VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Pavel Šeda

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

MOBILNÍ APLIKACE PRO SUBJEKTIVNÍ MĚŘENÍ KVALITY ZÁŽITKU STREAMOVANÉHO VIDEA
MOBILE APPLICATIONS FOR SUBJECTIVE MEASUREMENT QOE OF STREAMING VIDEO

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Pavel Šeda

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Dominik Kováč

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Pavel Šeda ***ID:*** 154208
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Mobilní aplikace pro subjektivní měření kvality zážitku streamovaného videa

POKYNY PRO VYPRACOVÁNÍ:

Práce se bude zabývat subjektivním měřením kvality zážitku streamovaných videí pomocí mobilní aplikace. Teoretická část práce se bude zabývat problematikou kvality zážitku (QoE) a jejími metodami měření. Cílem práce bude vytvoření mobilní aplikace pro operační systém Android. Aplikace bude uživatelům streamovat videa a následně jim umožní hodnotit jejich subjektivní dojem o kvalitě streamovaného videa a výsledky se budou ukládat do vzdálené databáze.

DOPORUČENÁ LITERATURA:

[1] GRIFFITHS, Dawn a David GRIFFITHS. Head First Android Development. O'Reilly Media, 2015. ISBN 1449362184.

[2] ITU-T, "Recommendation P.910: Subjective video quality assessment methods for multimedia applications," April 2008

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: Ing. Dominik Kováč
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

UPOZORNĚNÍ:
Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

Abstrakt

Diplomová práce se zabývá subjektivním měřením kvality zážitku ze streamovaného videa prostřednictvím mobilní aplikace. Po rozsáhlé analýze možností, jak uskutečnit subjektivní měření kvality zážitku, byla vytvořena mobilní a webová aplikace, které spolu vzájemně kooperují. Je tedy možné získávat uživatelské hodnocení streamovaných videí prostřednictvím webové i mobilní aplikace a následně je vyhodnocovat. Tato uživatelská hodnocení jsou ukládána do centrální databáze přes zabezpečenou REST API.

Klíčová slova

subjektivní Quality of Experience, adaptivní streamování, Java, Android, REST API, Hibernate, HTML5

Abstract

This thesis is focused on the subjective measurement of the quality of experience on streaming video through a mobile application. After wide analysis of possibilities how to realize subjective measurement of quality of experience, the mobile and web application were created. These tools enable to obtain user ratings of streaming videos and then evaluate them. The user ratings are stored to the central database via secured REST API.

Keywords

Subjective Quality of Experience, Adaptive Streaming, Java, Android, REST API, Hibernate, HTML5

ŠEDA, Pavel. *Mobilní aplikace pro subjektivní měření kvality zážitku streamovaného videa*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 87 s. Vedoucí diplomové práce Ing. Dominik Kováč.

Vysázeno pomocí balíčku `thesis` verze 2.61; http://latex.feec.vutbr.cz

Prohlášení

Prohlašuji, že svou diplomovou práci na téma „Mobilní aplikace pro subjektivní měření kvality zážitku streamovaného videa“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

Poděkování

Rád bych poděkoval Ing. Dominiku Kováčovi za seznámení s problematikou subjektivního měření kvality zážitku streamovaných videí, cenné rady, věcné připomínky, poskytnuté množství materiálů a vstřícný přístup při konzultacích a vypracování diplomové práce. Dále bych rád poděkoval svým rodičům, přítelkyni a dcerám za podporu a neskonalou trpělivost, kterou mi po celou dobu poskytovali.

Brno ....

podpis autora

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkyňova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

Poděkování

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

Obsah

Seznam obrázků

Seznam tabulek

Seznam algoritmů

Úvod

Kvalita zážitku (Quality of Experience – QoE) se v poslední době stalo hlavním paradigmatem, které klade důraz na uživatelské chápaní kvality komunikačních kanálů a služeb.

Quality of Experience původně patřilo do problematiky kvality služeb (Quality of Service – QoS), která se zabývá spíše výkonnostními charakteristikami služeb, jako je latence, míra chybovosti, šířka pásma, pravděpodobnost výpadku atd. QoS se tedy na rozdíl od QoE nezabývá koncovými uživateli.

S nárůstem množství služeb a uživatelů se stalo velmi žádoucí pro poskytovatele zkoumat služby nejen z pohledu QoS, ale i QoE, protože je vhodné, aby se kromě výkonnostních charakteristik formoval i celkový uživatelský dojem z jimi poskytovaných služeb.

Cílem této práce je realizace nástroje pro subjektivní měření kvality zážitku pomocí mobilní aplikace. Na základě tohoto nástroje bude možné provádět simulace adaptivního streamování pomocí scénářů přehrávání, kde je definováno, v jaké časové okamžiky se mění kvalita videa, jak dlouho přepnutí kvality trvá a na jakou kvalitu se video v daný čas přepne. Tyto scénáře bude možné jednoduše přizpůsobit dle potřeb měření změnou parametrů v databázi. Pro implementaci bude v první části vytvořena analýza projektu a testování obecně, v další části bude představena a zhodnocena možnost využití YouTube Android API a pokud jej nebude možné využít pro subjektivní QoE, pak budou nastíněny některé jiné možné přístupy a jeden z nich bude implementován.

1 Úvod do problematiky streamingu

Kapitola úvod do problematiky streamingu je rozdělena na sekce streaming, služby pro streamování videa a adaptivní streamování. Nejprve je uveden krátký popis, co je to streaming, poté jsou představeny nejpoužívanější streamingové služby a v poslední části jsou objasněny techniky streamování se zaměřením na techniku adaptivního streamingu.

1.1 Streaming

Streaming je technologie přenosu audiovizuálního materiálu ke koncovým uživatelům přes komunikační kanál, jako je například internet nebo vyhrazená IP (Internet Protocol) síť, která je řízena poskytovatelem služeb. Přenos může být v reálném čase (televize, rádio) nebo systémem video na vyžádání (*on demand*), což zprostředkovává například služba YouTube. Při streamování videa musíme mít k dispozici také streamovací server, který zajišťuje komunikaci s cílovými uživateli [1]. Služby, které poskytují streamovací servery, jsou popsány v sekci 1.2.

1.2 Služby pro streamování videa

V současné době existuje mnoho služeb pro streamování videa, které nabízejí streamování na vyžádání, na základě předdefinovaných playlistů nebo například podle žánrů. Mezi nejznámější streamovací služby patří:

- Netflix [2],
- Amazon Prime Instant Video [3],
- YouTube [4].

Jednou z nejpopulárnějších aplikací dnešního internetu je YouTube. Dnes představuje více než 30 % provozu na celém internetu. Jde o streamovací platformu, která nabízí hlavně malé až středně velké videoklipy. Video je zašifrované dle H.264/MPEG-4 kódování (Advanced Video Coding – AVC) jako výchozí kompresní formát videa. YouTube používá HTTP (Hypertext Transfer Protocol) jako streamovací technologii. Pro úplnost je v tabulce uvedeno 1.1 porovnání HTTP streamování s často vyskytujícím se UDP (User Datagram Protocol) streamováním [5].

Tab. 1.1: Srovnání HTTP a UDP streamingu.

HTTP	UDP
spolehlivý přenos	nespolehlivý přenos
kvalita videa není ovlivněna	kvalita videa je ovlivněna
většina poruch má dočasný charakter	poruchy způsobují vizuální degradaci

Klient stáhne data videa přes HTTP spojení a současně spustí video ještě předtím, než jsou data plně stažena. Klientská aplikace je předkompilovaná s využitím Adobe Flash player assembly, která běží u klienta ve webovém prohlížeči. Stažená data jsou uložena v dočasných souborech, které slouží jako buffer pro spuštění určité části videa. YouTube provádí před-bufferování, což znamená, že klientovi se spustí video poté, co je určitá část videa dostupná v bufferu. Čas od požadavku na spuštění videa do doby, než video začne hrát (úspěšně se naplní buffer), se nazývá počáteční zpoždění (startup delay). Když je video spuštěné, server znovu plní buffer periodicky přenášenými bloky s daty videa ke klientovi. Aktuálně příchozí data ke klientovi jsou regulována TCP (Transmission Control Protocol) protokolem a také dostupnou šířkou pásma [6].

Pro služby, které poskytují streamování videa, je rovněž podstatné, jaké používají techniky pro streamování videa. Tyto techniky jsou detailněji popsány v sekci 1.3. Volba správné techniky streamování je jeden z klíčových faktorů, které vedou k vyšší kvalitě zážitku ze streamovaného videa.

1.3 Techniky streamování videa

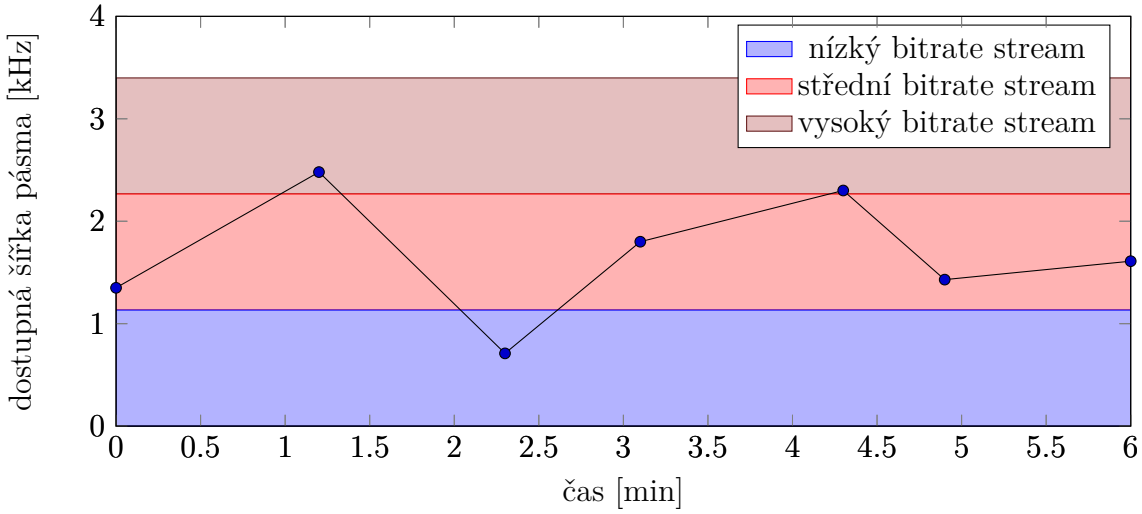
Služby pro streamování videa používají streamovací techniky, které se dají rozdělit na 3 základní kategorie:

- streaming,
- adaptivní streamování,
- progresivní stažení.

Při streamingu, když uživatel klikne na webové stránce na play u videa, které si chce spustit, video začne okamžitě hrát a hraje hladce až do konce. Aby to bylo možné, musí být rychlost přenosu dat kódovaného souboru menší než kapacita pásma vzdáleného diváka, jinak se video často zastavuje [7].

Jako prevence proti zastavování videa se v současné době velmi často používá technika adaptivního streamování, která řeší několik kritických aspektů při přenosu dat. Jedním z nejvýznamnějších je, že ze stejného zdrojového souboru vytváří několik

souborů, které jsou distribuovány uživatelům různých zařízení s různou přenosovou rychlostí. Neméně důležitá je adaptivní distribuce souborů, to znamená, že mění rychlost přenosu dat nebo přenosovou rychlost videa v závislosti na rychlosti připojení jednotlivého uživatele. Výhodou je transparentnost. Uživatel jedním klikem vybere předem bitrate a kvalitu videa, čímž se na pozadí nastaví přepínání streamování. Pokud video zaplní vyrovnávací paměť a využití procesoru je nízké, technologie adaptivního streamování může přepnout na vyšší kvalitu přenosu a zvýšit tak kvalitu zážitku. Klesne-li vyrovnávací paměť pod určitou úroveň nebo využití CPU (Central Processing Unit) přesáhne mezní hranice, může tato technologie naopak přepnout na nižší kvalitu streamu. Jeden z možných průběhů změny bitrate je vidět na obrázku 1.1, kdy mezi body 2 a 2.5 na ose znázorňující čas vidíme přetížení sítě, které vyvolalo právě snížení bitrate.



Obr. 1.1: Jeden z možných průběhů změny bitrate během streamingu.

Klíčovým rozdílem v implementaci napříč různými technologiemi je streamovací server. Některé technologie vyžadují streamovací server a neustálou komunikaci mezi serverem a přehrávači. Pokud je požadováno přepnutí streamu, server tento požadavek zprocesuje odesláním jiného streamu. Ostatní technologie fungují bez streamovacího serveru. Streamy různých kvalit jsou zveřejněny na různých adresách webového serveru nebo několika webových serverech. Přehrávač sleduje provozní heuristiky, jako je využití procesoru nebo stav vyrovnávací paměti, a rozhoduje, kdy je nutné provést změnu a začít načítání dat z jiného streamu.

Poskytovatelé služeb pro adaptivní streaming přehrávače spadají do tří hlavních kategorií: technologie vývojáře, poskytovatelé služeb a standard-based technologies. K prominentním technologickým poskytovatelům se řadí:

- Adobe s Flash-based Dynamic Streaming,

- Apple s HLS (HTTP Live Streaming),
- Microsoft s technologií Smooth Streaming pro Silverlight.

Několik dalších možností založených na WebM-based HTML5 (Hyper Text Markup Language 5) je dostupných nebo se vyvíjejí, a to včetně technologií od Anevia and Quavlive. Primárním poskytovatelem služeb je hlavně Akamai s jejich Akamai HD Network, což je platforma která může být použita na iOS zařízeních, Flash a Silverlight klientech. Některé společnosti, jako například Netflix, vyvinuly své vlastní technologie adaptivního streamování pro interní použití. Standard-based technologie zahrnují SVC (Scalable Video Coding) – rozšíření specifikace H.264. Vedle toho společnost Apple představila HSL protokol v IETF (Internet Engineering Task Force), který pracuje na základě standardizace procesů [8, 9].

Technika progresivního stažení je založena na stažení videa z HTTP web serveru upřednostněného před streaming serverem. Ve většině případů video doručené touto technikou je uloženo na pevném disku uživatele, jakmile je video přijato. Poté je video přehráváno přímo z pevného disku. V kontrastu se streaming videem, které obvykle není uloženo lokálně, takže pokud uživatel nemůže načíst video a přehrát ho v reálném čase, tak si ho nemůže přehrát vůbec [7].

Výše uvedené techniky streamingu ovlivňují kvalitu zážitku uživatele, která vzhledem k velkému rozmachu streamovaných videí přinesla i zintenzivnění výzkumu v oblasti QoE. Detailněji je rozebrána v kapitole 2.

2 Quality of Experience

QoE se v poslední době stalo hlavním paradigmatem, které klade důraz na uživatelské chápání kvality komunikačních kanálů a služeb. Pomocí QoE jsme schopni popsat uživatelské vnímání a výslednou spokojenost se službami v komunikační síti. Toho často využívají poskytovatelé internetu a operátoři, kteří věnují stále více pozornosti modelování a vyhodnocování QoE, aby mohli dosáhnout konkurenční výhody [10]. Standardně se QoE dělí na 2 přístupy:

- objektivní,
- subjektivní.

V případě subjektivních metod se kvalita služeb hodnotí uživateli a v případě objektivních metod se vyhodnocuje kvalita služeb strojově. Tyto přístupy jsou popsány detailněji v dalších sekcích.

Výzkum problematiky QoE vyšel původně z QoS, která se zabývá spíše výkonnostními faktory sítě, jako jsou:

- šířka pásma,
- latence,
- míra chybovosti,
- provozuschopnost / pravděpodobnost výpadku aj.

QoS na rozdíl od QoE se tedy nezabývá koncovými uživateli, ale spíše výkonnostními charakteristikami služeb, které jim poskytují. QoS je zvlášť důležité pro poskytovatele internetových služeb, kteří nabízejí velké množství provozně náročného obsahu, jako je VoIP, streamování médií, online hraní, videokonference, tedy cokoliv, co je závislé na konstantním a stabilním rychlém připojení.

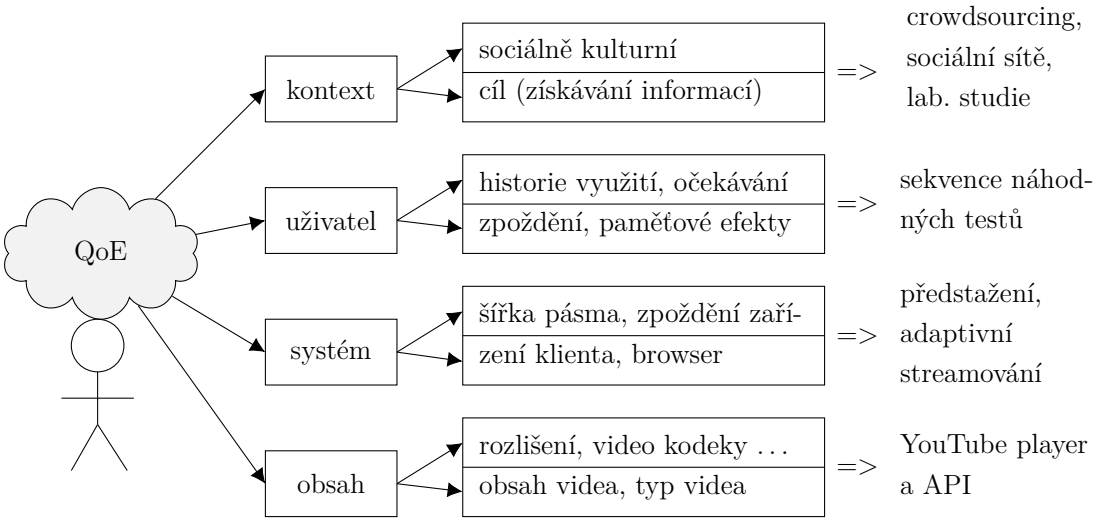
Z popisu výše vidíme, že problematiky QoS a QoE spolu úzce korelují. Pro poskytovatele služeb je tedy důležité, aby měli výkonnostní charakteristiky sítě na vysoké úrovni, ale neméně důležitá je také kvalita zážitku, která formuje celkový uživatelský dojem z jimi poskytovaných služeb [11, 12]. Tato práce je zaměřena na problematiku QoE a zejména na problematiku subjektivního QoE, která je popsána v dalších sekcích.

2.1 Faktory ovlivňující QoE

Vnímání kvality streamovaného videa pro uživatele na internetu je ovlivněno celou řadou faktorů. Tyto ovlivňující faktory jsou rozděleny do čtyř základních kategorií:

- kontext,
- uživatel,
- systém,
- obsah.

Kontextová úroveň zvažuje aspekty, jako je prostředí, ve kterém uživatel konzumuje danou službu. Uživatelská úroveň zahrnuje psychologické faktory, jako jsou očekávání uživatele, paměť, historie používání aplikace. Technické ovlivňující faktory jsou abstraktní na systémové úrovni. Pokrývají vlivy na přenosové sítě, zařízení, ale také implementace aplikace samotné, která může mít jiné strategie na bufferování videa. Pro doručení videa obsahová úroveň zkoumá video kodeky, formát, rozlišení, ale také délku videa, obsah videa a typ videa [24]. Všechny výše zmíněné kategorie ovlivňujících faktorů jsou přehledně znázorněny na obrázku 2.1.



Obr. 2.1: Faktory ovlivňující QoE.

2.2 Objektivní QoE

Objektivní QoE se vyhodnocuje zpravidla strojově, což vede na rozdíl od subjektivního QoE k menší náročnosti na lidské a finanční zdroje, proto se někdy upřednostňují objektivní metody před subjektivními. Objektivní metody měření kvality se dají rozčlenit do 3 skupin:

- plná reference,
- redukovaná reference,

- bez reference.

Některé z těchto metod jsou založeny na zahrnutí původního referenčního signálu v měření.

Plná reference vyžaduje původní materiál v celku. Operují na základě porovnávání původního s pozměněným či narušeným signálem k vypočítání degradace. Výpočty této degradace se pohybují od jednoduchých algoritmů typu odhadu chyb v signálu k velmi komplexním.

Redukovaná reference používá část původního videa pro výpočet degradace. Jsou vhodné pro situace, kde je původní obsah těžké uložit nebo přenést na místo určení, nebo je-li výpočetní výkon omezen.

Objektivní metoda bez reference nepoužívá žádnou část původního obsahu. Nezávisí na porovnání s původním signálem, ale na měření externích faktorů k vytvoření modelu QoE. Modely bez reference jsou obvykle určeny pro specifické aplikace a nejsou vhodné k obecnému použití, ale vyžadují nejméně zdrojů a jsou užitečné v případě, kdy původní obsah není dostupný [13]. Mezi metody objektivního měření QoE patří:

- PSNR (Peak Signal to Noise Ratio),
- VQM (Video Quality Metric),
- MPQM (Moving Picture Quality Metric),
- SSIM (Structural Similarity Index),
- NQM (Noise Quality Measure).

Vzhledem k tomu, že práce není zaměřena na objektivní metody QoE, bude zde popsána pouze metoda PSNR, která patří mezi jednu z nejrozšířenějších.

Metoda PSNR je navržena pro obecnější použití, protože dokáže detekovat chyby v jakémkoliv typu signálu. Díky své jednoduchost je často používána pro hodnocení kvality obrazu a hodnocení kvality videa. PSNR je odvozena nastavováním střední mocniny chyby (MSE), ve vztahu k maximální možné hodnotě jasu (například pro 8bitové hodnoty je to $2^8 - 1 = 255$) jako v následující rovnici:

$$MSE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[f(i,j) - F(i,j)]^2}{M \times N} \tag{2.1}$$

$$PSNR = 20 \cdot \log_{10}\left(\frac{255}{\sqrt{MSE}}\right) \tag{2.2}$$

kde:

$f(i,j)$:	původní signál na pixelu (i,j)
$F(i,j)$:	rekonstruovaný signál
$M \times N$:	velikost obrázku

Výsledkem je číslo v decibelech v rozmezí od 30 do 40 decibelů pro středně až vysoce kvalitní videa [13, 14, 15].

