České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů

Systém pro správu 3D obsahu

3D Content Management System

BAKALÁŘSKÁ PRÁCE

Vypracoval: Alexander Mensák
Vedoucí práce: Ing. Jiří Doležal
Rok: 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Mensák** Jméno: **Alexander** Osobní číslo: **466330**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Systém pro správu 3D obsahu

Název bakalářské práce anglicky:

3D Content Management System

Pokyny pro vypracování:

Navrhnite a implementujte systém uľahčujúci správu obsahu pre aplikácie využívajúce 3D dáta. Analyzujte a porovnajte aktuálne využívané formáty v aplikáciách na tvorbu a prezentáciu 3D modelov(.fbx, .gltf, .usdz, .sfb). Vyberte 2 z nich na základe kritérií: podpora aplikáciami, funkcionalita, možnosti kompresie.
Pomocou administračnej webovej aplikácie umožnite užívateľom nahrávať 3D obsah a spravovať ho. Umožnite aplikáciám tretích strán konzumovať obsah systému pomocou REST API.
Zabezpečte prístup k dátam a spravujte práva pomocou skupín. Prístup k úprave jednotlivých skupín modelov bude možný v administračnej aplikácii z poverených užívateľských účtov. Prístup k čítaniu dát cez API na základe generovaných API kľúčov.
Implementujte konvertory medzi zvolenými formátmi a vstupným formátom do systému: ".obj". Analyzujte a implementujte kompresiu aspoň 1 zvoleného formátu. Otestujte efektivitu kompresie a funkčnosť konvertoru na sade testovacích dát.

Seznam doporučené literatury:

[1] HUGHES, John F., Andries VAN DAM a Morgan MCGUIRE. Computer Graphics: Principles and Practice. 3 edition. Addison-Wesley Professional, 2013. ISBN 0321399528.
[2] MA, Yi, Stefano SOATTO a Jana KOSECKÁ. An Invitation to 3-D Vision: From Images to Geometric Models. Springer, 2010. ISBN 9781441918468.
[3] MCHENRY, Kenton a Peter BAJCSY. An Overview of 3D Data Content, File Formats and Viewers. An Overview of 3D Data Content, File Formats and Viewers. 2008, 2008, 1-3.
[4] RULE, Keith. 3D Graphics File Formats: A Programmer's Reference. ISBN 0201488353.
[5] SCHROEDER, Will, Ken MARTIN a Bill LORENSEN, 2003.The Visualization Toolkit An Object-Oriented Approach To 3D Graphics.3rd edition. Kitware. ISBN 1930934076.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jiří Doležal, storyous.com s.r.o.

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

Datum zadání bakalářské práce: **13.02.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

Ing. Jiří Doležal podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. _____
Datum převzetí zadání

_____
Podpis studenta

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky použité informačné zdroje v súlade s Metodickým pokynom o dodržovaní etických princípov pri príprave vysokoškolských záveverečných prác.

V Prahe dňa ....

Alexander Mensák

Poďakovanie

Ďakujem za pomoc pri vypracovaní tejto práce svojmu vedúcemu Ing. Jiřímu Doležalovi a oponentovi Ing. Davidovi Sedláčkovi Ph.D.

Alexander Mensák

Název práce:
Systém pro správu 3D obsahu

Autor: Alexander Mensák

Štúdijný program: Softwarové inženýrství a technologie

Druh práce: Bakalářská práce

Vedoucí práce: Ing. Jiří Doležal
–

Konzultant: Ing. David Sedláček, Ph.D
Katedra počítačů, Fakulta elektrotechnická, České vysoké učení technické v Praze

Abstrakt: Cieľom tejto práce je vyvinúť systém uľahčujúci správu obsahu pre aplikácie pracujúce s 3D dátami. Súčasťou je analýza aktuálne používaných technológii 3D obsahu a ich využitia. Výsledky analýzy sú aplikované v návrhu systému, tak aby systém podporoval perspektívne technológie, napríklad z oblasti v rozšírenej alebo virtuálnej realite. Práca popisuje návrh a implementáciu funkcii systému: užívateľské rozhranie pre správu dát, API pre aplikácie tretích strán, zabezpečenie zdrojov, konvertovanie a kompresiu dát. Hlavnými výstupmi sú: implementácia systému v jazyku TypeScript, konfigurácia nasadenia systému v cloudovom prostredí využitím serverless architektúry, dokumentácia verejného API, výsledky testovania nasadeného systému.

Kľúčové slova: Systém správy obsahu, 3D dáta, serverless, konvertovanie formátov, kompresia

Title:
3D Content Management System

Author: Alexander Mensák

Abstract: The goal of this bachelor thesis is to develop a system simplifying content management for applications working with 3D data. Part of work is an analysis of current 3D technologies and their usage. Results of the analysis are applied in system design to support perspective technologies from fields like augmented or virtual reality. The thesis describes the design and implementation of following system functions: the user interface for data management, API for third-party applications, resources security, conversion and compression of data. The main outputs are: system implementation using TypeScript language, the configuration of system deployment in cloud environment using serverless architecture, public API documentation, test results of the deployed system.

Key words: Content management system, 3D data, serverless, format conversion, compression

Obsah

Zoznam obrázkov

Úvod

Zvyšovanie výkonu mobilných zariadení a osobných počítačov prinieslo mnoho nových možností využitia trojrozmerného obsahu. Môžeme očakávať, že tento trend bude pokračovať a dopyt po aplikáciách využívajúcim 3D obsah bude rásť. Najväčšie využitie nájdeme v zábavnom priemysle - hry a v technickom priemysle - modely a plány. Ďalšou zaujímavou rastúcou oblasťou je prezentácia produktov v internetových obchodoch. Vyšší výkon mobilných zariadení umožnil rozvoj aplikácii ponúkajúcich rozšírenú či virtuálnu realitu čo môžeme označiť ako jeden z dôležitých míľnikov využitia 3D obsahu.

S novými možnosťami využitia sa zvyšujú nároky na správu dát. Každá aplikácia musí spravovať svoje dáta, prípadne ich konzumovať z inej služby. Požiadavky na funkcionalitu a integráciu sú často špecifické, čo zvyšuje náročnosť implementácie. Typ dát požadovaný aplikáciami nemusí byť zhodný s výstupom z nástrojov na tvorbu 3D obsahu a je potrebné prispôsobenie formátu či veľkosti.

Existuje priestor na automatizáciu veľkej časti úkonov s 3D dátami. Aplikácie je možné kategorizovať podľa funkcionalít. Často nájdeme podobné požiadavky: ukladanie, zabezpečenie, formátovanie alebo kompresia dát. Tieto požiadavky sa dajú adresovať obecným systémom ktorý funkcie poskytuje ako službu. Ak je systém dostatočne otvorený a prepoužiteľný tak sa jeho využitím môže výrazne znížiť náročnosť implementácie daných aplikácii.

Cieľom tejto práce je daný systém vyvinúť. Úspech systému stojí na predpokladoch: Cena prevádzky a vývoja musí byť menšia než ušetrené náklady na vývoj jednotlivých aplikácii, ktoré služby systému využívajú. Cena prevádzky je závislá najmä od ceny výpočetného výkonu. Keďže systém vyžaduje veľký výkon len pre určité operácie s dátami ktoré nie sú časté, mal by dynamicky alokovať zdroje. Vhodným riešením sa zdá byť architektúra bez serveru u poskytovateľov cloudových služieb. Ide o službu ktorá predstavuje abstrakciu infraštruktúry a spúšťa kód v reakciách na požiadavky. Systém by mal poskytovať stabilitu a bezpečnosť. Tiež by mal byť dostatočne otvorený, aby ho bolo možné využiť čo najväčším počtom aplikácii.

Konkrétne použitie môže systém nájsť napríklad u aplikácií pracujúcich s rozšírenou realitou ktoré zobrazujú určitú sadu 3D modelov. Ďalším príkladom je prezentácia produktov v internetových obchodoch využívajúca interaktívny 3D zobrazovač alebo konfigurátor v ktorom má zákazník možnosť produkt prehliadať otáčaním a približovaním. Tieto modely produktov môžu produkovať rôzny tvorcovia pre rôznych klientov. Je to skupina s rovnakými požiadavkami a využitie takej služby by zamedzilo duplicitnej implementácii rovnakej funkcionality v každej z aplikácii.

Kapitola 1

Analýza

Cieľom tejto kapitoly je uviesť technológie, s ktorými systém pracuje, analyzovať ich využitie a porovnať podobné služby. Pre motiváciu čitateľa, najskôr analyzujeme aktuálne spôsoby použitia 3D obsahu. V ďalšom kroku prejdeme technické detaily 3D dát. Porovnáme podobné existujúce služby na trhu.

1.1 Využitie 3D obsahu

Uplatnenie trojrozmernej počítačovej grafiky dnes nájdeme takmer v každom odvetví. Každé použitie závisí na špecifickej tvorbe a vizualizácii daného obsahu. Navrhovaný systém asistuje pri tvorbe dát a zjednodušuje proces vizualizácie. Popíšeme aktuálne a perspektívne využitia 3D obsahu pri ktorých môže nájsť systém uplatnenie.

1.1.1 Tvorba dát

Trojrozmerná počítačová grafika je reprezentácia geometrických dát v 3 dimenziách. Tieto dáta môže vytvoriť umelo človek alebo môžu vytvorené digitalizáciou reálneho sveta. Umelá tvorba dát prebieha v špecializovaných softwarových nástrojoch pre modelovanie grafiky. Rôzne typy grafiky ako filmové scény, herný obsah alebo technické modely vyžadujú rôzne nástroje a postupy tvorby. Dáta môžu byť tiež algoritmicky generované. Druhá kategória, digitalizácia reality, je proces pri ktorom najčastejšie pomocou optických alebo ultrazvukových zariadení, vzniká presná digitálna reprezentácia priestorových objektov v rôznej úrovni detailu. Proces sa často označuje ako 3D skenovanie. Existujú rôzne druhy postupov a typov digitalizácie ako napríklad fotogrametria, laserové skenovanie, tomografia.

Výstupy jednotlivých procesov sú rozličné. Existujú špecifické dátové formáty a tiež typy dát. Je obvyklé že aplikácie potrebujú zobraziť dáta ktoré neboli vytvorené za účelom zobrazovania na mobilných telefónoch alebo osobných počítačoch. Tieto dáta môžu byť príliš komplexné pre vizualizáciu alebo sú nekompatibilných formátov.

Úprava vyžaduje dodatočné softwarové nástroje s rôznym stupňom automatizácie. Je tu priestor pre zefektívnenie procesov.

Zábavný priemysel

Najväčšou oblasťou je herný priemysel. Pre veľkú časť tvorby obsahu sa obvykle využívajú nástroje Unity, Unreal engine, Blender, 3DsMax, Maya. Táto oblasť je rozsiahla, herné štúdiá často využívajú vlastné formáty. Medzi obvyklé patria formáty jednotlivých nástrojov: .ma, .max alebo všeobecné .fbx, .obj.

Ďalšou kategóriou sú filmy a umelecká tvorba. Existuje mnoho špecializovaných nástrojov na tvorbu daného obsahu. Používajú sa nástroje na sledovanie pohybu v priestore a čase na zachytávanie reálnych pohybov tela. Tie dáta sú využívané v animovanom obsahu. Obvyklé formáty sú: .fbx, .3ds, .dae.

