VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÝ NÁSTROJ PRO ANOTACI OBRAZOVÝCH DAT

WEB-BASED IMAGE ANNOTATION TOOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. TOMÁŠ VOSTŘEJŽ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAKUB ŠPAŇHEL**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Vostřejž Tomáš, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Vývoj aplikací
Název: **Webový nástroj pro anotaci obrazových dat**
Web-Based Image Annotation Tool
Kategorie: Zpracování obrazu
Zadání:

1. Seznamte se s principy vývoje webového nástroje za použití dostupných technologií pro vývoj interaktivních webových aplikací.
2. Navrhněte webový nástroj pro anotaci obrazových dat, zejména pro vkládání anotací typu - bod, přímka, tah, obdélník, polygon.
3. Zaměřte se na modularitu a lehkou rozšiřitelnost tohoto nástroje pro různé typy anotací.
4. Implementujte funkce pro manipulaci s obrazem (zoom, posunutí) a funkce pro transformaci obrazu (jas, kontrast, atd.).
5. Dále navrhněte a implementujte rozšíření pro anotaci videa s využitím trajektorie pohybu objektu.
6. Iterativně vyvíjejte a testujte vytvořený nástroj na jeho uživatelích.
7. Vytvořte plakát a video prezentující vaší práci, její cíle a výsledky.

Literatura:

- Tidwell et al.: Designing Interfaces: Patterns for Effective Interaction Design, O'Reilly, 2020.
- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516.
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016.
- Dále dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních tří bodů zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Špaňhel Jakub, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Tato práce se zabývá vývojem webové aplikace pro anotaci obrazových dat. Popisuje návrh a implementaci klientské a serverové části nástroje, který pracuje s video soubory. Podporuje sledování trajektorie objektu pomocí interpolace. Implementován je v jazyce JavaScript s využitím platformy Angular a knihovny Express. Uživateli umožňuje vytvářet anotace typu bod, přímka, tah, obdélník a polygon. Anotace se vytváří na základě anotačních šablon, které nástroj organizuje do skupin. Datasety mají jednu, nebo více anotačních skupin a uživatel má možnost je mezi datasety přenášet a znovu použít pomocí osobní knihovny. Nástroj exportuje výsledné anotace ve formátu JSON.

Abstract

This thesis deals with the development of a web application for image data annotation. Describes the design and implementation of the client and server side of a tool that works with video files. Supports object tracking using interpolation. It is implemented in JavaScript using the Angular platform and the Express library. Allows the user to create point, line, stroke, rectangle, and polygon annotations. Annotations are created based on annotation templates that the tool organizes into groups. Datasets have one or more annotation groups and the user has the option to transfer and reuse them between datasets using a personal library. The tool exports the resulting annotations in JSON format.

Klíčová slova

web, webová aplikace, video, obrazová data, anotace, Angular, REST API, Node.js, MongoDB, Mongoose, Express, Jasmine

Keywords

web, web application, video, image data, annotation, Angular, REST API, Node.js, MongoDB, Mongoose, Express, Jasmine

Citace

VOSTŘEJŽ, Tomáš. *Webový nástroj pro anotaci obrazových dat*. Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jakub Špaňhel

Webový nástroj pro anotaci obrazových dat

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jakuba Špaňhela. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Tomáš Vostřejž
17. května 2022

Poděkování

Děkuji vedoucímu diplomové práce Ing. Jakubovi Špaňhelovi za inspiraci, odbornou pomoc, připomínky a nápady poskytnuté během řešení této práce.

Obsah

Kapitola 1

Úvod

V dnešní době se spousta firem zabývá umělou inteligencí a vzniká velké množství projektů v oblastech autonomního řízení, robotiky, výroby, biologických věd a dalších. Pro učení modelu umělé inteligence je zapotřebí mít k dispozici velkou datovou sadu se správně vytvořenými anotacemi. Tyto anotace mohou být vytvořeny automatizovaně pomocí různých algoritmů nebo za pomoci jiné umělé inteligence, která umí detekovat objekty. Podstatnou část anotací ale vytváří lidé za pomoci dostupných nástrojů.

Cílem této práce je vytvořit webový nástroj k vytváření anotací pro obrazová data, konkrétně video data. Proces anotace videa je zdlouhavý, protože je třeba projít každý snímek a anotace ručně vytvářet a upravovat jejich pozice. Výsledný nástroj by měl nejen umožňovat vytváření anotací, ale taky se zaměřit na vylepšení, která by celý proces urychlila. Může se jednat například o možnost sdílení datové sady s týmem lidí, kteří si mohou práci rozdělit. Nebo využít trajektorii pohybu objektu pro interpolaci pozice anotace mezi snímky videa.

Kapitola 2 obsahuje krátký úvod do problematiky anotace video dat. Dále jsou analyzovány funkční i nefunkční požadavky na systém a sepsány do několika bodů. Na závěr jsou v kapitole uvedeny existující anotační nástroje obrazových dat, rozbor funkcionalit a uživatelské rozhraní, které může posloužit pro inspiraci při návrhu.

Kapitola 3 se zabývá analýzou současných technologií pro tvorbu webových aplikací. Pozornost je věnována populárním knihovnám a platformám pro tvorbu uživatelské rozhraní, ale i serverové a databázové části aplikace. Zmíněny jsou jejich klíčové vlastnosti a části. Součástí je i nástroj pro virtualizaci Docker a popis jeho role při vývoji a nasazení do produkce.

Návrh samotné aplikace je v kapitole 4. Jsou v ní uvedeny návrhy klíčových částí aplikace. Popsány jsou z pohledu používání uživatelem, nabízení funkcionalit a doplněny o návrh grafického uživatelského rozhraní. Dále je popsána architektura celé aplikace, komunikační rozhraní mezi klientskou a serverovou částí aplikace. Nakonec je uvedeno schéma databáze a zvolená databázová technologie.

Následující kapitola 5 popisuje samotnou implementaci nástroje. Uvedena je struktura celého projektu a jednotlivých částí. Popsány jsou klíčové části nástroje jako řešení ukládání dat na serveru a sdílení datasetů s ostatními uživateli. Dále řešení modularity anotací a proces exportování vytvořených anotací. Ukázána je implementace přehrávače videa s rozšiřujícími funkcemi pro manipulaci s obrazem.

Předposlední kapitola 6 se zabývá testováním. První část se týká testování na uživatelích. Popsána je metodika, scénář, úkoly testování a nakonec vyhodnocení testování. Druhá část

se zabývá automatizovaným testováním pomocí jednotkových testů. Poslední kapitola 7 obsahuje shrnutí práce, její hlavní přínosy a možná rozšíření nástroje.

Kapitola 2

Požadavky na nástroj a existující řešení

V kapitole je na úvod krátce popsána problematika anotací video dat. V druhé částí kapitole jsou sepsány funkční i nefunkční požadavky na nástroj, které budou výchozí jak pro návrh, tak i pro možné scénáře testování. Ve třetí části jsou uvedeny existující nástroje umožňující anotaci video dat. Mohou být inspirací jak po stránce funkční, tak z pohledu uživatelského rozhraní.

2.1 Anotace video dat

Jedním z možností učení umělé inteligence je učení s učitelem. Trénovaný model má k dispozici datovou sadu se správně vyznačenými výstupy. Někdo ale musí nejdříve takovou sadu vytvořit a manuálně správné výstupy označit. Tomuto procesu se říká anotace dat. V obrazových sadách se nejčastěji vyznačují objekty pomocí obdélníků, polygonů a dalších tvarů.

Datové sady mohou obsahovat mnoho souborů, v případě video dat taková sada nabývá i s délkou jednotlivých videí. Proces anotace je kvůli tomu zdlouhavý a kvalitní nástroj jej může významně urychlit. Cílem práce je takový nástroj vytvořit. Měl by umožňovat nejen tvorbu datových sad, ale i sdílení s ostatními lidmi, kteří si práci rozdělí.

Dále může umožňovat definování obecných anotací, které tvoří skupiny. Každá datová sada má pak jednu až několik skupin anotací. Tyto skupiny jsou modulární a mohou být znovu použité u dalších sad. Konkrétní anotace umožňuje doplnit o další informace, jako například atributy objektu.

Nástroj by měl mít přehledné a přívětivé uživatelské rozhraní. K ovládání používá primárně myš, hlavně při samotném procesu vytváření ohraničujících tvarů objektů v obraze. Urychlení ale poskytují i klávesové zkratky pro ovládání videa, nebo pro rychlý výběr anotační šablony. Možností, jak usnadnit anotaci dat, je mnoho.

2.2 Požadavky na nástroj

Kapitola obsahuje požadavky na vyvíjenou aplikaci. Formální i neformální požadavky na nástroj jsou sepsány v následujících seznamech podle [13]. Požadavky vycházejí ze zadání práce, obecných požadavků na samotný proces vytváření anotací. Nástroj bude určen pro

práci s video daty s tím, že se bude pracovat přímo s videi a ne s pouhou sekvencí snímků extrahovaných z videa.

Obecné požadavky

- Nástroj musí být webová aplikace. Nebude tak vyžadovat instalaci a k používání postačí webový prohlížeč. Díky tomu bude aplikace přístupná pro velké množství uživatelů a umožní snadné vyzkoušení.
- Nástroj musí mít uživatelsky přívětivé rozhraní. Ovládání musí být intuitivní, přehledné a efektivní, aby urychlilo anotaci datové sady. Primární ovládání myší doplňuje i několik klávesových zkratek.
- Aplikace musí umožnit vytvoření uživatelských účtů. Uživatelé mohou upravovat své osobní údaje.
- Nástroj pracuje přímo s video daty, dále je nezpracovává jako sekvenci samostatných obrázků.
- Aplikace musí umět vytvářet a sdílet datové sady obsahující video data. Uživatel, který sadu vytvoří, se stává vlastníkem a může do projektu přidávat další uživatele. Také musí mít možnost videa přidávat, nebo odebírat.
- Nástroj umí vytvořené anotace exportovat ve vhodném formátu, se kterým lze dále pracovat. Export je umožněn na úrovni videa, nebo celého datasetu.

Požadavky na práci s anotacemi

- Nástroj musí umět vytvářet několik typů anotací, jako je bod, přímka, tah, obdélník a polygon.
- Uživatel musí mít možnost vytvořit skupiny anotací a anotace samotné. Má tak vlastní knihovnu anotací, které může přidělit datovým sadám. Jedná se o obecné anotace, sloužící pouze jako šablona pro anotace konkrétní.
- Konkrétní anotace musí být možné doplnit o dodatečné informace. Také je možné upravit jejich parametry.
- Nástroj musí umět exportovat informace o anotacích ve vhodném formátu, který bude používán pro další zpracování jinými aplikacemi.
- Nástroj umí dopočítat pozici anotace v obrazu v daný čas, pokud není uživatelem předem určena. Použije k tomu informace z již dříve vytvořených anotací.

Požadavky na práci s videi

- Nástroj musí umět přehrávat videa ze serveru a částečně lokálně. Po vybrání lokálních souborů se na podnět uživatele začnou nahrávat na server. Než je ale nahrávání dokončeno, je možné pracovat s lokálními soubory.
- Rychlost přehrávání videa je možné změnit. Usnadní se tak práce s videi, kde jsou rychle se pohybující objekty.

