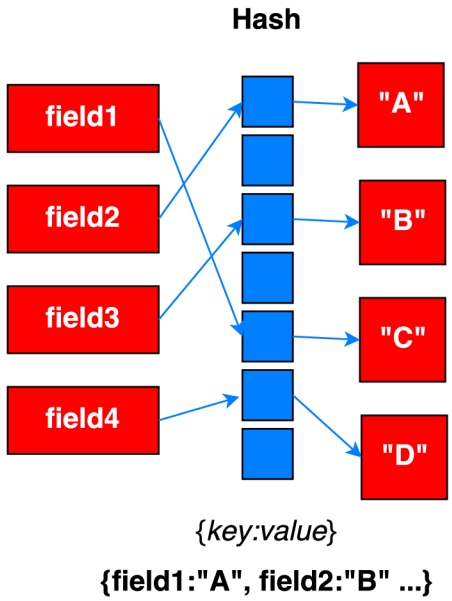
Hodnoty reprezentujú aktuálne dáta priradené ku zvolenému kľúču. Niekedy je vhodné uložiť dané dáta ako reťazce znakov, niekedy ako čísla a niekedy je vhodné uložiť ich ako serializované objekty typu JSON, XML, prípadne iné formáty. Vo väčšine prípadov Redis zaobchádza s hodnotami ako s poľom bytov a nezaujíma sa o to, čo sa v nich nachádza [9].

2.4.1 Dátový typ string

Reťazce znakov je najjednoduchší dátový typ, ktorý Redis ponúka. Jedná sa práve o základné uloženie typu kľúč-hodnota. Pri práci s týmto typom sa nám ponúkajú základné operácie, ako napríklad *strlen <key>, getrange <key> <start> <end>*, prípadne *append <key> <value>*, ktorá pripojí hodnotu k už existujúcej hodnote (v prípade že neexistuje, tak ju vytvorí). Avšak ako už bolo v tejto kapitole spomínané, Redis sa nestará o typ hodnoty priradenej ku kľúču. To znamená, že ku kľúču môžeme priradiť aj číslo, ktoré sa uloží vo formáte *integer*, no v tomto prípade operácie typu *getrange* nebudú dostupné. Namiesto nich pribudne možnosť použitia operácií nad dátovým typom *integer*, akými sú *incr, incrby, decr, decrby.* Aj vďaka tomu je tento dátový typ využiteľnejší, než sa na prvý pohľad zdá.

2.4.2 Dátový typ hash

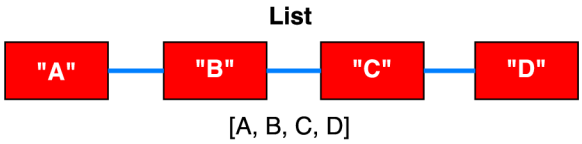
Dátový typ hash slúži ako dobrý príklad, prečo nie je vhodné nazývať a definovať Redis ako úložisko typu kľúč-hodnota. Vo viacerých smeroch sa podobá na dátový typ *string*, avšak tento dátový typ nám poskytuje väčšiu kontrolu. Pri uložení hodnoty je namiesto jednoduchej serializácii využitý hash, pre dosiahnutie presnejšej reprezentácie. Výhodou tohto prístupu je získanie alebo aktualizovanie časti dát bez potreby získania alebo zápisu celej hodnoty. A práve to môže byť nápomocné najmä z dôvodu lepšieho výkonu pri dotazovaní a väčšej čiastočnej kontroly nad dátami. Chápanie hashu z perspektívy presne ohraničených objektov, z pohľadu užívateľa, je kľúčom k pochopeniu ako to celé funguje.



Obr. 2.7: Ukážka spôsobu uloženia dátového typu hash databázy Redis

2.4.3 Dátový typ list

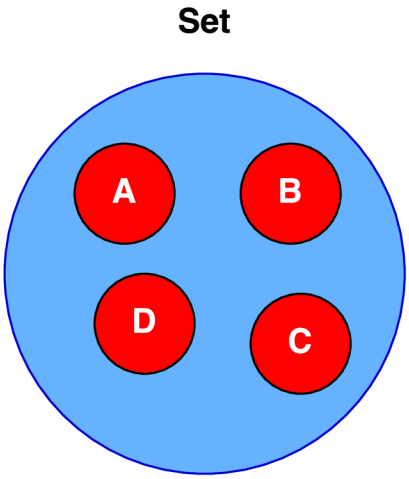
Listy nám umožňujú uložiť a manipulovať s poľom hodnôt priradených určitému kľúču. Hodnoty je možné k danému listu pridávať alebo odstraňovať. K priradeným hodnotám môžeme pristúpiť vyžiadaním si prvého alebo posledného prvku listu, prípadne môžeme manipulovať s prvkom na konkrétnom indexe. Listy spravujú poradie ich prvkov a disponujú efektívnymi operáciami založenými na indexoch. List je možné orezať tak, aby obsahoval len zvolený počet posledných pridaných hodnôt. To je možné príkazom *ltrim*, ktorého náročnosť odstránenia nadbytočných prvkov predstavuje náročnosť typu O(N), kde N reprezentuje počet odstraňovaných položiek listu.



Obr. 2.8: Ukážka spôsobu uloženia dátového typu list databázy Redis

2.4.4 Dátový typ set

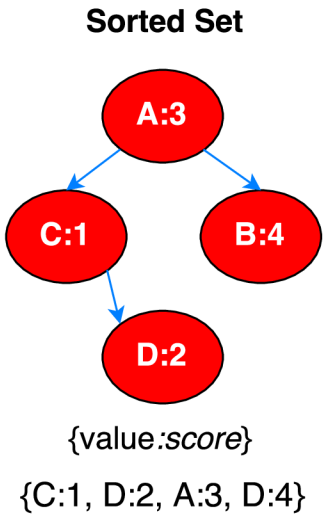
Tento dátový typ slúži na uloženie unikátnych hodnôt a poskytuje operácie založené na sadách dát ako napríklad zjednotenie. Nejedná sa o typ, ktorého dáta sú zoradené. Vhodný je najmä na označovanie alebo sledovanie akýchkoľvek vlastností hodnôt, ktorých duplicity nie je potrebné zaznamenať alebo je vyžadované aplikovať nad nimi operácie typu prienik, zjednotenie a im podobné.



Obr. 2.9: Ukážka spôsobu uloženia dátového typu set databázy Redis

2.4.5 Dátový typ sorted set

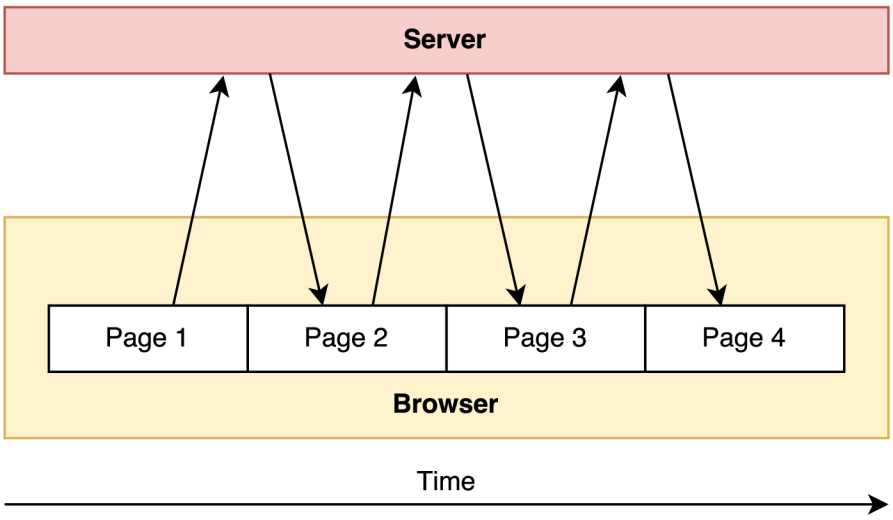
Jedná sa o posledný typ, často označovaný za najmocnejší dátový typ spomedzi ostatných ponúkaných. Sú podobné dátovému typu *set*, no navyše obsahujú skóre, ktorá umožňuje jednotlivé prvky triediť a hodnotiť podľa pridelenej celočíselnej hodnoty. Príkaz *zrank* zoradí množinu hodnôt vzostupne, čo je možné využiť v rôznych hodnotiacich systémoch.



