VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÁ APLIKACE PRO HODNOCENÍ KVALITY OBRAZU A VIDEA

WEB APP FOR IMAGE AND VIDEO QUALITY ASSESSMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. KAREL PÍČ
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. MARTIN ČADÍK, Ph.D.
SUPERVISOR

BRNO 2018

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání diplomové práce

22127

Student: **Píč Karel, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Webová aplikace pro hodnocení kvality obrazu a videa**
Web App for Image and Video Quality Assessment
Kategorie: Web
Zadání:

1. Seznamte se s problematikou metod pro automatické hodnocení kvality obrazu a videa a s moderními technologiemi pro vývoj webových aplikací.
2. Proveďte rešerši existujících systémů hodnocení kvality obrazu a videa na internetu a podobných aplikací.
3. Navrhněte a implementujte webovou aplikaci pro hodnocení kvality obrazu a videa (včetně HDR verzí). Při návrhu systému definujte požadavky na funkčnost a jednotlivé prvky prototypujte a testujte.
4. Se systémem experimentujte, posuďte jeho vlastnosti, proveďte uživatelský experiment a diskutujte možnosti budoucího vývoje.
5. Dosažené výsledky prezentujte formou videa, plakátu, článku, apod.

Literatura:

- http://cadik.posvete.cz/iqm/
- http://metrics.mpi-inf.mpg.de/

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Čadík Martin, doc. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Tato práce dala za vznik webové aplikaci pro hodnocení kvality obrazů a videí, a to především těm s vysokým dynamickým rozsahem (HDR). K hodnocení se používají metriky, jejichž výsledky budou sloužit pro vědecké účely a také pro přiblížení tohoto způsobu hodnocení mezi širší veřejnost. Ta bude mít k dispozici asistenta pro jednodušší pochopení celé aplikace. Aplikace má za cíl být co nejvíce autonomní a dynamická. Například, aby se mohly nové metriky přidávat z administračního prostředí a nebyla potřeba úprava na straně serveru. Administrátor bude moci nahrát metriku, lokální knihovny, spouštěcí skript a definovat parametry poskytující metrice potřebné údaje ze zpracovaného obrazu či videa. Na straně serveru bude vyžadována pouze instalace systémových aplikací a knihoven.

Abstract

This work will be a web application for image and video quality assessment, especially those with a high dynamic range (HDR). The metrics are used for the evaluation, the results of which will be used for scientific purposes. Broader public citizens will be acquainted with this issue. This app will have an assistant available for a simple understanding of the whole application. The application also tries to be autonomous and dynamic. For example, new metrics can be added from the administration environment and editing on the server side is not required. The administrator will be able to upload a metric, a local library, a startup script, and define the parameters. The parameters provide metrics necessary data from the processed image or video. Only installation of system applications and libraries will be required on the server side.

Klíčová slova

vysoký dynamický rozsah, obrazy, videa, metriky, hodnocení kvality, webová aplikace, MATLAB

Keywords

high dynamic range, images, videos, metrics, quality rating, web application, MATLAB

Citace

PÍČ, Karel. *Webová aplikace pro hodnocení kvality obrazu a videa.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Martin Čadík, Ph.D.

Webová aplikace pro hodnocení kvality obrazu a videa

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana docenta Ing. Martina Čadíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Karel Píč
15. května 2019

Poděkování

Touto cestou bych rád poděkoval svému vedoucímu práce, kterým je pan docent Ing. Martin Čadík, Ph.D., za cenné rady a připomínky.

Obsah

Kapitola 1

Úvod

V dnešní době existuje velké množství komunikačních kanálů, kde pracujeme rozdílnými způsoby s obrazy a videi. Je to například Facebook, kde sdílíme naše fotografie, nebo výzkumný informační systém, ve kterém nahráváme videa z naší práce. Ve většině případů používají odlišné komprimační nástroje, aby se snížila velikost na úložišti. Málokdy se nám stane, že se nahrávají surová data z fotoaparátu. Například samotný čip z fotoaparátu zachycující obraz používá komprimaci na formát, který dané zařízení podporuje a díky tomu umí fotografie zobrazit přímo v zařízení. S obrazy či videi můžeme pracovat v různých nástrojích a upravovat jejich barevný rozsah, velikost, datový typ atd. To má za následek ovlivnění jejich kvality. Při vývoji komprimačních nástrojů a aplikací, které manipulují s obrazy či videi je zapotřebí vybrat ty, které v nejmenší možné míře ovlivní jejich kvalitu a v případě komprimačních nástrojů mají i největší účinnost. Pro hodnocení kvality, tedy rozdílů dvou snímků, se používají algoritmy označovýny jako metriky, které vznikají ve vědeckých organizacích. Při vývoji je potřeba metriky testovat na různé vstupy a získávat z nich výstupy. Ty se porovnávají vůči výstupům jiných metrik. Bylo by vhodné, kdyby existoval nástroj, který vyhodnotí jednotlivé vstupy na různých metrikách a výsledky přehledně zobrazí tak, aby se mohly adekvátně vyhodnotit. Takový to nástroj bude poskytovat právě naše aplikace.

Tématem této diplomové práce je webová aplikace, která bude umožňovat uživatelům nahrávat obrazy nebo videa. Vstupní data se zpracují a aplikují se na ně uživatelem vybrané metriky. Vyhodnocení kvality uživatelova vstupu lze upřesnit parametry, které jsou k dispozici u jednotlivých metrik. Tyto výsledky mohou být prezentovány textově nebo graficky. V případě grafického výstupu se jedná o jeden nebo více obrazů a videí. Důležitou součástí této aplikace bude i administrační prostředí, kde administrátor bude moci nahrávat jednotlivé metriky, konfigurační soubory a další potřebné součásti pro správný běh metrik. Konfigurační soubory budou sloužit jako předpis, jak správně spustit vybranou metriku, správně definovat parametry a získat informace o obrazech či videích poskytnuté aplikací, jako je rozlišení, barevné schéma či datový typ. Díky možnosti nahrávání metrik a úprav z administračního prostředí, nebude muset administrátor kontaktovat správce serveru pro úpravu zdrojových kódů, protože si zvládne většinu provést sám z aplikace. Na správci serveru už zůstane pouze nasazení aplikace a nainstalování základních grafických programů a balíčků.

Největším problémem k vyřešení v této práci je různorodost metrik, které mohou být nahrány. Každá metrika se spouští odlišným způsobem, s jinými parametry a hodnotami a má jiné závislosti. Bude také potřeba vyřešit jakým způsobem prezentovat výsledky metrik. Protože pokud metrika po spuštění v aplikaci vytvoří dva obrázky nebo jedno video a do

textového souboru zapíše hodnoty nebo tabulku, je zapotřebí, aby si s tím aplikace poradila a přehledným způsobem tato data prezentovala uživateli. Dalším problémem, na který je potřeba se zaměřit je bezpečnost. Kromě zabezpečení administrace a spojení se serverem je v dnešní době nutné zabezpečit proti odcizení i obrazy a videa, které uživatel nahraje do aplikace, a také výsledky metrik, které vzniknou z analýzy nahraných dat.

Tato práce je perspektivní hlavně pro výzkumné organizace, které se zabývají analýzou obrazů a videí. Aplikace se obecně zaměří na dvě cílové skupiny. První je širší veřejnost, které chceme představit danou problematiku za použití novodobých technologií a i nasměrovat jejich zvědavost tímto směrem. Druhá skupina je vědeckého charakteru, která bude využívat možnost ověřit výstupy jednotlivých metrik vůči vlastním metrikám, nebo například pokud by chtěli zdarma využít výsledky pro svojí vědeckou práci. Pokud by někoho tato práce nadchla natolik, že by se chtěl zapojit do správy, je možné mu přidělit oprávnění, aby mohl nahrávat vlastní metriky a tím se prezentoval veřejnosti.

V následující kapitole nalezneme dosavadní řešení (2), které má být nahrazeno. Vysvětlíme si, jaké měla předchozí aplikace nedostatky, co jsme změnili a co doplnili. V kapitole 3 se dozvíme s čím se pracuje a co bylo potřeba, aby se s prací mohlo vůbec začít. Kapitola *Návrh webové aplikace* (4) pojednává o celkové architektonické koncepci aplikace a co se změnilo oproti předchozímu řešení. Součástí této kapitoly je kromě návrhu databáze i schéma logiky aplikace a to na jaké části jsme jí rozdělili, jak spolu tyto části souvisí a s jakými daty pracují. V předposlední kapitole *Implementace* (5) se můžeme dočíst, jak byly některé problematické celky vyřešeny, jak aplikace komunikuje s uživatelem a jak se serverem. V této kapitole nalezneme i způsob zpracování obrazu a videa a jaké údaje se z nich získávají. Poslední 6 kapitola pod názvem *Testování*, je zaměřena jak na aplikační část, kterou testují uživatelé, tak na programovou část, kde frontendová i serverová aplikace má zavedeny automatické testy, aby se mohla aplikace postupně rozšiřovat a vývoj nebyl brzděn repetitivním ověřováním.

Kapitola 2

Dosavadní řešení

Obdobné řešení této webové aplikace neexistuje, nebo nebylo nalezeno, kromě webu[1] pana docenta Ing. Čadíka, který byl vytvořen pod záštitou instituce MPII[2]. Tento web je sestaven pro konkrétních pět obrazových metrik a jednu video-metriku. Další možné aplikace už nejsou on-line, ale jelikož metriky jsou převážně matematické operace nad grafickými daty, dobře nám poslouží pro získávání výstupů i matematické programy jako je MATLAB. Ten si poradí s analýzami a vyhodnocováním kvality obrazů i videa, ale není dostupný v bezplatné verzi. V naší aplikaci je MATLAB využit na straně serveru, kde některé metriky jsou spouštěny nad jeho prostředím a výsledky poté zpracovány a prezentovány uživateli.

Specializované programy, které se zaměřují na tuto problematiku nejsou příliš rozšířené, ale například aplikace VQMT[3], kterou si je možné bezplatně stáhnout, podporuje analýzu a vyhodnocování videa pomocí přibližně deseti metrik a poskytuje kvalitní rozhraní pro zobrazení výsledků.

Hlavní příčinou proč neexistuje více on-line řešení, je různorodost metrik v jejich požadavcích. Můžeme mít metriku napsanou v jazyce Pythonu a používat naprosto jiné parametry a balíčky, než metriky napsané v jazyce C, a to nebereme v potaz různé verze téhož programovacího jazyku. Dále je to výše zmíněný problém, že každá metrika má různý datový výstup a ten není vždy je jen jeden. Metriky často potřebují ke svému běhu i velký počet externích knihoven, proto není jednoduché vše zahrnout do jedné aplikace. O některých knihovnách si ještě řekneme v kapitole 5.3 o vývojovém prostředí.

2.1 Nedostatky stávající aplikace

Pokud se zaměříme přímo na webovou aplikaci, je dosavadní řešení limitující ve statickém provedení, které neumožňuje dynamicky přidávat další metriky bez zásahu na serveru a úpravy zdrojového kódů. Dále je rozdělena aplikační logika pro videa a obrazy, ale jsou metriky, které akceptují oba dva datové vstupy. Pokud by se chtěl student nebo vědec odkázat na konkrétní výsledek své metriky, jedinou jeho možností je stáhnout si obrázek výstupu. Což v dnešní moderní době není úplně uživatelsky přívětivé. Další omezení se naskýtá v nahrávání souborů, protože u analyzování obrazu nelze nahrát více než jeden zdrojový a referenční obraz. Aplikace také neumožňuje kombinovat metriky a to z několika důvodů. Jedním z nich je například použití obrazové metriky na video, které je potřeba

[1] Online vyhodnocování kvality obrazu nebo videa - http://metrics.mpi-inf.mpg.de/

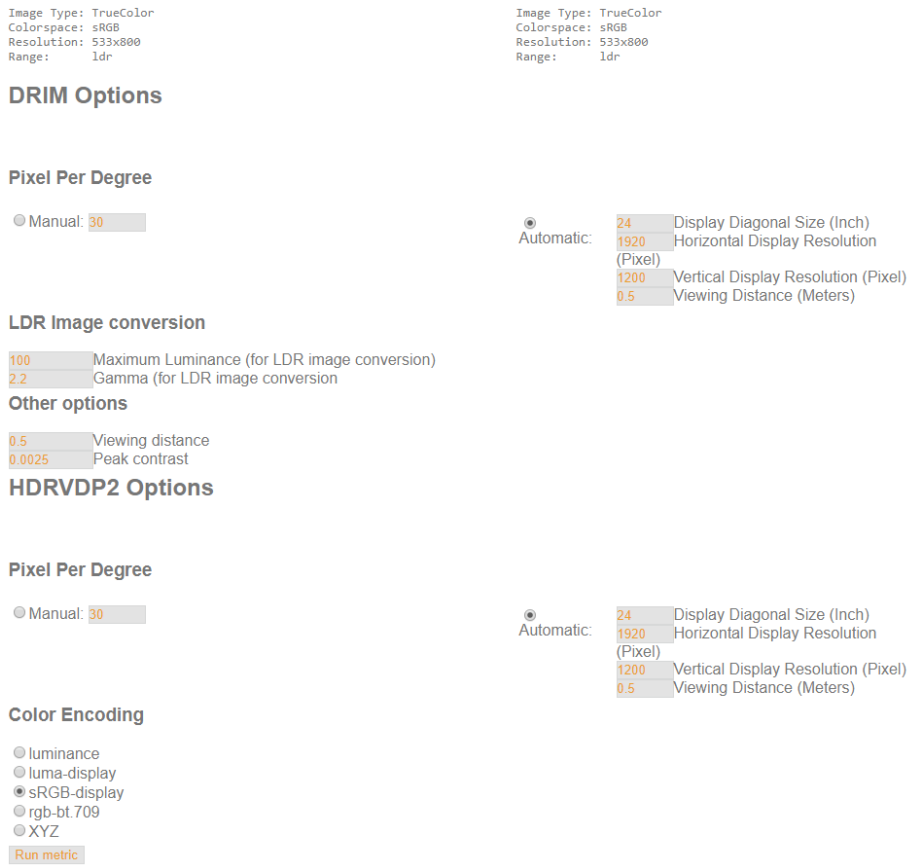
[2] MPII - Max-Planck-Institut für Informatik - https://www.mpi-inf.mpg.de/home/

[3] VQMT - Video Quality Measurement Tool, aplikace k dispozici na http://www.compression.ru/video/quality_measure/video_measurement_tool.html

nejdříve rozložit na jednotlivé snímky, aplikovat na ně metriku a poté je opět složit ve video.

Současná webová aplikace nemá žádné administrační rozhraní, kde by uživatel mohl dynamicky přidávat nový či upravovat stávající obsah. Není schopna podávat žádné statistické údaje správci aplikace, ani umožnit nahrát celé balíčky souborů, které by webová aplikace zpracovala a dokázala poskytnout jejich funkcionalitu svým uživatelům.

Na následujícím obrázku 2.1 je vidět konfigurace metrik po nahrání zdrojového a referenčního obrazu. Pro běžné uživatele by jednotlivé hodnoty mohly být nesrozumitelné. Chybí zde vysvětlení jednotlivých parametrů, abychom věděli, co přibližně způsobí s výsledky a na co je jaký parametr vhodnější. Například, zda definuje počet snímků, které se mají použít, nebo zda nastavuje světelnost.



Obrázek 2.1: Ukázka současné konfigurace k jednotlivým metrikám. Převzato ze stávající verze pro analýzu kvality obrazu.

2.2 Závěr ze stávajících aplikací

Námi implementovaná aplikace sjednotí rozdělené zpracování videa a obrazu. Doplní konfigurovatelnost parametrů z administrace a zvýší informovanost směrem k uživateli formou nápověd u jednotlivých parametrů. Umožní nahrávat sekvenci obrazů a definovat, jak budou obrazy zpracovány. V případě vyhodnocování jednotlivých metrik, může uživatel sledovat přímý přenos zpracování, jako je například postupné načítání výsledků u analyzování jed-

notlivých obrazů nahrané sekvence. Dokumentační část je doplněna o příklady analýz přímo z této aplikace, které může administrátor nastavit dle své potřeby. Co se týče bezpečnosti, tak zde byly zabezpečené proti odcizení pouze video soubory, které byly chráněny heslem a to přicházelo emailem po dokončení analýzy. V naší aplikaci jsou zabezpečené proti odcizení veškeré uživatelské datové vstupy a také výstupy z metrik jak pro video metriky, tak pro obrazové metriky. Pro přístup k údajům je zapotřebí heš definující analýzu a bezpečnostní token, který je jedinečný a dostatečně dlouhý, aby url adresa nemohla být uhodnuta. Přístup na analýzu je odeslán uživateli emailem vždy po dokončení analýzy.

Jelikož je stávajících aplikací řešící stejnou problematiku málo, bylo nutné investovat více času do architektonického a grafického návrhu. Rozšiřování a získávání nových nápadů přicházelo i během vývoje a při uživatelském testování, kdy si uživatelé sami říkali, co jim schází a čemu nerozumí. Jako příklad poslouží doplnění jazykových mutací, možnost stahování výsledků konkrétních souborů nebo celé analýzy, doplnění nápověd u psaní konfiguračních souborů nebo informování uživatele o tom, co se děje v průběhu analýzy.

Kapitola 3

Hodnocení kvality obrazu a videa

Hlavními vstupy pro tuto webovou aplikaci jsou obrazy a videa s různou velikostí a různého datového typu. Důvodem je rozmanitost metrik, které mohou vyžadovat specifický datový typ obrazu. Dalším důvodem je použití HDR (3.2) obrazu, který může mít několik desítek megabajtů a obsahuje informace, které do běžného (LDR[1]) obrazu nelze uložit. V následujících bodech popíšeme problematiku HDR obrazů a automatických metrik pro měření jejich kvality.

3.1 Metriky pro měření kvality

Metrika je algoritmus, který počítá rozdíly mezi dvěma obrazy, jako je například šum nebo rozmazanost. V mnoha případech je rozdíl totožný s tím, jak by to viděl uživatel. Máme dvě hlavní skupiny metrik pro analýzu kvality. Jsou to metriky pro hodnocení kvality obrazu a metriky pro hodnocení kvality videa. Právě těmto dvěma skupinám budou věnovány následující dvě podkapitoly.

3.1.1 Měření kvality obrazu (IQM)

Obrazové metriky potřebují na vstupu dva obrazy, zdrojový a referenční. Výjimkou jsou obrazové metriky bez reference, které na vstupu vyžadují pouze jeden obraz. Obrazové metriky lze aplikovat i na videa, protože to je pouze sekvence snímků. Nemůžeme tedy metriku použít přímo, ale postupnou aplikací na jednotlivé snímky videa lze IQM aplikovat. V následujících bodech uvedeme příklady obrazových metrik. U každé uvedeme, co je její podstatou, jaký se používá výpočet, silné a slabé stránky metody.

- PSNR – Peak signal-to-noise ratio [2]

  PSNR (Špičkový poměr signálu k šumu) je jedna z nejjednodušších a nejznámějších metrik. Vyjadřuje poměr mezi maximální možnou energií signálu a energií poškozujícího šumu, který snižuje kvalitu. PSNR se vyjadřuje v logaritmickém tvaru (vzorec 3.2), protože mnoho signálů má velmi vysoké HDR - vysoký dynamický rozsah. Nejdříve je zapotřebí definovat střední kvadratickou chybu MSE[2] pro dva obrazy f a g o rozměrech $m * n$ (vzorec 3.1).