Odborný priemysel

Jednou z najperspektívnejších oblastí je produktový design. Modely reálnych produktov sú tvorené často manuálne alebo použitím fotogrametrie. Používané modelovacie nástroje sú Blender, Maya. Využitie nájdu pri prezentácii produktov v internetových obchodoch.

Veľké využitie vidíme aj v oblasti strojovej techniky. Komplexné modely návrhov sú modelované manuálne využitím nástrojov CAD(Computer-Aided Design) ako napríklad AutoCAD od spoločnosti Autodesk. Vytvorené modely môžu byť použité v aplikáciách pre rozšírenú realitu na prototypovanie, testovanie, školenie zamestnancov alebo prezentáciu produktov.

Ďalšími významnými oblasťami sú architektúra a medicína. Pre lekárske vyšetrenia sú dáta tvorené tomografickými metódami. V architektúre nachádzajú uplatnenie CAD nástroje. Tieto dáta sú objemné a komplexné. Využitie nachádzajú len zriedka pri prezentácii výsledkov pacientom alebo klientom, no s zväčšujúcim sa výkonom osobných počítačov a mobilných zariadení to bude v budúcnosti bežnejšie. Vidíme priestor na automatizáciu optimalizácie dát pre následnú prezentáciu.

1.1.2 Vizualizácia dát

Vizualizácia je aktivita ktorá poskytuje užívateľom možnosť preskúmať a pochopiť dáta. Dáta musia byť získané z ich zdroju, ďalej transformované rôznymi metódami a následne mapované na vhodné formy pre prezentáciu užívateľovi[8]. V tomto dokumente budeme pod pojmom vizualizácia myslieť zobrazenie 3D modelov pomocou realistických renderovacích techník počítačovej grafiky v zariadení užívateľa.

Využitie nachádza v celej rade odvetví. Oblasti v ktorých môže navrhovaný systém pomôcť: Prezentácia produktov napríklad v internetových obchodoch v interaktívnom zobrazovači s možnosťou otáčania a približovania. Použitie je rozšíriteľné o

funkciu konfigurátoru vlastností a súčastí ako farba a komponenty. Hry s dynamickým obsahom vytvoreným užívateľmi alebo pochádzajúcim z aplikácii tretích strán. Vzdelávacie platformy s interaktívnym obsahom nahradzujúcim fyzické pomôcky alebo prostredie. Uplatnenie nájdu v školstve alebo v priemysle pri školení nových zamestnancov.

Web

Zobrazovanie 3D obsahu vo webových prehliadačoch je štandardizované softwarovou knižnicou WebGL ktorá rozširuje možnosti jazyka JavaScript. Umožňuje mu vytvárať interaktívnu 3D grafiku pre kompatibilné webové prehliadače[12]. Od roku 2013 podporujú WebGL všetky najpoužívanejšie webové prehliadače ako: Chrome, Safary, Firefox, IE. V čase písania textu(7.1.2019) má približne 92% používateľov kompatibilný prehliadač[13]. Pre zobrazenie uložených dát je potrebný software ktorý dokáže načítať použitý formát do pamäte a vykresliť ho pomocou funkcii WebGL. Najpoužívanejšou knižnicou s danou funkcionalitou je Three.js.

Rozšírená realita

Týmto pojmom označujeme prostredie kde fyzické a digitálne objekty spolu existujú a interagujú v reálnom čase. Najčastejším a najdostupnejším riešením je upraviť výstup kamery mobilného zariadenia tak, aby v reálnom čase zobrazovala pridaný obsah. Pokiaľ zariadenie detekuje okolité predmety a svoju polohu v priestore, môže obraz doplniť o digitálne objekty relevantné k obrazu[4].

Implementácia je veľmi komplexná úloha a vyžaduje si riešenie mnohých problémov. Sú však dostupné softwarové knižnice ktoré poskytujú potrebné funkcie k vytvoreniu aplikácii využívajúcich rozšírenú realitu. Najschopnejšie riešenia sú viazané ku konkrétnym operačným systémom.

Spoločnosť Google ponúka platformu ArCore ktorá má podporu mobilných systémov Android a iOS. Podporované sú len najnovšie modely vybraných výrobcov zariadení, čo pokrýva veľmi malú časť trhu. Platforma zabezpečuje hlavné funkcie ako: sledovanie polohy zariadenia, detekciu objektov, odhadovanie osvetlenia, spracovanie interakcie užívateľa a vykresľovanie 3D scény prekrývajúcej výstup kamery. Zobrazované môžu byť modely importované vo formáte .sfb.

Ďalšou populárnou platformou je ArKit od spoločnosti Apple. Podporovaná je len na zariadeniach s operačným systémom iOS. Ponúka podobné funkcie ako ArCore no má menšiu základňu podporovaných zariadení. Zobrazované môžu byť modely importované vo formáte .usdz.

Virtuálna realita

Na rozdiel od rozšírenej reality, pod týmto pojmom označujeme prostredie tvorené kompletne digitálnym obsahom na ktoré sa používateľ pozerá prostredníctvom obrazovky. Pre hlbšie ponorenie sa do digitálneho sveta existujú náhlavné sústavy s

optikou a displejmi, ktoré vďaka sledovania pohybu premietajú do očí používateľa virtuálny svet v ktorom sa môže pohybovať.

Vzhľadom na potrebný hardware rozlišujeme samostatné sústavy so špecializovaným hardwarom a okuliare do ktorých sa vkladá mobilné zariadenie. Riešenia sa líšia kvalitou displeja a optikou premietajúcej obraz do očí užívateľov.

Na tvorbu aplikácii sa zameriavajú nástroje najmä z herného priemyslu ako Unity a Unreal Engine. Riešení je mnoho a každé z nich používa vlastné formáty dát.

1.2 3D dáta

Trojrozmerné dáta môžu byť klasifikované do troch kategórii ako informácie o geometrii, výzore a scéne[6]. Každú z týchto kategórii ďalej popíšeme.

Geometria alebo tvar modelu je uložená ako sada 3D bodov (vrcholov). Povrch modelu tvorí séria polygónov, ktoré sú konštruované indexovaním vrcholov. Pre zaoblenie povrchu je s každým vrcholom uložená hodnota normály ktorá určuje ohyb okolitých polygónov[5].

Výzor je zaistený aplikovaním obrázku alebo textúry na povrch modelu. To je tvorené mapovaním trojrozmerných súradníc polygónov na dvojrozmerný obrázok. Ďalším spôsobom je fyzikálne modelovanie vlastností svetla, povrchu a okolia. To je realizované priradením sady vlastností každému polygónu.

Pod pojmom scéna alebo rozloženie rozumieme vzájomnú polohu modelov, kamery, osvetlenia a ďalších objektov. Každý objekt nesie ďalšie informácie o jeho vlastnostiach ako napríklad veľkosť a orientácia.

Spomenuli sme základné stavebné bloky 3D obsahu no rozličné formáty dát pracujú s rôznymi informáciami. Niektoré formáty neobsahujú všetky informácie a niektoré formáty vnášajú vlastné typy informácii. Je to dané účelom použitia.

1.2.1 Formáty

Táto časť analyzuje populárne formáty trojrozmerných dát s využitím ktoré sme diskutovali v predchádzajúcej podkapitole. Formáty sú často spojené s konkrétnymi nástrojmi a využitiami. Existujú tiež otvorené obecné formáty ktoré sa snažia byť štandardom pre určité oblasti. Každé z týchto formátov majú inú štruktúru. Niektoré využívajú textovú štruktúru a odkazujú na doplňujúce súbory obsahujúce potrebné dáta ako textúry. Ďalšie formáty zabaľujú informácie do binárnej podoby, ktorú sú schopné efektívne komprimovať.

.obj

Definuje len geometriu modelov. Menovite to sú pozície vrcholov, pozície textúry, normály, polygóny. Súradnice nemajú určené jednotky, ale súbor môže obsahovať

informácie o rozmere v čitateľnom texte poznámky. Formát je otvorený a verejne dostupný čím si našiel využitie v mnoho 3D grafických aplikáciách. Materiály ktoré popisujú vizuálne aspekty polygónov sú uložené v externých .mtl súboroch, na ktoré formát odkazuje. Súbory s materiálmi môžu ďalej odkazovať na súbory s obrázkami a textúrami.

.gltf

Otvorená a voľne dostupná špecifikácia pre efektívny prenos a načítanie 3D scén a modelov v aplikáciách. Formát minimalizuje veľkosť 3D obsahu a množstvo výkonu potrebného k rozbaleniu a použitiu. Definuje rozšíriteľný, spoločný, publikovateľný formát pre nástroje pre prácu s 3D obsahom a podporuje prenos obsahu medzi rôznymi oblasťami. Dáta popisuje v textovej štruktúre JSON a referencuje potrebné súbory. Existuje alternatívny binárny formát .glb ktorý obsahuje externé súbory.

.fbx

Vyvíjaný a vlastnený spoločnosťou Autodesk od roku 2006. Je často používaný k zaisteniu výmeny dát medzi aplikáciami na tvorbu digitálneho obsahu. Výhodou je podpora rôznych typov dát ako napríklad: 2D a 3D grafika, animácie, video, zvuk. Formát je reprezentovaný na disku ako binárne alebo ASCII dáta. Podporuje rozšírenia prostredníctvom hlavičkových súborov.

.usd

Universal Scene Description (USD) je prvý verejné dostupný software, ktorý adresuje dopyt robustne a škálovateľne vymieňať a voľne upravovať 3D scény ktoré môžu byť tvorené z mnohých zložiek. Vyvinutý spoločnosťou Pixar pre interné použitie. Spoločnosť Apple využíva upravenú verziu tohto formátu .usdz v ich platforme pre rozšírenú realitu ArKit. Ide o ZIP archív pôvodných súborov .usd formátu. Nepoužíva žiadnu kompresiu ani šifrovanie.

.sfb

Sceneform binary asset - je binárny súbor obsahujúci dáta geometrie, vizualizácie aj scény. Je generovaný pomocou textového súboru .sfa ktorý definuje jeho súčasti. Tento formát je používaný frameworkom Sceneform od spoločnosti Google. Obsahuje fyzikálne založený zobrazovač, ktorý je optimalizovaný pre mobilné zariadenia. Využíva ho platforma rozšírenej reality ArCore.

1.2.2 Konvertovanie formátov

Každá informácia, pokiaľ má byť spracovaná, musí dodržovať určitý formát. Keďže sa používa veľké množstvo formátov na rôzne účely, je časté dáta konvertovať do

formátu, ktorý vyžaduje cieľový systém. Táto operácia je síce algoritmicky jednoduchá, no vzhľadom k objemu dát môže byť časovo náročná. Jednoduchý príklad nutnosti konverzie je prípad kedy tvorca 3D modelu vytvorí súbor formátu, ktorý podporuje jeho nástroj, no model je využitý v aplikácii ktorá podporuje svoj špecifický formát pre zobrazenie. Navrhovaný systém môže pomôcť riešiť daný problém konverziou formátu potom, ako je súbor zdieľaný do systému.

Hlavnou podmienkou možnosti konvertovať formáty medzi sebou je podpora rovnakých funkcii a typov informácii. Pri konvertovaní do formátu s menej funkciami môže nastať situácia kedy budú nepodporované dáta stratené. Napríklad ak výsledný formát nepodporuje animácie, konverziou prídeme o tieto dáta. Ďalším rizikom je strata presnosti či chyby pri konverzii dát. Je potrebné byť opatrný a mať hlbšie znalosti o použitých formátoch.