2.3 Subjektivní QoE

Metody subjektivního QoE jsou založeny na hodnocení služeb samotnými pozorovateli, buď může jít o laboratorní testy, kde jsou pozorovatelé (lidé), kteří hodnotí jednotlivé představené služby, nebo může jít o hodnocení v rámci takzvaného crowdsourcingu viz sekce 2.3.1. Výsledky jsou poté vyhodnoceny (popřípadě ještě dodatečně korelovány viz sekce 2.3.2) a je z nich sestaveno MOS[1] (metody měření subjektivního QoE jsou detailněji popsány v sekci 2.3.1).

Nevýhodou laboratorních testů je, že jsou zpravidla náročné na finanční, časové a lidské zdroje. Mezi nevýhody také patří, že pozorovatel se může účastnit testovací události pouze jednou, protože pokud by se testů účastnil opakovaně, výsledek by se zkresloval, projevilo by se „(na)učení“.

Výhodou subjektivního QoE při dodržení veškerých podmínek je, že měření subjektivními metodami ze všech současných způsobů měření je nejpřesnější, proto se tyto metody často používají jako referenční u nových zařízení, multimediálních kodeků atd. [16, 17, 19].

2.3.1 Metody měření subjektivního QoE

V současné době bylo publikováno poměrně velké množství metod měření subjektivního QoE. V této práci budou nastíněny metody, které patří mezi nejpoužívanější měřicí metody subjektivního QoE současnosti. Jde o metody Crowdsourcing, ACR (Absolute Category Rating), ACR-HR (Absolute Category Rating with Hidden Reference), DCR (Degradation Category Rating), PC (Pair Comparison) method, DSCQS (The Double-Stimulus Continuous Quality-Scale) method. U všech těchto metod je vysvětlen jejich základní princip [18, 19].

[1] Mean opinion score (MOS) se vyhodnocuje na základě výsledků hodnocení služeb při použití nejčastěji 5úrovňových stupnic, které jsou uvedeny v metodách ACR, viz tabulka 2.1 a DCR tabulka 2.3.

Crowdsourcing

Crowdsourcing je poměrně nová metoda používaná k získávání dat od velkého množství uživatelů, kteří nejsou nuceni se účastnit laboratorních testů, jako je tomu v případě metod ACR, ACR-HR, DCR, PC, DSCQS aj. Crowdsourcing je založen na drobných operacích prováděných prostřednictvím služeb na internetu, které jsou snadno splnitelné:

- napsat článek / blok / komentář,
- umístit banner nebo link na webovou stránku,
- nahrát / stáhnout média,
- využívat služby sociálních sítí,
- instalovat software,
- provést YouTube Stalling test.

Mezi výhody crowdsourcingu oproti běžným laboratorním testům pro získávání dat pro studie subjektivního QoE patří široké spektrum uživatelů, rozmanitost uživatelů (např. z jiných geografických lokací nebo výrazně rozdílný věk atd.), uživatelské studie mohou být provedeny v krátkém čase, nízké náklady v porovnání s laboratorními testy a testování QoE pro internetové aplikace s reálnými nastaveními.

Crowdsourcing se v praxi využívá pro širokou škálu funkcionalit:

- online zlepšování pověsti / dobrého jména dané služby,
  - hodnocení pro videa nebo články,
  - pozitivní hodnocení profilu na základě hlasování,
- hodnocení obrázků,
- získávání nových uživatelů / využívání nových platforem,
- online reklamy / optimalizace vyhledávacích nástrojů,
  - umisťování informací na twitter,
  - tvorba zpětných odkazů,
- uživatelské dotazníky pro marketing / QoE studie,
- vytváření obsahů,
  - příspěvky na fórech či blozích,

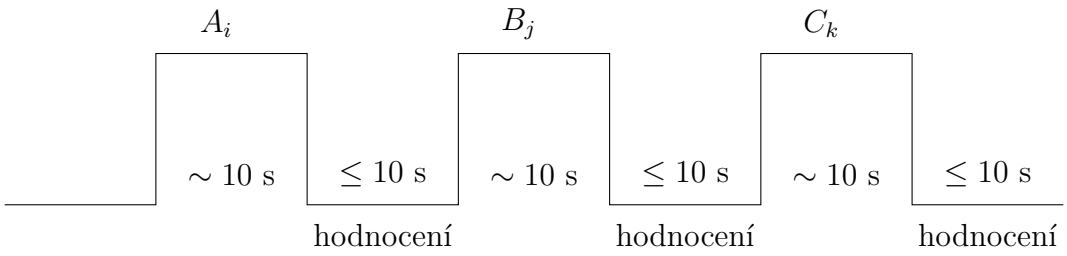
  - články,

- výzkum a vývoj.

Z předchozích bodů je vidět, že crodwsourcing je často používaná metoda skýtající mnoho výhod, avšak k jejím rizikům patří možnost ovlivňování komunity, jednostranná zaměřenost komunity atd., což může vést k irelevantním výsledkům [5, 23]. V takovém případě crowdsourcing nepřinese názorový průměr, ale názory jednotlivců, proto je vhodné provádět i laboratorní testy, které jsou popsány v dalších odstavcích.

Absolute Category Rating (ACR)

ACR je metoda, při které jsou testované sekvence prezentovány lidskému subjektu jedna po druhé a poté jsou sekvence nezávisle hodnoceny na stupnici přiřazené dané kategorii (tato metoda se také nazývá *metoda jediného stimulu*). Subjekty jsou požádány, aby po zhlédnutí nebo poslechu ohodnotily kvalitu prezentace, a to na základě očekáváné úrovně kvality pro danou sekvenci. Časový průběh testování metodou ACR je vidět na obrázku 2.2. Pokud je zvoleno konstantní hodnocení, pak by čas na vyhodnocení měl být menší nebo roven 10 s. Doba prezentace sekvence může být upravena v závislosti na obsahu testovaného materiálu.



Obr. 2.2: Stimulovaná prezentace ACR metody.

kde:

A_i :	sekvence A za testovacích podmínek i
B_j :	sekvence B za testovacích podmínek j
C_k :	sekvence C za testovacích podmínek k

Nejčastěji se používá pětiúrovňová stupnice hodnocení pro celkovou kvalitu. Tato stupnice je znázorněna v tabulce 2.1.

Tab. 2.1: Pětiúrovňová stupnice pro hodnocení kvality metodou ACR.

5	Excellent (vynikající)
4	Good (dobrá)
3	Fair (průměrná)
2	Poor (špatná)
1	Bad (mizerná)

Je-li požadováno vyšší rozlišení hodnocení, pak se používají 9 nebo 11úrovňová hodnocení [18, 19].

Absolute Category Rating with hidden reference (ACR-HR)

Tato metoda je založena na principu posuzování, kdy je subjektu předkládán jeden vzorek, který je následně nezávisle hodnocen dle dané stupnice. Předložená testovaná sekvence musí zahrnovat i referenční verzi každé sekvence. Tento přístup se označuje jako *hidden reference condition*. Při analýze dat bude hodnocena kvalita mezi každou testovací sekvencí a jí odpovídající referenční sekvenci. Tato metoda je známá jako hidden reference. Metoda určuje, že subjekt hodnotí kvalitu prezentované sekvence po každé prezentaci.

Je-li použit konstantní hodnotící čas (několik subjektů sleduje video současně), pak by čas pro hodnocení měl být menší nebo roven 10 s. Čas prezentace může být zkrácen nebo prodloužen, a to na základě obsahu testovaného materiálu. Pro tento typ hodnocení by měla být použita 5úrovňová stupnice, viz tabulka 2.1.

DV (Differential viewer scores) se určí z následující rovnice:

$$DV(PVS) = V(PVS) - V(REF) + 5 \tag{2.3}$$

kde:

V :	skóre pozorovatele získané ACR stupnicí
REF :	příslušná skrytá reference
PVS :	kódovaná sekvence

V této rovnici odpovídá DV hodnota 5 vynikající kvalitě a DV hodnota 1 odpovídá špatné kvalitě. Jakákoliv hodnota vyšší než 5, když hodnocená sekvence je hodnocena lepší kvalitou než skrytá referenční sekvence, bude obecně považována za platnou. Alternativně může být použita „2point crushing function“ k zabránění těmto individuálním ACR-HR hodnocením (DV) a ovlivňování MOS:

$$Crushed_{DV} = (7 \cdot DV)/(2 + DV) \text{ kde } DV > 5 \tag{2.4}$$

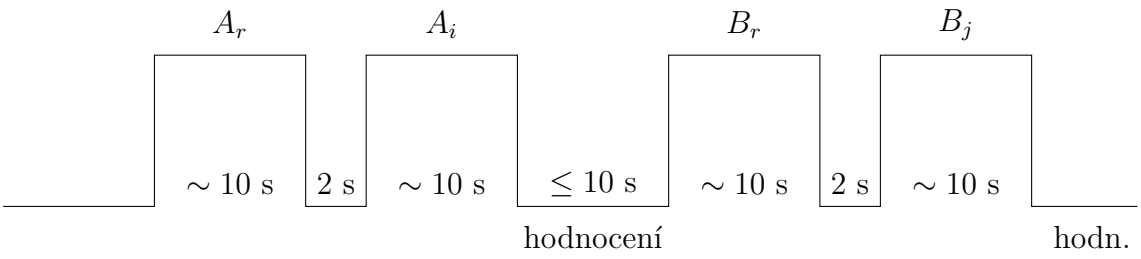
Je-li požadováno vyšší rozlišení, pak je použita 9úrovňová stupnice ACR.

Pro metodu ACR-HR se získá potřebný počet replikací opakováním stejné zkušební podmínky v různých časových intervalech během trvání testu. Tato metoda by měla být použita pouze s referenčním videem (sekvencí), která je experty v dané oblasti hodnocena jako dobrá nebo vynikající kvalita na výše zmíněné 5stupňové škále.

Metoda ACR-HR nemusí být vhodná pro analýzu neobvyklých zhoršení kvality vyskytujících se v první a poslední sekundě sekvence. Neobeznámenost subjektu s referenční sekvencí může způsobit, že jinak zřejmá poškození budou přehlédnuta (sekvence se zastaví těsně před koncem, subjekt nemusí být schopen rozeznat zda se jedná o záměr nebo chybu sítě) [19].

Degradation Category Rating (DCR)

V metodě DCR jsou testované sekvence uvedeny v párech. První stimul v každé dvojici je vždy zdrojový bez jakýchkoliv vad. Druhý z páru je ze stejného zdroje, ale zhoršen zkušebními podmínkami (tato metoda se také nazývá *Double Stimulus Impairment Scale* (DSIS)). Časový průběh testování metodou DCR je vidět na obrázku 2.3.



Obr. 2.3: Stimulovaná prezentace DCR metody.

kde:

A_i :	sekvence A za testovacích podmínek i
A_r, B_r :	sekvence A a B v referenčním zdrojovém formátu j
B_j :	sekvence B za testovacích podmínek j

Doba hodnocení by měla i v tomto případě být menší nebo rovna 10 s. V tomto případě jsou subjekty žádány, aby ohodnotily zhoršení druhého vzorku ve srovnání se vzorkem zdrojovým. Nejrozšířenější stupnicí pro toto hodnocení je opět 5úrovňová stupnice podle tabulky 2.2. V podstatě lze pro toto hodnocení použít jakoukoliv stupnice použitou při metodě ACR, pouze je nutno zaměnit hodnotící parametry.

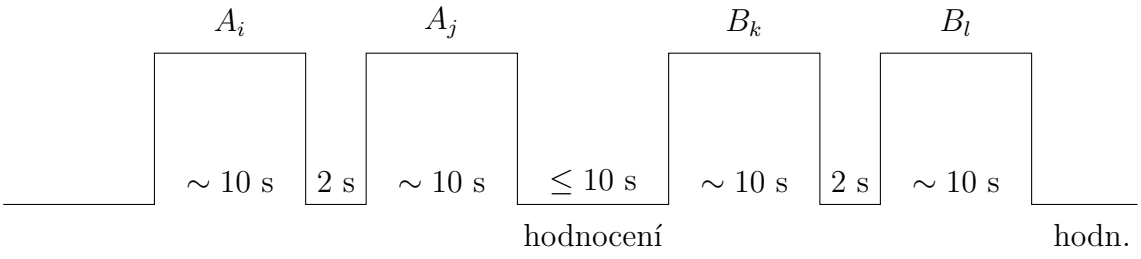
Prezentační čas může být upraven v závislosti na obsahu testovaného materiálu stejně jako u metody ACR [19].

Tab. 2.2: Pětiúrovňová stupnice pro hodnocení zhoršení metodou DCR.

5	Imperceptible (nepatrné)
4	Perceptible but no annoying (znatelné, ale ne nepříjemné)
3	Slightly annoying (mírně nepříjemné)
2	Annoying (nepříjemné)
1	Very annoying (velmi nepříjemné)

Pair comparison method (PC)

Metoda PC spočívá v tom, že testované sekvence jsou prezentovány v párech, sestavených ze stejné sekvence prezentované nejprve jedním testovaným systémem a následně dalším systémem. Testované systémy (A, B, C) jsou zpravidla kombinovány ze všech možných systémů $n(n-1)$ kombinací AB, BA, CA atd. Takto by všechny testované páry měly být prezentovány v obou možných pořadích (AB, BA). Po každém páru se hodnotí, který element dvojice je preferován v rámci zkušebního scénáře. Je-li použit konstantní hodnotící čas (několik subjektů video sleduje současně), pak by čas pro hodnocení měl být menší nebo roven 10 s. Čas prezentace může být zkrácen nebo prodloužen a to na základě obsahu testovaného materiálu. Časový průběh testování metodou ACR je vidět na obrázku 2.4.



Obr. 2.4: Stimulovaná prezentace PC metody.

Použije-li se menší rozlišení (CIF, QCIF, SIF), je vhodné zobrazit každý pár sekvencí současně na stejném monitoru. Pro PC metodu není obecně stanoven počet opakování, a to proto, že metoda sama o sobě obsahuje opakované prezentace sekvencí se stejnými podmínkami, i když v různých párech. Různé variace metody PC využívají kategorické měřítko pro další měření rozdílů mezi páry sekvencí [19].

The Double-Stimulus Continuous Quality-Scale method (DSCQS)

Metoda DSCQS je užitečná, není-li možné zajistit zkušební podmínky zahrnující celé spektrum dostupné kvality. Tato metoda může být použita například v síti, která zaručuje určitou úroveň QoS, například když MOS je větší než 3 na stupnici s rozsahem 5 bodů. Stejně jako v metodě DCR jsou i zde sekvence uvedeny ve dvojicích: referenční a zhoršená sekvence. Subjekty jsou žádány, aby posoudily kvalitu obou sekvencí v páru (nikoli však hodnotit zhoršenou sekvenci vzhledem k referenční, jako je tomu u DCR).

Nepozměněná sekvence slouží jako referenční, ale pozorovateli není řečeno, která to je. V sérii testů se náhodně mění pozice referenčního videa. Pozorovatelé jsou požádáni, aby ohodnotili celkovou kvalitu každé testované sekvence vložením hodnocení dle hodnotící stupnice (nejčastěji 5úrovňové) [19].

2.3.2 Metody korelace získaných výsledků

Během testování se získává velké množství dat, které zpravidla obsahuje i určitý zlomek šumových dat, a ta mohou mít za následek zkreslení výsledků, a tím negativně ovlivnit závěry. Pro jejich redukci je vhodné zařadit některou z metod korelace výsledků, která by měla pomoci odstranit tato šumová data. Mezi nejpoužívanější metody korelace patří Pearsonova korelace a Spearmanova korelace, které jsou popsány v dalších sekcích.

Pearsonova korelace

Pro použití Pearsonovy korelace se předpokládá, že vztah mezi rozsahem kvality a rozsahem skóre z hodnocení od pozorovatelů je lineární. Hlavním cílem je jednoduchou metodou ověřit, jestli skóre jednoho pozorovatele je konzistentní se skórem ostatních pozorovatelů z testovací události [20].

$$r(x,y) = \frac{\sum_{i=1}^{n} x_i\ y_i\ - \frac{\left(\sum_{i=1}^{n} x_i\right) \cdot \left(\sum_{i=1}^{n} y_i\right)}{n}}{\sqrt{\left(\sum_{i=1}^{n} {x_i}^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}\right) \cdot \left(\sum_{i=1}^{n} {y_i}^2 - \frac{\left(\sum_{i=1}^{n} y_i\right)^2}{n}\right)}} \tag{2.5}$$

kde:

x_i :	průměrné skóre od pozorovatelů z trojice (algo, bitrate, scéna)
y_i :	skóre individuálního pozorovatele ze stejné trojice
n :	(počet algo) × (počet scén)
i :	druh kodeku, bitrate, číslo scény [20, 21].

Spearman rank korelace

Spearman rank korelace může být na rozdíl od Pearsonovy korelace použita, i když vztah mezi rozsahem kvality a rozsahem skóre pozorovatelů není lineární[2] [20].

$$r(x,y) = \left(1 - \frac{6 \cdot \sum_{i=1}^{n} [R(x_i) - R(y_i)]^2}{n^3 - n} \right) \tag{2.6}$$

kde:

x_i :	průměrné skóre od pozorovatelů z trojice (algo, bitrate, scene)
y_i :	skóre individuálního pozorovatele ze stejné trojice
n :	(počet algo) × (počet scén)
$R(x_i$ nebo $y_i)$:	pořadová čísla hodnocení (ranking order)
i :	druh kodeku, bitrate, číslo scény [20, 22].

Finální kritéria pro vyřazení pozorovatele z testu

Spearman rank kolerace a Pearsonova kolerace jsou použity pro odstranění hodnocení pozorovatele dle následujicího pseudókódu.

Algoritmus 1 Finální vyřazující kritéria pro pozorovatele.

```
if [mean(r) − std(r)] > max korelační práh (MTC) then
    práh odmítnutí ← max korelační práh
else
    práh odmítnutí ← mean(R) − std(r)
end if
if [r(pozorovatel_i)] > práh odmítnutí then
    pozorovatel „i“ není vyřazen z testu
else
    pozorovatel „i“ je vyřazen z testu
end if
```

[2] Obecně dává Pearson korelace velmi podobné výsledky jako Spearman rank korelace

kde:

r :	min(Pearson korelace, Spearman rank korelace)
mean(r) :	střední hodnota korelace všech pozorovatelů testu
std(r) :	směrodatná odchylka všech korelací pozorovatelů testu
MTC :	0.85

MTC (Maximal Threshold Correlation) 0.85 je validní např. pro metodu DSCQS, ale pro metodu DCR by měla být použita hodnota MCT 0.7 [20].

3 Vývoj aplikace

Vývoj aplikace je v dnešní době komplexní proces, který vyžaduje náležitou analýzu vytvářeného projektu, to znamená, k čemu má být určen, jaké jsou návaznosti jednotlivých komponent, jaké zdroje bude projekt vyžadovat, jaké by měly být použity technologie a až poté je vhodné začít programovat. Proto je kapitola vývoj aplikace rozdělena na sekce analýza projektu, použité technologie a samotná implementace aplikace.

3.1 Analýza projektu

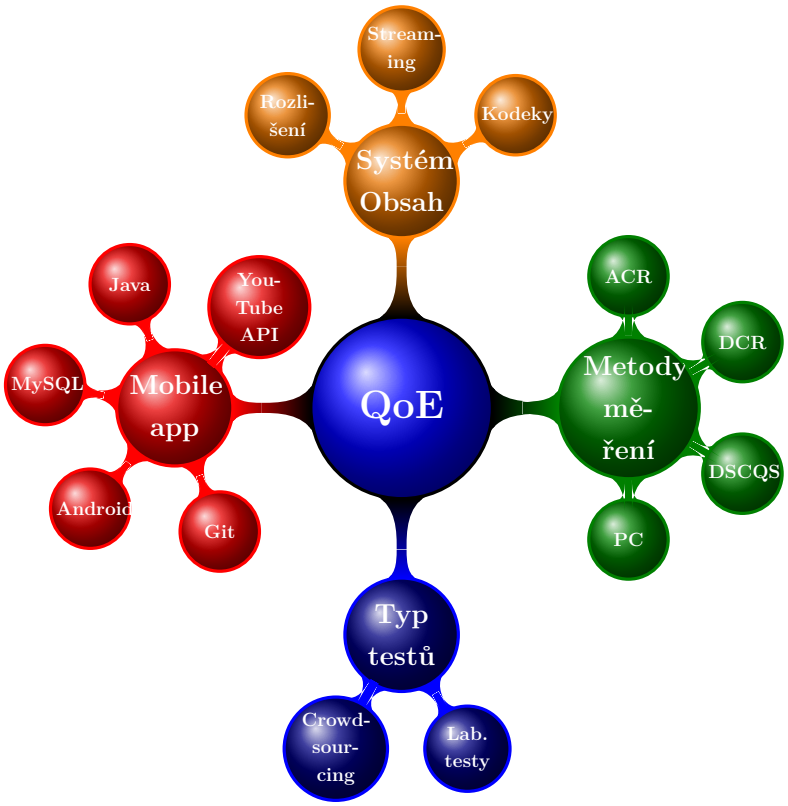
V každém větším softwarovém projektu je vhodné nejprve analyzovat problém a dekomponovat ho na menší podproblémy. K tomu slouží následující diagramy, které zobrazují uživateli činnosti v rámci běhu aplikace, rozložení logiky kódu aj. Záleží na typu diagramu, ke kterým z těchto aspektů je určen. Pro potřeby této práce jsou použity diagramy myšlenková mapa, diagram případů užití, stavový diagram a fyzický model databáze.

3.1.1 Myšlenková mapa

Myšlenková mapa při návrhu systému slouží k uspořádání myšlenek, shrnutí důležitých pojmů a přehledné vizualizaci řešeného tématu. Myšlenková mapa je hierarchická a ukazuje vztahy mezi jednotlivými částmi celého systému. Při tvorbě myšlenkové mapy je vhodné se držet následujícího doporučení:

1. začít uprostřed s obrázkem znázorňujícím téma,
2. použít obrázky, symboly, kódy a dimenze s pomocí myšlenkové mapy,
3. vybrat slova a vepsat je do struktury,
4. každé slovo / obrázek má být nejlépe v samostatné čáře či obrázku,
5. slova by měla mít přiměřenou délku,
6. použít více barev napříč myšlenkovou mapou pro vizuální stimul a také pro seskupení různých částí tématu [25].