  Dále potřebujeme definovat MAX_f, což je maximální hodnota pixelu v obrazu a to 255 pro 8-bitový obraz. U obrazů s barevným rozsahem RGB je MSE suma všech tří

[1] LDR - Low Dynamic Range - Nízký dynamický rozsah - opak k HDR
[2] MSE - mean squared error - https://en.wikipedia.org/wiki/Mean_squared_error

položek vydělena třemi. Čím kvalitnější obraz, tím vyšší hodnoty PSNR dostaneme. Pro shodné obrazy je MSE nula a tedy PSNR nemůžeme definovat. Tato metrika má špatný výkon vůči ostatním metrikám kvality.

$$MSE = \frac{1}{mn} \sum_{0}^{m-1} \sum_{0}^{n-1} \parallel f(i,j) - g(i,j) \parallel^2 \tag{3.1}$$

$$PSNR = 20log_{10} \left(\frac{MAX_f}{\sqrt{MSE}} \right) \tag{3.2}$$

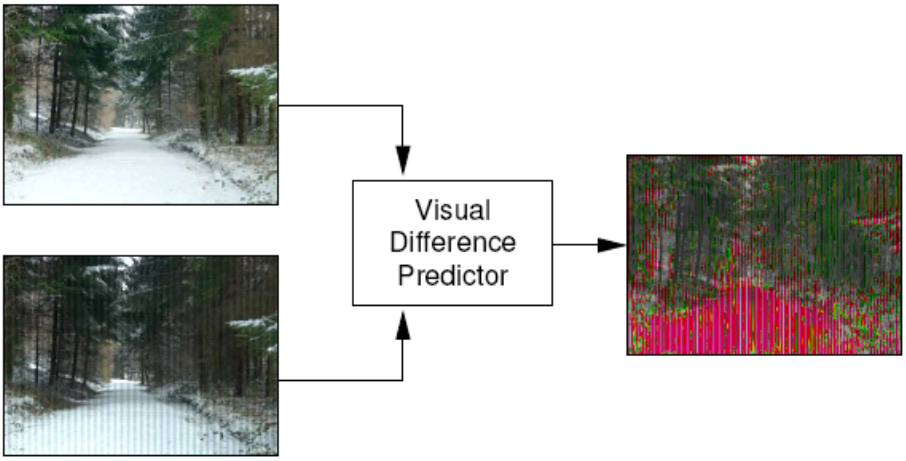
- DRIM – Dynamic Range Independent Metric [4]

  DRIM (Metrika nezávislá na dynamickém rozsahu) využívá rozmanitost zobrazovacích technologií a zavedení snímků s vysokým dynamickým rozsahem. Na základě toho jsou porovnávány obrazy s odlišným dynamickým rozsahem. Starší metriky nejsou schopny tohoto hodnocení, protože předpokládají, že referenční a zdrojový obraz mají stejný dynamický rozsah. DRIM využívá model lidského vizuálního systému a jeho hlavní myšlenka spočívá v nové definici viditelného zkreslení založeného na detekci a klasifikaci viditelných změn struktury obrazu. Příklad vyhodnocení metrikou DRIM můžeme vidět na obrázku 3.1.

Obrázek 3.1: Hodnocení kvality obrazu s nízkým dynamickým rozsahem (vlevo), generovaného mapováním tónů referenčního obrazu s vysokým dynamickým rozsahem (střed) pomocí operátoru mapování tónů. Vpravo je vidět mapa zkreslení vzniklá po použití metriky DRIM, kde zelenou barvou je ztráta viditelného kontrastu a červenou je vyznačeno obrácení kontrastu. Převzato z MPII.

- HDR-VDP – Hight Dynamic Range Visual Difference Predictor [6]

  HDR-VDP (Prediktor vizuálních rozdílů obrazu s dynamickým rozsahem) porovnává zdrojový a referenční obraz a předpovídá viditelnost a kvalitu. Viditelnost je vlastnost vyjadřující pravděpodobnost s jakou jsou pro běžného pozorovatele viditelné rozdíly mezi snímky. Příklad vyhodnocení pravděpodobnosti rozdílů dvou obrazů je na obrázku 3.2. Kvalita nám znázorňuje míru degradace kvality s ohledem na referenční obraz. Metriku lze využít například při ověřování věrohodnosti obrazu po kompresi nebo zda objekty na obrazu jsou dostatečně viditelné. Metrik s tímto záměrem existuje několik, ale HDR-VDP má hned několik výhod. Mezi první se řadí práce s plným rozsahem hodnot jasu a to hlavně díky podpoře HDR obrazu. Další výhodou je volně šiřitelný zdrojový kód a pro kalibraci a testování bylo využito rozsáhlé množství obrazových souborů. Nejnovější implementace tohoto přístupu má označení HDR-VDP2.



Obrázek 3.2: Zpracování referenčního (vlevo nahoře) a zkresleného (vlevo dole) obrazu metrikou HDR-VDP, tím vznikla obrazová mapa, která detekuje pravděpodobnost rozdílů. Červená barva znázorňuje vysokou pravděpodobnost, zelená nízkou. Převzato z MPII.

3.1.2 Měření kvality videa (VQM)

VQM metriky jsou určeny pro automatické hodnocení kvality video sekvencí. Při zpracování jednotlivých snímků videa mají k dispozici i hodnoty z předchozích snímků, díky tomu mohou reagovat například na změny v osvětlení. U měření kvality videa je problém ve vysokém paměťovém vytížení, jelikož videa mají několikanásobně větší velikost než obrazy a při práci s nimi (například u rozkladu na jednotlivé snímky) dochází i k velkému zatížení procesoru serveru. Z toho důvodu bude tato aplikace informovat uživatele o dokončeném vyhodnocení emailem, aby nemusel čekat minuty až hodiny u zařízení.

Naše aplikace umožní uživatelům spouštět i VQM na obrazy a to díky tomu, že umožňujeme uživateli nahrát sekvenci snímků. Aplikace snímky seřadí dle určení uživatele a vytvoří ze snímků videa. Na videa se aplikuje metrika a její výstup aplikace nabídne uživateli ke stažení v komprimovaném souboru.

3.2 Vysoký dynamický rozsah

V dnešní době se dostává do popředí stále více pojem HDR, který označuje vysoký dynamický rozsah. Jinými slovy je to rozsah mezi nejsvětlejším a nejtmavším místem ve scéně. Například datový typ obrázku JPEG je v rozsahu 255:1, to odpovídá velikosti 1 byte na jeden bod obrazu. Takto malý datový rozsah označujeme LDR jakožto nízký dynamický rozsah. Díky tomu máme sice věrohodné obrazy s minimální datovou velikostí, ale přicházíme o velké množství údajů. V minulosti měly monitory ještě menší barevný rozsah a ztráta kvality byla zanedbatelná. Například CRT monitory akceptují světlo maximálně v rozsahu 100:1. V dnešní době, kdy zobrazovací zařízení prošla značným vývojem, se do popředí dostává právě HDR, které má dostatečný rozsah pro uchování všech dat. Používáním HDR roste datová náročnost na média, ovšem při současných technologiích to lze zanedbat, protože paměťové disky se zmenšují a jejich kapacita naopak roste. Obrázek 3.3 nám zobrazuje dvě totožné scény s různým světelným rozsahem, vlevo můžeme vidět LDR a vpravo HDR.([7])

Obrázek 3.3: Vlevo máme LDR obraz, který nezachycuje kompletní rozsah světla. Vpravo je HDR obraz, který zachycuje rozsah světla v celé scéně. Převzato z knihy High Dynamic Range Imagine [7].

Běžný uživatel může přijít do kontaktu s pojmem HDR u fotoaparátu, který sice nedokáže pojmout celý rozsah světla, ale dokáže pořídit několik fotografií s různou světelnou citlivostí a posléze tyto fotografie zkombinovat v jeden HDR obraz. Výsledný obraz obsahuje živější barvy a ve většině případů je i realističtější. Výsledek si můžeme ověřit na následujícím obrázku 3.4, kde je vidět 9 vstupních obrázků a jeden výstupní. Můžeme si všimnout kombinace, kdy na prvním vstupním obrázku je vidět vitráž v okně, ale na posledním obrázku je tento detail příliš přesvícený, díky čemu zase můžeme vidět vnitřní prostor místnosti. Po kombinaci devíti obrázků nám vznikla fotografie, kde můžeme vidět celou místnost i s vitráží.

Obrázek 3.4: Na obrázku je vidět proces, kdy z 9 vstupních obrazů LDR s různou světelnou citlivostí vznikne jeden obraz HDR. Převzato z knihy High Dynamic Range Imagine [7].

Kapitola 4

Návrh webové aplikace

V následujících kapitolách si rozebereme nejdříve samotné požadavky na aplikaci, které budeme následně analyzovat. Dále si představíme případ užití webové aplikace a také jednotlivé modely návrhu aplikace. Na grafických modelech si popíšeme serverovou část, frontendovou část a celkovou funkčnost systému a na datových modelech si popíšeme jakým způsobem pracujeme s daty.

4.1 Požadavky na webovou aplikaci

Z analýzy problematiky a jednotlivých konzultací se zadavatelem vyplynuly níže popsané body, které jsou pro vývoj aplikace stěžejní.

- přívětivé uživatelské prostředí
- stránky jednotlivých metrik s ukázkami
- administrace, kde bude možné nahrávat metriky
- uživatelské účty, kde bude možné uchovávat výsledky
- rozesílání emailů, které bude informovat o výsledku hodnocení kvality
- úložiště na klientské straně, pro uložení jednotlivých mezikroků a statistik
- službu, která bude na straně serveru plánovat a spouštět jednotlivé procesy
- informace ze serveru, které budou ihned poskytovány uživateli
- možnost běžet na linuxovém serveru
- podpora jazyků, ve kterých jsou psané metriky - C (binární soubory), Python, MATLAB

Detailněji budou jednotlivé položky probrány v kapitole *Implementace* (5).

4.1.1 Návrh aplikace na modelu klient-server

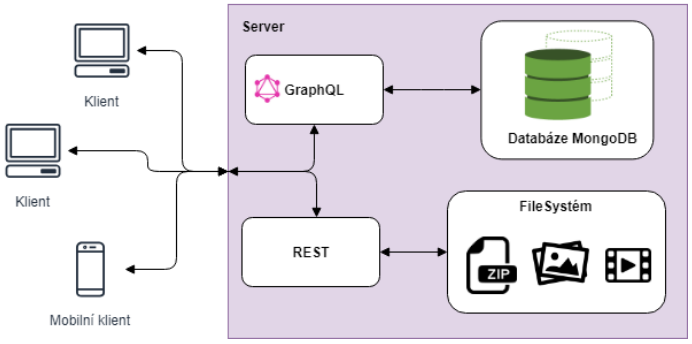
Model klient-server je základním konceptem, na kterém je založena webové aplikaci (klient) a API (server). Komunikují spolu jen v případě, že klient žádá o nějakou službu. Nejčastěji to bude požadavek na informace uložené v databázi, který webová aplikace zpracuje a

zobrazí uživateli. Server se skládá ze dvou částí. REST[1], které používáme pro poskytování souborů a GraphQL, které slouží pro získávání dat. Podrobněji se o GraphQL dočteme později v kapitole 5.2, zde nám stačí, že je to pouze nástroj obalující serverovou část sloužící k dotazování se na data serveru.

REST API je architektura definující jednoduchý přístup k datům serveru a to čtyřmi základními metodami, které můžeme znát pod pojmem CRUD (Create, Read, Update a Delete). Těmto metodám odpovídají následující implementace metod HTTP protokolu. GET nám slouží pro získání dat ze serveru. POST pro vytvoření datového záznamu na server. PUT pro změnu a DELETE pro mazání. REST využijeme pro poskytování obrazů a videí do webové stránky a pro stahování výsledků ve formátu ZIP [9]. GraphQL pracuje na jiném principu a na vše používáme pouze http metodu POST. Pomocí GraphQL budeme přistupovat přímo do databáze a získávat potřebná data.

Výhodou tohoto modelu a důvodem, proč jsme se pro něj rozhodli, je oddělení frontendové části (část, která pouze prezentuje získané údaje uživateli) a serverové části (část, která poskytuje data frontendovým aplikacím). Například, kdyby chtěl někdo jiný využívat naše data ve vlastní aplikaci, tak právě tento model takovýto přístup umožňuje.

Nevýhodou tohoto modelu je větší náročnost implementace, kdy je nutné udržovat dokumentaci API aplikace, aby zájemci věděli, jak mohou k API přistupovat. To přináší větší nároky na zabezpečení, kdy s API pracuje více aplikací, které s daty mohou nakládat odlišně. Z tohoto důvodu bylo využito právě GraphQL, které umožňuje definovat datový přístup na server. Pro generování dokumentace REST API jsme použili nástroj Swagger[2] a pro GraphQL jsme použili volně dostupný nástroj graphdoc. Dokumentace je dostupná na přiloženém CD A.



Obrázek 4.1: Obecné schéma komunikace naší aplikace na klientské straně s API na serverové straně.

4.2 Analýza

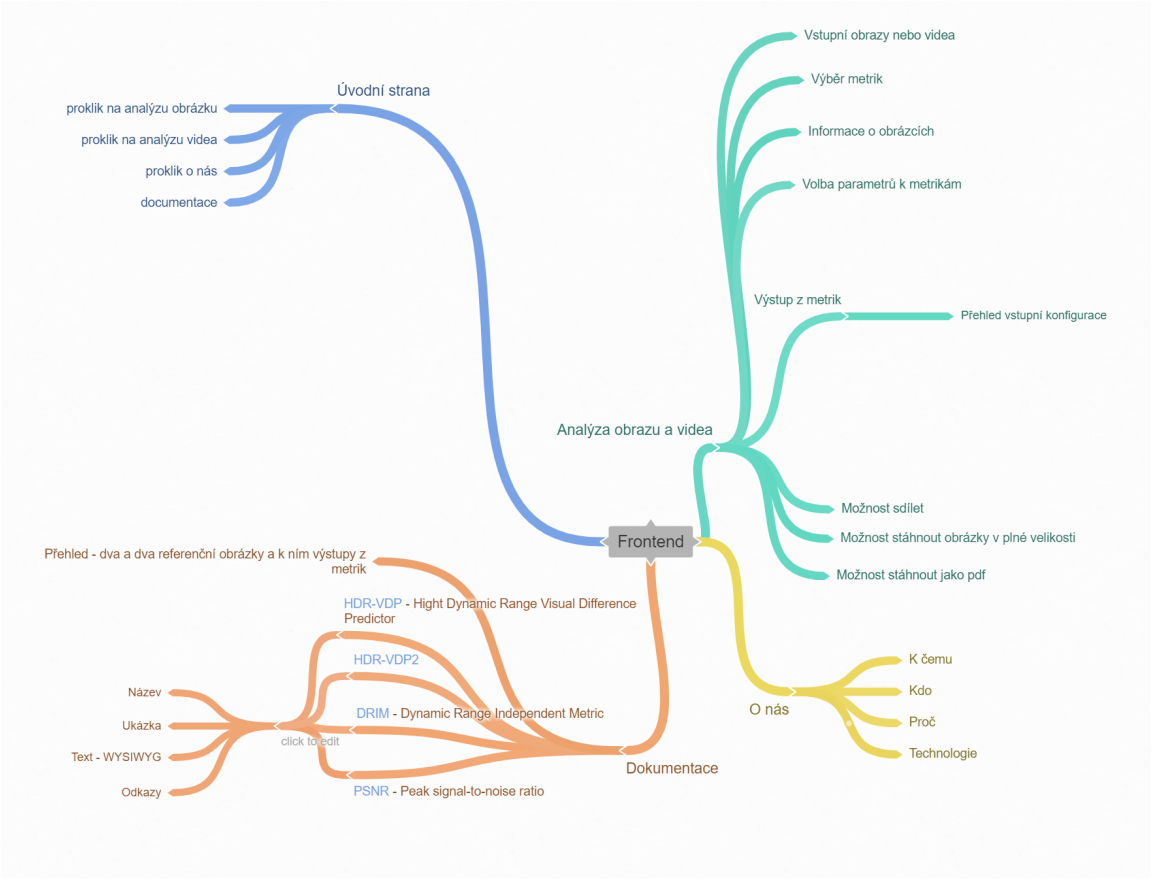
Pro utřídění myšlenek a názorů jsme použili myšlenkovou mapu, ve které nalezneme celkový přehled hlavních bodů celé aplikace. Je rozdělena na dvě části, uživatelská část (4.2) a administrační část (4.3).

Myšlenková mapa uživatelské části obsahuje čtyři větve. Hlavní strana sloužící jako rozcestník bude muset upoutat uživatelovu pozornost a na první pohled musí být jasné, co je účelem této aplikace. Na myšlenkové mapě je taktéž dokumentační část, kde má každá

[1] REST - Representational State Transfer - https://restfulapi.net/
[2] Swagger - open-source nástroj pro práci s API - https://swagger.io/

metrika vlastní stránku s názvem, popisem, referenčními daty a verzí metriky. Dále je vidět analýza obrazu a videa, kde bude moci uživatel nahrát svá data, vybrat metriky, nastavit konfiguraci a spustit analýzu. Tato část je totožná pro videa i pro obrazy. V poslední části je blok *O nás*, kde se uživatel dozví, proč tato aplikace vznikla, kdo jí vytvořil a bude zde kontakt pro případné dotazy uživatelům.

Díky této mapě jsme došli k závěru, že by se mohly následující dvě části, jak nahrávání obrazu, tak nahrávání videa, sloučit do jedné stránky. V původní aplikaci byly tyto části oddělené. Stejný princip využijeme i u možnosti nabízených metrik, protože IQM a VQM můžeme kombinovat jak pro videa tak pro sekvenci obrazů.

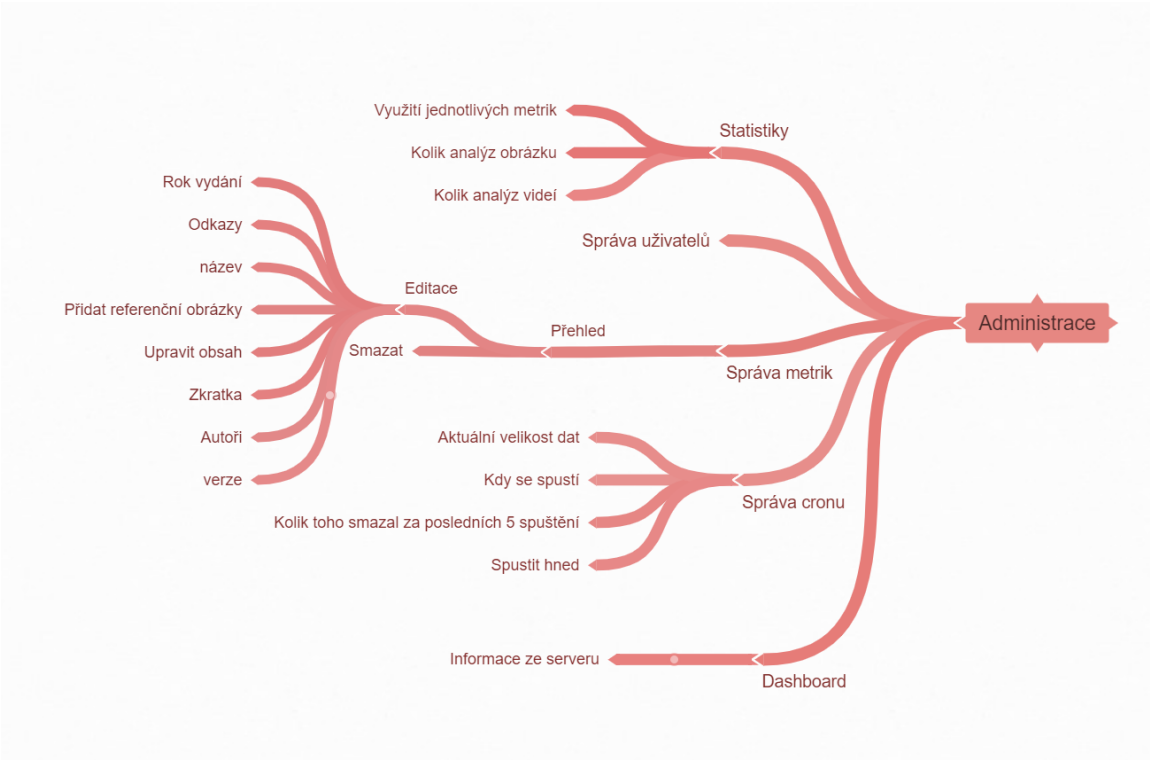


Obrázek 4.2: Myšlenková mapa uživatelské části aplikace. Hlavní strana (modře) bude rozcestník pro uživatele. Další oddíl je zpracování obrazu a videa (zeleně). Na stránce O nás (žlutě) bude mít uživatel přehled, kdo stojí za touto aplikací a na koho se obrátit v případě dotazů. Dokumentace (oranžově) bude informovat uživatele o tom, co se vlastně děje s jeho obrazy nebo videi.

Myšlenková mapa administrační části obsahuje pět hlavních bloků. Statistiky pro přehled, která metrika je nejvíce využívána, zda se používá více metrik pro obrazy nebo pro videa. Dalším blokem je správa uživatelů, aby měl administrátor přehled, kdo je jak aktivní, případně neaktivní nebo aby mohl změnit uživatelova práva v rámci aplikace. Ve správě metrik bude mít administrátor přehled o všech metrikách, bude moci přidat novou metriku a také bude mít možnost doplnit dokumentaci k stávajícím metrikám. V předposlední části

bude mít administrátor přístup k řízení cronu[3], kde uvidí logy z doposud zpracovaných dat, uvidí jeho příští naplánované spuštění a případně ho bude moci hned spustit. Nakonec zde bude přehled, kde administrátor uvidí informace o systému, jak je využíván procesor, nebo jak velká kapacita disku je zaplněna.