- Nástroj musí mít funkce pro jednoduchou manipulaci s obrazem, konkrétně zoom a posunutí, pro přesnější vytváření ohraničení.
- Nástroj musí mít funkce pro transformaci obrazu, jako je jas, kontrast a další. Může se totiž stát, že některé objekty ve videu budou špatně viditelné.

2.3 Existující anotační nástroje

Kapitola ze zabývá analýzou existujících nástrojů z pohledu uživatelského rozhraní a funkcionalit, které nástroje nabízejí. Zajímavé jsou webové anotační nástroje a nástroje pracující s videem. Na trhu existuje velké množství nástrojů pro anotace obrazových dat, ale většina z nich při práci s videem pouze extrahuje jednotlivé snímky dané frekvencí a dále s nimi pracuje jako se statickým obrazem. V textu jsou popsány pouze dva nástroje a to V7 Darwin a Roboflow.

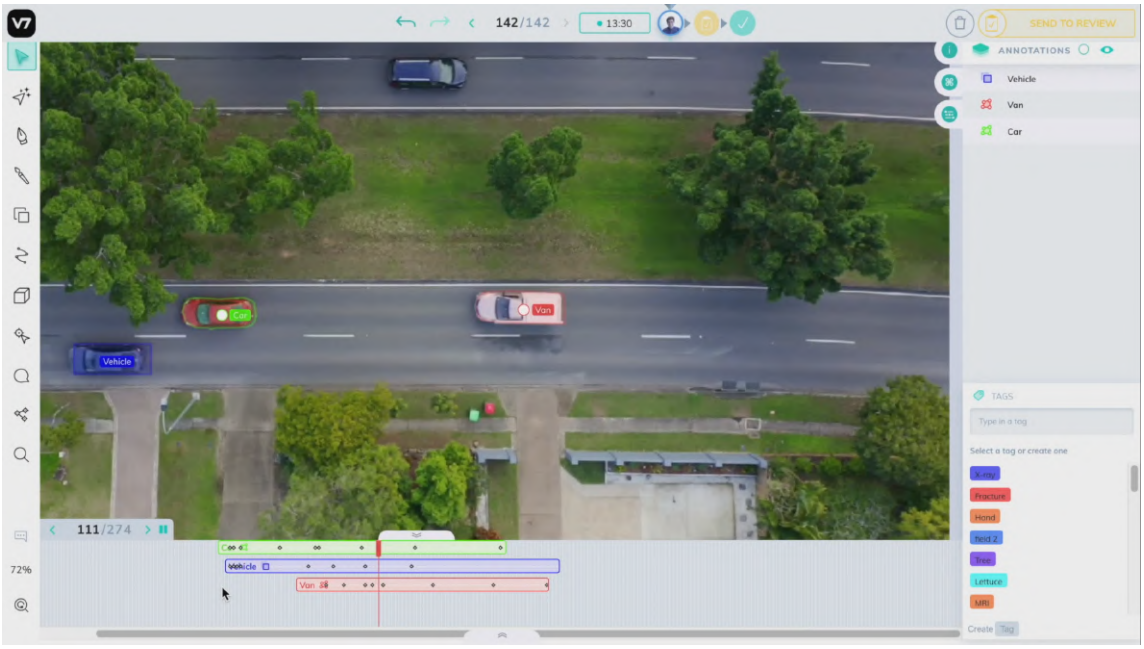
Nástroj V7 Darwin

Nástroj Darwin vytvořila společnost V7 labs, kterou v roce 2018 založil Albert Rizzoli a Simon Edwardson. Jedná se o komerčně používaný nástroj pro anotaci obrazových dat jak statických, tak video dat. V7 se specializuje na oblasti zdravotnictví, výroby, autonomního řízení, sportu, biologických věd a agrotechniky [21].

Darwin je dostupný jako samostatný software vyžadující instalaci. Datasety v něm vytvořené podporují běžně používané typy souborů, jako jsou JPG, PNG, a nebo video formáty MP4, MOV. Při importu videa do datasetu jsou z něj extrahovány jednotlivé snímky a frekvenci si uživatel může sám zvolit. Práce s videem potom probíhá jako sekvence snímků. Na datasetu je možné pracovat v týmu a funguje zde i systém ověření anotací, kdy jeden uživatel anotace vytvoří a další je poté zkontroluje a až poté je proces anotace dokončen.

Darwin také nabízí celou řadu ohraničujících tvarů pro označení objektů ve snímcích. Uživatel po vybrání jednoho z nich může vytvářet anotace tahy myší, nebo využít speciální funkci automatické anotace. Po zvolení obdélníkové oblasti v obrazu nástroj sám kolem nejdominantnějšího objektu vytvoří polygon, který je možné dále vylepšit. K vytvořeným anotací lze přidávat dodatečné tagy. Změna pozice anotace ve videu je značena jako „keyframe“, které uživatel definuje. Mezi „keyframy“ je potom pozice anotace ve videu interpolována a pokud se příliš vzdálí od objektu, je možné přidat další „keyframe“ pro zvýšení přesnosti. Vytvořený seznam anotací je možné exportovat v JSON formátu.

Uživatelské rozhraní je přehledné, intuitivní a konzistentní v barevné paletě. Hlavní část obrazovky zabírá aktuálně zpracovávaný snímek. Je možné provádět základní manipulace s obrazem přiblížení a posunutí. Horní lišta slouží pro navigaci v nástroji a datasetu. Také obsahuje informaci, zda je uživatel v módu vytváření anotací, nebo v módu validace. Levá postranní lišta nabízí volby ohraničujících tvarů a lišta na druhé straně obsahuje knihovnu anotací. Pokud není vybrán typ anotace, uživatel je dodatečně dotázán po vytvoření ohraničení. Dále obsahuje v dolní části dodatečné tagy, které lze k vytvořeným anotacím přidat. Časová osa videa se nachází v dolní liště pod snímkem. Zde jsou vyznačené anotace a jakou část videa zabírají. Ukázka uživatelského rozhraní je na Obrázku 2.1.



Obrázek 2.1: Anotační nástroj V7 Darwin a jeho uživatelské rozhraní. Nástroj pracuje s videem jako se sekvencí snímků a pro každý snímek je anotace vytvořena samostatně. Anotace ale nemusí být vytvořeny pro každý snímek, nástroj umí doplnit pozice chybějících anotací pomocí interpolace. Snímky lze přehrát v původní rychlosti a uživatel tak má pocit, že skutečně pracuje s videem. Převzato z [21].

Roboflow

Roboflow je komerční nástroj vytvořený stejnojmennou společností z Iowi, USA [20]. Jedná se o celou platformu, která neumožňuje pouze vytvářet a sdílet datasety s obrazovými daty, ale také nad nimi přímo vytvářet modely připravené k použití. Samotná aplikace funguje v prostředí webového prohlížeče a je postavena na moderních JavaScriptových knihovnách jako React[1], nebo PixiJS[2], která využívá grafické rozhraní WebGL. Ukázka uživatelského rozhraní je na Obrázku 2.2.

Základní částí v Roboflow není dataset, ale projekt, protože nástroj svojí bohatou nabídkou funkcionalit přesahuje možnosti klasického datasetu. Projekt je možné založit na vlastních datech, nebo si vytvořit kopii již existujícího. Možnost naklonovat existující projekt ocení noví uživatelé, kteří si chtějí nástroj v první řade vyzkoušet, ale nemají po ruce nějaký dataset. Projekt je privátní, ale je možné jej sdílet s konkrétními uživateli prostřednictvím týmů, nebo ho nastavit jako veřejný, a tím se stane dostupný pro celou komunitu Roboflow. Projekty se dají libovolně spojovat a tím sjednotit obrazová data, anotační třídy a již vytvořené anotace.

Nahrávání obrazových dat do projektu podporuje běžné formáty JPG, PNG, BMP a pro video data MOV, MP4 a AVI. Soubory je možné před finálním nahráním na server třídit a některé odstranit. Jelikož nástroj umožňuje i trénování modelů, soubory je možné rozdělit do tří kategorií - trénovací, validační a testovací. U video souborů je nutné zvolit

[1] React - dostupné z: https://reactjs.org/
[2] PixiJS - dostupné z: https://pixijs.com/

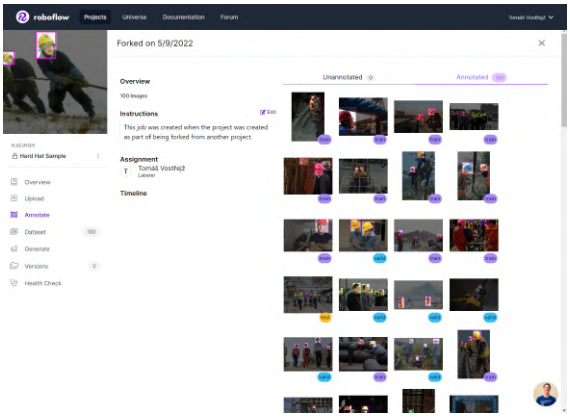
vzorkovací frekvenci. Nástroj po nahrání video rozdělí na jednotlivé snímky, které se anotují samostatně.

Samotný proces vytváření anotací podporuje dva tvary, a to obdélník a polygon. Každý projekt má jednu, nebo více skupin anotačních tříd, ze které jsou k anotacím přiřazeny konkrétní třídy po vytvoření ohraničujícího tvaru. Vybrat lze kliknutím, nebo dobře značenou klávesovou zkratkou. Obraz je možné přiblížit nebo posunout, ale není možné dočasně měnit jas nebo kontrast. K vytvořeným anotacím nelze přidávat dodatečné informace, například tagy. Pro zrychlení vytváření ohraničení objektů jde použít vestavěný asistent, který využívání existujících natrénovaných veřejných, nebo uživatelem nahraných, modelů a částečně anotovaný obraz dotáhnout do konce. Na anotacích je také možné pracovat v týmu, kde existuje několik rolí uživatelů například pro vytváření anotací, validaci existujících anotací, správce souborů a další.

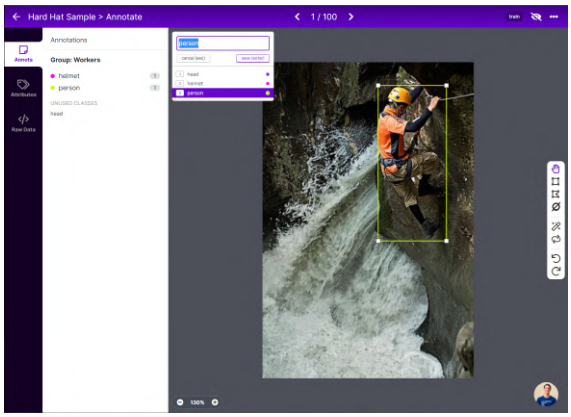
Další krok v projektu umožňuje vzít anotovaný dataset a aplikovat na něj různá předzpracování, které mohou pomoci zlepšit výsledky naučeného modelu. Proces předzpracování zahrnuje transformace obrazu jako aplikování automatické úrovně jasu a kontrastu, oříznutí obrazu, nebo převedení na odstíny šedé. Po předzpracování je možné do datasetu vygenerovat další soubory aplikováním stejných transformací jako v předchozím kroku, doplněné o další možnosti jako rotace obrazu, převrácení, přidáním šumu a další. Cílem je zvětšit objem datasetu pro zlepšení výsledků.