Na myšlenkové mapě na obrázku 3.1 je vidět ve středu nápis QoE, který značí, že tématem této práce je Quality of Experience. V levé části, vyznačené červenou barvou, je poukázáno, že má jít o mobilní aplikaci, jíž jsou přiřazeny technologie, které by mohly být použity při vývoji aplikace.

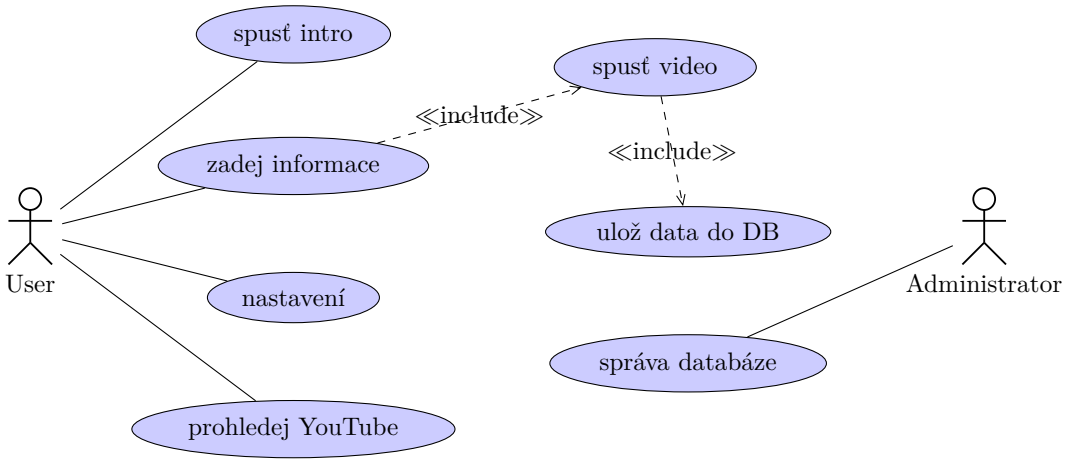


Obr. 3.1: Myšlenková mapa.

Ve spodní části obrázku je vidět modře vyznačené rozdělení na typy testů, které mohou být provedeny v rámci řešené problematiky. V pravé části obrázku, vyznačené zelenou barvou, jsou znázorněny některé z možných metod měření při laboratorních testech a v horní části některé prvky, které se dají zkoumat u multimediálního obsahu. Z výše uvedeného je patrné, že myšlenková mapa velmi přehledně shrnuje základní tematické body z teoretické části a vztahy mezi nimi a rovněž znázorňuje technologie, které by mohly být použity v implementační části pro zkoumanou problematiku.

3.1.2 Diagram případů užití

Diagram případů užití se skládá z množiny možných interakcí mezi systémem a uživatelem. Měl by obsahovat všechny systémové aktivity, které mají významný vliv na uživatele. Tento diagram si lze představit jako množinu možných sekvencí, které vedou k určitému cíli.



Obr. 3.2: Diagram případů užití znázorňující vytvářenou mobilní aplikaci.

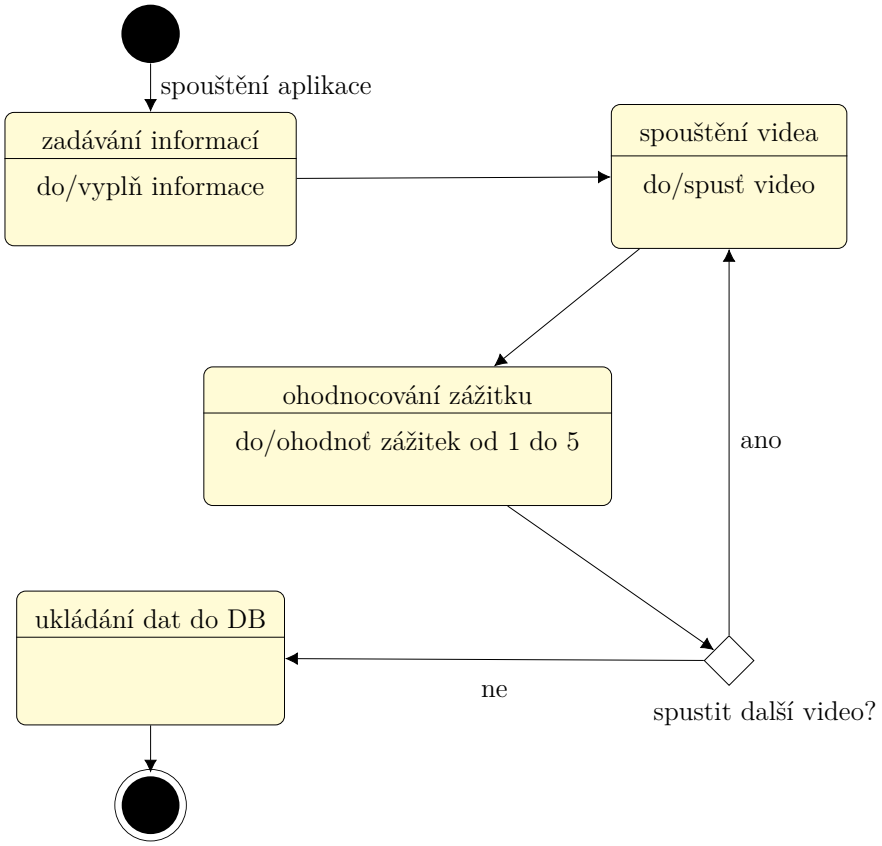
Pomocí tohoto diagramu kromě sekvencí určitých akcí také znázorňujeme různé typy uživatelů, v našem případě uživatel a administrátor, ale obecně jich může být daleko více. V diagramu případů užití se vyskytují tyto prvky:

- Případ užití (use case) – popisuje sled akcí, které poskytují nějakou měřitelnou hodnotu pro uživatele. Případ užití se obvykle kreslí jako horizontální elipsa.
- Aktor (actor) – osoba, organizace nebo externí systém, který hraje určitou roli v jedné nebo více interakcích se systémem. Aktor se obvykle kreslí jako panáček.
- Asociace (association) – asociace mezi aktory a případy užití jsou indikovány v diagramu případů užití tenkými čárami. Asociace existuje vždy, pokud je daný aktor nebo případ užití propojený s danou akcí. Indikace include značí, že po vykonání dané akce se nezbytně poté vykoná akce další.
- Systém (system) – je volitelný a můžeme jej nakreslit jako obdélník kolem prvků případu užití. Tento obdélník je nazýván systém. Všechno uvnitř tohoto obdélníku znázorňuje funkcionalitu, která je v rozsahu daného systému, a cokoliv mimo obdélník je mimo rozsah [26].

Na diagramu případů užití, zobrazující mobilní aplikaci, viz obrázek 3.2, která bude vytvořena v této diplomové práci, vidíme 2 aktory, uživatele a administrátora. Administrátor v systému funguje tak, že může přistupovat do databáze a manipulovat s daty. Uživatel naopak funguje v roli uživatele aplikace případně testera QoE. Uživatel si může spustit intro, může jít do nastavení, může si vyhledávat videa na serveru YouTube nebo může v roli testera pro QoE zadat informace, spustit a ohodnotit video, na základě čehož se bude vyhodnocovat QoE.

3.1.3 Stavový diagram

Stavový diagram je v počítačových vědách používán k popisu chování systému, zvažující všechny možné stavy objektu, kdy dojde k události. Toto chování je reprezentováno a analyzováno v sérii událostí, které se objeví v jednom nebo více možných stavech. Každý stavový diagram začíná s počátečním stavem, což je stav, kdy je objekt vytvořen, a končí koncovým stavem, kdy je daný objekt (program – část aplikace) ukončen [26].

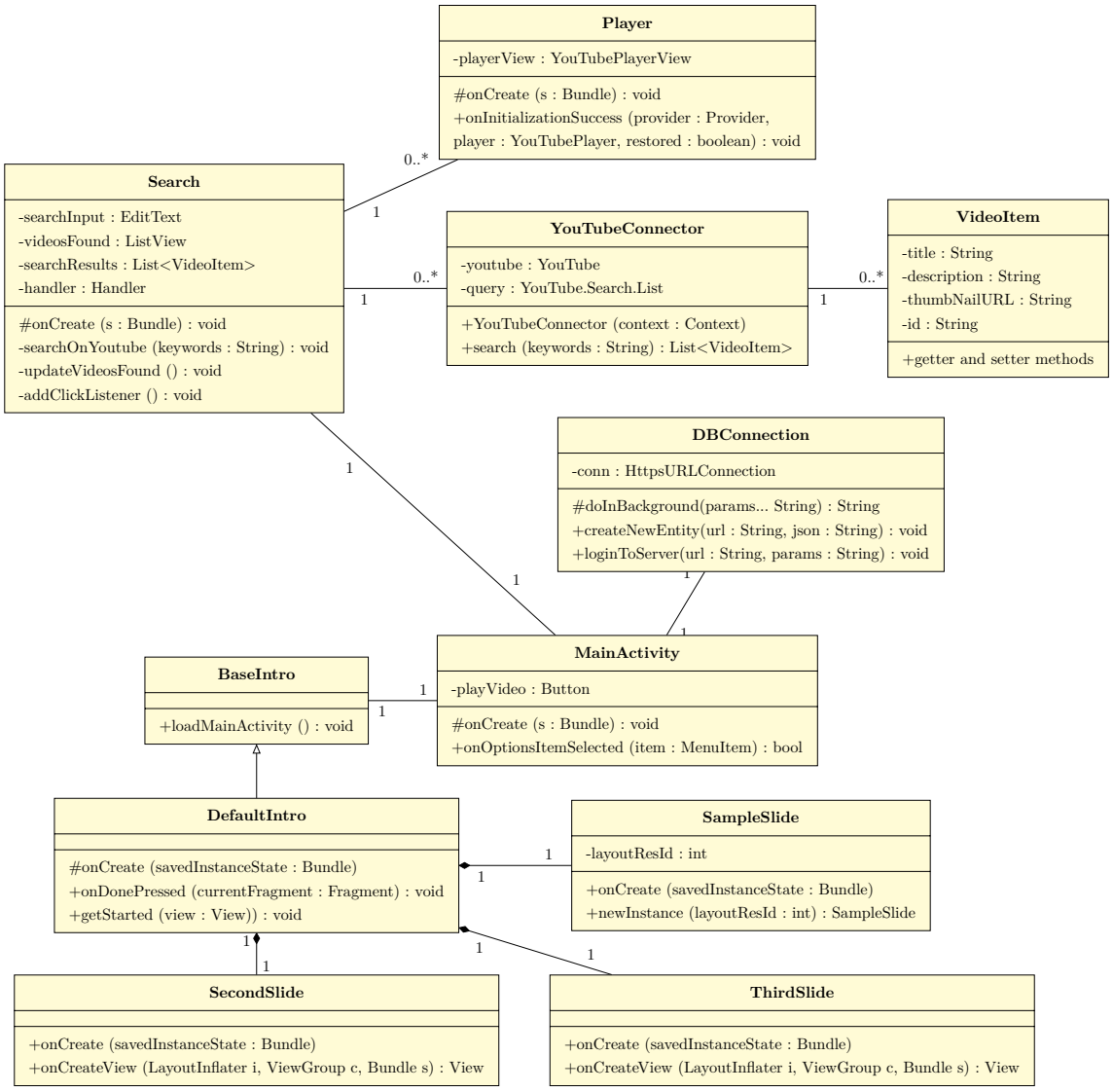


Obr. 3.3: Diagram stavů.

Na stavovém diagramu znázorněném na obrázku 3.3 vidíme průběh testování QoE. Uživatel nejprve zadá základní informace o sobě, poté spustí video, ohodnotí ho a poté pokračuje dalším videem, až se nakonec data uloží do databáze.

3.1.4 Diagram tříd

Diagram tříd znázorňuje návrh implementace aplikace. Měl by tedy obsahovat všechny třídy, které aplikace bude obsahovat. Z diagramu na obrázku 3.4 lze vidět, že z třídy `MainActivity` si lze spustit intro, vyhledávat videa, spustit video a uložit data do databáze.

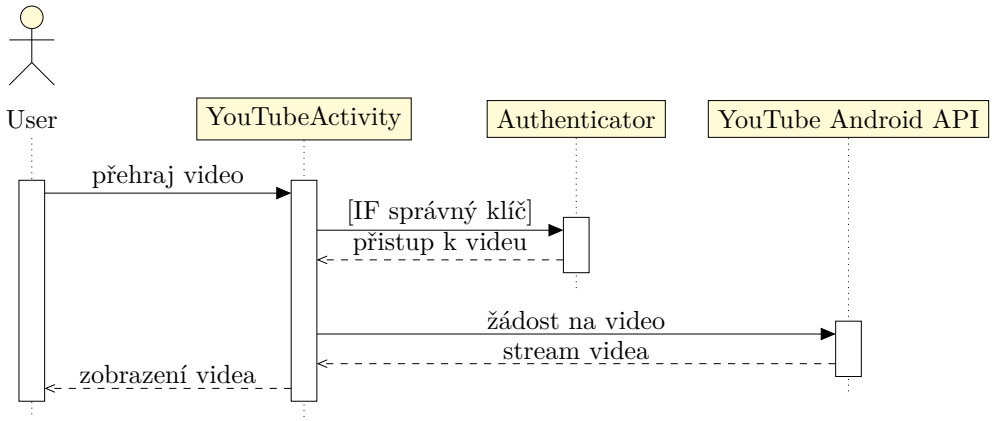


Obr. 3.4: Diagram tříd mobilní aplikace.

Tento diagram by měl tedy sloužit jako podrobný návod, jak implementovat aplikaci [26].

3.1.5 Sekvenční diagram

Sekvenční diagram zachycuje časově uspořádanou posloupnost zaslaných zpráv mezi objekty [26]. Na obrázku 3.5 je znázorněna časová řada zpráv zasílaných v případě žádosti uživatele o přehrání videa.



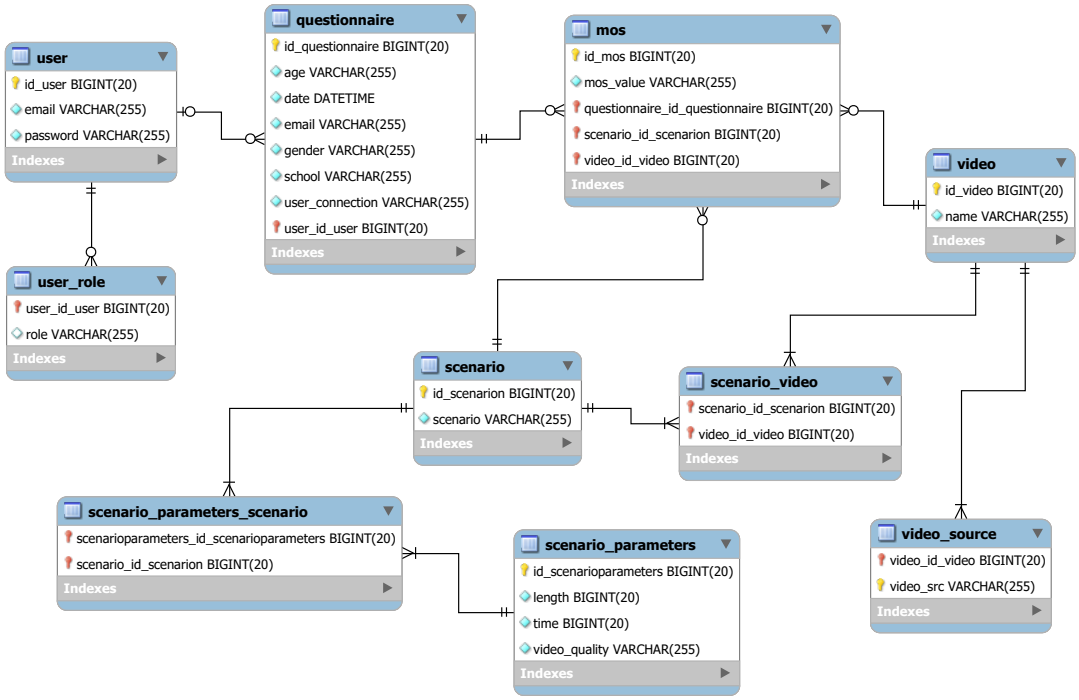
Obr. 3.5: Sekvenční diagram použití YouTube Android API.

3.1.6 Entitně relační diagram

Entitně relační diagram (ERD) se používá pro konceptuální znázornění dat. Obvykle se pomocí ERD modelují relační databázové systémy [26]. Pro potřeby Android projektu je vhodné využít některý z dostupných nástrojů pro vytváření ERD, například MySQL Workbench, ze kterého se následně dají vygenerovat potřebné SQL (Structured Query Language) příkazy, které vytvoří modelované tabulky v databázovém systému MySQL. Tyto příkazy jsou shrnuty v příloze B.1.

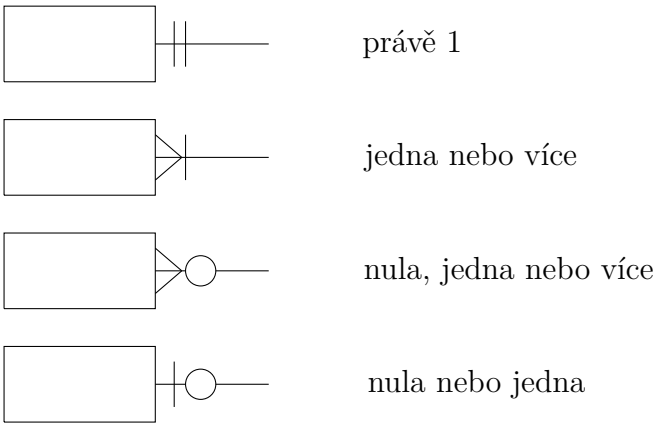
Na obrázku 3.6 je vidět, že v aplikaci budou ukládány základní údaje o uživateli, který podstoupil testování a vyplnil dotazník, jako je věk, pohlaví, mailová adresa, škola, připojení (může jít např. o wifi připojení, kabelové, připojení přes mobilní data, ...), dále po zhlédnutí videa vyplní jeho hodnocení, které se uloží do tabulky `mos`. Ta se odkazuje na přehrávané video a scénáře, které ve videu proběhly (např. v 15. sekundě videa, byl loading 3 sekundy). Tímto databázovým návrhem je tedy možné po provedení testování získat jednoduše údaje na základní otázky, jako jsou:

1. Jak hodnotili uživatelé přehrávané video s tímto scénářem?
2. Další možné statistické dělení hodnocených videí dle:
   - věku,
   - dosaženého vzdělání,
   - typu připojení,
   - pohlaví.



Obr. 3.6: Entitně relační diagram databáze subjektivních hodnocení.

ERD je vytvořen v následující notaci:



3.2 Použité technologie

V této sekci budou nastíněny některé technologie, které byly použity pro vytvoření softwarové části projektu. Mezi další použité technologie, které zde ovšem nejsou popsané, patří: Spring framework, Java EE (Java Enterprise Edition), JSP (Java Server Pages), Tomcat, Maven, Boostrap, JSON (Javascript Object Notation), XML (Extensible Markup Language), HTML5, CSS (Cascading Style Sheets), jQuery, phpMyAdmin, XAMPP.

3.2.1 Git

Git byl vytvořen Linusem Tolvardsem v roce 2005 k vývoji operačního systému Linux. Je licencován pod GNU General Public License version 2. Jde o distribuovaný systém pro správu verzí, určený zejména k softwarovému vývoji. Mezi základní příkazy Gitu patří [28]:

- `git init` – inicializace repozitáře Gitu v existujícím adresáři,
- `git clone [url]` – vytvoří kopii existujícího repozitáře Gitu do aktuální složky, například pro naklonování projektu této práce stačí dát příkaz: `git clone` https://github.com/SedaQ/Measurement-of-User-Experience,
- `git status` – příkaz, pomocí kterého zjišťujeme stav jednotlivých souborů, to znamená, které soubory jsou změněné, které jsou přidané, nebo odebrané atd.,
- `git add .` – po příkazu `git add .` přidáme všechny změněné soubory k sledovaným, což znamená, že jsou připraveny k zapsání a můžeme je zapsat na server nebo poslat do lokálního repozitáře,
- `git add README` – tento příkaz slouží k vytvoření README souboru, který zpravidla obsahuje konfigurační instrukce, instalační instrukce, copyright a licenční informace, kontaktní informace na distributora nebo programátora, známe bugy aj.,
- `git commit -m "Initial commit."` – zaznamená změny do lokálního repozitáře,
- `git push` – pomocí tohoto příkazu zašleme soubory, které jsme přidali přes příkaz `git add` na server.

Projekt byl nahrán na server GitHub, kde jej můžeme najít na adrese: https://github.com/SedaQ/Measurement-of-User-Experience.

V projektu byl Git upřednostněn před verzovacím systémem Subversion (SVN), protože vývojáři umožňuje plnou lokální kontrolu (např. lokální větve, historie aj.), je rychlejší a umožňuje intuitivnější manipulaci s větvemi než SVN [29].