Přehled jsme při implementaci upravili a na místo umístění informací na samostatnou stránku jsme z něho vybrali jen nejdůležitější informace (vytížení procesoru, volná kapacita disku) a umístili je do hlavičky tak, aby měl administrátor neustálý přehled o vytížení serveru.



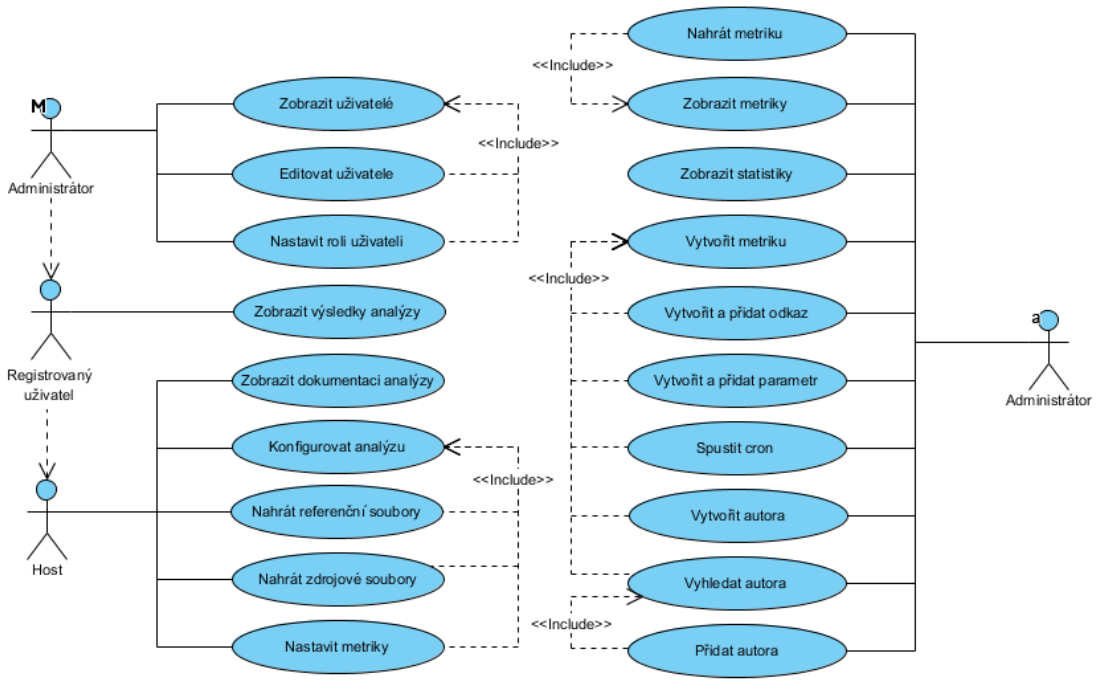
Obrázek 4.3: Myšlenková mapa administrační části. Statistiky s přehledem o využití metrik, správa uživatelů s přehledem o aktivitě jednotlivých uživatelů, správa metrik sloužící pro přidávání metrik a editaci jejich dokumentace. Správa cronu slouží ke snížení množství nutných zásahů na serveru. Přehled s informacemi o stavu systému.

4.3 Případ užití

Pro webovou aplikaci jsme sestrojili případ užití 4.4 za účelem definování možností a oprávnění uživatelů, kteří budou s aplikací pracovat. Aplikace má tři základní role uživatelů. Běžný příchozí uživatel, který má možnost vyzkoušet aplikaci nahráním obrazů nebo prozkoumání dokumentace metrik, či sdílení výsledků. Registrovaný uživatel má výhodu v přehledu svých analýz nebo podílení se na rozvoji aplikace například poskytnutím vlastních metrik. Role administrátor, má na starost administrační rozhraní a tedy správu metrik. S tím souvisí nahrávání balíčků ne kterých jsou metriky závislé, vytváření spouštěcích

[3]Cron - služba běžící na serveru, která vykonává dle časového plánu různé instrukce

skriptů, či definování parametrů, se kterými se metrika spouští. Administrátor má možnost ke každé metrice vytvořit dokumentaci, ke které může doplnit autory nebo odkazy. Administrátor má taktéž přehled o samotném serveru, tedy o zatížení procesoru a vytížení diskové kapacity. S místem na disku může manipulovat pomocí spouštění cronu, který bude odmazávat staré záznamy.



Obrázek 4.4: Diagram případu užití znázorňuje, jaké bude mít host (neregistrovaný uživatel) možnosti pro interakci se systémem. Registrovaný uživatel má navíc možnost nahlédnout na všechny své analýzy. Administrátor má oprávnění pracovat přímo s metrikami, spouštět cron nebo upravovat oprávnění uživatelů. Dále je na diagramu vidět dědění možností od hosta až po administrátora nebo závislost některých možností označených parametrem *«include»*

4.4 Modely aplikace

Protože dnešní generace uživatelů využívá pro rychlou komunikaci a zjišťování informací mobilními telefony, je samozřejmostí, že i tato aplikace musí být responzivní. V mobilním zařízení zřejmě nebude probíhat přímo testování, ale například prezentace výsledku skrze mobilní telefon je praktickou možností pro uživatele. Aplikace slouží převážně kvalifikovaným uživatelům pro vědecké účely, ale druhotným záměrem je i oslovení širší veřejnosti a přiblížit jim nové možnosti.

4.4.1 Frontend

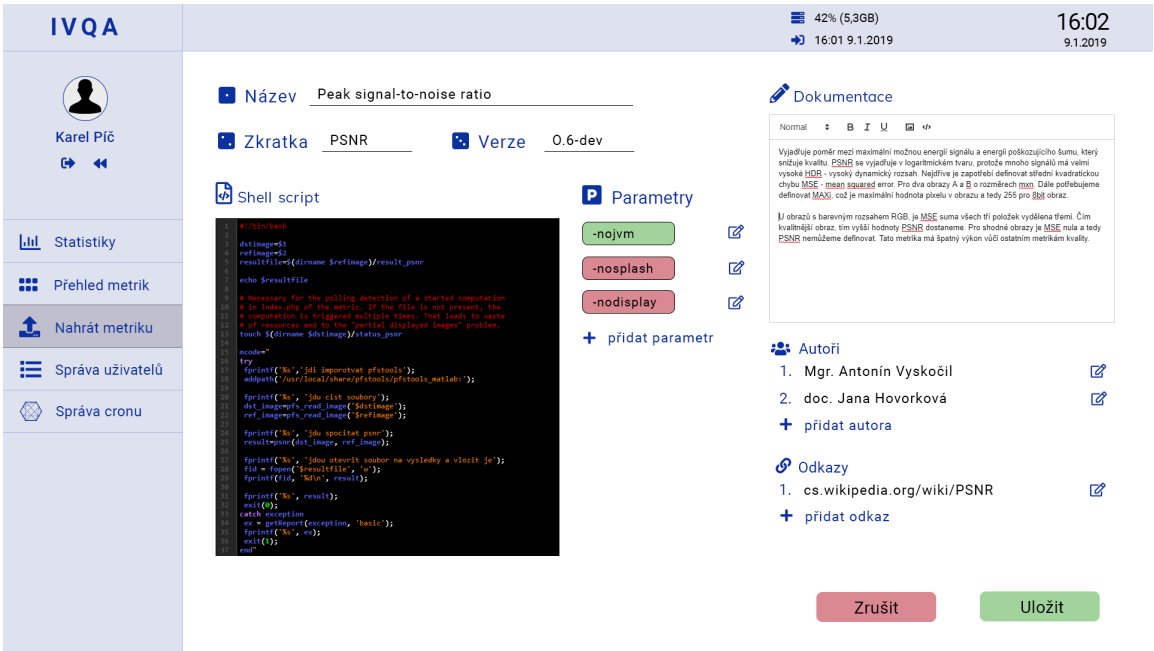
V podkapitole frontend si popíšeme webovou a mobilní část aplikace, a také celý koncept systému, tedy jak spolu jednotlivé objekty komunikují.

Webová část

Webová aplikace se skládá z uživatelské a administrační části. Jako první si rozebereme uživatelskou, která obsahuje pět hlavních stránek: *Hlavní strana*, *Přehled metrik*, *Dokumentace metriky*, *O nás* a *Analýza kvality*. Poslední stránka *Analýza kvality* je složená z více kroků: z nahrání dat, nastavení konfigurace a vyhodnocení. Při návrhu bylo dbáno na udržení jednoduchosti, použitelnosti a plnohodnotnosti aplikace. Hlavní vývoj byl zaměřen zejména na stránky, kde probíhá analýza.

Analýza kvality řeší interakci s uživatelem, jaké jsou potřeba nahrát data a navolit metriky. Po odeslání navolené konfigurace začne probíhat analýza a uživateli se postupně zobrazují výsledky. Navržený model musí akceptovat různé výstupy z metrik. Proto jsme se rozhodli použít modulární přístup k jednotlivým metrikám formou šablon. V základní konfiguraci aplikace máme dvě konkrétní šablony a jednu univerzální, které můžeme vidět v příloze E. Označení *univerzální* dostala pro schopnost zobrazit v seznamu všechny druhy výstupů, mezi které patří textové, obrazové, zkomprimované a video výstupy. Tyto šablony lze jednoduše naimplementovat a integrovat do aplikace. Administrátor si může v administraci zvolit ke každé metrice libovolnou šablonu, která se postará o správné vykreslení výsledků metriky uživateli.

Administrační část se skládá ze 6 stránek, a to: *Statistiky*, *Přehled metrik*, *Založení metriky*, *Správa uživatelů*, *Správa autorů* a *Správa cronu*. Následující obrázek návrhu zachycuje vytváření metriky, kde si můžeme také všimnout základního rozložení prvků, kde v levé části je umístěné menu s profilem a v pravé horní části jsou informace o vytížení serveru a aktuální čas.



Obrázek 4.5: Návrh administračního prostředí zachycující formulář pro tvorbu metriky. V levé části je vidět přihlášený uživatel a menu. V pravé horní části přehled o vytížení systému.

Mobilní část

V prvotní fázi jsme testovali, zda-li by neměla být mobilní verze zjednodušena na samotnou dokumentaci k metrikám kvůli velikosti nahrávaných dat a složitosti konfigurace. Nakonec jsme zvolili cestu plnohodnotné aplikace z důvodu velkého pokroku v oblasti mobilních zařízeních, kde displeje dnešních telefonů mají už vysoké rozlišení a tedy lze zobrazit plnohodnotné obrazy. Dnešní moderní telefony již umí ukládat fotografie do formátu RAW[4]. RAW je třída formátů, která nabízí fotografie v surovém stavu ještě před tím, než je zpracovaný, tedy přímo data poskytnuta senzory.

Na následujícím obrázku 4.6 můžete vidět porovnání mobilního návrhu (vlevo) oproti reálné aplikaci (vpravo). Můžete si všimnout, zvětšení velikosti textu a tlačítek pro lepší ovládání na mobilním zařízení. Je patrné, že jsme webovou aplikaci označili názvem IVQA a sjednotili jsme možnosti pro nahrávání obrazu a videa pod jedno tlačítko.

Obrázek 4.6: Vlevo můžete vidět návrh responzivního zobrazení na telefonu a vpravo námi vytvořenou aplikaci. Tlačítka a text byly zvětšeny proti návrhu pro lepší manipulaci na mobilním zařízení.

[4]RAW - z anglického slova RAW [surový]

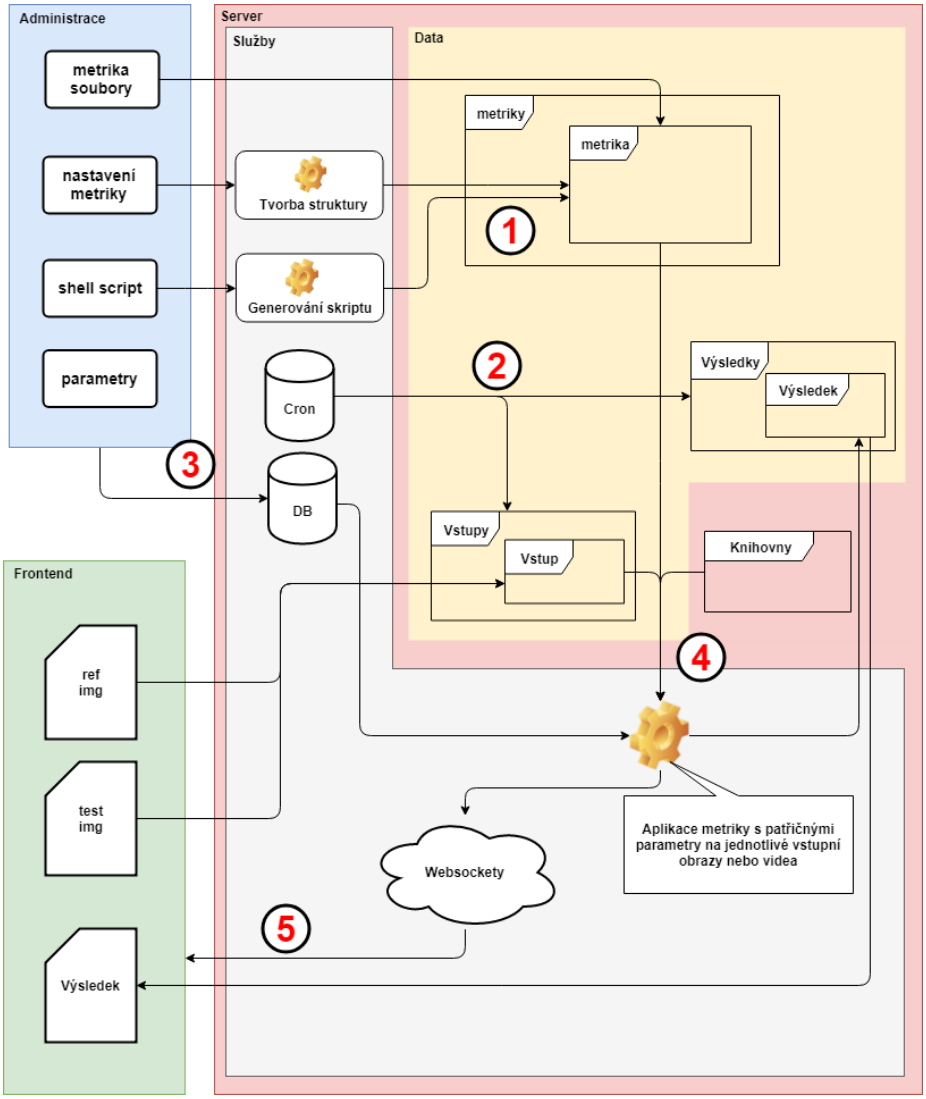
4.4.2 Systém funkčnosti

Pro lepší pochopení backendu a propojení jednotlivých služeb jsme sestavili schéma (obrázek 4.7). Schéma znázorňuje celý postup od uložení metriky, definování parametrů pro spuštění metriky a spouštěcího skriptu. Dále nahrání obrazů uživatelem a aplikaci metriky na uživatelský vstup. Je zde zachyceno, jak uživatel dostává aktuální informace o průběhu analýzy přes WebSockety[5] a postupné poskytování výsledků, které průběžně dostáváme k dispozici.

Administrace zachycuje definování metriky, tím je pro metriku vytvořena složka, kam může administrátor nahrát zdrojové soubory metriky a složka, do které je generován shell skript (na obrázku bod 1), sloužící ke spuštění metriky, který je zásadním faktorem pro běh analýzy. Při každé úpravě je znovu přegenerován, tak aby reflektoval změněné parametry. Parametry potřebné pro spuštění metriky jsou vloženy do databáze (bod 3), mohou mít různý datový typ, definovanou výchozí hodnotu a omezení pro uživatelský vstup. Po vložení obrazů přichází základní zpracování, kde se získají obecné informace o obrazu a jsou poskytnuty uživateli při čekání na výsledky analýzy. Po zahájení analýzy se nad uživatelskými obrazy spouští shell skript s definovanými parametry, do kterých se dosadí údaje zadané uživatelem (bod 4).

Výsledky jsou uloženy na server a kontrolovány cronem, zda nevypršela jejich expirační doba a zda není potřeba uvolnit místo na datovém úložišti serveru (bod 2). Ve výchozím nastavení jsme stanovili dobu, po kterou budou uchovány uživatelské data, na sedm dní od dokončení analýzy. Jinou hodnotu lze nastavit v konfiguračních souborech serveru. Odložené analýzy, tedy ty které jsou označeny jako náročné, se spouštějí, až když má server volnější zdroje pro hodnocení. Po dokončení každé analýzy je uživatel informován emailem, který obsahuje odkaz na výsledky.

[5] WebSockety - typ spojení mezi klientem a serverem



Obrázek 4.7: Návrh systému zachycuje vytvoření metriky z administračního prostředí, kde číslo 1 označuje vytvoření datové struktury a shell skriptu. Dále bod 3 zaznamenává uložení dat do databáze. Při nahrání obrazů pro analýzu z frontendové části se v bodě 4 vezmou údaje z databáze, uložené vstupy a metrika se shell skriptem, který se spustí a zpracuje nahrané obrazy a výsledky uloží na server. V bodě 5 můžeme vidět, jak WebSockety informují o aktuálním stavu analýzy. V poslední řadě v bodě 2 cron promazává uživatelova stará nahraná data a vygenerované výsledky.

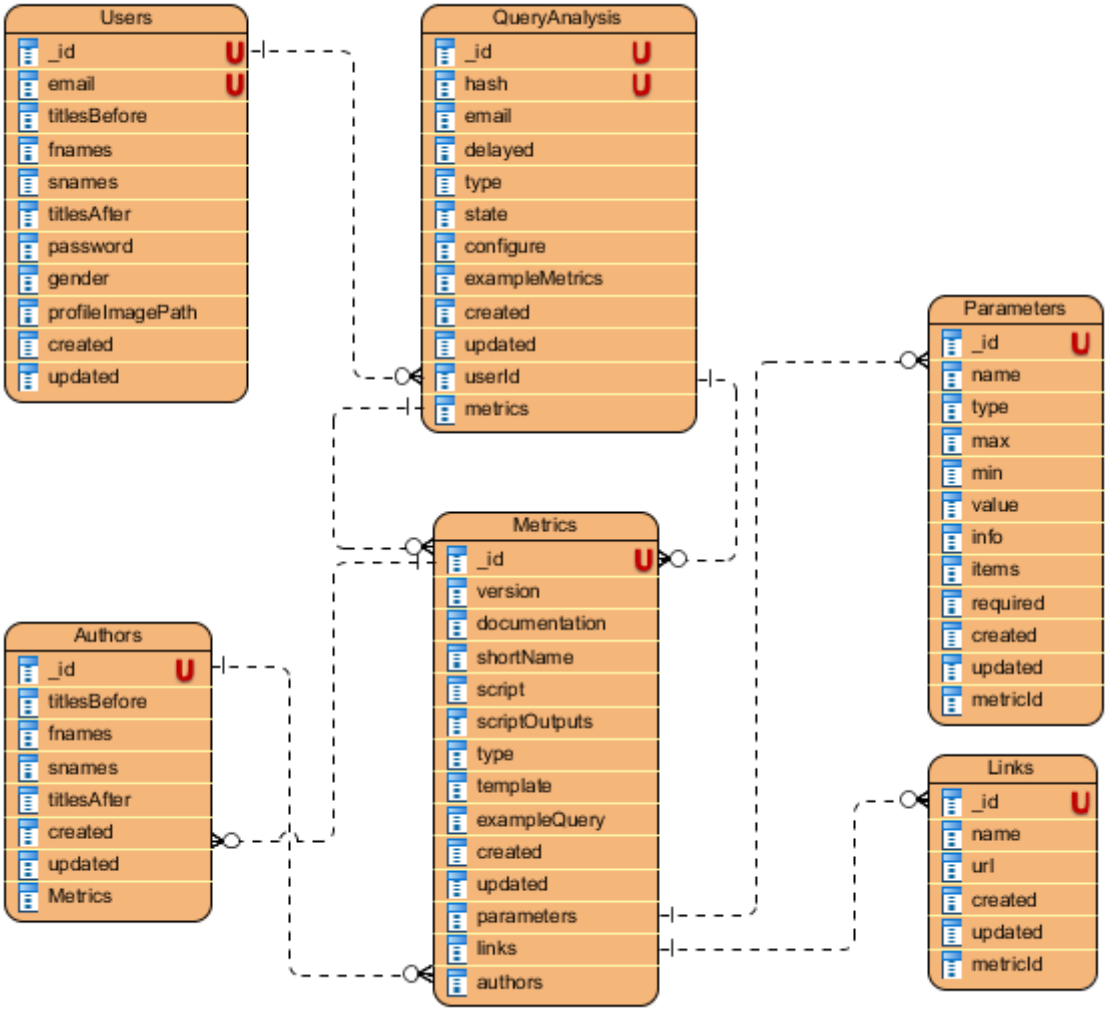
4.5 Datový model

Z datového modelu, jak je uveden níže na obrázku (4.8), můžeme vidět rozdělení do šesti základních entit potřebných pro správný chod aplikace. Mezi nejdůležitější se řadí *QueryAnalysis*, *Metrics* a *Parameters*, které si následně vysvětlíme.

