VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Matúš Paulech

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

INTERAKTIVNÍ WEBOVÁ PREZENTACE AUDIOVIZUÁLNÍCH DĚL
INTERACTIVE WEB PRESENTATION OF AUDIOVISUAL WORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Matúš Paulech**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Pavel Sikora**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Matúš Paulech ***ID:*** 211268
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Interaktivní webová prezentace audiovizuálních děl

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit interaktivní webovou prezentaci video nahrávek. Díla jsou označena klíčovými slovy a archivářskými daty, díky kterým bude umožněno uživateli webové prezentace filtrovat jednotlivé položky v databázi a ty následně přehrát. Obsah jednotlivých snímků video záznamů je zároveň označen klíčovými slovy, která budou uživateli zobrazována v průběhu přehrávání. Student nastuduje problematiku tvorby backendu a frontendu a na základě nabytých znalostí zhotoví webovou prezentaci. Pro backend bude využita platforma .NET Core.

DOPORUČENÁ LITERATURA:

[1] ASP.NET documentation. 2020 Microsoft, [cit. 2020-09-13]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core,

[2] CONNOLLY, Randy a Ricardo HOAR. Fundamentals of Web Development. 2. Pearson, 2017. ISBN 978-0134481265.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: Ing. Pavel Sikora

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Bakalárska práca obsahuje prezentáciu audiovizuálnych diel spracované kalsifikátorom založenom na neurónových sieťach (deep learning a umelá inteligencia). Práca sa zaoberá nahrávaním video súborov do webovej aplikácie a ich následným prehraním na webovej stránke. K videám sú pridané dáta, ktoré boli spracované neurónovými sieťami. Tieto dáta sú rozdelené podľa klasifikácie do daných tagov, v ktorých je zapísané čo sa na danom snímku práve nachádza. Pomocou týchto tagov sa dajú klasifikovať videá a taktiež sa tieto klasifikácie zobrazujú v reálnom čase počas prehrávania videa.

KĽÚČOVÉ SLOVÁ

web, webová stránka, webová aplikácia, video, prezentácia

ABSTRACT

The bachelor thesis contains a presentation of audiovisual works processed by a calibrator based on neural networks (deep learning and artificial intelligence). The work deals with uploading video files to a web application and then playing them on a website. Data, that have been processed by neural networks, are added to the videos. This data is divided according to the classification into given tags, in which it is written what is currently in the image of video. These tags can be used to classify videos, and these classifications are also displayed in real time during video playback.

KEYWORDS

web, website, web application, video, presentation

PAULECH, Matúš. *Interaktivní webová prezentace audiovizuálních děl*. Brno, 2021, 46 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Pavel Sikora

VYHLÁSENIE

Vyhlasujem, že svoju bakalársku prácu na tému „Interaktivní webová prezentace audiovizuálních děl“ som vypracoval samostatne pod vedením vedúceho bakalárskej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád bych poděkoval vedoucímu diplomové práce panu Ing.Pavel Sikora, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Zoznam obrázkov

Úvod

Internet dnes využíva takmer každý a je bežnou súčasťou nášho života. Dá sa využiť na rôzne účely, ako napríklad šírenie informácií, kolektor dát a rôzne iné. Táto práca sa zameriava na šírenie audiovizuálnych diel na internete a tým lepšie sprístupnenie umenia pre verejnosť, ktorá je schopná sa týmto spôsobom aj vzdelávať.

Ak sa chcú umelci v dnešnej dobe presadiť je pre nich dôležité, aby boli jeho súčasťou a kráčali s dobou. Ešte lepšie je ak má daný umelec vlastnú webovú stránku kde je schopný prezentovať svoju prácu, alebo časť práce, a tým vedel nalákať záujemcov na prípadné výstavy, vernisáže, a takto zvýšiť svoju šancu na úspech presadiť sa medzi konkurenciou.

Podstatou práce je prezentácia audiovizuálnych diel, ktoré boli spracované klasifikátorom založenom na neurónových sieťach (deep learning, umelá inteligencia) a ich výsledné dáta sa ďalej v práci využívajú. Práca sa zaoberá nahrávaním audiovizuálnych diel a dát spracovaných neurónovými sieťami do databázy a následne ich zobrazovaním na webovej stránke pre širokú verejnosť. Webová aplikácia umožňuje nahrávať a následne prehrávať audiovizuálne diela, a zároveň umožní filtrovať videá do kategórií a tým je obsah webovej aplikácie prehľadný a pre používateľa ľahko ovládateľný. Cieľom práce je vytvoriť interaktívnu webovú prezentáciu video nahrávok. Diela sú označené kľúčovými slovami a archivárskymi dátami, vďaka ktorým bude umožnené užívateľovi webovej prezentácie filtrovať jednotlivé položky v databáze a tie následne prehrať.

1 Úvod do webových technológií

Webové technológie v súčasnej dobe majú veľký význam pre dnešnú spoločnosť. Mnoho vecí dnes prebieha práve na internete, ktorého súčasťou sú rôzne webové stránky. Vďaka tomu sa otvára možnosť pre mnohé odvetvia zviditeľniť sa na internete. Takmer každá väčšia firma má v sú časnosti svoju vlastnú webovú stránku, pomocou ktorej sa môže prezentovať širokej verejnosti a v mnohých prípadoch sa dá na takýchto stránkach aj nakupovať.

Každý kto sa v súčasnej dobe rozhodne prezentovať nejakou formou na internete, má túto možnosť veľmi jednoduchú. Na rozdiel od firiem, ktoré si často platia programátorov na tvorenie webovej stránky, človek ako jedinec častokrát nemá takéto možnosti. Na internete existuje však mnoho šablón na vytvorenie jednoduchej webovej stránky kde daný jedinec môže prezentovať seba, svoje myšlienky, alebo napríklad aj svoj biznis a tak sa zviditeľniť širokej verejnosti.

1.1 World Wide Web

World Wide Web (v skratke WWW) sa dostalo na verejnosť v deväťdesiatych rokoch minulého storočia. Myšlienka WWW bola vytvoriť univerzálnu databázu na predávanie informácií užívateľom a umožnenie ľahkého prístupu k týmto informáciám [3].

Spoločnosť veľmi rýchlo zistila, že internet má veľmi veľké využitie. Firmy si začali vytvárať svoje vlastné webové stránky, kde zverejňovali svoje produkty, robili si reklamu a pomaly začali aj predávať svoj tovar cez internet. Začali vznikať nové komunikačné prostriedky, ako bol email a neskôr sociálne siete, ako Facebook a Twitter, kde sa zrazu ocitlo veľmi veľa ľudí a začali medzi sebou zdieľať rôzne informácie ako napríklad fotky zo svojho života, rôzne statusy, kde vyjadrujú svoje názory a podobne.

Bežný užívateľ si zrazu, bez toho aby vyšiel vôbec von, mohol prezrieť rôzne veci na internete, kúpiť si ich, mohol komunikovať so svojimi priateľmi, kolegami v práci, alebo rodinou pomocou emailu a sociálnych sietí, a z nich sa dozvedieť rôzne informácie o živote svojich priateľov.

1.2 Vytváranie webovej aplikácie

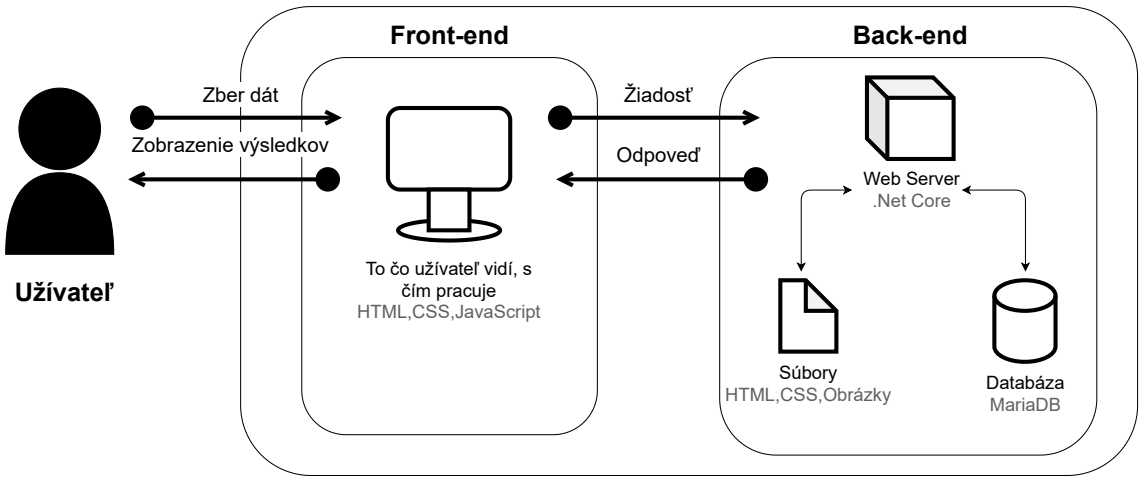
Webová aplikácia sa rozdeľuje na dve základné časti a to je front-end a back-end. Každá táto časť má inú úlohu a iné využitie. Front-end sa tiež môže nazývať užívateľská strana. Je to časť, kde je koncentrovaný hlavne webový dizajn, zatiaľ čo

back-end taktiež nazývaný ako serverová strana, kde sa dejú najmä operácie s databázou, so serverom a rôzne iné operácie, ktoré sú bežným užívateľom skryté, viď. obr. 1.1.

- **Front-end**
  Front-end (užívateľská strana) je v podstate viditeľná časť webovej stránky, určená pre užívateľa. Front-end programátor sa často nazýva web dizajnér. Na vytváranie webovej stránky sa najviac používajú jazyky ako je HTML (HyperText Markup Language) a CSS (Cascading Style Sheet). Pomocou týchto jazykov je zabezpečená vizuálna podoba stránky. Je to tá časť webovej aplikácie, kde užívateľ vidí rôzne elementy, ako je napríklad text, obrázok, pozadie atď. Tieto elementy sú presne definované a každý z týchto elementov môže mať rôzne parametre, ktoré sa týkajú napríklad veľkosti, pozície na webovej stránke, ale taktiež aj akcie, ktorú má vykonávať. Elementy sú vytvárané pomocou jazyka HTML, čo je v podstate hypertextový dokument, ktorý je uložený na mieste zvanom webový server. Jazyk CSS sa používa na následné vytváranie štýlov pre dané elementy. Štýly môžu byť definované na každý element zvlášť a tým vytvoriť grafickú podobu stránky.
- **Back-end**
  Back-end (serverová strana) určuje ako sa má stránka správať pri určitých príkazoch užívateľa. Back-end programátor sa môže nazývať aj developer. Na rozdiel od webových dizajnérov, ktorí sa starajú hlavne o to, ako sa má stránka užívateľovi zobrazovať, developeri sa starajú najmä o to, ako sa má stránka správať. Snažia sa vytvoriť dynamickú stránku, ktorá je schopná odpovedať na dané zmeny a aktualizovať sa v reálnom čase. Back-end sa vytvára v jazykoch ako sú napríklad PHP a .Net. Tieto jazyky sa starajú o operácie, ktoré sú spojené s komunikáciou so serverom. Keď užívateľ pošle požiadavku, napríklad o zobrazení informácií o jeho profile, tieto jazyky zabezpečia to, aby sa na webovej stránke zobrazil daný obsah. Proces prebieha tak, že .Net spracuje požiadavku z webovej stránky, ktorú zadal klient. Následne ho pošle na server, odkiaľ vyberie informácie z jeho profilu, ktoré majú byť zobrazené (meno, fotka...). Následne tieto informácie pošle na webovú stránku a tá sa už postará o to, aby boli správne užívateľovi zobrazené. Back-end sa teda stará o nahrávanie, editovanie, mazanie a selektovanie vecí na servery.

1.3 Model-View-Controller

Model-View-Controler, tiež známy pod skratkou MVC je model, ktorý sa používa hlavne pri vytváraní webových aplikácií [2]. Model rozdeľuje aplikáciu do troch častí a tieto časti sú navzájom na sebe závislé, viď. obr. 1.2. Zabezpečuje spracovanie



Obr. 1.1: Architektúra webovej aplikácie.

požiadaviek užívateľa a následné zobrazenie výsledku užívateľovi. Tento model sa stal najpoužívanejší pre webové aplikácie a pre vývoj grafického používateľského rozhrania, v skratke GUI (Graphical User Interface).

Ako už bolo spomenuté, MVC sa skladá z troch častí:

- Model,
- View,
- Controller.

1.3.1 Model

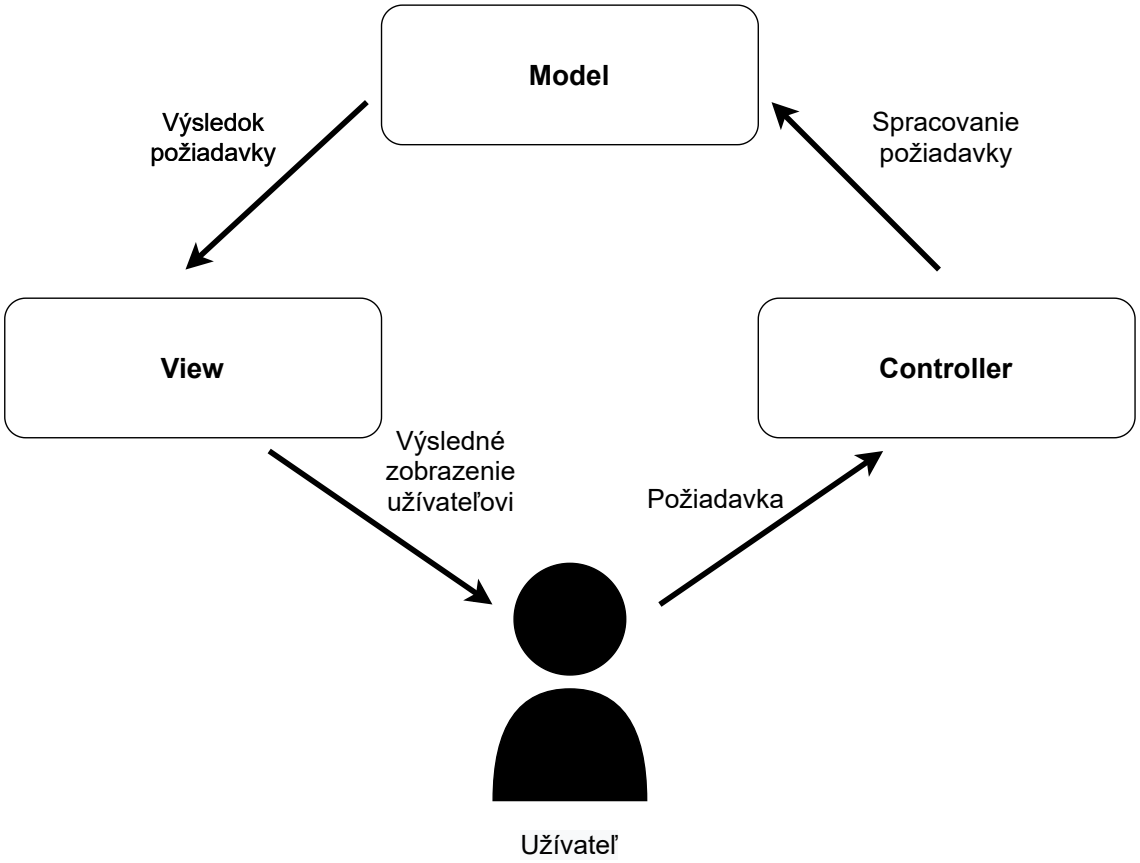
Model je v podstate kolekcia tried, kde aplikácia pracuje s takzvanou data logic, alebo tiež bussines logic [2]. Je to tá časť aplikácie, ktorá teda obsahuje iba aplikačné informácie. Neobsahuje žiadne informácie o tom, ako zobraziť dáta užívateľovi a je nezávislý na užívateľskom rozhraní.

