DELTA – Střední škola informatiky a ekonomie, s.r.o.
Ke Kamenci 151, Pardubice

Timespan Snippet Maker

Příjmení, jméno: BALAN Vlad
Třída: 4.A
Studijní obor: 18-20-M/01 Informační technologie
Školní rok: 2023-24

Zadání maturitního projektu z informatických předmětů

Jméno a příjmení:	Vlad Balan
Školní rok:	2023/2024
Třída:	4. A
Obor:	Informační technologie 18-20-M/01
Téma práce:	*Systém pro vytváření úryvků z videí/podcastů*
Vedoucí práce:	*RNDr. Jan Koupil, Ph.D.*

Způsob zpracování, cíle práce, pokyny k obsahu a rozsahu práce:

Cílem práce je vytvořit webovou aplikaci pro vytváření úryvků z videí/podcastů, které pak bude možné stáhnout.

Systém bude sloužit jako jedna velká knihovna podcastů s možností vyhledávání a filtrování, a to nejen podcastů ale i úryvků. Detail jakéhokoliv podcastu bude kromě vlastního přehrání nabízet také list úryvků, které jsou z něj vytvořeny. Úryvky budou opatřeny metadaty (popisek, timespan, hashtag, transkript apod.)

Uživatel bude mít přístup nejen k svojí knihovně (pokud je přihlášen) ale i k veřejné knihovně, kterou může prohledávat, stahovat, sdílet a duplikovat do své knihovny, kde s podcasty lze i manipulovat. Svoje podcasty s úryvky pak uživatel může navrhnout pro zveřejnění, samotné zveřejnění ale musí schválit administrátor.

Systém bude obsahovat také oprávnění superadministrátora, který má oprávnění řídit i administrátorské účty.

Stručný časový harmonogram (s daty a konkretizovanými úkoly):

- **Září:** Analýza problému, tvorba UML usecase diagramů, návrh architektury aplikace a databázového modelu
- **Říjen:** Výběr vhodných technologií, tvorba webové aplikace
- **Listopad–prosinec:** Tvorba webové aplikace
- **Leden–únor:** Testování, dokumentace projektu
- **Březen:** Dokončení dokumentace

Prohlášení

Prohlašuji, že jsem maturitní projekt vypracoval samostatně, výhradně s použitím uvedené literatury.

V Pardubicích 30. 3. 2024

Poděkování

Chtěl bych poděkovat mému školiteli, RNDr. Janu Koupilovi, za jeho úžasné tipy, triky a připomínky, bez kterých by nevznikla tato práce. Dále bych chtěl poděkovat mé rodině a přítelkyni, za to, že mě dostatečně zásobili kávou.

Rád bych také zmínil a poděkoval ChatGPT, umělé inteligenci vyvinuté společností OpenAI [1], za její nepostradatelný přínos při opravě pravopisu a formulací vět tak, aby odpovídaly českým jazykovým normám. Obsah a nápady prezentované v této práci jsou výsledkem mé práce s použitím dostupné literatury. ChatGPT výhradně přispěl k gramatické korekci a formulaci vět.

Abstrakt

Práce dokumentuje návrh a vývoj webové aplikace nazvané Timespan Snippet Maker (zkráceně TSM), která slouží k střihu a organizaci videí. Aplikace je vybavena funkcemi jako tagování a transkribování pro efektivní filtrování obsahu. Realizace projektu zahrnuje backendovou část ve formě RESTful API, webový frontend a další nástroje, které umožňují plnou funkcionalitu aplikace.

Klíčová slova

FFmpeg, Timespan Snippet Maker (TSM), video úryvek, uživatelské rozhraní, lokální uložiště, transkript, střih videa, miniatura.

Abstract

The work documents the design and development of a web application called Timespan Snippet Maker (abbreviated as TSM), which is used for video trimming and organization. The application is equipped with features such as tagging and transcription for efficient content filtering. The project implementation includes a backend part in the form of a RESTful API, a web frontend, and additional tools that enable the full functionality of the application.

Key Words

FFmpeg, Timespan Snippet Maker (TSM), snippet, user interface, local storage, transkript, video cut, thumbnail.

Obsah

1 Úvod **7**
1.1 Popis problému . . . 7
1.2 Předpokládáná funkcionalita . . . 8

2 Využité technologie **10**
2.1 React . . . 10
2.2 Laravel . . . 10
2.3 FFmpeg . . . 10
2.4 Whisper AI . . . 10

3 Realizace **11**
3.1 Architektura . . . 11
3.2 Struktura databáze . . . 12
3.3 Backend . . . 13
3.3.1 Validace dat . . . 13
3.3.2 Autentizace . . . 13
3.3.3 Ukládání souborů . . . 14
3.3.4 Stříhání videí . . . 14
3.3.5 Transkripce . . . 16
3.3.6 Stahování videí . . . 17
3.3.7 Publikace videí . . . 18
3.3.8 Duplikace videí . . . 18
3.4 Frontend . . . 19
3.4.1 Validace uživatelských dat . . . 19
3.4.2 Správa dat . . . 19
3.4.3 Optimistické aktualizace . . . 19
3.4.4 Infinite scroll . . . 20

4 Změny v průběhu práce **21**

5 Nevyřešené problémy **21**
5.1 Google Drive . . . 21

6 Závěr **22**

1 Úvod

V poslední době jsme svědky výrazného vzrůstu popularity podcastů [2], což je podporováno faktory jako zvýšená dostupnost chytrých telefonů, zlepšená internetová konektivita a rostoucí touha lidí po zábavě i během cestování [3]. Paralelně s tímto trendem však pozorujeme nárůst popularity obsahu ve formátu short-form na platformách jako TikTok, YouTube Shorts a Instagram Reels [4, 5].

Tento posun k preferenci kratšího obsahu lze přičíst různým společenským změnám, včetně touhy po rychlých a snadno stravitelných zážitcích v době, kdy jsme přetíženi informacemi a časově omezeni. Digitální éra přinesla také zkrácení pozornosti, což vyžaduje adaptaci formátů obsahu [6, 7].

I přestože podcasty nabízejí hlubší diskuse a příběhy, často vyžadují značný časový závazek od posluchačů. V reakci na tuto překážku tvůrci obsahu a platformy extrahují klíčové momenty z dlouhých podcastových epizod do krátkých videí, které trvají maximálně minutu, což vyhovuje preferencím publika [8].

Tento symbiotický vztah mezi podcasty a krátkým obsahem se rozšiřuje i na propagační úsilí. Mnoho podcastů využívá platformy jako TikTok, YouTube Shorts a Instagram Reels k představení epizod, sdílení citátů nebo ukázek hostů, což láká diváky k prozkoumání obsahu v plné délce [9, 7].

Rostoucí trend podcastů a krátkého obsahu vyžadoval nové způsoby organizace obsahu podcastů podle časových úseků. Každý, kdo se snažil vytvořit z dlouhých podcastů krátký obsah, si rychle uvědomil, jaké obtíže přináší organizace videí a získání relevantních částí. Tato výzva vedla k vytvoření Timespan Snippet Makeru.