Obr. 2.10: Ukážka uloženia dátového typu sorted set databázy Redis

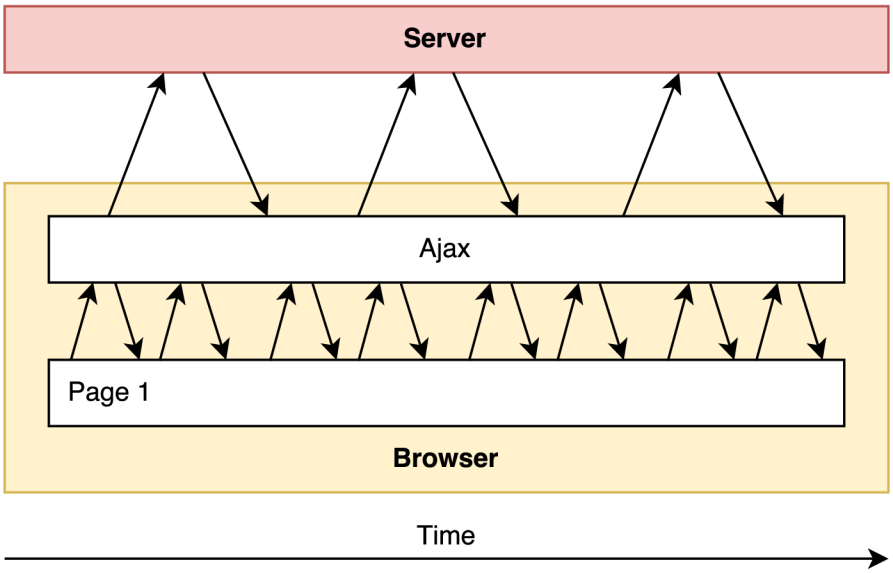
2.5 Webová technológia Ajax

Hlavným prvkom technológie Ajax je objekt *XMLHTTPRequest*. Ten umožňuje aby jazyk *JavaScrip*, využívaný pri tvorbe webových aplikácií, mohol formulovať *HTTP* dotaz a následne ho odoslať na server. Tradičné webové aplikácie vytvárajú takéto dotazy synchrónne, v spojení s udalosťou vyvolanou užívateľom, akým je napríklad kliknutie na odkaz alebo odoslanie vyplneného formulára. Odpoveďou na takýto dotaz je aktualizovaná, prípadne nová stránka, poskytnutá prehliadaču pre jej zobrazenie. Tradičný spôsob webových aplikácií je založený na zobrazení jednotlivých stránok individuálne, kedy každá stránka je načítaná zvlášť, ako jeden celok. Opísaný spôsob zobrazenia je možné vidieť na Obrázku 2.11. Jedným z problémom tohto prístupu je citeľná pauza medzi zobrazením jednotlivých stránok. Ako ďalší problém je možné uviesť povinné načítanie celej stránky, aj v prípade, že väčšia časť zobrazovaného obsahu ostáva nezmenená.



Obr. 2.11: Zobrazenie ilustruje presmerovanie na novú webovú stránku, počas ktorého užívateľ musí čakať na načítanie a zobrazenie celej stránky

Využitím *XMLHTTPRequest* je možné vytvoriť opisovaný dotaz asynchrónne, na pozadí, umožňujúc užívateľovi naďalej stránku využívať bez prerušenia či opakovaného načítania zobrazovanej stránky. Aktualizované sú len určité elementy predstavujúce odpoveď servera na vygenerovaný dotaz. Ajax pridáva webovým aplikáciam rozhranie, pomocou ktorého sa formát a funkcionalita webových aplikácií približuje klasickým aplikáciam a grafickým rozhraniam bežiacim lokálne u klienta. Ako je možné vidieť na Obrázku 2.12, pre dosiahnutie tohto výsledku je medzi webový server a zobrazovanú stránku pridaná špeciálna vrstva spracovania. Táto vrstva, často označovaná ako *Ajax Engine* alebo *Ajax Framework* zachytáva požiadavky od klienta a na pozadí spracováva komunikáciu so serverom, zatiaľ čo užívateľ pokračuje v prezeraní, klikaní a písaní v aktuálnej stránke.



Obr. 2.12: Ilustrácia využitia technológie Ajax, ktorá umožňuje zmenu časti zobrazovanej webovej stránky bez nutnosti načítania celej stránky odznova

Vhodným príkladom využitia spomínanej technológie je návrh možných výsledkov ponúkaných vyhľadávacím nástrojom Google. Tento nástroj je opisovanou technológiou rozšírený o zobrazenie ponuky navrhovaných výsledkov už pri písaní kľúčových slov a ich obsah je aktualizovaný pri každej zmene kľúčového slova [1].

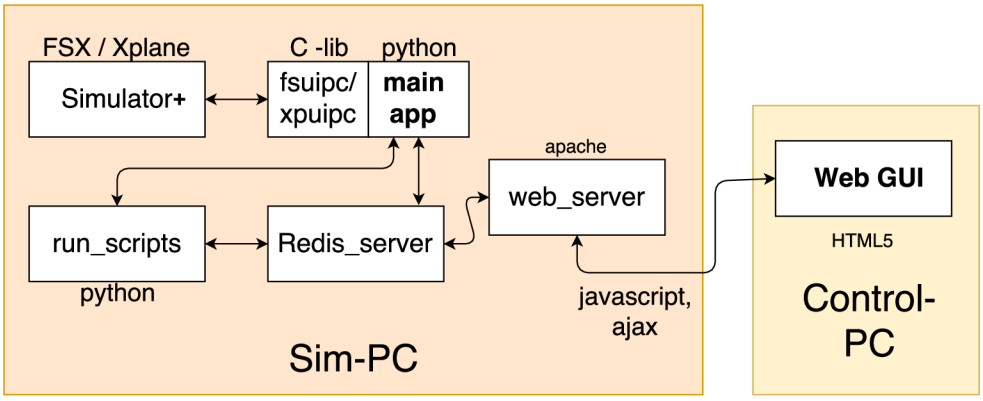
Kapitola 3

Nástroj pre správu scenárov

Táto kapitola sa zaoberá popisom postupného vývoja štruktúry výsledného systému a použitia webových technológií. Tie sú systémom využívane pri komunikácii jednotlivých prvkov a tiež pri zobrazení grafického užívateľského rozhrania.

3.1 Štruktúra výslednej aplikácie

Návrh štruktúry systému patril medzi prvé kroky, ktorými bolo nutné sa po definovaní požiadavkov zaoberať. Na Obrázku 3.1 je možné vidieť prvotný návrh štruktúry aplikácie. Tento návrh značí využitie simulačného počítača označovaného *Sim-PC*, zodpovedajúceho za spustenie a správu všetkých požadovaných častí systému. Výnimkou je len samotné webové užívateľské rozhranie využívajúce pre svoje zobrazenie kontrolný počítač.



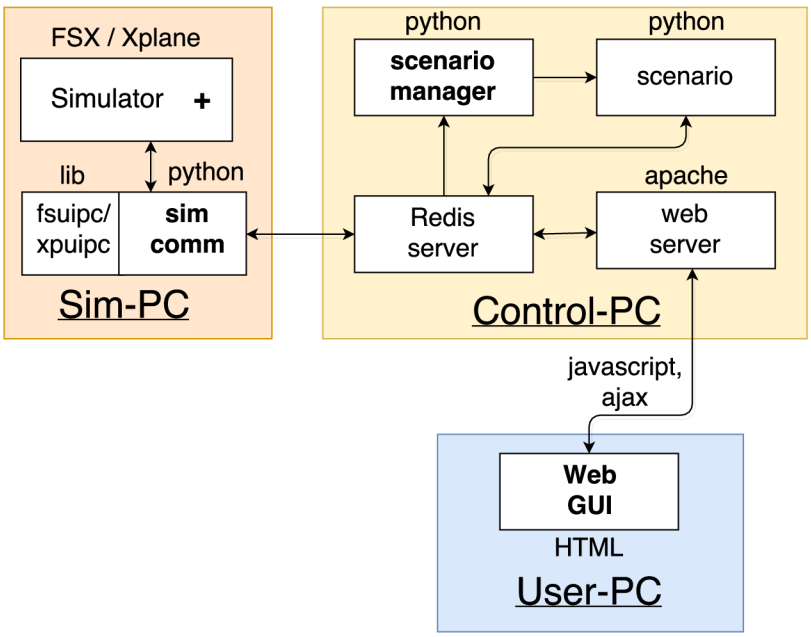
Obr. 3.1: Prvotný návrh štruktúry výsledného systému, využívajúci simulačný počítač na spustenie všetkých častí vytvoreného systému

Slabinou prvotného návrhu je využitie simulačného počítača na správu všetkých prvkov systému, čo má za príčinu aj nutnosť uloženia vytváraných scenárov práve v tomto počítači. To môže znepríjemňovať prácu so systémom, keďže je nutné pri výbere scenárov, ich tvorbe a úprave využívať počítač, kde je spustený aj simulátor a jeho využitie musí byť pri tejto činnosti pozastavené. Ďalším negatívnym vplyvom je komplikovanosť a komplexnosť hlavnej aplikácie označovanej *main app*, ktorá zabezpečuje komunikáciu so simulátorom, ale aj spúšťanie a správu scenárov. Tieto procesy by mali za následok nadbytočné zaťaženie simulačného počítača, čo bolo vyhodnotené ako výrazne negatívny a nechcený jav.