Software na konverziu väčšinou dodáva tvorca formátu a to hlavne s podporou populárnych a otvorených formátov. V prípade voľne dostupných formátov nájdeme veľké množstvo verejných a komunitou udržovaných riešení. Vo väčšine prípadov nie je nutné vyvíjať vlastné riešenie.

1.3 Porovnanie existujúcich systémov

V tejto časti preskúmame existujúce systémy a služby z oblasti správy 3D obsahu s podobnými funkciami ako navrhovaný systém. Zameriame sa na ich spoločné charakteristiky ale aj rozdielne a jedinečné funkcie. Analyzujeme ich použitie a pokúsime sa identifikovať oblasť trhu kde by mal navrhovaný systém uplatnenie.

Do analýzy sme vybrali 3 najpopulárnejšie služby na správu 3D obsahu ktoré majú najväčší objem užívateľov. Ich systémy sú využívané tvorcami obsahu pre jeho zdieľanie a prípadné upravovanie. Systémy zároveň ponúkajú spôsoby aplikáciám tretích strán ako ich obsah konzumovať.

1.3.1 Sketchfab

Najväčšia platforma z vybraných troch vzhľadom na počet užívateľov a funkcii. Pôvodné zameranie bolo na verejné zdieľanie 3D modelov. Súčasťou je trhovisko na ktorom môžu užívatelia predávať svoje modely a nakupovať modely ostatných. Poskytuje interaktívny zobrazovač modelov, ktorý môžu užívatelia pridať na svoj web a umožniť ich zákazníkom prehliadať modely. Podporuje desiatky formátov pre nahranie do systému no výstup umožňuje len vo formáte .gltf. Poskytované API umožňuje správu informácii o modelov, používateľských účtov, transakcii a je k dispozícii API k sťahovaniu dát modelov.

1.3.2 Vectary

Služba primárne poskytuje nástroj na modelovanie 3D objektov. Súčasťou je privátne úložisko, do ktorého môžu užívatelia importovať súbory v rôznych formátoch a následne ich konvertovať pri exportovaní. Tieto modely môžu ďalej kombinovať a upravovať. Zaujímavou funkciou je možnosť fotorealistického redneringu uložených modelov. Služba neposkytuje API pre aplikácie tretích strán.

1.3.3 Google Poly

Poly od spoločnosti Google je služba ktorá slúži ako verejné úložisko 3D modelov. Využitie nie je spoplatnené a užívatelia môžu voľne nahrávať a sťahovať obsah. Nahrané modely sú vždy dostupné verejne bez možnosti privátneho obsahu. Služba poskytuje API prostredníctvom ktorého je možné modely nahrávať a sťahovať.

1.3.4 Vyhodnotenie

Sketchfab sa zdá ako ideálny kandidát na podporu procesu od tvorby obsahu až po zobrazenie ale je stavaný ako verejný zoznam a jeho API je obmedzené. Vectrary sa sústredí na tvorbu dát a možnosti na podporu aplikácii zobrazujúcich obsah ostali pri možnosti manuálneho exportu dát. Poly je zas banka zameraná výhradne na konzumáciu existujúceho obsahu pre jednoduché implementovanie do aplikácii tretích strán. Ak chceme podporiť aplikácie využívajúce 3D dáta, musíme sa sústrediť na celý životný cyklus dát od vzniku až po distribúciu do zariadení klientov. Mali by sme sa zamerať na 3 základné časti ktoré vidíme pri analyzovaných službách osobitne: zabezpečenie privátnych dát, robustné API, podpora mnohých formátov a konverzii.

Kapitola 2

Návrh systému

Na základe analýzy z prvej kapitoly predstavíme návrh systému využívajúci popisované technológie a adresujúci nedostatky existujúcich riešení.

2.1 Cieľová skupina

Účel systému je poskytovať služby správy 3D obsahu. Tieto služby môžu využiť tvorcovia obsahu alebo spoločnosti vyvíjajúce aplikácie závislé na 3D obsahu. Využitie vidíme už dnes v existujúcich aplikáciách v oblastiach ako sú napríklad: internetové obchody, produktový design, vzdelávanie alebo herný priemysel. Konkrétne použitie obsahu môže byť interaktívne zobrazovače produktov, konfigurátory produktov, interaktívne návody alebo hry využívajúce rozšírenú realitu a mnohé ďalšie.

Vzoroví príklad pre ilustráciu a lepšie pochopenie čitateľa je internetový obchod ktorý potrebuje zobrazovať svoje produkty v interaktívnom 3D zobrazovači. Tento zobrazovač si nechal implementovať do svojich webových stránok. Produktové modely si nechá na zákazku vytvoriť externou firmou. Keďže môžu pribúdať nové produkty je vhodné proces pridania a upravenia ponuky čo v najväčšej miere automatizovať. Použitím ideálneho systému by mohol tvorca modely produktov nahrať na server vo formáte ktorý exportuje jeho nástroj. Dáta by boli automaticky konvertované do formátu akceptovaného interaktívnym zobrazovačom na webe. Správca internetového obchodu by nový model priradil k dátam o produkte a systém by model začal zobrazovať.

2.2 Prípady použitia

Pred návrhom jednotlivých funkcii musíme pochopiť ako a kto bude systém využívať. Popíšeme podrobne postup pri použití. Predtým definujeme troch aktérov:

Klient - správca aplikácie využívajúcej službu

Tvorca obsahu - osoba vytvárajúca obsah pre klienta

Admin systému - správca celej služby

2.2.1 Používateľské účty

Vytvorenie používateľa / skupiny

Aktér: *Admin systému*

1. Admin sa prihlási do aplikácie.
2. Systém autentizuje admina.
3. Admin vyplní informácie o novom užívateľovi / skupine.
4. Systém uloží informácie.

Úprava informácii o užívateľovi / skupine

Aktér: *Admin systému*

1. Admin sa prihlási do aplikácie.
2. Systém autentizuje admina.
3. Admin upraví informácie.
4. Systém uloží informácie.

Odstránenie používateľa / skupiny

Aktér: *Admin systému*

1. Admin sa prihlási do aplikácie.
2. Systém autentizuje admina.
3. Admin zvolí užívateľa / skupinu k odstráneniu.
4. Systém odstráni dáta.

2.2.2 Dáta modelov

Vytvorenie / úprava / odstránenie modelu

Aktér: *Klient, Tvorca obsahu, Admin systému*

1. Užívateľ sa prihlási do aplikácie.
2. Systém autentizuje užívateľa.
3. Užívateľ vyplní / upraví / odstráni informácie o modely.
4. Systém uloží / odstráni informácie.

Prehliadanie modelov

Aktér: *Klient, Tvorca obsahu, Admin systému*

1. Užívateľ sa prihlási do aplikácie.
2. Systém autentizuje užívateľa.
3. Systém zobrazí modely patriace skupine užívateľa.
4. Užívateľ zvolí konkrétny model.
5. Systém zobrazí informácie o zvolenom modely.

Konvertovanie / komprimovanie formátu

Aktér: *Klient, Tvorca obsahu, Admin systému*

1. Užívateľ sa prihlási do aplikácie.
2. Systém autentizuje užívateľa.
3. Užívateľ zvolí model a nový formát.
4. Systém konvertuje / komprimuje formát modelu.

Správa API kľúčov

Aktér: *Klient, Admin systému*

1. Užívateľ sa prihlási do aplikácie.
2. Systém autentizuje užívateľa.
3. Užívateľ vyplní informácie o aplikácii požadujúcej prístup.
4. Užívateľ vyplní informácie ku ktorým modelom udeľuje prístup.
5. Systém uloží informácie.

Stiahnutie jednotlivého modelu / listu modelov

Aktér: *Klientská aplikácia*

1. Aplikácia požiada o dáta.
2. Systém overí platnosť API kľúču.
3. Systém overí povolenie k prístupu k dátam.
4. Systém vráti požadované dáta.

2.3 Funkcie a požiadavky

Z poznatkov získaných v analýze a z prípadov použitia vyvodíme konkrétne funkcie systému ktoré budú implementované.

2.3.1 Funkčné požiadavky

1. Účet má jednu rolu udeľujúcu práva z dvoch možných: užívateľ, admin.
2. Užívateľský účet patrí do jednej skupiny.
3. Užívateľ / admin sa môže prihlásiť do / ohlásiť zo systému.
4. Admin môže vytvárať / prehliadať / upravovať / odstrániť užívateľské účty a skupiny.
5. Admin môže upravovať a pristupovať k všetkým dostupným dátam v systéme.
6. Užívateľ môže vytvárať / prehliadať / upravovať / odstrániť informácie modelov patriacich jeho skupine.
7. Užívateľ môže do systému nahrať dáta 3D modelov v .obj formáte
8. Užívateľ môže vyžiadať zmenu formátu dát modelu na formát .gltf a .glb.
9. Užívateľ nemôže upravovať dáta patriace inému účtu / skupine.
10. Užívateľ môže upravovať dáta API kľúčov jeho skupiny.
11. Aplikácie tretích strán sa môžu autentizovať v doručených žiadostiach systému.
12. Aplikácie tretích strán môžu pomocou prideleného autorizačného kľúču spravovať všetky prostriedky danej skupiny.
13. Aplikácie tretích strán môžu bez autorizačného kľúču pristupovať k všetkým 3D modelom.
14. Systém používa maximálnu kompresiu pri formátoch u ktorých je možná.
15. Systém ukladá dáta modelov do cloudového úložiska.

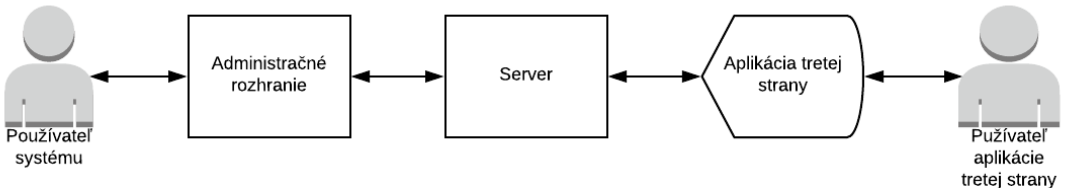
2.3.2 Nefunkčné požiadavky

1. Systém je možné integrovať do aplikácii tretích strán zobrazujúcich dáta.
2. Administračné rozhranie musí byť dostupné vo webových prehliadačoch.
3. Systém musí zabezpečiť ochranu dát pred neautorizovaným prístupom.
4. Systém neudržuje rezervované hardwarové kapacity keď nie je využívaný.

2.4 Architektúra

Systém má pomerne jednoduchú architektúru. Hlavnou časťou je server ktorý sa stará a logiku aplikácie a spravuje dáta. Užívatelia služby komunikujú so systémom

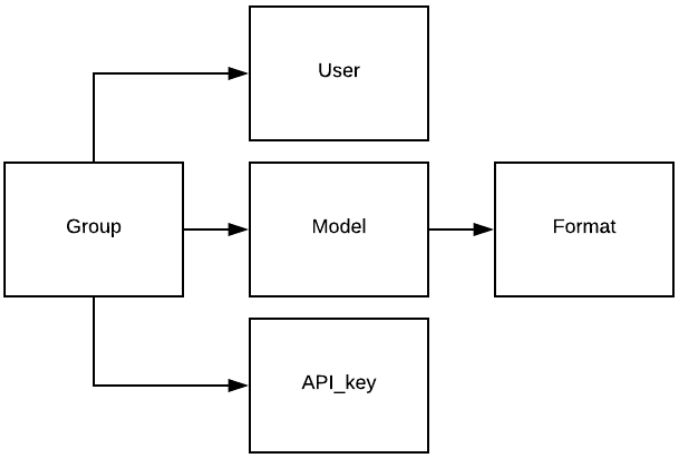
pomocou webovej aplikácie s administračným rozhraním. Táto aplikácia komunikuje so serverom pomocou interného API. Obsahuje funkcie na prehliadanie a správu dát. Integrované aplikácie komunikujú so serverom pomocou verejného API. Na prístup k dátam musí mať integrovaná aplikácia povolenie.