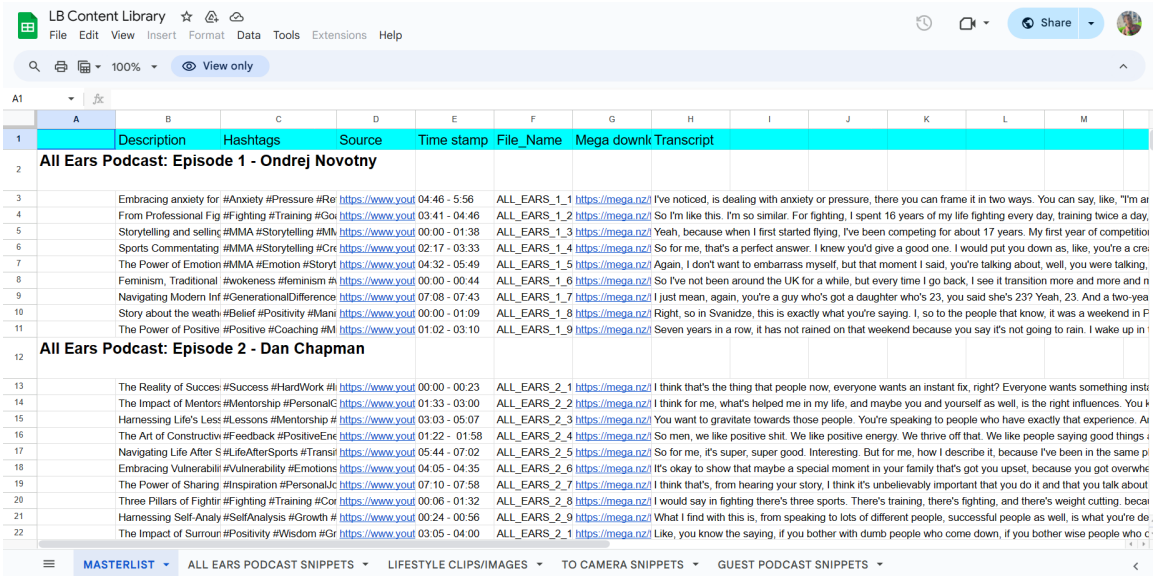
Cílem tohoto projektu je vytvořit webovou aplikaci pro stříhání a archivaci videí s rozsáhlými funkcemi, jako je například tagování a transkripce. Inspirací pro tento projekt slouží uživatelsky přívětivý design populárních aplikací, jako je Notion a YouTube, které uživatelům umožňují intuitivně a efektivně pracovat s obsahem.

1.1 Popis problému

Tvorba úryvků pro short-form content představuje výzvu při získávání a organizaci těchto fragmentů.

Uživatelé často zaznamenávají vybrané pasáže z videí do různých záznamových systémů. Jako příklad je na obrázku č.1 uveden Google Sheets. Zde si obvykle zapisují titulek, časový údaj (označovaný jako Time Stamp) a odkaz na původní video. V případě detailnějšího záznamu mohou uživatelé přidat hashtagy, odkazy na cloudová úložiště nebo dokonce transkripty.

Složitost procesu spočívá v množství různých aplikací a softwarů potřebných k takovému záznamu. Uživatelé musí zaznamenávat úryvky během sledování videa, které často bývá na



Obrázek 1: Ukázka oranizace úryvků v Google Sheets.

platformě YouTube, a buď rozdělit obrazovku, aby měli přehled o obou oknech, nebo neustále přepínat mezi nimi.

Pokud uživatel potřebuje získat určitý úryvek, musí nejprve stáhnout celý podcast, poté ho otevřít v editačním softwaru pro videa, najít požadovanou část a nakonec ji exportovat.

Důležité je, aby si uživatel dobře pojmenoval úryvek, aby ho mohl snadno najít v souborovém systému či jiném úložišti. V opačném případě, pokud ho nezachová nebo smaže a později ho potřebuje, musí celý proces opakovat znovu. Alternativou je ukládání úryvků do cloudu a jejich odkazování, jak je vidět na obrázku.

Pro zlepšení organizace úryvků mohou uživatelé přidat transkripty, což jim umožní rychle vyhledávat slova, která se v daném videu objevují. Avšak pro získání transkriptu musí být úryvek nejprve zpracován pomocí transkripčního softwaru.

1.2 Předpokládáná funkcionalita

Cílem tohoto projektu je vytvořit aplikaci, která sjednotí všechny potřebné funkce do jednoho nástroje. Jedinou výjimkou bude nutnost nahrát celé video na server.

Uživatelé budou mít k dispozici knihovnu videí obsahující náhledové obrázky, podobné těm na YouTube, a také parametry jako typ videa, hostitelé a hosté. Po kliknutí na libovolnou položku ve videoknihovně se zobrazí samotné video. Na rozdíl od YouTube, kde je na pravé straně seznam videí, zde uživatel uvidí seznam snippetů příslušících k danému videu.

Každý snippet bude doprovázen formulářem, do kterého uživatel zadá časový rozsah, titulek a tagy. Uživatel si pak bude moci snippet stáhnout bez nutnosti otevírat střihový software. Po exportu videa se také automaticky vygeneruje transkript. Díky těmto funkcím bude uživatel

schopen snadno vyhledávat videa nebo snippety pomocí miniatur, tagů, hostů nebo transkriptu.

Všechny tyto soubory budou uloženy na uživatelův Google Drive, což usnadní přístup k jejich obsahu odkudkoli a zároveň odstraní jakékoliv možné problémy s úložištěm, které by se mohly objevit při publikování projektu.

Jako bonusová funkce projektu bude možnost publikovat videa a sdílet je s ostatními uživateli. Ti budou mít možnost stahovat úryvky z videí nebo dokonce duplikovat celá videa a vytvářet si vlastní úryvky podle svých potřeb.

2 Využité technologie

2.1 React

React je knihovna JavaScriptu, která je používána k tvorbě uživatelských rozhraní. Používá se k vytváření interaktivních a responzivních webových aplikací. Knihovna React je vyvinuta společností Facebook a je známá svou efektivní a deklarativní metodologií pro tvorbu uživatelských rozhraní. [10]

2.2 Laravel

Larevel je open-source framework pro vývoj webových aplikací v jazyce PHP. Je navržen tak, aby zjednodušil a urychlil proces tvorby moderních a robustních webových aplikací. Laravel poskytuje širokou škálu funkcí a nástrojů, včetně routování, správy databáze, šablonování, ověřování, testování a mnoho dalšího. [11]

2.3 FFmpeg

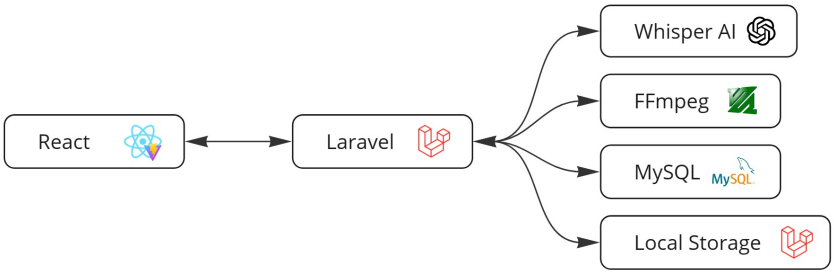
FFmpeg je open-source software určený pro zpracování multimediálních souborů. Tento nástroj umožňuje uživatelům manipulovat s různými formáty zvuku a videa, včetně jejich konverze, střihu, spojování a převodu. FFmpeg je využíván široce ve vývoji softwaru, při tvorbě videí, streamování médií a mnoha dalších aplikacích. Je známý pro svou výkonnost, flexibilitu a podporu mnoha různých formátů. [12]

2.4 Whisper AI

Whisper AI je technologie umělé inteligence vyvinutá společností OpenAI. Tato technologie se zaměřuje na oblast zpracování přirozeného jazyka a je navržena k porozumění a generování lidského jazyka. Whisper AI je schopna analyzovat a interpretovat textové informace, porozumět kontextu a generovat odpovídající textové výstupy. Tato technologie má široké uplatnění v oblastech jako jsou chatboti, analýza emocí, automatické generování obsahu a další aplikace, které vyžadují porozumění lidskému jazyku a jeho produkci. [13]

3 Realizace

3.1 Architektura



Obrázek 2: Architektura aplikace.