Postupným implementovaním jednotlivých častí systému sme získavali lepší pohľad na daný problém. To nás inšpirovalo k rozdeleniu hlavnej aplikácie na dve menšie. Dôsledkom tejto zmeny boli vytvorené dve aplikácie, z ktorých každá má svoju vlastnú, jasne danú úlohu. Táto zmena umožňuje väčšiu kontrolu nad systémom a tiež možnosť oddelenia správy scenárov na zvlášť počítač. Na tento počítač, označovaný *Control-PC*, bol postupne presunutý aj webový server. V dôsledku toho bolo umožnené umiestnenie aj samotných scenárov na daný počítač. Posledným takto presunutým prvkom je server Redis. Všetky tieto zmeny, ako aj výslednú podobu návrhu, je možné vidieť na Obrázku 3.2.

Porovnaním prvotného návrhu zobrazenom na Obrázku 3.1 a finálneho návrhu umiestnenom na Obrázku 3.2 je možné sledovať posun a skvalitnenie výsledného systému, ku ktorému došlo počas jeho vývoja. Schéma výslednej štruktúry bola navrhnutá pre optimálny chod celého systému. Nakoľko je aplikácia rozdelená do troch častí, teda troch samostatných počítačov, toto rozdelenie nie je nutné dodržať, keďže sa jedná len o návrh rozloženia častí systému, ktorý je aplikáciou podporovaný. Jednotlivé rozdelenie nie je nutné dodržať a spôsob implementácie výsledného systému nám umožňuje ľubovoľné umiestnenie spomínaných základných častí systému. To znamená, že úplná funkcionalita vytvorenej aplikácie je dosiahnuteľná aj s využitím len jedného počítača.

Medzi hlavné požiadavky výslednej aplikácie patrí snaha o čo najnižšie zaťaženie počítača, na ktorom je letecký simulátor spustený. Splnenie tejto požiadavky je podstatné najmä z dôvodu, že oba využívané letecké simulátory vyžadujú pre svoj plynulý chod a čo najreálnejšie grafické zobrazenie vysoké požiadavky technického vybavenia počítača, na ktorom budú spustené. Práve preto náš návrh rozloženia jednotlivých prvkov oddeľuje simulátor od ostatných častí systému, akými sú webový server a databáza Redis.



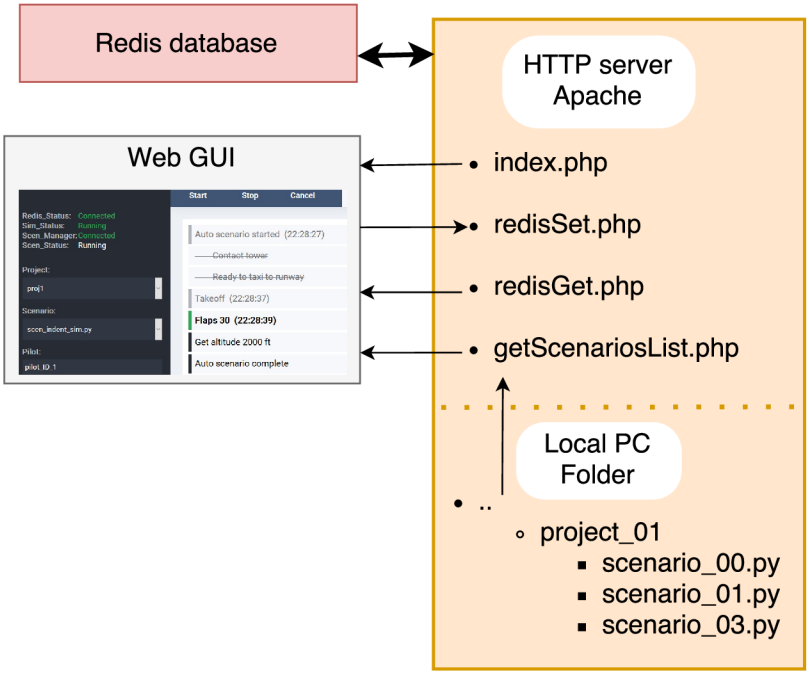
Obr. 3.2: Schéma rozloženia výsledného systému využívajúca až tri počítače

Simulátor je spúšťaný na počítači, ktorý je v návrhu štruktúry pomenovaný *Sim-PC*. Simulačný počítač obsahuje len jednoduchú komunikačnú aplikáciu, ktorá opakovane číta požadované premenné simulátora a odosiela ich na server Redis. Ten sa nachádza na kontrolnom počítači, označený ako *Control-PC*. Táto časť systému je taktiež zodpovedná za

správu scenárov. Scenáre, ktoré sú spravidla na tomto zariadení aj uložené, sú sprístupnené pre webový server a po výbere požadovaného scenára sú následne správcom scenárov aj spustené. Posledným prvkom tejto časti systému je aj samotný webový server, pomocou ktorého sú užívateľovi zobrazované a sprístupnené potrebné dáta a tiež ovládacie prvky aplikácie.

3.2 Webový server

Webový server tvorí jadro grafického rozhrania a s jeho využitím sú užívateľovi zobrazované potrebné informácie a poskytnuté vhodné prvky na obsluhu aplikácie. Pomocou neho je aj riadený chod správcu scenárov, predaním užívateľom zadaných dát o zvolenom scenári práve ich správcovi, ktorý ich následne spustí. Pre korektné a prehľadné zobrazenie sú využité technológie *HTML* a *CSS*. Pomocou nich je užívateľovi zobrazená hlavná webová stránka predstavujúca grafické užívateľské rozhranie. Pre získanie korektných dát, zobrazujúcich sa na tejto stránke, sa na strane servera využíva technológia *PHP*. Využitím tejto technológie boli vytvorené skripty, zabezpečujúce pripojenie do databázy Redis. Pomocou nich je možné sprístupniť dáta, ktoré sú požadované grafickým rozhraním. Týmto prenosom sú získané zobrazujúce sa údaje ako napríklad aktuálna výška a rýchlosť, ale aj aktuálny krok spusteného scenára. Takto vytvorené spojenie je využívané webovým serverom aj v opačnom smere, kedy sú dáta získané webovým serverom od užívateľa odosielané do databázy. Medzi tieto dáta patrí názov zvoleného projektu, scenára a ID pilota. Jazyk *PHP* je využívaný aj na prehliadanie lokálnej zložky počítača, v ktorej sú uložené jednotlivé projekty a nim prislúchajúce scenáre. Vďaka tomu sú užívateľovi cez webové grafické rozhranie zobrazené a na výber ponúknuté projekty, zodpovedajúce hierarchii ich uloženia v danom počítači. Na Obrázku 3.3 je znázornený spôsob využitia opisovaného webového servera a ním riadených a komunikujúcich komponentov výsledného systému.



Obr. 3.3: Princíp využitia servera Apache

Pre komunikáciu webového servera a databázy Redis bol zvolený jazyk *PHP* z dôvodu možnosti implementácie jednoduchých univerzálnych programov, ktorým budú v prípade potreby predané informácie o požadovaných dátach. Tieto programy podľa potreby prijaté dáta zapíšu na server Redis alebo v prípade požiadavky na ich čítanie tak učinia a získané dáta webovému serveru sprístupnia.

Pripojenie na server Redis nie je samotným jazykom *PHP* podporované, a preto bolo zvolené využitie verejne dostupného komunikačného klienta pre tento server, ktorý je určený práve pre jazyk *PHP*. Spomedzi viacerých dostupných klientov bol vybratý klient *Predis*, ktorý patrí medzi odporúčaných klientov aj samotnými tvorcami využívanej databázy.

Ako webový server bol zvolený HTTP server *apache*, ktorý svojimi vlastnosťami pre dané využitie v našom systéme najviac vyhovoval. Pre jeho správu bola využitá voľne dostupná aplikácia *xampp*. Tá nám umožňuje jeho spustenie lokálne na počítači, kde je spustený aj server Redis. V prípade, že simulačný počítač je priamo pripojený do systému alebo je spustený na rovnakom počítači ako aj webový server, pripojenie k sieti internet nie je vyžadované, a aj napriek tomu je systém plne funkčný.

Viacero jednotlivých prvkov grafického rozhrania sa počas chodu celého systému pravidelne mení. Pre ich aktuálne zobrazenie využíva webový server technológie *JavaScript* a *Ajax*. Technológia *Ajax* umožňuje dynamicky meniť obsah zobrazovanej stránky, bez potreby jej kompletného znovunačítania. Bližší popis tejto technológie sa nachádza v kapitole 2.5. Využitie týchto technológií umožňuje tvorbu užívateľského rozhrania, ktoré užívateľovi poskytuje vždy aktuálne informácie o stave simulátora, letu a spusteného scenára s minimálnym oneskorením.