Obr. 2.1: Architektúra systému

2.5 Dátový model

Z funkčných požiadaviek môžeme navrhnúť dátový model. Všetky dáta v systéme patria pod svoju skupinu. Skupina je teda dôležitá entita, ktorá nesie všetky ostatné dáta. Každá skupina je jeden klient systému, napríklad spoločnosť vlastniaca internetový obchod. Užívateľské účty zabezpečujú možnosť riadenia prístupu k dátam systému. Existujú dva typy účtov, administrátor systému, ktorý má práva zasahovať do celého systému a užívateľ, ktorý môže spravovať len svoju skupinu. Dáta 3D modelov sú uložené v cloudovom úložisku a referencujeme ich pomocou identifikátoru. Ten je uložený spolu s informáciou o formáte dát v entite formát. List dostupných formátov modelu spolu s ďalšími informáciami patrí entite model. Zabezpečenie dát pre prístup cez verejné API je realizované pomocou API kľúčov, ktoré pre každú skupinu určujú, ktoré aplikácie majú prístup k jej dátam.



Obr. 2.2: Dátový model (Všetky väzby sú z 1 -> n v smere šípky)

2.6 API

Rozhranie pre programovanie aplikácii (API - Application Programming Interface) nášho serveru umožňuje komunikáciu s ďalšími aplikáciami. Podporuje správu a zdieľanie dát. Pre aplikácie patriace systému, ako napríklad administračné rozhranie, slúži interné API. Pre komunikáciu s aplikáciami tretích strán, ako napríklad internetový obchod klienta, slúži verejné API. Rozdelenie je dané rôznym zabezpečením. Komunikácia bude prebiehať využitím protokolu HTTP a návrhového vzoru REST. Je to dnes štandardný postup ktorý podporuje väčšina aplikácii.

Možné operácie nad dátami odpovedajú funkciám definovaním v sekcii s funkčnými požiadavkami. Obecne to sú CRUD operácie (vytvoriť / čítať / upraviť / zmazať) využívajúce HTTP správy (POST / GET / PUT / DELETE) nad entitami dátového modelu. Ďalej obsahuje operácie pre registráciu, prihlásenie, odhlásenie užívateľa, nahrávanie dát modelov alebo vyžiadanie akcie ako napríklad konvertovanie alebo kompresia modelu.

2.6.1 Interné

Hlavnou úlohou interného API je umožnenie správy dát aplikácie pomocou administračnej aplikácie. Je zabezpečené autorizáciou prístupu k dátam a teda dostupné až po prihlásení užívateľa(kapitola 2.7).

2.6.2 Verejné

Verejné API slúži k umožneniu komunikácie s aplikáciami tretích strán. Aplikácie ktoré zobrazujú 3D modely uložené v systéme môžu systém žiadať o dané dáta. Systém ich poskytuje prostredníctvom HTTP požiadaviek ako pri internom API. Prístup k dátam je povolený len predom definovaným aplikáciám(kapitola 2.7)na základe prístupového kľúču.

2.7 Zabezpečenie

Autentizácia užívateľa prebieha prostredníctvom informácií jeho profilu uloženého v databáze systému. Profil musí vytvoriť administrátor a užívateľovi poslať prihlasovacie údaje. Užívateľ pri prihlásení zadá tieto údaje čím sa identifikuje a systém ho autentizuje. Po úspešnej autentizácii je užívateľovi vystavený JWT token. Token musí byť priložený ku každej požiadavke na server aby mohol systém overiť identitu žiadateľa. Token je posielaný v hlavičke HTTP požiadavku v parametri 'Authorization'.

Autorizácia užívateľa k potvrdeniu práv vykonať požadovanú akciu je prevedená pomocou užívateľských rolí a zaradeniu do skupín. Systém pri požiadavku od užívateľa načíta jeho rolu a identifikátor skupiny do ktorej patrí z JWT tokenu v hlavičke

HTTP požiadavku. Rola administrátor dáva držiteľovi plné práva na všetky operácie v systéme. Rola používateľ dáva držiteľovi právo len na operácie s modelmi a API kľúčmi ktoré patria jeho skupine. Pri autorizovaní žiadosti systém overí túto podmienku pomocou identifikátoru skupiny z tokenu.

Aplikácie tretích strán využívajú takmer rovnaký postup, no ich JWT token vygeneruje systém na základe požiadavku. Prístup je kontrolovaný vyhľadaním kľúču v databáze systému.

2.8 Administrácia

Správa dát systému je možná cez administračné rozhranie. Pre zabezpečenie dostupnosti a kompatibility s bežnými zariadeniami a operačnými systémami je implementovaná ako webová aplikácia. Aplikácia komunikuje so serverom cez REST API. Pre zjednodušenie implementácie sa javí ako vhodné využiť softwarovú knižnicu obsahujúcu predefinované grafické prvky užívateľského rozhrania. Pri výbere technológie je rozumné zohľadniť aktuálnu popularitu daného riešenia čo zabezpečí existenciu aktívnej komunity vývojárov. Vďaka tomu má vývojár možnosť poradiť sa ohľadne riešení jeho problémov a tiež to zaručuje dostupnosť rozšírení na ďalšie novo vzniknuté technológie. Aktuálny trend je implementovať webové stránky pomocou jazyku JavaScript a dostupných frameworkov. Zvolili sme preto knižnicu React vyvíjanú spoločnosťou Facebook. Ako nadstavbu využijeme sadu hotových grafických a funkčných komponent pre administračné rozhrania React-admin využívajúcich REST API.

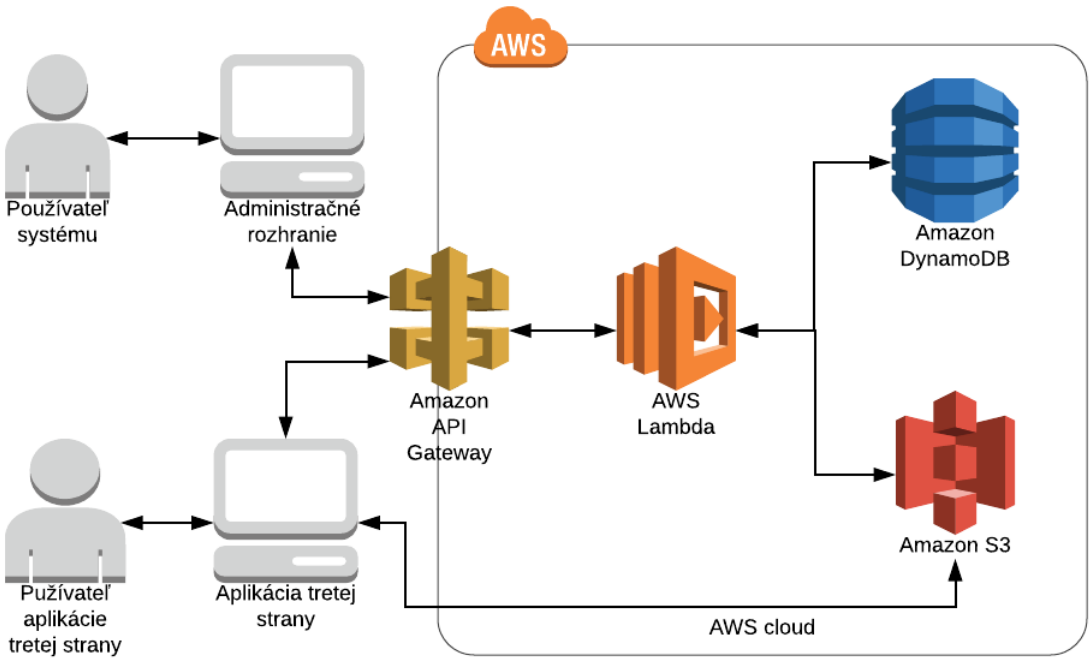
Kapitola 3

Implementácia

Systém bol implementovaný tak aby odpovedal na požiadavky z návrhu systému. V tejto kapitole podrobne popíšeme spôsob implementácie jednotlivých častí systému a ich funkcií.

3.1 Architektúra serveru

Architektúra odpovedá na návrh systému a prestavuje 3 hlavné komponenty: Aplikačný server, databáza a úložisko súborov. Výber dodávateľa cloudových služieb a prostredia aplikačného serveru podmienil výber ostatných častí.



Obr. 3.1: Nasadenie systému

3.1.1 Serverless Architektúra

Architektúra systému je z veľkej časti ovplyvnená výberom "serverless"návrhu. Pojem serverless označuje cloudovú službu ktorá zabezpečuje spúšťanie programu len pri prichádzajúcich požiadavkách. Hlavnou výhodou je odpadnutie nutnosti udržovania vlastnej či už hardwarovej alebo virtuálnej infraštruktúry. Pri implementácii aplikácie pracujúcej v tomto prostredí je nutné dbať na: rýchlosť studeného štartu, minimalizáciu závislostí k iným systémom a hlavne bezstavovosť programu[10].

Rýchlosť štartu aplikačného serveru zabezpečuje výber technológie NodeJs[7], ktorá je optimalizovaná pre serverless prostredia. Minimalizácia závislostí je dosiahnutá využívaním výhradne služieb v rámci rovnakého clodu vďaka čomu sú navrhnuté k jednoduchému používaniu. Bezstavovosť programu je dosiahnutá presunutím všetkých dát do databáze.

Aplikačný server

Výkonná časť systému je tvorená cloudovými funkciami v službe AWS Lambda[2]. Kód týchto funkcii je vykonaný len pri prichádzajúcom požiadavku. Pre každú požiadavku je aplikačný server NodeJs spustený znova. Po vykonaní je udržaný zapnutý niekoľko minút než sa kompletne uvoľnia zdroje ktoré si alokoval. Prvý takzvaný studený štart býva problematický pretože môže trvať dlhšie než by sa dalo čakať.

Momentálna implementácia systému je postavená len ja 1 funkcii, ktorá obsluhuje požiadavky na všetkých cestách API. Je to vhodné pri malej záťaži keďže po prvej požiadavke ostáva funkcia aktívna a všetky ďalšie sú obslúžené bez čakania na studený štart. Pri vysokej záťaži dáva zmysel časti API rozdeliť do jednotlivých funkcií aby sa na nich rovnomerne rozprestrela záťaž a nedošlo k alokovaniu príliš vysokého počtu funkcii.

Funkcie sú spúšťané len na základe HTTP požiadaviek z API. Komunikujú s ostatnými časťami systému využitím balíčkov s implementáciou klientského rozhrania pre konzumovanie daných služieb. Tieto balíčky sú optimalizované tak aby komunikácia bola rýchla a nenáročná. Od klasických riešení sa líšia ľahkým štartom nadviazania spojenia so službou.

Databáza

Pre ukladanie dát je využívaná noSQL databáza AWS DynamoDB[1]. Je poskytovaná ako plne spravovaná cloudová služba. Nie je potrebné starať sa o infraštruktúru ako pri často používaných databázových riešeniach. Poskytuje garantovanú rýchlosť operácii do 10 milisekúnd pri akejkoľvek záťaži a takmer neobmedzené možnosti okamžitého škálovania dostupnej kapacity. Pre serverless architektúru to z nej robí ideálnu voľbu.