3.2.2 Java

Jako programovací jazyk při tvorbě aplikace byl zvolen programovací jazyk Java, který je nativním jazykem pro platformu Android. Programovací jazyk Java byl vytvořen Jamesem Goslingem v roce 1991. Java je široce rozšířena zejména díky

své jednoduchosti, robustnosti, bezpečnosti, vysoké výkonnosti, nezávislosti na platformě, efektivní podpoře vícevláknových aplikací a své dynamičnosti. Byla použita verze Java 7, která je bez větších úprav podporována platformou Android. Verze Javy 7 umožňuje oproti dřívějším verzím tzv. diamond operátor, kdy nemusíme v instanciovaném objektu vkládat typ, viz následující kód:

```
1 //před verzí Java 7
2 List<String> bezDiamondOperatoru = new ArrayList<String>();
3
4 //verze Java 7 a výš (umožnění diamond operátoru)
5 List<String> diamondOperator = new ArrayList<>();
```

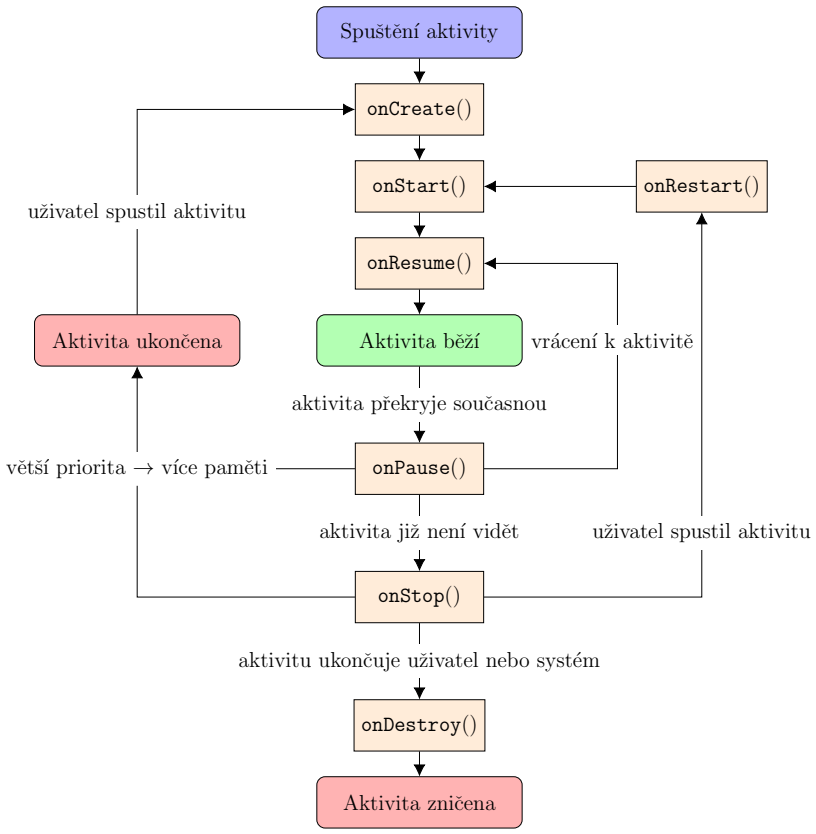
Podporuje takzvaný try-with-resources blok, který využívá `Autocloseable` interface, a tudíž nemusíme zavírat stream ve finally bloku. Také umožňuje řetězení typů výjimek v jednom catch bloku, viz kód níže:

```
1  //před verzí Java 7
2  try{
3    //vytvoř stream k souboru, databázi aj., zpracovávej data
4  } catch(IOException e){ //zpracuj vyjímku
5  } catch(Exception e){ //zpracuj vyjímku
6  } finally{
7    //zavři vytvořený stream
8  }
9
10 //verze Java 7 a výš (umožnění try-with-resources bloku a zřetězených
       vyjímek)
11 try(vytvoř stream k souboru, databázi aj.){
12     //zpracovávej data
13 } catch(IOException | Exception e){
14     //zpracuj vyjímku
15 }
```

Dále byla do Javy 7 přidána mimo jiné bezpečnostní a kryptografická implementace eliptické křivky (Elliptic Curve Cryptography – ECC) [31, 32, 33].

3.2.3 Android a Android Studio

Android je operační systém pro mobilní zařízení, který byl vyvinut firmou Google. Mezi základní prvky operačního systému Android patří aktivity představující akce, které se vykonají po volbě uživatele, například v menu. Tyto aktivity mají daný svůj životní cyklus, který je znázorněn na obrázku 3.7.



Obr. 3.7: Životní cyklus Android aktivity.

Z obrázku je vidět, že první metoda, která se volá po vytvoření Android aktivity, je metoda `onCreate()`. Při ukončení aktivity se volá metoda `onStop()`, popřípadě `onDestroy()`. Z tohoto faktu lze jednoduše odvodit, že metoda `onCreate()` by měla být přepisována ve většině tříd už jen z důvodů nastavení cesty k layout souboru, který se zobrazuje uživateli na displeji jeho zařízení. V diagramu tříd na obrázku 3.4 je vidět, že většina tříd přepisuje metodu `onCreate()` [30]. Pro potřeby znázornění životního cyklu Android aktivity v této práci, stejně jako u většiny předchozích obrázků, byl použit samotný program LaTeX a balíček tikz[1].

Android Studio je oficiální IDE (Integrated Development Environment) pro Android. Poskytuje nejrychlejší nástroje pro kompilování aplikací na každý typ Android zařízení: editace tříd, debugging, výkonnostní nástroje, flexibilní kompilovací systém atd. [27].

[1]Balíček tikz je hlavní balíček v distribuci LaTeX pro kresbu vektorové grafiky. Na adrese: http://www.texample.net/media/pgf/builds/pgfmanualCVS2012-11-04.pdf je obsažena dokumentace k balíčku tikz. Pro snadnost použití balíčku tikz je též vhodné se podívat na příklady obrázků, které jsou dostupné na stránce: http://www.texample.net/tikz/examples/. Obrázek znázorňující Android aktivitu byl po komunikaci s provozovatelem stránky umístěn na tento server na adrese http://texample.net/tikz/examples/android/.

3.2.4 YouTube API

YouTube API (Application Programming Interface) umožňuje vývojářům přistupovat k videím na serveru YouTube. Vývojáři využívají YouTube přehrávače, data a nástroje, které poskytuje YouTube API.

YouTube přehrávač

Videa z YouTube obvykle zobrazujeme v různých přehrávačích, jako je IFrame, Android Player nebo iOS player.

IFrame se používá zejména tehdy, pokud chceme vkládat video ze serveru YouTube do webové stránky a poté řídit toto video pomocí JavaScriptu. Požadavkem na možnost využití tohoto API pomocí tagu <iframe> je, že koncový uživatel musí mít prohlížeč s podporou HTML5. Pomocí funkcí JavaScript rozhraní API můžeme videa přehrát, pozastavit, upravit hlasitost přehrávače, změnit kvalitu videa a získávat informace o přehrávaném videu. Můžeme přidat i různé posluchače událostí (listeners), které vyvolají akci na základě vyhodnocení předchozího stavu, např. spuštění videa, změny kvality přehrávání na HD (High Definition), ...

Příklad použití tagu <iframe> je vidět na následujícím kódu:

```
1 <!DOCTYPE html>
2 <html>
3     <body>
4     <iframe id="p" type="text/html" width="640" height="390"
5     src="http://www.youtube.com/embed/M7lc1UVf-VE?
          enablejsapi=1&origin=http://example.com"
6     frameborder="0"></iframe>
7     </body>
8 </html>
```

Android player a iOS player umožňují přistupovat k videu v Android a iOS aplikaci. Android player, jeho možnosti a integrace do projektu je popsána v podsekci 3.3.4.

YouTube data a zdroje

YouTube Data API (v3) slouží k přidávání vlastností zahrnujících možnost nahrávání videa na YouTube server, vytváření a vedení playlistů atd.

YouTube Analytics API se využívá k získání statistik, populárních metrik a další pro YouTube videa a kanály. Metriky mohou být například: Jak často jsou vaše videa zobrazována? Kolik času uživatelé typicky stráví pozorováním vašeho videa? Jak se zobrazení odlišuje v závislosti na zemi?

YouTube Live Streaming API umožňuje vytvářet, aktualizovat a spravovat živé přenosy na YouTube. Pomoci tohoto API můžete naplánovat události a asociovat je s video streamy.

Server YouTube podporuje 2 typy volání této API, a to REST (Representational State Transfer) a XML-RPC (Extensible markup language – remote procedure call). Volání pomocí REST API si předává zprávy ve formátu JSON, což je v současnosti upřednostňováno před XML vzhledem k menší paměťové náročnosti formátu JSON [40].

3.2.5 Gradle

Gradle je open source automatizovaný systém pro sestavení projektů, který je založen na konceptu Apache Ant a Apache Maven a představuje doménově specifický jazyk (DSL), na rozdíl od Antu a Mavenu, které jsou založeny na XML konfiguracích [34]. Část souboru `build.gradle` z projektu je vidět na následujícím kódu.

```
1 apply plugin: 'com.android.application'
2 repositories {
3     mavenCentral()
4 }
5 dependencies {
6     compile fileTree(dir: 'libs', include: ['*.jar'])
7     compile 'com.android.support:appcompat-v7:24.2.1'
8     compile 'com.github.paolorotolo:appintro:4.0.0'
9 }
```

V části `dependencies` {...} je vidět, které knihovny budou zkompilovány a rovněž část, kde je definován typ aplikace jako Android aplikace pomocí `apply plugin 'com.android.application'`.

3.2.6 MySQL a MySQL Workbench

MySQL je open source relační databázový systém vytvořený švédskou firmou MySQL AB v roce 1995. V současnosti vlastní MySQL společnost Oracle Corporation, která patří mezi přední poskytovatele relačních databázových systémů. V tomto projektu byl databázový systém MySQL použit z důvodů jeho volné rozšiřitelnosti a dostačující podpory požadovaných funkcionalit. Jako volně dostupný MySQL databázový hosting byl použit hosting na serveru: https://openshift.redhat.com/app/console/application_types od společnosti RedHat, kde byla vybrána aplikace s podporou Javy, konkrétně Tomcat 7 (JBoss EWS 2.0). Na tomto serveru je možné mít zadarmo 3 aplikace s maximální velikostí 1 GB.

MySQL Workbench je nástroj určený pro databázové architekty a vývojáře. Umožňuje navrhnout strukturu databáze, na základě níž se dají vygenerovat SQL příkazy, poté se připojit k určené databázi a vygenerovat tabulky [35].

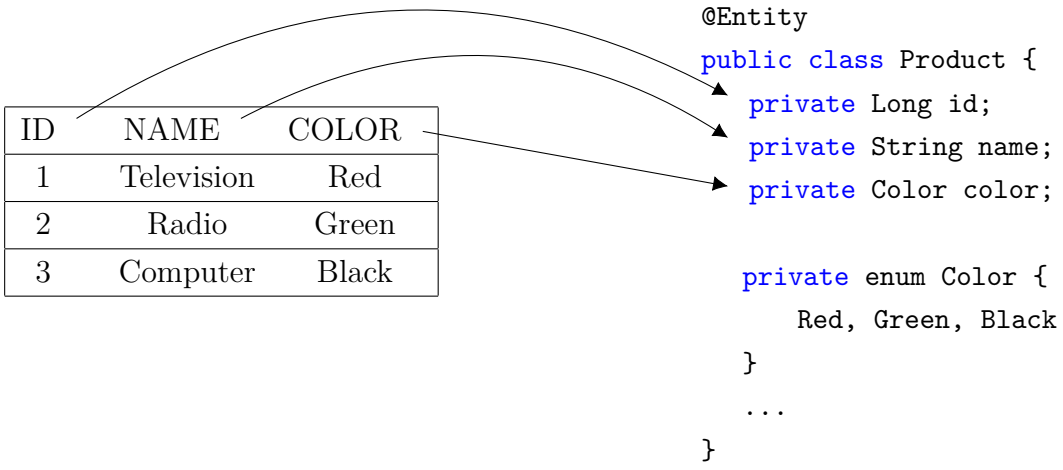
3.2.7 Hibernate

Hibernate je framework, který slouží k objektově relačnímu mapování (Object Relation Mapping). Jde o implementaci JPA (Java Persistence API), která je standardem pro objektově relační mapování v Javě. Pomocí tohoto frameworku není nutné psát základní MySQL dotazy pro `insert, update, delete, select` dat z databáze, ale vše lze provést pomocí základních operací nad entitami vytvořenými v jazyku Java.

Názornou ukázku principu tohoto mapování je vidět na obrázku 3.8. V takto namapovaných třídách jsou poté pomocí anotací nad jednotlivými atributy třídy definovaná různá omezení, např. anotace pro nenulový a unikátní atribut by vypadala následovně: `@Column(nullable=false, unique=true)`.

Obvykle se objektově relační frameworky používají v informačních systémech.

Mezi výhody tohoto frameworku patří menší množství kódu, ukládání do mezipaměti, dávkové operace, a také není svázán s konkrétním SQL dialektem, takže se dá jednoduše přejít na jiný relačně databázový systém [36].



Obr. 3.8: Ukázka mapování v JPA.

Nevýhody tohoto frameworku jsou v potenciálních výkonnostních problémech, velké abstrakci, učící křivce a v horší kontrole nad finálními SQL dotazy.

3.2.8 REST API

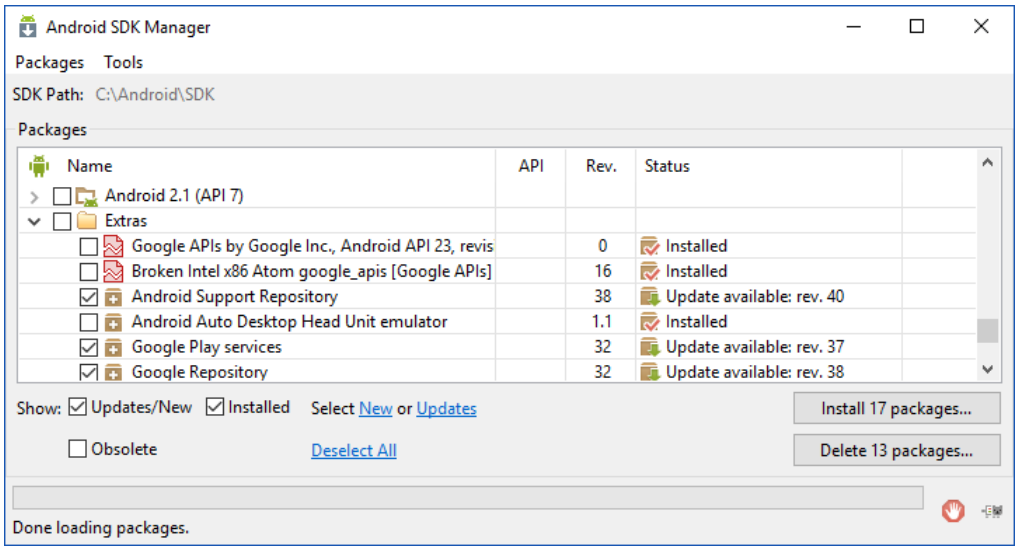
Tato architektura pro webové API byla vytvořena jako Ph.D. práce Roye Fieldinga v roce 2000. Jde o softwarovou architekturu, která je určitým druhem reverzního inženýrství vzhledem k funkcím webu. HTTP a URI byly napsány s REST principy ještě předtím, než byly formalizovány. REST byl vytvořen v kontextu HTTP protokolu, ale není limitován pouze na tento protokol. Využívá metody HTTP, jako jsou `@GET, @PUT, @POST, @DELETE, @HEAD, @OPTIONS, @PATCH`. Pomocí těchto metod a daných URL (Uniform Resource Locator) přistupuje ke zdrojům služby (zdroje služby jsou unikátně identifikovatelné pomocí URIs), které modifikuje, čte či vytváří [37].

3.3 Implementace aplikace

V následující části této práce jsou popsány prerekvizity k vytvoření Android projektu, stručný postup vytvoření, kompilace projektu a následná integrace YouTube API do takto vytvořeného projektu.

3.3.1 Prerekvizity k vytvoření Android projektu

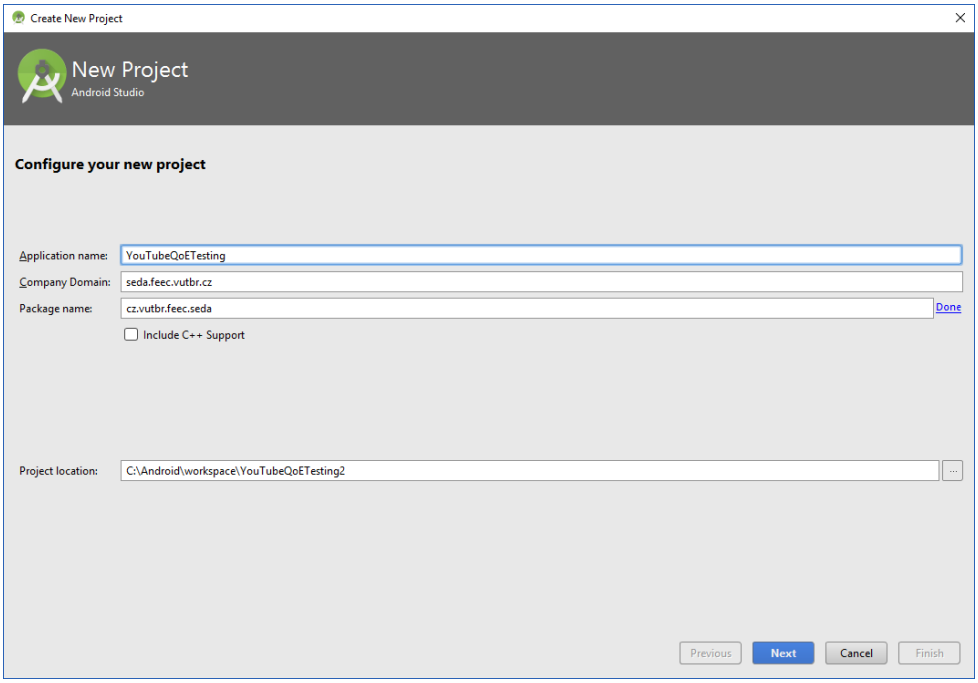
Pro vytvoření Android aplikace bylo potřeba si nejprve stáhnout JDK (Java Development Kit), je k dispozici například na stránce `http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html`, kde v části Java se Development Kit 7u79 byly přijaty licenční podmínky a stažen balíček jdk-7u79 dle operačního systému (OS). Po stažení následovala instalace JDK na OS. Dalším krokem je nainstalování vývojového prostředí, ve kterém se vyvíjí aplikace. Pro účely projektu je použito vývojové prostředí Android studio, které se dá stáhnout např. na stránce `https://developer.android.com/studio/index.html`. Po stažení Android Studia bylo nutné zkontrolovat, jestli součástí balíčku byl i SDK (Software Development Kit), který představuje množinu vývojářských nástrojů používaných k vývoji aplikací na platformě Android. Obsahuje emulátor, debugger, požadované knihovny, dokumentaci pro aplikační programová rozhraní Android API atd. Součástí SDK by měl být i SDK Manager, který se používá pro stahování požadovaných knihoven zabudovaných v Android API. Na obrázku 3.9 s ukázkou SDK Manageru je vidět, že nabízí aktualizování Google Play services atd.



Obr. 3.9: Ukázka SDK Manageru.

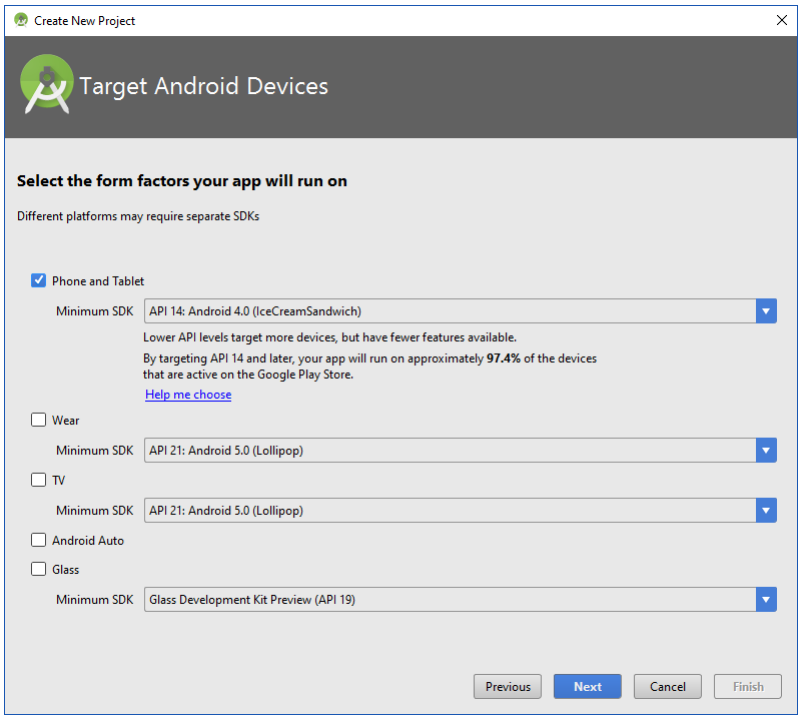
3.3.2 Vytvoření Android projektu

Po spuštění Android studia se klikne na **Start a new Android Studio project**. Otevře se okno znázorněné na obrázku 3.10. Odtud je patrné, že jako Application name bylo nastaveno YouTubeQoETesting a Company domain je seda.feec.vutbr.cz. Z Application name a Company domain se generuje Package name, které bylo editováno na cz.vutbr.feec.seda z cz.vutbr.feec.seda.youtubeqoetesting.



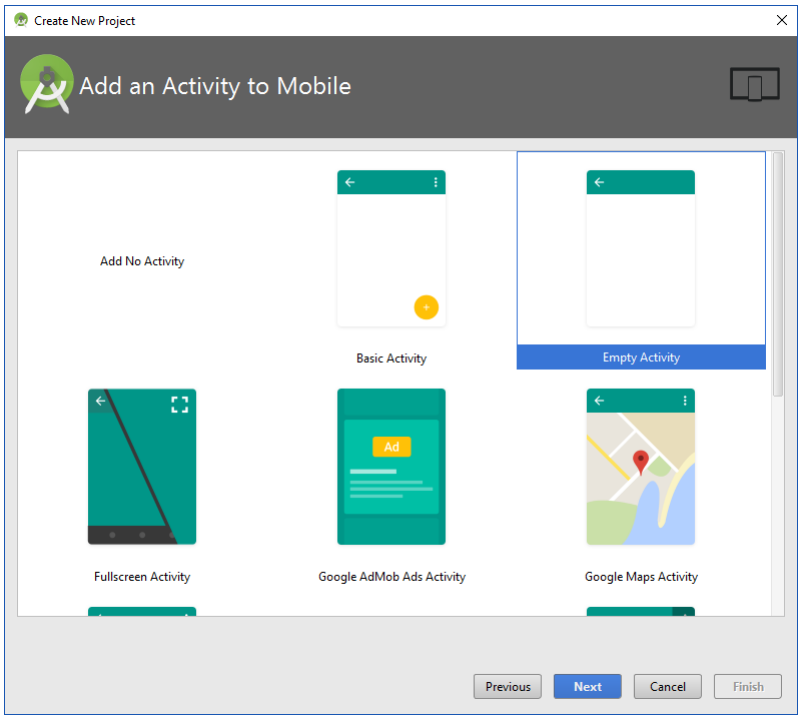
Obr. 3.10: Vytváření nového projektu v Android Studiu.