3.3 Webové užívateľské rozhranie

Pre interakciu s vytvorenou aplikáciou bolo zvolené webové užívateľské rozhranie, z dôvodu možnosti jeho zobrazenia a obsluhy na rôznych vzdialených zariadeniach. Tento krok zjednodušuje využívanie aplikácie a tým aj riadenie spustenie simulácie, keďže jeho obsluha je možná napríklad z tabletu. To umožňuje využitie našej aplikácie za zvýšeného pohodlia, bez nutnosti nachádzať sa v bezprostrednej blízkosti simulačného počítača.

Grafické rozhranie prešlo viacerými úpravami. Jeho hlavnú časť reprezentujúcu výslednú podobu webového grafického rozhrania je možné vidieť na Obrázku 3.4.

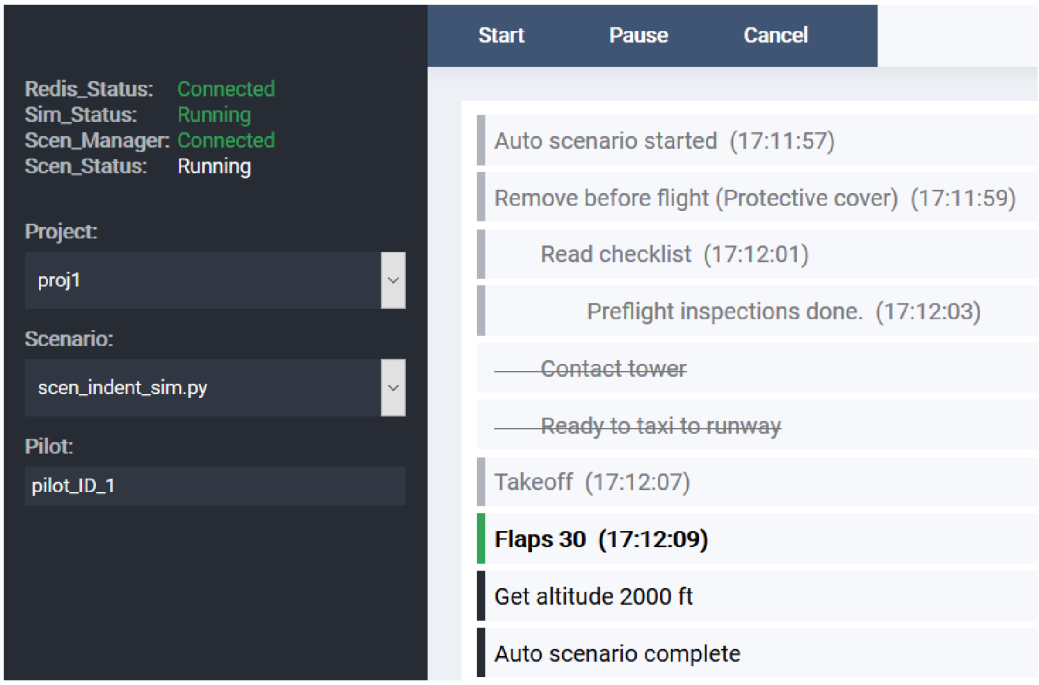
Dôvody zmien vznikali najmä v dôsledku rozšírenia funkcionality výsledného systému. Na začiatku práce na tomto projekte boli v spolupráci s firmou Honeywell definované základné prvky rozhrania, ktoré sú nevyhnutné pre chod výsledného systému. Prvotný návrh obsahoval možnosť výberu scenára, jeho spustenie a základné zobrazenie priebehu jeho spustenia. Prvky výberu projektu a následne jemu odpovedajúcemu scenára je možné vidieť v ľavej časti užívateľského rozhrania. Po úspešnom otestovaní takéhoto pomerne jednoduchého systému došlo k postupnému rozšíreniu vizualizácie prvkov, ktoré reprezentujú stav pripojenia jednotlivých komponentov. Umiestnené su v ľavom hornom rohu grafického rozhrania. Pomocou nich sú užívateľovi poskytnuté informácie o aktuálnom stave komunikačnej aplikácie, správcu scenárov, pripojenia na server Redis a spustenia simulátora. Pre lepšiu identifikáciu pilota riadiaceho letecký simulátor bola pridaná možnosť pre zadanie jeho mena alebo iného formátu označenia, ktorá umožňuje jeho neskoršiu identifikáciu.

Tlačidlo *Start*, ktoré je umiestnené v hornej lište rozhrania, umožňuje spustenie zvoleného scenára. Možnosť riadenia scenára bola neskôr rozšírená o možnosť predčasného ukončenia scenára (tlačidlo *Cancel*), prípadne dočasného pozastavenia jeho vykonávania (tlačidlo *Pause*). Posledným tlačidlom, ktoré však na obrázku reprezentujúcom základné

zobrazenie grafického rozhrania nie je vidieť, je tlačidlo *Pause Sim*. Jeho podobu a umiestenie je možné vidieť v Prílohe A.1. Ako už z jeho názvu vyplýva, jeho úlohou je pozastaviť samotnú simuláciu letu.

Príjemným no nie nevyhnutným rozšírením je vizualizácia jednotlivých krokov spusteného scenára, ktorú je možné vidieť v hlavnej časti užívateľského rozhrania. Prvotné riešenie umožňovalo výpis jednotlivých krokov, ktoré boli scenárom úspešne vykonané. Rozhodli sme sa rozšíriť túto možnosť o zobrazenie všetkých krokov scenára a ich vizualizáciu zobrazujúcu už vykonané kroky, aktuálne vykonávaný krok i nasledujúce kroky. Výsledok tohto zobrazenia je možné pozorovať na obrázku zobrazujúcom užívateľské rozhranie. Sivou farbou sú značené kroky, ktoré už boli vykonané. Kroky, ktoré neboli vykonané, ale boli preskočené nesplnením určitých zadefinovaných podmienok, sú tiež značené sivou farbou, ale sú navyše aj preškrtnuté. Aktuálny krok je označený zeleným návestím a kroky, ktoré budú nasledovať po ňom, sú označené čiernym návestím. Možnosť takto podrobného zobrazenia krokov bola získaná spracovaním a analýzou príslušného scenára tesne pred jeho spustením. Toto predčasné spracovanie sme sa rozhodli využiť aj na kontrolou zanorenia príslušných častí scenára prevažne v podmienkových blokoch, čo sa odráža odsadením príslušných krokov pri ich zobrazení v užívateľskom rozhraní.

Poslednou úpravou výpisu jednotlivých krokov bolo pridanie času dosiahnutia daného kroku. To nám zvyšuje prehľad o tom, akým spôsobom je daný scenár vykonávaný. Je možné pozorovať, v ktorých častiach scenára mal pilot určité problémy a naopak, ktoré kroky zvládol rýchlo a bez problémov.

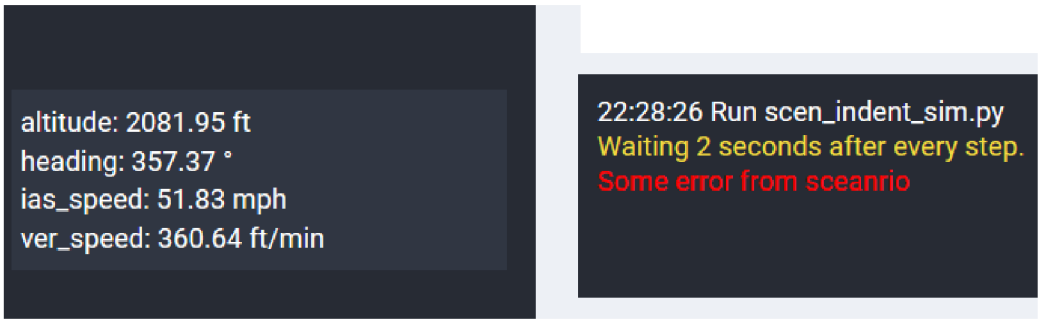


Obr. 3.4: Základné prvky užívateľského rozhrania

Prvky, ktoré boli doteraz opísané, súviseli najmä so samotnými scenármi. Keďže scenáre bezprostredne súvisia so samotným simulátorom, rozhodli sme sa zvýšiť informovanosť užívateľa aplikácie aj v tomto smere. Boli zvolené štyri premenné, ktoré patria medzi základné parametre reprezentujúce prebiehajúci let v spustenom simulátore. Tieto premenné sú zobrazené a pravidelne aktualizované v spodnej časti grafického rozhrania a je možné

ich vidieť v ľavej časti Obrázka 3.5. Informujú o aktuálnej dosiahnutej výške (altitude), v ktorej sa pilot nachádza, kurze letu (heading), indikovanej vzdušnej rýchlosti (IAS) a vertikálnej rýchlosti (VER). Tieto údaje slúžia užívateľovi najmä pre kontrolu spojenia, správneho chodu simulátora, ale vytvárajú aj základnú predstavu o aktuálnom stave a smerovaní lietadla.