Vzhľadom na to, že použitý dátový model má silné väzby medzi entitami a nepraje nám fakt že služba neponúka dostatočné možnosti na pracovanie so zanorenými dokumentami v stromovej štruktúre, museli sme implementovať pre každú

entitu vlastnú tabuľku. Odkazujú na seba identifikačnými kľúčmi. Výkon systému to výrazne neovplyvňuje pretože všetky operácie v API sú implementované tak aby nevracali celú stromovú štruktúru modelu.

Prístup k databáze je riadený entitami služby AWS. Pre aplikačný server je vytvorená entita ktorá má oprávnenie k operáciám vytvárať, čítať a odstraňovať dáta z databázy. Je to jediný spôsob ako sa dá k dátam pristúpiť.

Úložisko súborov

Dáta 3D modelov nie je možné pre ich veľkosť ukladať v databáze. K tomu účelu je použitá služba AWS S3 (Simple Storage Service)[3] ktorá je líder v škálovateľnosti, dostupnosti dát, bezpečnosti a výkone. Je plne spravovaná a teda ideálna k podobnému zbytku nášho systému.

Prístup je podobne ako pri databáze riadený entitami AWS s povoleným prístupom len z aplikačného serveru. Výnimkou je nahrávanie 3D modelov, ktoré je dovolené priamo od klienta prostredníctvom podpísanej adresy s dočasnou platnosťou k nahraniu súboru pod vopred určený kľuč. Toto riešenie je potrebné pre obídenie limitu na maximálnu veľkosť 10MB prílohy, ktorú môže cloudová funkcia dostať v HTTP požiadavku.

3.1.2 Programovací jazyk

Kód severu je písaný v jazyku TypeScript[11] a následne je prekladaný do jazyku JavaScript v ktorom je spúšťaný v prostredí NodeJs. Architektúru kódu diktuje knižnica Express.js ktorá je populárna pri serveroch s REST API. Z jej použitia sú dané návrhové vzory ktorými sa inšpirujú všetky časti programu.

3.1.3 Návrhové vzory

Spracovanie prichádzajúcich HTTP požiadaviek je v Express.js riešené striktne funkcionálnym programovaním. Knižnica definuje sadu pomocných funkcii ktoré pomáhajú požiadavky spracovávať. Pri vlastnom zásahu do procesu spracovania je použitý vzor Middleware - pridávanie medzivrstiev do postupností funkcii spracovávajúcich prichádzajúce požiadavky. Takto môžeme napríklad autentizovať užívateľa na základe prístupového kľúču skôr než sa k požiadavke dostanú ostatné funkcie.

Pre prístup k úložisku je použitý objektový prístup vďaka ktorému sme získali dostatočnú abstrakciu a možnosť prepoužiť kód vďaka generickým typom. Využívame tak 3 vrstvy: úložiskovú(Repository), servisnú(Service) a pomocnú HTTP vrstvu(CRUDHelper).

Úložisková vrstva pozná konkrétnu implementáciu klientského rozhrania databázového úložiska. Vykonáva tak priame zápisy a čítania dát. Servisná vrstva využíva abstrakciu úložiskovej vrstvy a zaisťuje konzistenciu dát pri operáciách: vytvoriť,

čítať, upraviť a zmazať. Pomocná HTTP vrstva prekladá požiadavky z parametrov v požadovaných adresách a posúva ich servisnej vrstve.

3.2 Architektúra administračnej aplikácie

Užívateľské rozhranie je statická jednostránková webová aplikácia využívajúca framework ReactJs pre vykresľovanie grafických prvkov jazykom JavaScript. Je navrhnutá tak aby umožňovala všetky operácie nad dátami systému prehľadne a jednoducho.

3.2.1 Programovací jazyk

Kód aplikácie je napísaný v jazyku TypeScript ktorý je, rovnako ako pri aplikačnom servery, pred spustením preložený do jazyku JavaScript v ktorom je vykonaný. Keďže sa jedná o web tak projekt využíva aj jazyky HTML a CSS na dotvorenie niektorých prvkov užívateľského rozhrania.

Použitý framework ReactJs taktiež diktuje jasnú architektúru kódu. Aplikácia je rozdelená na strom komponent ktoré sa navzájom obaľujú. Rozpadom z väčších častí na malé tak elegantne tvoríme komplexné funkcie.

Táto časť však nie je jedinečnou súčasťou systému a obsahuje funkcie pre správu dát ktoré sú v podobných aplikáciách časté. Vznikli tak knižnice poskytujúce grafické rozhranie naviazané na dnes už takmer štandardizované typy API. Konkrétne využívame knižnicu ReactAdmin ktorá poskytuje komponenty na zobrazovanie dát, formuláre na úpravu a ďalšie pomocné funkcie ktoré sa priamo napájajú na rôzne typy REST JSON API.

3.2.2 Zabezpečenie

Pre prístup do administračnej aplikácie je potrebné prihlásenie. Server vráti token ktorý musí byť použitý pre každú ďalšiu požiadavku. Token je uložený v lokálnej pamäti prehliadaču a ma obmedzenú platnosť (viac v sekcií 3.5).

Užívateľské rozhranie sa líši v závislosti na roli užívateľa. Pre admina sú naviac dostupné záložky pre správu skupín a užívateľov. Tiež má možnosť pri každej editácii dát vyberať, s ktorou skupinou pracuje.

Základný užívateľ má k dispozícii modely s ich formátmi a API kľúče. Pri editácii vždy pracuje s dátami svojej skupiny.

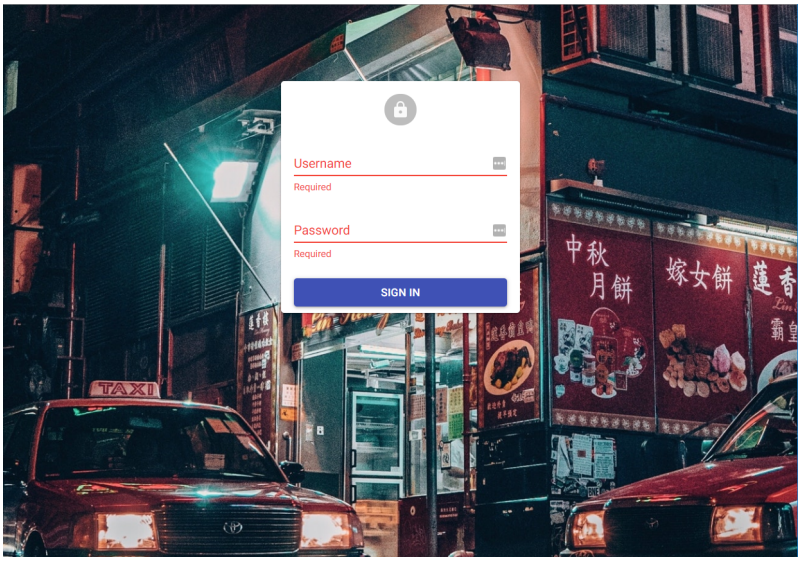
3.2.3 Obsah

Používateľské rozhranie sa skladá z 3 základných častí: Prihlásenie, tabuľky dát a formulárové okná pre vytváranie,zobrazenie, editáciu dát. Vývoj je uľahčený hotovou

grafikou a komponentami z balíčku ReactAdmin. Komponenty tabuliek dát a okien sú medzi entitami prepoužité.

Prihlásenie

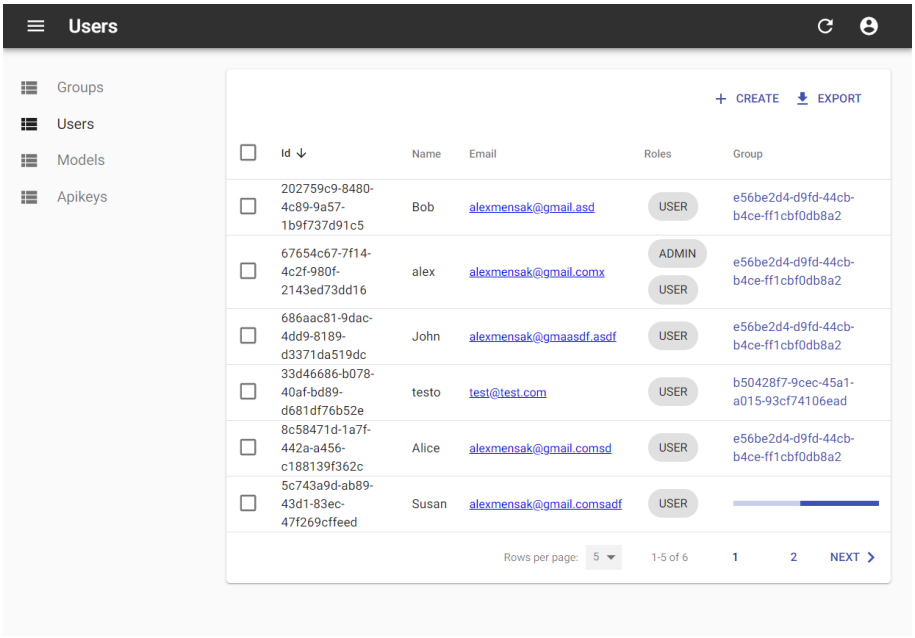
Jednoduchý formulár pre prihlásenie.



Obr. 3.2: Prihlásenie do aplikácie

Tabuľky

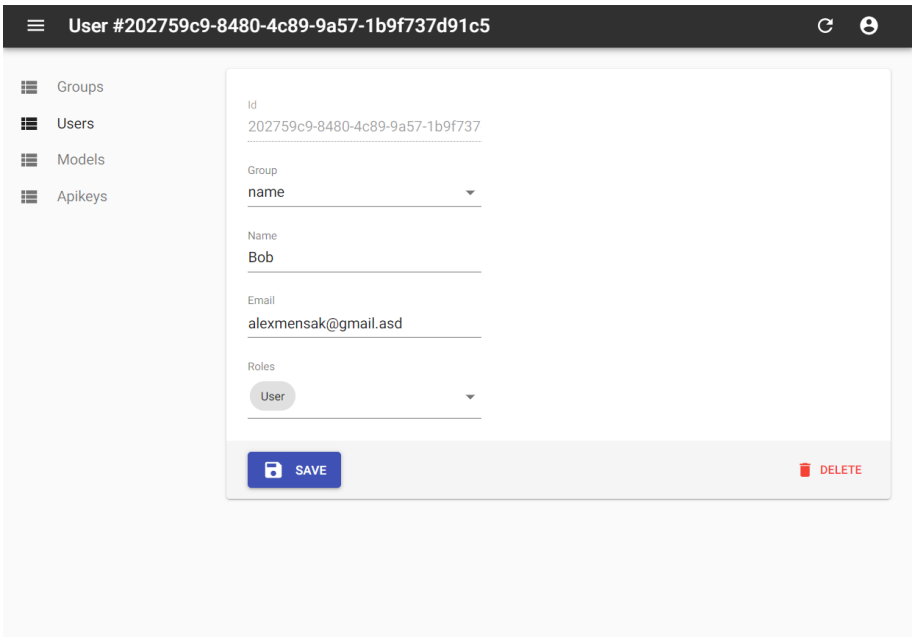
Praktická tabuľka pre zobrazovanie dát. Umožňuje export do súboru .csv, radenie podľa vybraného parametru, stránkovanie.