Takto připravený dataset lze použít pro trénování modelů přímo v prostředí aplikace, nebo jej exportovat do jednoho z několika nabízených formátů (např. COCO pro JSON, Pascal VOC pro XML). Pokud se uživatel rozhodne pro první volbu, pro dokončení trénovaní si může prohlédnout různé grafy o průběhu a výsledné statistiky úspěšnosti. S hotovým modelem může dále pracovat a dataset iterativně zlepšovat, protože aplikace má vestavěné verzování.

V uživatelském rozhraní je lehké se zorientovat. Při první návštěvě nástroj provádí uživatele celým procesem práce na projektu pomocí malých nápověd. Rozhraní je jednoduché, moderní a minimalistické. K ovládání používá dobře značené klávesové zkratky. Ovládací prvky doplňují jednoduché ikony a každá akce má svoji odezvu. Při vytváření anotací se ukládá historie změn a uživatel má možnost předělat poslední úpravy.



(a) Roboflow umožňuje vytvořit kopii existujícího veřejného projektu a uživatel tak může nástroj rychle vyzkoušet na reálných datech. Obrazová data jsou rozdělena do tří kategorií - trénovací, validační a testovací. Data jsou také rozdělena podle toho, zda už jsou vytvořeny anotace.



(b) Vytváření ohraničení podporuje tvar odlélník a polygon. Po dokončení tvaru uživatel zadá anotační třídu objektu a může použít příslušnou klávesovou zkratku. K anotací nelze přidávat dodatečné údaje, nebo měnit barvu ohraničení. Položka *Attributes* po levé straně referuje na atributy obrázku, nikoliv anotací.

Obrázek 2.2: Ukázka uživatelského rozhraní aplikace Roboflow. Rozhraní je moderní, minimalistické a přehledné. Převzato z [20].

Kapitola 3

Technologie pro tvorbu webových aplikací

V kapitole jsou popsány technologie vhodné pro tvorbu moderních interaktivních webových aplikací. Úvodní část představuje jazyk JavaScript a knihovnu Express, která je velmi populární pro tvorbu serverů používající REST rozhraní. Druhá část obsahuje informace o vývojové platformě Angular používané především pro tvorbu SPA[1] aplikací. Dále je popsán jazyk TypeScript, který Angular používá. Třetí část se zabývá databází MongoDB. Na závěr je zmíněn nástroj pro virtualizaci Docker a jak může být využit jak při vývoji aplikace, tak při samotném nasazení do produkce.

3.1 Express

Express je knihovna postavena na prostředí Node.JS a jazyce JavaScript. JavaScript je interpretovaný skriptovací jazyk a pro tvorbu webových aplikací je dnes velmi populární volbou. Vznikl v roce 1995 a ze začátku moc používán nebyl, protože prohlížeče jeho specifikaci neimplementovaly v plném rozsahu a obsahovaly chyby. Situace se změnila v roce 2009, kdy vyšla nová specifikace ECMAScript 5, která přitáhla širší pozornost vývojářů a všechny moderní prohlížeče od roku 2012 plně podporují ECMAScript verze 5.1 [18].

JavaScript ale není omezen pouze na použití ve webovém prohlížeči. Spustit zdrojový kód je možné i pomocí interpretačních nástrojů, jako je například V8[2]. Jedná se o volně dostupný a šiřitelný software napsaný v jazyce C++. Používají ho například prohlížeče Google Chrome, Opera GX a nový Microsoft Edge. Interpretačních nástrojů existuje celá řada a všechna zařízení s některým z nástrojů dokážou interpretovat JavaScriptový kód.

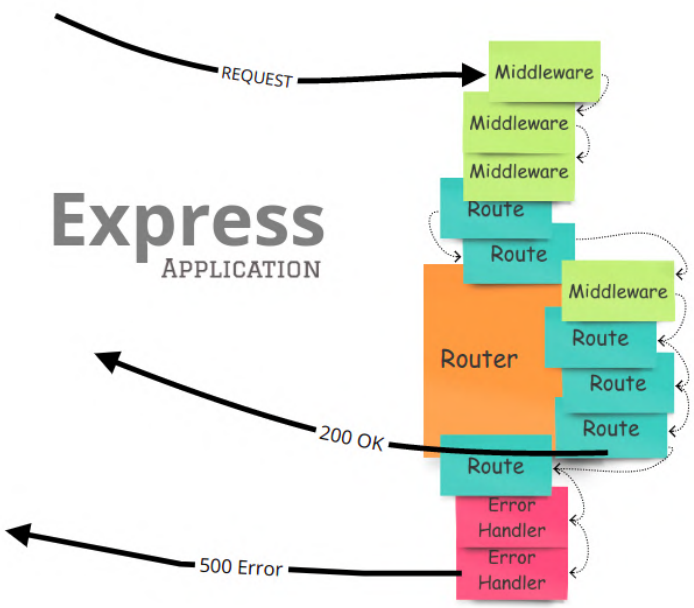
Právě na výše zmíněném nástroji V8 staví běhové prostředí Node.js, které se používá pro vývoj serverových aplikací. Z něj jsou odstraněny nepotřebné funkce, které využily pouze webové stránky, jako je funkce geolokace. Na druhé straně je nástroj V8 rozšířen o funkce užitečné pro běh na lokálním stroji, například o programové aplikační rozhraní umožňují přístup k souborovému systému. Díky němu lze pomocí Node.js číst a zapisovat soubory na lokálním úložišti.

[1] SPA - Single-page application, česky: Jednostránková aplikace
[2] V8 - dostupné z: https://v8.dev/

Knihovna Express

K vytvoření jednoduché serverové aplikace postačuje Node.js, ale obsluhování velkého počtu požadavků na různých koncích rozhraní REST API usnadní knihovna Express. Express je malá, flexibilní a volně šiřitelná knihovna [5]. Umožňuje jednoduchou definici posluchačů pro příchozí HTTP požadavky na základě adresy a typu požadavku. Také nabízí možnost použít několika middleware. Middleware lze použít například pro kontrolování autentizačního tokenu v hlavičce, parsování těla požadavku a nebo konfiguraci CORS[3].

Knihovna je i výkonná z pohledu obsluhy velkého množství příchozích požadavků. Používá neblokující a událostmi řízený vstupně/výstupní model. Tok programu tak řídí výskyt událostí, které jsou zachytávají takzvaní „posluchači“. Posluchač definuje název události, na kterou reaguje, a příslušnou funkci, kterou následně spustí. Kód je asynchronní a díky tomu se při obsluze požadavků nečeká na dokončení těch předchozích. Ilustrace znázorňující zpracování požadavku je na Obrázku 3.1.



Obrázek 3.1: Diagram ilustrující zpracování požadavku pomocí Express aplikace. Požadavek nejdříve prochází globálním middleware, může jít například o nastavení CORS a parsování obsahu těla zprávy. Poté je požadavek zpracován jednou z rout definované na Routeru. Ta může obsahovat dodatečný middleware, například ověření registrace uživatele. Pokud není požadavek zpracován žádnou routou, je zachycen middlewarem pro chyby. Převzato z [6].

Node Package Manager

Ve většině JavaScriptových aplikací a knihoven se setkáme se správcem balíčků NPM - Node Package Manager. Naprogramovat veškerou funkcionalitu v projektu může zabrat hodně času. Je pravděpodobné, že někdo v minulosti podobnou funkcionalitu potřeboval a vytvořil pro něj řešení. Toto řešení je možné sdílet s ostatními v podobě balíčku a kód je tak

[3]CORS – Cross-Origin Resource Sharing

znovupoužitelný. Vlastní implementace tak doplňují balíčky od jiných vývojářů a součástí projektu jsou jako závislosti, které lze doinstalovat.

Nástroj NPM se stará o přidávání balíčků, založení projektu, aktualizování verzí a podobné. V souboru `package.json` se nacházejí všechny potřebné informace jak o projektu, tak i o balíčcích, na kterých projekt závisí [10]. Také může obsahovat krátké skripty. Ty uživatel definuje sám, nebo může použít scripty ke spuštění aplikace a testování, které `npm` generuje automaticky při založení. Hlavní výhoda je v tom, že k přenosu projektu na jiný stroj postačí zdrojový kód doplněný o jeden soubor, podle kterého se stáhnou a doinstalují všechny balíčky, na kterých spuštění aplikace závisí, včetně odpovídajících verzí. Také jde rozlišit, jaké balíčky se nainstalují pro fázi vývoje a které jsou určeny pro produkční nasazení. Při instalování jde volitelně nainstalovat jen některé, nebo oboje.

3.2 Angular

Angular je populární vývojová platforma pro tvorbu interaktivních webových aplikací postavená na jazyce TypeScript. TypeScript je volně šiřitelný programovací jazyk vytvořený společností Microsoft. Jedná se o nadstavbu nad jazykem JavaScript, takže každý korektní JavaScriptový kód je i korektním TypeScriptovým kódem. Navíc proti JavaScriptu nabízí především statické typování [11] a pokročilejší konstrukce, například výčtový datový typ (`enum`).

Statické typování může zlepšit čitelnost kódu a do určité úrovně dovoluje kontrolovat jeho správnost za běhu. Také umožňuje IDE[4] našeptávat programátorům při psaní kódu. Používání statického typování není povinné, pokud není uživatelem specifikovaný datový typ, je prováděno dynamicky, nebo může být použit výchozí typ `any`.

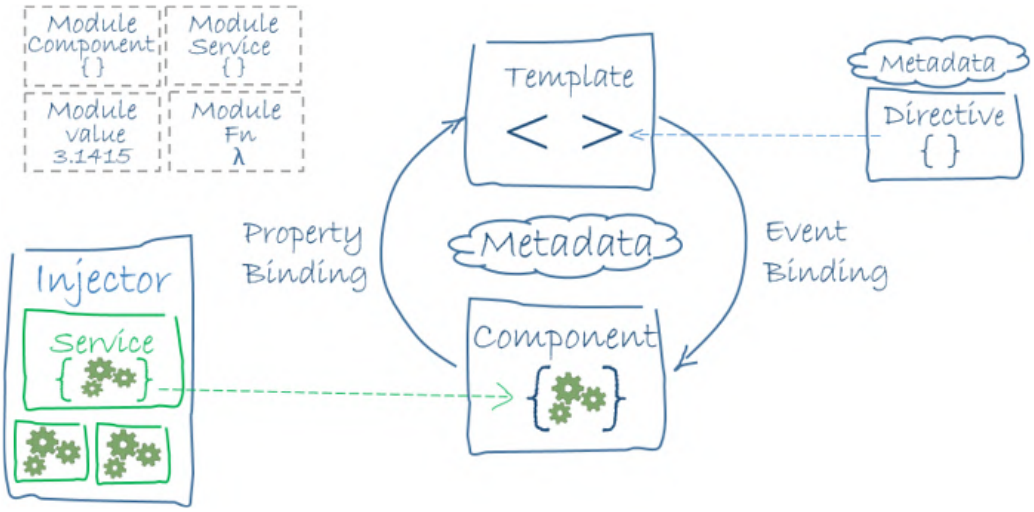
Jelikož webové prohlížeče s TypeScriptem neumí pracovat, pro programování webových aplikací se nepoužívá přímo a zdrojové kódy se kompilují do JavaScriptu. Výhoda kompilace je v tom, že v jejím průběhu lze odhalit chyby, které by se jinak v běžném JavaScriptovém kódu projevily až při běhu aplikace, nebo během testování. JavaScript je dynamicky typovaný jazyk, takže definované datové typy z TypeScriptu za běhu aplikace nemusí být během používání dodrženy. Typickým příkladem jsou data z formulářů, které vyplňuje uživatel.