Projekt představuje moderní a robustní systém, který využívá nejnovější technologie k zajištění efektivního a spolehlivého provozu. Backend je postaven na Laravelu, populárním PHP frameworku určeném pro tvorbu webových aplikací a API. Laravel poskytuje solidní základ pro vytváření RESTful API, což usnadňuje komunikaci s frontendem a dalšími systémy. Dokumentaci API lze nalézt na této stránce: Dokumentace TSM RESTful API.

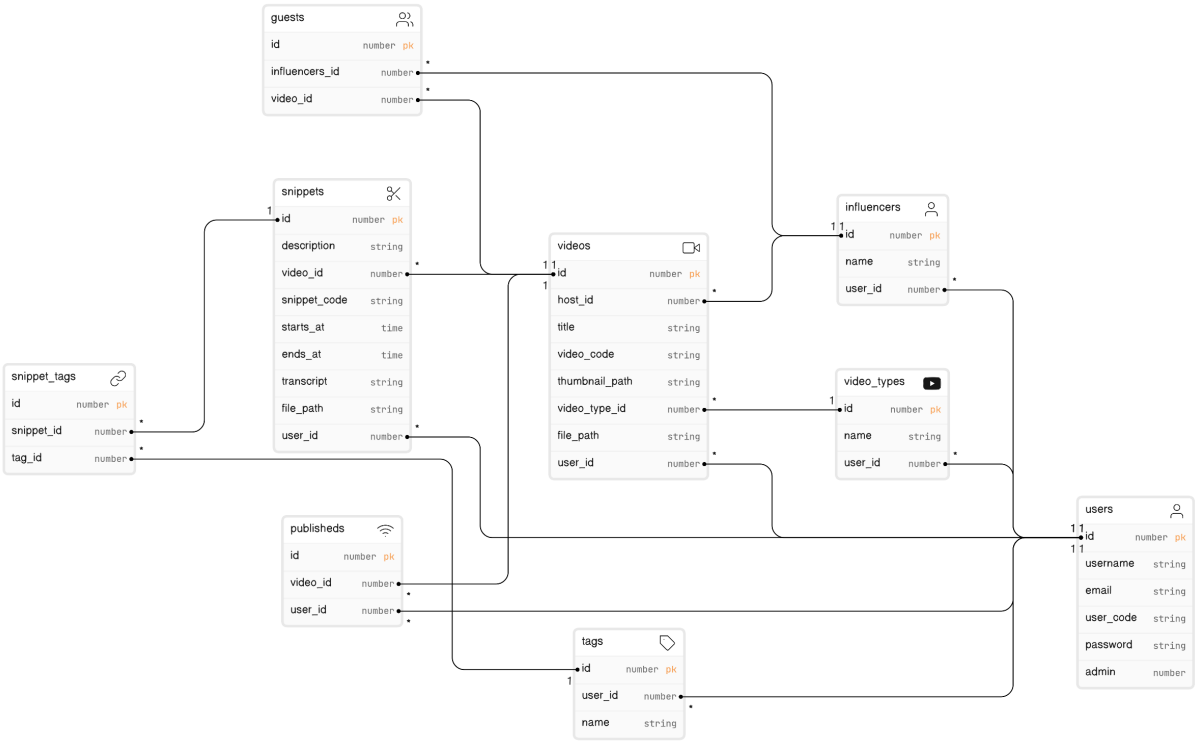
Databáze je propojena s backendem pomocí Laravel Migrations a využívá relační databázi MySQL pro spolehlivé ukládání dat. Pro ukládání uživatelských souborů, jako jsou videa a náhledové obrázky, je využíváno Laravel Local Storage, což umožňuje efektivní správu souborů přímo v rámci aplikace.

Kromě běžné komunikace s databází projekt využívá dalších API, včetně Whisper AI pro transkripci videí. Laravel dále využívá nástroje jako je FFmpeg pro stříhání videí, což umožňuje pokročilou manipulaci s multimediálními soubory.

Pro autentizaci uživatelů se využívá Laravel Sanctum API token a middleware, což zajišťuje bezpečný a efektivní proces ověřování uživatelů a zabezpečení API.

Frontend aplikace je postavena na moderním nástroji Vite spolu s frameworkem React, což umožňuje rychlý vývoj a vysokou výkonnost frontendu. Pro komunikaci mezi frontendem a backendem je využívána knihovna axios, která zajišťuje spolehlivé zasílání požadavků mezi oběma částmi aplikace. Autentizace probíhá pomocí Bearer tokenu.

Celkově je architektura projektu navržena tak, aby poskytovala moderní, výkonný a spolehlivý systém, který plně vyhovuje potřebám uživatelů a zajišťuje efektivní provoz aplikace. Architekturu názorně shrnuje schéma na obrázku č.2.



Obrázek 3: Schéma databáze.

3.2 Struktura databáze

Jak můžete vidět na obrázku č.3, databáze projektu je navržena s ohledem na efektivní ukládání a správu dat, což má za cíl podporovat požadovanou funkcionalitu a usnadnit provoz systému. Vztahy mezi tabulkami jsou definovány pomocí cizích a primárních klíčů, což zajišťuje integritu dat a propojení mezi jednotlivými entitami.

Každá hlavní tabulka je spojena s tabulkou ”users” pomocí user_id, což umožňuje uživatelům přistupovat pouze k datům, která jim patří, a zlepšuje organizaci a řízení jejich vlastních knihoven bez zbytečných cizích dat.

Nejdůležitější tabulkou je ”videos”, která jsou vždy přiřazena konkrétnímu uživateli a mohou mít definovaného hostitele videa (host_id) a hosty, kteří se nacházejí ve videu (tabulka guests). Každé video je také přiřazeno typu (video_type_id) pro snadnější filtrování a obsahuje snippety, které slouží k dalšímu rozčlenění obsahu.

Snippety dále obsahují tagy a transkript pro další filtrování podle různých kritérií. Navíc disponují sloupci starts_at a ends_at, což usnadňuje správu časových úseků videa.

Kromě toho tabulky ”users”, ”videos”a ”snippets” obsahují unikátní řetězce znaků (user_code, video_code, snippet_code) pro organizaci lokálního úložiště. Sloupce file_path a thumbnail_path pak umožňují přímý přístup k souborům v Laravel Local Storage.

Publikační systém je implementován pomocí tabulky ”published”, která obsahuje pouze ID publikovaného videa a umožňuje spojení s dalšími souvisejícími daty.

3.3 Backend

3.3.1 Validace dat

K zajištění platnosti a bezpečnosti dat poskytnutých uživatelem před jejich dalším zpracováním využívá Laravel na serverové straně tzv. Request data validation. Tato metoda je efektivním nástrojem pro udržení integrity a bezpečnosti dat v aplikaci.