Po vyplnění údajů na obrázku 3.10 je třeba stisknout tlačítko **Next**. Otevře se nové okno, viz obrázek 3.11. Zde je vidět, že cílová zařízení jsou mobily a tablety. Tato cílová zařízení musí mít verzi Androidu 4.0 (IceCreamSandwich) a vyšší. Dle informací poskytnutých Android Studiem by restrikce na verzi 4.0 a výše měla pokrýt přibližně 97.4 % zařízení, která jsou aktivována na Google Play Store.



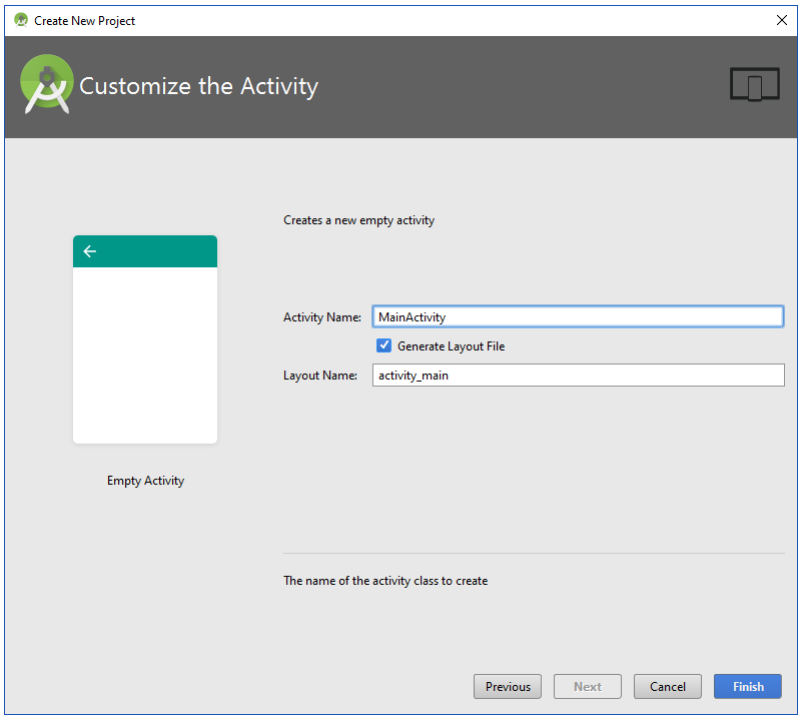
Obr. 3.11: Výběr cílových zařízení.

Po selekci cílových zařízení a požadavků na cílová zařízení se pokračuje stisknutím tlačítka **Next**. Otevře se okno s výběrem základní aktivity projektu, viz obrázek 3.12.



Obr. 3.12: Výběr Android aktivity do projektu.

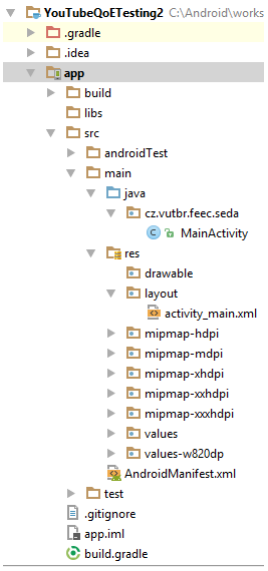
Jako základní je vybrána aktivita typu Empty Activity a stiskne se tlačítko **Next** (výběr této aktivity znamená, že projekt zatím neobsahuje žádné předpřipravené aktivity, které nabízí Android Studio). Zobrazí se okno na obrázku 3.13, kde do pole Activity Name je vloženo MainActivity, do pole Layout Name je vloženo activity_main a je zaškrtnuto Generate Layout File, což způsobí, že se vytvoří základní layout soubor pro aktivitu MainActivity. Poté se stiskne tlačítko **Finish**, čímž se vytvoří nový Android projekt.



Obr. 3.13: Dokončení vytvoření nového Android projektu.

Struktura takto vytvořeného projektu ve vývojovém prostředí nyní vypadá jako na obrázku 3.14, kde je vidět třída[2] `MainActivity`, layout soubor `activity_main.xml`, `AndroidManifest.xml`, `build.gradle` a další. Třída `MainActivity` je `.java` soubor, představující aktivitu, která po sestavení a spuštění aplikace načte soubor `activity_main.xml`. V tomto `.xml` souboru je definován vzhled spuštěné aplikace, tzn. do tohoto souboru přidáváme různá tlačítka, textová pole a další, která pak uživatel vidí ve své aplikaci (`.java` soubory dodávají logiku k těmto nadefinovaným tlačítkům, polím atd.).

[2]Třída je množina objektů s určitými vlastnostmi. Přitom samotná třída nedefinuje konkrétní objekty třídy, jen udává, jaké vlastnosti bude mít každý objekt třídy (podobně představa o tom, co je telefon, může existovat, i kdyby všechny existující telefony zanikly).



Obr. 3.14: Struktura nově vytvořeného projektu ve vývojovém prostředí.

Soubor `AndroidManifest.xml` popisuje základní charakteristiky aplikace a definuje jednotlivé komponenty. Pro nově vytvořený projekt `android manifest` soubor vypadá následovně:

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="cz.vutbr.feec.seda">
4      <application
5          android:allowBackup="true"
6          android:icon="@mipmap/ic_launcher"
7          android:label="@string/app_name"
8          android:supportsRtl="true"
9          android:theme="@style/AppTheme">
10         <activity android:name=".MainActivity">
11             <intent-filter>
12                 <action android:name="android.intent.action.MAIN"/>
13                 <category
                       android:name="android.intent.category.LAUNCHER"/>
14             </intent-filter>
15         </activity>
16     </application>
17 </manifest>
```

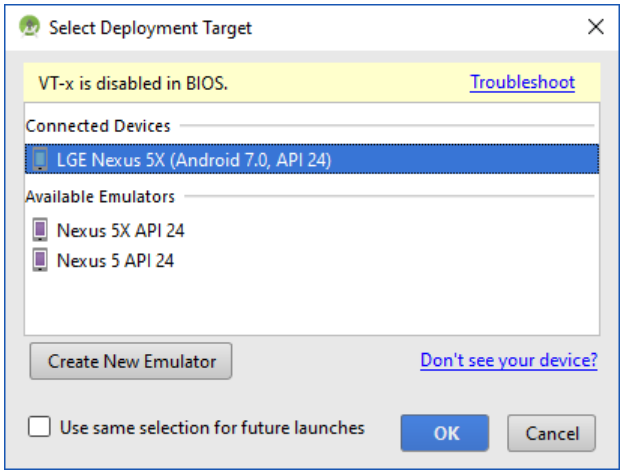
Na kódu výše je vidět, že v tomto souboru se dá definovat ikona aplikace, její jméno, základní kompozice projektu a všechny aktivity, které jsou naprogramovány v `.java`

souborech.

Soubor `build.gradle` je soubor, který slouží k překompilování/stáhnutí závislostí na různé knihovny, konfigurace minimální verze Androidu na cílových zařízeních atd.

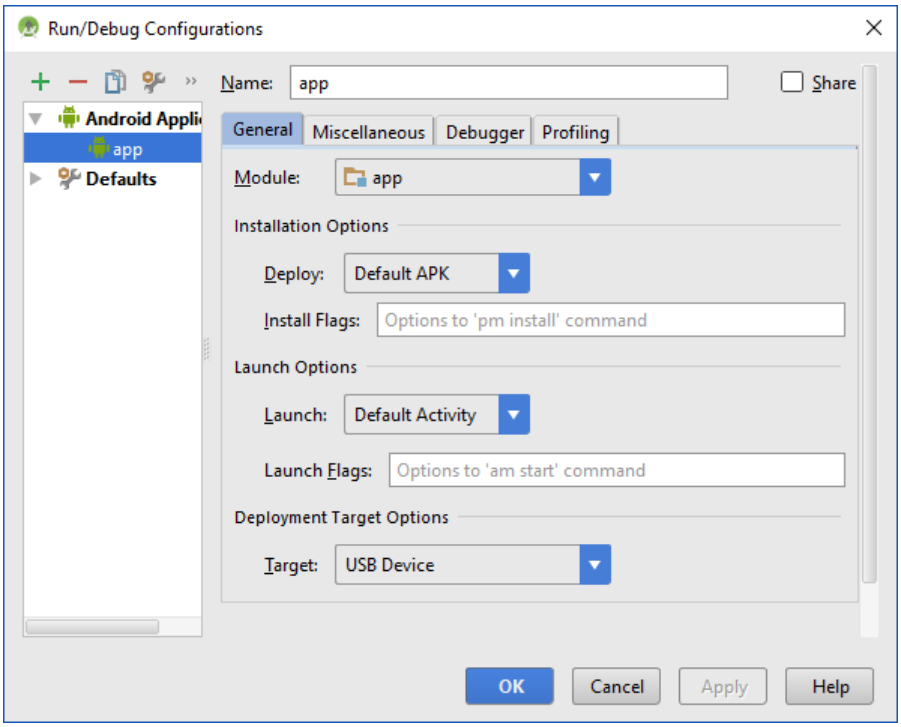
3.3.3 Kompilace Android aplikace

Pro kompilaci aplikace v Android Studiu se používá emulátor nebo připojené zařízení, viz obrázek 3.15. Toto dialogové okno se objeví, pokud se v Android Studiu v daném projektu stiskne tlačítko **Run** a není vybrán žádný Deployment Target.



Obr. 3.15: Výběr cílového zařízení pro kompilaci emulátor/připojené zařízení.

V případě, že bude pro testovací účely vybráno připojené zařízení, pak je nutné u něj povolit tzv. USB debugging. To bývá zpravidla v nastaveních telefonu v sekci Informace o telefonu. Pro kontrolu je ještě nutné se podívat do Run Configurations, kde je nutné v případě, že už tak není učiněno, nastavit Deployment Target Options → Target na USB Device, viz obrázek 3.16. Po těchto nastaveních, připojení cílového zařízení a stisknutém tlačítku **Run** v Android studio je nahrán na cílové zařízení `.apk` soubor, který představuje spustitelný soubor pro mobilní aplikace a je spuštěna vytvořená aplikace.



Obr. 3.16: Nastavení cílového zařízení na testování.

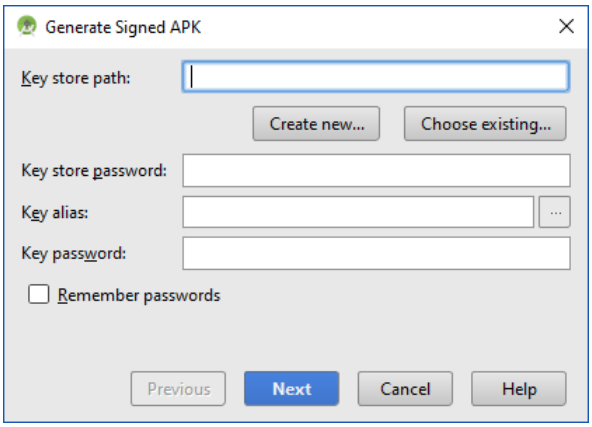
3.3.4 Integrace YouTube Android API do projektu

Prvním stěžejním bodem integrace YouTube API do vlastního Android projektu je stažení balíčku YouTubeAndroidPlayerApi, který je dostupný na adrese `https://developers.google.com/youtube/android/player/downloads/`. Dalším bodem je registrace aplikace, včetně digitálně podepsaného souboru `.apk` veřejným klíčem na Google Developers Console, aby se dalo použít YouTube Android Player API z aplikace. Pro registraci aplikace je nutné postupovat podle následujících kroků:

1. Na stránce `https://console.developers.google.com` se v levém horním rohu stiskne tlačítko **Dashboard** a poté **Enable API** (nebo stačí jít rovnou přes **Library**). Ze široké nabídky poskytovaných API od společnosti Google se vybere sekce YouTube APIs a konkrétně YouTube Data API, která poskytuje přístup k datům YouTube, jako jsou videa, playlisty a kanály. V nově otevřeném okně se stiskne tlačítko **Enable**. Tím je zajištěno, že je povoleno využívat službu YouTube Data API v3.

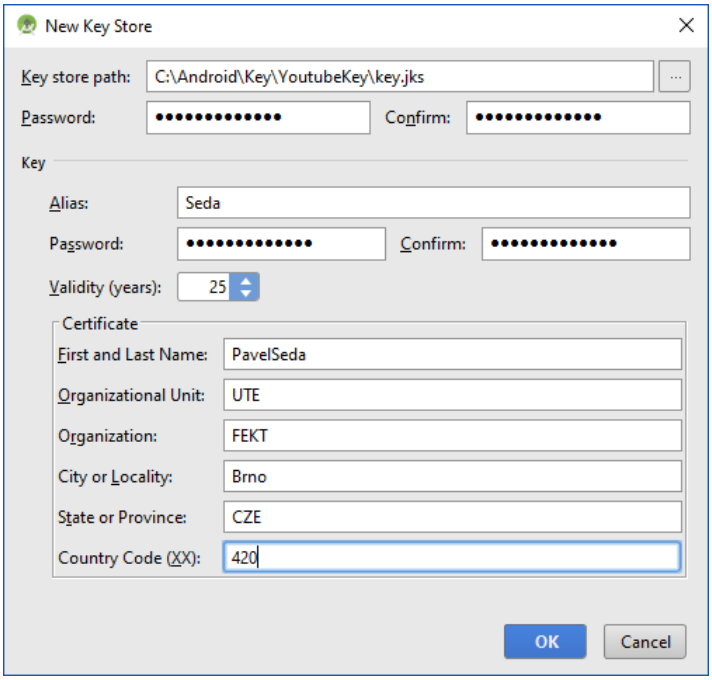
2. Nyní se k nově vytvořenému projektu YouTubeQoETesting nastaví přihlašovací údaje k YouTube Data API v3. YouTube Data API v3 umožňuje dva způsoby přístupu, buď OAuth 2.0, nebo pomocí API klíče. Na stránce je nutné vytvořit projekt s názvem YouTubeQoETesting a poté pro tento projekt v levé

liště nahoře se stiskne tlačítko **Credentials**, poté tlačítkem **Create credentials** se vybere typ, např. API key, stiskne se **RESTRICT KEY**, pokud není žádoucí, aby tento API klíč mohla použít neautorizovaná osoba. V sekci key restriction se vybere Android apps. Nyní ve spuštěném Android studiu se stiskne **Build** → **Generate Signed APK**, což otevře okno, znázorněné na obrázku 3.17.



Obr. 3.17: Generování podepsaného .apk souboru nastavení, část 1.

Stiskne se tlačítko **Create new...** a otevře se okno, které se vyplní obdobně, jak je vidět na obrázku 3.18. Zapíše se heslo, protože bude potřebné pro vygenerování SHA1 (Secure Hash Algorithm 1) klíče.

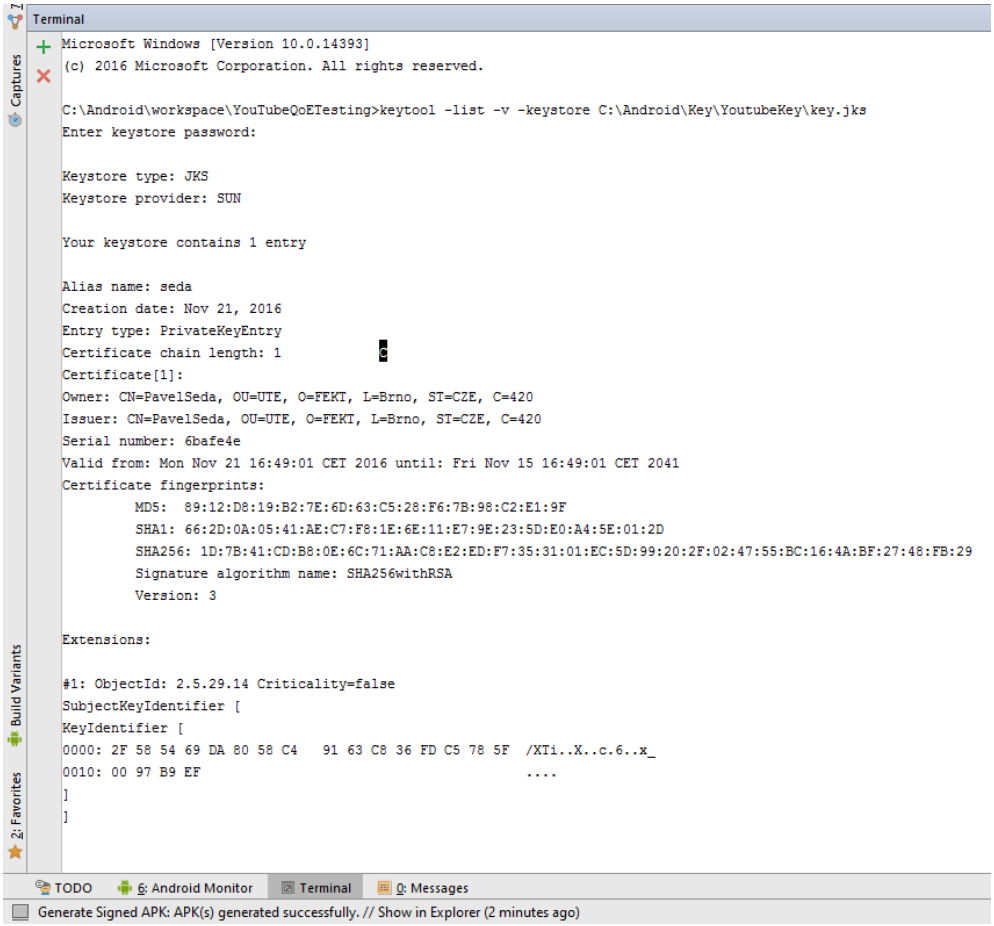


Obr. 3.18: Generování podepsaného .apk souboru nastavení, část 2.

Otevře se terminál v Android Studiu, kde se vloží příkaz dle následujícího vzoru:

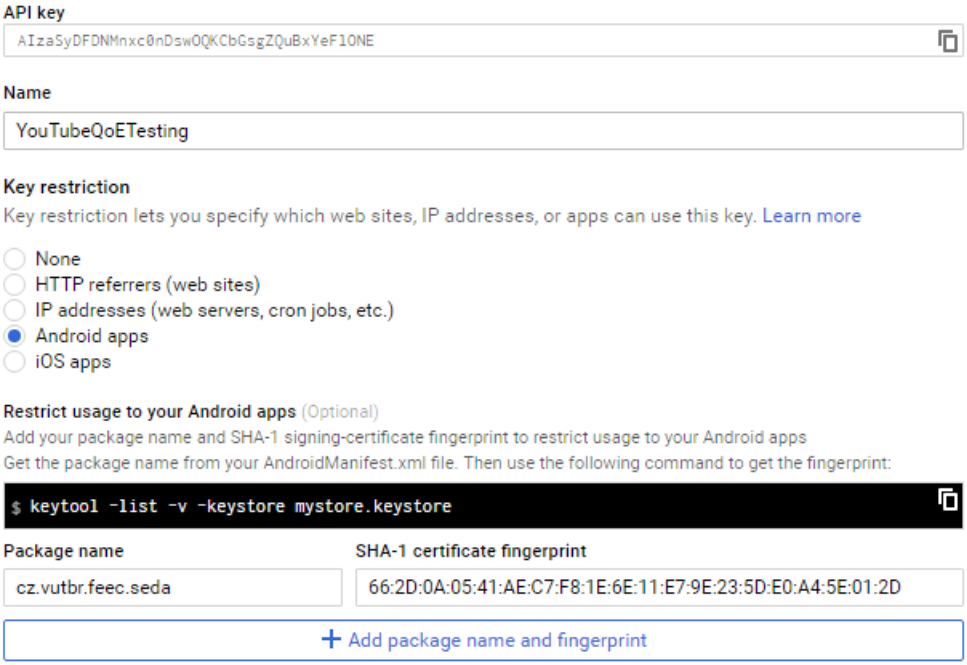
```
$ keytool -list -v -keystore mystore.keystore.path
```

Terminál z Android Studia poté bude vyžadovat heslo, které jsme si zapsali při generování podepsaného `.apk` souboru ve výše popsaných krocích, zadá se heslo a poté je vidět v terminálu vygenerovaný SHA1 klíč obdobně jako na obrázku 3.19



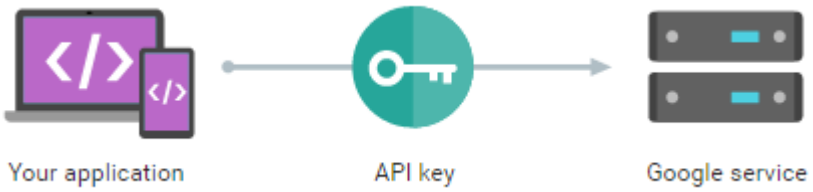
Obr. 3.19: Vygenerovaný SHA1 soubor pro podepsaný .apk soubor.

Zkopíruje se vygenerovaný SHA1 hash a na stránce, kde je vytvořený YouTubeQoETesting projekt se stiskne tlačítko **Add package name and fingerprint** a vloží se package name vytvořeného projektu a vygenerovaný SHA1 hAsh, stiskne se tlačítko **Save**. Stránka by nyní měla vypadat jako na obrázku 3.20.



Obr. 3.20: Vložený SHA1 hash a package name.

Tím je zaručeno, že žádná neautorizovaná osoba nevyužije tento API key na přístup k YouTube Android Player API. Využití samotného API klíče je znázorněno na obrázku 3.21.

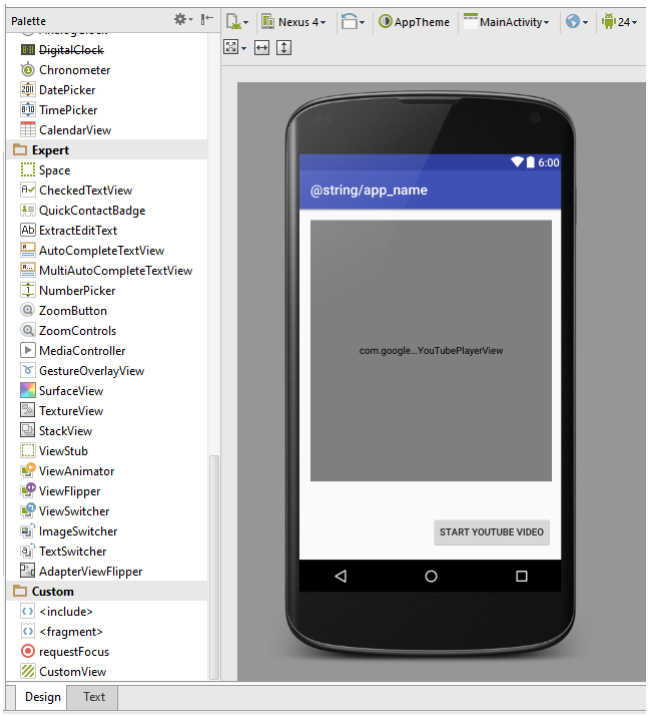


Obr. 3.21: Znázornění využití API klíče ke Google službě.

3. Posledním krokem je vložení API klíče a YouTube Data API v3 do projektu YouTubeQoETesting [41, 42]. Ve vytvořeném Android projektu se vloží knihovna YouTubeAndroidPlayerApi.jar do složky libs, API klíč se uloží v souboru `strings.xml` v následující podobě:

```
1 <string name="YOUTUBE_API_KEY"> Generated API key </string>
```

Otevře se soubor `activity_main`, v kterém se otevře perspektiva Design a na paletě komponent v sekci Custom se vybere CustomView, otevře se dialogové okno, ve kterém se vybere YouTubePlayerView a vloží se do obrazovky, poté by měla vypadat perspektiva Design zhruba obdobně jako na obrázku 3.22.



Obr. 3.22: Vložení YouTube API do layout souboru.

V perspektivě Text by nyní měla být vidět následující část kódu, ve které se upravilo id na youtube_view:

```
<com.google.android.youtube.player.YouTubePlayerView
    android:id="@+id/youtube_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="0.29" />
```

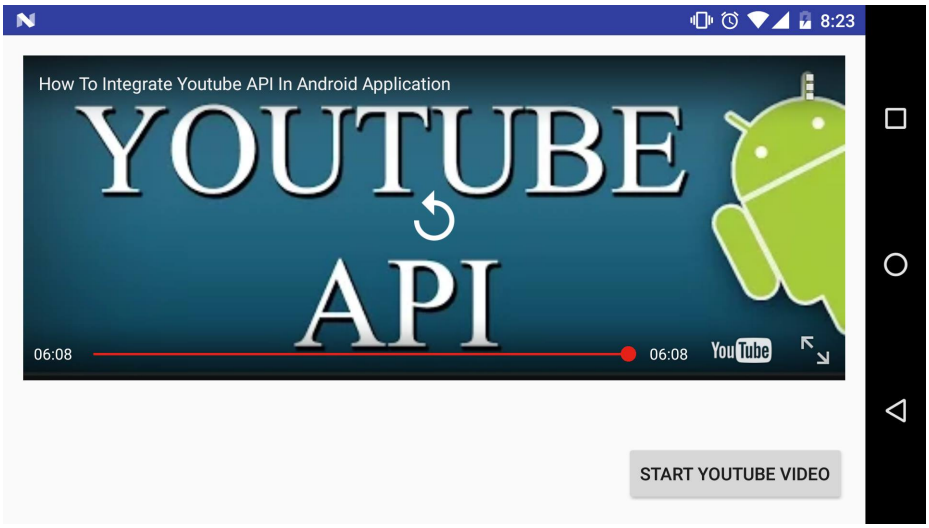
Jako poslední konfigurační záležitost je nutné nastavit v souboru `AndroidManifest.xml` uvnitř tagu manifest následující blok kódu, který vyžaduje, že dané zařízení musí mít přístup k internetu.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Nyní je nezbytné napojit na layout soubor `activity_main` logiku v javovském souboru `MainActivity.java`. Tato třída po implementování vypadá následovně:

```
1  public class MainActivity extends YouTubeBaseActivity {
2      @Override
3      protected void onCreate(Bundle savedInstanceState) {
4          super.onCreate(savedInstanceState);
5          setContentView(R.layout.activity_main);
6
7           YouTubePlayerView youTubePlayerView =
                (YouTubePlayerView) findViewById(R.id.youtube_view);
8          YouTubePlayer.OnInitializedListener initListener = new
               YouTubePlayer.OnInitializedListener(){
9              @Override
10             public void onInitializationSuccess
                   (YouTubePlayer.Provider provider, YouTubePlayer
                   youTubePlayer, boolean b) {
11                 youTubePlayer.loadVideo("a4NT5iBFuZs");
12 };}}}
```

Pro každou Android aktivitu, která má přistupovat k YouTube API je nutné, aby dědila z třídy `YouTubeBaseActivity`, což vidíme v předcházejícím kódu `MainActivity extends YouTubeBaseActivity`. V kódu je spuštění YouTube videa implementováno jako událost, která se spouští po stisknutí tlačítka **start youtube video**, tím se inicializuje `youTubePlayerView` s uloženým API klíčem (pro zkrácení ukázky kódu bylo toto tlačítko vyňato z ukázkového kódu stejně jako nutnost `@Override onInitializationFailure` při inicializaci třídy `YouTubePlayer.OnInitializedListener()`). Při úspěšné inicializaci je spuštěno video s klíčem `a4NT5iBFuZs`, což je identifikátor určitého videa na YouTube, celá adresa tohoto videa je `https://www.youtube.com/watch?v=a4NT5iBFuZs`, pro spuštění ovšem stačí zadávat pouze identifikátor videa. Nyní je vše nastavené a po zkompilování aplikace bychom ji měli vidět obdobně jako na obrázku 3.23.



Obr. 3.23: Screen save z mobilní aplikace po integraci YouTube API.

3.3.5 Úprava aplikace pro potřeby subjektivního testování

Pro subjektivní testování je důležité zajistit, aby bylo v přehrávaném videu možné měnit rozlišení, nastavovat adaptivní streamování aj. Bohužel při bližším prozkoumání YouTube Android API bylo zjištěno, že toto API pro Android v současné době neumožňuje z programového kódu měnit rozlišení videa ani nastavení na adaptivní streamování. Absence této možnosti jde vidět přímo ve výčtu podporovaných metod z dokumentace YouTube Android API [43] v části public methods. Bohužel v této třídě `YouTubePlayer` při volání metody `loadVideo` nebo `cueVideo` nejde spustit video s určitým rozlišením ani při použití alternativního řešení v podobě přidání řetězce (např. `"?vq=hd1080"`) k identifikátoru videa, který definuje typ rozlišení, ve kterém má být video spuštěno (tato alternativa funguje pouze prostřednictvím webového rozhraní, ale YouTube Android API to nepodporuje).

I přesto může být YouTube Android API využito při dalších možnostech aplikace, jako je vyhledávání videí na YouTube, viz podsekce 3.3.13.

Východiskem nemožnosti použití YouTube Android API pro subjektivní testování v rámci nastavování typu streamingu a kvality videa je použití třídy `WebView`, která umožňuje pouštět webové stránky z mobilní aplikace.

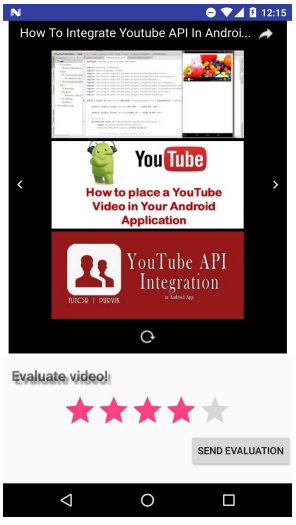
Alternativou využití YouTube Android API tedy představují tři základní možnosti pro mobilní aplikace, které jsou popsány dále.

Přístup k videím přímo na YouTube a načítání do `WebView` pomocí `iframe` tagu

Ve variantě přistupování k videím přímo na YouTube se dá video spustit s příznakem `/embed/`, což značí, že se video zobrazí na celé stránce, a tedy celá stránka bude působit pouze jako video. V této variantě už se dá využít <iframe> tag, viz podsekce 3.2.4, a aplikovat příznak definující rozlišení videa. Příklad použití třídy `WebView` s puštěním YouTube videa je vidět na následujícím kódu:

```
1 String frameVideo = "<html><body><iframe width=\"100\%\"
      height=\"100\%\"
      src=\"https://www.youtube.com/embed/a4NT5iBFuZs?vq=large\"
      frameborder=\"0\" allowfullscreen></iframe></body></html>";
2 WebView displayYoutubeVideo = (WebView) findViewById(R.id.webview1);
3 displayYoutubeVideo.setWebViewClient(new WebViewClient() {
4     @Override
5     public boolean shouldOverrideUrlLoading(WebView view, String url)
          {
6         return false;
7     }
8 });
9 displayYoutubeVideo.loadData(frameVideo, "text/html", "utf-8");
```

Screenshot z aplikace používající `WebView` je vidět na obrázku 3.24.



Obr. 3.24: Aplikace při použití WebView místo YouTube Android API.

Je použito stejné video jako na obrázku 3.23 v integraci YouTube Android API. Zajímavé je, jak se změnila URL, kterou voláme na `https://www.youtube.com/`

embed/a4NT5iBFuZs?vq=large, kde je patrná nutnost vložení /embed/ a ?vq=large pro zobrazení videa ve specifikovaném rozlišení.

Pro testování subjektivního QoE bude zřejmě nutné přidat javascript, kterým bude možné po určité době měnit rozlišení či streamování. Nevýhodou tohoto přístupu je, že nemáme plnou kontrolu nad videi.

Uložení videí v mobilním zařízení a spouštění videí přímo v mobilním zařízení bez využití vlastního serveru či YouTube serveru

Tato varianta nebyla použita, protože by vedla k tomu, že při vysokém počtu videí by aplikace byla příliš velká. Mobilní aplikace mívají velikost zhruba okolo 10 – 100 MB a jelikož pro měnění kvality videí, kde máme například video o 4 kvalitách, je nutné mít uložené toto video 4× v dané kvalitě. Taková aplikace by mohla mít při vyšším počtu videí i 1 GB, a to je pro mobilní zařízení nepřípustné. Nicméně kdyby tato varianta byla použita, pak by byla výhodná ve smyslu decentralizované zátěže vykonávání operací a možnosti využití metody měření Crowdsourcingu, kde se simuluje adaptivní streamování s lokálními videi. Operace spojené s videem by se vykonávaly na mobilních zařízeních, a nebyl by tedy zatěžován žádný centralizovaný prvek.

Implementace serverové části, kde jsou uložena videa a zobrazována z mobilní aplikace

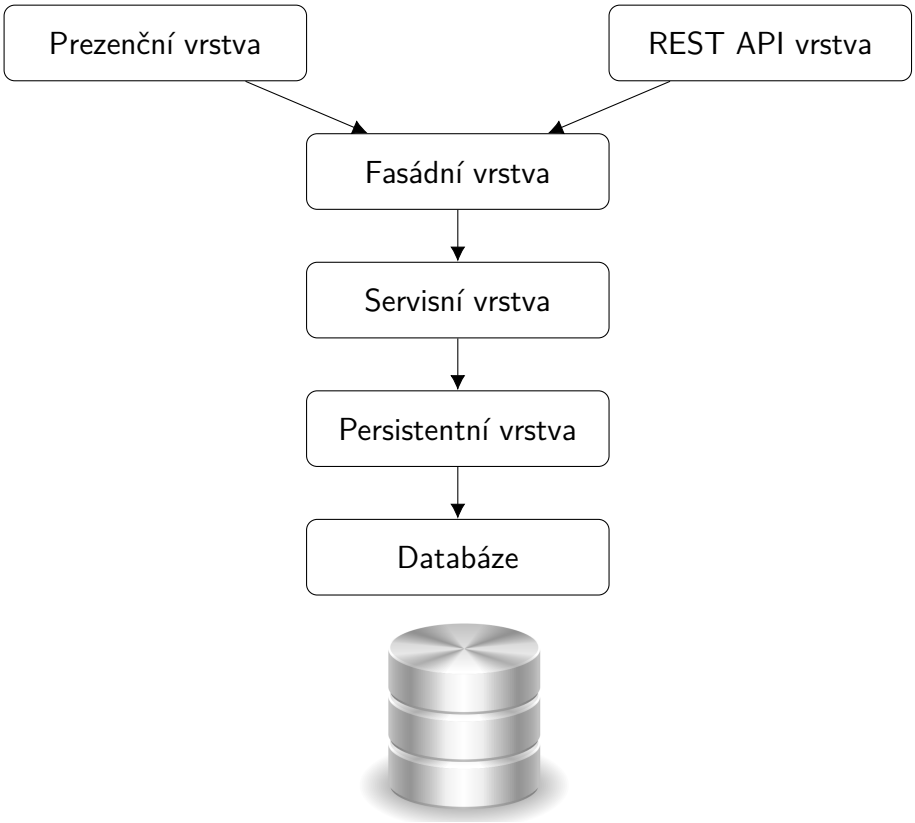
Na serveru se vykonávají akce, jako je načtení parametrů z databáze o změně rozlišení v určitých časových okamžicích, načtení loadovacího obrázku atd. Všechny akce se vykonávají na serveru a mobilní zařízení si to pouze zobrazuje. Výhodou tohoto přístupu je, že máme plnou kontrolu nad videi, můžeme je bez výraznějších restrikcí modifikovat na straně serveru pomocí javascriptu, dále i to, že na tomto serveru může být centrální databáze přístupná pomocí REST API, a tak se i případné větší množství aplikací zabývajících se subjektivním měřením může připojovat k obecnému REST API a ukládat data do centrální databáze, kde potom mohou být jednoduše interpretována. Z výše uvedených důvodů byla tato varianta implementována.

Obecně nevýhodou řešení na serveru je přenesení zátěže na stranu serveru, což by při vysokém počtu uživatelů vyžadovalo výkonnost a stabilitu serveru.

3.3.6 Implementace serverové části aplikace

Pro implementaci serverové části aplikace byla zvolena robustní architektura v Java EE, která zajišťuje udržitelnost a dobrou rozšiřitelnost systému. Architektura

serverové části je rozdělena do několika vrstev, které jsou vidět na obrázku 3.25.



Obr. 3.25: Architektura serverové části aplikace implementované v Java EE.

Tato architektura zajišťuje, že uživatelské požadavky v prezenční či REST API vrstvě volají metody z fasádní vrstvy, kde jsou vytvořeny DTO (Data Transfer Object) třídy, které obsahují různé validace pro formuláře atd. Metody z fasádní vrstvy volají servisní vrstvu, kde jsou obsaženy všechny stěžejní algoritmy. Servisní vrstva získává a zapisuje data z persistentní vrstvy, která pracuje s databází.

Tyto vrstvy výrazným způsobem napomáhají k rozšiřitelnosti a čitelnosti systému, protože se části systému dělí do více vrstev s patřičnou logikou, a tím se předchází zbytečné robustnosti jednotlivých dílčích komponent.

3.3.7 Implementace integračních testů

Na serverové straně byly implementovány integrační testy, které pokrývají ověření funkcionality metod na persistentní, servisní, fasádní a REST vrstvě. Obvykle testy implementuje jiný programátor než ten, který implementoval funkcionalitu metody, ale vzhledem k tomu, že nejde o týmový projekt, byly testy psány přímo autorem funkčních metod.

Pro potřeby testů je použita na persistentní vrstvě in-memory Derby databáze, čímž se testy nevykonávají nad produkční databází, a nejsou tedy ohrožena produkční data. Vytvoření těchto testů napomáhá dobré udržitelnosti projektu, snazšímu hledání a odhalení chyb při „běhu“ aplikace. V následujícím kódu je ukázka testu na persistentní vrstvě, kde je vytvořeno video s náležitými atributy a je u něj změněno jméno. Tato změna je uložena do databáze a následně je otestováno, zda se opravdu změnilo jméno u videa (`em` je instance `PersistenceContextu`, který pracuje s databází).

```
1  @Test
2  public void testUpdate() {
3     video.setName("Jine jmeno pro video");
4     videoDao.save(video);
5     Assert.assertEquals(em.find(Video.class, video.getId()).getName(),
6           video.getName());
7  }
```

Naproti tomu na servisní vrstvě se zpravidla používá Mockování[3], kde nejsme vázáni závislostmi dané mocknuté instance, a vytvoří se simulace reálného chování daných instancí. Příklad takového testu je vidět v následujícím kódu, kde voláme na servisní vrstvě příkaz `update` k modifikaci nějakého parametru u videa a poté ověřujeme, zda se v metodě `update` na servisní vrstvě zavolala patřičná metoda z persistentní vrstvy, v tomto případě metoda `save`. Tím je ověřeno, že metoda na servisní vrstvě skutečně volá metodu z persistentní vrstvy a dané změny budou uloženy do databáze.

```
1  @Test
2  public void testUpdate() {
3     videoService.update(video);
4     verify(videoDao, times(1)).save(any(Video.class));
5  }
```

Podobně se testují metody i na fasádní vrstvě, kde testujeme, jestli se zavolaly metody ze servisní vrstvy, popřípadě testujeme ještě funkčnost dílčích volaných částí dané metody.

V rámci práce byly implementovány testy i na REST API vrstvě. Příklad části takového testu lze vidět v další ukázce kódu, kde `questionnaireDTO` je objekt představující předdefinovaně vytvořený dotazník, `questionnaireFacade` je in-

[3]Jde o o techniku psaní určitého druhu automatických testů. Prakticky jde o nahrazení reálného objektu testovací fasádou, která neprovádí žádnou funkcionalitu nahrazovaného objektu, jen se jako tento objekt tváří. Místo původní logiky objektu je vloženo chování, které je v testu potřeba, jsou testovány jen nezbytné součásti, nikoliv závislosti na daný objekt.

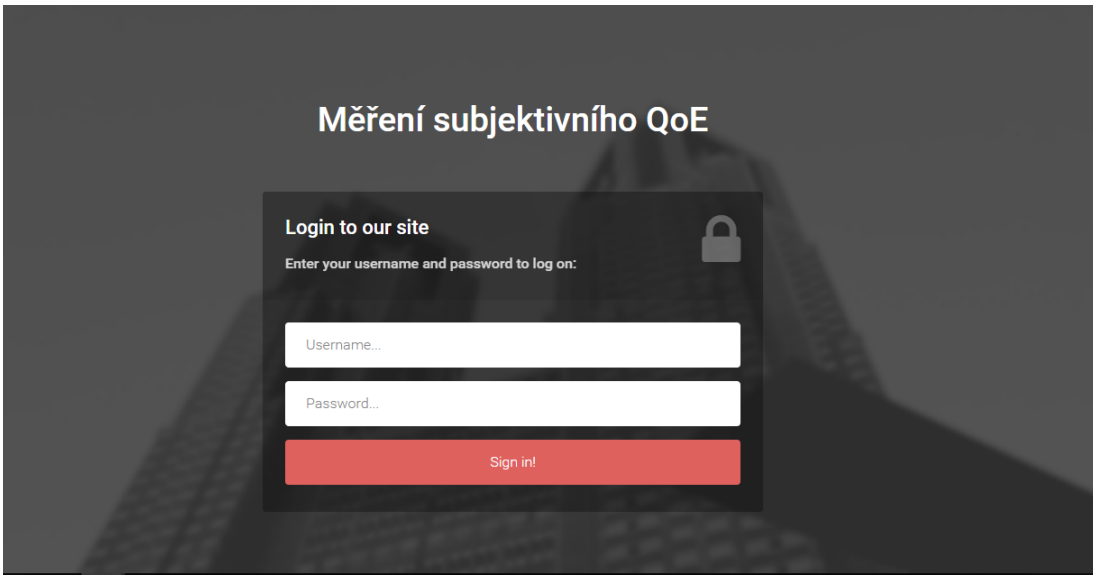
stance fasádního objektu zajišťujícímu operace nad dotazníky pro servisní vrstvu, `questionnaire` je objekt, který posíláme na REST API přes metodu `POST` a `convertObjectToJsonBytes` je metoda, která převede objekt na řetězec. Pomocí mockovacích testů se tedy pošle daný objekt na konkrétní URL, díky části `doReturn ...` se nevykonávají operace na vrstvách „pod“ REST vrstvou, ale vrátí se předdefinovaný objekt `questionnaireDTO`, čímž opravdu testujeme funkcionalitu samotné REST API a odstíníme se od závislosti na jiné vrstvy. Z volání této URL očekáváme, že HTTP status bude 200 (bezchybový) a že vrácený JSON bude obsahovat dotazník s emailem "testrest@seznam.cz" (předdefinovaný objekt `questionnaireDTO` obsahuje tento email), tím je otestováno, že daná URL je funkční.