Posledným prvkom grafického rozhrania je informačná konzola, ktorá dopĺňa informácie o spustených scenároch, ich chybách a dôvodoch ukončenia. Vidieť ju je možné v pravej časti Obrázka 3.5. Ako príklad jej využitia je možné uviesť spustenie scenára, kedy sú práve do tejto konzoly odoslané informácie o času a názve spusteného scenára. Ak spustený scenár bude z dôvodu chyby v tele jeho programu predčasne ukončený, bude správa nesúca dôvod ukončenia odoslaná práve do informačnej konzoly. Dáta do tejto konzoly môžu byť odosielané každým prvkom pripojeným k serveru Redis. To znamená, že túto možnosť môže využiť aj samotný tvorca scenárov v prípade potreby informovania alebo upozornenia užívateľa na určitý fakt. K dispozícii sú tri typy správ, ktoré je možné konzolou zobraziť, a to informačná (prvý riadok, biela farba), výstražná (druhý riadok, žltá farba) a chybová (posledný riadok, červená farba).



Obr. 3.5: Vľavo informácie o stave lietadla, vpravo časť informačnej konzoly

3.4 Komunikácia a sprístupnenie dát

Najpodstatnejším prvkom komunikácie je server Redis, ktorý zastupuje NoSQL databázu. Zaisťuje prepojenie jednotlivých častí systému a umožňuje spoľahlivú a rýchlu komunikáciu aj v prípade, že sa jednotlivé prvky nachádzajú na rozličných počítačoch, ktoré sú navzájom prepojené, prípadne majú prístup k internetovému pripojeniu.

Všetká komunikácia so simulátorom a spustenou aplikáciou prebieha cez verejne dostupný doplnok *FSUIPC* [4] v prípade využívania simulátora *Microsoft Flight Simulator X* alebo *XPUIPC* [13] v prípade využitia simulátora *X-Plane*. Tento doplnok je doinštalovaný do príslušného simulátora a spolu s priloženou komunikačnou knižnicou, ktorá je využívaná aj vytvorenou aplikáciou, umožňuje, aby spustený simulátor komunikoval s rôznymi externými zdrojmi. Práve touto cestou sú získavané potrebné premenné zo simulátora, ktoré sú pomocou vytvorenej komunikačnej aplikácie odoslané na server Redis. Touto činnosťou je zabezpečená možnosť analýzy prebiehajúceho letu a jeho logovanie. Pre potreby konfigurácie a riadenia spusteného letu, je možné zapísať potrebné dáta priamo do simulátora cez už naviazané spojenie.

Informácie o priebehu letu sú využívané najmä spusteným scenárom, ktorý je taktiež pripojený na server Redis a väčšinou periodicky číta dáta, ktoré sú pre daný scenár zaujímavé. Použité môžu byť scenáre, ktoré dáta len čítajú a ukladajú ich pre potreby neskoršej

analýzy. Vytvoriť je možné aj scenáre, ktoré dáta na server iba zapisujú. Po tom, čo sú tieto dáta zapísané do databázy, komunikačná aplikácia bežiaca na pozadí leteckého simulátora, označená na Obrázku 3.2 ako *sim_comm*, zaregistruje túto zmenu a zapíše dané dáta priamo do simulátora. Takéto scenáre môžu byť použité najmä po spustení simulátora pred začatím konkrétneho letu, teda testu, a slúžia na upresnenie konfigurácie daného leteckého prostriedku, ako napríklad úprava stavu paliva, ovládacích prvkov a im podobné.

Ďalšou časťou systému, s ktorou je taktiež nutné zabezpečiť komunikáciu a jej pripojenie, je webový server. Ten využíva čítanie dát z databázy, najmä pre ich vizualizáciu a informovanie užívateľa, v akej aktuálnej výške sa lietadlo nachádza, kam smeruje, no taktiež získava a zobrazuje informácie o kroku scenára, v ktorom sa aktuálne spustený scenár nachádza. Využívaný je aj zápis dát do databázy, z dôvodu prenosu informácií o zvolenom projekte, scenári a ID pilota.

Posledným prvkom komunikujúcim s databázou je správca scenárov označený na Obrázku 3.2 *scenario_manager*. Ten využíva toto pripojenie len na získanie informácií o zvolenom scenári, pilotovi a požadovaný stav zvoleného scenára, čiže jeho spustenie, pozastavenie, prípadne ukončenie.

Kapitola 4

Program pre sprístupnenie dát a riadenie letu

Výsledný systém je tvorený z troch hlavných častí. Jedná sa o správcu scenárov, komunikačný program a samotné webové rozhranie. Táto kapitola sa zaoberá popisom prvých dvoch programov, ich činnosti, možnosti využitia a spôsobu konfigurácie.

4.1 Správca scenárov

Hlavnou úlohou správcu scenárov je zistenie, ktorý scenár bol užívateľom zvolený, spustenie zvoleného scenára a jeho následná kontrola, v ktorej časti, respektíve kroku, sa aktuálne nachádza. Pre implementáciu tohto správcu bol zvolený programovací jazyk *python* vo verzii *3.6*. Ten využíva pre komunikáciu s databázou doplnok jazyka *python* s názvom *redis-py* [8], považovaný a odporučaný aj samotnou firmou Redis, za vhodný nástroj pre komunikáciu s ich serverom.

Podľa nami navrhovaného rozloženia jednotlivých komponentov systému zobrazených na Obrázku 3.2, by mal byť tento správca spustený na kontrolnom počítači spolu s webovým serverom a databázou Redis. Ako už bolo spomínané, rozloženie týchto komponentov nie je nutné dodržať, avšak hlavnou podmienkou korektného spustenia zvolených scenárov je ich uloženie v počítači, na ktorom bude správca scenárov spustený.

Jednotlivé scenáre je možné triediť a usporiadať v zložkách reprezentujúcich určitý projekt. Pri prístupe a zobrazení jednotlivých scenárov správca scenárov rovnako ako webový server predpokladá zoznam zložiek s názvami projektov a v nich uložené jednotlivé scenáre. Iný spôsob uloženia, akým je napríklad využitie ďalších podzložiek v jednotlivých projektoch, už nie je podporovaný.

Po spustení správcu prebieha najskôr načítanie konfiguračného súboru, ktorý definuje podstatné prvky správcu scenárov, akými je napríklad lokálna cesta uloženia jednotlivých scenárov. V tomto prípade je nutné uviesť cestu ku hlavnej zložke obsahujúcej jednotlivé podzložky reprezentujúce projekty. Ďalšou možnou konfiguráciou je zvolenie interpretera, ktorým má byť požadovaný scenár spustený. Táto možnosť môže byť využitá v prípade, že vytvorené scenáre využívajú staršiu verziu jazyka *python*, no umožňuje nám aj spustenie scenára implementovaného v ľubovoľnom programovacom jazyku. V našom prípade je využitý práve už spomínaný jazyk *python*. Pre uľahčenie tvorby scenárov v tomto jazyku bola autormi práce vytvorená podporná komunikačná knižnica *comModule*, ktorá zabezpečí pripojenie scenárov k databáze Redis. Knižnica uľahčuje tvorcovi scenárov prácu s databázou

poskytnutím funkcií pre pridanie a vizualizáciu nového kroku scenára, zápis dát do simulátora a výpis hlášok do informačnej konzoly webového grafického rozhrania. Úlohu tejto knižnice by tvorca scenárov využívajúcich iný programovací jazyk musel sám nahradiť.

Po úspešnej konfigurácii správcu nastáva kontrola dostupnosti serveru Redis. Aj v tomto prípade je pre korektný chod aplikácie toto spojenie nevyhnutné. V prípade, že toto spojenie nie je možné nadviazať, aplikácia počká určitú dobu a následne sa o nadviazanie daného spojenia pokúša znova. Po pripojení sa aplikácia dostáva do stavu, kedy je očakávaný vstup od užívateľa. Ten s využitím webového grafického rozhrania definuje dáta obsahujúce informácie o zvolenom projekte a zvolenom scenári príslušného projektu. Po stlačení tlačidla *Start* nachádzajúcom sa vo webovom grafickom rozhraní, je vybraný scenár správcom spustený. Správca spustí požadovaný scenár ako jeho podproces pomocou volania funkcie *Popen* z balíčka *subprocess*. Takéto spustenie scenára umožňuje správcovi scenárov kontrolovať stav, v akom sa spustený scenár nachádza. Je možné rozpoznať a signalizovať beh scenára, jeho úspešné ukončenie a tiež jeho neúspešné ukončenie z dôvodu výskytu chyby v scenári. Správca scenárov odosiela tieto informácie na webový server. Ten pomocou grafického rozhrania informuje užívateľa o stave, v ktorom sa ním zvolený scenár nachádza.