1.3.2 View

View je v podstate HTML kód, ktorý rozhoduje ako bude UI (User Interface) vyzerať [2]. Spracováva informácie a zobrazuje ich v rôznych grafoch, tabuľkách, či súboroch, ako je napríklad obrázok, alebo video. Je to v podstate časť MVC, ktorá je zodpovedná za vizualizáciu informácie.

1.3.3 Controller

Hlavnú časť programu zabezpečuje práve Controller. Spracováva požiadavky užívateľa, ktoré na základe inštancií jednotlivých modelov napĺňa dátami. Tieto dáta



Obr. 1.2: Štruktúra Model-View-Controller.

následne posiela do časti zvanej View, kde sa spracujú a zobrazia. Controller sa stará o to, aké presne informácie sa majú poslať do časti View. Tieto informácie častokrát berie zo serveru, z databázy a podľa toho, ktorý objekt je zavolaný, také dáta sa buď pridajú, odstránia, upravia, alebo vytiahnu z danej databázy (z daného servera).

1.4 Databáza

Databáza nie je nič iné, ako súbor súvisiacich informácií [1]. Aj keď v dnešnej dobe si pod slovom databáza veľa ľudí predstaví nejaké úložisko informácií v elektronickej podobe, nemusí tomu byť vždy tak. Databáza môže byť nielen v elektronickej podobe, ale aj v papierovej. Takouto papierovou databázou je napríklad zoznam zamestnancov firmy, ktorý uchováva informácie, ako je napríklad meno, telefónne číslo, adresa a pracovná pozícia. Podstatou databázy je to, aby boli dáta v nej nejako rozumne usporiadané, napríklad zoznam zamestnancov podľa abecedy, aby boli prehľadné, aby boli ľahko upravovateľné a aby sa v nich dali čo najjednoduchšie potrebné informácie vyhľadať. Do databázy sa musia dať vložiť nové informácie, musia sa dať v nej upravovať už existujúce informácie a poprípade sa musia dať z databázy

dané informácie odstrániť.

Už v dávnych dobách ľudia využívali databázy, aj keď vtedy to tak nenazývali. Takéto databázy využívali napríklad už Rimania, keď uskutočňovali sčítanie ľudu. Táto databáza bola samozrejme na papieri a bola pri tom vyžadovaná veľmi veľká logistika, za cieľom dosiahnutia výsledku. Takéto databázy sa robili stáročia. Ľudia si zapisovali svoje rodokmene, koľko majú akých vecí, potravy a podobne.

So začínajúcim nárastom počtov strojov v spoločnosti si však ľudia začali klásť otázku, ako tieto stroje využiť na zber informácií. Prvým praktickým spôsobom ako spracovať informácie pomocou strojov boli dierne štítky. Tento spôsob bol využitý pri sčítaní ľudu v Spojených štátoch amerických v roku 1890. Vďaka tomuto spôsobu bolo sčítanie 65 miliónov ľudí vyhodnotené už za štyri týždne, čo bol na tú dobu veľký pokrok [1].

S príchodom prvých počítačov prišli aj prvé programy na databázu v počítači. Prvý takýto program prišiel už v päťdesiatych rokoch minulého storočia a zaoberal sa hlavne programovými jazykmi a algoritmami. So stále zväčšujúcim sa záujmom o počítače a začatím ich využívania v reálnom svete prišli aj väčšie požiadavky. V roku 1960 Charles William Bachman navrhol prvý DBMS (Database Managment System). Bol to databázový systém, ktorý umožňoval získať konkrétne informácie z databázy.

Ako sa v šesťdesiatych rokoch dvadsiateho storočia zvyšovala rýchlosť, flexibilita a dostupnosť, počítače začali byť čím ďalej, tým viac populárne. Pribúdali rôzne druhy databázových systémov. V spoločnosti bola požiadavka na vytvorenie nejakého štandardu. Tejto myšlienky sa opäť zhostil už zmienený Bachman a vznikla takzvaná Database Task Group, ktorej lídrom bol práve Bachman. Táto spoločnosť si dala za úlohu vytvoriť štandardizovaný jazyk, ktorý pomenovala COBOL (Common Business Oriented Language). Tento štandard spoločnosť predstavila v roku 1971 a je známy pod názvom "CODASYL approach". CODASYL aproach bol veľmi komplikovaný a vyžadoval veľké znalosti. Vyhľadávanie pomocou tohto štandardu bolo možné vykonať jednou z troch možností:

- používanie primárneho kľúča,
- presúvanie vzťahov z jedného záznamu do druhého,
- skenovanie všetkých záznamoch pomocou sekvenčných príkazov.

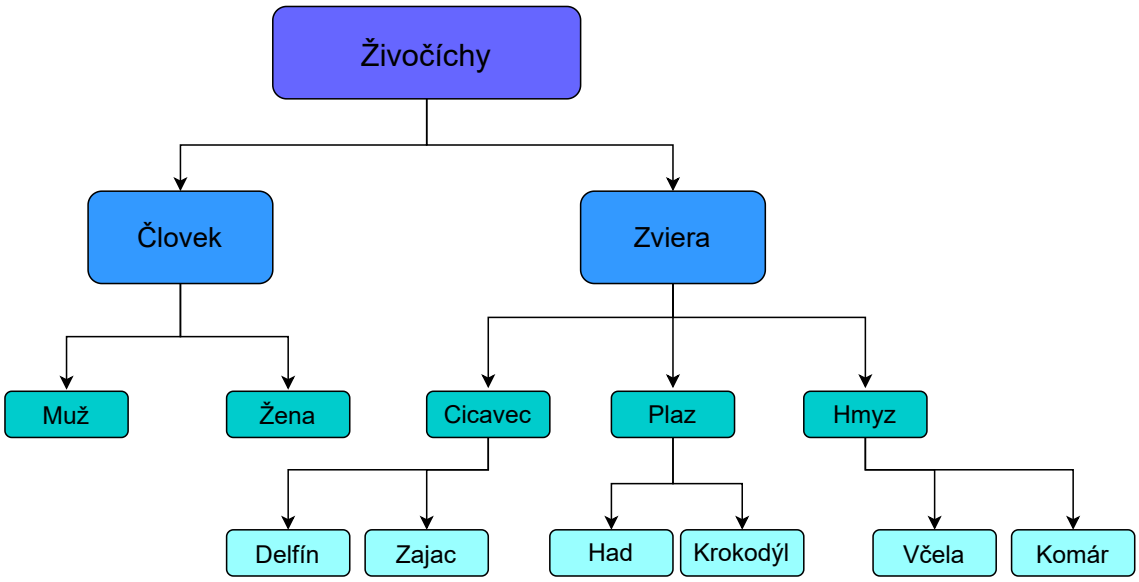
Výsledkom veľkej zložitosti tohto štandardu Americká spoločnosť IBM v roku 1974 navrhla vlastný jazyk SEQUEL (Structured English QUEry Language), ktorý sa neskôr zmenil na SQL (Structured Querry Language) a rýchlo sa stal jednotkou na trhu. Tento jazyk je používaný dodnes a jeho posledná verzia bola vydaná v roku 2019 s názvom SQL:2019 [7].

1.4.1 Typy databáz

Prvým typom databázy (databázovým modelom) bol hierarchický model, viď. obr. 1.3. Tento model vznikol v šesťdesiatych rokoch dvadsiateho storočia a zakladal sa na dátových vzťahoch rodič-potomok [1].

Po čase bol tento model nedostatočný a nahradil ho sieťový model, kde mohol mať jeden potomok nie iba jedného rodiča, ale aj viacero. Tento model sa udržal podstatne dlhšie, ale s vývojom technológií aj on už prestával byť dostatočný. Preto sa musel navrhnúť úplne nový model a tento model sa volal relačná databáza [1].

Podstatou relačnej databázy je uchovávanie dát v tabuľkách. V týchto tabuľkách sú dáta uchovávane v riadkoch a stĺpcoch, kde stĺpce reprezentujú rovnaký typ dát, ako je napríklad meno, priezvisko a vek a riadky reprezentujú nové konkrétne dáta, napríklad Peter Kováč 42. Tento typ databázy pracuje s jazykom SQL. Relačné databázy sa využívajú dodnes. Tieto databázy v súčasnosti využívajú rôzne systémy ako je napríklad Microsoft SQL Server, MySQL, MariaDB a SQLite.



Obr. 1.3: Hierarchický model databázy.

1.4.2 MariaDB

MariaDB vznikla odtrhnutím sa skupiny vývojárov od MySQL. To sa stalo na základe odkúpenia MySQL spoločnosťou Oracle Corporation v roku 2010. MariaDB je založená na MySQL a tým pádom je plne kompatibilná s MySQL. Takisto ako MySQL aj MariaDB využíva jazyk SQL. To že je MariaDB založená na MySQL

umožňuje jednoduchý presun dát z MySQL do MariaDB bez potreby niečo meniť. V súčasnosti sa MariaDB vyznačuje vyššou rýchlosťou oproti MySQL a taktiež väčšou funkcionalitou [4].

1.5 SQL

SQL (Structured Querry Language) bol vyvinutý Americkou firmou IBM (International Business Machines Corporation). V roku 1974 Donald Chamberlin a další vyvinuli System R pre IBM ako prvý prototyp relačnej databázy a jazyk pomenovali SEQUEL (Structured English QUEry Language). Medzi rokmi 1976 a 1977 bol tento jazyk kompletne prepísaný a nazvali ho SQL a to z právnych dôvodov, pretože spoločnosť Hawker Siddeley Aircraft Company si tento názov rezervovala pre seba. V roku 1978 sa SQL podrobilo skúške ako obstojí v reálnom svete. V testoch uspel na výbornú pretože dokázal svoju praktickosť. V júni 1979 Relational Software (dnes známa ako Oracle) vydala prvý SQL produkt RDBMS (relational database managment system), ktorý slúžil na spravovanie relačnej databázy. SQL sa stal oficiálnym štandardom v roku 1986. Nasledovali ďalšie vylepšené verzie ako SQL-92, známy aj ako SQL2, SQL:1999 známy aj ako SQL3. SQL jazyk je aj v dnešnej dobe stále aktuálny a používaný. Jeho najnovšia verzia vyšla v roku 2019 a má názov SQL:2019 [5].

1.6 .NET

.Net je platforma vyvinutá spoločnosťou Microsoft. Je určená pre budovanie rôznych aplikácií, ako sú napríklad desktopové aplikácie, webové aplikácie, mobilné aplikácie, hry atď. Je veľmi flexibilný, má ohromné množstvo funkcií a knižníc. Knižnice sú jednou z najväčších výhod a preto je .NET v dnešnej dobe tak populárny. Výhoda je tá, že pokiaľ je potrebné použiť nejakú knižnicu v programe, tak je takmer vždy obsiahnutá v .NET, a preto sa nemusí sťahovať do programu knižnica z tretej strany. Najviac používané jazyky .NET sú C# F#. .NET má takzvanú vlastnosť „Runtime“ čo znamená že akonáhle skompilujete kód, viete ho spustiť na mobile, desktope, alebo na servery. Ďalšou výhodou .NET je, že je veľmi kompatibilný. Aplikácia vytvorená pomocou .NET môže byť spustená na zariadeniach so systémom Windows, Linux, či dokonca macOS. To isté platí aj pre mobilné zariadenia. Aplikácia, ktorá bude kompatibilná s operačným systémom Android, bude kompatibilná aj s IOS. Podpora pre .NET a takisto samotný .NET je zdarma. Na pracovanie s platformou .NET sa využíva program, ktorý je taktiež vytvorený spoločnosťou Microsoft a nazýva sa Microsoft Visual Studio.

1.7 Microsoft Visual Studio

Microsoft Visual Studio je vývojové prostredie vytvorené spoločnosťou Microsoft. Je to editor kódu, ktorý má veľmi širokú škálu využitia. Microsoft Visual Studio sa dá využiť na vytvorenie konzolovej aplikácie, ale aj na vytvorenia aplikácie s grafickým rozhraním, tzv. GUI.

Ako bolo už spomenuté Microsoft Visual Studio môže pracovať na platforme .NET ale aj na iných platformách, ako je napríklad Windows API. To umožňuje vytvorenie takmer akejkoľvek aplikácie, ako sú napríklad rôzne počítačové programy, webové aplikácie a mobilné aplikácie.

Microsoft Visual Studio podporuje širokú škálu programovacích jazykov v rátane C++, C#, Visual Basic .Net, ale aj jazyky ako napríklad CSS, HTML, JavaScript, pomocou ktorých vieme vytvoriť webovú stránku. Program je dostupný v bezplatnej verzii (Microsoft Visual Studio Community), ale ja v platenej verzii s mnohými vylepšeniami (Microsoft Visual Studio Professional a Microsoft Visual Studio Enerprise). Microsoft Visual Studio je určené najmä pre operačné systémy Windows. V súčastnosti sa však dá nainštalovať aj verzia pre macOS a takisto verzia pre Linux. Najnovšia verzia Microsoft Visual Studia je Visual Studio 2019.