```
1  @Test
2  public void createQuestionnaire() throws Exception {
3     QuestionnaireCreateDTO questionnaire = new
          QuestionnaireCreateDTO();
4     questionnaire.setEmail("pavelseda@email.cz");
5
6     doReturn(questionnaireDTO).when(questionnaireFacade).create(
7           any(QuestionnaireCreateDTO.class));
8
9     String json = convertObjectToJsonBytes(questionnaire);
10
11    mockMvc.perform(
12          post(ApiEndPoints.QUESTIONNAIRE_HATEOS).contentType(
13                MediaType.APPLICATION_JSON).content(json))
14          .andExpect(status().isOk())
15          .andExpect(
16                  content().contentTypeCompatibleWith(
17                        MediaType.APPLICATION_JSON_VALUE))
18          .andExpect(
19                jsonPath("\$.[?(@.id==1)].email").value(
20                      "testrest@seznam.cz"));
21 }
```

Obdobně jsou otestovány další metody, tím je zaručeno, že jsou jednotlivé metody implementovány správně, a lze tedy bez problémů využít tuto API pro posílání dat z mobilní aplikace do centrální databáze.

3.3.8 Implementace REST API na serveru

Pro komunikaci mobilní aplikace s databází je využito implementované REST API, čímž je vytvořena další abstraktní vrstva, která umožňuje komunikaci jakékoliv aplikace s databází přes tuto API pomocí HTTP dotazů na konkrétní URL. Aby nebylo možné číst, modifikovat a zapisovat do databáze komukoliv jen na základě znalosti URL a formátů dat, tak je také implementována nutnost autentizace, kdy při zadání jakékoliv URL mající vzor `../rest/**`, kde pro neautentizované uživatele je zobrazena vytvořená login stránka, viz obrázek 3.26, ve které je nutné se autentizovat pro operace nad REST API.



Obr. 3.26: Autentizační stránka pro přístup k REST API.

Hesla uživatelů jsou v databázi zašifrována pomocí blokové šifry AES v režimu CBC s velikostí bloku definovaným jako PKCS5Padding. V aplikaci se lze pro účely testování přihlásit pod Username `kovacd@feec.vutbr.cz` a heslem `admin`.

Implementace REST API byla provedena v jazyku Java ve spojení s frameworkem Spring, který je použit na serverové části aplikace. Zkrácená část implementace z jazyku Java pro vrácení všech dotazníků je vidět na kódu níže:

```
1 @RequestMapping(value = ApiEndPoints.QUESTIONNAIRE_HATEOS, method =
      RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
2 public final Collection<QuestionnaireDTO> getQuestionnaires()
3       throws JsonProcessingException {
4     try {
5         return questionnaireFacade.getAllQuestionnaires();
6     } catch (ResourceNotFoundException ex){
```

```
7           throw new ResourceNotFoundException(ex);
8       }
9   }
```

Ve skutečnosti je REST implementován pomocí principu HATEOS, kde je zobrazena konkrétní entita a na ostatní entity, popřípadě na operace s danými entitami, jsou představeny odkazy (links). Komplexní implementace REST API je vidět na GitHubu v balíčku `rest` [45].

Ukázka dotazu `HTTP GET` na adresu https://qoe-qtestq.rhcloud.com/rest/hateos/questionnaires a odpovědi ze serveru je vidět na obrázku 3.27.

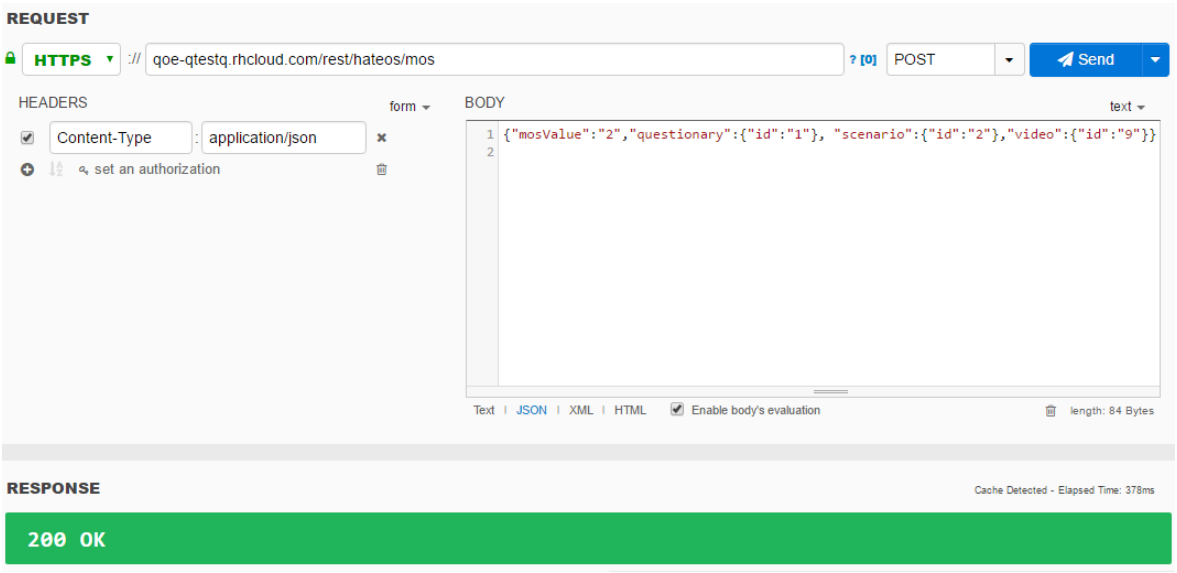
Obr. 3.27: Ukázka odpovědi serveru pomocí REST API na vrácení dotazníků z databáze.

Kromě základních dotazů byla implementována i možnost vyhledávání/filtrování výsledků v daném dotazu, které může mít například následující formát: https://qoe-qtestq.rhcloud.com/rest/hateos/questionnaires?search=age==24;school==zakladni, kde lze vidět, že se přidal parameter `search`, kde v daném dotazníku musí být atribut `age` roven 24 a atribut `school` s hodnotou základní.

Použitá notace pro vyhledávání a filtrování dat je definována v EBNF ISO

14977 [38]. Pro přehledné zobrazení JSON souboru byl použit plugin JSONView [39] do prohlížeče Chrome.

Kromě metody `HTTP GET` lze rovněž aplikovat další metody protokolu `HTTP`, jako je např. `POST`, kterým se dají posílat data na server, a tím ukládat data z mobilní aplikace do vzdálené databáze. Pro testování dotazů je vhodné využít některý z dostupných RESTClient pluginů [44], kde po poslání nového hodnocení MOS v JSON formátu, které se odkazuje na dotazník s id = 1, scénář s id = 2 a video s id = 2, je vidět na obrázku 3.28, že akce projde, a data jsou tedy uložena do databáze přes REST API. Pro znalost daných formátů k zaslání, získání a modifikaci dat je vytvořena dokumentace k REST API, která je přístupná na GitHubu [45]. K vykonávání dotazů je nejprve nutné být autentizován.

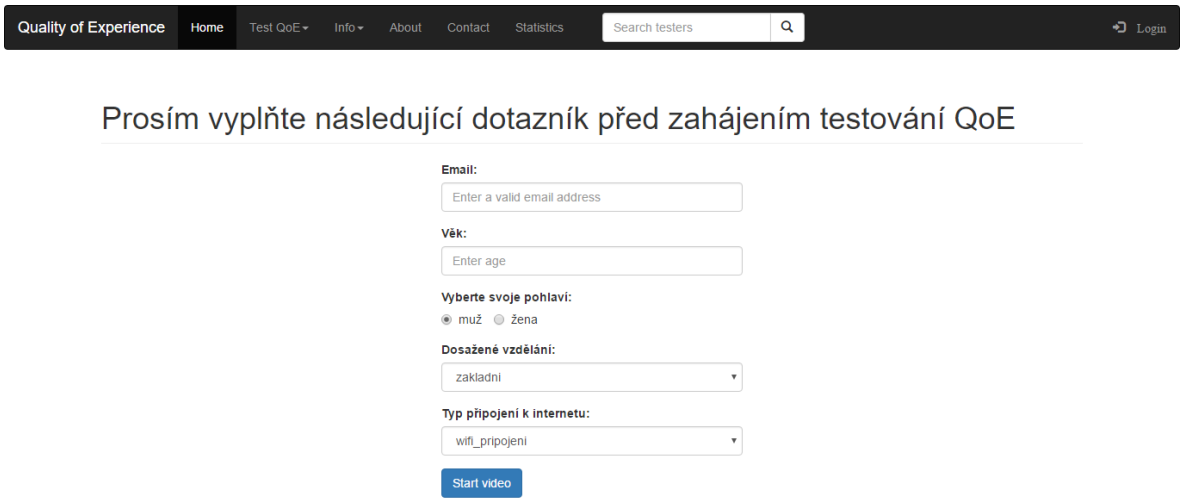


Obr. 3.28: Ukázka HTTP POST pomocí REST API.

3.3.9 Dotazník

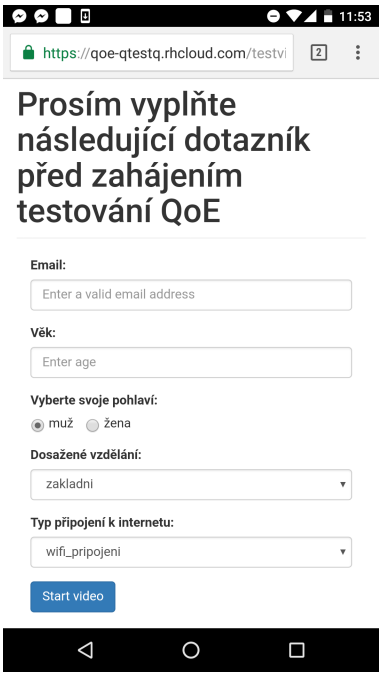
Uživatelé, kteří se zúčastní testování QoE, musí nejdříve vyplnit dotazník, kde zadají svůj email, věk, pohlaví, dosažené vzdělání a typ připojení k internetu. Tyto údaje mohou sloužit později pro analýzu výsledných dat.

Dotazníky jsou pro srovnání vidět na obrázcích 3.29 – 3.31. Jde o zobrazení dotazníku přímo ve webové aplikaci, na mobilním zařízení přistupujícímu k webové aplikaci a přímo v mobilní aplikaci.

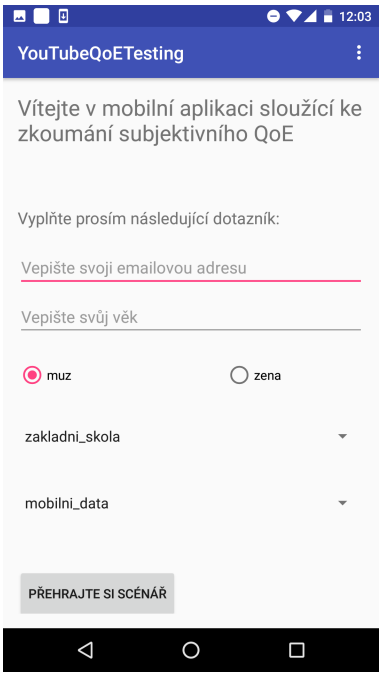


Obr. 3.29: Dotazník na webu.

Obrázek 3.30 ukazuje dotazník z webové aplikace spuštěný přes prohlížeč Chrome na mobilním zařízení LG Nexus 5X.



Obr. 3.30: Dotazník na webu zobrazený přes prohlížeč Chrome na mobilním zařízení LG Nexus 5X.



Obr. 3.31: Dotazník přímo v mobilní aplikaci.

Obrázek 3.31 demonstruje dotazník přímo v mobilní aplikaci.

3.3.10 Video

Pro zobrazení videa na serveru byl využit tag <video>. Byl upřednostněn před tagem <iframe>, který je více svázán s YouTube videi a byl by vhodnější pro alternativy 1 a 2 ze všech možných, které byly nastíněny v podsekci 3.3.5.

Pro video tag existuje poměrně velké množství různých frameworků, které poskytují zjednodušení manipulace s videem. Vhodné frameworky pro efektivní manipulaci s videem jsou:

- Video.js,
- MediaElement.js,
- Dash.js,
- jPlayer,
- JW player.

Jelikož pro potřeby subjektivního testování QoE, kde chceme změnit kvalitu v určité časy není příliš velká potřeba většiny funkcí zmíněných frameworků, bylo upřednostněno použití čistého javascriptu v kombinaci s jQuery pro zobrazování

loading gifu. Kdyby se požadavky na manipulaci s videem rozšiřovaly, ze zmíněných frameworků by byl nejspíše použit framework Video.js, který se jeví jako poměrně snadno použitelný, s dobrou dokumentací a vhodnou škálou metod pro akce nad videem.

3.3.11 Problematika přepnutí kvality

Pro video, které má více typů kvalit (1080p, 720p, 480p, ...), existuje video soubor pro každý typ kvality. Pro přepnutí kvality daného videa je tedy nutné zobrazit video, které má jako svůj zdroj soubor s danou kvalitou videa. Jako nejjednodušší řešení přepnutí kvality se tedy jeví změnit atribut `source` u videa na nový zdrojový soubor s danou kvalitou videa. Nevýhodou tohoto přístupu na méně výkonných serverech je, že se zde zobrazí během změny zdrojového videa přebliknutí úvodní obrazovky nového videa. To je způsobeno tím, že tento nový zdroj videa se musí znovu nahrát, čímž video začíná „od nuly“ a je nutné ho posunout na původní čas videa. I přes zmíněný defekt byl z důvodu jednoduchosti tento přístup použit pro přepnutí kvality.

Mezi další možné přístupy přepnutí kvality u videa patří:

- 2 video tagy, z čehož jedno video je přehrávané a druhé je použito pouze jako překryv během změny kvality. V případě nutnosti překryvu je pro video měnící zdroj nastaven styl `display:none`, čímž je video skryto pro uživatele a nahrazeno novým videem, u kterého se změní styl z `display:none` na `display:block`. V tomto okamžiku je přepnut zdroj videa a následně zase skryto překrývající video a zobrazeno původní video se změněnou kvalitou.

- Podobný princip by se též dal aplikovat tak, že dle počtu kvalit daného videa by stránka obsahovala patřičný počet video tagů s určitou kvalitou videa, kde by se přehrávalo pouze jedno z nich a u ostatních by byl nastaven opět styl `display:none`, tím by se docílilo toho, že by se nikdy nepřepínal zdroj videa, ale pouze by se nastavoval `display:block` u videa s požadovanou kvalitou. Toto řešení je poměrně efektivní, co se týče uživatelského vnímání přebliknutí, nicméně je velmi nevhodné z hlediska zatížení sítě.

- Další možností je použití tzv. thumbnails[4] (thumbnails využívá například YouTube, kde můžeme získat například základní obrázek pro video podle vzoru `http://img.youtube.com/vi/[identifikátor videa]/default.jpg`), kde by pro každé video byla vytvořena série obrázků, která by překryla úvodní obrazovku během změny zdroje videa.

[4]Thumbnail představuje termín používaný zejména u grafiků a fotografů, jde o malou prezentaci většího obrazu, která slouží pro snadný a rychlý náhled na skupinu větších obrazů.

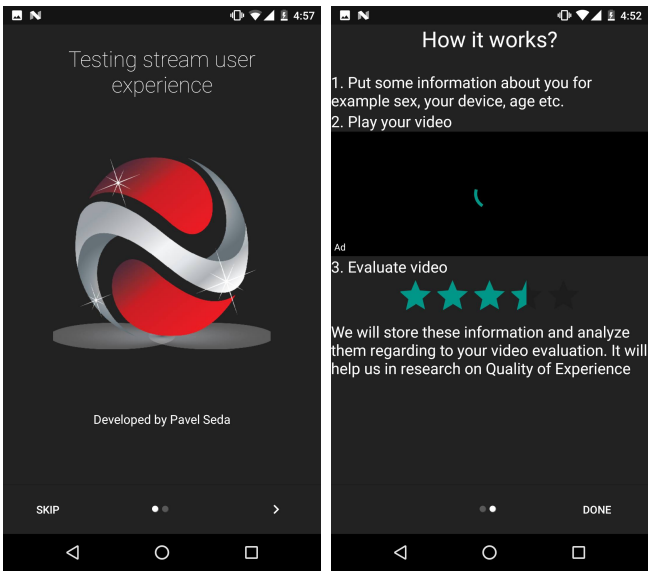
3.3.12 Vytvoření ikony aplikace

Na vytvoření ikon pro mobilní aplikace existuje v dnešní době velké množství online generátorů, které umožní zobrazení ikonky v různých velikostech, přizpůsobí tvar okraje, nastaví pozadí, efekty, tvar atd. Pro vytvoření ikony aplikace byl použit generátor, který je dostupný na webové stránce `https://romannurik.github.io/AndroidAssetStudio/icons-launcher.html`. Ikona aplikace je vidět na obrázku 3.32.

Obr. 3.32: Vytvořená ikona aplikace.

3.3.13 Možnosti aplikace

Mezi další možnosti patří spuštění intra, které popisuje, jak aplikace funguje a k čemu slouží, viz obrázek 3.33.

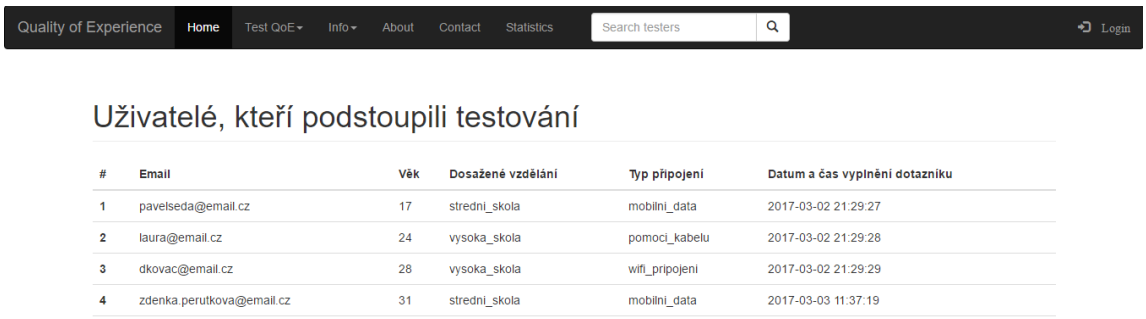


Obr. 3.33: Spuštění prezentace jako intro k aplikaci.

V aplikaci kromě subjektivního testování je plánována také možnost vyhledávání a spouštění videí na samotném YouTube. Tuto ideu je vidět v diagramu tříd na obrázku 3.4, kde třídy `Search, Player, VideoItem, YouTubeConnector` by měly zajišťovat zmíněnou funkcionalitu. Uživatel přes menu aplikace stiskne tlačítko **Search**

YouTube a spustí se nová Android aktivita, ve které uživatel zadá do textového pole vstupní data, podle kterých chce vyhledávat a najde požadovaná videa.

Ve webové aplikaci je možné vyhledat seznam uživatelů, kteří se zúčastnili testování. Vyhledávání je možné podle emailu, věku, dosaženého vzdělání nebo podle typu připojení. Pokud se ve vyhledávacím poli nezadá žádný vyhledávací řetězec, zobrazí se všichni uživatelé. Příklad takového vyhledání viz obrázek 3.34.



Obr. 3.34: Příklad vyhledání uživatelů, kteří se zúčastnili testování.

4 Závěr

V diplomové práci byla zkoumána problematika měření subjektivní kvality zážitku u streamovaných videí pomocí mobilní aplikace využívající YouTube Android API. V úvodní části je popsána problematika streamingu, základní rozdělení QoE, charakteristiky QoE a rozdíly proti QoS. Detailněji je popsáno subjektivní QoE, metody jeho měření a korelace získaných výsledků.

V další části této práce je navrhnuta analýza k vytvářené aplikaci, kde je nastíněno, jak by měla aplikace fungovat, v jakých stavech by se měl účastník testování pro měření subjektivního QoE nacházet, jaké třídy by měly být v aplikaci zahrnuty a návrh databáze, ve které se budou ukládat data z měření aj. Stručně jsou popsány technologie, které byly použity pro vývoj aplikace, a v neposlední řadě je zde znázorněno, jakým způsobem se aplikace vytvářela, jak se integruje YouTube Android API do aplikace a co bylo potřebné nastavit, aby aplikace byla uzpůsobena subjektivnímu měření kvality zážitku. V této části bylo zjištěno, že YouTube Android API se nedá pro měření subjektivního QoE použít, protože nedovoluje z programového kódu měnit rozlišení ani typ streamování. Toto je považováno za standardní vývoj aplikace, kde nejsou vždy zpočátku zcela zřetelná všechna omezení, která daný přístup přinese, a proto je nutné přijít s alternativními řešeními. V závislosti na těchto zjištěních byla navržena tři řešení, která jsou detailněji popsána v sekci 3.3.5, z čehož jedno z navržených řešení bylo implementováno.

Zvolená alternativa vyžadovala implementaci serverové části aplikace v Java EE, kde jsou přehrávaná videa zobrazována v mobilní aplikaci. Tento přístup má výhodu zejména v plné kontrole nad přehrávanými videi a poměrně jednoduché parametrizaci videí javascriptem na serveru. Vzhledem k nutnosti implementace serverové části byla při této příležitosti implementována možnost testování subjektivního QoE i přímo na serveru nad rámec zadání práce. Nevýhoda zvoleného přístupu je v centralizovaném pojetí, kde jsou všechny akce s videem vykonávány na serveru a při větším počtu uživatelů by byl v takovém případě nutný stabilní a výkonný server, který zvládne zpracovat větší množství požadavků. Z tohoto důvodu by bylo vhodnější v případě umožnění změny rozlišení z programového kódu při integraci YouTube Android API využít spíše tento přístup. Proto je v práci ponechána část ohledně integrace tohoto API do mobilní aplikace, i když není ve své podstatě plně využita.

Pro přenos dat mezi mobilním zařízením a vytvořeným serverem byla implementována REST API, která je přístupná pouze autentizovaným uživatelům, kteří mají uložená hesla v databázi algoritmem AES v režimu CBC. Toto API vytváří další abstraktní vrstvu, která může sloužit pro další aplikace, které by četly, modifikovaly či ukládaly data do společné databáze bez nutnosti přímého přístupu k databázi,

pouze se znalostí dokumentace k vytvořené REST API.

Implementovaný systém slouží tedy pro hodnocení kvality zážitku u streamovaných videí, kde se dá simulovat adaptivní streamování pomocí přehrávání scénářů nad danými videi. Tyto scénáře se dají jednoduše upravovat podle potřeby přímo v databázi popřípadě přes REST API. Velkou výhodou tedy je, že se nemusí v programovém kódu nic měnit a jsme schopni pouze změnou parametrů v databázi docílit velkého počtu různě nadefinovaných scénářů pro simulaci adaptivního streamování.

Literatura

[1] MACK, Steve. *Streaming media bible*. New York, NY: Hungry Minds, c2002. ISBN 0764536508.

[2] Netflix. *Netflix* [online]. [cit. 2017-03-17]. Dostupné z: https://www.netflix.com/

[3] Amazon. *Amazon* [online]. [cit. 2017-03-17]. Dostupné z: https://www.amazon.com/gp/video/primesignup/ref=sturl_primeinstantvideo

[4] YouTube. *YouTube* [online]. [cit. 2017-03-17]. Dostupné z: https://www.youtube.com/

[5] HOSSFELD Tobias, Raimund SCHATZ, Michael SEUFERT, Matthias HIRTH, Thomas ZINNER, Phuoc TRAN-GIA. *Quantification of YouTube QoE via Crowdsourcing.* IEEE International Workshop on Multimedia Quality of Experience – Modeling, Evaluation, and Directions (MQoE 2011), Dana Point, CA, USA, December 2011.

[6] WAMSER, F., D. HOCK, M. SEUFERT, B. STAEHLE, R. PRIES a P. TRAN-GIA. Using buffered playtime for QoE-oriented resource management of YouTube video streaming. In: *Transactions on Emerging Telecommunications Technologies.* 2013, 24(3), s. 288-302. DOI: 10.1002/ett.2636. ISSN 21613915. Dostupné také z: http://doi.wiley.com/10.1002/ett.2636

[7] OZER, Jan. *Streaming Vs. Progressive Download Vs. Adaptive Streaming* [online]. [cit. 2016-10-21]. Dostupné z: http://www.onlinevideo.net/2011/05/streaming-vs-progressive-download-vs-adaptive-streaming/

[8] STOCKHAMMER, Thomas. Dynamic adaptive streaming over HTTP. In: *Proceedings of the second annual ACM conference on Multimedia systems – MMSys '11* [online]. New York, New York, USA: ACM Press, 2011, s. 133- [cit. 2016-10-16]. DOI: 10.1145/1943552.1943572. ISBN 9781450305181. Dostupné z: http://portal.acm.org/citation.cfm?doid=1943552.1943572

[9] OZER, Jan. *What is Adaptive Streaming?* [online]. [cit. 2016-10-02]. Dostupné z: http://www.streamingmedia.com/Articles/Editorial/What-Is-.../What-is-Adaptive-Streaming-75195.aspx

[10] SCHATZ, Raimund, Tobias HOSSFELD a Pedro CASAS. Passive YouTube QoE Monitoring for ISPs. In: *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing* [online]. IEEE, 2012, s.

358-364 [cit. 2016-10-12]. DOI: 10.1109/IMIS.2012.12. ISBN 978-1-4673-1328-5. Dostupné z: http://ieeexplore.ieee.org/document/6296879/

[11] ORGANIZERS, ETRI a IEEE ComSoc SPONSORS. *The 12th International Conference on Advanced Communication Technology lCT for Green Growth and Sustainable Development, Phoenix Park, Korea, Feb. 7-10, 2010, proceedings: ICACT 2010*. Piscataway, N.J: IEEE, 2010. ISBN 9788955191462.

[12] WU, Hong Ren. QoE Subjective and Objective Evaluation Methodologies. *Multimedia Quality of Experience (QoE)* [online]. Chichester, UK: John Wiley & Sons, Ltd, 2015, s. 123 [cit. 2016-10-28]. DOI: 10.1002/9781118736135.ch6. ISBN 9781118736135. Dostupné z: http://doi.wiley.com/10.1002/9781118736135.ch6

[13] MENKOVSKI, Vlado. Objective QoE Models. *Computational Inference and Control of Quality in Multimedia Services* [online]. 2015, s. 15 [cit. 2016-10-10]. DOI: 10.1007/978-3-319-24792-2_2. ISBN 978-3-319-24792-2. Dostupné z: http://link.springer.com/10.1007/978-3-319-24792-2_2

[14] WINKLER, S. a P. MOHANDAS. The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics. *IEEE Transactions on Broadcasting* [online]. 2008, 54(3), 660-668 [cit. 2016-10-10]. DOI: 10.1109/TBC.2008.2000733. ISSN 0018-9316. Dostupné z: http://ieeexplore.ieee.org/document/4550731/

[15] PIAMRAT, K., VIHO, C., BONNIN, J.M., KSENTINI, A.: *Quality of Experience Measurements for Video Streaming over Wireless Networks*. (Apr 2009) 1184–1189

[16] KONEČNÝ, Jakub. *Optimalizace univerzitní bezdrátové sítě pro provoz hlasových slu- žeb*. Brno, 2015. Diplomová práce. Mendelova univerzita v Brně. Vedoucí práce Ing. Petr Zach.

[17] ITU-T, *Methods for subjective determination of transmission quality*, Recommendation ITU-T P.800., September 1996

[18] CHOE, Ji-Hwan, Tae-Uk JEONG, Hyun-Soo CHOI, Eun-Jae LEE, Sang-Wook LEE a Chul-Hee LEE. Comparison of subjective video quality assessment methods for multimedia applications. *Journal of Broadcast Engineering* [online]. 2007, 12(2), 177-184 [cit. 2016-10-28]. DOI: 10.5909/JBE.2007.12.2.177. ISSN 1226-7953. Dostupné z: http://koreascience.or.kr/journal/view.jsp?kj=BSGHC3&py=2007&vnc=v12n2&sp=177

[19] ITU-T, *Subjective video quality assessment methods for multimedia applications*, Recommendation ITU-T P.910, April 2008

[20] ITU-R, *Methodology for the subjective assessment of video quality in multimedia applications*, Recommendation ITU-R BT.1788, 2007

[21] GLEN, Stephanie. *Pearson Correlation: Definition and Easy Steps for Use* [online]. 2016 [cit. 2016-10-29]. Dostupné z: http://www.statisticshowto.com/what-is-the-pearson-correlation-coefficient/

[22] BOLBOACA, Sorana-Daniela a Lorentz JÄNTSCHI. *Pearson versus Spearman, Kendall's Tau Correlation Analysis on Structure-Activity Relationships of Biologic Active Compounds* [online]. [cit. 2016-10-29]. Dostupné z: http://ljs.academicdirect.org/A09/179_200.htm

[23] HIRTH, Matthias, Tobias HOSSFELD, Phuoc TRAN-GIA. *Anatomy of a Crowdsourcing Platform – Using the Example of Microworkers.com.* Workshop on Future Internet and Next Generation Networks (FINGNet), Seoul, Korea, June 2011. Also available as technical report *Human Cloud as Emerging Internet Application – Anatomy of the Microworkers Crowdsourcing Platform.* Technical Report 478, January 2011.

[24] BIERSACK, Ernst et al.: *Data traffic monitoring and analysis: from measurement, classification, and anomaly detection to quality of experience.* 1st ed. New York: Springer, 2013, pp. 264-301, ISBN 3642367836.

[25] TONY BUZAN WITH BARRY BUZAN. *The mind map book: how to use radiant thinking to maximize your brain's untapped potential.* New York: Plume, 1996. ISBN 9780452273221.

[26] ARLOW, Jim a Ila NEUSTADT. *UML 2.0 and the unified process: practical object-oriented analysis and design.* 2nd ed. Boston: Addison-Wesley, c2005. Addison-Wesley Object Technology Series. ISBN 978-0-321-32127-5.

[27] Google. *Android Studio.* [online] [cit. 2016-11-24]. Dostupné také z: https://developer.android.com/studio/index.html

[28] CHACON, Scott. *Pro Git.* Praha: CZ.NIC, c2009. CZ.NIC. ISBN 978-80-904248-1-4.

[29] PEARCE, Shawn. *GitSvnComparison* [online]. 2013. vyd. [cit. 2016-09-30]. Dostupné z: https://git.wiki.kernel.org/index.php/GitSvnComparison.

[30] GRIFFITHS, Dawn a David GRIFFITHS. *Head first Android development.* Sebastopol: O'Reilly, 2015. Head first series. ISBN 1449362184.

[31] SIERRA, Kathy. a Bert. BATES. *Head first Java.* 2nd ed. Sebastopol, CA: O'Reilly, 2005. ISBN 05-960-0920-8.

[32] FINEGAN, Edward. a Robert. LIGUORI. *OCA Java se 7 programmer I study guide (exam 1Z0-803).* New York: McGraw-Hill, 2013. ISBN 0071789421.

[33] GUPTA, Mala. *OCP Java se 7 programmer II certification guide: prepare for the 1ZO-804 exam.* Shelter Island, NY: Mannning, 2015. ISBN 9781617291487.

[34] BERGLUND, Tim a Matthew MCCULLOUGH. *Building and Testing with Gradle* [online]. Sebastopol, CA: O'Reilly Media, Inc., 2011 [cit. 2016-09-30]. Dostupné z: http://berddk.ru/media/doc/2013/11/27/Building_and_testing_with_Gradle.pdf

[35] Oracle, *What is MySQL?* [online]. [cit. 2016-11-13]. Dostupné z: http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

[36] ELLIOTT, James, Ryan FOWLER a Tim O'BRIEN. *Harnessing Hibernate: revue générale des publications françaises et étrangères* [online]. Sebastopol: O'Reilly, 2008 [cit. 2017-03-03]. ISBN 978-0-596-51772-4.

[37] FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures.* Irvine, 2000. Disertace. University of California.

[38] ISO/IEC 14977. *Cam* [online]. [cit. 2017-03-17]. Dostupné z: http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf

[39] JSONView. *Google* [online]. [cit. 2017-03-17]. Dostupné z: https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc

[40] YouTube, *YouTube Developer Documentation* [online]. listopad 16, 2015 [cit. 2016-11-6]. Dostupné také z: https://developers.google.com/youtube/documentation/

[41] YouTube, *YouTube Android Player API* [online]. květen 23, 2015 [cit. 2016-10-4]. Dostupné také z: https://developers.google.com/youtube/android/player/

[42] VUJOVIC, Filip. (2015, prosinec). *How to Integrate YouTube API In Android Application.* Dostupné také z: https://www.youtube.com/watch?v=a4NT5iBFuZs

[43] YouTubePlayer. *Android Player API* [online]. 2015 [cit. 2017-04-06]. Dostupné z: https://developers.google.com/youtube/android/player/reference/com/google/android/youtube/player/YouTubePlayer

[44] Restlet Client. *Restlet Client – DHC* [online]. 2017 [cit. 2017-04-06]. Dostupné z: https://chrome.google.com/webstore/detail/restlet-client-dhc/aejoelaoggembcahagimdiliamlcdmfm

[45] GitHub. *GitHub* [online]. [cit. 2017-03-17]. Dostupné z: https://github.com/SedaQ/QoE-SS/tree/master/qoe-rest

Seznam symbolů, veličin a zkratek

ACR	Absolute Category Rating
ACR-HR	Absolute Category Rating with hidden reference
API	Application Programming Interface
Authenticator	Autentizátor
AVC	Advanced Video Coding
CSS	Cascading Style Sheets
CPU	Central Processing Unit
CIF	Common Intermediate Format
EBNF	Extended Backus–Naur form
ECC	Elliptic Curve Cryptography
ERD	Entitně relační diagram
GNU	General Public License
DB	Databáze
DCR	Degradation Category Rating
DSCQS	The Double-Stimulus Continuous Quality-Scale method
DSIS	Double Stimulus Impairment Scale
DSL	Domain Specific Language
DTO	Data Transfer Object
DV	Differential Viewer
HD	High Definition
HLS	HTTP Live Streaming
HTTP	Hypertext Transfer Protocol
HTML5	Hyper Text Markup Language 5
IDE	Integrated Development Environment

IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Organization for Standardization
Java EE	Java Enterprise Edition
JSP	Java Server Pages
JDK	Java Development Kit
JPA	Java Persistence API
JSON	Javascript Object Notation
MOS	Mean Opinion Score
MPQM	Moving Picture Quality Metric
MSE	Mean Square Error
MTC	Maximal Threshold Correlation
MySQL	My Structured Query Language
NQM	Noise Quality Measure
ORM	Object Relation Mapping
PC	Pair comparison method
PSNR	Peak Signal to Noise Ratio
PVS	kódovaná sekvence
QCIF	Quarter Common Intermediate Format
QoE	Quality of Experience
QoS	Quality of Service
REST	Representational State Transfer
REF	skrytá reference
SDK	Software Development Kit
SHA1	Secure Hash Algorithm 1

SIF	Source Input Format
SQL	Structured Query Language
SSIM	Structural Similarity Index
SVC	Scalable Video Coding
SVM	Subversion
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
User	Uživatel
VoIP	Voice over Internet Protocol
VQM	Video Quality Metric
XML	Extensible Markup Language
XML-RPC	Extensible Markup Language Remote Procedure

Seznam příloh

A Zdrojové kódy

Zdrojové kódy webového Java EE projektu a mobilní aplikace jsou dostupné na serveru GitHub. Přesné adresy jsou uvedeny v A.1 a A.2.

A.1 Java EE projekt

Náhled implementace Java EE aplikace je k dispozici na stránce: https://github.com/SedaQ/QoE-SS v projektech: `qoe-api`, `qoe-persistence`, `qoe-rest`, `qoe-service`, `qoe-spring-mvc`.

A.2 Mobilní aplikace

Zdrojové kódy mobilní aplikace lze najít na stránce: https://github.com/SedaQ/QoE-CS, ve složce `app` jsou všechny třídy projektu.

A.3 Spuštění projektu

Pro spuštění Java EE projektu stačí mít instalovaný Maven, Git a XAMPP v systému.

- Pro instalaci Mavenu může sloužit oficiální webová stránka: http://maven.apache.org/.
- Pro instalaci Gitu do systému stačí na linuxových systémech zadat příkaz `yum install git` do příkazové řádky a na windows systémech lze postupovat podle následující stránky https://git-scm.com/downloads.
- XAMPP lze nainstalovat prostřednictvím oficiální stránky: https://www.apachefriends.org/index.html.

Pro jednoduché spuštění projektu je nutné mít instalované programy Git, Maven a XAMPP. Spustit program `xampp-control` a v něm moduly MySQL a Apache. Poté, pokud tak není nastaveno, je nutné upravit soubor `persistence.xml` v podprojektu `qoe-persistence`, kde se musí změnit property se jménem `hibernate.hbm2ddl.auto` na `value="create"`. Následně spustit příkazovou řádku a zadat do ní příkaz `git clone https://github.com/SedaQ/QoE-SS.git`, čímž se stáhne celý projekt do aktuální složky, a poté v kořenové složce tohoto projektu, což je složka `qoe-parent`, spustit příkazovou řádku a zadat příkaz `mvn clean install` a následně příkaz `mvn`. Po sérii těchto příkazů je projekt přístupný na adrese: `http://localhost:8080/`.

B MySQL databáze

B.1 MySQL dotazy k vytvoření databáze