Webové grafické rozhranie disponuje aj informačnou konzolou, ktorú je možné vidieť na Obrázku 3.5. Po spustení zvoleného scenára sú do tejto konzoly odoslané správy, informujúce užívateľa o názve a čase spustenia scenára. V prípade, že scenár je chybný a nepodarí sa ho spustiť, alebo je počas jeho vykonávania nečakane ukončený, úlohou správcu je zachytiť túto chybu a upozorniť na ňu užívateľa odoslaním dôvodu pádu aplikácie do spomínanej konzoly. V prípade úspešného vykonania príslušného scenára správca opäť informuje užívateľa o úspešnom prevedení a pripraví sa, teda čaká, na spustenie ďalšieho scenára.

Správcu scenárov je možné ovládať pomocou troch tlačidiel grafického rozhrania. Tlačidlo *Start* spustí vybraný scenár. Pokiaľ je scenár spustený, stlačenie tohto tlačidla nevyvoláva žiadnu akciu. To znamená, že spustenie viacerých scenárov správcom nie je možné. Tlačidlo *Pause* pozastaví vykonávanie spusteného scenára jeho úplným pozastavením. Správca na tento úkon využíva funkciu *suspend* z voľne dostupného balíka *psutil*. Takto pozastavený scenár je možné pomocou tlačidla *Start* znova spustiť. Posledným tlačidlom je tlačidlo *Cancel* slúžiace na manuálne ukončenie spusteného scenára.

V nasledujúcej ukážke obsluhy stlačenia tlačidla *Cancel* správcom scenárov je možné vidieť, že spustený scenár je prv správcom ukončený a následne je do informačnej konzoly odoslaná chybová správa oznamujúca čas jeho ukončenia. Zmenená je aj zobrazovaná položka, označená ako *sceniario-status*, ktorá vo webovom rozhraní informuje užívateľa o aktuálnom stave scenára.

```
if((myredis.get("scenario")) == "cancel"):
  scenario.kill()
  myredis.lpush("error",
     time.strftime('%X') + " Terminate " + myredis.get("scen"))
  myredis.set("scenario-status","Terminated")
  break;
```

4.2 Komunikačná aplikácia

Predstavuje jednu z troch základných častí systému a jej cieľom je nadviazať a udržať spojenie so spusteným simulátorom, za účelom vzájomnej výmeny dát. Dáta predstavujú

všetky premenné, ktoré sú užívateľom žiadané. Komunikačná aplikácia je v schéme výslednej aplikácie zobrazenej na Obrázku 3.2 označená názvom *sim_comm*.

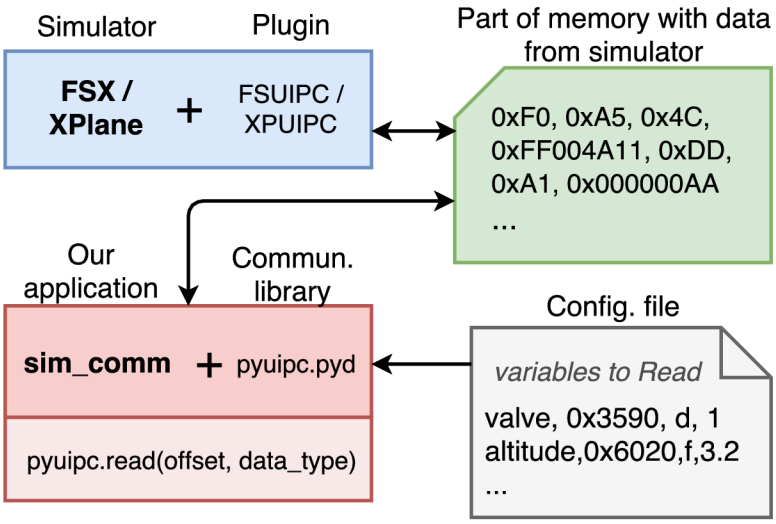
Využívané sú verejne dostupné doplnky použitých leteckých simulátorov a k nim odpovedajúce komunikačné knižnice, umožňujúce vytvorenej komunikačnej aplikácii spoľahlivé a rýchle čítanie dát zo simulátora, prípadne ich zápis. Všetky dáta, ku ktorým simulátor v spolupráci s doplnkom umožňuje pristúpiť, sú uložené v špeciálnom bloku pamäte. K daným premenným je možné pristúpiť pomocou konkrétnych offsetov, ktoré udávajú ich presnú polohu v sprístupnenom bloku pamäte. Zvolené umiestenie a veľkosť čítaných, prípadne zapisovaných dát, tvoria parametre čítacích a zápisových funkcií, dodávaných k danému doplnku simulátora, a umožňujú prácu s požadovanými dátami. Keďže komunikačná aplikácia je implementovaná taktiež v jazyku *python*, na komunikáciu je využitá príslušná knižnica *pyuipc.pyd*.

Na Obrázku 4.1 je možné vidieť princíp využitia a fungovania vytvorenej komunikačnej aplikácie. Jej úlohou je v pravidelných intervaloch čítať dáta požadované užívateľom a následne ich odoslať na server Redis. Takto získané dáta sú využívané najmä spusteným scenárom, ale časť z nich je využívaná aj samotným webovým rozhraním, ktoré tieto dáta zobrazuje. Tento princíp komunikácia vytvorenej aplikácie a doplnku simulátora funguje aj opačne a to v prípade, že scenár vyžaduje zápis určitých dát do simulátora. Scenár zvolené dáta zapíše do databázy Redis, odkiaľ sú komunikačnou aplikáciou tieto dáta prečítané a následne zapísané do pamäťového bloku. Tým je komunikačnou aplikáciou prevedené ovplyvnenie behu spustenej simulácie.

Následná ukážka kódu zobrazuje proces zapísania zvolených dát komunikačnou aplikáciou priamo do spusteného simulátora. Využitý je slovník *dictVarInfo* obsahujúci informácie o jednotlivých premenných, ktoré sú získané z konfiguračného súbora.

```
key = myredis.lpop("toWrite")
value = int(myredis.lpop("toWrite"))
try:
   varToWrite = W_varSpecif_dict[key]
   pyuipc.write([(int(varToWrite[dictVarInfo.offset], 16),
       varToWrite[dictVarInfo.data_type], value)])
```

Pri spustení tejto aplikácie dochádza k načítaniu konfiguračného súbora, ktorého obsah definuje konfiguračné premenné, akou je napríklad adresa servera Redis. Dôležitou súčasťou tohto súbora je najmä zoznam premenných, ktoré majú byť zo simulátora pravidelne čítané a odosielané na server Redis. Tento zoznam obsahuje názov premennej, pod ktorým bude zvolená premenná uložená, polohu v sprístupnenej pamäti (offset), dátový typ čítanej premennej a konštantu násobenia. Daná konštanta je určená a opísaná v dokumentácii doplnku simulátora a pre získanie korektnej hodnoty je nutné prečítané dáta ňou vynásobiť. Konfiguračný súbor obsahuje okrem zoznamu premenných určených na čítanie aj zoznam premenných, ktoré bude možné do simulátora zapísať. Formát ich zápisu odpovedá premenným určených na čítanie.



Obr. 4.1: Sprístupnenie dát zadaných konfiguračným súborom, ktoré sú s využitím komunikačnej knižnice čítané z pamäti sprístupnenej doplnkom simulátora

Po úspešnom načítaní konfiguračného súbora dochádza k snahe o pripojenie k databázy Redis. V prípade neúspechu skript čaká, pokiaľ toto spojenie bude možné zrealizovať, keďže práve to tvorí základný komunikačný prvok celého systému. Po úspešnom pripojení k databáze dochádza k snahe o pripojenie k spustenému simulátoru. Pokiaľ simulátor spustený nie je, aplikácia čaká na jeho spustenie a správa o tomto čakaní je odoslaná webovému rozhraniu, ktoré informuje užívateľa aj o pripojení a stave simulátora. Poslednou a hlavnou fázou, v ktorej sa aplikácia môže ocitnúť, je fáza zápisu a čítania dát do simulátora. Tu zotrváva až do jej ukončenia. Ak aj v tomto bode dôjde ku zlyhaniu čítania alebo zápisu dát, užívateľ je o tom opäť informovaný pomocou webového grafického rozhrania. Zvlášť signalizovaný je aj stav, kedy je simulátor spustený, spojenie nadviazané, ale samotná simulácia je pozastavená. Komunikačná aplikácia sleduje a informuje aj o tomto stave, aby užívateľ poznal dôvod, prečo informácie o aktuálnom stave leteckého dopravného prostriedku nie sú aktualizované.

V prípade úspešného spojenia so simulátorom je vyžadovaná stála dostupnosť a pravidelná aktualizácia získaných dát. Dostatočnú rýchlosť čítania dát zabezpečuje možnosť knižnice doplnku simulátora vopred si zadefinovať špeciálny list premenných, ktoré majú byť pravidelne aktualizované, teda čítané. Tento list je vytvorený a postupne rozšírený pri načítavaní konfiguračného súbora. Následne je celý zoznam premenných naraz načítaný a to jedným volaním funkcie *read*. Možnosť vopred si definovať zoznam premenných je možné rovnako využiť aj v prípade použitia funkcie *write*. Po nastavení hodnôt všetkých premenných určených k zápisu je zapisovacej funkcii predaný list týchto hodnôt a volanie danej funkcie je vykonané iba jeden krát.