2 Popis implementácie

Interaktívna webová prezentácia audiovizuálnych diel je aplikácia, ktorá umožňuje nahrať videá na webovú stránku vo formáte mp4 a k nim dáta ktoré boli spracované klasifikátorom založenom na neurónových sieťach (deep learning, umelá inteligencia) v podobe súborov s príponou *.json*. Videá sa dajú nahrať z lokálneho súboru počítača a dajú sa k nim pridať parametre ako je napríklad, názov videa, meno autora a mnohé iné. Tieto videá je následne možné pomocou webovej aplikácie prehrať. Webová aplikácia má v sebe zabudovanú aj možnosť vymazania videí, ktoré sa po kliknutí na tlačidlo *vymazat* vymažú z databázy, a tak sa na stránke už nezobrazujú. Taktiež tu existuje možnosť úpravy videí, pomocou tlačidla *upravit*. Aplikácia má taktiež možnosť vyhľadávať videá podľa názvu videa. Webová stránka aplikácie je plne responzívna, čo znamená, že je určená pre rôzne druhy zariadení a teda aj pre rôzne druhy displejov. Nie je teda problém zobraziť si stránku na notebooku, ale aj mobile, alebo tablete.

Hlavným významom webovej aplikácie je, že si užívateľ môže triediť obsah, ktorý je klasifikovaný pomocou neurónových sietí. Tieto klasifikácie sa vo webovej aplikácii nazývajú tagy a majú danú klasifikáciu, ako je napríklad *body*, *face* ale aj mnohé iné. Webová aplikácia obsahuje dvadsať takýchto tagov. Tieto tagy sa takisto zobrazujú pri prehrávaní videí.

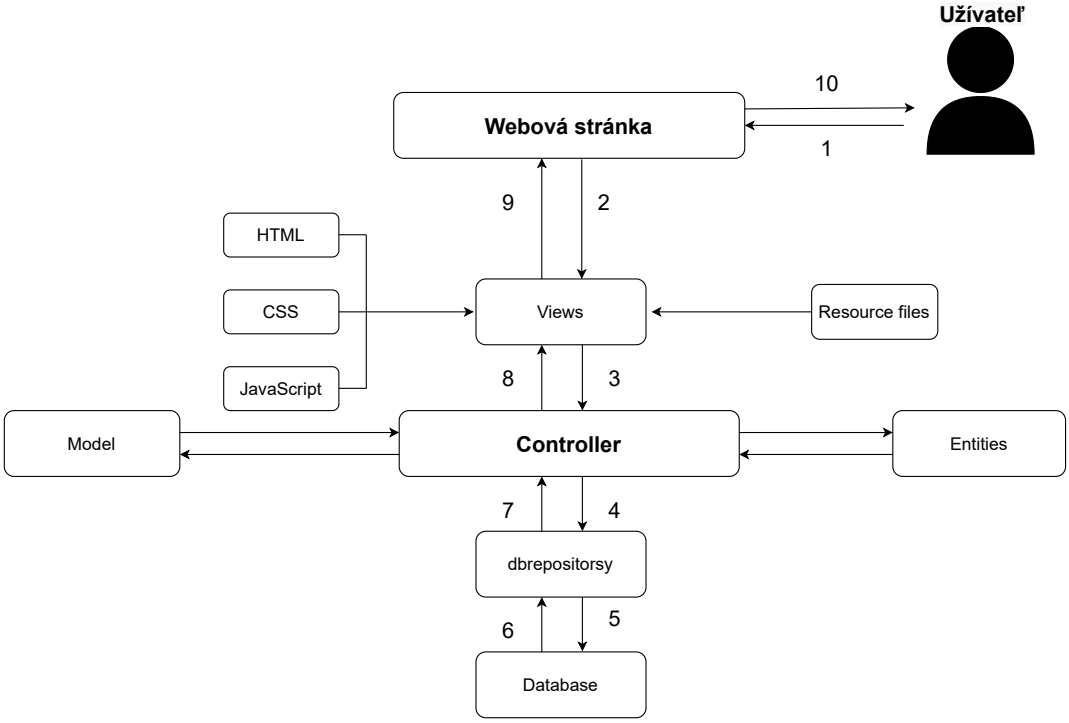
2.1 Štruktúra aplikácie

Aplikácia je rozdelená na Front-end a Back-end, teda na užívateľskú a serverovú stranu. Je postavená na modely MVC. Na vývoj aplikácie je použitá platforma.NET. Celá aplikácia je písaná pomocou objektovo orientovaného programovania, čo znamená, že metódy, ktoré majú niečo spoločné sú v jednej triede, ktorá ich združuje. Toto všetko zabezpečuje prehľadnosť aplikácie a ľahšie hľadanie prípadných chýb. Front-end sa stará o vizuálnu časť aplikácie. Používajú sa tu súbory s príponou .cshtml takzvané views. V týchto súboroch je pomocou jazykov CSS, HTML a JavaScript určená finálna podoba webovej stránky.

Back-end sa stará o serverovú časť aplikácie, a teda časť, ktorá je užívateľovi skrytá a má k nej prístup iba developer. Tu sa nachádzajú všetky súbory ktoré zabezpečujú komunikáciu medzi vizuálnou časťou aplikácie a jej back-end prostredím. V tejto časti sa spracovávajú všetky požiadavky ktoré užívateľ do aplikácie zadal a taktiež sa tu zrodí odpoveď na danú požiadavku a táto odpoveď je zaslaná späť užívateľovi, ktorá sa zobrazí prostredníctvom daného view.

Každá aplikácia má svoju vlastnú štruktúru. To že sa napríklad dve aplikácie zakladajú na MVC neznamená, že majú úplne rovnakú štruktúru. Takisto aj

aplikácia vytvorená v tejto práci má svoju špecifickú štruktúru. To akú štruktúru má aplikácia, ktorá bola v tejto práci vytvorená popisuje nasledujúci obrázok 2.1. Je tu znázornený celý proces od počiatku, keď užívateľ zadá nejakú požiadavku, až po jej spracovanie a následné vrátenie výsledku užívateľovi.



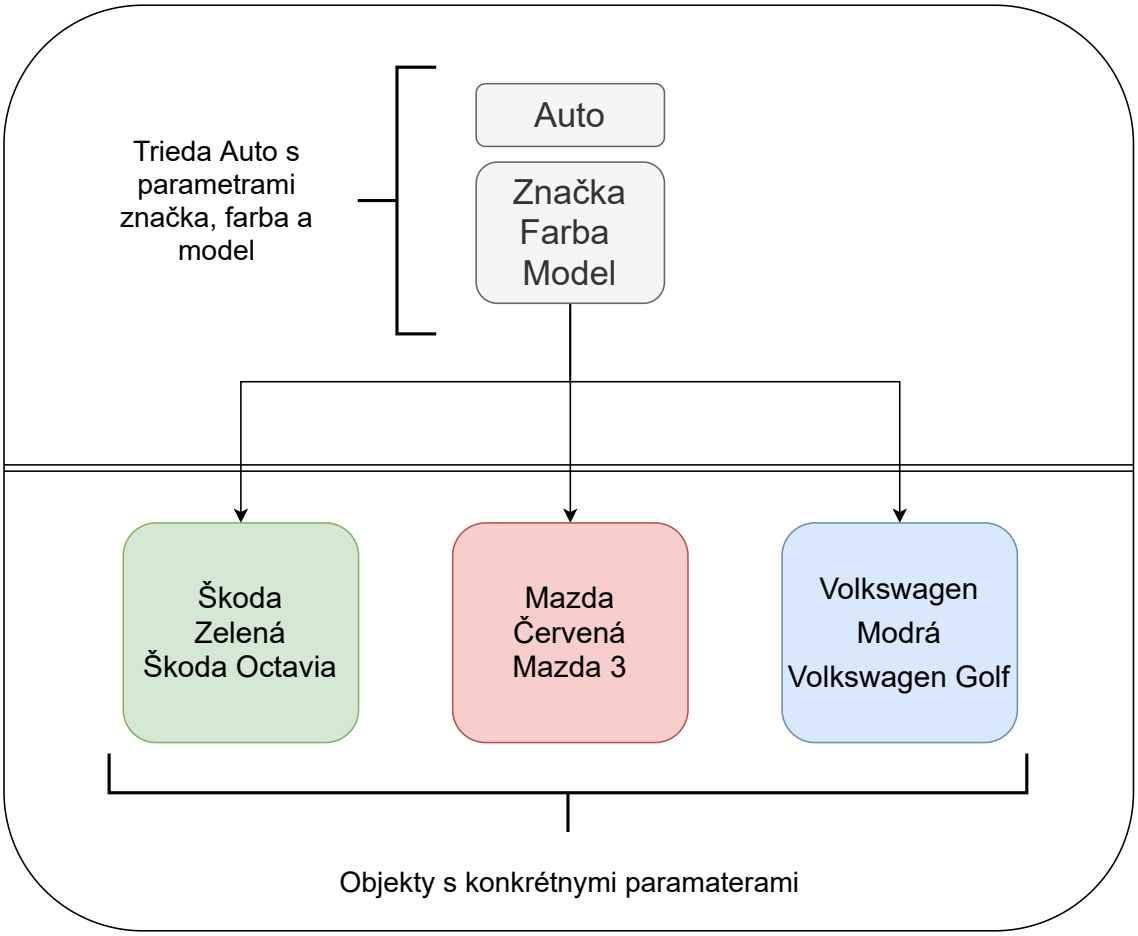
Obr. 2.1: Štruktúra aplikácie.

1. Užívateľ zadá na webovej stránke požiadavku.
2. Táto požiadavka sa dostane do views, ktorý umožňuje komunikáciu medzi webovou stránkou(užívateľom) a controllerom(aplikáciou).
3. V controllery je zadefinované čo sa s danými požiadavkami bude robiť. Controller teda vyhodnotí požiadavku a na základe potreby zavolá danú metódu. Controller v prípade potreby pracuje s *Entities* a *Model*.
4. Metóda ktorá sa volá v controllery sa nachádza v *DBRepository*, kde metóda pracuje s predanou hodnotou z controllera. V tejto metóde sa nachádza dotaz na databázu.
5. Zavolá sa dotaz nad databázou a to či už uloženie dát do nej alebo vytiahnutie dát z nej.
6. Z databázy sa vrátia dáta nad ktorými bol vykonaný dotaz.
7. Metóda v *DBRepository* vráti hodnotu so spracovanými dátami z databázy do controllera.
8. Dáta z controllera sa predajú do potrebného view.

9. View určuje pomocou CSS, HTML a JavaScript ako sa budú dáta zobrazovať. Taktiež view čerpá dáta z Resource files kde sa nachádza český alebo anglický ekvivalent textových dát a následne sa všetko odošle na webovú stránku kde sa všetko zobrazí.
10. Na konci tohto procesu je odpoveď na dotaz ktorý užívateľ na začiatku procesu zadal v podobe zobrazených dát na webovej stránke.

2.2 Objekty

Objekt je inštancia triedy. To v preklade znamená, že trieda, napríklad auto, môže mať vlastnosti, ako je značka, farba a model. Zatiaľ čo trieda je všeobecná, a tak má iba všeobecné vlastnosti, objekt už presne tieto vlastnosti určuje, napríklad do vlastnosti značka priradí Škoda, do farby priradí červená a do modelu priradí Škoda Octavia, viď. obr. 2.2 . Toto je základný princíp objektovo orientovaného programovania.



Obr. 2.2: Objekt.

Vo webovej aplikácii interaktívna webová prezentácia audiovizuálnych diel sa nachádza mnoho súborov a v nich mnoho tried. Tieto triedy sú častokrát medzi sebou poprepájané, čo umožňuje ich komunikáciu medzi nimi. Napríklad trieda *VideoDbContext.cs* sa stará o pripojenie celej aplikácie k databáze, trieda *DBVideosRepository.cs* sa stará o operácie s databázou, ako je napríklad pridanie nového videa, odstránenie videa, prehľadávanie databázy. To že dané metódy sú takto podelené v triedach zabezpečuje prehľadnosť aplikácie a tým zjednodušuje akúkoľvek prácu. Keď sa aplikácia potrebuje pripojiť k inej databáze, s rovnakými tabuľkami, stačí sa pozrieť do triedy *VideoDbContext.cs* a jednoducho metódu zmeniť. Tým že je aplikácia písaná pomocou objektového programovania, je schopná na túto zmenu reagovať bez akejkoľvek ďalšej úpravy. Toto všetko uľahčuje prácu vývojárom, ktorí vedia ľahko upravovať aplikáciu, poprípade ľahko nájsť chyby a opraviť ich.

2.3 Model-View-Controller

Celá aplikácia je založená na MVC. To opäť prispieva k prehľadnosti aplikácie. V aplikácii sa nachádzajú súbory s rovnakým názvom:

- Models,
- Views,
- Controllers.

2.3.1 Models

V priečinku *Models* sa nachádzajú triedy ako je napríklad *UploadModel.cs*, alebo . Tieto triedy definujú vlastnosti údajov, ktoré sa do databázy budú ukladať a taktiež pomocou nej vieme dostať údaje z databázy a poslať ich do controllera. Tieto vlastnosti sa nastavujú pomocou kľúčových slov *get* a *set*. V každej triede sa pracuje s inými propoertami a to hlavne preto, pri rôznych úkonoch potrebujeme pracovať s inými dátami. Napríklad trieda *UploadModel.cs* umožňuje združovanie viacero tabuliek z databázy do jednej premennej. V tejto triede existuje jeden bezparametrový konštruktor a mnohé parametrové konštruktory. V týchto konštruktoroch aplikácia určuje ktoré údaje z ktorých tabuliek sa majú spracovať a ktoré údaje sa do tabuliek budú zapisovať. To umožňuje kombináciu dát z viacerých tabuliek a uložiť tieto dáta do listu alebo premennej, s ktorou sa ďalej môže pracovať. Taktiež tu existuje metóda *CreateVideo*, pomocou ktorej sa vytvorí nová inštancia triedy Video a zapisujú sa do nej údaje, ktoré sú priradené k daným stĺpčekom v danej tabuľke Video. To umožní, že keď aplikácia potrebuje naplniť triedu vide, zavolá sa iba metóda CreateVideo, a nemusí sa tak obsah celej tejto metódy vypisovať do controllera.

2.3.2 Views

V priečinku *Views* sa nachádza všetok HTML kód stránky. Je tu zadefinované čo sa kde má užívateľovi zobraziť. Mnoho týchto views čerpá z preiečinku *wwwroot/css/site.css* kde je uložený všetok CSS kód aplikácie. Pomocou tohto kódu sa určuje grafické prostredie stránky. Taktiež je v súbore *site.css* zadefinovaná už daná responzívnosť webovej aplikácie. V priečinku *Views* sa nachádzajú ďalšie priečinky a to:

- *About*,
- *Credits*,
- *Home*,
- *Login*,
- *MachineVision*,
- *Shared*,
- *Upload*,
- *Video*.