```
-- -----------------------------------------------------
-- Schema qoe
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `qoe` DEFAULT CHARACTER SET latin1;
USE `qoe`;
-- -----------------------------------------------------
-- Table `qoe`.`video`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `qoe`.`video` (
  `id_video` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id_video`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
-- -----------------------------------------------------
-- Table `qoe`.`user`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `qoe`.`user` (
  `id_user` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `email` VARCHAR(255) NOT NULL,
  `password` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id_user`),
  UNIQUE INDEX `UK_ob8kqyqqgmefl0aco34akdtpe` (`email` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
-- -----------------------------------------------------
-- Table `qoe`.`questionnaire`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `qoe`.`questionnaire` (
  `id_questionnaire` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `age` VARCHAR(255) NOT NULL,
  `date` DATETIME NOT NULL,
  `email` VARCHAR(255) NOT NULL,
  `gender` VARCHAR(255) NOT NULL,
  `school` VARCHAR(255) NOT NULL,
  `user_connection` VARCHAR(255) NOT NULL,
```

```
  `user_id_user` BIGINT(20) NULL DEFAULT NULL,
  PRIMARY KEY (`id_questionnaire`, `user_id_user`),
  INDEX `FK_kqe64trwck8brh7xb0ig732s2` (`user_id_user` ASC),
  CONSTRAINT `FK_kqe64trwck8brh7xb0ig732s2`
    FOREIGN KEY (`user_id_user`)
    REFERENCES `qoe`.`user` (`id_user`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
-- -----------------------------------------------------
-- Table `qoe`.`scenario`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `qoe`.`scenario` (
  `id_scenarion` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `scenario` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id_scenarion`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
-- -----------------------------------------------------
-- Table `qoe`.`mos`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `qoe`.`mos` (
  `id_mos` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `mos_value` VARCHAR(255) NOT NULL,
  `questionnaire_id_questionnaire` BIGINT(20) NOT NULL,
  `scenario_id_scenarion` BIGINT(20) NOT NULL,
  `video_id_video` BIGINT(20) NOT NULL,
  PRIMARY KEY (`id_mos`, `questionnaire_id_questionnaire`,
     `scenario_id_scenarion`, `video_id_video`),
  UNIQUE INDEX `UK_b3fdd3d04hx4embleojfkusy1`
     (`questionnaire_id_questionnaire` ASC),
  UNIQUE INDEX `UK_m7ex95hp7xyf7n6hkvimm0ijt`
     (`scenario_id_scenarion` ASC),
  INDEX `FK_nf3opjabvjdbcqfjvwn8ftl2p` (`video_id_video` ASC),
  CONSTRAINT `FK_nf3opjabvjdbcqfjvwn8ftl2p`
    FOREIGN KEY (`video_id_video`)
    REFERENCES `qoe`.`video` (`id_video`),
  CONSTRAINT `FK_b3fdd3d04hx4embleojfkusy1`
    FOREIGN KEY (`questionnaire_id_questionnaire`)
    REFERENCES `qoe`.`questionnaire` (`id_questionnaire`),
  CONSTRAINT `FK_m7ex95hp7xyf7n6hkvimm0ijt`
    FOREIGN KEY (`scenario_id_scenarion`)
```

```
    REFERENCES `qoe`.`scenario` (`id_scenarion`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
-- -----------------------------------------------------
-- Table `qoe`.`scenario_parameters`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `qoe`.`scenario_parameters` (
  `id_scenarioparameters` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `length` BIGINT(20) NOT NULL,
  `time` BIGINT(20) NOT NULL,
  `video_quality` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id_scenarioparameters`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
-- -----------------------------------------------------
-- Table `qoe`.`scenario_parameters_scenario`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `qoe`.`scenario_parameters_scenario` (
  `scenarioparameters_id_scenarioparameters` BIGINT(20) NOT NULL,
  `scenario_id_scenarion` BIGINT(20) NOT NULL,
  PRIMARY KEY (`scenarioparameters_id_scenarioparameters`,
     `scenario_id_scenarion`),
  INDEX `FK_8f85rqx2hhrya6fua4eb0kb0r` (`scenario_id_scenarion` ASC),
  CONSTRAINT `FK_7cojo2dvye8rc0fkbusiuht71`
    FOREIGN KEY (`scenarioparameters_id_scenarioparameters`)
    REFERENCES `qoe`.`scenario_parameters` (`id_scenarioparameters`),
  CONSTRAINT `FK_8f85rqx2hhrya6fua4eb0kb0r`
    FOREIGN KEY (`scenario_id_scenarion`)
    REFERENCES `qoe`.`scenario` (`id_scenarion`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
-- -----------------------------------------------------
-- Table `qoe`.`scenario_video`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `qoe`.`scenario_video` (
  `scenario_id_scenarion` BIGINT(20) NOT NULL,
  `video_id_video` BIGINT(20) NOT NULL,
  PRIMARY KEY (`scenario_id_scenarion`, `video_id_video`),
  INDEX `FK_sgpxr0cuaojmah6sw3iasytst` (`video_id_video` ASC),
  CONSTRAINT `FK_is4i58t2higcljhq0npahj2mi`
    FOREIGN KEY (`scenario_id_scenarion`)
```

```
    REFERENCES `qoe`.`scenario` (`id_scenarion`),
  CONSTRAINT `FK_sgpxr0cuaojmah6sw3iasytst`
    FOREIGN KEY (`video_id_video`)
    REFERENCES `qoe`.`video` (`id_video`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
-- -----------------------------------------------------
-- Table `qoe`.`user_role`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `qoe`.`user_role` (
  `user_id_user` BIGINT(20) NOT NULL,
  `role` VARCHAR(255) NULL DEFAULT NULL,
  INDEX `FK_2584ngkwmqwo82e2b2rs6enx9` (`user_id_user` ASC),
  PRIMARY KEY (`user_id_user`),
  CONSTRAINT `FK_2584ngkwmqwo82e2b2rs6enx9`
    FOREIGN KEY (`user_id_user`)
    REFERENCES `qoe`.`user` (`id_user`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
-- -----------------------------------------------------
-- Table `qoe`.`video_source`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `qoe`.`video_source` (
  `video_id_video` BIGINT(20) NOT NULL,
  `video_src` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`video_id_video`, `video_src`),
  CONSTRAINT `FK_efwm9bhmskltjmlsnhjo8n8bb`
    FOREIGN KEY (`video_id_video`)
    REFERENCES `qoe`.`video` (`id_video`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
```

C Metriky kódu

Počet tříd, rozhraní a řádků v projektech

Tab. C.1: Webová část aplikace implementovaná v Java EE. Rozděleno dle dílčích vrstev projektu. Do statistik se počítají pouze .java soubory, ostatní (např. konfigurace v .xml a soubory pro „view“ .jsp) se do statistiky nezapočítávaly.

projekt	**počet tříd**	**počet rozhraní**	**počet řádků kódu**
qoe-api	23	6	1258
qoe-persistence	21	6	1167
qoe-rest	29	0	966
qoe-service	43	7	2564
qoe-spring-mvc	40	0	1611
celkem	156	19	7566

Tab. C.2: Mobilní aplikace. Do výsledku se nezapočítávaly předdefinované třídy typu R.java, s kterými by aplikace měla 11187 řádků.

projekt	**počet tříd**	**počet rozhraní**	**počet řádků kódu**
app	16	0	1381

Tab. C.3: Celkový součet počtu tříd, rozhraní a řádků kódu v mobilní a webové aplikaci.

	počet tříd	**počet rozhraní**	**počet řádků kódu**
celkový součet	172	19	8947