Obr. 3.3: Dátová tabulka

Formulárové okná

Formulárové okná pre zobrazovanie alebo úpravu dát. Okno s detailom modelu ponúka extra akcie: konvertovať formát do .gltf alebo komprimovať do .glb.



Obr. 3.4: Formulárové okno

3.3 Nasadenie systému

3.3.1 Server

Systém je nasadený v prostredí AWS cloudu od spoločnosti Amazon. Využíva API gateway pre riadenie požiadaviek a následné prideľovanie aplikáciám v službe AWS Lambda. Systém ďalej využíva úložisko Amazon S3 pre ukladanie dát 3D modelov a poskytovanie týchto dát klientskym aplikáciám. Pre uchovanie informácii je využitá NoSQL databáza DynamoDB. Viac infografika(3.1) na prvej strane kapitoly.

Výhodou používania serverless technológii je fakt že pre daný systém neexistuje a nie je využívaná žiadna hardwarová ani virtuálna infraštruktúra. Teda odpadá nutnosť udržovať súčasti aktualizované a funkčné čo je oproti starším systémom obrovské zjednodušenie. Ďalšou výhodou sú možnosti škálovania. Dostupná kapacita využívaných služieb je obmedzená no pri požiadaní sa dá veľmi jednoducho zvýšiť. Služby sú stavané tak aby postačovali aj najnáročnejším využitiam.

Konfigurácia

Cloudové funkcie v službe AWS Lambda sú konfigurované nasledovne:

1. Prostredie: NodeJs vo verzii 8.10
2. Pamäť inštancie: 1024MB
3. Časový limit pre požiadavky: 240 sekúnd

Databáza DynamoDB obsahuje 5 tabuliek entít. Pre momentálne testovacie účely je jej kapacita rezervovaná na 1 súbežný zápis a 1 čítanie. Pre ďalšie použitie je vhodné túto kapacitu zvýšiť.

Úložisko súborov AWS S3 využíva 1 sektor 'cms-model-data' ktorý má zabezpečené editujúce operácie a otvorené čítanie tak aby dáta 3D modelov boli dostupné verejne.

Cena

Služba cloudových funkcií je účtovaná za počet požiadaviek a čas aktívneho používania zaokrúhleného na 100 milisekúnd a závisí od výkonu inštancie. Cena 1 miliónu požiadaviek je 4.58 kč, cena za GB/sekundu je 0.0003823 kč. V rámci voľnej kapacity je nám poskytnutých zdarma každý mesiac 1 milión požiadaviek a 400 000 GB/sekúnd. To je ekvivalentné takmer 5 dňovej nepretržitej prevádzke. Pri malých desiatkach užívateľov to ale postačí plne pokryť našu mesačnú spotrebu.

Databáza DynamoDB je účtovaná za počet operácii, veľkosť uložených dát a objem prenesených dát. Cena 1 miliónu zápisov je 28.69 kč, 1 milión čítaní je 5.74 kč. Cena 1 GB dát uloženého mesiac je 5.74 kč. V rámci voľnej kapacity nám je poskytnutý priestor na 25 GB dát a približne 2 milióny operácii.

Súborové úložisko je účtované od objemu uložených dát mesačne. Cena 1 GB je 0.53 kč a znižuje sa s nárastom celkového využívaného objemu. V rámci voľnej kapacity je poskytnutých 5 GB mesačne zdarma.

Pri malých desiatkach mesačných užívateľoch s priemerným nahrávaním 10 modelov týždenne bude naša približná spotreba vyzerať nasledovne: 10 užívateľov * 10 nahratí modelu týždenne * 4 týždne * 20 operácii k nahratiu modelu = 4000 invokácii cloudových funkcii. Pre zápisy do databáze to je 4000 invokácii * priemerne 5 požiadaviek na databázu = 20 000 operácii. Počet modelov: 10 užívateľov * 10 nahratí modelu týždenne * 4 týždne * 10MB priemerná veľkosť modelu * 3 formáty dát pre model = 12 000 MB. Vidíme že počet požiadaviek sa neblíži voľnej kapacite, museli by sme mať malé tisícky užívateľov aby sme prekročili voľné kapacity v každej metrike. Najdrahšia zložka bude úložisko 3D modelov ktoré môžu dosahovať veľký objem. Zaujímavým riešením by bolo účtovanie ceny služby nášho systému od objemu využitých dát.

Spúšťanie

Pri využívaní platformy ako služby cloudového poskytovateľa je oproti využívaniu infraštruktúry ako služby výhoda že pre spustenie vlastnej aplikácie je potrebné dodať len konfigurácie a kód programu. Nie je potrebné starať sa o infraštruktúru a jej zdroje.

V tomto smere využívame opensource nástroj 'Serverless'[9] ktorý poskytuje jednoduchý prístup k správe aplikácie u rôznych cloudových poskytovateľov cez príkazový riadok a stručný konfiguračný súbor.

3.3.2 Administračný klient

Administračná aplikácia má podobu sady statických súborov, ktoré sú umiestnené vo webovom hostingu. Nie je potrebné špeciálne prostredie a tak môžeme využiť rôznych služieb na poskytovanie obsahu. Pre testovaciu fázu je umiestnená v službe 'Firebase' ktorá poskytuje hosting a doménu zdarma s jednoduchým prístupom cez príkazový riadok. V neskoršej fáze pri požadovanej väčšej robustnosti môže byť presunutá na platformu AWS.

3.4 REST API

Aplikačný server ponúka vonkajším aplikáciám, ako napríklad administračné prostredie alebo aplikácie tretích strán, API založené na princípoch REST. REST (Representational State Transfer) je architektúra, ktorá umožňuje CRUD operácie (vytvoriť / čítať / upraviť / odstrániť) na dátach pomocou HTTP požiadaviek.

Dáta sú prenášane v tele požiadavky vo formáte JSON (JavaScript Object Notation) ktorý popisuje jednotlivé objekty dátových entít. Ďalšie parametre sú súčasťou

adresy a cesty HTTP požiadavky. Jedná sa o špecifikujúce identifikátory alebo parametre pre filtrovanie a radenie požadovaných dát. Podrobná dokumentácia API je popísaná na konci dokumentu v dodatku A, prílohe 'Dokumentácia API'.

3.4.1 Zabezpečenie

Požiadavky sa voči aplikačnému serveru autentizujú prístupovým kľúčom v hlavičke požiadavky vo formáte: 'Authorization: Bearer <PRÍSTUPOVÝ KĽÚČ>'. Prístupový kľúč je získaný po prihlásení do systému pre daný používateľský účet a má časovo obmedzenú platnosť alebo je možné vygenerovať API kľúč pre danú skupinu ktorý má neobmedzenú platnosť a je určený pre aplikácie tretích strán.

Autorizácia prebieha na základe používateľských rolí a priradeniu do užívateľských skupín. Používateľské role sú dve: 'USER' - štandardný používateľ s prístupom len k jeho skupine a 'ADMIN' - administrátor systému s prístupom k všetkým dátam systému. Autorizácia podľa užívateľských skupín je dôležitá pre spravovanie viacerých spoločností v systéme ktoré medzi sebou nezdieľajú dáta.

Prístupový kľúč je vo formáte JWT (JSON Web Token). Obsahuje informácie o používateľovi ktorému bol vydaný alebo skupine ktorej API kľúč patrí. Kľúč slúži ako garancia úspešnej autentizácie jeho držiteľa niekedy v minulosti. Jeho nezpochybniteľnosť zabezpečuje podpis dát kľúču privátnym kľúčom serveru ktorého pravosť môžeme overiť verejným kľúčom.

3.4.2 Nahrávanie modelov

Pre nahranie dát nového 3D modelu do systému je nutné vytvoriť nový objekt modelu. K tomuto modelu vytvoriť nový objekt formátu. Po vytvorení server zašle adresu ktorú obdržal zo služby AWS S3 a na ktorú je možné dočasne a jednorazovo nahrať dáta. Je dôležité, kvôli bezpečnosti, model nahrať okamžite po získaní tejto adresy, nikde ju neukladať a nezdieľať.

3.4.3 Výhody

Poskytnutím verejného API ku ktorému je možné riadiť prístup sme získali výhodu oproti mnohým konkurenčným službám. Vďaka tomu je možné systém využiť na automatizáciu mnohých procesov pri práci s 3D dátami. Dôraz na jednoduchosť operácii tomuto faktu výrazne pomáha.

Príklad môže byť napojenie na informačný systém e-shopu, ktorý zobrazuje svoje produkty pomocou 3D modelov. Vďaka API môže poskytnúť kontrolovaný prístup k dátam svojmu dodávateľovi produktových modelov a ďalej ich napájať na záznamy produktov v svojom systéme. Bez nášho riešenia by takáto funkcionalita vyžadovala veľké množstvo manuálnej práce pri nahrávaní a spravovaní dát, prípadne vysoké náklady na implementáciu rovnakej funkcionality do existujúceho systému.

3.5 Práca s 3D modelmi

Systém umožňuje nahrať dáta 3D modelu vo formáte .obj. Tento formát je bez kompresie a obsahuje len informácie o 3D priestore. Informácie k vizuálnej podobe sú uložené v súbore .mtl ktorý definuje materiály povrchu priestorových modelov a ďalej odkazuje na súbory obrázkov, ktoré využíva ako textúry. Z dôvodu viacerých súborov je nutné nahrať dáta vo archíve typu .zip.

3.5.1 Konvertor formátu

Vstupné dáta vo formáte .obj je možné konvertovať do ďalšieho formátu. Ideálny stav systému bude ponuka viacerých najpoužívanejších formátov. Momentálne je podporovaný len formát .gltf. Tento formát má najväčšiu podporu pre výmenu medzi rôznymi nástrojmi. Po konvertovaní je vytvorený nový formát daného modelu obsahujúci informácie o novom .gltf súbore. Tento formát je možné ďalej efektívne komprimovať.

Konvertovanie prebieha použitím verejne dostupnej knižnice 'obj2gltf a prebieha na aplikačnom servery. Server si stiahne dáta z úložiska S3, pomocou knižnice 'adm-zip' rozbalí archív s .obj dátami, súbor skonvertuje a následne ho znovu ukladá do úložiska.

3.5.2 Kompresia dát

Dáta vo formáte .gltf podporujú veľmi účinnú komprimáciu pomocou nástroju 'Draco' od spoločnosti Google. Jedná sa o bezstratovú kompresiu, ktorá je určená pre urýchlenie prenosu modelov. Účinnosť kompresie sa líši v závislosti na dátach. Dokáže však dosahovať extrémne vysokú úroveň a to až niekoľko násobné zmenšenie dát.

Pre komprimáciu využívame knižnicu 'gltf-pipeline' pre prostredie NodeJs. Aplikačný server stiahne z úložiska AWS S3 model vo formáte .gltf a pri komprimácií z neho vytvorí .glb formát ktorý je úspornejší a tak dosiahneme ešte väčšej úspory veľkosti.

Kapitola 4

Testovanie

Implementovaný systém bol testovaný v najdôležitejších oblastiach z hľadiska jeho hlavnej funkcionality, a to je práca s 3D dátami prostredníctvom API. Cieľom testovanie je popísať približný čas potrebný na vykonanie operácii. Cieľom nie je podať presné štatistické informácie o rýchlosti operácie merané na reprezentatívnom vzorku.