Tieto priečinky sú zväčša rozdelené podľa položiek v menu a tým zabezpečujú prehľadnosť aplikácie. V priečinku *Shared* sa nachádzajú views, ktoré majú rovnaký názov ako popísané súbory vyššie. Nachádza sa v nich iba kód ktorý je pre danú skupinu views rovnaký ako je napríklad menu stránky. V ostatných priečinkoch sa nachádzajú view ktoré patria buď pod danú súčasť menu stránky, ako je napríklad *About*, *Credits* alebo *Home* alebo sa týkajú danej funkcie ako je napríklad Upload alebo Video.

Celá aplikácia tak čerpá všetok svoj vizuálny vzhľad z priečinku *Views*. Tieto views sú navzájom prepojené čo zabezpečuje prehľadnosť kódu. Na základe spojenia zo súborom site.css je zabezpečená finálna vizuálna podoba webovej stránky, teda Front-End programu.

2.3.3 Controllers

Zložka *Controllers* obsahuje všetky *controllers* triedy aplikácie. Tieto triedy sa starajú o Back-end aplikácie. Určuje sa tu logika aplikácie, dynamickosť aplikácie, čo sa pri ktorej akcii má vykonávať. Zabezpečujú pridanie, odstránenie a vyhľadávanie súboru. Hlavnou úlohou je zabezpečiť dynamickú časť aplikácie, ktorá je veľmi dôležitá na komunikáciu medzi uzívateľom a serverom.

Controller spracováva požiadavky ktoré užívateľ zadá do aplikácie a následne ich pomocou rôznych metód z *DBRepozitory* spracuje, čoho výsledkom je odpoveď na danú požiadavku užívateľa. Controller je srdcom Back-end časti aplikácie. Stará sa o väčšinu procesov ktoré sa v aplikácii vyskytujú. Je to základná časť spojenia medzi view (to čo vidí užívateľ) a zbytkom aplikácie.

Vo webovej aplikácii sa všetky controllery nachádzajú v priečinku *Controllers* a to:

- *AboutController*,
- *AccountController*,
- *CultureController*,
- *HomeController*,
- *LoginController*,
- *MachiveVisionController*,
- *UplaodController*,
- *VideoController*.

Tieto controllery sú nazvané výstižnými názvami a to zabezpečuje veľkú prehľadnosť webovej aplikácie.

V controlleroch ako je *AboutController*, *AccountController*,*CreditsController*, *HomeController* a *MachineVisionController* sa nachádzajú iba metódy ktoré zabezpečujú zobrazovanie daných view na daných URL adresách stránky. Controllery ako je *CultureController*, *LoginController*, *UploadController* a *VideoController* však už obsahujú aj metódy pomocou ktorých sa predávajú dáta z databázy alebo mení jazyk stránky.

CultureController sa stará práve o to aký jazyk sa nastaví do cookie súborov webovej stránky, na základe ktorého budú do views stránky predané dané textové dáta z Resource files.

LoginController sa stará práve o prihlásenie užívateľa do webovej aplikácie. Po prihlásení do aplikácie sa užívateľovi nastavia práva admina a tak môže určité dáta ako napríklad videa v aplikácii upravovať, mazať a taktiež môže tieto dáta do aplikácie nahrať.

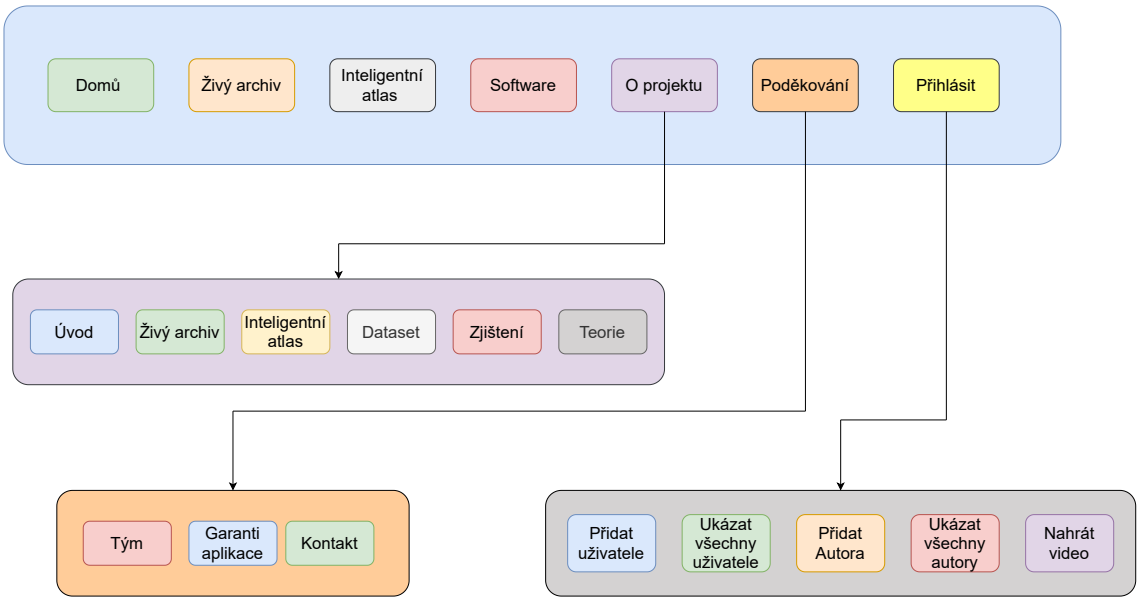
O nahratie videí a súborov typu JSON sa stará práve *UploadController*. Ten umožňuje užívateľovi s právami admina, teda prihlásenému užívateľovi, nahrávať pomocou formulára dáta do aplikácie.

VideoController sa stará o všetky videá a k nim priradené súbory JSON a dáta ako je autor, názov videa, alebo popis videa. *VideoController* zabezpečuje zobrazenie videí užívateľovi a daných dát, ktoré sú k týmto videám priradené. Taktiež umožňuje vyhľadávanie videí podľa názvu alebo triedenie videí podľa tagov.

2.4 Webová stránka

Základom komunikácie užívateľa s aplikáciou je webová stránka. Nachádza sa tu všetko k čomu by mal mať užívateľ prístup, 2.4. O celú funkčnosť stránky sa starajú triedy *controllers*, ktoré posielajú tie dáta na stránku, ktoré sa majú zobrazovať a umožňujú tak užívateľovi komunikovať so serverom. V každej časti stránky sa

nachádza niečo iné. Niekde sú zobrazené napríklad videá, inde zasa informácie o projekte.



Obr. 2.3: Diagram štruktúry navigačného menu.

Grafický návrh a návrh funkcionality frontend časti webu bol vytvorený s využitím výstupov riešenia projektu TAČR TL02000270 v spolupráci s Ing. MDes.Dušanem Barokem.

V pravom hornom rohu sa nachádza *combobox*, ktorý umožňuje užívateľovi prepínať stránku medzi anglickou a českou verziou 2.4. Taktiež sa tu nachádza políčko ktoré umožňuje užívateľovi vyhľadávať videá podľa ich názvu.

Sekcia *Domů* je základná časť webovej stránky. Užívateľ tu môže nájsť stručný a výstižný popis o čom je daný projekt. Tento popis uvádza užívateľa do obrazu, čo môže na webovej stránke nájsť a čo môže od nej očakávať.

Živý archiv je hlavná časť webovej stránky. Užívateľ tu nájde všetky videá, ktoré webová aplikácia obsahuje. Taktiež sa tu nachádzajú tagy, podľa ktorých si užívateľ môže zvoliť parametre videí ktoré chce aby sa mu zobrazili. Pokiaľ užívateľ chce v sekcii Živý archiv nájsť konkrétne video, môže tak učiniť pomocou zadania daného textu do políčka a stlačením tlačidla Hledat.

V sekcii *Inteligentní* atlas sa nachádza obrázok ktorý pozostáva z jednotlivých snímkov daných videí.

Obsah sekcie *Software* obsahuje odkaz na službu github, kde je možné si stiahnuť software pre obraz a software pre audio. Táto sekcia taktiež ukazuje čiastočné výčitky z tohto softvéru.

V sekcii *Prihlášení* sa nachádza prihlasovací formulár pomocou ktorého sa užívateľ môže do aplikácie prihlásiť a tým sa rozšíria možnosti tvoriť požiadavky na

aplikáciu.

Dalšie sekcie ako je O projektu a *Poděkování* obsahujú informácie o webovej aplikácii a taktiež informácie o kalsifikátore založenom na neurónových sieťach (deep learning a umelá inteligencia).

Obr. 2.4: Domovská stránka.

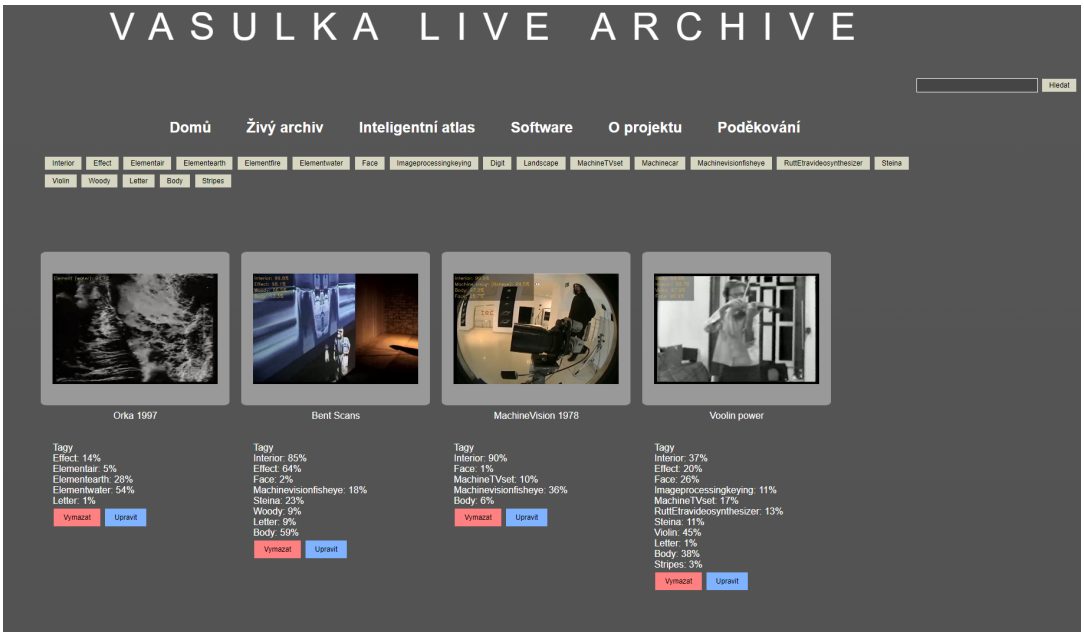
2.4.1 Video

Základom webovej aplikácie je prezentácia video obsahu, ktorého obsah je klasifikovaný pomocou neurónových sietí. Toto video sa dá zobraziť, prehrať, nahrať, upraviť, ale aj vymazať. Počas prehrávania sa vedľa videa zobrazujú klasifikácie ktoré sa na aktuálnom snímku videa nachádzajú. Bežný užívateľ bez prihlásenia má prístup iba k základným funkciám a to zobraziť videá ktoré obsahuje webová aplikácia a možnosť prehrať si ich. S ďalšími popísanými funkciami má možnosť pracovať iba užívateľ, ktorý sa do webovej aplikácie prihlási. Videá boli čerpané z literatúry [8, 9, 10, 11].

2.4.2 Zobrazenie videí

Neodmysliteľnou súčasťou aplikácie je prehrávanie videí. Túto časť užívateľ nájde na stránke v menu, kde sa nachádza položka Živý archiv 2.5. Po kliknutí sa užívateľovi zobrazí stránka kde sa nachádzajú všetky videá, ktoré sú v databáze uložené. Pod menu sa virtuálny filter vo forme tlačidiel, ktorý umožňuje užívateľovi triediť videá podľa klasifikácie ktorá bola vytvorená neurónovou sieťou. Každé tlačidlo obsahuje názov jednej klasifikácie (tagu) a po kliknutí na dane tlačidlo sa zobrazia videá ktoré obsahujú daný tag. Pod videom sa nachádza názov daného videa a taktiež rok kedy toto video vzniklo. Pod týmito informáciami sa nachádzajú informácie o tagoch, ktoré sa nachádzajú vo videu, a ich percentuálnom zastúpení.

O to aké videá sú predané stránke určuje *VideoController.cs*e a v ňom metóda DbView, v ktorej sa volá metóda *GetAllVideos*, ktorá sa nachádza v zložke *Infrastructure/video*. V tejto zložke sa nachádza interface *IVideosRepository*. Tu sa nachádza metóda *GetAllVideos*. Metóda *GetAllVideos* zabezpečuje, že aplikácia prejde



Obr. 2.5: Živý archiv.

celú databázu a riadok po riadku ju uloží do listu. V DbView sa volajú ešte dve metódy a to metóda *GetAllAuthors*, ktorá podobne ako metóda *GetAllVideos* uloží všetkých autorov z databázy do listu. Pomocou funkcie foreach sa následne prejde celý list videí a k daným videám sa priradia pomocou metódy *GetTagWeightsForVideo* tagy a ich percentuálne zastúpenie vo videu. Všetky tieto údaje sa uložia do listu *UploadModel*. Tento list sa následne pošle do view *DbView.cshtml* (Živý archiv). View podľa predom danej šablóny CSS pridá videá na patričné miesto stránky. Pod týmito videami sa nachádzajú základné údaje ktoré sú asociované k videu, ako je názov videa a rok. Videá sa na stránke zobrazujú bez controlls, čo znamená, že video sa nedá prehrať, posunúť ani nijako s ním pracovať. Funguje preto iba ako odkaz, ktorý odkáže na ďalšiu stránku *play.cshtml*, kde sa už video zobrazí so všetkými parametrami a už aj s možnosťou prehrať video.