Platforma Angular

Angular je vývojová platforma určená pro vývoj jednostránkových webových aplikací. Prvotní verze stavěla na JavaScriptu a vyšla pod názvem AngularJS v roce 2010. Framework byl později zcela přepsán do jazyka TypeScript a v roce 2016 vyšla novější verze, označena pouze jako Angular. Někdy se také setkáme s označením Angular 2. Všechny následující verze rovněž používají TypeScript [1].

Angularu staví na konceptu dekompozice webové stránky na znovupoužitelné komponenty. Velikost a počet komponent je čistě v rukou programátora. Pro manipulaci s komponentami a řízení vykreslování používá Angular inkrementální DOM. Každá komponenta je zkompilována do sekvence instrukcí. Pomocí instrukcí pak vytváří stromovou strukturu DOMu a aktualizace se provádí přímo v daném místě, když dojde ke změně zobrazovaných dat. Angular toho jako vývojová platforma nabízí o mnoho víc. Součástí je i vlastní CLI[5] pro založení, spouštění a testování aplikace. Propojení mezi funkčními částmi Angularu je na Obrázku 3.2.

[4] IDE – Integrated Development Environment, česky: Vývojové prostředí
[5] CLI – Command Line Interface, česky: Příkazový řádek



Obrázek 3.2: Na obrázku jsou vidět základní části architektury aplikace v Angularu a jejich vzájemné vztahy. Převzato z [1].

Komponenty

Komponenty jsou základním stavebním blokem aplikace a jejich úkolem je vykreslit nějakou část uživatelského rozhraní. Komponenty jsou uspořádány do stromové struktury a mohou si mezi sebou předávat zprávy, vždy však pouze o jednu úroveň. Komponenty můžeme rozdělit na dva typy:

- **Bezstavové komponenty** - nemají vnitřní stav a jsou plně závislé na svých parametrech. Příkladem může být komponenta vkreslující chybovou hlášku, která pouze zobrazí zprávu a případně mění barvy textu.
- **Stavové komponenty** - mají vnitřní stav a také mohou dostávat parametry. Příkladem může být tlačítko, které může být povoleno a zakázáno, ale navíc bude mít vnitřní logiku, která ho zakáže po určitý počet sekund po stisknutí. Po uplynutí lze kliknout znovu, ale stav si tlačítko řídí samo.

Moduly

Spolu související komponenty se v Angularu seskupují do větších bloků - modulů. Modul může volitelně definovat i svůj router a příslušné komponenty odpovídající stránkám. Moduly jsou uspořádány ve stromové struktuře. Modul s vyšší úrovní může používat komponenty z modulů z nižší úrovně. Moduly na nižší úrovni lze „líně“ načítat a modul je tak načten až v momentě, kdy je skutečně potřeba.

Vkládání závislostí

Součástí Angularu je vlastní framework pro vkládání závislostí. Jeho účelem je zvýšení efektivnosti a modularity aplikace. Závislosti jsou v Angularu externí služby nebo objekty, které třída vyžaduje ke správnému fungování. Podstatou tohoto přístupu je, že třída získá zdroje od vnějšího poskytovatele, místo vytváření zdrojů vlastních. Při vytváření instancí se framework stará o poskytnutí všech definovaných závislostí. Ty můžeme definovat na úrovni

komponent, modulů, nebo na úrovni celé aplikace. Pokud nějakou závislost definujeme na úrovni aplikace, ale nakonec ji použijeme jen v jedné komponentě, Angular je natolik sofistikovaný, že při kompilaci aplikace tuto skutečnost detekuje a optimalizuje.

Navigace v aplikaci

Navigace u jednostránkové aplikace probíhá tak, že se vykreslují a skrývají odpovídající komponenty, bez potřeby znovu načítat stránky v prohlížeči. Toho můžeme docílit udržováním globálního stavu aplikace, na jehož hodnoty budou komponenty reagovat. Nebo můžeme použít Angular Router, který se o změnu zobrazených komponent postará. Výhodou je, že se stav aplikace promítne i do samotné URL. Router adresu URL zadanou do prohlížeče interpretuje jako sekvence instrukcí následně měnící vzhled stránky. Router také obsahuje možnosti přidat podmínky, které musí být splněny pro umožnění přístupu k daným stránkám. Takové podmínky implementují takzvaní Guardi, kteří také definují případnou navigaci na jinou stránku.

Knihovna RxJS

Součástí Angularu je i knihovna RxJS, která se používá pro udržování stavu aplikace, nebo pro sdílení dat mezi komponentami. Je založena na asynchronním programování a obsluze událostí. Používá návrhový vzor pozorovatele, kde pozorovaný objekt udržuje seznam svých pozorovatelů a svůj vlastní stav. Pokud dojde ke změně stavu, jsou všichni pozorovatelé upozorněni na nová data. Komponenty se registrují u pozorovaného objektu a předávají mu funkci, která se provede při změně stavu. Dokud se komponenta neodhlásí, bude dostávat měnící se data.

3.3 MongoDB

MongoDB je databázový systém navržený pro internet a webové aplikace. Jedná se o NoSQL databázi postavenou na dokumentech [16]. Datový model a strategie persistence podporují vysokou propustnost jak při čtení, tak při zápisu dat. Umožňuje snadné škálování pomocí automatického převzetí služeb při selhání.

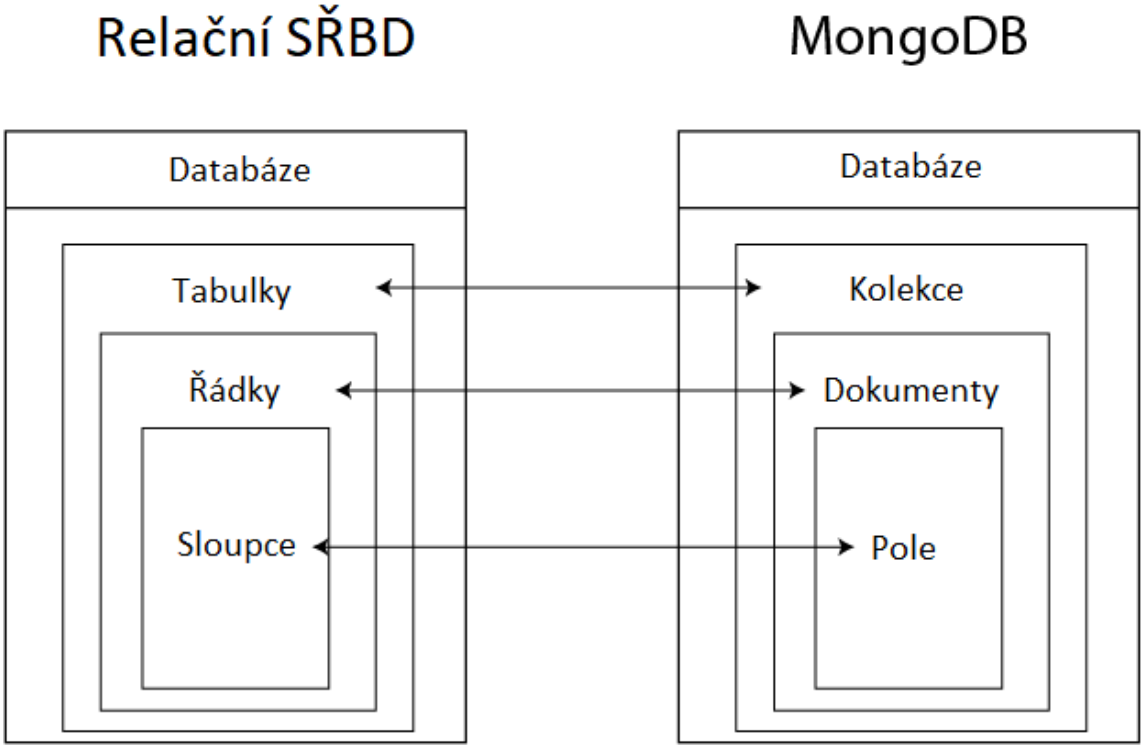
Na datovém modelu z dokumentů MongoDB je snadné stavět aplikace, protože má vlastní podporu pro nestrukturovaná data a nevyžaduje nákladné a časově náročné migrace při změně požadavků aplikace na databázovou strukturu. Dokumenty MongoDB jsou kódovány ve formátu podobném JSON, nazývanému BSON. BSON se přirozeně hodí pro moderní objektově orientované programovací metody, je nenáročný a rychlý. MongoDB používá BSON jako přenosový formát po síti pro dokumenty.

BSON na první pohled vypadá jako BLOB, ale existuje důležitý rozdíl: MongoDB databáze rozumí vnitřní struktuře formátu BSON. To znamená, že MongoDB může přistupovat k objektům v BSON, dokonce i k libovolně zanořeným, pomocí tečkové notace. To umožňuje MongoDB vytvářet indexy a najít tak hledané objekty podle zadaných výrazů v dotazu na jak nejvyšší úrovni, tak i pro vnořené klíče BSON.

MongoDB také podporuje rozsáhlé dotazy a úplné indexování. Tím se odlišuje od jiných databází založených na dokumentech, kde je zapotřebí samostatné serverové vrstvy, která slouží ke zpracování složitých dotazů. Mezi jeho další funkce patří automatické rozdělování dat mezi clustery, replikace a snadné ukládání.

Hlavní komponenty MongoDB jsou popsány v následujícím seznamu. Vizuální srovnání komponent MongoDB a tradičního Relačního SŘBD je na Obrázku 3.3.

- **Databáze** - je fyzický kontejner pro kolekce. Každá databáze má svou vlastní sadu souborů v systému souborů. Jediný server MongoDB má obvykle více databází.
- **Kolekce** - je skupina dokumentů MongoDB. Je to ekvivalent tabulky u Relačního SŘBD. Kolekce existuje uvnitř jediné databáze. Kolekce může svým obsahem porušovat strukturu definovaného schéma. Dokumenty v rámci jedné kolekce mohou mít různá pole. Obvykle jsou všechny dokumenty ve sbírce podobné, jelikož obvykle slouží ke stejnému účelu.
- **Dokument** - je sada dvojic klíč–hodnota. Dokumenty mají dynamické schéma. Dynamické schéma znamená, že dokumenty ve stejné kolekci nemusí obsahovat stejné pole nebo strukturu a společné pole v kolekci dokumentů mohou obsahovat různé typy dat.