Na backendu se validace dat provádí pomocí formulářových požadavků, což jsou vlastní třídy zodpovědné za ověření přicházejících dat ještě předtím, než dorazí k řadiči. Tyto formulářové požadavky obsahují pravidla, která definují, jak mají být data ověřena, jako například povinnost vyplnění pole, typy dat a omezení.

Když je požadavek odeslán na backend, Laravel automaticky směruje požadavek přes odpovídající třídu formulářového požadavku na základě typu požadavku. Formulářový požadavek poté provede validaci podle pravidel definovaných v něm. Pokud validace selže, Laravel automaticky vrátí odpověď s validačními chybami. V opačném případě umožní požadavku pokračovat k odpovídající metodě řadiče.

Pro požadavky, které zahrnují nahrávání souborů, jako jsou například videa nebo obrázky, je validace nahrávaných souborů prováděna ve třídách formulářových požadavků pomocí validačních pravidel specifických pro nahrávání souborů poskytovaných Laravel, jako jsou MIME typy a omezení velikosti souboru. [14]

3.3.2 Autentizace

Uživatel se registruje pomocí API endpointu, který obvykle přijímá e-mail, heslo a další potřebné údaje pro vytvoření nového účtu. Laravel Sanctum zpracovává tyto informace a ukládá nový záznam do databáze, který reprezentuje registrovaného uživatele. Zároveň je generován jedinečný API token, který slouží k autentizaci tohoto uživatele. Tento token je následně odeslán na frontend, kde je uložen do lokálního úložiště. Na základě tohoto tokenu a uživatelských dat se na frontendu určuje, kam uživatele přesměrovat.

Když uživatel získá svůj vygenerovaný API token, tento token je začleněn do hlaviček požadavku nebo předán jako součást dotazovacích parametrů při interakci se serverem.

Laravel Sanctum poskytuje middleware funkce, které pečlivě monitorují příchozí požadavky a ověřují autentičnost API tokenu poskytnutého uživatelem. Pokud je token platný, požadavek pokračuje ke svému zamýšlenému cíli, ať už je to kontroler nebo routa. Naopak, pokud je token chybějící nebo neplatný, Sanctum okamžitě vydá chybovou hlášku oznamující neoprávněný pokus o přístup.

Po úspěšném ověření API tokenu Sanctum načte přidružený uživatelský záznam z databáze. Tento klíčový krok umožňuje aplikaci přistupovat k důležitým informacím ověřeného uživatele a provádět akce na jeho účet.

Laravel Sanctum preferuje stavový přístup, který využívá dotazů do databáze k ověření uživatele. To znamená, že každý požadavek na API vyžaduje dotaz do databáze k ověření platnosti uživatelova tokenu. [15]

3.3.3 Ukládání souborů

Při registraci uživatele je generován unikátní user_code, složený z náhodných znaků. Tento kód slouží k vytvoření uživatelského adresáře v lokálním úložišti, kde se uchovávají všechny soubory uživatele. Tím je zajištěno, že každý uživatel má svůj vlastní bezpečný prostor pro ukládání a správu svých souborů. [16]

Při nahrávání videa je videu přidělen unikátní kód, nazývaný video_code, který slouží k vytvoření adresáře přímo pro dané video v uživatelském úložišti. Tento adresář pak slouží jako úložiště pro všechny soubory související s videem, včetně samotného videa. Ihned po vytvoření adresáře videa je automaticky vytvořen podadresář ”snippets” pro uložení všech úryvků extrahovaných z videa. Úryvky jsou navíc označeny unikátní sadou písmen, tzv. snippet_code, která slouží k pojmenování souborů s úryvky. Teprve poté, co je tento adresářový strom vytvořen, je video nahráno do adresáře videa jako soubor ”video.mp4”.

Při nahrávání obrázku pro video, který je známý také jako miniatura, je tento obrázek uložen v adresáři videa pod odpovídajícím názvem souboru (například thumbnail.jpg/png/...).

Cesty ke všem souborům jsou ihned uloženy ve vytvořeném záznamu, kde tato cesta vychází z uživatelského adresáře. Stejná cesta je pak použita k získání URL pro tento prvek při požadavku daného typu.

Při mazání prvků se buď smaže jednotlivý soubor nebo celý adresář v závislosti na typu prvku.

3.3.4 Stříhání videí

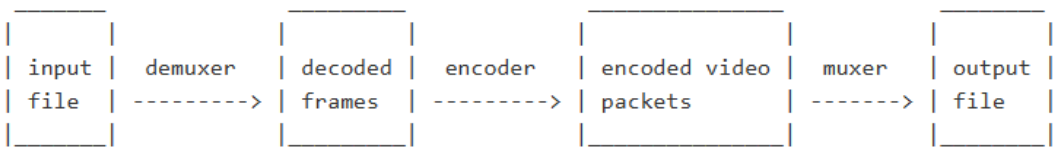
Pro střih videa musí uživatel nejprve nahrát video soubor, ze kterého bude stříháno, na server. Tento proces je součástí vytvoření záznamu snippetu. Po vytvoření záznamu snippetu má uživatel možnost specifikovat časový rozsah videa, tedy začátek a konec úryvků videa, které chce extrahovat. Teprve poté může uživatel odeslat požadavek na střih videa na server. Po zvalidování uživatelského vstupu server volá nástroj FFmpeg a specifikuje mu požadavky a nastavení pro střih videa. Poté FFmpeg spustí proces stříhání.

Při střihu pomocí programu FFmpeg se proces dělí na několik klíčových kroků. Nejprve poskytneme FFmpegu původní multimediální soubor jako vstup a následně určujeme začátek a konec segmentu (starts_at, ends_at), který chceme vyjmout ze vstupního souboru.

FFmpeg dekóduje vstupní soubor, aby multimediální data byla připravena k dalšímu zpracování. Poté se posune na určený začáteční bod ve vstupním souboru a začne číst data od tohoto místa až do dosažení určeného koncového bodu.

Jakmile FFmpeg získá požadovaný segment, pokračuje v kódování tohoto úseku do nového multimediálního souboru. Tento proces může zahrnovat transcoding zvukových nebo video proudů do jiného formátu nebo bitrate, v závislosti na nastavení výstupu.

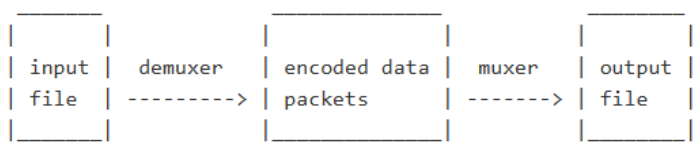
Nakonec je zakódovaný segment uložen jako výstupní soubor, který obsahuje pouze extrahovanou část původního mediálního souboru. Tento soubor je ukládán do adresáře ”snippets” pod adresářem videa, ze kterého bylo provedeno extrahování. Každý extrahovaný soubor je pojmenován pomocí unikátního kódu snippet_code, který je automaticky vytvořen při vytváření záznamu v databázi.



Obrázek 4: Základní stříhací process v FFmpegu.

Uživatelé mají možnost nastavit si režim střihu videí, který vyhovuje jejich potřebám. K dispozici jsou dva hlavní režimy střihu. Prvním je základní střihový proces, který vyžaduje několik fází kódování a dekódování, což může prodloužit dobu střihu. Druhou možností je rychlý střih, který provádí střih videa prostým kopírováním, aniž by došlo k opětovnému komprimování obsahu. Tento rychlý střih však může obsahovat nepřesnosti v prvních několika sekundách videa.