Videá ktoré sú nahraté vo webovej aplikácii majú často na prvom snímku v čase nula sekúnd iba čierny snímok, alebo snímok kde sú badateľné iba nepatrné rozdiely ako je napríklad nadpis, čo je však pri malom rozlíšení na stránke takmer nerozoznateľný. Toto by vytváralo prostredie stránky, kde by takmer všetky videá mali rovnaké alebo veľmi podobné prvé snímky a tým by bola stránka veľmi neprehľadná. Vo webovej aplikácii je tento problém riešený pomocou jazyka JavaScript a to konkrétne funkciou *setVideoStartTime*, ktorá zabezpečí, že všetky videá ktoré sa načítajú na stránke sa nastavia tak, aby sa prvý snímok zobrazil na čase dvadsať sekúnd a to zabezpečí jasnú rozdielnosť medzi rôznymi videami. Keďže každé video má nastavenú triedu video-time JavaScript pomocou funkcie *setVideoStartTime* dokáže zistiť

koľko videí ja na danej stránke načítaných. Daný počet si uloží do premennej. Táto premenná určuje dĺžku cyklu for na základe ktorého sa danému počtu videí nastaví počiatočný čas na dvadsať sekúnd.

Zdroj odkiaľ má aplikácia videá čerpať je daný na súbor *upload*, ktorý je súčasťou webovej aplikácie a do ktorého sa pri nahrávaní nahrajú videá. Následne v tabuľke *videos* v stĺpci *VideoName* je uložený názov súboru. Aplikácia teda vie, že sa má pozrieť do zložky *upload* a odtiaľ vybrať potrebný súbor. Aplikácia týmto spôsobom zobrazí všetky videá spolu so všetkými parametrami, ktoré sú uložené v liste *UploadModel* a sú predané do view *DbView* kde daný view určí čo všetko sa z tohto listu na stránke zobrazí.

2.4.3 Prehrávanie videí

Po vybratí daného videa v sekcii *Živý archiv* sa dané video otvorí na novom odkaze */Play* viď. obrázok 2.6. V tejto časti sa video už dá prehrať pomocou vlastností HTML prehrávača nazvaných atribútom controlls. To umožňuje užívateľovi posúvať sa na časovej osi, nastaviť hlasitosť a taktiež zobraziť video na celú. Prehrávanie videí zabezpečuje samotná view *Play.cshtml*. To aké video sa zobrazí na danej stránke je zabezpečené pomocou metódy *Play*, ktorá sa nachádza vo *Videocontroller*. V tejto metóde sa volá metóda *getVideoById*, ktorá sa nachádza v interface IVideosRepository. Táto metóda porovná po kliknutí na dané video v sekcii Živý archiv jeho jedinečné Id a priradí mu dáta z databázy s daným *Id*. Toto *Id* je taktiež uložené v url adrese. Na stránke *Play* sa teda zobrazia dáta ku ktorým je pridelené toto jedinečné *Id*.

To, ktoré údaje sa zobrazia na stránke pod videom už určuje view zvaný Play.cshtml, do ktorého sú poslané všetky údaje priradené k videu. View Play.cshtml pomocou podmienok kontroluje, ktoré dáta priradené k videu, ako je napríklad autor, názov videa, alebo komentár, sú nenulové a tie zobrazí na stránke. Ak sú dáta nulové, to znamená, že daný stĺpec v tabuľke v danom riadku neobsahuje žiadne dáta, na stránke sa tieto dáta nezobrazia. V databáze sa však nachádzajú dvojjazyčné dáta ako je napríklad *VideoCommentCs* a *VideoCommentEn*. Tieto dáta sa musia zobrazovať na základe zvoleného jazyka ktorého princíp je popísaný nižšie. Webová aplikácia rieši tento problém tak, že v zložke Resources sa nachádza resource file *Play.cs.resx*, kde sa nachádza riadok s *Name „Language“* a *Value „cs“*. V obdobnom resource file Play.en.resx sa nachádza riadok s Name „Language“ a Value „en“. View Play.cshtml následne pomocou podmienky *if* pri zobrazovaní kontroluje práve túto hodnotu (value) , či je nastavená na *cs* alebo *en*. Na základe toho potom zobrazí iba dáta ktoré majú byť zobrazené, v tomto prípade *VideoCommentCs* alebo *VideoCommentEn*.

Pod videom sa taktiež nachádza časť, kde sú kapitoly k danému videu. Tieto kapitoly umožňujú užívateľovi lepší prehľad daných častí videa, bez toho aby sa musel preklikávať videom. Ak užívateľ klikne na vybratú kapitolu, video sa presunie na vybratý snímok ktorý sa nachádza na danom čase. Funkcionalita kapitol je zabezpečená pomocou jazyka JavaScript. Tu sa nachádza funkcia *setCurTime*. Táto funkcia sa stará o to, že pri načítaní videa na danej stránke, sa načíta dĺžka videa a tá sa rozdelí na desať častí teda desať kapitol. V tejto funkcii sa nachádza aj cyklus *for*, ktorý zabezpečuje aby sa k danej kapitole priradil aj správny snímok z videa.

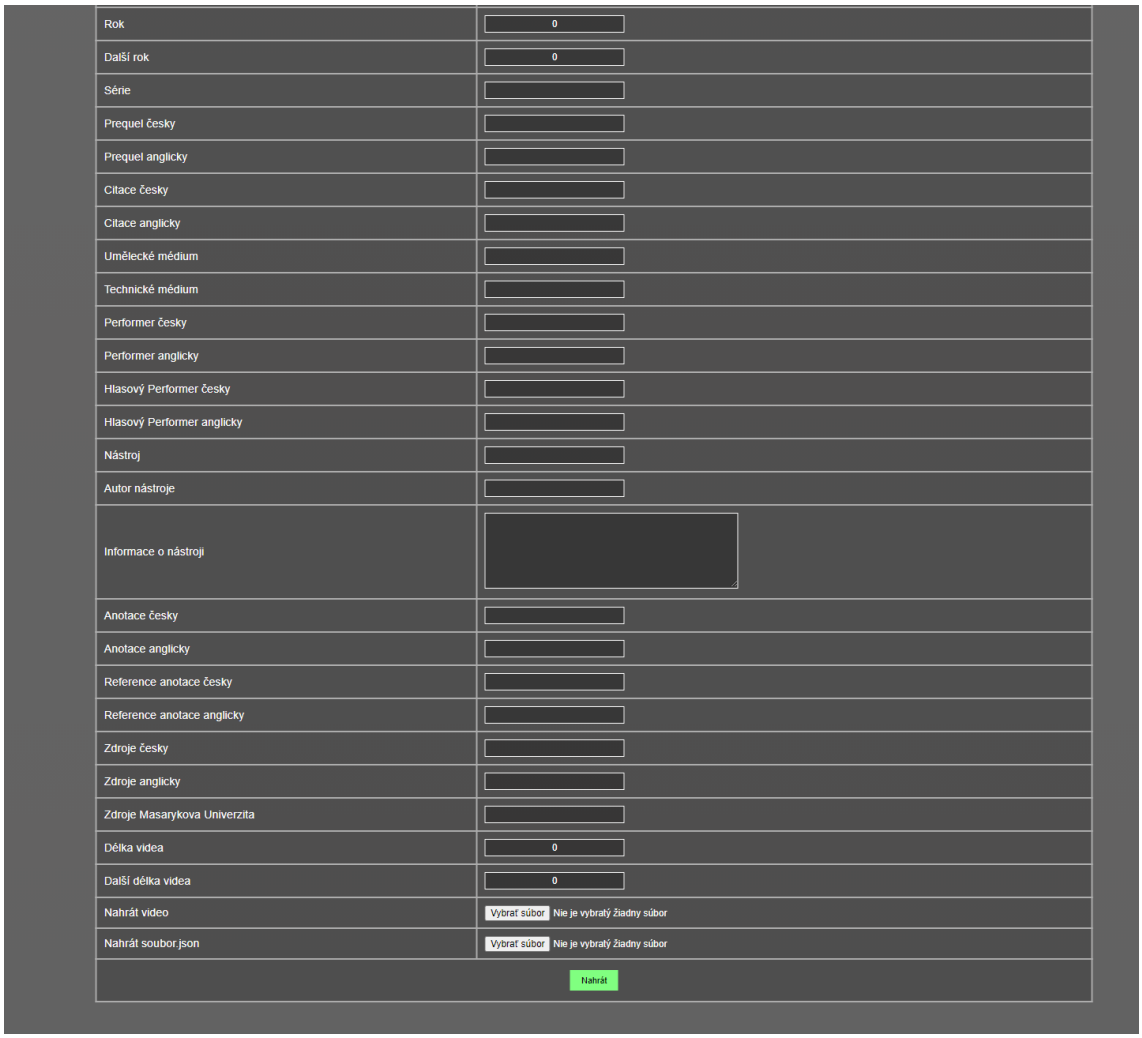
Obr. 2.6: Stránka na prehrávanie videí.

2.4.4 Nahrávanie videí

Po kliknutí v menu na položku *Přihlášení* a následnom prihlásení, sa zobrazí menu, kde sa nachádza položka *Nahrát video* . Po kliknutí na danú položku sa zobrazí stránka *upload.cshtml*, kde sa nachádza formulár, pomocou ktorého je prihlásenému uživateľovi umožnené nahrať dáta do databázy. O celú štruktúru procesu sa stará *UploadController.cs*. 2.7.

Ďalej sa tu nachádza tlačidlo, ktoré umožňuje vybrať súbor, ktorý sa má do databázy nahrať a následne je tu tlačidlo upload, ktoré všetko potvrdí a video sa nahrá do databázy.

O nahranie do databázy sa však konkrétne nestará *UploadController.cs*. Tento controller následne predá parameter triede *IVideoRepository.cs*, ktorá pomocou metódy *SaveVideo* zadané parametre spracuje a uloží ich do databázy. To je celý zmysel



Obr. 2.7: Stŕanka na nahrávanie videa.

aplikácie. Controller by sa mal starať o uloženie do databázy, ale vôbec by nemal vedieť akým spôsobom prebieha proces ukladania. Preto je v aplikácií vytvorená zložka *Infrastructure*, kde je zložka *Database* a tu sú uložené všetky metódy na prácu s databázou.

- *Název videa*,
- *Autor*,
- *Spoluautor*,
- *Komentář videa česky*,
- *Komentář videa anglicky*,
- *Komentář autora česky*,
- *Komentář autora anglicky*,
- *Rok*,
- *Další rok*,

- *Série*,
- *Perquel česky*,
- *Perquel anglicky*,
- *Citace česky*,
- *Citace anglicky*,
- *Umělecké medium*,
- *Technické médium*,
- *Performer česky*,
- *Performer anglicky*,
- *Hlasový performer česky*,
- *Hlasový performer anglicky*,
- *Nástroj*,
- *Autor nástroje*,
- *Informace o nástroji*,
- *Anotace česky*,
- *Anotace anglicky*,
- *Reference anotace česky*,
- *Reference anotace anglicky*,
- *Zdroje česky*,
- *Zdroje anglicky*,
- *Zdroje masarykova univerzita*,
- *Délka videa*,
- *Další délka videa*.

Ďalej sa tu nachádza tlačidlo, ktoré umožňuje vybrať súbor, ktorý sa má do webovej aplikácie nahrať. Pri autorovi a spoluautorovi sa nachádza takzvaný combobox. Keďže existuje zvlášť tabuľka na autorov, ktorej význam je popísaný v odstavci, do danej tabuľky *videos*, kde sa nahrávajú všetky dáta z formulára, sa nahrá iba číslo (*AuthId*). Aby uživateľ nemusel poznať naspamäť všetky tieto *Id*, je tu práve *combobox*. Ten vo formulári zobrazí konkrétnych autorov, ktorí sú uložený v tabuľke *authors*, a po vybraní daného autora zapíše číslo do stĺpca *AuthId* v tabuľke *videos*. Na konci formulára sa nachádza tlačidlo *Nahrát*, pomocou ktorého sa všetky dáta vyplnené v danom formulári predajú do *UploadController.cs*.

O nahranie do databázy sa však konkrétne nestará *UploadController.cs*. Tento controller iba predá parameter do interface *IVideoRepository.cs*, ktorý pomocou metódy *SaveVideo* zadané parametre spracuje a uloží ich do databázy. To je celý zmysel aplikácie. Controller by sa mal starať o uloženie do databázy, ale vôbec by nemal vedieť akým spôsobom prebieha proces ukladania. Preto je v aplikácií vytvorená zložka Infrastructure, kde je zložka *Database* a tu sú uložené všetky metódy na prácu s databázou.

2.4.5 Vyhľadávanie videí

V pravej časti stránky sa nachádza textové pole a vedľa neho tlačidlo Hledat, ktoré slúži na vyhľadávanie videí v databáze home. Vyhľadávač slúži na vyhľadávanie podľa názvu videa. Po zadaní písmena, skupiny písmen, alebo slov a následného kliknutia na tlačidlo *Hledat*, sa zobrazia videá s danou zhodou.

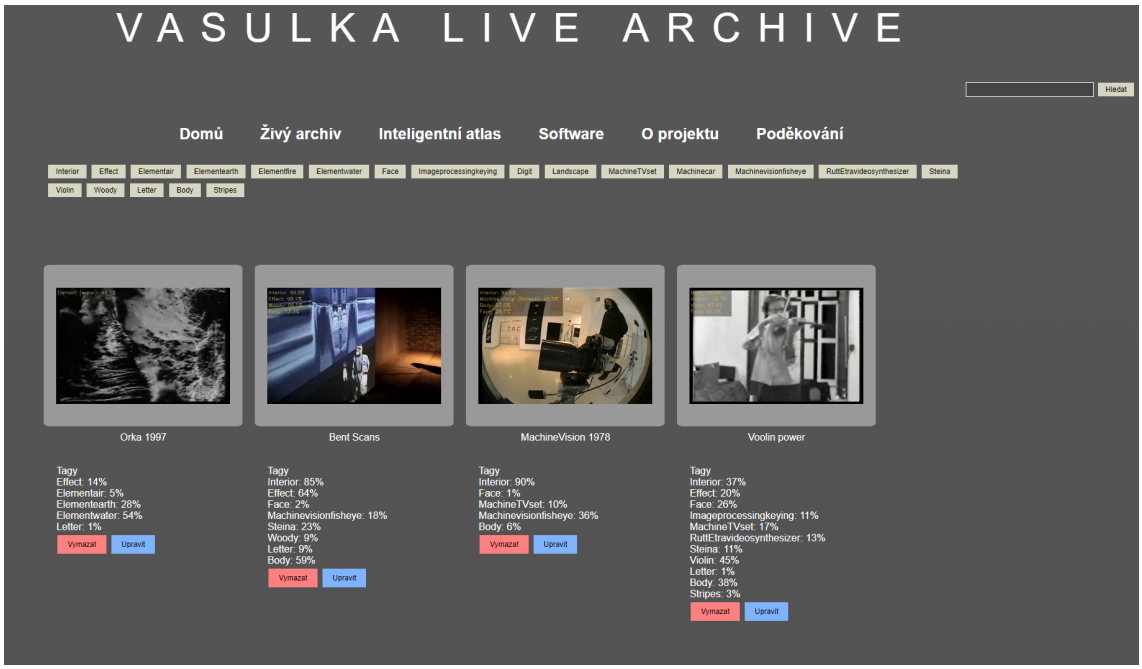
Princíp je taký, že pri kliknutí na tlačidlo *Hledat* sa parameter z textboxu predá danému controlleru a to konkrétne *VideoController.cs*, kde sa spracuje a následne sa predá do triedy *DBVideoRepository.cs*. Tu sa nachádza metóda *SearchVideo*, ktorá obsahuje jeden parameter (*string fileName*). Parameter ktorý controller poslal sa porovná s údajmi v databáze a to konkrétne s údajmi *Name* a *Author*. Pri zhode aspoň s jedným s parametrov, sa video uloží do listu. Keď je celá databáza skontrolovaná a všetky potrebné údaje sú v liste, aplikácia sa opäť presunie do controlleru, kde následne tento list pošle stránke *SearchResults.cshtml* a tu sa podľa vopred danej CSS šablóny dáta zobrazia.