První entita je *Users*, která obsahuje jednotlivé uživatele, je svázaná s entitou *QueryAnalysis*, ve které bude pořadí záznamů analýz obsahující konfiguraci způsobu aplikace metrik a především zde budou čekající analýzy před zpracováním. V případě, že bude aplikace vytížená, se nebudou spouštět analýzy automaticky, ale budou odložené. Pravidelně

spouštěný příkaz službou cron kontroluje počty běžících analýz a pokud zjistí, že má server volné zdroje, spustí další analýzy, které jsou na řadě. Fronty jsou napojené na entitu *Metrics*, podle tohoto spojení fronty poznají, kterou metriku uplatnit. Na metriky se pojí autoři (*Authors*), abychom mohli poskytovat tyto informace v rámci dokumentace metriky. Ta obsahuje také tabulku s odkazy (*Links*), které odkazují na místa, ze kterých bylo čerpáno a kde je možné čerpat další informace o dané metrice.

V prvotním návrhu byla tabulka QueryAnalysis zvlášť rozdělena na obrazové a video analýzy, ale při návrhu jsme dospěli k závěru, že mají společnou strukturu a datový návrh jsme konsolidovali a optimalizovali.



Obrázek 4.8: Entity-relationship diagram znázorňující databázovou strukturu 6 tabulek. Z nichž stěžejní pro naší aplikaci je entita Metrics, která obsahuje shell skript a informace o tom, jaké má metrika výstupy.

Nejdůležitější entitou je entita *Metrics*, kde kromě dokumentace, verze metriky, autorů a odkazů, které budou poskytnuty uživatelům, jsou i parametry. Parametr musí mít definovaný název, typ a výchozí hodnotu. Výchozí hodnota je použita v případě, kdy uživatel nezadá žádnou hodnotu u parametru, který metrika vyžaduje. Dále je možné definovat

maximální a minimální omezení pro případ, kdy bude uživatel volit hodnotu sám tak, aby nepřesáhl dané rozmezí. Kromě parametrů obsahuje entita Metrics předpis pro shell skript a definici výstupních údajů z metriky, abychom byli schopni uživateli prezentovat data vygenerovaná metrikou.

Kromě shell skriptu jsou pro aplikaci základní ještě další dvě schémata. Prvním schématem je popis výstupů metrik. Z analýzy nám vyplynuly následující možné typy pro výstup metriky:

- **Řetězec** - pro možnost zaznamenání číselné hodnoty poměru kvality
- **Obrazový výstup** - pro očekávaný výstup z obrazové metriky
- **Video výstup** - pro očekávaný výstup z video metriky
- **Komprimovaný výstup** - pro možnost stažení více souborů najednou

Komprimovaný výstup byl přidán, protože mohou nastat situace, kdy metrika pracuje s videem a výstupem jsou upravené jednotlivé snímky. Jelikož jich bude velké množství, tak je server zkomprimuje do formátu ZIP a nabídne uživateli ke stažení.

Druhé schéma je zaznamenaná konfigurace analýzy samotným uživatelem. Jak můžeme vidět na následující straně v ukázce záznamu konfigurace analýzy 4.1 je uvedena heš identifikující samotnou konfiguraci. Na řádku č.4 je email, na který bude odeslán odkaz pro zobrazení nebo sdílení výsledků analýzy. Na řádku č.5 je informace o tom, zda je metrika odložená, nebo se vyhodnocuje přímo. Například, pokud se jedná o zpracování videa, uživatel si musí počkat na doručení emailu s odkazem na vyhodnocenou analýzu. Od řádku č.7 je pole předpisů pro jednotlivé metriky, v této ukázce je jen jedna metrika se dvěma parametry a následují tři konfigurace pro zpracování souborů. Každá konfigurace souborů obsahuje informaci o zdrojovém a referenčním vstupu a unikátní identifikátor pro oddělení výsledků z metriky. Také si můžete všimnout zaznamenávání stavu analýzy na úrovni samotné analýzy (řádek č.6), metriky (řádek č.10) a konfigurací souborů (řádek číslo 22, 28 a 34). Stavy mohou nabýt následujících 5 hodnot:

- **PENDING** - čekání na zpracování
- **START** - zpracování probíhá
- **END** - zpracování bylo dokončeno
- **ERROR** - zpracování neproběhlo úspěšně
- **DELETED** - stará analýza, která byla už odebrána

```
{
  "id": "5cc1c8367c87eb61a848dcb3",
  "hash": "IQM-TEST",
  "email": "admin@ivqa.cz",
  "delayed": false,
  "state": "START",
  "configure": [
    {
      "metricId": "5cb318d4bcb64b5718797ea3",
      "state": "START",
      "parameters": [
        {
          "parameterId": "5cb363c9c82b47449039fd6c",
          "value": "100"
        },
      ],
      "files": [
        {
          "srcName": "test_image_01.jpg",
          "refName": "test_image_03.png",
          "uniqueId": "TFq6rBtqV",
          "state": "END"
        },
        {
          "srcName": "test_image_01.jpg",
          "refName": "test_image_05.jpg",
          "uniqueId": "V_V3d5uH_I",
          "state": "START"
        },
        {
          "srcName": "test_image_02.jpg",
          "refName": "test_image_05.jpg",
          "uniqueId": "V_V3d19H_I",
          "state": "PENDING"
        }
      ]
    }
  ]
}
```

Výpis 4.1: Ukázka datového uložení konfigurace analýzy. Jedná se o jednu metriku a tři zpracování, kde první je už dokončené, druhé probíhá a třetí vyčkává

Kapitola 5

Implementace

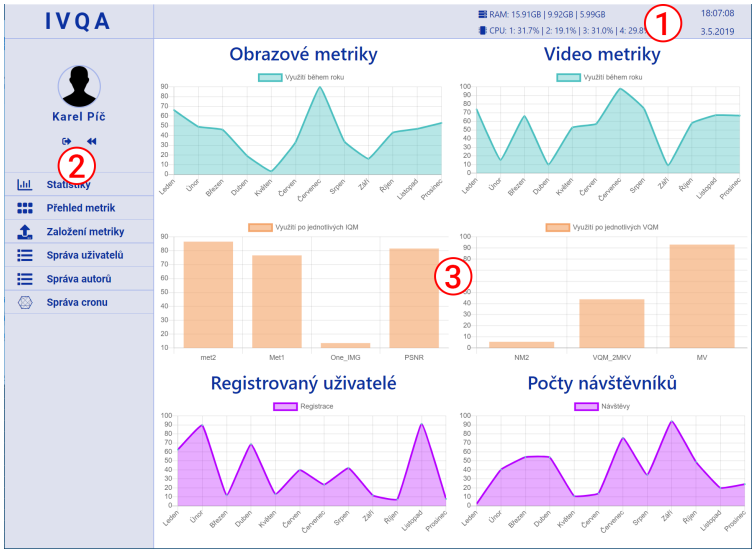
Aplikace je implementována a následně nasazena na virtuálním stroji s operačním systémem Linux (postavený na distribuci Ubuntu) z důvodu nutnosti mít nasazené grafické balíčky a nástroje pro správný chod aplikace. Virtuální stroj je vhodný s ohledem na přenositelnost mezi různými platformami.

V této kapitole si také rozebereme jednotlivé vrstvy aplikace, jak vypadá a čeho se nám podařilo docílit, dále jaké technologie jsme použili, v čem mají výhody a nevýhody. Následuje popis vývojového prostředí s programy a balíčky, se kterými aplikace komunikuje a jsou potřeba pro správný běh. Také se dozvíte v podkapitole 5.4, jakým způsobem aplikace pracuje s obrazem a videem, za kterou následuje popis, jak spolu komunikují její jednotlivé celky, k čemu byl využit cron, v čem spočívá dynamika aplikace a v čem nám pomůže uživatelský pomocník. V posledních dvou podkapitolách se dozvíme, jak byla řešena bezpečnost aplikace (5.9) a jaké máme možnosti konfigurace jednotlivých částí aplikace (5.10).

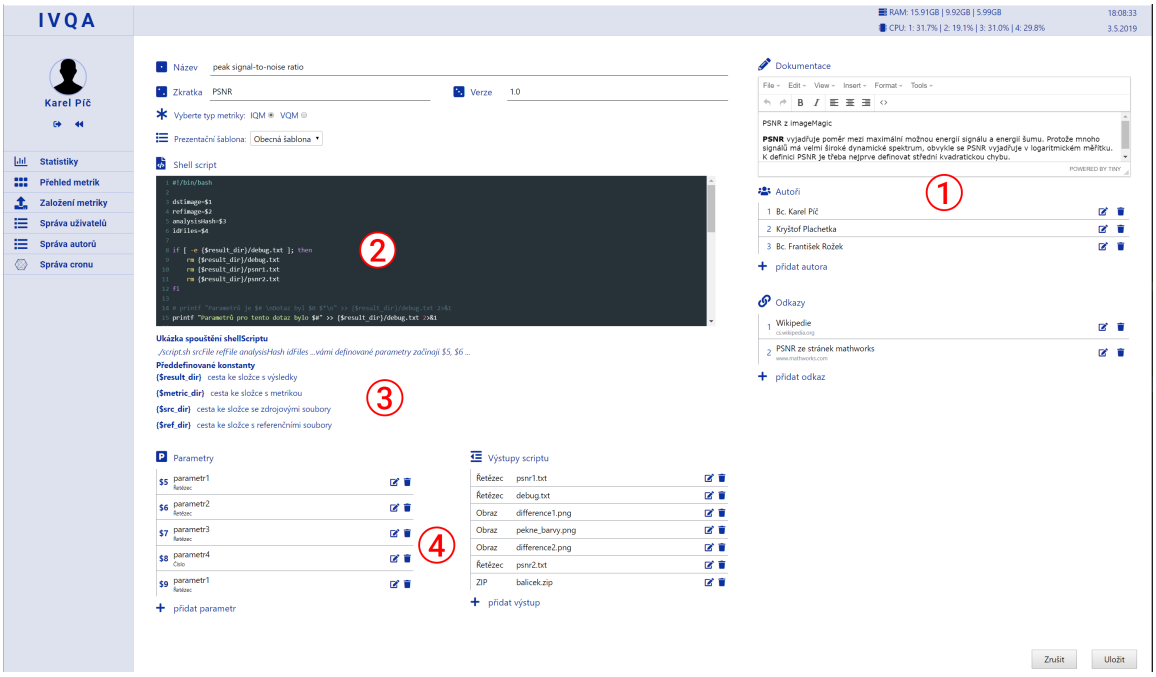
5.1 Vrstvy aplikace

V této podkapitole si ukážeme, jak aplikace vypadá a popíšeme si základní vrstvy aplikace. Začneme administračním prostředím. Na obrázku 5.1 bod jedna vyobrazuje umístění základních informací. Kromě času je zde údaj o vytížení jednotlivých jader procesoru a kolik je využité operační paměti serveru. Následuje bod dvě, který označuje jednoduchou základní navigaci v administraci. Pokračujeme bodem tři, který nám označuje statistiky v grafech, kde levé dva grafy zobrazují statistiky pro obrazové metriky. První graf je využití v čase a druhý graf je využití jednotlivých metrik. Stejné vlastnosti platí pro pravé dva grafy, jen s tím rozdílem, že se jedná o video metriky. Poslední dva spodní grafy nám zobrazují počty registrovaných uživatelů a počty návštěvníků.

Obrázek 5.2 zachycuje formulář pro tvorbu, či editaci metriky. V bodě jedna si můžeme všimnout dokumentace, autorů a odkazů. Tyto položky se využívají při výpisu dokumentace metriky. Dále bod 2 zachycuje editaci shell skriptu s nápovědou konstant jako jsou cesty ke zdrojovým a referenčním souborům v bodě tři. Bod čtyři označuje vstupní parametry metriky a výstupní data z metriky, která potřebujeme znát pro správné zobrazení výsledků při vyhodnocení analýzy.

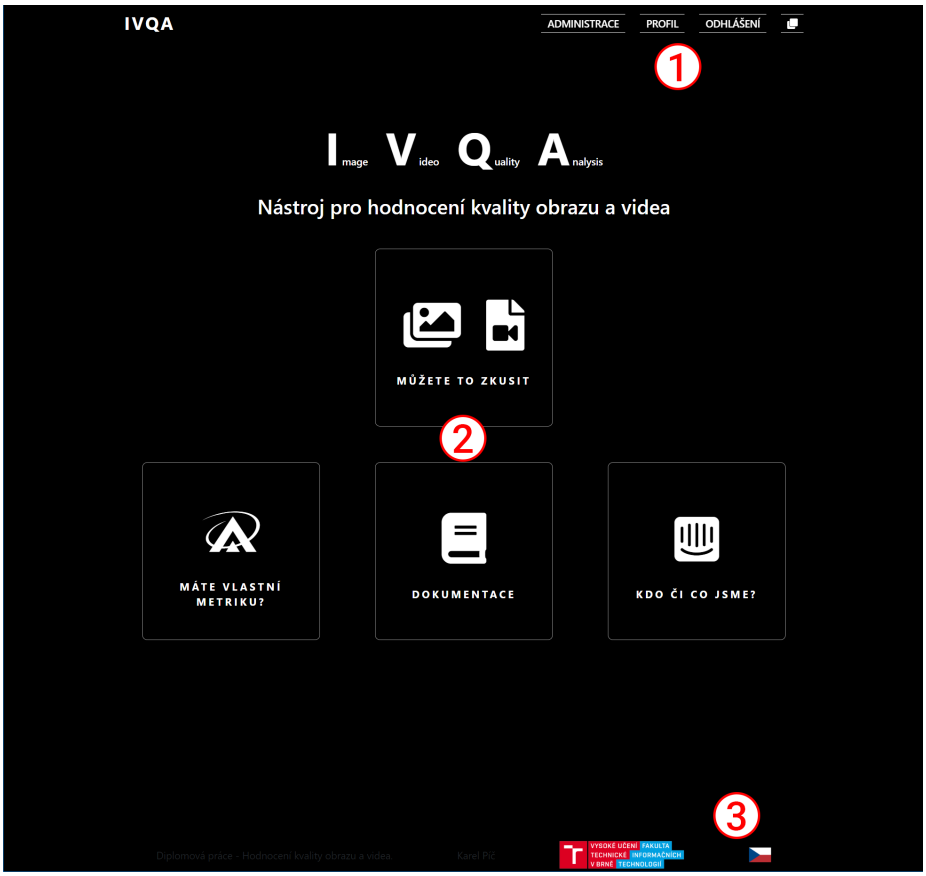


Obrázek 5.1: Administrační prostředí se základními údaji v bodě jedna a menu v bodě dva. Poslední bod tři označuje statistiky, kde horní dva grafy vyobrazují informace o využití obrazových metrik. První jsou informace využití v čase a druhý je využití podle jednotlivých metrik. Stejné parametry platí i pro pravé dva grafy s tím rozdílem, že se jedná o video metriky. Poslední dva spodní grafy zachycují množství registrovaných uživatelů a počty návštěvníků.



Obrázek 5.2: Formulář pro vytvoření a editování metriky s dokumentaci, autory a odkazy v bodě jedna. Editace shell skriptu v bodě dva, dále s konstantami pro dosazení cesty ke zdrojovým nebo referenčním souborům v bodě tři. Poslední bod 4 ukazuje na parametry metriky a výstupní data z metriky.

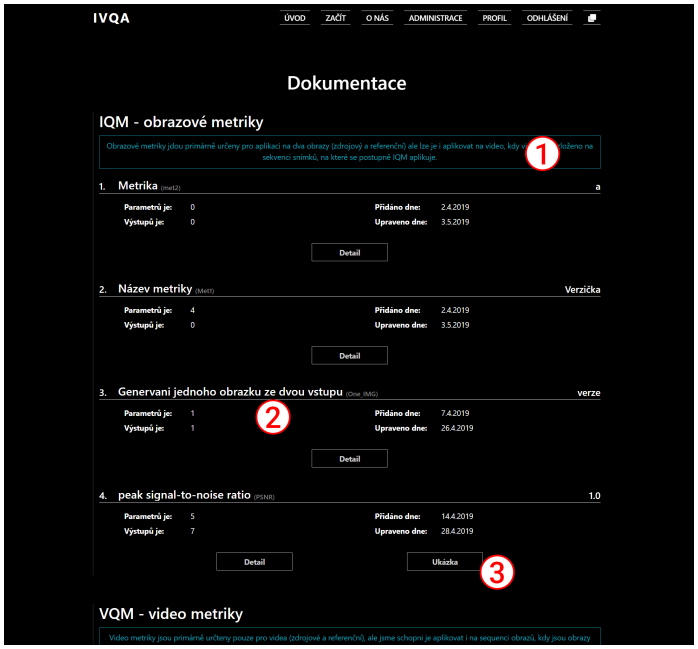
Na obrázku 5.3 je zachycena hlavní strana naší aplikace, kde bod jedna označuje hlavní menu aplikace, kde se dynamicky objevují tlačítka podle polohy v aplikaci. Bod dvě zachycuje hlavní 4 tlačítka na nejdůležitější stránky. První *Můžete to zkusit* vede na konfiguraci analýzy. Další tlačítko nabízí možnost uživateli informovat nás o své vlastní metrice. Třetí tlačítko směřuje na dokumentaci a poslední tlačítko přesměruje uživatele na informace o tomto projektu. Posledním bodem tři na tomto obrázku je možnost přepínání jazykové verze aplikace. Dostupná je čeština a angličtina pro frontendovou část aplikace.



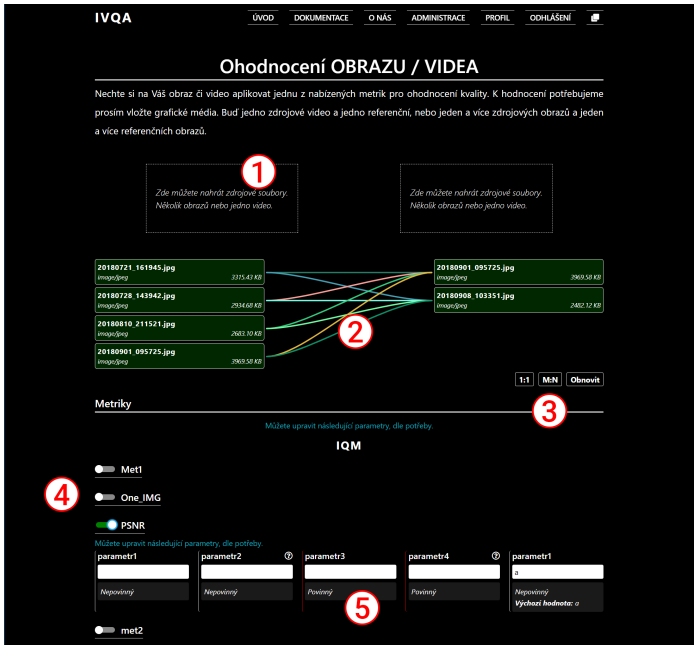
Obrázek 5.3: Hlavní strana aplikace poukazující na menu v bodě jedna. Hlavní rozcestník v bodě dvě a možnost přepnutí jazykové mutace v bodě tři.

Dokumentace je zachycena obrázkem 5.4, kde jsou vypsány všechny obrazové metriky se základními informacemi. Bod jedna znázorňuje informační hlášku pro nové návštěvníky. Bod dvě označuje jeden záznam metriky a bod tři je umístěn u metriky, která nabízí ukázku analýzy. Můžeme si všimnout, že ukázka nemusí být nastavena u všech metrik. Stránka pokračuje video metrikami, které jsou vypsány stejným způsobem.