Důvodem pro poskytování obou režimů je základní proces střihu videa ve FFmpeg, který vyžaduje dekódování videoproudu ze vstupního souboru a následné zakódování do výstupního formátu, aby se zachovala přesnost snímků. Pro lepší představu o celém procesu lze nahlédnout na obrázek č. 4. Tento proces dekódování a opětovného zakódování může být velmi náročný jak z hlediska výpočetního výkonu, tak času, zvláště u rozsáhlých nebo vysoce kvalitních videí. V některých případech může export úryvku trvat déle než samotná délka původního videa.



Obrázek 5: Stříhací process FFmpegu použitím ”copy” parametru. Zdroj: [17]

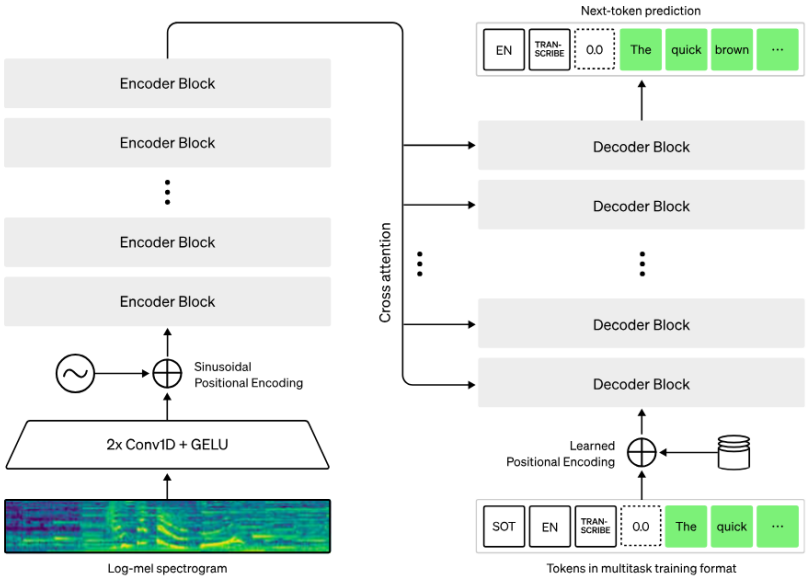
Pro urychlení exportu videa je k dispozici alternativní metoda, která využívá parametr ”copy” v příkazu ffmpeg. Tímto způsobem se zachovává stejný video kodek jako ve zdrojovém videu, přičemž se pouze kopíruje datový tok videa z jednoho souboru do druhého. Tento proces je názorně popsán v obrázku č.5. Díky této metodě je proces výrazně urychlen, protože není potřeba opakovaně komprimovat video. Avšak tato technika má jednu nevýhodu - nepřesnost

na začátku videa. Z vlastní zkušenosti se ukázalo, že tato nepřesnost často trvá mezi 3 až 10 sekundami.

Během použití rychlého režimu střihu videa se často vyskytuje problém ztráty prvních pár sekund videa. Tento jev je způsoben charakterem klíčových snímků videa. FFmpeg při použití parametru ”c:v copy” začne kopírovat videoproud od nejbližšího klíčového snímku před určeným bodem střihu. Tato nutnost vyplývá z požadavku, aby FFmpeg začal právě od klíčového snímku pro přesné dekódování a kódování videa. V důsledku toho může docházet ke ztrátě několika sekund na začátku videa, pokud se klíčový snímek nenachází přesně v místě určeného bodu střihu. Tento jev je běžným důsledkem omezení klíčových snímků a často se vyskytuje při použití rychlého režimu střihu videa. [17, 18]

3.3.5 Transkripce

Transkripce (převod řeči na text) probíhá pouze na snippetech ihned po střihu videa. Laravel v této fázi zasílá HTTP požadavek na koncový bod OpenAI API s audio souborem, který má být převeden na text. Po úspěšné validaci komunikace je spuštěna transkripce prostřednictvím aplikace Whisper AI.

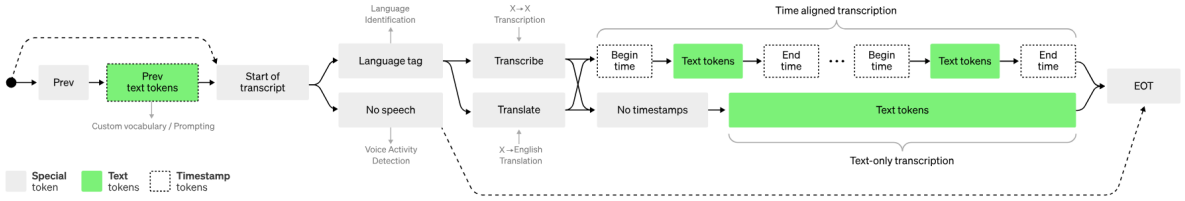


Obrázek 6: Whisper sequence to sequence proces. Zdroj: [13]

OpenAI Whisper je model umělé inteligence navržený k porozumění a přepisu mluveného jazyka. Jedná se o automatický systém rozpoznávání řeči (ASR), který je schopen převádět mluvený jazyk na psaný text. Prošel tréninkem na rozsáhlém datovém souboru obsahujícím 680 000 hodin různorodých multijazyčných a multitaskových dat získaných z internetu. Tento rozsáhlý trénink zlepšil schopnost Whisper zvládat různé výzvy, jako jsou přízvuky, pozadí

šumu a odborný žargon.

Architektura Whisper AI je založena na zpracovává vstupního zvuk ve 30sekundových segmentech, převede jej na logaritmický Mel spektrogram a poté ho předává kódéru k vygenerování odpovídajícího textového přepisu. Tento process názodně shrnuje schéma na obr. 6.



Obrázek 7: Whisper multitasking format. Zdroj: [13]

Whisper AI nabízí několik funkcí nad rámec základního přepisu. Umí identifikovat jazyky, poskytovat časové razítko na úrovni frází, přepisovat multijazyčnou řeč a překládat řeč do angličtiny [13, 19]. Tento process podrobněji znázorňuje schéma na obr. 7.

Po přijetí transkriptu od OpenAI API je transcript uložed do databáze k určitému snippetu.

Bohužel není tato služba zcela bezplatná a má svá omezení. Používání OpenAI API stojí $0.006 za minutu transkripce, přičemž uživatel má k dispozici pouze $5 kreditu po přihlášení. To znamená, že transkripce může být prováděna pouze do vyčerpání tohoto kreditu.

Jedním z hlavních omezení je maximální velikost souboru pro nahrávku, stanovená momentálně na 25 MB. To vede k nutnosti provést dodatečné úpravy nebo rozdělit nahrávku na menší části v případě větších souborů, což může zpomalit proces a snížit efektivitu aplikace [20].

Pro komunikaci s OpenAI API je v projektu využíván nástroj ffmpeg, který převede video na zvukový formát před odesláním k transkripci. Tento krok je nezbytný pro omezení objemu dat odesílaných na server a zajištění dodržení maximální velikosti 25 MB.

3.3.6 Stahování videí

Když uživatel aktivuje akci stahování, obvykle kliknutím na tlačítko ”Stáhnout”nebo jiný prvek uživatelského rozhraní, vyvolá se HTTP požadavek na backend. Tento požadavek je nastaven tak, aby očekával binární odpověď, což znamená, že server vrátí binární data reprezentující video soubor.