2.4.6 Úprava videí

Keď má užívateľ prihlasovacie údaje, tak webová aplikácia na základe týchto údajov umožňuje prihlásenie. vid odkaz na login. Na základe toho, že je užívateľ prihlásený, webová aplikácia zobrazuje viac vecí na stránke ako bežnému neprihlásenému užívateľovi, ktorý si môže stránku iba prezerať viď. obr. 2.8. Jednou z týchto vecí je aj možnosť upravovať videá. Ak je užívateľ prihlásený, stránka Živý archiv, pod videom zobrazí tlačidlo s názvom *Upravit*. Po kliknutí na toto tlačidlo sa užívateľovi zobrazí podobný formulár, aký je pri nahrávaní videa vid. Odkaz na obrázok upload s tým rozdielom, že sa tu nenachádzajú tlačidlá na nahratie videa a nahratie súboru JSON. Všetky dáta ktoré sú v danej dobe k videu pridané sa zobrazia v políčkach a tie ktoré nie sú momentálne v databáze vyplnené, tie sa zobrazia ako prázdne políčka. Tieto dáta môžu byť užívateľom rôzne upravované. Dané dáta as môžu premenovať, do políčok ktoré sú prázdne doplniť údaje, ktoré k videu ešte nie sú priradené, alebo odstrániť dané údaje ktoré nie sú už viac k videu potrebné. Keď sú všetky údaje zmenené podľa užívateľovej potreby, na spodku formulára sa nachádza tlačidlo Upravit. Po stlačení tohto tlačidla sa celý formulár odošle a dáta sa upravia.

Údaje s formulára sa odošlú do *VideoController.cs*. Tu sa nachádza metóda *Edit* a má ako vstupný parameter Id. V metóde *Edit* sa volá metóda *getVideoById* ktorá sa nachádza v interface *IVideorepository.cs*. Pomocou tejto metódy sa prejde celá databáza a vyberie sa video ktoré má rovnaké Id aké bolo predané do metódy.

Ďalšia metóda je metóda *HttpPost Edit* ktorá zabezpečuje prepísanie dát. V tejto metóde sa volá ďalšia metóda a to konkrétne metóda *Edit*. Táto metóda sa



Obr. 2.8: Úprava videí.

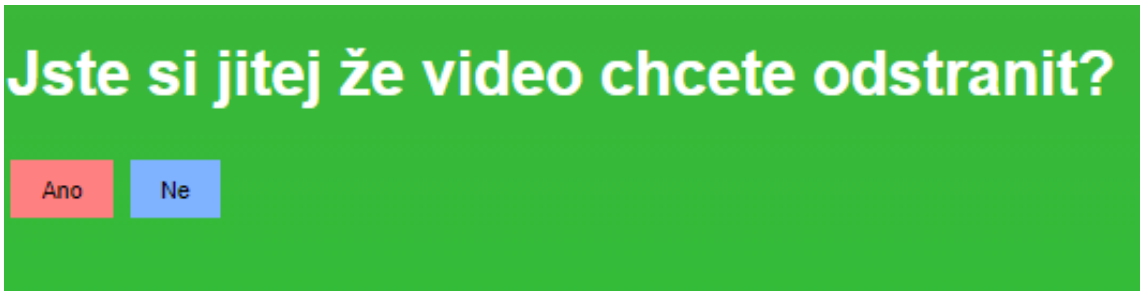
nachádza v interface *IVideorepository.cs*. Pomocou metódy *Edit* sa všetky pôvodné dáta z formulára prepíšu dátami ktoré boli predané pomocou formulára a nahrajú sa do databázy. Po celom procese webová aplikácia presmeruje užívateľa späť na stránku *živý archiv* kde webová aplikácia zobrazí už upravené údaje. Keďže vo webovej aplikácii môže byť mnoho videí a vyhľadanie konkrétneho videa na upravu by mohlo byť pre užívateľa problematicé, existuje možnosť, vyhľadania si daného videa podľa názvu, ako je popísané odkaz na vyhladavanie videi. Pri vyhľadaných videách sa tak isto zobrazí tlačidlo Upravit ako tomu je na stránke *živý archiv*.

2.4.7 Mazanie videí

V prípade ak je užívateľ prihlásený, na stránke *živý archiv* sa nachádza tlačidlo s názvom *Vymazat*. Toto tlačidlo ako už samotný názov vypovedá slúži na odstránenie videa z databázy. Po kliknutí na toto tlačidlo webová aplikácia zistí Id daného videa a predá ho pomocou url adresy. Následne webová aplikácia presmeruje daného užívateľa na stránku *deleteVideo*, kde je v url adrese ešte pridané id číslo daného videa viď. obr.2.9. Na tejto stránke sa zobrazí oznam kde sa webová aplikácia pýta užívateľa či dané video má byť naozaj odstránené a pod textom sa nachádzajú dve tlačidlá *Ano* a Ne. V prípade kliknutia na tlačidlo Ne, užívateľ je automaticky presmerovaný na stránku *živý archiv* a video sa nevymaže. Avšak pri kliknutí na tlačidlo *Ano* odošle sa požiadavka na odstránenie videa.

Získané Id sa odošle do *VideoController.cs*. Tu sa nachádza metóda *deleteVideo*.

V tejto metóde sa volá metóda *getVideoById* ktorá je popísaná Odkaz na úpravu videí Ďalej sa vo *VideoController.cs*. nachádza metóda *HttpPost deleteVideo*, ktorá zabezpečuje vymazanie dát z webovej aplikácie. V tejto metóde sa volá metóda RemoveVideo. Táto metóda sa nachádza v interface *IVideorepository.cs*. Pomocou metódy RemoveVideo sa vymažú všetky dáta ktoré spadajú k danému Id, ktoré je k videu priradené. Nakoniec webová aplikácia presmeruje užívateľa na stránku *živý archiv*.



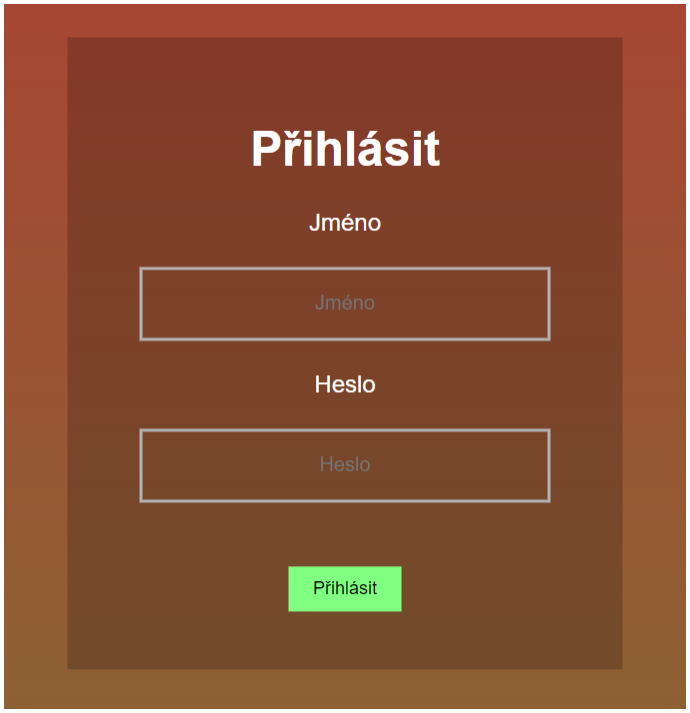
Obr. 2.9: Mazanie videí.

2.5 Prihlásenie

Celá aplikácia sa zakladá na tom aké dáta sa budú užívateľovi zobrazovať. To záleží či daný užívateľ má prihlasovacie údaje do webovej aplikácie alebo nie. Tu webová aplikácia odlišuje bežného užívateľa od prihláseného užívateľa užívateľa s príznakom Admin.

V pravom hornom rohu na stránke sa nachádza odkaz na *Přihlášení* vid odkaz na *Home*. Tento odkaz presmeruje užívateľa na prihlasovací formulár. Pomocou tohto formulára je danému užívateľovi umožnené prihlásiť sa do webovej aplikácie a tým vykonávať zmeny v aplikácii viď. obr.2.10. Prihlasovací formulár obsahuje dve políčka ktoré musia byť užívateľom vyplnené aby bolo danému užívateľovi umožnené sa prihlásiť. Políčka musia obsahovať správne údaje, inak webová aplikácia vypíše chybu prihlásenia. Následne sa na spodku formulára nachádza tlačidlo Přihlásit, pomocou ktorého sa daný formulár odošle. O celé prihlásenie sa stará *CultureController.cs*. V tomto controllery sa nachádza metóda Login v ktorej sa volá metóda UserCheck. Na základe zhody, buď v databáze alebo s pevným užívateľom v kóde, poprípade nezhody sa vráti do controlleru výsledok true alebo false. V controllery sa podľa výsledku vykoná ďalšia akcia. Ak je výsledok false, tak je užívateľ presmerovaný na stránku *Přihlášení* a vo formulári sa mu zobrazí správa o tom, že meno alebo heslo je nesprávne. Ak je však výsledok true, do parametru identity sa pridá hodnota ktorá je uložená v *userName*. Do cookie súborov stránky sa uloží záznam o

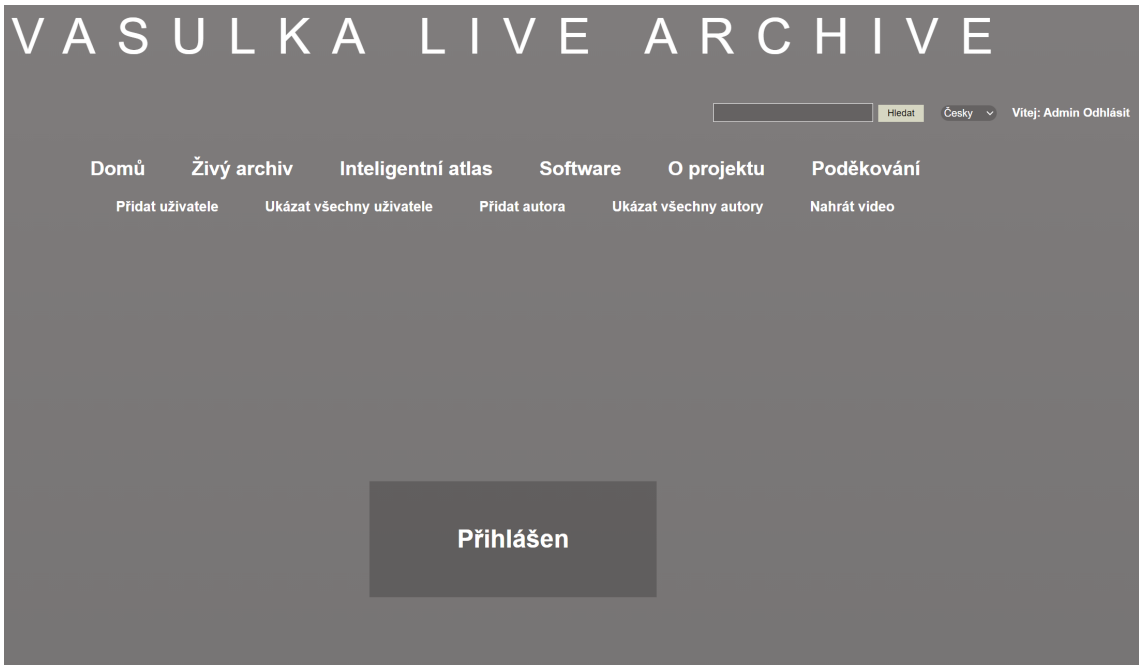
tom, že užívateľ je overený a teda prihlásený. Taktiež sa k tomuto užívateľovi pridá rola Admin. Keď je užívateľ prihlásený, tak sa na stránke *živý archiv*, sú zobrazené aj tlačidla na možnosť úprav daného videa a vymazanie daného videa. V prípade, že sa chce neprihlásený užívateľ dostať na stránku, kde má oprávnenie iba prihlásený užívateľ, tak ho aplikácia presmeruje na prihlasovací formulár. Ak je užívateľ prihlásený tak v pravej strane stránky sa zobrazí jeho meno. Po kliknutí na jeho meno ho aplikácia presmeruje na stránku login.cshtml kde sa zobrazí menu ktoré umožňuje veci ako nahratie dát do databázy, pridávanie autora, alebo pridávanie nového užívateľa.



Obr. 2.10: Prihlásenie užívateľa.

2.5.1 Pridanie užívateľa

V sekcii Přidat uživatele, sa nachádza formulár, ktorý umožňuje pridávanie užívateľa do databázy. Podobne ako v prihlasovaciom formulári aj tu sa nachádzajú dve políčka s názvom meno a heslo, kde si užívateľ určí tieto dva parametre pre nového užívateľa. Po potvrdení tlačidlom Přidat sa parametre predajú *CultureController.cs*, kde sa v metóde AddNewUser zavolá odkaz na metódu SaveUser, ktorá sa nachádza v infrastructure v interface IUsersRpository a je jej predaný parameter s hodnotou z formulára. Parameter *password* je kvôli zabezpečeniu hashovaný pomocou metódy HashSha256. Nový uživateľ je následne vložený do tabuľky users.



Obr. 2.11: Prihlásený užívateľ.