Na konfiguraci hodnocení kvality obrazu a videa na snímku 5.5 můžeme vidět možnost nahrávání zdrojových a referenčních souborů označených bodem jedna. Druhý bod označuje propojení, respektive výběr, které soubory budou spolu předány metrice a tím i vyhodnoceny. Bod tři poukazuje na rychlé menu, které nabízí možnost propojení 1:1, M:N nebo zrušit navolené propojení. Následuje výběr metrik v bodě čtyři a výběr parametrů v bodě pět. Metrika může nabízet povinné a nepovinné parametry. Povinné jsou navíc zvýrazněny červeně.



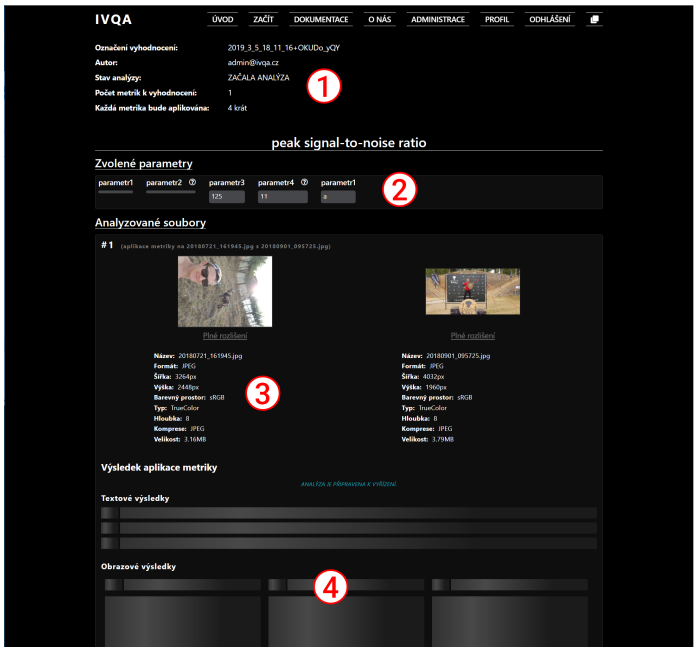
Obrázek 5.4: Dokumentace metrik s výpisem všech metrik. Bod jedna zachycuje hlášku pro nové návštěvníky, bod dvě označuje jeden záznam metriky a bod tři poukazuje na možnost zobrazení ukázky aplikování metriky.



Obrázek 5.5: Postupné získávání informací o obrazech, kdy vlevo jsou už načtené a vpravo je stále aktivní načítání.

Poslední obrázek 5.6 v této kapitole je probíhající analýza, kde nahoře v bodě jedna máme základní informace o aktuální analýze, v bodě dva jsou navolené parametry, které jsou podány dané metrice při spuštění analýzy. Bod tři zachycuje výpis informací, které jsme

o daném obrazu zjistili. Poslední bod čtyři zachycuje načítání výstupů metriky. Už z načítání víme, že dostaneme tři textové výstupy a tři obrazové výstupy. Protože jednotlivé metriky mají různý datový výstup, lze v administraci k jednotlivým metrikám přiřadit šablonu, která upravuje tuto prezentaci výsledků k dané metrice.



Obrázek 5.6: Postupné získávání informací o obrazech, kdy vlevo jsou už načtené a vpravo je stále aktivní načítání.

5.2 Použité technologie

- Node.js[1]

  Node.js je javascriptový engine uvolněný jako open-source. Nabízí široké uplatnění v backendu. Vytváří prostředí, kde lze spouštět zdrojový kód napsaný v JavaScriptu, bez nutnosti prohlížeče. Díky možnosti asynchronních akcí a vysoké škálovatelnosti je Node.js schopen obsloužit několik datových proudů najednou a tím se dostává do popředí. V dnešní době to je už velmi používaný nástroj. Využívá se především pro sestavování serverových částí a SPA[2]. V našem případě pod Nodem poběží celý server, tedy převážná většina služeb a balíčků, které níže zmiňujeme.

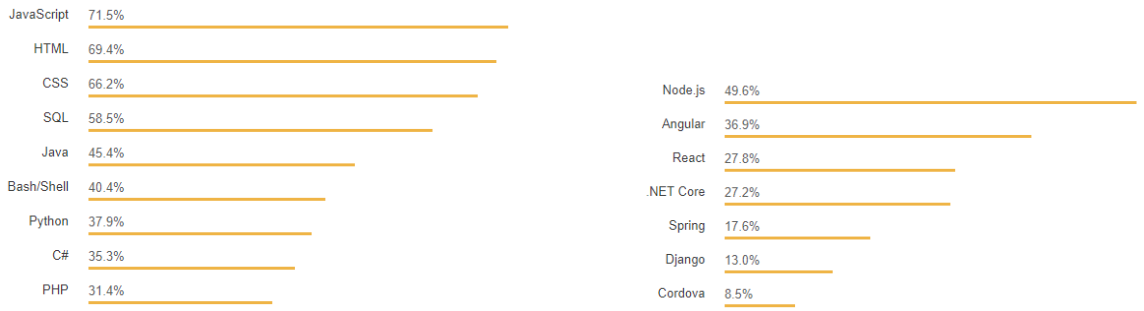
  JavaScript již před několika lety začal konkurovat PHP, které zaznamenalo menší pokles a JS, který běží nejen na klientské straně, ale díky Node.js i na serverové straně, zažívá velký rozmach. Samotný Node.js zaznamenává růst, co se týče vývoje, nových uživatelů a balíčků, které rozšiřují jeho funkcionalitu.

  Následně si zobrazíme dva grafy (obrázek 5.7), kde první znázorňuje porovnání mezi užívání programovacích, skriptovacích a značkovacích jazyků. Druhý obrázek porovnává užívání frameworků, knihoven a nástrojů. Tyto grafy jsou ze stránek Stack Over-

[1] Node.js - javascriptový engine - https://nodejs.org/en/
[2] SPA - Single-Page Application - https://en.wikipedia.org/wiki/Single-page_application

flow, kde se můžete dozvědět více o tomto hodnocení[3]. Stack Overflow dělá každý rok rozsáhlý průzkum o tom, jak jsou aktivní vývojáři, o co se zajímají, jak moc jsou rozšířené technologie, se kterými se v dnešní době pracuje a také jaké finanční ohodnocení mají odborníci s danými jazykovými dovednostmi.

Výsledky, o kterých zde píšeme, jsou pro rok 2018. Z daných stránek se také můžeme dozvědět, že JavaScript je už 6. rokem nejoblíbenějším a nejvíce užívaným jazykem. Graf obsahuje i informaci o Reactu, který je velmi populární. I ten jsme se rozhodli použít pro naší aplikaci. Přímo v průzkumu se o Reactu zmiňují a uvádí, že je druhým rokem na předních příčkách.



Obrázek 5.7: Statistiky, kde vlevo je porovnání mezi programovacími, skriptovacími a značkovacími jazyky a vpravo je porovnání mezi frameworky, knihovnami a nástroji. Ze stránek Stack Overflow.

- React.js[4]

  React.js je open-source knihovna pro vytváření uživatelského prostředí vyvinutá Facebookem. Tato knihovna je minimalistického řešení a využívá sílu jednoduchosti. Veškeré části webu se skládají z komponent, které popisují, jak má vypadat výsledek.

  Hlavní výhodou je možnost přecházení mezi jednotlivými stránkami, při kterém nedochází k načítání vždy celého webu, ale pouze k překreslení potřebné části. To například oproti PHP velice zrychluje zpětnou vazbu a tím dokáže více upoutat uživatele a neodradí ho dlouhým načítáním.

  Základním stavebním kamenem je komponenta, která má dva hlavní parametry. Jsou to *Props* a *State*. První zmíněné props, je objekt vstupních parametrů a daná komponenta je nesmí upravovat, ale pouze číst. Props znázorňují vnější stav komponenty. Opačně State je objekt, který znázorňuje vnitřní stav komponenty a lze ho měnit pouze voláním funkce k tomu určené - *this.setState*. Dále komponenta má několik základních funkcí, mezi nejdůležitější patří *componentDidMount*, která inicializuje komponentu a nastavuje základní vlastnosti. Obdobou je *componentWillUnmount*, která se stará o pročištění aplikace při odebrání/ukončení komponenty. Součástí pročištění může být například smazání posluchačů na elementy z DOMu, časovačů, nebo k ukončení právě běžících procesů. Dále máme funkci *render* vracející elementy, které se uživateli vykreslí. Tyto elementy jsou popsány syntaxí JSX, která byla vytvořena pro jazyk JavaScriptu. Slouží pro zjednodušení a urychlení zápisu.

[3]Hodnocení - https://insights.stackoverflow.com/survey/2018
[4]React.js - javascriptový framework od Facebooku - https://reactjs.org/

Na níže uvedeném obrázku 5.1 můžeme vidět v horní části základní komponentu Reactu, která ukazuje, jakým způsobem dochází ke změně vnitřního stavu. Při změně stavové vlastnosti komponenty dojde k překreslení na stránce pouze konkrétní hodnoty a ne celé komponenty. Komponentě je to umožněno díky přístupu k virtuálnímu DOMu, kdy se snaží upravovat jen položky u kterých došlo ke změně a to porovnáním klasického a virtuálního DOMu. Dále ve spodní části se nachází zpracování překladačem JSX syntaxe.

```
1  import React, { Component } from 'react';
2  import { getDateFormat } from '../../utils';
3
4  class Clock extends Component {
5    constructor(props) {
6      super(props);
7      this.state = { date: new Date() };
8      this.updateTime = this.updateTime.bind(this);
9
10     this.timerId = setInterval(this.updateTime, 1000);
11   }
12
13   componentWillUnmount() {
14     clearInterval(this.timerId);
15   }
16
17   updateTime() {
18     this.setState({ date: new Date() });
19   }
20
21   render() {
22     const { date } = this.state;
23     return (
24       <div>
25         <div>Text, ktery se nebude prekreslovat</div>
26         <div>{date.toLocaleTimeString())}</div> // Tento div se bude prekreslovat
27       </div>
28     );
29   }
30 }
31
32 // ukazka prekladu JSX
33 const element = (
34     <h1 className="myClass">
35         Nadpis
36     </h1>
37 );
38
39 const element = React.createElement(
40     'h1',
41     {className: 'myClass'}
42     'Nadpis'
43 );
```

Výpis 5.1: Ukázka zápisu v Reactu. Základní “reactí“ komponenta, na které je ukázaný příklad změny stavu datumu. Pod komponentou je zobrazena ukázka překladu syntaxe JSX.

- Redux

  Redux je důležitý kešovací nástroj. My ho používáme pro ReactJS, kdy více komponent pracuje se stejnými informacemi a proto potřebují držet jednotný stav. K tomu nám poslouží globální úložiště, kam se budou komponenty dívat a aktualizovat se na základě změn v datech. Redux drží data v RAM po celou dobu relace v internetovém prohlížeči. Ve výpisu 5.2 můžete vidět základní složky keše v reduxu, pro názornost je vypsána pouze do hloubky 1.

  Jediná náročnost, která se skrývá za používáním Reduxu je úprava globálního stavu, kdy změny musejí probíhat postupně. V jednu chvíli může měnit úložiště jen jedna akce. K práci s centrální keší slouží *Actions* a *Reducers*. Akce slouží k vyvolání změny, obsahuje typ a objekt s údaji, které se mají změnit. Následně se akce zařadí do fronty, kterou postupně odbavuje Redux. Při odbavení si akci přebírá reducer, který na vstupu má současný stav keše a danou akci. Dle typu akce vybere postup v ohromném rozcestníku a ten s danou akcí provede. Reducer začne vytvořením nového stavu, do kterého vloží současný stav a upraví ho o údaje z akce. Po dokončení úpravy, reducer vrací nový stav. Důležité je, aby nový stav nebyl původní zmutovaný stav (objekt). Důležité je, aby nový stav nebyl původní zmutovaný stav (objekt), protože "reactí" komponenty neporovnávají obsah objektů, ale pro vyšší rychlost porovnávají jejich adresy. Zmutovaným stavem je myšlen případ, kdy komponenta obdrží starý objekt s aktualizovanou hodnotou, ale se stejnou adresou.

```
1  {
2    notifications: [...],
3    user: {...},
4    users: {...},
5    app: {...},
6    ws: {...},
7    files: {...},
8    metrics: {...},
9    links: {...},
10   authors: {...},
11   parameters: {...},
12   analysis: {...},
13   debug: {...}
14 }
```

Výpis 5.2: Toto je struktura keše spravovaná reduxem a je vypsána pouze do hloubky úrovně 1.

- Webpack

  Důležitý nástroj pro kompilaci souborů a jejich minimalizaci, někdy bývá nazýván jako balíčkovací nástroj. Webpack prochází všechny javascriptové soubory a odebírá z kódu komentáře, zbytečné řádky a kód, který nebude nikdy proveden. Následně všechny soubory sloučí dohromady. To je základní funkčnost webpacku. Ale můžeme ho rozšířit o různé loadery. Loadery jsou nástroje pro podporu dalších typů souborů, respektive jazyků. Námi přidaná podpora pro React zpracuje JSX syntaxi a vytvoří čistý JS. Ten webpack už zpracuje běžným způsobem.

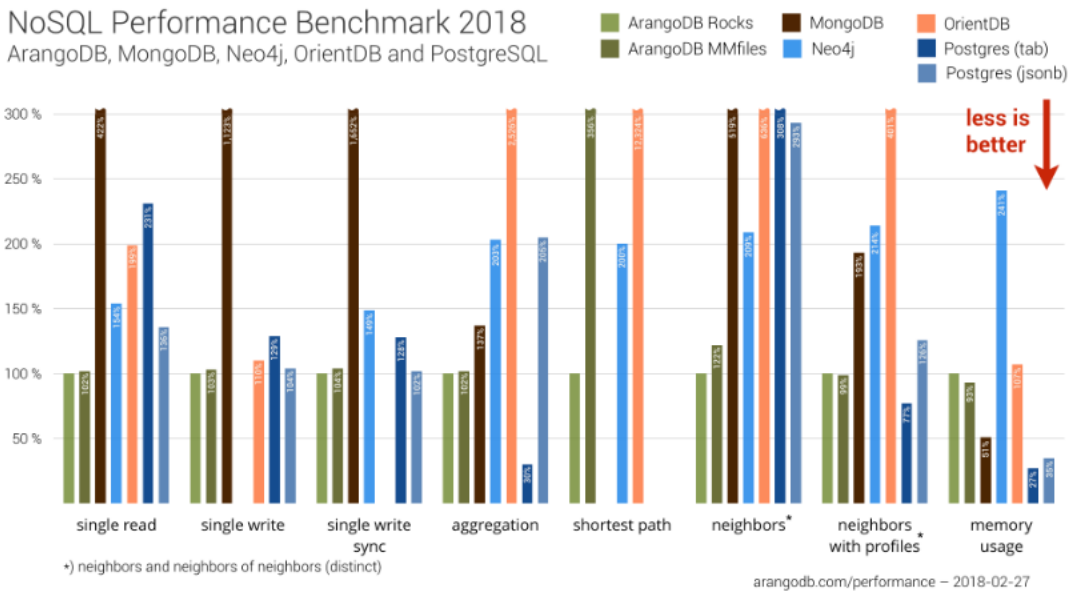
Další loader, který jsme do Webpacku přidali je css-loader a scss-loader. Pokud tedy webpack při procházení všech souborů narazí na stylové soubory SCSS, tak je převede na CSS a ty sloučí do jednoho celku. Pokud narazí při průchodu na nějaké statické soubory jako jsou fonty nebo obrázky, tak je překopíruje do výstupní složky. Všechny tyto loadery a postupy, jak se má webpack zachovat, když narazí na soubory, které nezná, jsou popsány v konfiguraci umístěné v souboru *webpack.config.js*. V našem případě je překopíruje do výstupní složky.

Dobrou funkcí je generování výsledného balíku javascriptu s heší v názvu, protože při vydání nové verze aplikace chceme, aby si uživatel stáhl novou verzi a přepsal tím tak starou verzi uloženou v keši prohlížeče. Webpack nabízí i prostředky pro nastavení kompatibility s různými prohlížeči. V našem případě necháváme nastavení ve výchozím stavu a podporujeme všechny prohlížeče, které ještě nemají ukončenou vlastní podporu.

- MongoDB[5]

  MongoDB je open-source dokumentová databáze, která se neskládá z tabulek, ale z kolekcí. Ty obsahují datovou strukturu JSON s informacemi, které mají tvar klíč - hodnota. MongoDB nepatří mezi relační databáze, její vlastnosti odpovídají databázovému konceptu NoSQL. Samotné jádro MongoDB je napsáno v C++.

  Na obrázku 5.8 se můžeme podívat na porovnání výkonosti databáze MongoDB vůči jiným NoSQL databázím. Pro nás jsou zajímavé hnědé sloupce značící Mongo a je vidět z prvních dvou grafů jeho sílu v jednoduchých dotazech. Z posledního grafu vyplývá výborná práce Monga s pamětí.



Obrázek 5.8: Výkonnostní benchmark NoSQL databází, kde se porovnává rychlost čtení a zápis jednoho záznamu. Synchronní zápis, agregace a další. V posledním grafu lze vidět i vytížení paměti samotnými databázemi. MongoDB je znázorněno hnědými sloupci. Graf přebrán z webu www.arangodb.com.

[5]MongoDB: Open-Source Document Database https://www.mongodb.com/

Všechny datové záznamy uložené v kolekcích Monga, nenastavíme-li je jinak, mají identifikátor *_id*. Ten představuje 12 - bajtový řetězec, který je jedinečný v rámci celé databáze. Je to díky rozdělení řetězce na 4-bajty pro časové razítko, 3-bajty pro identifikaci vašeho stroje, 2-bajty pro identifikaci procesu samotného Monga a poslední 3-bajty jsou přírůstkové hodnoty. Výhody MongaDB spočívají v jednoduchosti vedení záznamů, ve kterých lze ukládat kromě běžných typů i zanořené struktury. V těchto zanořených strukturách je možné i vyhledávat. Pro urychlení vyhledávání se zavádějí indexy, které lze umístit nad libovolný atribut kolekce.

Mongo bylo použito místo MySQL z důvodu schopnosti zaznamenávat celé objekty v JSONu a možnosti v nich vyhledávat. V našem případě je to vlastnost *configure* v kolekci *QueryAnalysis* a vlastnost *scriptOutputs* v kolekci *Metrics*. MySQL také nabízí možnosti pro práci s JSONem, ale Mongo je na práci s ním lépe navrženo.

- GraphQL[6]

  GraphQL je open-source dotazovací jazyk pro rozšíření API o striktně typovaný přístup. Dovoluje nám deklarativně definovat požadavky na data, čímž dostáváme možnost zapisovat dotazy formou předpisu, neboli popisu struktury požadovaných dat. GraphQL nám ošetřuje výjimečné stavy, kdy například nejsou data k dispozici, žádáme o neexistující datovou strukturu, nebo na dotazované informace nemáme oprávnění. Také nám snižuje datovou náročnost při komunikaci se serverem, protože klient obdrží pouze datovou strukturu, kterou si popsal v předpisu.

  Tento dotazovací jazyk se skládá ze 4 základních klíčových slov. Je to dvojce *Queries* s *Mutations* a *Schemas* s *Types*. Jednoduše řečeno dotazy (Queries) slouží pro získání dat z databáze a mutace (Mutations) pro jejich úpravu. Schémata (Schemas) (příklad je na výpisu 5.3), kde jsou definovány datové typy (Types) a vlastnosti, na které se lze dotazovat. Schémata obsahují i předpisy dotazů a mutací, kde jsou definovány názvy funkcí a parametry, u kterých je uveden datový typ a jeho povinnost. Díky této komplexní definici těchto 4 klíčových slov dokážeme vždy zadat přesný dotaz, na základě kterého dostaneme přesný výsledek. To umožňuje GraphQL validovat datové struktury probíhající v komunikaci serveru a klienta.

```
1  type Link {
2      id: String
3      name: String
4      url: String
5      created: DateTime
6      updated: DateTime
7      metricId: String
8  }
9
10 input LinkInput {
11     name: String!
12     url: String!
13 }
14
15 type Query {
16     link(id: String!): Link
17     allLinks: [ Link ]
18 }
```

[6]GraphQL - dotazovací jazyk a slouží k nadstavbě databáze - https://graphql.org/

```
19
20  type Mutation {
21      addLinks(links: [LinkInput], metricId: String!): [ Link ]
22      addLink(name: String!, url: String!, metricId: String!): Link
23      deleteLink(id: String!): Link
24      updateLink(id: String!, name: String, url: String): Link
25  }
```

Výpis 5.3: Schéma je nedílnou součástí GraphQL a slouží pro definici informací, které mohou být v kolekci zaznamenány, na jaké datové typy můžeme obdržet dotaz a jak s API lze komunikovat.