Server přijme požadavek a ověří uživatele, aby zajistil, že pouze autorizovaní uživatelé mohou stahovat video. Poté získá příslušný video soubor z úložiště na základě poskytnutého identifikátoru. Binární data souboru jsou následně streamována jako tělo odpovědi zpět klientovi, což umožňuje prohlížeči spustit proces stahování [21].

Po úspěšném přijetí odpovědi frontend extrahuje název souboru z hlaviček odpovědi a vytvoří URL pro binární data. Nakonec je v DOM dynamicky vytvořen odkaz s nastaveným atri-

butem pro stahování, který odkazuje na URL s binárními daty. Kliknutím na tento odkaz spustí uživatel proces stahování videa ve svém prohlížeči [22].

3.3.7 Publikace videí

Před publikací videa pečlivě ověřujeme uživatelské přihlašovací údaje a oprávnění, aby bylo zajištěno, že daný uživatel disponuje nezbytnými právy k provedení této akce.

Dále, ještě před samotným pokračováním v procesu publikace, systém provádí validaci videa, aby se ujistil, že splňuje všechna požadovaná kritéria. Tato kontrola zahrnuje například ověření existence základních souborů, jako jsou samotné video a jeho náhled.

Jakmile je tato validace úspěšně dokončena, video je označeno jako publikované vytvořením nového záznamu v databázi. Tímto způsobem je také zachycena vazba mezi publikovaným videem a uživatelem, který tuto akci inicioval.

Po dokončení procesu publikace obdrží uživatel zpětnou vazbu, která označuje výsledek této operace. Zpráva o úspěchu potvrzuje, že video bylo úspěšně publikováno. V případě jakýchkoliv problémů během procesu obdrží uživatel odpovídající chybovou zprávu.

Jakmile je video úspěšně publikováno, stává se přístupným pro ostatní uživatele nebo veřejnost, kteří mohou video najít na veřejné stránce. Pokud uživatel není administrátorem, nebude mít možnost jakkoliv měnit nebo upravovat záznam videa. Jeho možnosti jsou omezeny na prohlížení dat, stahování videa nebo duplikaci celého záznamu do vlastní knihovny.

Pokud tvůrce videa později rozhodne o schování videa, bude tento záznam odstraněn z databáze.

3.3.8 Duplikace videí

Uživatel spouští proces duplikace videa pomocí uživatelského rozhraní aplikace, například kliknutím na tlačítko ”Duplikovat” ve správě videí.

Backend aplikace přijímá požadavek na duplikaci konkrétního videa. Před samotným zahájením procesu duplikace systém ověřuje, zda je video veřejné nebo soukromé. Pouze veřejná videa jsou povolena k duplikaci. Poté backend vytváří kopii vybraného videa.

Novému videu je přidělen nový náhodný identifikátor, aby byla zajištěna jeho jedinečnost. Informace o vlastníkovi nového videa jsou nastaveny na uživatele, který provedl duplikaci.

Všechny související prvky s původním videem jsou duplikovány a přiřazeny novému uživateli do jeho knihovny. Soubory videa (samotné video a miniatura) jsou zkopírovány do složky uživatele v úložišti. Nová umístění souborů jsou aktualizována v databázi pro nově vytvořené video.

Po dokončení duplikace uživatel obdrží odpověď z aplikace, která obsahuje informace o nově vytvořeném videu.

3.4 Frontend

3.4.1 Validace uživatelských dat

Kontrolujeme vstupní data uživatele před odesláním na backend pomocí knihovny Zod v jazyce TypeScript. Zod nabízí elegantní způsob definice datových schémat a ověřování vstupů podle těchto schémat, která určují strukturu dat a pravidla jejich validace v různých typech dat.

Při pokusu o odeslání dat procházejí tato data procesem validace Zod, který kontroluje vstup od uživatele v souladu s konkrétním schématem. V případě chyb během validace jsou uživatelům zobrazeny informace o těchto chybách. Pokud proběhne validace úspěšně, jsou data připravena k odeslání na server pomocí axios, což zaručuje bezpečnou manipulaci s daty a minimalizuje riziko vzniku chyb [23, 24].

3.4.2 Správa dat

Na frontendu využíváme knihovnu Tanstack-Query pro efektivní správu dat a stavů aplikace. Tato knihovna automaticky ukládá výsledky dotazů do mezipaměti, což výrazně zvyšuje výkon a snižuje síťovou zátěž. Díky centralizovanému ukládání dat mezi komponentami reaguje aplikace rychleji a plynuleji na uživatelské interakce, čímž významně zlepšuje uživatelskou zkušenost.

V případě selhání mutace Tanstack-Query automaticky obnovuje dotazy, které byly touto změnou ovlivněny. Tímto způsobem udržuje konzistenci dat a zjednodušuje zpracování chyb. Tato funkce přispívá k bezproblémovému chodu aplikace i v nečekaných situacích [25, 26].

Tanstack-Query je klíčovým stavebním prvkem pro další funkcionality, které vylepšují uživatelskou interakci s aplikací, jako jsou optimistické aktualizace a infinity scroll. Díky těmto funkcím je uživatelský zážitek z používání aplikace ještě plynulejší a příjemnější.

3.4.3 Optimistické aktualizace

Optimistické aktualizace představují strategii často využívanou ve vývoji frontendu a distribuovaných systémů, která má za cíl zlepšit uživatelskou zkušenost a výkon aplikace. Princip spočívá v pokračování operace bez čekání na potvrzení ze serveru nebo jiných částí systému, předpokládající úspěšný výsledek operace. Tanstack-Query výrazně usnadňuje tento proces poskytnutím integrované podpory pro optimistické aktualizace.

Když uživatel provede akci vyžadující interakci se serverem, například odeslání formuláře, aplikace ihned spustí operaci a data z formuláře uloží do mezipaměti v Tanstack-Query, aniž by čekala na potvrzení.

Po obdržení odpovědi ze serveru nebo externí služby aplikace porovná aplikace lokální změny s aktuálním výsledkem akce. V případě úspěchu jsou lokální data aktualizována podle

potřeby. Pokud dojde k selhání, aplikace může vrátit lokální změny a informovat uživatele o výskytu chyby [27].

3.4.4 Infinite scroll

Infinite scroll je často využívaná technika ve vývoji webových stránek, která umožňuje kontinuální načítání obsahu bez nutnosti manuální navigace na další stránku nebo kliknutí na tlačítko "Načíst další". Tato metoda je běžně používána na stránkách nebo aplikacích, které zobrazují velké množství obsahu, jako jsou sociální sítě, galerie obrázků nebo seznamy novinek.

Když uživatel navštíví webovou stránku, je načtena počáteční sada obsahu, obvykle s předem definovaným počtem položek nebo příspěvků. Jak uživatel posouvá stránku dolů, posluchač událostí detekuje posun. Když uživatel dosáhne určitého bodu, spustí se událost pro načtení dalšího obsahu. Tato událost vyvolá AJAX požadavek na server, který požaduje další sadu obsahu. Server tento požadavek zpracuje a vrátí požadovaný obsah.

Po obdržení odpovědi kód v JavaScriptu analyzuje data a přidává je k existujícímu obsahu v DOM. Obvykle se to děje prostřednictvím dynamického vytvoření HTML prvků, které reprezentují nový obsah, a jejich následného přidání na konec kontejneru obsahu.

Tento proces pokračuje s každým posunem uživatele dolů na stránce, kdy se automaticky spouštějí další požadavky na načtení dalšího obsahu. Tímto způsobem je zajištěn plynulý uživatelský zážitek, jelikož eliminuje potřebu stránkování a umožňuje uživatelům nepřetržitě procházet obsahem bez přerušení [28].