2.5.2 Ukázať všetkých užívateľov

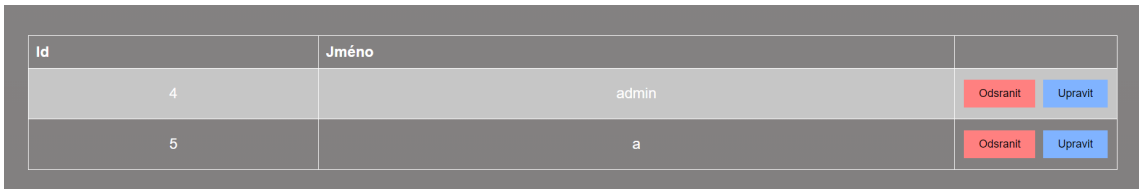
Na stránke login.cshtml sa v menu nachádza taktiež sekcia Ukázat všehny uživatele viď obr.2.11. Po kliknutí na tento odkaz sa užívateľovi zobrazí tabuľka, v ktorej sa nachádzajú všetci užívatelia ktorí sú v databáze viď. obr.2.12. Pri kliknutí na odkaz Ukázat všehny uživatele sa do *CultureController.cs* vyšle požiadavka na zobrazenie užívateľov z databázy. V *CultureController.cs* sa nachádza metóda *Allusers*. V nej sa volá metóda *GetAllusers*. Pomocou tejto metódy sa uložia všetky dáta z tabuľky do listu. Na stránke Ukázat všehny uživatele sa tak v tabuľke zobrazia údaje ktoré sa nachádzajú v danom uloženom liste.

2.5.3 Vymazanie užívateľa

Na stránke Ukázat všehny uživatele sa taktiež nachádza tlačidlo *Vymazat*, ktoré umožňuje užívateľovi vymazať daného užívateľa.V *CultureController.cs* sa nachádza metóda RemoveUser. V metóde RemoveUser sa volá metóda *deleteUser* ktorá na základe predaného Id vymaže užívateľa z databázy. Následne je pomocou *CultureController.cs* užívateľ presmerovaný späť na stránku Ukázat všehny uživatele

2.5.4 Úprava užívateľa

Na stránke Ukázat všehny uživatele sa vedľa tlačidla *Vymazat* nachádza tlačidlo *Upravit*. Toto tlačidlo umožňuje užívateľovi upravovať užívateľov v databáze. Po



Obr. 2.12: Všetci užívateľa.

stlačení tlačidla je užívateľ presmerovaný na upravovací formulár, ktorý je podobný ako prihlasovací formulár. Nachádza sa tu meno užívateľa a heslo užívateľa. Na spodku formulára sa nachádza tlačidlo *Upravit*.

Po stlačení tlačidla v tabuľeke sa v *CultureController.cs* zavolá metóda *Edit*. V tejto metóde volá metóda *getUserById* ktorej výsledkom je zvolený užívazteľ. Ďalej sa vo *CultureController.cs*. nachádza metóda *HttpPost Edit*. V tejto metóde sa volá metóda metódu EditExistingUser. Pomocou tejto metódy sa prepíšu všetky dáta užívateľa s daným Id podľa toho, čo užívateľ zadal do formulára pre úpravu užívateľa. Nakoniec je užívateľ webovou aplikáciou presmerovaný na stránku Ukázat všehny uživatele.

2.5.5 Pridanie autora

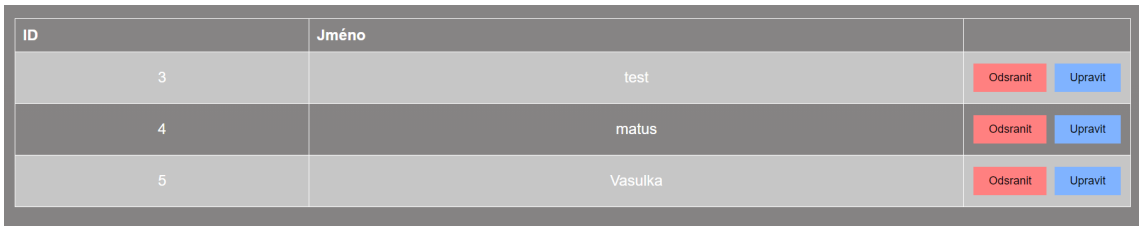
V sekcii Přidat autora, sa nachádza formulár ktorý umožňuje pridávanie užívateľa do databázy. Formulár obsahuje políčko pre meno autora. Na spodku formulára sa nachádza tlačidlo Přidat. Po potvrdení tlačidlom Přidat sa parametre predajú *CultureController.cs*, kde sa v metóde AddNewAuthor zavolá metóda SaveAuthor. Pomocou tejto metódy sa uloží do tabuľky *authors* nový autor, ktorému je pridelené jedinečné *AuthId*.

2.5.6 Ukázať všetkých autorov

Na stránke login.cshtml sa v menu nachádza taktiež sekcia Ukázat všehny autory. Po kliknutí na tento odkaz sa v *CultureController.cs* zavolá metóda *Allauthors*. V nej sa volá metóda *GetAllAuthors*. Pomocou tejto metódy sa uložia všetky dáta z tabuľky do listu viď. obr. 2.13.

2.5.7 Vymazanie autorov

Na stránke Ukázat všehny autory sa taktiež nachádza tlačidlo *Vymazat*, ktoré umožňuje užívateľovi vymazať daného autora.V *CultureController.cs* sa nachádza metóda RemoveAuthor. V metóde RemoveAuthor sa volá metóda *deleteAuthor*. Na základe daného Id ktoré bolo metóde predané, metóda vymaže záznam o danom autorovi z



Obr. 2.13: Všetci autori.

databázy. *CultureController.cs* Následne presmeruje užívateľa späť na stránku Ukázat všehny autory.

2.5.8 Úprava autora

Na stránke Ukázat všehny autory sa vedľa tlačidla *Vymazat* nachádza tlačidlo *Upravit*. Toto tlačidlo umožňuje užívateľovi upravovať autorov v databáze. Po stlačení tlačidla je užívateľ presmerovaný na upravovací formulár, ktorý je podobný ako formulár pre pridanie autora. Nachádza sa tu meno autora. Na spodku formulára sa nachádza tlačidlo *Upravit*. Po stlačení tlačidla v tabuľeke sa v *CultureController.cs* zavolá metóda *Edit*. V metóde *Edit* sa nachádza odkaz na metódu *getAuthorById* ktorej výskedkom je daný autor z databázy s daným Id.

Ďalej sa vo *CultureController.cs*. nachádza metóda *HttpPost EditAutor*. Táto metóda sa zavolá po potvrdení upravovacieho formulára tlačidlom *Upravit*, V tejto metóde sa volá metóda *EditExistingAuthor*. Pomocou tejto metódy sa prepíšu všetky dáta autora s daným Id podľa toho, čo užívateľ zadal do formulára pre úpravu užívateľa. Nakoniec je užívateľ presmerovaný na stránku Ukázat všehny autory.

2.5.9 Odhlásenie

Odhlásenie užívateľa prebieha pomocou kliknutia užívateľa na sekciu odhlásiť v pravej hornej časti stránky. Následne sa zavolá metóda Logout, ktorá sa nachádza v *CultureController.cs*. Pomocu tejto metódy sa vymaže užívateľ z cookie súboru stráky a webová aplikácia následne presmeruje užívateľa na prihlasovací formulár.

2.6 Zmena jazyka aplikácie

V pravej hornej časti sa na stránke nachádza combobox ktorý užívateľovi umožňuje zmeniť jazyk stránky. Na výber sú dva jazyky a to Český a Anglický viz obrazok home.

V triede Startup.cs sa nachádza list List<CultureInfo>. V tomto liste sa nachádzajú dva parametre a to cs ako český jazyk a en ako anglický jazyk. Ako základný

jazyk pri načítaní stránky je zvolený parameter cs, teda český jazyk Pri kliknutí na sa v *CultureController.cs* zavolá metóda SetCulture, ktorá má dva parametre typu string a to culture a returnUrl. Do parametru returnUrl sa načíta aktuálna url adresa stránky. Do parametru culture sa zasa zapíše parameter z List<CultureInfo>. Tento parameter sa zapíše do cookie súborov stránky. Nakoniec je užívateľ presmerovaný na rovakú url adresu na ktorej sa nachádza a to vďaka uloženej url adrese v parametri returnUrl.

Vo webovej aplikácii sa nachádza zložka Resources. V Resources sa nachádza zložka Views. Štruktúra tohto súboru je presne taká istá ako je súbor Views, kde sa nachádzajú všetky Views.cshtm, vďaka ktorým je aplikácia prístupná pre užívateľa, no namiesto súborov Views.cshtm sa tu nachádzajú Resource files. Tieto Resource files majú ten istý názov ako Views.cshtm, s tým rozdielom že sú tu vždy dva krát, pretože jeden má príponu cs a druhý en. Resource files k tomuto view majú cestu Resources/Views/Video. Tu sa nachádzajú dva Resource files s rovnakým názvom ale rozdielnou príponou a to Play.cs.resx a Play.en.resx. Každý tento resource file obsahuje tabuľku, ktorá obsahuje tri stĺpce a to Name, Value a Comment. V stĺpci Name majú oba resource file rovnaké hodnoty, no v stĺpci Value majú hodnoty rozdielne. Stĺpec Value sa vypĺňa podľa toho akú príponu daný resurce file má a teda buď cs (česky) alebo en (anglicky). To teda znamená, že ak má Play príponu cs, v stĺpci Value bude český ekvivalent daného slova alebo skupiny slov, zatiaľ čo v Play s príponou en v stĺpci Value bude zasa anglický ekvivalent daného slova alebo skupiny slov.

Všetky dáta čo sú zapísané v resorce files sa dajú zobraziť v danom view.cshtml a to pomocou kľúčového slova ktoré je zadané v stĺpci Name. Keď teda slovo ktoré sa nachádza v resource file Play, by malo byť zobrazené na stránke podľa daného jazyka, treba ho vložiť do Play.cshtml. Do Play.cshtml musí byť vložený(inject) localizer v ktorom je uložená hodnota cs alebo en. Na stránke sa následne slovo z resource file play zobrazí pomocou príkazu @localizer[“Hello“]. Výsledok bude, že ak bude jazyk nastavený na český (cs) tak sa na stránke Play.cshtml namiesto @localizer[“Hello“] načítali dané dáta z Play.cs.resx, v tomto prípade Dobrý den. Ak by bol jazyk nastavený na anglický (en), tak by sa na stránke Play.cshtml namiesto @localizer[“Hello“] načítali dané dáta z Play.en.resx, v tomto prípade Hello.

V aplikácii sa teda Resource files na zobrazenie kontextu podľa zvoleného jazyka.

2.7 Tagy

Obsahom webovej aplikácie sú taktiež dáta ktoré boli spracované neurónovými sieťami (deep learning, umelá inteligencia). Na základe týchto dát je možné filtrovať videá podľa ich daných klasifikácii (tagov).

Webová aplikácia obsahuje triedenie podľa dvadsiatich tagov a to:

- *Body*,
- *Digit*,
- *Effect*,
- *Element(air)*,
- *Element(earth)*,
- *Element(fire)*,
- *Element(water)*,
- *Face*,
- *Image processing keying*,
- *Interior*,
- *Landscape*,
- *Lette*,
- *Machine (TV set)*,
- *Machine (car)*,
- *Machine vision(fisheye)*,
- *RuttExtraVideoSynthesizer*,
- *Steina*,
- *Woody*,
- *Volin*,
- *Stripes*.

Každá z týchto tagov obsahuje dva parametre a to time a prediciton

```
using (VideoDbContext dbContext = new VideoDbContext())
{
    List<Tag> allTags = dbContext.tags.ToList();

    TagsJson.Rootobject parsedJson = Newtonsoft.Json.JsonConvert.DeserializeObject<TagsJson.Rootobject>(json);

    foreach (TagsJson.TagsInfoData tid in parsedJson.GetTagListsList())
    {
        if (tid.TagInfos == null)
            continue;
        float predCount = 0;
        int sections = 0;

        Dictionary<float, float> tagdic = new Dictionary<float, float>();


        foreach (TagsJson.TagInfo tagI in tid.TagInfos)
        {
            if (tagdic.ContainsKey(tagI.time))
            {
                if (tagI.prediction > tagdic[tagI.time])
                {

                    tagdic[tagI.time] = tagI.prediction;
                }
            }
            else
            {
                tagdic.Add(tagI.time, tagI.prediction);
            }
        }

        foreach (KeyValuePair<float, float> dic in tagdic)
        {
            TagPosition tags = new TagPosition();
            tags.Prediction = dic.Value;
            tags.Time = dic.Key;
            tags.VideoID = video.ID;
            tags.Tag = allTags.Where(x => x.Name == tid.TagName).First();
            dbContext.tagPositions.Add(tags);
            predCount += dic.Value;
            sections++;
        }


        TagWeight tagWeight = new TagWeight();
        if ((int)((predCount * 100) / sections) != 0)
        {
            tagWeight.Percentage = (int)((predCount * 100) / sections);
        }
        tagWeight.Tag = allTags.Where(x => x.Name == tid.TagName).First();
        tagWeight.VideoID = video.ID;
        dbContext.TagWeight.Add(tagWeight);
    }
    dbContext.SaveChanges();
```

Obr. 2.14: Práca so súborom JSON.

Parameter time určuje v akej časti videa sa vyskytuje daný tag a parameter prediction určuje aké percentuálne zastúpenie má tento tag na danom snímku.

Pri nahrávaní videa do webovej aplikácie sa k videu vloží JSON súbor. Tento súbor sa pri nahrávaní parsuje a ukladá do premennej json. To sa deje v *UploadController.cs* v metóde UploadFile. V tejto metóde sa volá metóda SaveVideo, ktorej je predaná premenná json.

Premenná json je spracovávaná nasledujúcim kódom viď. obr. 2.14:

Vďaka tomuto kódu sa prejde celý naparsovaný json a dáta z tohto súboru sa

zapíšu do daných tabuliek. To čo sa do jednotlivých tabuliek zapisuje je taktiež ukázané na diagrame databázy vid.obrazok database diagram. Vďaka tomu že sa súbor celý spracoval a uložil do databázy nie je potreba ukladať tento súbor aj do webovej aplikácie.

Na základe týchto dát má každé video pridelené určité tagy ktoré sa v danom videu nachádzajú. Pri nahrávaní do databázy daný kód vid obrazok kod, taktiež ráta percentuálne zastúpenie daného tagu v celom videu. Keďže je tento údaj už zapísaný v databáze nie je potreba už rátať dané percentuálne zastúpenia, čo značne ušetrí nároky na webovú aplikáciu pri zobrazovaní dát. Dáta z databázy sú zobrazované pri zobrazení všetkých videí, kde sa zobrazuje pod videom percentuálne zastúpenie tagov v danom videu vid obrazok dbview.Pod videom sa zobrazujú teda iba tagy ktoré majú vo videu väčšie zastúpenie ako 1 percento Na stránke DbView sa nachádzajú tlačidlá s názvami tagov. Po kliknutí na dané tlačidlo sa do *VideoController.cs* sa zavolá metóda TagResults ktorej výsledkom je list videí, ktoré obsahujú daný tag, na ktorý užívateľ klikol. To umožňuje prehľadnejšie zobrazovanie videí a tak triedenie videí podľa dát získaných neurónovými sieťami.