Pro ukázku bohatosti dotazování v GraphQL poslouží dva obrázky 5.9 a 5.10. Na obou můžete vidět dvě části. V levé části je vždy dotaz na autora s předpisem, který se vkládá do těla POSTu a v pravé části jsou vrácené informace z databáze. Při porovnání obou obrázků můžeme vidět rozdílný předpis v dotazu a jak GraphQL vrací přímo hodnoty ze zanořených vlastností. Například při propojení s jinou kolekcí nebo pokud je v dokumentu uložený přímo nějaký objekt. Pro úpravy návratových dat nám stačí upravit datový předpis v dotazu a nemusíme upravovat celé API. V tomto je síla GraphQL, kde pouze schématy popíšeme, na co se může klient dotazovat.

```
1  query author($id: String!) {
2    author(id: $id) {
3      id
4      titlesBefore
5      fnames
6      snames
7      titlesAfter
8      metrics {
9        shortName
10     }
11   }
12 }
```

```
{
  "data": {
    "author": {
      "id": "5ca9a763e5d7d552a85e3491",
      "titlesBefore": "",
      "fnames": "Kryštof",
      "snames": "Plachetka",
      "titlesAfter": "",
      "metrics": [
        {
          "shortName": "One_IMG"
        },
        {
          "shortName": "PSNR"
        }
      ]
    }
  },
  "extensions": {}
}
```

Obrázek 5.9: V levé části je dotaz do GraphQL na autory se zkratkou všech metrik, kterými jsou autory a v pravé části odpověď z GraphQL.

```
1  query author($id: String!) {
2    author(id: $id) {
3      _id
4      titlesBefore
5      fnames
6      snames
7      titlesAfter
8      metricIds
9      metrics {
10       id
11       name
12     }
13     created
14     updated
15   }
16 }
```

```
{
  "data": {
    "author": {
      "_id": "5ca9a763e5d7d552a85e3491",
      "titlesBefore": "",
      "fnames": "Kryštof",
      "snames": "Plachetka",
      "titlesAfter": "",
      "metricIds": [
        "5ca9a763e5d7d552a85e348f",
        "5cb318d4bcb64b5718797ea3"
      ],
      "metrics": [
        {
          "id": "5ca9a763e5d7d552a85e348f",
          "name": "Genervani jednoho obrazku ze dvou vstupu"
        },
        {
          "id": "5cb318d4bcb64b5718797ea3",
          "name": "peak signal-to-noise ratio"
        }
      ],
      "created": "2019-04-07T07:31:47.602Z",
      "updated": "2019-04-27T19:26:57.275Z"
    }
  },
  "extensions": {}
}
```

Obrázek 5.10: V levé části je dotaz do GraphQL na autory s id a názvem všech metrik, kterými jsou autory a v pravé části odpověď z GraphQL.

GraphQL využívá pouze metodu POST (výjimečně GET), čím se liší od běžného REST API využívající všechny metody - GET, POST, PUT, DELETE a další. V těle dotazu typu POST má GraphQL uložen předpis pro získání dat. Jeho nevýhodou je vyšší složitost při práci se soubory, jako je nahrávání nebo stahování souborů, ale k tomu není primárně určené. Proto na našem serveru máme pro práci se soubory připravené REST API. GraphQL bylo vybráno hlavně kvůli typovosti a validaci dotazů. V článku o GraphQL [8], který jsme využili při startu tohoto projektu, jsou uvedeny další rozdíly oproti REST API.

- WebSocket[7]

  WebSocket je komunikační protokol, který využívá plně duplexní komunikaci přes jedno TCP spojení. To nám dovoluje živý přenos a tedy klientovi doručit zprávu, aniž by se na ní dotazoval. V naší aplikaci obdrží například frontendová část informaci o tom v jaké fázi se nachází analýza metriky a ta jí dá signál pro dotázání se na část výsledků. Uživatel díky tomu má rychlejší zpětnou vazbu a my jsme tím docílili postupného načítání výsledků.

  V následujícím bloku si ukážeme jednotlivé dotazy a jakým způsobem lze komunikaci škálovat. My jsme použili knihovnu sokcet.io[8] a tedy i následující dotazy jsou popsány touto knihovnou.

[7] WebSocket - https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
[8] Socket.io - WebSockety - https://socket.io

socket.emit('params') odešlou se data pouze klientovi
io.emit('params') odešlou se data všem klientům včetně odesílatele
socket.broadcast.emit('params') odešlou se data všem klientům kromě odesílatele
socket.broadcast.to('room').emit('params') odešlou se data všem klientům kromě odesílatele z místnosti room
io.in('room').emit('params') odešlou se data všem klientům včetně odesílatele z místnosti room
socket.to('room').emit('params') odeslání příjemnci jen pokud je v místnosti room
io.of('ABC').emit('params') odeslání všem příjemcům, které jsou v namespace ABC včetně odesílatele
socket.broadcast.to(socketId).emit('params') odeslání přímo na konkrétní socketId

Z důvodu komunikace s více klienty, kteří se dívají na stejnou analýzu jsme socket.io rozšířili o Redis[9], který se stará o to, aby zprávy byly jednotné napříč všemi procesy na serveru. Protože jsme narazili na problém, kdy jedno spojení běželo na jiném vlákně procesoru a při odeslání informace do místnosti nebyla zpráva doručena všem návštěvníkům.

- NodeMailer[10]

  NodeMailer je mailový modul Node.js uvedený pod MIT licencí, který zjednodušuje napojení na poskytovatele mailové brány. A obecně zjednodušuje manipulaci s odesíláním emailu. V aplikaci je využit pro informování klienta o dokončení analýzy, která byla odložena z důvodu časové náročnosti, jako je například zpracování videa nebo použití VQM metrik.

  Tato webová aplikace je v současnosti napojena na gmail účet, přes který NodeMailer zasílá zprávy. Maily obsahují html šablony s informacemi o aktuální analýze a odkaz přes který se na ní dostanou. Přístupy k mailové bráně lze upravit změnou konfiguračního souboru.

5.3 Vývojové prostředí

Než jsme začali implementovat webovou aplikaci, bylo nejdříve nutné zajistit prostředí pro vývoj. Sestavení vývojového prostředí nebylo vůbec jednoduché, protože u dostupných metrik převažují "matlabovské" implementace, které navíc využívají další grafické balíčky. Pro zprovoznění základních metrik jsme nainstalovali MATLAB[11] a k němu několik knihoven, jako je Pfstools[12] nebo ImageMagick[13]. Problém se vyskytl u jednotlivých knihoven, které

[9] Redis - serverový nástroj pro práci s pamětí - https://redis.io/
[10] NodeMailer - mailový modul Node.js - https://nodemailer.com/
[11] MATLAB - **mat**rix **lab**oratory - https://www.mathworks.com/products/matlab.html
[12] Pfstools - nástroje pro práci s HDR soubory - http://pfstools.sourceforge.net/
[13] ImageMagick - nástroje pro práci s obrazy - https://www.imagemagick.org/script/index.php

nejsou dostupné pro nejnovější verze linuxu, v našem případě byl otestován linux Mint 19.1, Ubuntu 16.04 a nakonec se podařilo zprovoznit dané balíčky na Ubuntu verzi 14.04.

MATLAB je software obsahující velké množství funkcí a metod pro různé matematické úkony. Pro výpočty se používá skriptovací jazyk. MATLAB zvládá jednoduché operace, pro různé algoritmy, 2D a 3D grafy funkcí. Dokáže analyzovat data a vytvářet aplikace s jednoduchým uživatelským rozhraním. Výhodou je možnost spustit tento nástroj i v příkazovém řádku a proto se perfektně hodí pro naší webovou aplikaci.

Pfstools patří mezi nejdůležitější nástroje pro práci s HDR, protože to je sada programů, které lze spouštět z příkazové řádky. Výhodou je i integrace přímo do MATLABu. Obrazy HDR umí nejen číst ale i zapisovat. Mezi základními funkcemi je čtení právě HDR obrazu a mapování tónů. Používáme aktuální verzi 2.1.0. [5]

ImageMagick je základní balíček pro práci s rastrovými obrazy. Umí mnoho funkcí, od nejzákladnější změny velikosti, filtrování až po úpravy bodů v obraze. Umí pracovat s různými datovými typy a mezi nimi převádět.

Také potřebujeme pracovat s videi a k tomu využíváme nástroj **FFmpeg**[14], což je kolekce volně šiřitelných funkcí pro práci s videi. Dovoluje nám převádět mezi jednotlivými datovými typy, odebírat z videa zvuk, komprimovat či jinými způsoby upravovat.

Řešili jsme i možnost nasazování pomocí Dockeru[15], ale za současného stavu to nebylo úplně možné z důvodu velkého množství balíčků, které se mohou nahrávat a využívat samotnou aplikací. Při práci s Dockerem byl například problém s instalací Matlabu. Ten je příliš veliký, aby byl uložen v jeho kontejneru. Další možností by bylo napojit Matlab jako službu, ale tím by přišel o balíčky, které budou do aplikace teprve do nahrány přes administrační prostředí. Bylo by vhodné časem doplnit dockerové řešení s instalací matlabu, které momentálně s dosavadními znalostmi není možné.

5.4 Zpracování obrazu a videa

Zpracování obrazu proběhne až po nahrání obrazu na server, ještě před tím, než na něj bude aplikována metrika. Pro zpracování obrazu a získání informací slouží právě výše zmíněný nástroj ImageMagick. Ten je přímo využíván ve zdrojovém kódu serveru a získává pro nás následující informace jako formát, šířka, výška, barevný prostor, typ, hloubka, komprese a velikost souboru. Aktuální verze si poradí i s HDR obrazy. ImageMagick nám také tvoří miniatury, pro ušetření stahovaných dat, protože HDR obrazy mohou mít až desítky megabajtů.

Obdobným způsobem probíhá zpracování videa. Při dotazu na informace z videa je využit výše zmíněný nástroj FFmpeg, který nám z videa dokáže získat následující informace: formát, délku, kodek, šířku, výšku, počet snímků za sekundu, bitrate a velikost souboru. Tento nástroj je dále využit při aplikaci VQM na sekvenci snímků, kdy je zapotřebí ze snímků vytvořit video. V dalším případě, kdy se aplikuje IQM na video, se musí video pomocí FFmpeg rozložit na jednotlivé snímky.

5.5 Komunikace aplikace

Aplikace se nám skládá z frontendové části napsané v Reactu a serverové části, kde je API. Server je postaven na schématu REST a GraphQL, dále obsahuje implementaci pro

[14] FFmpeg - kolekce funkcí pro práci s videem a audiem https://ffmpeg.org/
[15] Docker - izolování aplikace od prostředí - https://www.docker.com/

WebSockety. V následujících dvou podkapitolách bude vysvětleno, jak aplikace komunikuje s databázovou vrstvou pomocí stylizovaných dotazů GraphQL. Dále dotazování se na výsledky pomocí WebSocketů.

5.5.1 Práce s databází

Databáze se skládá ze 6 kolekcí a je postavena na principu API, které běží na MongoDB a to je obohaceno o GraphQL. Díky tomu jsou data poskytována přímo v požadovaném formátu. Všechny endpointy jsou zabezpečené a kontrolované, zda na ně má uživatel oprávnění.

Zápis do databáze je převážně z administračního prostředí, kde máme možnost upravovat všechny entity databáze přes patřičné formuláře. Z uživatelské části lze vkládat záznamy pouze do fronty s požadavky na analýzu a upravovat vlastní profil pro registrované uživatele.

Z databáze jsou pro uživatelskou část poskytovány veškeré informace kromě uživatelů a front. Obsah webové aplikace je dynamický, protože administrátor má možnost upravovat údaje a tím provést aktualizace frontendu, například u dokumentace metrik.

5.5.2 Práce s WebSockety

WebSockety jsou použity na získávání aktuálního stavu probíhající analýzy a princip je následující. Při otevření webové stránky s analýzou zažádá klient o přidání do místnosti, kde jsou všichni klienti, kteří chtějí být informování o změně stavu konkrétní analýzy. Název místnosti je unikátní pro každou analýzu a skládá se z heše analýzy. V průběhu analýzy server ukládá jednotlivé změny stavu analýzy do databáze a také informuje patřičnou místnost o změně stavu. To způsobí odeslání zpráv přes WebSockety všem klientům, kteří v dané místnosti byli.

Přes WebSockety komunikují i crony se serverem, kde komunikace musí být zabezpečená a probíhá v rámci localhostu. Cron informuje server, které analýzy může spustit. Logika analýzy by mohla být umístěna i na straně cronu, abychom neřešili duplicitní zdrojový kód a optimalizovali ho na jednom místě. V následujícím seznamu si sepíšeme, jaké zprávy cron a klient přijímají a které naopak odesílají. Server neuvádíme, protože přijímá to, co klient s cronem odesílají a odesílá to, co klient s cronem přijímají.

Klient

- **Odesílá:**
  - *JOIN_TO_ROOM* - Připojení do místnosti.
  - *LEAVE_TO_ROOM* - Odchodu z místnosti.
- **Přijímá:**
  - *ANALYSIS_START* - Začátek analýzy.
  - *ANALYSIS_METRICS_PROGRESS* - Analýza konkrétní metriky.
  - *ANALYSIS_FILES_PROGRESS* - Analýza konkrétních souborů.
  - *ANALYSIS_END* - Ukončení analýzy.
  - *SYSTEM_CPU* - Využití procesoru serveru.
  - *SYSTEM_MEMORY* - Využití operační paměti serveru.
  - *SYSTEM_U_SIZE* - Informace kolik složka *uploads* zabírá místa na serveru.

CRON

- **Odesílá:**
  - *ANALYSIS_START* - Pokyn pro začátek analýzy.
- **Přijímá:**
  - *ERROR* - Neoprávněný přístup.

5.6 Cron

Cron plánuje v aplikaci spouštění dvou skriptů. První skript slouží ke spouštění odložených analýz a druhý nám udržuje pořádek v souborovém systému. Cron si načítá informace z konfiguračních souborů, podle kterých zjistí kdy a jaký skript spustit. Není problém jednoduchým zásahem upravit intervaly spouštění skriptů.

Jako první si rozebereme hlídání aktuálně běžících analýz a spouštění analýz, které čekají ve frontě. Ve výchozím nastavení dovolujeme v jednu chvíli běžet maximálně třem odloženým analýzám. Skript začíná vyhledáním všech analýz, které aktuálně běží. Poté si vyhledá nejstarší tři analýzy, které čekají ve frontě. Dále si musí spočítat aktuálně běžící analýzy, které si vyhledal a pokud je možné spustit další, tak přidá další z vyhledaných čekajících analýz. Spuštění analýzy probíhá pomocí komunikace přes WebSockety, pro případ, kdy by se skripty spouštěly samostatně v jiném vlákně. Protože by mohlo dojít k podvržení spojení a spouštět libovolné analýzy útočníkem, musí se skript na začátku své relace přihlásit na server pomocí bezpečnostního tokenu, aby měl možnost spouštět analýzy. Tento skript se spouští každých deset minut.

Cron starající se o stav serveru, a to především o hlídaní velikosti složky s nahranými daty, průběžně maže po pevně definované době staré soubory. Ve výchozím nastavení jsme tuto dobu nastavili na sedm dní. Cron tedy spustí skript, který se připojí k databázi, vyhledá si všechny analýzy, které nejsou dokončené nebo čekají na vyřízení a nebo jsou vyřízené, ale neuplynula doba sedmi dní. Tímto získáme seznam analýz, u kterých nesmíme mazat jejich soubory. V nahraných souborech a výsledných analýzách vybereme seznamy složek ke smazání. Pokračuje smazáním všech souborů, které nemají v názvu heš z analýz, které nepřijdou smazat. Tímto docílíme pročištění nahraných dat od těch, které už nebudeme potřebovat. Skript na mazání se bude pouštět jednou denně, ale administrátor může tuto akci urychlit z administrace a spustit skript okamžitě.

5.6.1 Práce se soubory

V následující části si rozebereme práci na úrovni file-systému serveru. Ze strany aplikace se na server nahrávají obrazy a videa, které jsou náročné na místo na disku. Z administračního prostředí se na server ukládají data metrik a různých knihoven. Samotný server pracuje s obrazy, které skládá do videa a nebo s videi, které rozkládá na jednotlivé snímky. Dále se vytváří ke každému obrazu nebo videu miniatura, pro snížení přenosových dat pro klienta.

Soubory na serveru jsou všechny ve složce *uploads* rozděleny do následujících složek.

- *metrics/[metricId]/metric/** - soubory metriky
- *metrics/[metricId]/script/script.sh* - skript metriky
- *profiles/[profileId]/[hash + namefile]* - profilová fotka uživatele

- *user_data/[analyse_hash]/src/** - zdrojové soubory analýzy
- *user_data/[analyse_hash]/src/min/** - miniatury zdrojových souborů analýzy
- *user_data/[analyse_hash]/ref/** - referenční soubory analýzy
- *user_data/[analyse_hash]/ref/min/** - miniatury referenčních souborů analýzy
- *user_results/[analyse_hash]/[metric_id]/[file_unique_id]/** - výsledky z analýz jednotlivých souborů

Unikátní heš analýzy má formát $YYY\_MM\_DD\_hh\_mm\_ss + uniqueId$ a je zvolen pro lepší práci a řazení složek v souborovém systému podle datumu. Tento heš je poskytnut uživateli, pro kterého je čitelný.

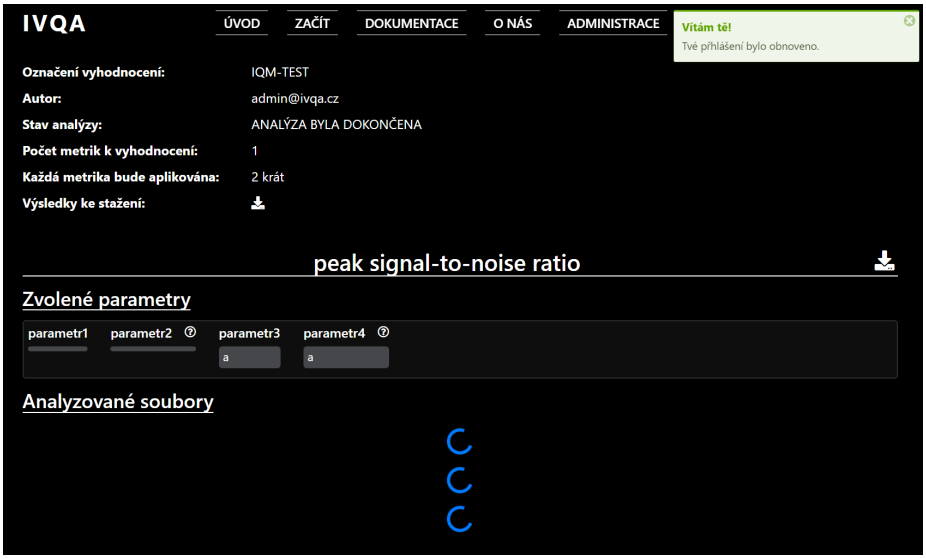
5.7 Dynamika aplikace

Díky WebSocketům je u aplikace zajištěna okamžitá odezva ze serveru u důležitých událostí jako je sledování stavu zpracování. Administrátor má k dispozici vždy aktuální informace a to především hodnoty vytížení serveru a informace o jeho zaplnění.

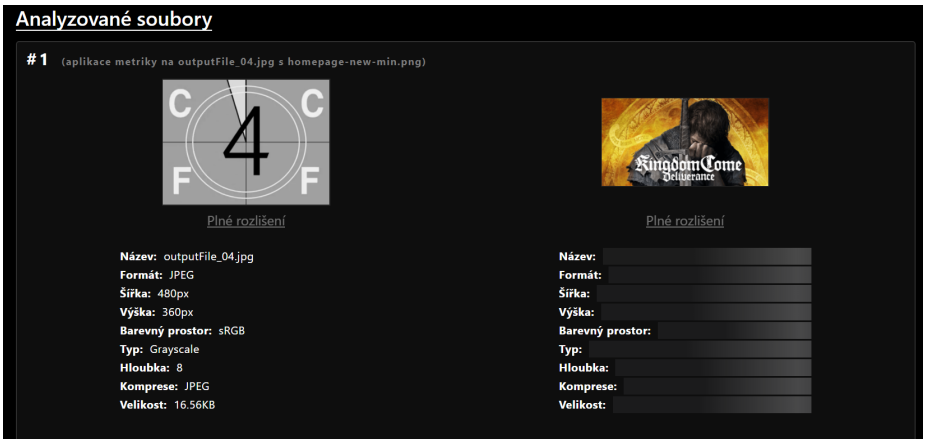
Dynamika aplikace bude také docílena díky vývoji aplikace na ReactJS, který zajišťuje okamžitou aktualizaci dat na webu a to jenom té části, kde se DOM[16] (prezentace webové stránky) liší od virtuálního DOM Reactu. Další výhoda spočívá při přechodu mezi jednotlivými stránkami aplikace, kdy není nutné načítat celou stránku, ale pouze se překreslí změněné části DOMu. Aplikace ví, jak má vypadat, pouze se serveru doptává na informace, které má zobrazit. Tímto se odlišuje od běžné webové aplikace implementované v PHP, která se doptává vždy na celou stránku.