4 Změny v průběhu práce

Původní plán maturitního projektu zahrnoval ruční schvalování publikovaného obsahu administrátory. Avšak během vývoje bylo rozhodnuto tuto funkci zrušit. Důkladné zvážení ukázalo, že ruční kontrola obsahu může vyvolávat několik problémů, které by mohly negativně ovlivnit efektivitu a uživatelskou spokojenost:

Jedním z hlavních důvodů je zpoždění, které může nastat v důsledku čekání na schválení administrátorem. Pokud administrátor není okamžitě k dispozici nebo pokud je fronta obsahu k posouzení velká, může to prodloužit dobu, než se nový obsah objeví na platformě. Tato zdržení mohou být frustrující pro uživatele, zejména pokud očekávají časově citlivé aktualizace.

Dalším problémem je administrativní zátěž spojená s ručním schvalováním. Proces vyžaduje pozornost administrátorů, kteří musí provádět kontrolu obsahu a rozhodovat o jeho publikaci. To může být zvláště obtížné na větších platformách s vysokým objemem obsahu. Administrátoři musí věnovat hodně času a úsilí, což může vést ke zpoždění v publikaci obsahu.

Navíc se mohou objevit problémy s konzistencí při stanovení kritérií pro schvalování obsahu. Různí administrátoři mohou mít odlišné názory nebo standardy, což může vést k nejednotnosti ve schvalování a zmatení uživatelů. Nepočítáme-li riziko lidských chyb spojených s ručním schvalováním.

Celkově lze tedy říci, že i když ruční kontrola může poskytnout vyšší míru kontroly nad obsahem, může to také přinést řadu problémů a výzev, které mohou negativně ovlivnit uživatelskou zkušenost a efektivitu provozu platformy. Proto bylo rozhodnuto zrušit ruční ověřování publikovaného obsahu.

5 Nevyřešené problémy

5.1 Google Drive

V původním plánu implementace projektu bylo uvažováno o ukládání veškerých uživatelských souborů na Google Drive. Tato strategie by umožnila uživatelům připojit svůj Google Drive a automaticky ukládat soubory do této cloudové služby. Hlavním důvodem pro tuto volbu byla snaha vyřešit problém omezeného úložiště, který momentálně řešíme ukládáním souborů lokálně přímo na serveru pomocí Laravel Local Storage.

Bohužel se ukázalo, že implementace této funkcionality není zcela jednoduchá. Problémy se objevily již při pokusu o nastavení ověření Google (Google Auth) na serveru. Kvůli architektuře serveru, který slouží pouze jako API, a kvůli politikám Google Cross bylo nemožné zobrazit registrační okno Google Auth přímo ve frontendu aplikace.

Přestože se nám podařilo úspěšně otevřít okno pro ověření Google (Google Authentikaci) přes frontend a úspěšně se přihlásit, narazili jsme na další překážku. Po úspěšném připojení

a autentizaci na Google Drive nebyl vrácen refresh token, který je klíčový pro automatické získání přístupového tokenu pro autentizaci Google Drive. Tímto by uživatel nemusel opakovaně zadávat přihlašovací údaje.

Absence refresh tokenu by vedla k nekomfortnímu uživatelskému zážitku, protože by uživatel musel při každé práci s Google Drive projít celým procesem autentizace znovu. S ohledem na tyto komplikace a časové nároky na jejich vyřešení bylo rozhodnuto odstoupit od původního plánu a použít alternativní metodu ukládání souborů pomocí Laravel Local Storage.

6 Závěr

Cílem tohoto projektu bylo vytvořit funkční webovou aplikaci pro stříhání a organizaci videí. Tento cíl byl úspěšně dosažen, neboť aplikace disponuje všemi klíčovými funkcemi, které efektivně pracují díky využití moderních technologií a frameworků, jako je Laravel pro backend a React pro frontend.

I přesto, že aplikace nemá všechny funkce specifikované v původním zadání a existují určitá omezení, projekt stále funguje dobře. Uživatelé mohou nahrávat videa, vytvářet jejich úryvky a prohledávat je pomocí různých filtrů. Funkční je také publikační systém umožňující prohlížení prací ostatních uživatelů a jejich případné stažení.

Projekt byl navržen s důrazem na moderní standardy výkonu, spolehlivosti a uživatelského zážitku. Poskytuje komplexní řešení pro správu a úpravu videí, které pokrývá různé potřeby uživatelů a poskytuje stabilní základnu pro budoucí rozšíření a vývoj.

Literatura

[1] ChatGPT. *Open AI*[online]. [cit. 2024-03-27]. Dostupné z: https://openai.com/chatgpt.

[2] ROBINSON, Brett. Podcast Statistics: Facts Trends from 2024 Data. *Wire Cip*[online]. [cit. 2024-03-20]. Dostupné z: https://wiredclip.com/podcast-statistics/.

[3] HAAHR, Tae a Matthew MCLEAN. Why Are Podcasts So Popular? 5 Reasons Why People Love Pods. *The Podcast Host*[online]. 2024 [cit. 2024-03-26]. Dostupné z: https://www.thepodcasthost.com/mindset/why-are-podcasts-so-popular/.

[4] Battle of the Shorts – TikTok vs Reels vs Shorts. *THE GRAYGENCY*[online], . [cit. 2024-03-26]. Dostupné z: https://thegraygency.com/battle-of-the-shorts-tiktok-vs-reels-vs-shorts/.

[5] BULLAS, Adam. THE BATTLE OF THE SHORT-FORM VIDEO PLATFORMS: YOUTUBE SHORTS VS TIKTOK VS INSTAGRAM REELS [UPDATED]. *Opace*[online]. 2023 [cit. 2024-03-26]. Dostupné z: https://www.opace.co.uk/blog/short-form-video-platforms-youtube-shorts-vs-tiktok-vs-instagram.

[6] PUGALIA, Rishabh. Unleash the Power of Short Form Videos: 25 Examples. *Content B*[online]. 2023 [cit. 2024-03-26]. Dostupné z: https://www.contentbeta.com/short-form-video/#:~:text=Benefits%20of%20Short%20Form%20Videos%201%201.%20Higher,6.%20Mobile-Friendly%20...%207%207.%20Staying%20Trendy%20.

[7] MILEVA, Geri. 10 Short-Form Video Trends to Watch Out for in 2024. *Influencer Marketing Hub*[online]. 2023 [cit. 2024-03-26]. Dostupné z: https://influencermarketinghub.com/short-form-video/.

[8] STELLINGWERF, Lili. 7 reasons your brand should use short form video content. *The Attention Agency*[online]. 2023 [cit. 2024-03-26]. Dostupné z: https://www.attentionagency.co.nz/blog/7-reasons-why-short-form-video.

[9] Over 6.5 Million Short-Form Podcast Videos created with Headliner! *The Headliner*[online], . 2023 [cit. 2024-03-26]. Dostupné z: https://www.headliner.app/blog/2023/04/26/6-5-million-short-form-podcast-videos-created-with-headliner/.

[10] React. *React*[online]. [cit. 2024-02-14]. Dostupné z: https://react.dev/.

[11] The PHP Framework for Web Artisans. *Laravel*[online], . [cit. 2024-02-14]. Dostupné z: https://laravel.com/.

[12] FFmpeg. *FFmpeg*[online], . [cit. 2024-02-14]. Dostupné z: https://ffmpeg.org/.