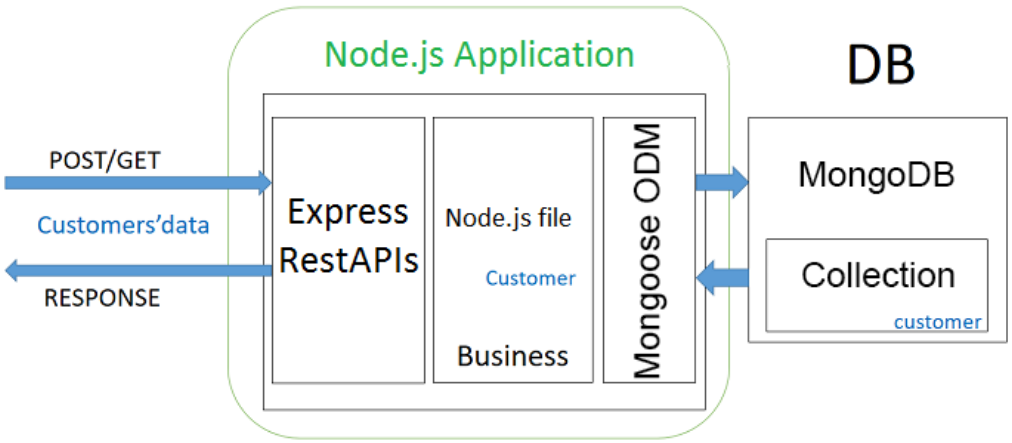


Obrázek 3.3: Srovnání dílčích částí tradičního Relačního SŘBD a MongoDB. Zdroj: autor.

3.4 Mongoose

Mongoose je malá JavaScriptová knihovna pro Objektově dokumentové mapování v Node.js [8]. Mongoose nám umožňuje připojit se k databázi a provádět operace asynchronně bez psaní nezpracovaných dotazů. Abstrahuje dotazy pro MongoDB a usnadňuje interakci s databázovými modely jako s objekty. MongoDB neumožňuje definovat schéma dokumentů a jejich vzájemné vztahy. Mongoose tento nedostatek vyplňuje. Základní části knihovny jsou:

- **Schémata** - zatímco MongoDB pracuje bez schémat, databáze používající SQL definuje schéma prostřednictvím definice tabulky. Schéma v Mongoose je datová struktura dokumentu (nebo tvar dokumentu), která je vynucována prostřednictvím aplikační vrstvy.
- **Modely** - jsou konstruktory vyššího řádu, které berou schéma a vytvářejí instanci dokumentu ekvivalentní záznamům v relační databázi. Definuje názvy a typy jeho polí. Model také může definovat svůj vztah s jiným modelem.



Obrázek 3.4: Ukázka ekosystému serverové aplikace používající knihovnu Mongoose a MongoDB. Převzato a upraveno z [2].

3.5 Docker

Docker je rozšířený nástroj pro virtualizaci, hojně používaný pro nasazení webových aplikací a služeb. Svoje místo má ale i v procesu vývoje software. Tradiční nástroje poskytují abstrakci hardwarové úrovně svým virtuálním strojům. Takové prostředí pak musí mít vlastní operační systém, knihovny a ovladače. Docker volí alternativní přístup a se svými virtuálními prostředími sdílí část operačního jádra. Prostředí si potom doinstaluje potřebné knihovny pro fungování aplikace a může sloužit například jako webový server, nebo databáze [3, 4]. Ilustrace obou přístupů je na Obrázku 3.5.

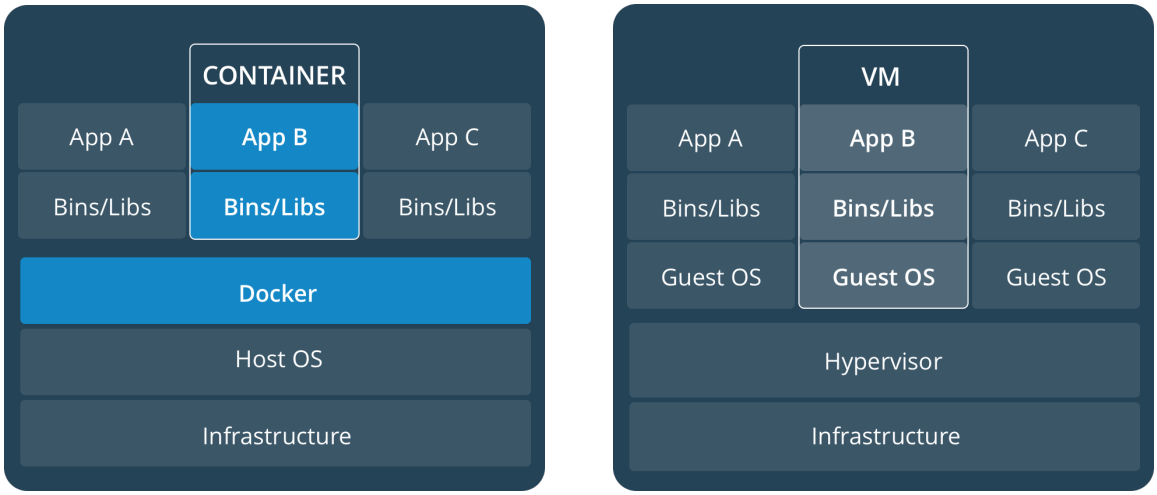
Mít takto zapouzdřenou část aplikace je výhodné i během vývoje aplikace. Docker vytváří vlastní běhové prostředí pro každou část projektu a ten je díky tomu přenositelný mezi počítači a při správné konfiguraci i rychle spustitelný. Navíc dovoluje rychlé testování a konfiguraci různých technologií a knihoven, aniž by docházelo k ovlivnění nastavení hostitelského systému. Další výhodou zapouzdření do Dockeru je stálost verzí prostředí. Občas se stane, že běhové prostředí má aktualizaci a část aplikace přestane fungovat, zvláště u déle trvajících projektů. Základní části Dockeru jsou:

- **Obrazy disků** - obsahují souborový systém a konfiguraci prostředí. Slouží k vytváření kontejnerů. Docker má svoji vlastní knihovnu předem připravených obrazů na webu Docker Hub[6]. Uživatelé je mohou libovolně použít, modifikovat pro svoji aplikaci a přidat do veřejné knihovny. Alternativní možnost je použít **Dockerfile** soubor.

[6]Docker Hub – dostupné z: https://hub.docker.com/

Do něj lze zapsat posloupnost příkazů, který na začátku vezme nějaký obraz disku a postupně jej modifikuje podle potřeby, jako doinstalovat knihovny a nastavit proměnné prostředí. Vznikne tak nový lokální obraz.

- **Kontejnery** - jsou instance vytvořené na základě obrazů disků. Jednotlivé instance jsou od sebe odděleny a obsahově se vlivem běhu aplikace můžou lišit, i když vznikly ze stejného obrazu. Pro správu více instancí najednou se používá **docker-compose.yml** soubor. Obsahuje konfigurace mapování portů a složek mezi kontejnerem a hostitelem. Dále může definovat síťová rozhraní mezi kontejnery a vzájemné závislosti, aby se například kontejner se serverem spustil první a až potom kontejner s databází.



(a) Ukázka ekosystému Dockeru.

(b) Tradiční virtualizační nástroje.

Obrázek 3.5: Porovnání způsobu virtualizace Dockeru oproti tradičním systémům. Docker sdílí vrstvu hostitelského operačního systému a šetří tak místo i čas při vytváření a správě kontejnerů. Převzato z [4].

Kapitola 4

Návrh nástroje a uživatelského rozhraní

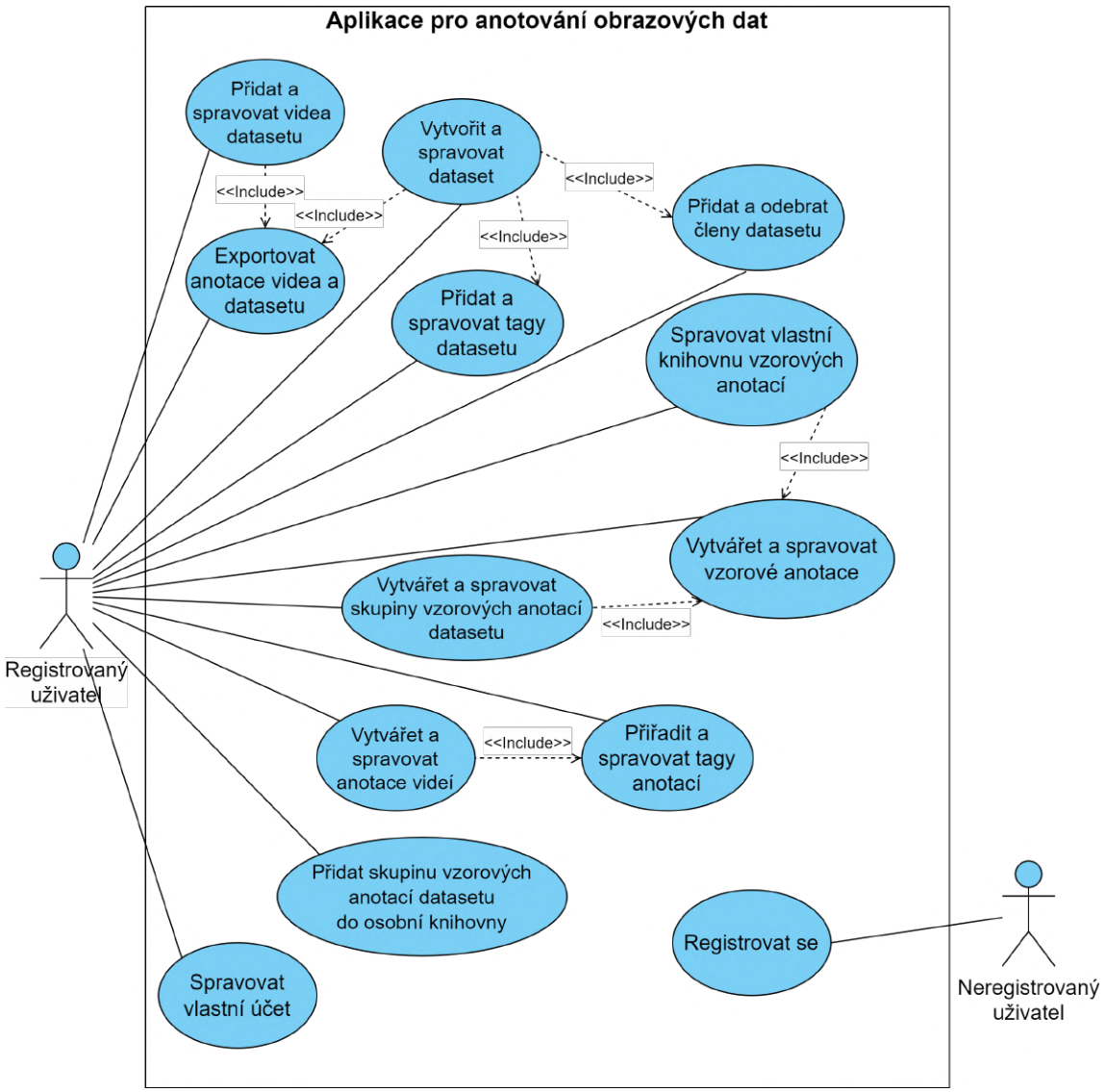
Kapitola se zabývá návrhem nástroje jako celku, jeho hlavních částí a jejich funkcionalit. Součástí návrhu je i uživatelské rozhraní. Návrh byl vytvořen na základě požadavků z kapitoly 2. Pro inspiraci posloužil i rozbor existujících řešení, zejména některé užitečné funkcionality.