4.1 Rýchlosť odpovede API

Systém vykonáva všetku svoju funkcionalitu v reakcii na prichádzajúce požiadavky z API, je teda dôležité zaistiť aby boli jeho odpovede čo najrýchlejšie. Ideálna doba odozvy pre náš systém je do 100 milisekúnd a akceptovateľná doba vo výnimočných situáciách je do 1 sekundy(pokiaľ nepočítame výpočetné operácie nad dátami 3D modelov).

4.1.1 Studený štart

Špecifikom systému je využitie serverless technológii, teda nie je nutne v každý okamžik spustený virtuálny server, ktorý by ihneď odpovedal a dochádza tak k studeným štartom. Výzvou je zabezpečiť aby takéto studené štarty serveru trvali rozumnú dobu.

Rýchlosť studeného štartu je závislá na viacerých faktoroch ako napríklad: výkon virtuálneho serveru, technológia aplikácie a veľkosť zdrojového kódu programu. Služba AWS Lambda je najlepšie optimalizovaná vykonávanie jazyku JavaScript, to je v náš prospech. Slabou stránkou je veľkosť zdrojového kódu,ktorá je momentálne 50 MB a tvoria ju závislé balíčky používaných knižníc. Zdrojový kód je bez minifikácie takže sa dá v prípade potreby zmenšiť.

Pre konfiguráciu s 1 GB operačnej pamäte bola nameraná priemerná doba studeného štartu na úrovni 2 921 milisekúnd. Meranie bolo prevedené manuálne pomocou

testovacieho nástroju Insomnia na vzorke 10 štartov. Pre zaistenie studeného štartu je nutné vždy znovu nahrať novú verziu funkcie do služby AWS Lambda.

Táto hodnota nie je dostačujúca pre produkčnú prevádzku a je potrebné zvýšiť výkon. Ďalší test bol vykonaný s veľkosťou operačnej pamäte 2 GB. Priemerná doba studeného štartu klesla na 943 milisekúnd, čo už je postačujúca hodnota.

4.1.2 Rýchlosť CRUD operácii

Trvanie od momentu odoslania požiadavky až po obdržanie odpovede je závislé okrem rýchlosti výpočtu servera aj na externých okolnostiach ako napríklad rýchlosť internetového pripojenia. Nedá sa teda určiť presné číslo garantované pre každého klienta. Potrebujeme sa ale uistiť, že odpoveď je schopná doraziť v rozumnom čase. Pri testovaní s veľkosťou operačnej pamäte 1 GB sme dosahovali výsledky v rozmedzí 90 - 130 milisekúnd čo je postačujúca úroveň. Pri testoch s 2 GB operačnej pamäte sa čas odpovede zrýchlil na 60 - 110 milisekúnd. Merania boli prevedené pomocou nástroja na záťažové testy serverov 'dotcom-monitor'.

4.1.3 Obmedzenia

Počas testovania sme narazili na niekoľko závažných obmedzení serverless služieb ktoré komplikujú ich používanie.

Časový limit odpovede

Služba AWS Lambda využíva na smerovanie a príjmanie HTTP požiadaviek službu API Gateway. Táto služba však nastavuje fixný limit na maximálnu dobu odozvy integrovaného servera na 29 sekúnd. V prípade služby AWS Lambda túto hodnotu nieje možné zmeniť aj keď samotné cloudové funkcie majú možnosť vykonávať svoju funkciu až 5 minút. Pre využitie plného limitu je potrebné vyvolať funkcie inými spôsobmi ako napríklad 'pub/sub' udalosti alebo priamo cez implementáciu AWS klienta.

Tento fakt komplikuje výpočetne náročné operácie nášho API a diktuje limit za ktorý musia byť prevedené. V budúcnosti bude musieť byť použitá alternatíva.

Kapacita priepustnosti databazy

Služba AWS DynamoDb definuje cenu používania na základe niekoľkých faktorov a jeden z nich je rezervovaná kapacita priepustnosti požiadaviek na čítanie / zápis. Pri testovaní nášho systému bola táto hodnota nastavená na 1 jednotku čítania a 1 jednotku zápisu za sekundu. Pri záťažovom teste so 100 emulovanými klientami s opakovaným dotazovaním 1 krát za sekundu sme narazili na spomínaný limit. Výsledok bol zabrzdenie exekúcie cloudových funkcii ktoré čakali na prístup a vytváranie

nových inštancii pre ďalšie prichádzajúce požiadavky. Čas odpovede postupne dosiahol limit a väčšina požiadaviek končila chybou.

Po zvýšení kapacity na 20 jednotiek čítania a zápisu a znížení počtu požiadaviek v záťažovom teste problém zmizol a mohli sme namerať rýchlosť odpovede CRUD operácii. Pre ďalšie zvyšovanie kapacity priepustnosti sme sa nerozhodli z dôvodu ceny služby.

4.2 Konverzia formátu modelov

Testovanie konverzie formátu .obj na formát .gltf prebehlo formou kontroly správnosti výstupu. Konvertované modely boli manuálne kontrolované pomocou nástroja Blender. Kontrolovaná bola korektnosť súboru, teda či je možné súbor načítať a ďalej korektnosť modelu v 3D priestore. Testovaná bolo zároveň doba potrebná pre operáciu konvertovania cez API. Čas bol meraný v nástroji Insomnia. Nasledujúca tabuľka ukazuje namerané časy konvertovania pre rôzne veľké modely.

NO.	Veľkosť originálneho formátu	Čas konverzie [ms]	korektnosť
1	173 kb	1297	ok
2	1,53 MB	1235	ok
3	26,7 MB	8906	ok
4	98,0 MB	25621	ok

Pre modely väčšie ako 100 MB končí požiadavka konvertovania cez API chybou kvôli časovému limitu rýchlosti odpovede ktorý sme popísali v sekcii 4.1.3.

4.3 Kompresia dát

Kompresia dát 3D modelov bola testovaná na vzorke dát 4 modelov vytvorených násobením počtu polygónov originálneho modelu. Kompresie bola vyvolaná cez API. Efektivitu kompresie znázorňuje nasledujúca tabuľka.

	verts	faces	tris	size B	
.obj	1 322	1 251	2 600	177 266	173 kB
.gltf	1 913	2 600	2 600	107 538	105 kB
Zmena	45%	108%	0%	-39%	
.glb	1 913	2 600	2 600	15680	15,3 kB
Zmena	0%	0%	0%	-85%	
.obj	21 411	21 728	43 456	1 639 519	1,56 MB
.gltf	38 824	42 824	42 824	1 606 432	1,53 MB
Zmena	81%	97%	-1%	-2%	
.glb	38 824	42 824	42 824	70864	69,2 kB
Zmena	0%	0%	0%	-96%	
.obj	345 021	346 262	692 524	27 140 838	25,8 MB
.gltf	534 777	690 064	690 064	28 233 175	26,7 MB
Zmena	55%	99%	0%	4%	
.glb	534 777	690 064	690 064	786624	1,76 MB
Zmena	0%	0%	0%	-97%	
.obj	1 293 495	1 295 206	2 590 412	102 275 650	97,5 MB
.gltf	1 916 297	2 586 639	2 586 639	102 829 070	98,0 MB
Zmena	48%	100%	0%	1%	
.glb	1 916 297	2 586 639	2 586 639	2874548	2,74 MB
Zmena	0%	0%	0%	-97%	

Obr. 4.1: Testovanie kompresie

Výsledná efektivita kompresie dosahuje až 97% zmenšenie veľkosti. Tento fakt môže byť podmienený typom dát, ktoré len násobia počet polygónov. Zdá sa že jedna z metód komprimácie formátu .gltf je eliminovanie opakujúcich sa zbytočných polygónov. Rýchlosť komprimácie zachytáva nasledujúca tabuľka.

NO.	Veľkosť originálneho formátu	Čas komprimácie [ms]	korektnosť
1	173 kb	1078	ok
2	1,53 MB	1453	ok
3	26,7 MB	7539	ok
4	98,0 MB	28369	ok

Rovnako ako pri konverzii, pre modely väčšie ako 100 MB končí požiadavka konvertovania cez API chybou kvôli časovému limitu rýchlosti odpovede ktorý sme popísali v sekcii 4.1.3.

Záver

Hlavným výstupom práce je funkčný systém pre správu obsahu 3D modelov tvorený zo serverovej a administračnej časti. Systém je spustený v produkčnom prostredí a je pripravený na využitie. Ďalšími výstupmi sú: Analýza 3D technológii, zdrojové kódy, inštrukcie k spusteniu, dokumentácia API, výsledky testovania. Použitím výstupov je teoretický možné navrhnutú službu zduplikovať.

Zadanie práce bolo splnené nakoľko bola implementovaná funkcionalita: zmeny formátu dát, správa dát, komprimácia, webové administračné rozhranie, verejné API, zabezpečenie pomocou užívateľských skupín, prístup aplikácii tretích strán pomocou prístupového kľúču a tiež bol systém otestovaný. Služba je dostupná na adrese 'https://bc-cms.firebaseapp.com' kde sa nachádza administračné rozhranie. API je dostupné na adrese: 'https://rl0459u769.execute-api.eu-central-1.amazonaws.com/dev'. Podrobný popis ako systém spustiť pomocou zdrojových kódoch sa nachádza v jednotlivých projektoch v súbore 'README.md'.

Oproti pôvodnému očakávaniu neboli implementované 2 rôzne formáty, do ktorých by bolo možné konvertovať vstupné dáta. Tento nedostatok je kompenzovaný formátom .glb, do ktorého je model konvertovaný pri komprimácii. Spolu so základným formátom konvertovania .gltf tak spĺňajú zadanie no patria k rovnakému typu formátu. V budúcnosti bude vhodné implementovať ďalšie formáty pre maximalizáciu použiteľnosti systému.

Ďalšou potrebnou funkcionalitou, ktorá nie je implementovaná, je možnosť užívateľov upravovať alebo aspoň pridávať ďalšie používateľské účty pod svoju skupinu. Tento nedostatok by pri reálnom používaní systému spôsobil zvýšenú záťaž na personál podpory služby ktorý by musel všetky požiadavky spracovávať.

V rámci implementácie a testovania sme objavili limity použitých služieb. Prvým závažným bol limit veľkosti požiadavku na službu AWS Lambda vo výške 6 MB. Obišli sme ho využitím podpísaných jednorazových nahrávacích adries služby S3 kde klient priamo nahrá svoje súbory. Ďalšou limitáciou je maximálny povolený čas odpovede služby AWS Lambda vo výške 29 sekúnd, ktorý obmedzuje možnosť výpočetne náročných operácií. Limituje veľkosť súboru ktorý dokáže server spracovať na 100 MB pri konfigurácii s 2 GB operačnej pamäte. Tretím limitom je kapacita priepustnosti operácii databáze. Služba AWS dynamoDB spoplatňuje rezervovanú kapacitu a tak pri nastavená nízkeho limitu môže dosť k preťaženiu pri rýchlom náraste požiadavkou.

Využitie služieb platformy AWS od spoločnosti Amazon hodnotíme ako správnu

voľbu aj napriek spomínaným limitáciám. Využité služby excelentne splnili účel. Silnou stránkou je ich kompatibilita a poskytnuté nástroje. K uľahčeniu využitia spomínaných služieb výrazne prispel nástroj 'Serverless' ktorý uľahčuje konfiguráciu a správu nasadenia využitých služieb.

Systém v aktuálnom stave dokáže asistovať v prípade nášho ilustračného príkladu s internetovým obchodom ktorý využíva 3D modely k prezentácii produktov. Možnosť ukladať a komprimovať dáta nahradzuje nutnosť využitia iných služieb či potrebných úprav na strane obchodu. Prepojenie dát modelov s produktami obchodu je možné cez dodatočný parameter odkazujúci na adresu 3D modelu. V prípadne komplexnejších aplikácii pomôže dostupné API.