[13] Introducing Whisper. *OpenAI*[online]. [cit. 2024-02-14]. Dostupné z: https://openai.com/research/whisper.

[14] Validation. *Laravel*[online], . 2023 [cit. 2024-03-26]. Dostupné z: https://laravel.com/docs/10.x/validation.

[15] Laravel Sanctum. *Laravel*[online], . 2023 [cit. 2024-03-26]. Dostupné z: https://laravel.com/docs/10.x/sanctum.

[16] File Storage. *Laravel*[online], . 2023 [cit. 2024-03-26]. Dostupné z: https://laravel.com/docs/10.x/filesystem.

[17] FFmpeg Documentation. *FFmpeg*[online], . 2023 [cit. 2024-03-26]. Dostupné z: https://ffmpeg.org/ffmpeg.html.

[18] NELSON, Karen. How to Trim/Cut Video Using FFmpeg (Detailed Guide). *Video Solo*[online]. 2023 [cit. 2024-03-26]. Dostupné z: https://videosolo.net/ffmpeg-trim-video/.

[19] SHIVANANDHAN, Manish. How to Turn Audio to Text using OpenAI Whisper. *Free Code Camp*[online]. 2024 [cit. 2024-03-26]. Dostupné z: https://www.freecodecamp.org/news/how-to-turn-audio-to-text-using-openai-whisper/.

[20] Speech to text. *OpenAI*[online]. [cit. 2024-03-26]. Dostupné z: https://platform.openai.com/docs/guides/speech-to-text.

[21] MIDDLETON, Scott. REST API File Upload Best Practice. *Terem*[online]. 2023 [cit. 2024-03-26]. Dostupné z: https://terem.tech/api-file-upload-best-practice/.

[22] SPUNGIN, Steven. How to download files using axios. *Stack Overflow*[online]. 2022 [cit. 2024-03-26]. Dostupné z: https://stackoverflow.com/posts/53230807/revisions.

[23] Zod. *Zod.dev*[online]. [cit. 2024-03-18]. Dostupné z: https://zod.dev/.

[24] MOHAN, Mehul. Zod For Data Validation In TypeScript. *Code Damn*[online]. 2023 [cit. 2024-03-26]. Dostupné z: https://codedamn.com/news/javascript/zod-getting-started.

[25] TanStack Query v5. *TanStack*[online]. [cit. 2024-03-26]. Dostupné z: https://tanstack.com/query/latest.

[26] COLLINS, Lachlan. Overview. *TanStack*[online], . 2023 [cit. 2024-03-26]. Dostupné z: https://tanstack.com/query/v4/docs/framework/react/overview.

[27] COLLINS, Lachlan. Optimistic Updates. *TanStack*[online], . [cit. 2024-03-26]. Dostupné z: https://tanstack.com/query/latest/docs/framework/react/guides/optimistic-updates.

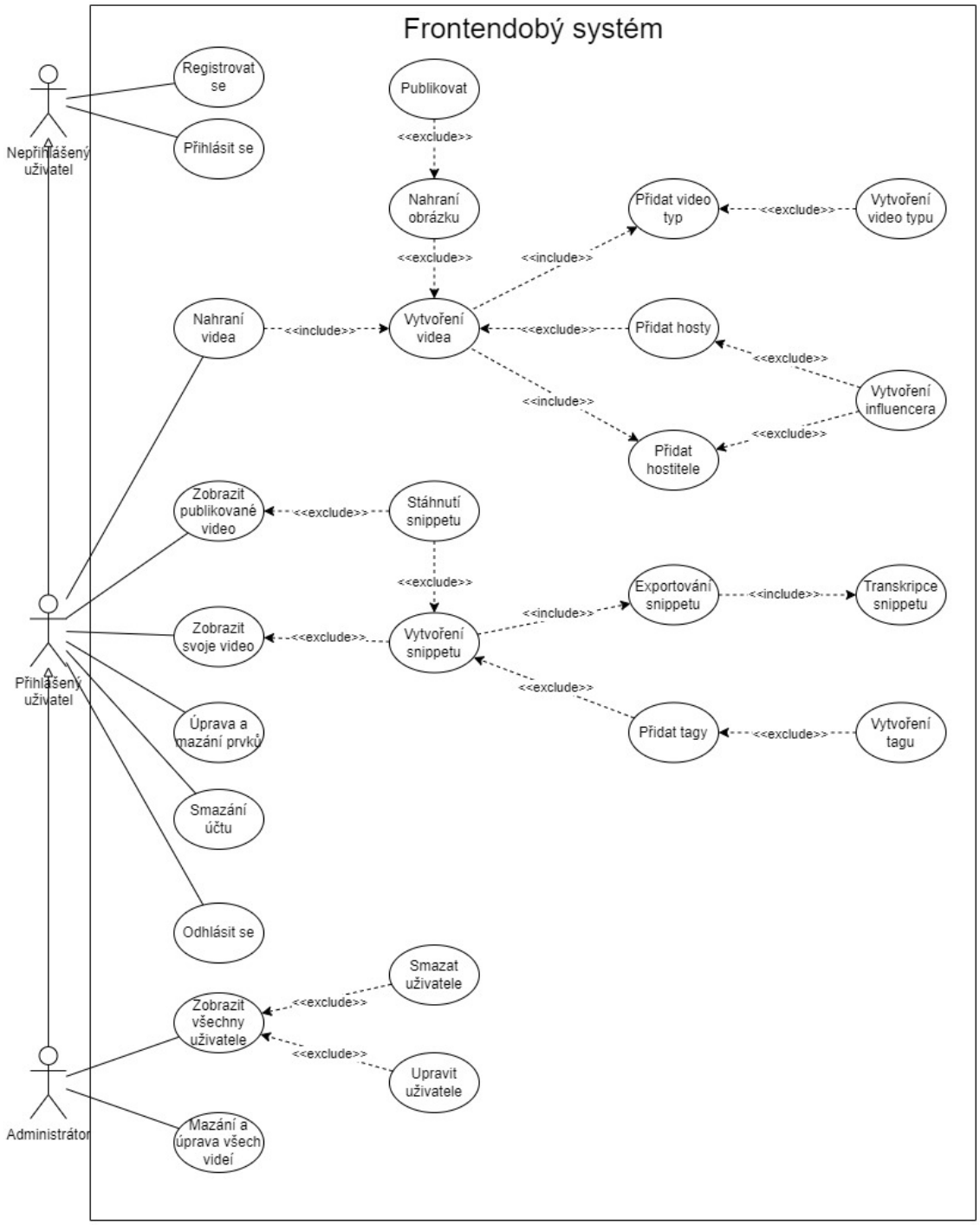
[28] COLLINS, Lachlan. Infinite Queries. *TanStack*[online], . [cit. 2024-03-26]. Dostupné z: https://tanstack.com/query/latest/docs/framework/react/guides/infinite-queries.

Seznam obrázků

1 Ukázka oranizace úryvků v Google Sheets. 8
2 Architektura aplikace. 11
3 Schéma databáze. 12
4 Základní stříhací process v FFmpegu. 15
5 Stříhací process FFmpegu použitím ”copy” parametru. Zdroj: [17] 15
6 Whisper sequence to sequence proces. Zdroj: [13] 16
7 Whisper multitasking format. Zdroj: [13] 17
8 Use case diagram 27

Seznam příloh

Use case diagram



Obrázek 8: Use case diagram