Ako už bolo v tejto kapitole spomínané, na komunikáciu a prístup k bloku pamäti načítaných dát simulátora je použitá knižnica *pyuipc.pyd*. Tá nahradila nami vytvorený podprogram, ktorý bol súčasťou prvotných návrhov, bol implementovaný pomocou jazyka *C* a jeho úlohou bolo jednorázovo prečítať zvolené dáta a odoslať ich komunikačnej aplikácii. Tá takto prijaté dáta spracovala a odoslala na server Redis. Pre zápis dát do simulátora bol vytvorený podobný podprogram, ktorý zvolené dáta jednorázovo zapísal. Takto zvolený spôsob komunikácie síce fungoval, no počas procesu zdokonaľovania a zrýchlenia aplikácie

bol označený ako zbytočne náročný. Prvotnou úpravou bola zmena prístupu spustenia opisovaného podprogramu. Ten bol upravený tak, aby čítanie dát nebolo jednorázové a nebolo nutné jeho opakované volanie, ale samotný podprogram bol spustený iba raz a on sám čítal dáta periodicky. Týmto spôsobom nebolo nutné pre každé individuálne čítanie dát otvárať nové spojenie so simulátorom, ktoré muselo byť po jeho ukončení uzatvorené, ale bolo možné udržať prvotné spojenie a využívať ho počas celej doby komunikácie. Touto úpravou bol dosiahnutý výrazne rýchlejší prenos získaných dát, avšak toto riešene stále obsahovalo nutnosť podprogramu transformovať a následne odoslať získané dáta komunikačnej aplikácii. Rozhodli sme sa odstrániť tento proces komunikácie odosielaním získaných dát na server Redis už samotným podprogramom. V tomto prípade by nebola nutná komunikácia medzi podprogramom a komunikačnou aplikáciou. Tá by daný podprogram len spustila, kontrolovala a na server Redis by boli odosielané len dáta informujúce o spustení simulátora a korektného pripojenia. Počas implementácii opisovaného riešenia vznikol problém pri pripojení podprogramu priamo na databázu. Problémom je využitie jazyka *C* a jeho pripojenia na server Redis. Ten na svojich webových stránkach síce uvádza podporu aj jazyka *C*, no preklad príslušných knižníc nie je na operačnom systéme Windows možný. Na základe tohto zistenia sme sa rozhodli nahradiť komunikačnú knižnicu jazyka *C* už spomínanou knižnicou *pyuipc.pyd*, ktorá je využívaná aj v aktuálnej verzii. Využitie tejto knižnice umožňuje samotnej komunikačnej aplikácii získať potrebné dáta zo simulátora, tie spracovať a následne odoslať na server Redis bez nutnosti výmeny týchto dát s podprogramom ako to bolo doteraz. Nevýhodou tohto riešenia a dôvodom, prečo práve ono nebolo prvou voľbou počas návrhu aplikácie, je vytvorenie knižnice *pyuipc.pyd* pre verziu jazyka *python 2.7*. Závislosť na staršej verzii jazyka *python* sa podarilo odstrániť pomocou programu *cx_freeze*, ktorý zo skriptu jazyka *python* vytvoril spustiteľný (exe) program a tým umožnil jeho spustenie bez potreby inštalácie a prítomnosti jazyka *python 2.7*.

4.3 Spôsob konfigurácie správcu scenárov a komunikačnej aplikácie

Pre zvýšenie použiteľnosti výsledného systému bol už počas jeho vývoja kladený dôraz na to, aby bol modulárny a tiež vysoko konfigurovateľný. Ako už bolo spomínané, návrh rozloženia jednotlivých komponentov systému zobrazený na Obrázku 3.2 nie je nutné dodržať. Náš návrh zobrazuje využitie až troch zariadení najmä pre zníženie záťaže simulačného počítača a pohodlnej obsluhy webového rozhrania napríklad z tabletu. Výsledný systém je možné nakonfigurovať aj pre potreby spustenia všetkých troch základných častí systému na jednom zariadení.

Vyššie opísané úpravy a úpravy ním podobné je možné realizovať z dôvodu existencie konfiguračného súboru. Ten je aplikáciami pri ich štarte načítaný a jednou z hlavných informácií tohto súbora je adresa servera Redis. Tá je definovaná pre obe hlavné aplikácie zvlášť, keďže je predpoklad ich spustenia na dvoch rozličných počítačoch. Uvedená je aj lokálna cesta k zložke obsahujúcej projekty a im patriace scenáre. Táto informácia je využitá správcom scenárov pri ich spúšťaní. Všetky tieto konfigurovateľné premenné sa nachádzajú v prvej časti spomínaného súboru. Každá takáto časť je oddelená kľúčovým slovom reprezentujúcim ukončenie danej časti.

Obe nasledujúce časti súboru obsahujú zoznam premenných a im priradených dát. Prvá časť obsahuje zoznam premenných simulátora, ktoré majú byť aplikáciou čítané a sprístupnené pre ich možné použitie v scenároch. Druhá časť obsahuje rovnaký zoznam, ale ten

definuje premenné, ktoré je možné v simulátore meniť, teda zapísať požadovanú hodnotu na príslušné miesto bloku pamäte. Samotný záznam sa skladá z názvu, ktorý bol užívateľom priradený k premennej. Tá sa nachádza na offsete, ktorý je tiež zadaným užívateľom. K odpovedajúcim dátam daného offsetu je následne možné pristúpiť práve pomocou zvoleného mena. Offset predstavuje konkrétnu pozíciu zvolených dát v bloku pamäti poskytovanej doplnkom simulátora. Zoznam pozícií ponúkaných premenných je dostupný v dokumentácii priloženej k príslušnému doplnku leteckého simulátora.

Ďalej je nutné zadať dátový typ premennej. To predstavuje počet bitov, ktoré majú byť na danej pozícii v pamäti čítané, prípadne zapísané. Typ konkrétnej premennej je uvedený spolu s jej offsetom v spomínanej dokumentácii. Posledný prvok záznamu predstavuje konštantu, ktorou musí byť hodnota, získaná zo simulátora, vynásobená. Tento krok je nutný v dôsledku uloženia dát vyhovujúcemu príslušnému doplnku, ktoré však neodpovedá jednotkám využívaným v leteckom priemysle.

V nasledujúcej časti je možné vidieť ukážku konfiguračného súbora pomenovaného *config.txt*. Formát tohto súbora bol navrhnutý autorom práce tak, aby výsledný užívateľ mohol jednoduchou cestou zmeniť adresu servera Redis alebo pridať nové premenné, ktoré majú byť čítané.

```
# ------------ Definition of CONFIG variables ------------
# NAME , OFFSET , SIZE , CONSTANT

   redis_host_scenMan = localhost
   redis_host_simCom = 192.168.0.102
   redis_port = 6379
   scen_interpreter = python
   scen_path = C:\simCom\scenarios

end_of_config

# ------------Definition of SIM variables to Read------------
# NAME , OFFSET , SIZE , CONSTANT

   altitude,0x6020,f,3.2808399
   ispaused, 0x264, b , 1

end_of_R_vars

# ------------Definition of SIM variables to Write------------

   parking_brake, 0xbc8, H, 32767
   pause, 0x0262, b , 1

end_of_W_vars
```

Snaha o načítanie konfiguračného súbora je prvým krokom oboch aplikácií. V prípade, že tento súbor neexistuje, je táto situácia riešená vygenerovaním nového konfiguračného súboru. Tento preddefinovaný súbor obsahuje všetky základné prvky, ktoré sú potrebné pre správne fungovanie systému. Následne je možné tento súbor upraviť podľa vlastných potrieb.

4.4 Použitie výslednej aplikácie v praxi a jej vyhodnotenie

Jedným z prípadov využitia výsledného systému je jeho použitie počas voľnočasového využitia leteckého simulátora. Užívateľovi je umožnené spustenie simulácie aj vytvorenej aplikácie s využitím len jedného počítača. V tomto prípade je možné použitie scenárov pre úpravu svojho voľného letu a prispôsobenie si poveternostných podmienok, prípadne navodenie náhodných porúch dopravného prostriedku.

Druhým spôsobom použitia výsledného systému je jeho nasadenie v prostredí, kde dochádza k testovaniu elektrotechnického vybavenia, prípadne zručností pilota. Práve v tomto použití je využité rozdelenie komponentov systému na jednotlivé počítače. Vďaka tomu môže osoba, ktorá je zodpovedná za priebeh testov, pohodlne spúšťať a riadiť príslušné scenáre. Tento spôsob použitia výsledného systému je využívaný aj vo firme Honeywell.