Kvůli velkému množství informací a datové náročnosti, protože se pracuje s grafikou, je snaha co nejvíce zapojit postupné načítání. To můžeme vidět na následujících dvou obrázcích (5.11 a 5.12), kdy na prvním se načítají jednotlivé analyzované soubory, na druhém jsou vlevo už donačtené informace o obrazu a vpravo ještě načítání probíhá.

[16] DOM - Document Object Model - https://cs.wikipedia.org/wiki/Document_Object_Model



Obrázek 5.11: Postupné načítání analyzovaných souborů při analýze, která byla už dokončená.



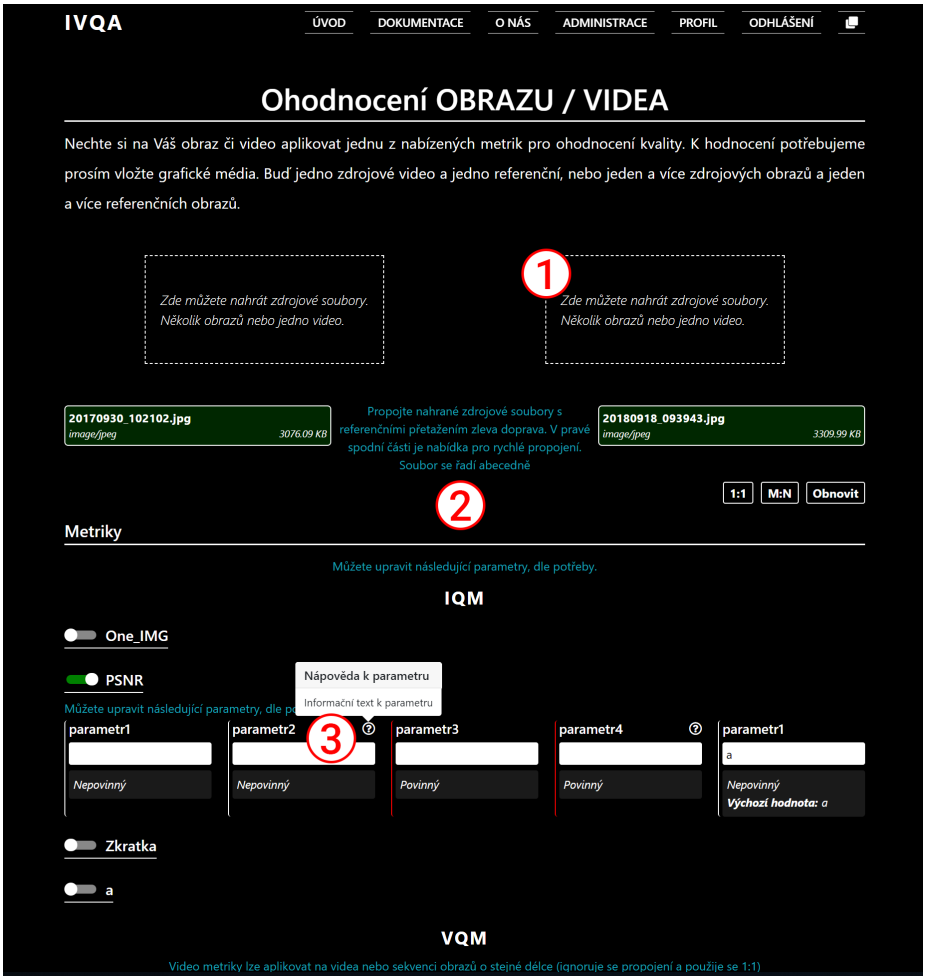
Obrázek 5.12: Postupné získávání informací o obrazech, kdy vlevo jsou už načtené a vpravo je stále aktivní načítání.

5.8 Uživatelský pomocník

Webová aplikace pro hodnocení kvality obrazu a videa by měla být dostupná i pro širší veřejnost, a proto jsme se snažili postavit aplikaci schopnou poskytnout co největší pohodlí. Je to formou nápověd s informacemi o tom, co k čemu slouží, jaké názvy co znamenají a především, co daná věc způsobí pokud je zvolena.

Tyto informace (viz. obrázek 5.13) jsou převážně poskytovány při najetí na otazník. Ty jsou pak zobrazeny na místech, které jsme v rámci testování uznali za vhodné. Cílem bylo, aby nápovědy nerušily zběhlé uživatele a zároveň se nový uživatel neztratil. Jako je například situace po nahrání zdrojových a referenčních obrazů, kdy je uživatel vyzván pro označení obrazů jejich propojením, aby aplikace věděla, které má vůči sobě vyhodnotit. Toto mu je také ulehčeno rychlým menu, které nabízí propojení souborů **1:1** (propojí se

první s prvním, druhý s druhým atd.), **M:N** (propojí se každý s každým) a také je zde možnost odebrat všechny propojení.



Obrázek 5.13: Konfigurace analýzy, kde bod 1 poukazuje na prostor určený pro nahrávání, kde je nápověda vepsána přímo v daném prostoru. Bodem 2 je označena nápověda, co se má provést s nahranými soubory, tedy je potřeba je propojit. Bod 3 je umístěn u parametrů a poukazuje nám na otazník, který skrývá nápovědu pro daný parametr.

Snažíme se uživateli ušetřit práci. Například uživatel při analýze videa musí vložit jedno zdrojové a jedno referenční video do dvou oken k tomu určených. Po úspěšném nahrání jednoho videa daným oknem/komponentou je okno skryto. Po nahrání druhého videa je i druhé okno skryto a soubory jsou automaticky propojeny 1:1.

Pokud uživatel analyzuje sekvenci souborů, tak jsou soubory po vložení řazeny podle názvu. Zde bylo potřeba upravit řazení, protože pokud řadíme prostý řetězec, tak se čísla berou jako znaky (*image2* by bylo až za *image10*), protože dvojka je větší jak jednička. Tento problém jsme vyřešili rozdělením názvu na textovou část a číselnou část. Díky tomu můžeme řadit čistě jen podle textové části a pokud je shodná, řadí se podle řetězcové části. Viz přiložená ukázka implementace řazení 5.4.

```
1  files.sort((a, b) => {
2      const aStr = a.name.replace(/[0-9]/g, '');
3      const bStr = b.name.replace(/[0-9]/g, '');
4      const aNum = parseInt(a.name.replace(/[^0-9]/g, ''), 10);
5      const bNum = parseInt(b.name.replace(/[^0-9]/g, ''), 10);
6      if (aStr > bStr) return 1;
7      if (aStr < bStr) return -1;
8      if (aNum > bNum) return 1;
9      if (aNum < bNum) return -1;
10     return 0;
11 }),
```

Výpis 5.4: Ukázka řazení stringu, který obsahuje číselnou část označující pořadí pořadí.

5.9 Bezpečnost aplikace

Zaměření na bezpečnost je nedílnou součástí všech webových aplikací a hlavně u takových, které mají administrační prostředí a nebo je nahráván uživatelský obsah. V našem případě to je jak administrační prostředí, kde má administrátor přístup k metrikám, tak soubory nahrávány uživateli pro vyhodnocení v metrikách.

K zabezpečení komunikace se využívá JWT token. JWT token, neboli JSON web token, je otevřený standard (RFC 7519), který definuje kompaktní a samostatný způsob bezpečného přenosu informací mezi klientem a serverem jako objekt JSON [3]. Tento token je podepsán protokolem RSA pomocí veřejného a soukromého klíče.

Bezpečnost administrace, respektive dotazů do API, které může provádět pouze administrátor je zajištěna tokenem. Ten nese informace o ID uživatele, emailu a jeho roli. Obsažené informace jsou zašifrovány algoritmem RS256 a jsou zasílány v hlavičce dotazu v *authorization* s prefixem *Bearer*. Token není přenášen v sessions, protože by mohl byt zneužit při útoku CSRF (Cross-site Request Forgery). API GraphQL na začátku každého dotazu kontroluje validitu tokenu a následně načte údaje o uživateli do kontextu, aby s ním mohla aplikace pracovat, jinak s uživatelem pracuje jako s nepřihlášeným. Na každém endpointu je také kontrola uživatelského oprávnění. Nestane se nám, aby se neautorizovaný uživatel úspěšně dotázal na endpointy, které slouží pouze pro administraci, jako je mazání uživatelů, editace metrik apod.

Uživatelská data jsou zabezpečena také JWT tokenem. V něm je zašifrována algoritmem RS256 heš analýzy a email uživatele. Tentokrát si můžeme token dovolit uložit do sessions, protože po případném odposlechnutí by musel útočník znát také heš a to by mělo útočníka odradit. Sessions je použita, aby mohly být do stránky načítány obrázky, nebo stahovat soubory, které jsou zabezpečené. Token má nastavenou platnost pouze na 7 dní. Poté musí být přegenerován a tuto možnost mají pouze registrovaní uživatelé. Při vytvoření analýzy, je první token zaslán na email. Přes něj dostává přístup k výsledkům i nepřihlášený uživatel.

5.10 Konfigurovatelnost aplikace

Pro každou aplikaci je důležitá konfigurovatelnost, kterou my řešíme přes proměnné z prostředí. Ty nastavujeme díky balíčku dotenv[17]. Ten drží konfiguraci oddělenou od aplikace

[17] Dotenv - balíček pro oddělení konfigurace od aplikace - https://github.com/motdotla/dotenv

v souboru *.env*. V tomto souboru je zápis konfigurace, kterou dotenv zpracuje a nastaví *process.env*. Díky tomu můžeme přistupovat v aplikaci k těmto proměnným. Ukázku výchozího nastavení konfiguračních souborů nalezneme v příloze B.

Konfigurační soubor serverové části obsahuje číslo portu serveru, přístupy do databáze Monga, přístupy do keše Redisu, RSA klíče, zapnutí mock serveru, přístupy k emailové bráně, adresu webu, konfigurace cronu (kdy se mají spouštět jednotlivé skripty), název balíku pro stažení analýzy, maximální počet běžících analýz a debug režim.

V aplikační části je to trochu složitější, protože výsledná aplikace běží na straně klienta. Zde nejsou dostupné proměnné z prostředí serveru a tedy konfiguraci je potřeba nastavit ještě před odesláním aplikace uživateli. S tím nám pomůže webpack, který běží na straně serveru a tím má přístup k proměnným z prostředí. Webpack nám už jen nastaví konstanty, které budeme mít dostupné v aplikaci. Konfigurace klientské aplikace je především z důvodu vývoje, kdy chceme například měnit adresu API na lokální server. Konfigurace obsahuje název aplikace, email, klíč k editoru TINY MCE, adresu REST API a GraphQL API, dobu zobrazení notifikace, umístění notifikace a vývojové konstanty pro synchronizaci prohlížeče.

Kapitola 6

Testování

V této kapitole popisujeme testovací prostředí, které bylo zavedeno pro jednodušší vývoj a udržovatelnost aplikace. Odhalení chyb při vývoji je jednodušší, než opravovat škody vzniklé po nasazení projektu. V základu jsme použili dva typy testů. První Unit testy nebývají náročné a označují se za ty nejdůležitější, které by měla mít každá aplikace. Dalším typem jsou integrační testy, které jsme zavedli pro otestování složitějších funkcí a vzájemné komunikace jednotlivých komponent. V průběhu finalizace vývoje bylo prováděno i uživatelské testování, které nám pomohlo doladit především vizuální část aplikace.

6.1 Funkční část

Pro testování funkčních částí byl použit balíček Jest[1], který vyvinula společnost Facebook přímo pro React. Tento nástroj umí pracovat i s asynchronními testy, je jednoduchý, má rozsáhlou komunitu a poskytuje podporu při řešení problémů. K dnešnímu dni má přes tři a půl milionu aktivních uživatelů. V Jestu budeme psát hlavně Unit testy [1].

Pro psaní testu stačí do projektu umístit libovolný soubor s koncovkou **.test.js* nebo pokud umístíme test do složky s názvem *__tests__*. Následně můžeme v konzoli spustit testy a získáme přehledný výstup, jak uvidíme dále. Pro testování komponent bylo potřeba doplnit rozšíření Enzyme[2], které nám zpřístupní testování vizuálních složek jednotlivých komponent. Příkladem je například test, zda existuje tlačítko s třídou *my-link* a simulovat jeho stisknutí.

Testovací skripty jsou postaveny především pro otestování správného zobrazení přijatých dat ve webové aplikaci a funkcí, které zpracovávají data. Dále jsou testovány jednotlivé dotazy na API, aby nedocházelo k nepřiřazeným záznamům nebo ke špatně zadaným záznamům. V poslední řadě byly vytvořeny i testy pro správnou funkci cronu, který se stará o data uložená na disku, kdy by mohlo dojít ke kolizi a pádu aplikace například v případě špatně smazaných souborů.

Pro frontendovou část můžeme vidět výsledky v tabulce 6.1 a pro serverovou část v tabulce 6.2. Vidíme zde, že ve finální verzi proběhly všechny testy bez chyby.

[1] Jest - testovací prostředí javascriptu - https://jestjs.io/

[2] Enzyme - testování "Reactích" komponent - https://github.com/airbnb/enzyme

Testování aplikační části			
Test Suites:	**0 failed**	**15 passed**	15 total
Tests:	**0 failed**	**32 passed**	32 total
Snapshots:	**0 failed**	**21 passed**	21 total
Time:	11.547s		

Tabulka 6.1: Výsledky testovacího prostředí na frontendové části aplikace.

Testování serverové části			
Test Suites:	**0 failed**	**10 passed**	10 total
Tests:	**0 failed**	**23 passed**	23 total
Snapshots:	**0 failed**	**1 passed**	1 total
Time:	8.376		

Tabulka 6.2: Výsledky testovacího prostředí na serverové části aplikace.

6.2 Aplikační část

Vizuální stránku aplikace je nutné otestovat i u běžných uživatelů, aby sami vyjádřili svůj názor na umístění jednotlivých objektů na webu. Díky tomu se můžou nějaké věci přesunout, přidat nebo dokonce odebrat, pokud by se při testování přišlo na něco, co by působilo rušivým dojmem a nemělo to vlastní význam.

Toto testování proběhlo v podobě, kdy bylo vybráno několik uživatelů, kteří si aplikaci zkoušeli. Jednotlivé poznatky se zaznamenávaly a poté konzultovaly se všemi uživateli najednou. Při testování byl kladen důraz na průchod uživatele skrze celou analýzu, od nahrávání přes konfiguraci až po zobrazení výsledků. V následující tabulce je přehled otázek, které přinesli nějaký vývoj v aplikaci. V tabulce je heslovitě zachycena spokojenost uživatele, kde v pravém sloupci je zapsán závěr testování. Připomínáme, že testování probíhalo v průběhu vývoje, tedy dané připomínky byly zavedeny do současné verze.

- **Nahrávání souborů**

  U1: Zvětšil bych prostor pro nahrávání.

  U2: Co vše lze nahrávat?

  U4: Pokud už nelze nic nahrát, skryl bych tu možnost.

  Závěr: Prostor pro nahrávání byl zvětšen, a doplněn o informaci, kdy se nahrává obraz a kdy video.

- **Výběr/propojení souborů**

  U1: Co když toho nahraji hodně, to musím vše po jednom označovat?

  U3: Jak mohu odebrat spojení?

  Závěr: Doplněno menu s výchozími možnostmi propojení 1:1, M:N a smazání všeho, dále přidáno mazání kliknutím na propojení.

- **Konfigurace metrik**

  U3: Doplnil bych do přehledu, kolik metrik jsem navolil a kolik souborů se zpracovává.

Závěr: Doplněno dle připomínky.

- **Nápovědy**

  U3: Nápovědy vypadají dobře, ale zobrazoval bych je při najetí na otazník, ne až na kliknutí.

Závěr: Doplněno dle připomínky.

- **Chybové hlášky**

  U1: Doplnil bych srozumitelnější text, co mám dělat v případě chyby.

  U4: Pokud nahraji špatný soubory, rád bych o tom věděl.

Závěr: Texty odladěny a chybové soubory jsou vidět v notifikacích po validaci.

- **Přehled analýzy**

  U1: Nejsem si jistý, co k čemu patří.

  U2: Metriky bych od sebe oddělil nějak viditelněji.

Závěr: Plochy podbarveny odlišnou barvou s ohraničením a dále jsme metriky očíslovali.

- **Postupné načítání**

  U3: Bylo by dobré vidět, co očekáváme při načítání.

Závěr: Doplněný skeleton loading pro jednotlivé záznamy, tedy vidíme počet a typ výstupů metrik.

- **Výstupy z analýzy**

  U1: Doplnil bych možnost zobrazení v novém okně, například u obrázků.

Závěr: Všem obrázkům byla přidána možnost zobrazit v plném rozlišení.

Kapitola 7

Závěr

Diplomová práce navazuje na semestrální projekt, ve kterém jsme provedli analýzu projektu, prostudovali dosavadní řešení a připravili si vývojové prostředí pro následný vývoj aplikace. Zde jsme narazili na překážku vzniklou grafickými balíčky a obtížnou instalací MATLABu. Při návrhu architektonického a datového schématu celé aplikace jsme si ujasnili, které nástroje budeme potřebovat a jak budou jednotlivé celky mezi sebou komunikovat. V dalším kroku jsme si definovali, které uživatelské role budou v naší aplikaci existovat, abychom mohli připravit model zabezpečení aplikace.

V úvodu práce bylo zapotřebí se obeznámit s doposud neznámými grafickými balíčky a nastudovat si jejich základní použití. Nejprve jsme implementovali základní model API, ke kterému jsme následně sestavili administraci. V iteracích jsme implementovali metody, které jsme v administraci potřebovali. Po dokončení administrace jsme se mohli pustit do frontendové aplikace. Při implementaci konfigurace analýzy bylo nutné navrhnout, jakým způsobem bude uživatel volit soubory, které budou spolu předány metrikám. Nakonec jsme zvolili interaktivní volbu, v které si uživatel jednotlivé soubory propojuje.

Po zhotovení implementace odeslání souborů na analýzu a uložení konfigurace spuštění analýzy, jsme pokračovali přípravou aplikování metrik. Nejdříve jsme si zanalyzovali data, která nám poskytl uživatel. Zde jsme využili nastudované grafické balíčky, kde jsme pomocí nástroje ImageMagick získali informace o obrazu a pomocí nástroje FFmpeg jsme zanalyzovali videa. Pro samotnou aplikaci metrik jsme si definovali všechny cesty k zdrojovým i referenčním souborům ke složce s výsledky a k samotné metrice. Konfigurační schéma jsme si připravili a rozdělili na jednotlivé soubory, které spolu budou předávány metrice, respektive shell skriptu, který bude s metrikou pracovat. Následně bylo zapotřebí z aplikace spustit konzolový příkaz, zachytit výstup a na případné chyby nějakým způsobem reagovat. To se nám povedlo, a po získání prvních výstupů jsme se mohli vrátit k frontendové části a implementovat prezentování výstupů metrik. Hlavním cílem bylo přehledné zobrazení různorodých výstupů metrik, který jsme vyřešili doplněním o popis výstupu, kdy administrátor definuje konkrétní výstupy. Při práci s videem, kdy se může generovat velké množství snímků, byly výstupy doplněny ještě o komprimovaný formát dat typu ZIP. Výstup metrik jsme implementovali modulárně, tak aby bylo možné v administraci pro každou metriku nastavit vlastní šablonu, která bude prezentovat její výsledky. Pro vývoj a počáteční používání nově nasazených metrik jsme implementovali i univerzální šablonu, která prezentuje všechny výsledky nadefinované v metrice.