Systém pravdepodobne nájde využitie pri integrácii s aplikáciami na prezentovanie produktov alebo pri produktových konfigurátoroch v eCommerce. Tento fakt je daný zvyšujúcim sa dopytom po 3D modeloch a zároveň faktom že využívané systémy a rozpočty na IT infraštruktúru sú na nízkej úrovni. V ostatných oblastiach existujú služby ponúkajúce špecifické riešenia alebo neexistuje veľká potreba využívať obecné externé systémy.

Literatúra

[1] AWS DynamoDB. AWS DynamoDB [online]. [cit. 2019-05-20]. Dostupné z: https://aws.amazon.com/dynamodb/

[2] AWS Lambda. AWS lambda [online]. [cit. 2019-05-20]. Dostupné z: https://aws.amazon.com/lambda/

[3] AWS S3. AWS S3 [online]. [cit. 2019-05-20]. Dostupné z: https://aws.amazon.com/s3/

[4] BARFIELD, Woodrow a Thomas CAUDELL. Fundamentals of wearable computers and augumented reality. Mahwah, NJ: Lawrence Erlbaum Associates, c2001. ISBN 0805829024.

[5] HUGHES, John F., Andries VAN DAM a Morgan MCGUIRE. Computer Graphics: Principles and Practice. 3 edition. Addison-Wesley Professional, 2013. ISBN 0321399528.

[6] MCHENRY, Kenton a Peter BAJCSY. An Overview of 3D Data Content, File Formats and Viewers. An Overview of 3D Data Content, File Formats and Viewers. 2008, 2008, 1-3.

[7] NodeJs. NodeJs [online]. [cit. 2019-05-20]. Dostupné z: https://nodejs.org/en/

[8] SCHROEDER, Will, Ken MARTIN a Bill LORENSEN, 2003. *The Visualization Toolkit An Object-Oriented Approach To 3D Graphics.* 3rd edition. Kitware. ISBN 1930934076.

[9] Serverless. Serverless [online]. [cit. 2019-05-20]. Dostupné z: https://serverless.com/

[10] STOJANOVIC, Slobodan a Aleksandar SIMOVIC. Serverless applications with Node.js: using AWS Lambda and Claudia.js. Shelter Island: Manning Publications, [2019]. ISBN 1617294721.

[11] TypeScript. TypeScript lang [online]. [cit. 2019-05-20]. Dostupné z: https://www.typescriptlang.org/

[12] WebGL. Khronos [online]. Khronos [cit. 2019-01-07]. Dostupné z: https://www.khronos.org/webgl/

[13] WebGL support. Can I use [online]. [cit. 2019-01-07]. Dostupné z: https://caniuse.com/#feat=webgl

Dodatok A

Dokumentácia API

A.1 Autentizácia

POST	/auth/login	Autentizuje užívateľa prihlasovacími údajmi a vráti kľúč.
Parametre	JSON Body	{ credentials: { email: String, password: String } }
Odpoveď	200 OK	{ token: String, roles: [<USER | ADMIN>] }
	400 BAD REQUEST	// neplatný tvar prihlasovacích údajov
	401 UNAUTHORIZED	// neplatné prihlasovacie údaje

A.2 Konvertovanie formátu

Operácie ktoré konvertujú formát modelu.

POST	/models/:id/convert/to-gltf	Konvertuje do .gltf
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor modelu.
Odpoveď	200 OK	{ id: String groupId: String modelId: String extension: String s3Key: String url: String size: Number }
	400 BAD REQUEST	// model nemá žiadne dáta alebo formát už existuje
	500 SERVER ERROR	// chyba pri konvertovaní
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

A.3 Komprimácia formátu

Operácie ktoré komprimujú formát modelu.

POST	/models/:id/compress	Komprimuje .gltf a konvertuje na .glb
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor modelu.
Odpoveď	200 OK	{ id: String groupId: String modelId: String extension: String s3Key: String url: String size: Number }
	400 BAD REQUEST	// model nemá žiadne dáta alebo formát už existuje
	500 SERVER ERROR	// chyba pri konvertovaní
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

A.4 Skupiny 'Groups'

Operácie nad entitou skupina. Prístup k týmto operáciám je povolený len užívateľskej role 'ADMIN'.

POST	/groups	Vytvorí novú skupinu
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	JSON Body	{ name: String }
Odpoveď	200 OK	{ id: String name: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

GET	/groups/:id	Vráti skupinu podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor skupiny.
Odpoveď	200 OK	{ id: String name: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

GET	/groups	Vráti všetky skupiny.
Hlavička	Authorization:	Bearer <TOKEN>
Odpoveď	200 OK	[{ id: String name: String }]
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

PUT	/groups/:id	Upraví skupinu podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor skupiny.
	JSON Body	{ id: String name: String }
Odpoveď	200 OK	{ id: String name: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

DELETE	/groups/:id	Odstráni skupinu podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor skupiny.
Odpoveď	200 OK	{ id: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

A.5 Používatelia 'Users'

Operácie nad entitou používateľ. Prístup k týmto operáciám je povolený len užívateľskej role 'ADMIN'.

POST	/users	Vytvorí nového používateľa
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	JSON Body	{ name: String groupId: String email: String password: String roles: [<USER | ADMIN>] }
Odpoveď	200 OK	{ id: String groupId: String name: String email: String password: String roles: [<USER | ADMIN>] }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

GET	/users/:id	Vráti používateľa podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor používateľa.
Odpoveď	200 OK	{ id: String groupId: String name: String email: String password: String roles: [<USER | ADMIN>] }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

GET	/users	Vráti všetkých používateľov.
Hlavička	Authorization:	Bearer <TOKEN>

Odpoveď	200 OK	[
		{
		id: String
		groupId: String
		name: String
		email: String
		password: String
		roles: [<USER | ADMIN>]
		}
		]

	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

PUT	/users/:id	Upraví používateľa podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor používateľa.

	JSON Body	{
		id: String
		groupId: String
		name: String
		email: String
		password: String
		roles: [<USER | ADMIN>]
		}

Odpoveď	200 OK	{
		id: String
		groupId: String
		name: String
		email: String
		password: String
		roles: [<USER | ADMIN>]
		}

	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

DELETE	/users/:id	Odstráni používateľa podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor používateľa.
Odpoveď	200 OK	{ id: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

A.6 Modely 'Model'

Operácie nad entitou Model. Prístup k týmto operáciám je povolený len používateľom danej skupiny alebo užívateľskej role 'ADMIN'.

POST	/models	Vytvorí nový model
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	JSON Body	{ name: String groupId: String }
Odpoveď	200 OK	{ id: String groupId: String name: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

GET	/models/:id	Vráti model podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor modelu.
Odpoveď	200 OK	{ id: String groupId: String name: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

GET	/models	Vráti všetky modely.
Hlavička	Authorization:	Bearer <TOKEN>
Odpoveď	200 OK	[{ id: String groupId: String name: String }]
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

PUT	/models/:id	Upraví model podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor modelu.
	JSON Body	{ id: String groupId: String name: String }
Odpoveď	200 OK	{ id: String groupId: String name: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

DELETE	/models/:id	Odstráni model podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor modelu.
Odpoveď	200 OK	{ id: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

A.7 Formát modelu 'Format'

Operácie nad entitou Format. Prístup k týmto operáciám je povolený len používateľom danej skupiny alebo užívateľskej role 'ADMIN'. Každý formát je priradený nejakému modelu.

POST	/formats	Vytvorí nový formát
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	JSON Body	{ groupId: String modelId: String extension: String s3Key: String url: String size: Number }
Odpoveď	200 OK	{ id: String groupId: String modelId: String extension: String s3Key: String url: String size: Number uploadUrl: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

GET	/formats/:id	Vráti formát podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor formátu.
Odpoveď	200 OK	{ id: String groupId: String modelId: String extension: String s3Key: String url: String size: Number }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

GET	/formats	Vráti všetky formáty.
Hlavička	Authorization:	Bearer <TOKEN>
Odpoveď	200 OK	[{ id: String groupId: String modelId: String extension: String s3Key: String url: String size: Number }]
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

PUT	/formats/:id	Upraví formát podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor formátu.
	JSON Body	{ id: String groupId: String modelId: String extension: String s3Key: String url: String size: Number }
Odpoveď	200 OK	{ id: String groupId: String modelId: String extension: String s3Key: String url: String size: Number }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

DELETE	/formats/:id	Odstráni formát podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor formátu.
Odpoveď	200 OK	{ id: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

A.8 Api kľúč 'ApiKey'

Operácie nad entitou ApiKey. Prístup k týmto operáciám je povolený len používateľom danej skupiny alebo užívateľskej role 'ADMIN'.

POST	/apikeys	Vytvorí nový Api kľúč
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	JSON Body	{ groupId: String name: String value: String }
Odpoveď	200 OK	{ id: String groupId: String name: String value: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

GET	/apikeys/:id	Vráti Api kľúč podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor Api kľúču.
Odpoveď	200 OK	{ id: String groupId: String name: String value: String }
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

GET	/apikeys	Vráti všetky Api kľúče.
Hlavička	Authorization:	Bearer <TOKEN>

Odpoveď	200 OK	[
		{
		id: String
		groupId: String
		name: String
		value: String
		}
		]

	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

PUT	/apikeys/:id	Upraví Api kľúč podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor Api kľúču.

	JSON Body	{
		id: String
		groupId: String
		name: String
		value: String
		}

Odpoveď	200 OK	{
		id: String
		groupId: String
		name: String
		value: String
		}

	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

DELETE	/apikeys/:id	Odstráni Api kľúč podľa parametru ':id'.
Hlavička	Authorization:	Bearer <TOKEN>
Parametre	:id (query)	Identifikátor Api kľúču.
Odpoveď	200 OK	`{ id: String }`
	400 BAD REQUEST	// neplatný tvar požiadavku
	401 UNAUTHORIZED	// chýbajúci alebo neplatný kľúč
	403 FORBIDDEN	// chýbajúci prístup k požadovanej operácii

Dodatok B

Prístupové údaje k produkčnému prostrediu

Systém je produkčne spustený v službe AWS. Pre testovanie a demonštráciu funkcionality je možné využiť nasledujúce prihlasovacie údaje:

Admin

meno: admin@cms.com
heslo: password.

Užívateľ

meno: test@furniture.com
heslo: furniture

Administračné prostredie je dostupné na adrese:

`https://bc-cms.firebaseapp.com`

API serveru je dostupné na adrese:

`https://rl0459u769.execute-api.eu-central-1.amazonaws.com/dev/`

Dodatok C

Obsah priloženého dátového archívu

Archív obsahuje zdrojový kód systému spolu s popisom ako jednotlivé časti spustiť a nasadiť do produkčného prostredia.

/backend

Zložka obsahuje zdrojový kód aplikačného serveru. V súbore 'README.md' sa nachádza popis spustenia a produkčného nasadenia. Pre produkčné spustenie je nutné vytvoriť si vlastný účet v službe AWS.

/administration

Zložka obsahuje zdrojový kód administračnej webovej aplikácie. V súbore 'README.md' sa nachádza popis spustenia a produkčného nasadenia.

/test-data

Zložka obsahuje súbory 3D modelov ktoré boli použité pri testovaní.

BAKALARSKA-PRACA.pdf

Text práce vo formáte PDF.