Aplikace má dvě role uživatelů, kde jedna z nich je role neregistrovaného uživatele. Ten má jediný případ užití a tím je vytvoření účtu. Veškerá funkcionalita nástroje je tedy schována za podmínkou vlastního účtu. Registrovaný uživatel po přihlášení může nástroj plně ovládat. Role a jejich případy užití jsou znázorněny na digramu 4.1.

Aplikace má specifické použití a je navržena pro zkušenější uživatele orientující se prostředí umělé inteligence. Pro nově registrovaného uživatele aplikace vytvoří nový prázdný dataset s výchozím názvem, který lze kdykoliv změnit, a přesměruje ho na hlavní stránku nástroje. Zde začne přidáním prvního videa do datasetu a poté už může začít vytvářet anotace. Video je možné načíst lokálně pro okamžitou práci, nebo nahrát na server. Uživatel má možnost s videi a datasety provádět CRUD[1] operace, navíc do datasetu může přidat další členy. Dále má přístup ke svému profilu, kde lze změnit osobní a přihlašovací údaje.

[1] CRUD - operace pro vytvoření, čtení, aktualizace a mazání

Use Case Diagram



Obrázek 4.1: Diagram případů užití pro návrh aplikace. Zdroj: autor.

4.1 Uživatelské rozhraní

Návrh uživatelského prostředí počítá s tím, že uživatel je zkušenější v práci s počítačem. Přesto se snaží být jednoduchý, přehledný a intuitivní. V následující sekci je popsán návrh hlavní strany nástroje, skládající se především z přehrávače videa a kanvasu pro vytváření ohraničujících tvarů. Další ukázky uživatelského rozhraní jsou popsány v dalších sekcích. Ukázka návrhu hlavní obrazovky je na Obrázku 4.2.

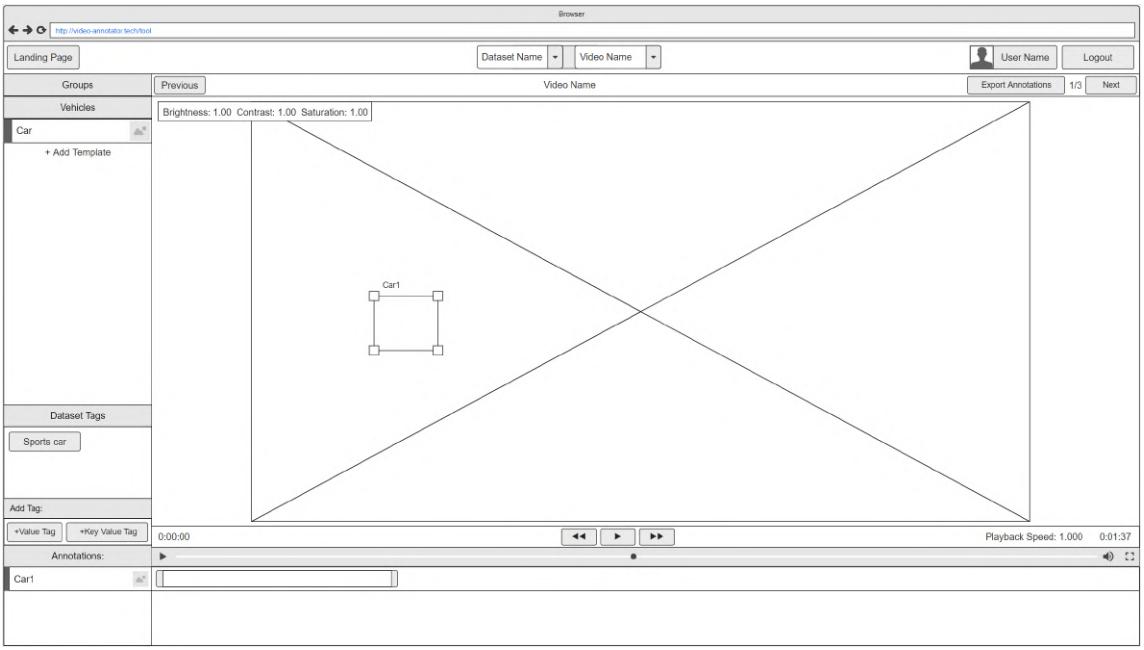
Horní část obrazovky zabírá navigační menu. Uživatel zde vidí aktuální dataset a video, na kterém pracuje. Zároveň může na název videa nebo datasetu kliknout pro jeho změnu. Také je zde možnost přesměrování na uživatelský profil a možnost odhlášení. Navigační

menu v dolní části doplňuje lišta pro posunutí na předchozí a následující video s informací, na jakém indexu videa se uživatel nachází z celkového počtu.

Největší část hlavní obrazovky zabírá přehrávač videa. Od ostatních přehrávačů se ale liší tím, že ovládací prvky videa nejsou jeho přímou součástí, ale jsou odděleně zabudovány v dolní části nástroje. Důvodem je, že obraz lze libovolně přiblížit a posunout pro přesnější vytváření a manipulaci s anotacemi. Ovládací prvky by se touto transformací lehce dostaly mimo dosah uživatele. Součástí ovládacích prvků je i možnost měnit rychlost přehrávání. Ovládací prvek podporuje změnu hodnoty scrollováním, nebo pomocí vyskakovacího posuvníku. Stejnou funkcionalitu nabízejí ovládací prvky pro změnu filtru obrazu v levém horním rohu videa. Zde jsou možnosti pro úpravu jasu, kontrastu a sytosti videa. Jejich pozice je statická a vždy k dispozici. Hodnoty filtrů lze libovolně kombinovat a jejich efekt je pouze dočasný a nemění tak originální video. Video překrývá plátno, na kterém jsou vykresleny existující anotace a na kterém uživatel může tahem nebo kliknutím vytvořit nové.

Po levé straně obrazovky se nacházejí skupiny anotačních šablon. Šablony slouží k vytváření konkrétních anotací a obsahují jen základní údaje. Dataset může mít několik skupin a jsou sdíleny se všemi členy datasetu. Po vybrání šablony uživatel může začít vytvářet ohraničující tvar nějakého objektu přímo ve videu. Spodní část panelu vedle videa zabírají tagy. Tagy jsou definovány na úrovni datasetu a jsou dvojího typu. První obsahuje pouze hodnotu, druhý se skládá z dvojice klíč a hodnota. Tyto tagy jde libovolně přiřazovat ke konkrétním anotacím, nebo pro ně vytvořit úplně nové.

Dolní část obrazovky zabírá prostor pro konkrétní anotace. Po levé straně je seznam anotací s příslušným pořadovým označením podle dané šablony. Hlavní část okna má shodnou šířku s časovou osou videa, protože obsahuje informaci o tom, jak dlouho anotace ve videu trvá. Výchozí délka pro nové anotace je předem daná, ale uživatel ji může libovolně zkrátit, nebo prodloužit potažením krajů. Kliknutím se také může ve videu libovolně posouvat.



Obrázek 4.2: Ukázka uživatelského prostředí nástroje. Na obrázku je hlavní strana nástroje určená pro přehrávání videí a vytváření anotací. Nástroj umožňuje obraz přiblížit, posunout, nebo dočasně upravit jas, kontrast a nasycení obrazu. V dolní části se nacházejí konkrétní anotace vytvořené na základě šablon, které se nacházejí v levé části. Zdroj: autor.

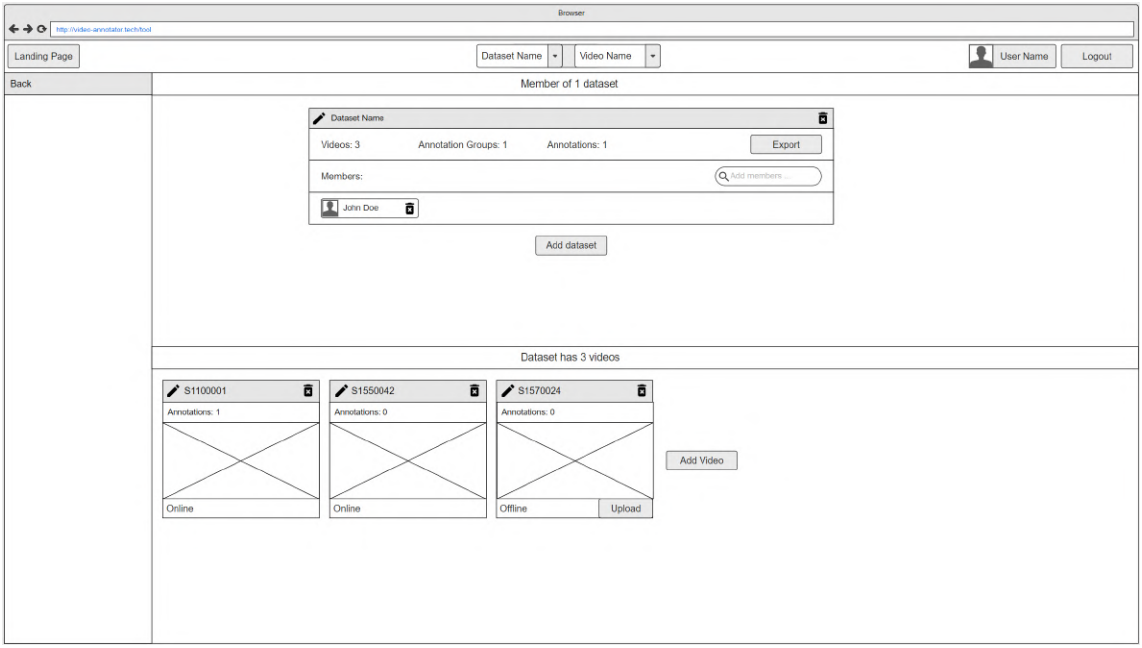
4.2 Správa datasetů a videí

Datasety jsou základním prvkem nástroje. Uživatel může datasety libovolně zakládat, upravovat a mazat. Dataset se skládá z videí, skupin anotačních šablon a tagů. Tyto prvky jsou sdíleny mezi všemi členy datasetu, které lze přidat. Členové mají stejná oprávnění jako vlastník datasetu a mohou tak libovolně přidávat videa a další prvky. Někteří členové mohou například vytvářet anotace pouze z jedné konkrétní skupiny šablon. Ukázka návrhu uživatelského rozhraní je na Obrázku 4.3.

Většina obrazovky pro přehled datasetů a videí je shodná a liší se tak pouze obsahem. V případě datasetů hlavní část obrazovky zabírají karty s jednotlivými datasety. Zde jsou zobrazeny základní informace jako počet videí, anotačních skupin a celkový počet anotací napříč videi. Pokud dataset obsahuje alespoň jednu anotaci, je možné vygenerovat soubor obsahující informace o videích a jejich anotacích. V opačném případě je uživatel upozorněn, že dataset neobsahuje žádné anotace. V dolní části je seznam všech členů datasetu kromě majitele samotného, aby nemohl být omylem odstraněn a zamezuje se tak existenci datasetu bez majitele. Pokud uživatel chce do datasetu přizvat další členy, může je vyhledat podle názvu účtu a procházet výsledky. Ze seznamu vyhledaných jsou vyřazeni stávající členové.