Tieto tagy sa však zobrazujú aj na stránke Paly. Na tejto stránke sa zobrazujú klasifikácie (tagy) ktoré sa na danom snímku nachádzajú a s akým percentuálnym zastúpením. Tieto tagy sa zobrazujú na pravej strane vedľa videa vid obrazok play. Zobrazovanie týchto tagov v reálnom čase je riešené pomocou JQuerry a metódy ajax ktorú JQuerry obsahuje. Vďaka tomu webová aplikácia vie zistiť daný čas videa ktoré sa prehráva a tak zobrazovať iba tie tagy, ktoré sa vo videu nachádzajú na aktuálnom snímku. Pomocou metódy ajax teda webová aplikácia každú sekundu kontroluje aktuálny čas prehrávaného videa, zobrazí dané tagy, ktoré majú väčšie zastúpenie ako 50percent na aktuálnom snímku a tým je užívateľovi umožnené sledovať dáta spracované pomocou neurónových sietí v aktuálnom čase videa.

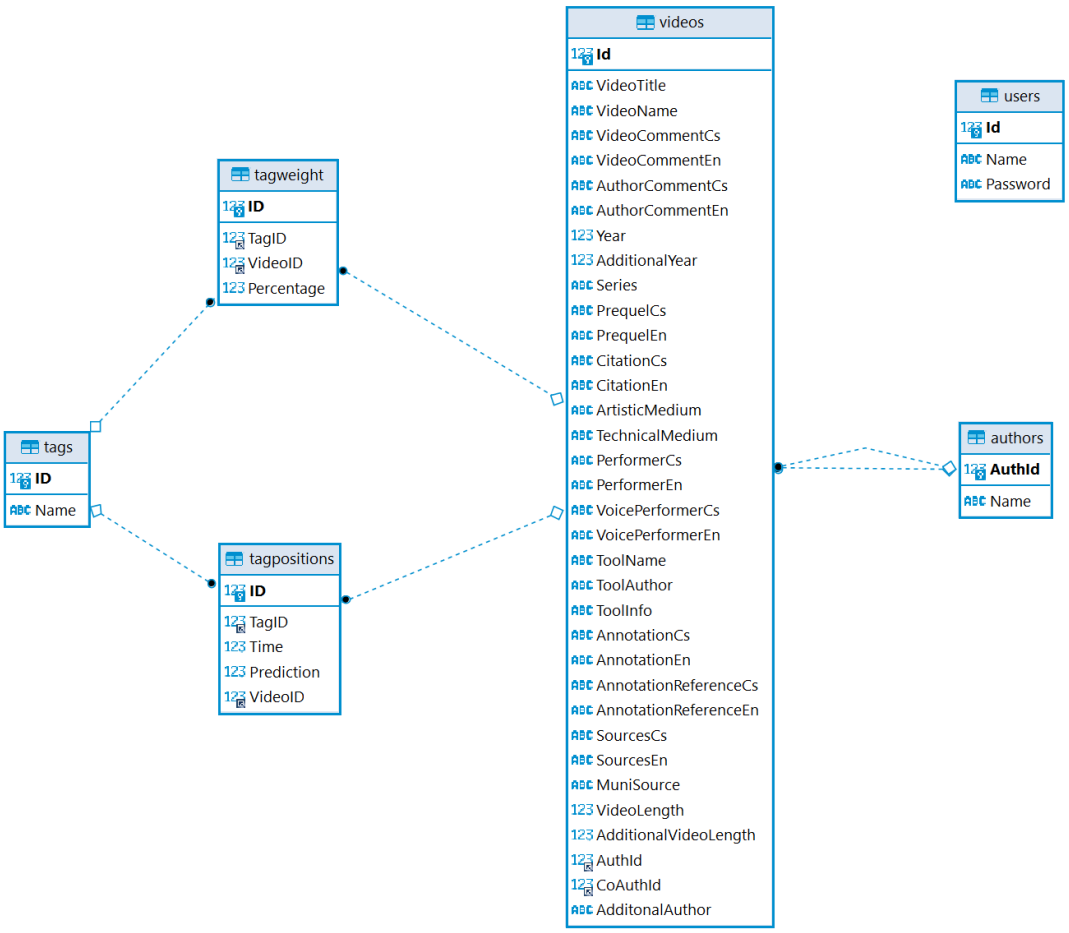
2.8 Databáza

Aplikácia využíva databázu MariaDB, Táto databáza je rýchla a to zabezpečuje rýchle a hladké zapisovanie, upravovanie, mazanie a čerpanie dát z databázy. Taktiež je veľmi dobrá pre začiatočníkov, pretože podpora MariaDB je veľmi dobrá. Keď má programátor nejaký problém a obrátiť sa s tým na podporu, zvyčajne podpora veľmi rada pomôže tento problém vyriešiť.

Na vytváranie databázy bol použitý softvér HeidiSQL. Tento softvér umožňuje pripojenie sa k MariaDB databáze a následne s ňou pracovať. Po prihlásení sa do databázy užívateľ môže vytvárať priečinky a čo je podstatné tabuľky. Databáza ktorá bola k aplikácii vytvorená nesie názov *Videopresentation* a v nej sa nachádza poddatabáza s názvom *video_presentation*. Vo *video_presentation* je vytvorená tabuľka

videos. V tejto tabuľke sa nachádzajú 4 stĺpce inak nazývané aj polia. Každý tento stĺpec sa volá inak a je určený na zápis iných hodnôt.

2.8.1 Štruktúra databázy



Obr. 2.15: ER Diagram.

2.8.2 Tabuľky

V každej tabuľke sa nachádza primárny kľúč. Jeho dátový typ je integer, čo znamená že sa tu budú zapisovať čísla. Ďalej je zadefinované, že tento primárny kľúč bude auto increment, čo zaručí že s pridaním každého záznamu sa primárny kľúč zvýši o jedna. To že každý nový záznam má vlastný primárny kľúč, ktorý sa vďaka funkcii auto increment vždy navyšuje o jedna zabezpečuje že sa v tabuľke nebudú nachádzať dva rovnaké záznamy. Preto webová aplikácia pri mazaní záznamu pracuje s týmto primárnym kľúčom a nie s inými parametrami. Je zaručené, že vždy sa vymaže iba

ten konkrétny záznam, ktorý užívateľ zadal. Primárny kľúč nemôže mať dva rovnaké záznamy.

V tabuľkách sa takisto nachádza aj cudzí kľúč. Na základe tohto cudzieho kľúča môžu byť dané stĺpce tabuliek prepojené a webová aplikácia tak dokáže spraviť dotaz nad týmito tabuľkami práve na základe cudzieho kľúča. ER Diagram zobrazuje tabuľky ktoré sú orámované modrou farbou viď. obr. 2.15. V nich každý riadok znázorňuje jeden stĺpec v danej tabuľke. Prvý riadok je vždy primárny kľúč. Na ľavej strane tabuľky je znázornený typ ktorý obsahuje daný stĺpec a to buď ABC alebo 123. ABC značí, že hodnoty, ktoré bude obsahovať daný stĺpec sú typu TEXT, čo je obyčajný súbor znakov. 123 značí, že hodnoty, ktoré bude obsahovať daný stĺpec budú typu INT čo je číselná hodnota.

Tabuľky sú v ER Diagrame pospájané modrou prerušovanou čiarou čo značí že tabuľky majú navzájom pospájané stĺpce, pomocou cudzieho kľúča. Koľko týchto čiar je znázornených v ER Diagrame, toľkými cudzími kľúčmi sú tabuľky pospájané.

2.8.3 Videos

Tabuľka videos má najviac stĺpcov spomedzi všetkých tabuliek v danej databáze . Tabuľka má dva cudzie kľúče a to konkrétne pre stĺpce *AuthId* a *CoAuthId*. Tieto cudzie kľúče sú prepojené na tabuľku *authors* na stĺpec *AuthId*. Do tabuľky videos sa zapisujú dáta ktoré súvisia s daným videom. Na základe toho, že má tabuľka videos dva cudzie kľúče, môže webová aplikácia k týmto dátam pridať aj dáta z tabuľky *authors*.

2.8.4 Users

Tabuľka users obsahuje dáta užívateľov webovej aplikácie. Táto tabuľka nemá žiadne cudzie kľúče a ani žiadne cudzie kľúče nie sú prepojené na túto tabuľku,čo znamená, že nie je prepojená so žiadnou inou tabuľkou v databáze.

2.8.5 Authors

V tabuľke *authors* sú uložené dáta autorov. Tabuľka neobsahuje žiadne cudzie kľúče no napriek tomu je prepojená s tabuľkou *videos*, keďže tá obsahuje dva cudzie kľúče ktoré sú na túto tabuľku naviazané.

2.8.6 Tagweight

Tabuľka tagweight obsahujedáta spracované neurónovými sieťami. Táto tabuľka obsahuje dva cudzie kľúče *TagID* a *VideoID*. Kľúč *VideoID* je prepojený na tabuľku

videos na stĺpček Id. To umožňuje webovej aplikácii priradiť dané tagy k danému videu. Kľúč *TagID* je prepojený na tabuľku tags vďaka čomu má webová aplikácia možnosť pristúpiť na názov daného tagu. V tejto tabuľke je taktiež zapísaná hodnota percentage, ktorá určuje aké percentuálne zastúpenie má daný tag na danom videu.

2.8.7 Tagpositions

Tabuľka tagposition obsahuje tak isto dáta spracované neurónovými sieťami. Takisto ako tabuľka tagweight aj táto tabuľka má dva cudzie kľúče *TagID* a *VideoID* ktoré sú prepojené na tabuľky tag a *videos*. V tabuľke sú priradené dáta k tagom a to konkrétne ich daný čas a ich daná percentuálna hodnota v danom čase.

2.8.8 Tags

Tabuľka tags je naplnená danými dvadsiatimi tagami ktoré sú spomenuté. Táto tabuľka nemá žiadne cudzie kľúče no podobne ako tabuľka *authors* aj táto tabuľka je prepojená s tabuľkami tagweight a tagposition keďže tieto tabuľky obsahujú cudzie kľúče prepojené na túto tabuľku.

Záver

Bakalárska práca sa zaoberala navrhnutím webovej aplikácie, ktorá bude schopná nahrávať videá a dáta spracované neurónovými sieťami (deep learning a umelá inteligencia) a následne tieto videá prehrávať, triediť ich podľa dát spracovaných neurónovými sieťami a taktiež tieto dáta zobrazovať pri prehrávaní videa. Pri riešení práce bol použitý .Net Core na Back-end a na Front-end boli použité jazyky HTML, CSS a JavaScript. Ako databáza je použitá MariaDB. Celá webová aplikácia je postavená na modely MVC.

Webová aplikácia umožňuje užívateľovi po prihlásení sa nahrávať videá, k nim potrebné dáta a taktiež nahrávanie dát spracovaných neurónovými sieťami a ich priradenie k videám. Tieto dáta sa zaznamenávajú do databázy, odkiaľ ich je následne možné vybrať a zobraziť na stránke kde si ich užívateľ môže prehrať. Webová aplikácia taktiež umožňuje zobrazovanie a filtráciu dát z klasifikátora založenom na neurónových sieťach. Pri prehrávaní daného videa webová aplikácia zobrazuje dáta z klasifikátora v reálnom čase.V aplikácii je taktiež zabudovaná funkcia vyhľadávania. Táto funkcia umožňuje užívateľovi vyhľadať video v databáze a to na základe dvoch parametrov ako je názov videa. Do webovej aplikácie sú zakomponované aj tlačidlá vymazať a upraviť, ktoré umožňuje užívateľovi vymazať záznamy z databázy, alebo ich upraviť.

Literatúra

[1] BEAULIEU, A. *Learning SQL* [online]. Vyd. 2. United States of America: Mary E. Tresler, 2009, 321 s. ISBN 978-0-596-52083-0. Dostupné z URL <http://www.r-5.org/files/books/computers/languages/sql/mysql/Alan_Beaulieu-Learning_SQL-EN.pdf>

[2] SAHAY, R. *Hands-on with ASP.NET MVC Covering MVC 6* [online]. Vyd. 1. Quills Ink Publishing: Rahul Sahay, 2014, 554 s. ISBN 978-93-84318-54-3. Dostupné z URL <https://www.pdfdrive.com/hands-on-with-aspnet-mvc-covering-mvc-6-e158694936.html>

[3] Pennington, C.,Cardoso,J.,Miller, J.,Patterson, R., Vasquez, I. *Introduction to Web Services* [online]. Vyd. 1. Department of Mathematics and Engineering University of Madeira: Jorge Cardoso, 2006, 39 s. Dostupné z URL <https://www.researchgate.net/publication/236860265_Introduction_to_Web_Services>

[4] Widenius, M. *MySQL-MariaDB History talk* [online]. Dostupné z URL <https://mariadb.org/wp-content/uploads/2019/11/MySQL-MariaDB-story.pdf>

[5] Kalis, F. *A Brief History of SQL* [online]. Dostupné z URL <https://www.labouseur.com/courses/db/s2-A-Brief-History-of-SQL.pdf>

[6] Laurenčík, M. *SQL Podrobný průvodce uživatele* [online]. Vyd. 1. DG-rada Publishing, a.s.U Průhonu 22, Praha 7: JVěra Slavíková, 2018, 216 s. ISBN 978-80-271-2154-0. Dostupné z URL <https://www.databook.cz/sql-podrobny-pruvodce-uzivatele-5198>

[7] Foote, K. *A Brief History of Database Management* [online]. Dostupné z URL <https://www.dataversity.net/brief-history-database-management/>

[8] Vašulka, S. *Bent_Scans_Demo 2002* [online]. Dostupné z URL <https://vasulkakitchen.org/>

[9] Vašulka, S. *Machine_Vision_Demo 1978* [online]. Dostupné z URL <https://vasulkakitchen.org/>

[10] Vašulka, S. *Orka 1997* [online]. Dostupné z URL <https://vasulkakitchen.org/>

[11] Vašulka, S. *ViolinPower 1978* [online]. Dostupné z URL <https://vasulkakitchen.org/>