Další rozšíření, které bylo implementováno, je možnost kombinace metrik, v rámci kterého lze aplikovat IQM na videa a opačně VQM na sekvenci snímků. Zde jsme opět museli prostudovat práci s videem, protože jsme potřebovali rozložit video na sekvenci snímku

tak, abychom na jednotlivé snímky mohli aplikovat IQM. Opačně tak u druhé možnosti, kde bylo potřeba ze sekvence snímků vytvořit videa, které jsme následně poskytli VQM metrice a zachovali tak původní očekávaný vstup shell skriptu.

Prototyp Webové aplikace pro hodnocení kvality obrazu a videa jsme se serverovou částí implementovali a doladili jsme všechny chyby a připomínky, na které jsme narazili při testování. Budoucí rozšíření tohoto projektu shledáváme v implementaci více šablon pro jednotlivé metriky a uvedení do praxe ve výzkumné skupině ústavu počítačové grafiky, kde bude plnohodnotný MATLAB. Následně je možno aplikaci využít mimo jiné při výzkumu nových komprimačních algoritmů, nebo optimalizace programů, které mění datový typ či jiným způsobem manipulují s obrazy a videi.

Literatura

[1] *Testing in React with Jest and Enzyme: An Introduction. 2012, [Online; navštíveno 26.2.2019].*
URL https://medium.com/@rossbulat/testing-in-react-with-jest-and-enzyme-an-introduction-99ce047dfcf8

[2] *Peak Signal-to-Noise Ratio as an Image Quality Metric. 2018-12-07, [Online; navštíveno 23.12.2018].*
URL http://www.ni.com/white-paper/13306/en/

[3] *Auth0.com: JSON Web Token. 2013, [Online; navštíveno 03.02.2019].*
URL https://jwt.io/introduction/

[4] *Myszkowski, K.; Seidel, H.-P.; Aydın, T.; aj.: Dynamic Range Independent Image Quality Assessment. [Online; navštíveno 23.12.2018].*
URL http://resources.mpi-inf.mpg.de/hdr/vis_metric/

[5] *Rafal, M.: Pfstools. [Online; navštíveno 24.12.2018].*
URL http://pfstools.sourceforge.net/index.html

[6] *Rafal, M.: HDR-VDP. 2015, [Online; navštíveno 23.12.2018].*
URL http://hdrvdp.sourceforge.net/wiki/

[7] *Reinhard, E.: High dynamic range imaging. Burlington, MA: Morgan Kaufmann/Elsevier, druhé vydání, c2010, ISBN 978-0-12-374914-7.*

[8] *Scott, J. R.: How to build a full GraphQL server with Node.js. 2014, [Online; navštíveno 08.12.2018].*
URL https://medium.freecodecamp.org/graphql-zero-to-production-a7c4f786a57b

[9] *Wieruch, R.: How to create a REST API with Express.js in Node.js. 2019, [Online; navštíveno 15.01.2019].*
URL https://www.robinwieruch.de/node-express-server-rest-api/

Příloha A

Obsah přiloženého paměťového média

Adresářová struktura CD je následovná:

- **app** Zdrojové soubory webové aplikace
- **api** Zdrojové soubory serveru s API
- **dokumenty** Dokumenty
    - **DP** Text DP
    - **app dokumentace** Dokumentace k webové aplikaci
    - **api dokumentace** Dokumentace k serveru s API
- **plakát** Plakát ve formátu A1

Příloha B

Konfigurační soubor

B.1 Konfigurace API serveru

```
SERVER_PORT=9000
SERVER_MOCK=false
ADMIN_TEST_TOKEN=
SYSTEM_WS_NAME_ROOM=SYSTEM-INFO
SYSTEM_CPU_WS_INTERVAL=5000
SYSTEM_MEM_WS_INTERVAL=5000
SYSTEM_U_DIR_WS_INTERVAL=5000
DB_LOCAL=true
DB_HOST=
DB_USER=root
DB_PASS=
DB_PORT=37643
DB_NAME=diplomka-api
DB_URI="mongodb://localhost:27017/diplomka-api"
REDIS_HOST=localhost
REDIS_PORT=6379
REDIS_PASSWORD=
DEBUG=true
SEND_EMAIL=false
JWT_SECRET_PRIVATE=
JWT_SECRET_PUBLIC=
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=465
EMAIL_SECURE=true
EMAIL_AUTH_USER=
EMAIL_AUTH_PASS=
EMAIL_FROM=
WEB_URL=http://localhost:3000
API_URL=http://localhost:9000/graphql
DOWNLOAD_ANALYSIS_NAME=analysis
MAX_RUNNING_ANALYSIS=3
CRON_NAME_1=CRON1
```

```
CRON_1_INTERVAL="*/10 * * * * *"
CRON_NAME_2=CRON2
CRON_2_INTERVAL=" */30 * * * *"
CRON_2_DELETE_INTERVAL=7
CRONS_RUN=true
```

Výpis B.1: Příklad konfiguračního souboru pro API

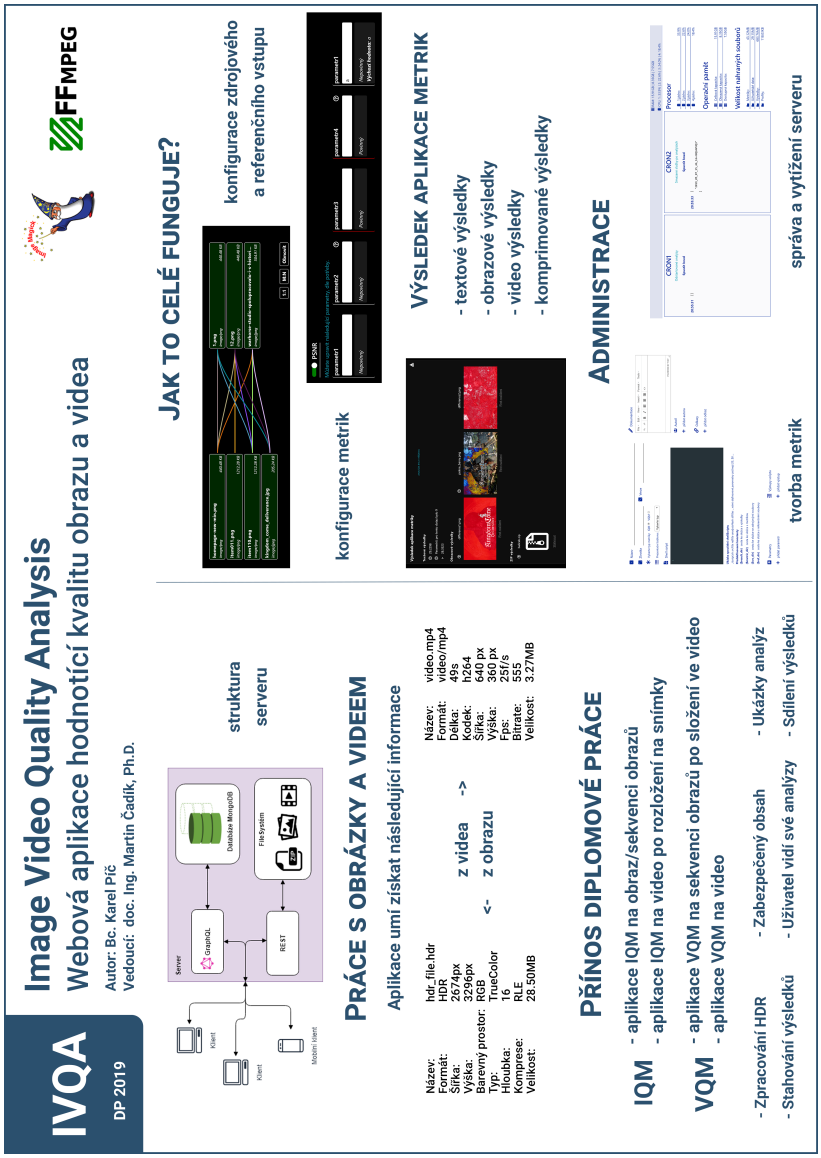
B.2 Konfigurace frontendové aplikace

```
APP_NAME=ivqa
APP_EMAIL=
API_KEY_TINY_MCE=
API_URL=http://localhost:9000
API_URL_GRAPHQL=http://localhost:9000/graphql
NOTIF_AUTO_DISMISS=5
NOTIF_POSITION=tr
FORM_MIN_LENGTH_PASSWORD=6
DEV_BROWSER_SYNC_URL=http://localhost:8000/
DEV_BROWSER_SYNC_PORT=3000
DEV_BROWSER_SYNC_HOST=localhost
SYSTEM_WS_NAME_ROOM=SYSTEM-INFO
SYSTEM_LENGTH_HISTORY=10
```

Výpis B.2: Příklad konfiguračního souboru pro frontend.

Příloha C

Plakát



Obrázek C.1: Náhled plakátu. Originál na CD.

Příloha D

Postup instalace a spuštění aplikace

1. Připravení prostředí pro aplikaci
   1.1. Instalace FFmpeg
   1.2. Instalace ImageMagick
   1.3. Instalace Redis - je potřeba redis-server
   1.4. Instalace MongoDB
   1.5. Instalace NodeJS (možnost použít npm nebo balíček yarn)
2. Instalace API
   2.1. Stažení API z přiloženého CD
   2.2. Instalaci API: *yarn install*
   3.2. Kompilace API: *yarn build*
   2.3. Spuštění inicializačního skriptu pro DB: *yarn init*
3. Instalace frontendové části
   3.1. Stažení app z přiloženého CD
   3.2. Instalace app: *yarn install*
   3.2. Kompilace app: *yarn build*
4. Spuštění API
   4.1. Spuštění redis-serveru (externí aplikace)
   4.2. Spuštění API *yarn start*
5. Spuštění frontendové části pomocí *yarn start*

Volitelně:

1. Připravení prostředí pro metriky
   1.1. Instalace MATLAB

1.2. Instalace Python dle potřebné verze metrik
   - Plus instalace další knihoven, které vyžadují konkrétní metriky

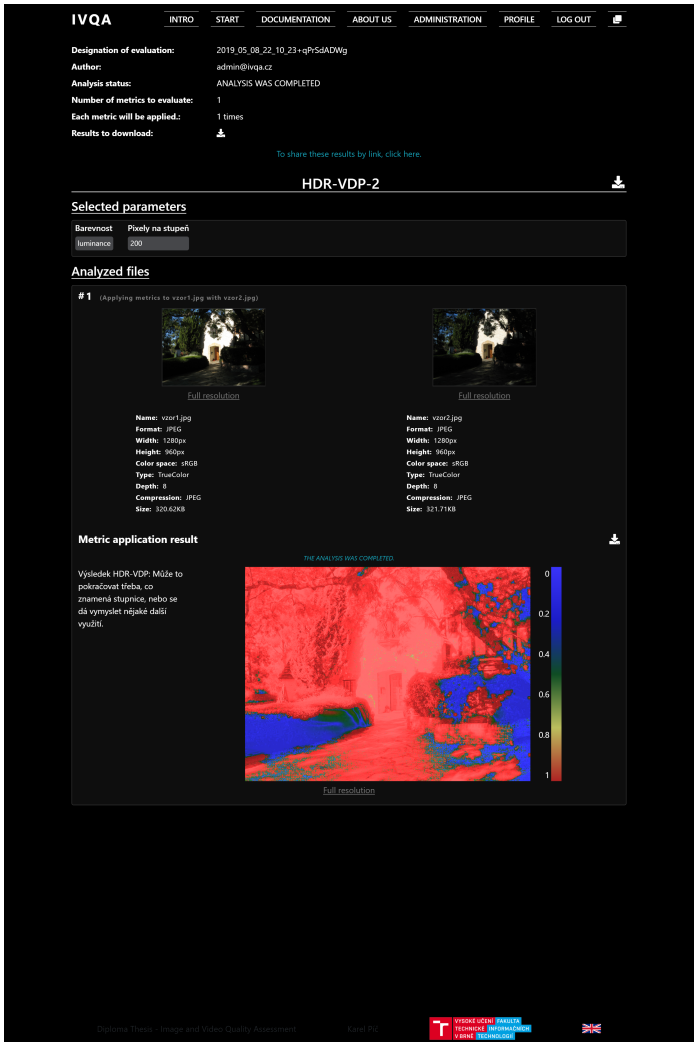
2. Generování překladu pro frontendovou část

   2.1. Generování překladu z tabulky typu excel *yarn build:xlsToJson*
   - podrobnější popis práce s překlady je v dokumentaci aplikace

Příloha E

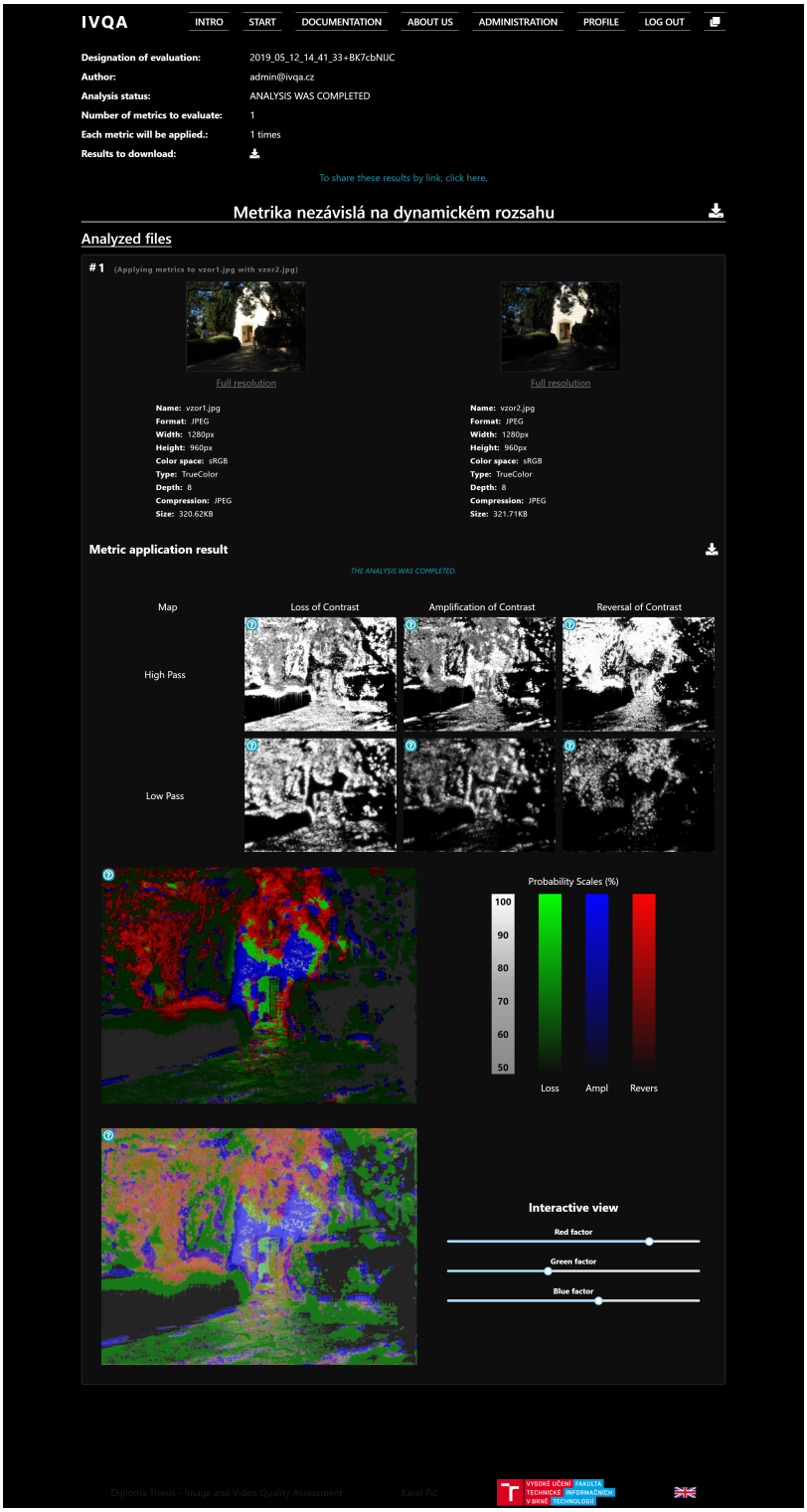
Šablony metrik

E.1 Šablona pro metriku HDR-VDP



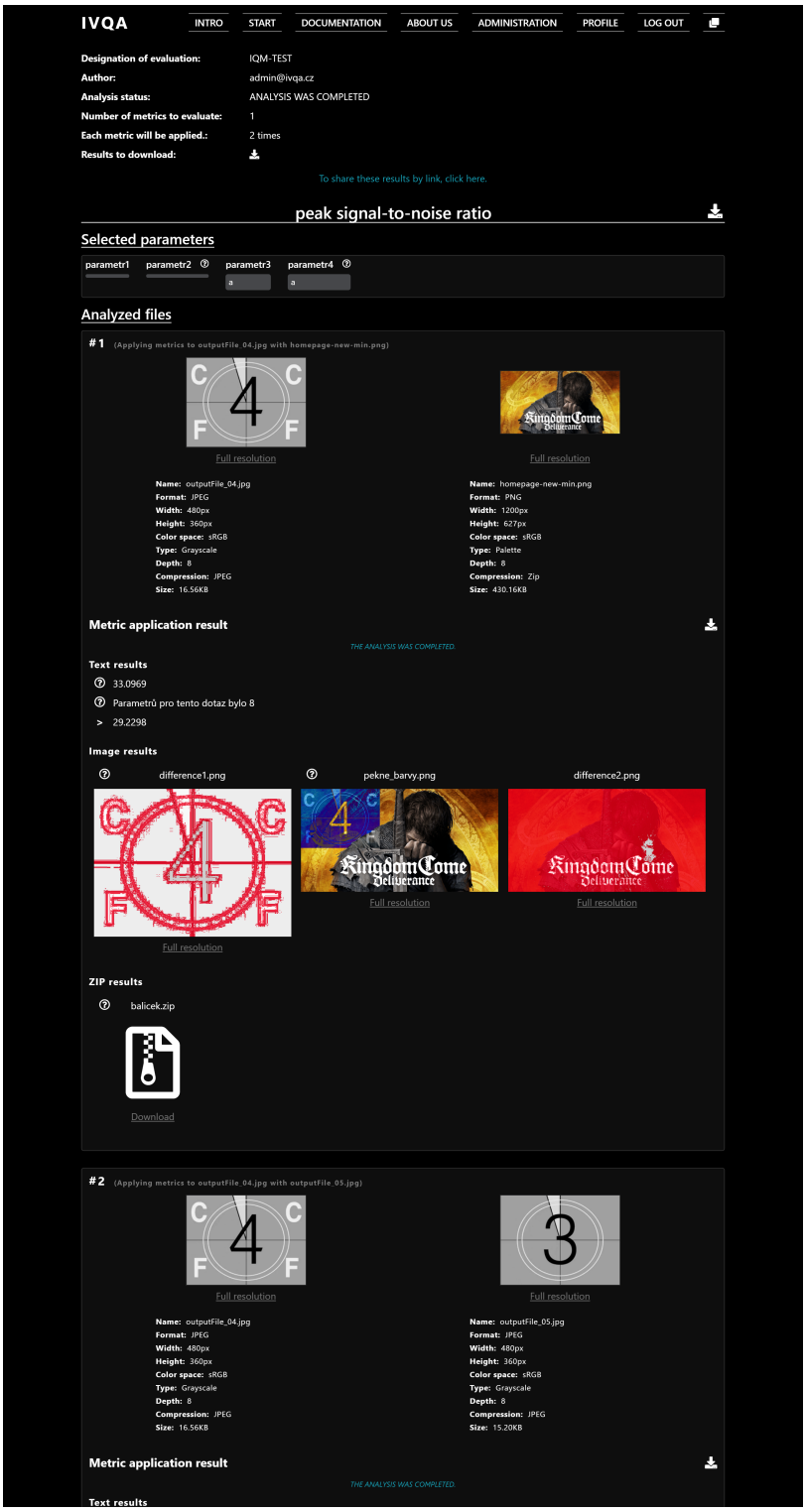
Obrázek E.1: Šablona pro metriku HDR-VDP. Zobrazuje jediný grafický výstup s vysvětlením a stupnicí, pro lepší pochopení obrazu.

E.2 Šablona pro metriku DRIM



Obrázek E.2: Šablona pro metriku DRIM. Zobrazuje tabulku složenou z výstupů metriky a interaktivní obraz, u kterého lze manipulovat s jednotlivými faktory.

E.3 Univerzální šablona



Obrázek E.3: Univerzální šablona pro metriku. Zobrazuje libovolné textové, obrazové, komprimované a video výstupy. V tomto případě šablona zobrazuje zpracování dvou analýz.