Přehled videí se taktéž skládá z karet reprezentujících jednotlivá videa. Karta obsahuje základní informace jako název videa, počet vytvořených anotací, náhled a dostupnost. Uživatel může měnit název videa. Při nahrání nového videa si systém interně uloží originální název, který nelze změnit. V souboru s exportovanými anotacemi je pak uveden stávající i původní název. Při odstranění videa se po uživateli požaduje potvrzení v modálním okně. Nově přidané video se nezačne okamžitě nahrávat na server, ale uživatel může zvolit, zda jej načíst lokálně, nebo nahrát.

Pokud se rozhodne pracovat s videem lokálně, soubor je načten do mezipaměti, ale základní informace o videu se stále na server nahrají, včetně vytvořených anotací. Pokud uživatel stránku aplikace opustí, při další návštěvě je pro pokračování v práci požádán o opětovné vybrání souboru. Ať už je soubor načten lokálně, nebo jen vybrán, uživatel může kdykoliv soubor nahrát na server a bude tak vždy dostupný a to i pro ostatní členy. Po dokončení nahrávání nástroj nahradí lokální soubor za odkaz na video na serveru, aby uvolnil místo v paměti.

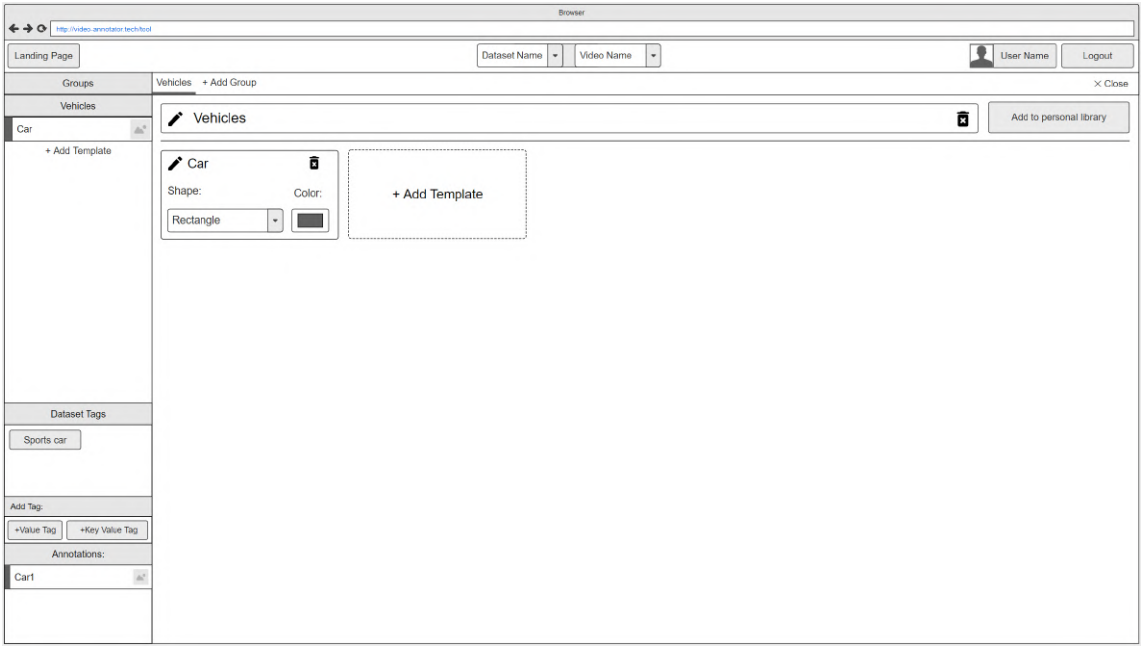


Obrázek 4.3: Ukázka návrhu uživatelského prostředí pro správu datasetů a videí. Obrázek obsahuje oba typy v jednom návrhu, protože stránky vypadají podobně a liší se pouze obsahem karet. Karty zobrazují základní informace a umožňují měnit název a smazat objekt. Zdroj: autor.

4.3 Modularita anotací

Součástí řešení je i modularita anotací. To je vyřešeno pomocí anotačních šablon. Šablony obsahují pouze základní informace jako je název třídy anotace, barva a tvar ohraničení. Tyto šablony slouží pro vytváření konkrétních anotací, které základní informace dědí, ale lze je doplnit o tagy. Tagy reprezentují dodatečné informace a vždy obsahují pole pro hodnotu a volitelně i pole klíče. K anotaci jde přiřadit libovolný tag definovaný na úrovni datasetu, nebo vytvořit zcela nový pouze pro jednu anotaci. Anotace jsou samostatné objekty a přestože jejich tvar nelze po vytvoření změnit, jejich název a barvu ano.

Šablony jsou shlukovány do skupin a patří k nějakému datasetu. Pokud uživatel vytvoří nějakou skupinu a chtěl by ji v budoucnu použít i jiného datasetu, může si ji zkopírovat do své osobní knihovny. Knihovna je dostupná na stránce s profilem uživatele, kde může libovolně vytvářet nové, nebo upravovat již existující. V datasetu je možnost vytvořit novou a prázdnou skupinu, nebo přiřadit některou z osobní knihovny. Ukázka návrhu uživatelského rozhraní je na Obrázku 4.4.



Obrázek 4.4: Na obrázku je ukázka překryvového okna pro úpravu anotačních šablon. Šablony tvoří skupiny, které jsou dostupné všem členům datasetu. Uživatelé si je mohou přidat do osobní knihovny, nebo je z knihovny přiřadit datasetu. Podobné okno je použito i pro úpravu konkrétních anotací. Navíc však obsahuje i časové údaje, kdy se ve videu objevuje. Zdroj: autor.

4.4 Vytváření ohraničení anotací

Každá anotace má tvar, který ohraničuje nějaký objekt v obraze. V případě videa je možné tento tvar znovu vytvářet pro všechny snímky, nebo je vytvořit pouze pro některé a chybějící pozice mezi nimi dopočítat. K tomuto účelu je nutné vybrat, nebo vytvořit, robustní knihovnu, která podporuje požadované tvary zadání a dále s nimi dokáže manipulovat. V prostředí webu je běžné, že velikost okna se za běhu aplikace mění a s ním i obsah. Dále uživatel může video libovolně přibližovat a posouvat. Knihovna si s těmito změnami musí poradit a zároveň být dostatečně rychlá ve vykreslování, jelikož pozice anotací v přehrávaném videu se mohou měnit v řádu desítek za vteřinu. V následujícím textu jsou popsány dvě kandidátní knihovny a jejich ukázka je na Obrázku 4.5.

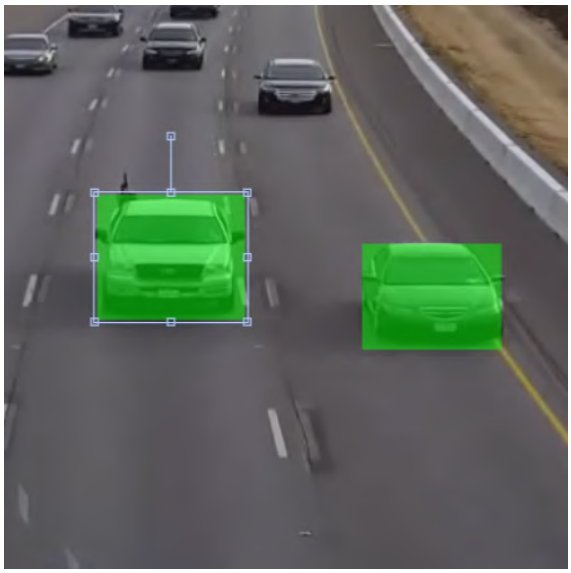
FabricJS

FabricJS[2] je malá JavaScriptová knihovna. Umožňuje vytváření většiny požadovaných tvarů a malou programovou nadstavbou by zvládala potřebné manipulace (např. vytvoření tvaru tahem myší). Knihovna pracuje s HTML kanvasem, který perfektně překrývá video. Podporuje i funkce přiblížení a posunu, ale pouze za předpoklady, že velikost kanvasu se nemění. Velikost se ale mění nejen při změně velikosti okna, ale i při transformaci videa a knihovna se chová nepředvídatelně.

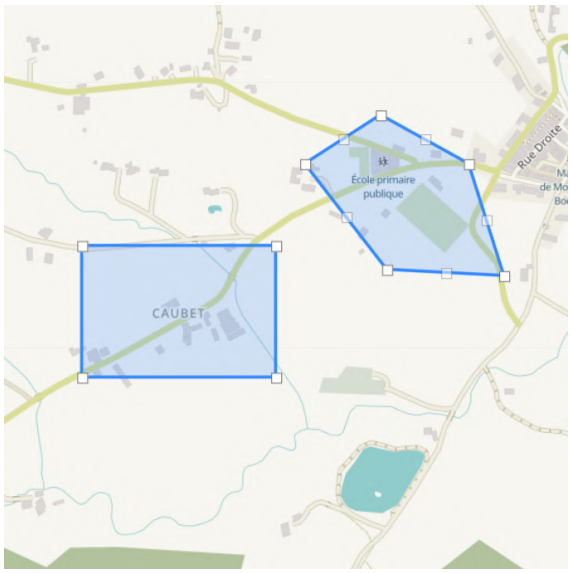
[2]FabricJS - dostupné z: http://fabricjs.com/

Leaflet.Editable

Dalším kandidátem je plugin Leaflet.Editable[3], který rozšiřuje populární knihovnu Leaflet[4]. Leaflet primárně slouží pro zobrazení geodat, ale součástí její implementace je i zakreslování geometrických tvarů do map. Podporuje všechny potřebné tvary a díky rozšíření Editable lze s tvary libovolně manipulovat. Nevýhodou je, že obě části počítají s geologickými souřadnicemi (zeměpisná délka a šířka), které je nutné přepočítávat na klasické souřadnice obrazu videa.



(a) Testování knihovny FabricJS přímo v aplikaci. Knihovna podporuje základní tvary, jako čtverec a polygon, včetně jejich editace. Zdroj: autor.



(b) Knihovna Leaflet s rozšířením Leaflet.Editable. Leaflet slouží primárně k zobrazování geodat a podporuje vykreslování různých tvarů. Převzato z [7].

Obrázek 4.5: Porovnání knihoven FabricJS a Leaflet.Editable pro vytváření ohraničení objektů.

4.5 Formát exportovaných anotací

Aby bylo možné s vytvořenými anotacemi dále pracovat, nástroj musí umět anotace exportovat. Exportování se provádí ve formátu JSON ve formě staženého souboru. Tento soubor může načítat další aplikace, nebo jej může nějaký skript transformovat do jiného formátu. Anotace lze exportovat pro jednotlivá videa, nebo pro celý dataset. Formát byl částečně inspirován výstupem, který používá nástroj Darwin7 [21].

Každý export pro video obsahuje základní informace o videu. Těmi jsou: název, originální název (název v momentě přidání do nástroje) a jeho parametry: délka trvání v sekundách, šířka a výška videa v pixelech. Dále obsahuje seznam anotací, kde každá anotace má svůj název, tvar, seznam tagů a seznam pozic ve videu včetně časového otisku a souřadnic pro daný tvar. Pokud je exportování provedeno na úrovni datasetu, tak soubor obsahuje seznam videí, kde mají videa formát popsaný výše. Kompletní zápis formátu je v následujícím textu.