V oboch prípadoch použitia je nutné zabezpečiť pravidelnú aktualizáciu premenných, ktoré sú získané zo simulátora a následne odoslané na server Redis. Minimálna frekvencia aktualizácie týchto dát bola po konzultácii s externým konzultantom stanovená na 20 Hz. Využitím výsledného systému je pri pravidelnom čítaní štyridsiatich premenných možné dosiahnuť frekvenciu aktualizácie o výške približne 45 Hz. Túto rýchlosť je možné dosiahnuť aj v prípade, že server Redis nie je umiestnený priamo na počítači, kde je simulácia spustená.

Kapitola 5

Záver

Výsledkom tejto práce je aplikácia, pomocou ktorej je možné spustiť príslušný scenár letu a sledovať proces jeho vykonávania. Zvolený scenár umožňuje upraviť priebeh letu a konfiguráciu použitého leteckého prostriedku, prípadne zaznamenávať priebeh spustenej simulácie. Základnú funkcionalitu dopĺňa informačná konzola, prednačítanie všetkých krokov zvoleného scenára a ich vizualizácia rozlišujúca preskočené, vykonané a budúce kroky. Na základe týchto vlastností vytvorenej aplikácie môže užívateľ spustiť požadovaný scenár, a tak jednoduchým spôsobom upraviť priebeh letu, prípadne jeho priebeh zaznamenať.

V práci sú opísané použité technológie ako *Ajax* a *NoSQL* databázy, pomocou ktorých bolo možné zrealizovať daný projekt. S navrhnutým rozložením príslušných častí systému a využitím kontrolného počítača nedochádza k nadbytočnému zaťaženiu počítača, na ktorom je spustený simulátor, čo taktiež patrí medzi požiadavky na výslednú aplikáciu.

Výsledný systém bol počas svojho vývoja testovaný vo firme Honeywell. Po ukončení jeho vývoja došlo k jeho odovzdaniu a schváleniu splnenia všetkých požiadaviek, určených firmou na začiatku, čo umožňuje jeho využitie v praxi. Práca sa zúčastnila aj konferencii Excel@Fit 2017.

Možnosť rozšírenia vytvorenej aplikácie sa odvíja aj od jej dlhodobejšieho používania v praxi. V aktuálnej podobe prichádza do úvahy pridanie možnosti spustenia viacerých scenárov súčasne a ich prehľadná, separátna vizualizácia. Vhodným rozšírením je aj možnosť úpravy konfiguračného súbora a uloženie nového scenára pomocou webového rozhrania.

Literatúra

[1] Ballard, P.; Moncur, M. G.: *Sams teach yourself Ajax, JavaScript, and PHP all in one*. Indianapolis : Sams, 2009, ISBN: 9780672329654.

[2] Chang, F.; Dean, J.; Ghemawat, S.; aj.: Bigtable: A Distributed Storage System for Structured Data. https://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf, [Online; navštíveno 21.03.2017].

[3] DeCandia, G.; Hastorun, D.; Jampani, M.; aj.: Dynamo: Amazon's Highly Available Key-value Store. http://s3.amazonaws.com/AllThingsDistributed/sosp/amazon-dynamo-sosp2007.pdf, [Online; navštíveno 21.03.2017].

[4] Dowson, P.: FSUIPC4, plugin MS Flight Simulator X. http://www.schiratti.com/dowson.html, [Online; navštíveno 05.04.2017].

[5] Evans, E.: NoSQL: What's in a name? http://blog.sym-link.com/2009/10/30/nosql_whats_in_a_name.html, [Online; navštíveno 21.03.2017].

[6] FlyAway: XPUIPC, plugin XPlane. https://flyawaysimulation.com/knowledge/page/29/microsoft-flight-simulator-x/, [Online; navštíveno 26.4.2017].

[7] Holubová, I.; Kosek, J.; Minařík, K.; aj.: *Big Data a NoSQL databáze*. Grada, 2015, ISBN: 9788024754666.

[8] McCurdy, A.: redis-py. https://github.com/andymccurdy/redis-py, [Online; navštíveno 05.04.2017].

[9] Redmond, E.; Wilson, J.: *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*. The Pragmatic Programmers, 2013, ISBN: 9781934356920.

[10] Rychlý, M.; Kolár, D.: NoSQL databáze. http://www.fit.vutbr.cz/~rychly/public/docs/slides-nosql-databases/slides-nosql-databases.pdf, [Online; navštíveno 21.03.2017].

[11] Seguin, K.: The Little Redis Book. http://openmymind.net/redis.pdf, [Online; navštíveno 21.03.2017].

[12] Strauch, C.: Building scalable databases. http://www.christof-strauch.de/nosqldbs.pdf, [Online; navštíveno 20.03.2017].

[13] TOSI: XPUIPC, plugin XPlane.
http://www.tosi-online.de/XPUIPC/What_is_it.html, [Online; navštíveno 05.04.2017].

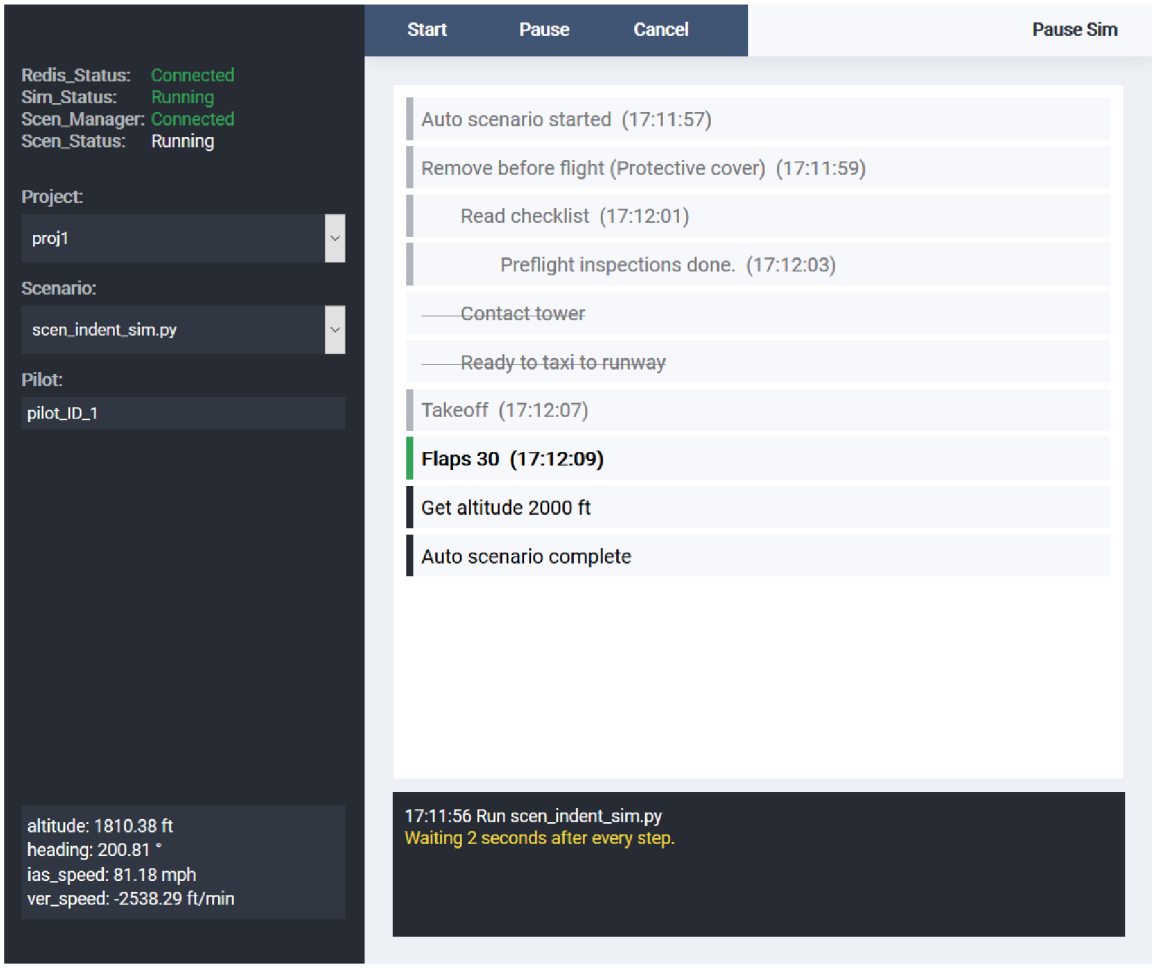
[14] West, J. V.; Lane-Cummings, K.: *Microsoft Flight Simulator X For Pilots: Real World Training*. Wiley Publishing, 2007, ISBN: 9780764588228.

[15] XPlane: About X-Plane.
http://www.x-plane.com/files/manuals/X-Plane_10_Desktop_manual.pdf, [Online; navštíveno 1.5.2017].

Prílohy

Príloha A

Kompletný vzhľad webového grafického užívateľského rozhrania



Obr. A.1: Kompletné zobrazenie webového grafického rozhrania

Príloha B

Plagát