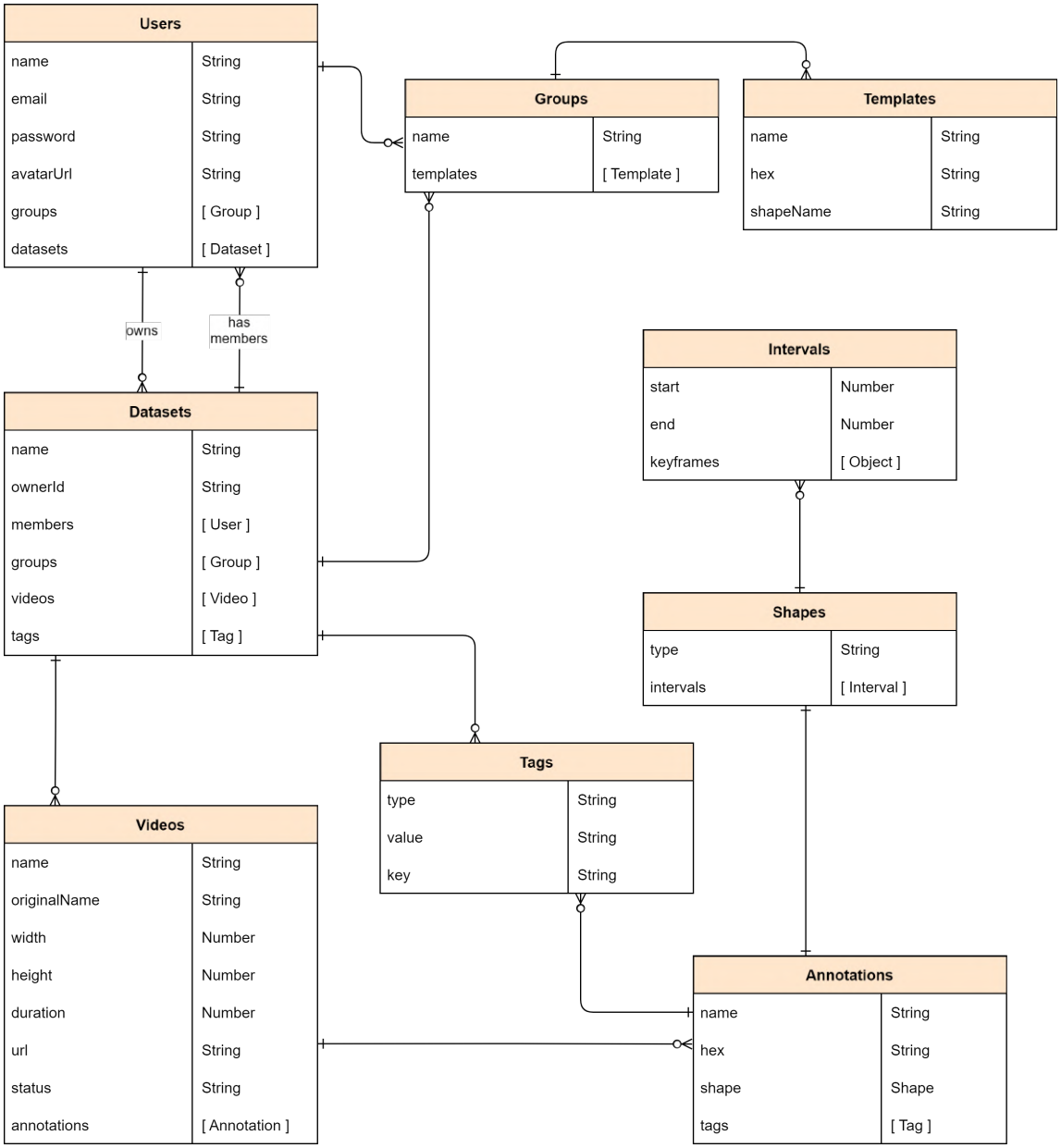
[3]Leaflet.Editable - dostupné z: https://leaflet.github.io/Leaflet.Editable
[4]Leaflet - dostupné z: https://leafletjs.com/

```
{
    "video": {
        "name": string,
        "originalName": string,
        "width": number,
        "height": number,
        "duration": number
    },
    "annotations": [ Annotation ]
}
Annotation = {
    "shape": string,
    "name": string,
    "tags": [ Tag ],
    "keyframes": [ Keyframe ]
}
Tag = {
    "key": string (optional),
    "value": string
}
Keyframe = Rectangle | Polygon | KeyPoint | Line | PolyLine
Point = {
    "x": number,
    "y": number
}
Rectangle = {
    "timestamp": number,
    "x": number,
    "y": number,
    "w": number,
    "h": number
}
Polygon = {
    "timestamp": number,
    "vertices": [ Point ]
}
KeyPoint = {
    "timestamp": number,
    "x": number,
    "y": number
}
Line = {
    "timestamp": number,
    "start": Point
    "end": Point
}
PolyLine = {
    "timestamp": number,
    "vertices": [ Point ]
}
```

4.6 Schéma databáze

Pro ukládání dat byla zvolena NoSQL databáze MongoDB. Datový model je vytvořen na základě požadavků a návrhu funkcionalit nástroje. Znázornění pomocí diagramu je na Obrázku 4.6. MongoDB, na rozdíl od klasických SQL databází, nevyžaduje striktní dodržení

počtů a typů polí v dokumentu. Zajímavá může být tabulka `Intervals` reprezentující intervaly anotací ve videu. Interval má vždy začátek a konec (vztahuje se na čas videa), dále obsahuje `keyframes`, neboli body v čase videa, kdy je definována pozice a velikost dané anotace. Každý ohraničující tvar anotace obsahuje jiné informace, a tak může zůstat definovaný abstraktně jako klasický Objekt. Některé 1:N vztahy nemusí být v MongoDB řešeny jako cizí klíče s odkazem na jiný dokument, ale mohou být přímo součástí rodičovského dokumentu v podobě sub-dokumentů a jsou uspořádány v seznamu.



Obrázek 4.6: Na obrázku je ER diagram databáze. Pro každou entitu MongoDB automaticky generuje jednoznačný identifikátor `_id` typu *String*, který není v tabulkách uveden. Zdroj: autor.

4.7 Rozhraní REST API

Nástroj pro komunikaci mezi klientskou a serverovou částí použije rozhraní REST API. Typ rozhraní REST poprvé představil Roy Fielding ve své disertační práci v roce 2000 [12]. Implementace musí splňovat určitá kritéria, jako je bezstavová komunikace, jednotné rozhraní a další, ke kterým je třeba přihlédnout už ve fázi návrhu. REST přesně nedefinuje formát pro komunikaci, takže je zvolen populární JSON.

Navržené koncové body odpovídají jednotlivým objektům nástroje. To znamená, že existují koncové body pro uživatele, datasety, videa a tak dále. Jediné veřejné koncové body jsou pro přihlášení a registraci. Ostatní obsahují `middleware`, který umožní přístup k datům pouze pro přihlášené uživatele. Každý koncový bod má typicky množné číslo pro přístup k celé kolekci objektů, nebo pro vytvoření nového objektu. Jednotné číslo vyžaduje další parametr URI, typicky unikátní identifikátor. Například koncový bod `/datasets` slouží pro správu datasetů a `/dataset/345` umožňuje provádět akce pro dataset s identifikátorem `345`. Seznam navržených koncových bodů vypadá následovně:

/users - Koncový bod pro správu uživatelů. Primárně slouží získání informací o přihlášeném uživateli a pro úpravu osobních údajů.

/user/(login|register) - Koncové body pro přihlášení a registraci. Jako jediné jsou veřejné.

/user/groups - Koncový bod pro správu uživatelské knihovny se skupinami anotačních šablon.

/datasets - Koncový bod pro správu datasetů.

/dataset/:id/tags - Koncový bod pro správu tagů datasetů.

/dataset/:id/groups - Koncový bod pro správu skupin anotačních šablon datasetu.

/dataset/:id/members - Koncový bod pro správu členů datasetu.

/dataset/:id/videos - Koncový bod pro správu videí datasetu. Taktéž slouží pro nahrání video souborů.

/dataset/:id/video/:id/annotations - Koncový bod pro správu anotací daného videa datasetu.

Kapitola 5

Implementace nástroje

Tato kapitola se věnuje implementaci navrženého webového nástroje pro vytváření anotací obrazových dat, konkrétně pro videa. Implementace vychází z návrhu předchozí kapitoly, ale obsahuje další úpravy provedené během vývoje a hlavně změny po první iteraci uživatelského testování. Podkapitoly popisují výslednou podobu uživatelského rozhraní, strukturu projektu a klíčové části aplikace.

5.1 Struktura projektu a konfigurace Dockeru

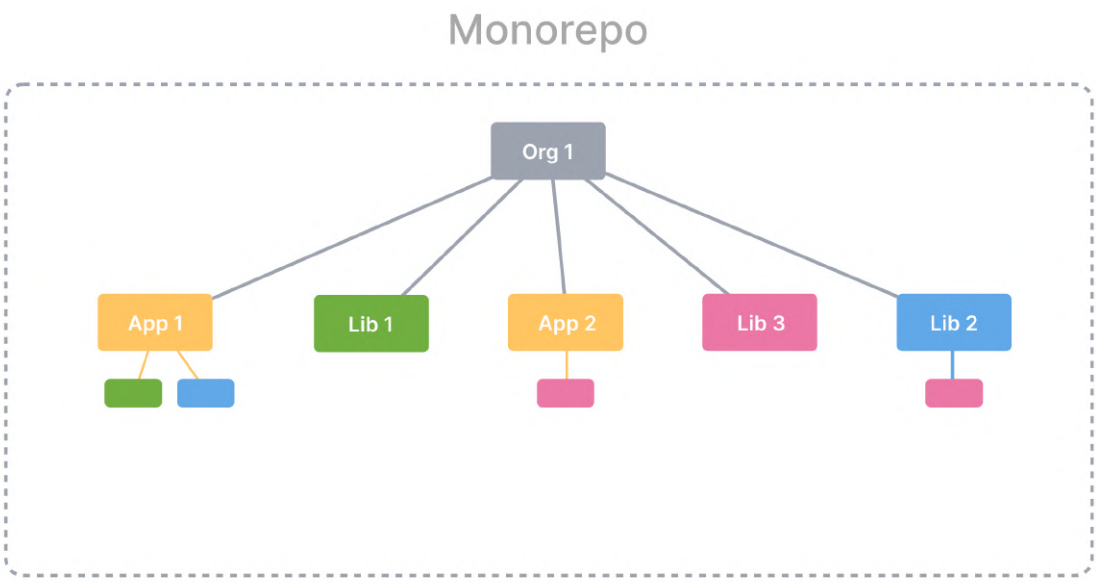
Webová aplikace je navržena jako klient-server a skládá se tedy z klientské a serverové části. Klientská část je implementována pomocí vývojové platformy Angular, napsané v jazyce TypeScript. Pro serverovou část je zvolen jazyk JavaScript, knihovna Express pro vytvoření jádra aplikace a další podpůrné knihovny. Obě části spolu budou komunikovat pomocí REST API rozhraní pomocí protokolu HTTP. Verzování obstarává nástroj Git a obě části jsou v jednom repozitáři, tzv. `monorepo`. Ilustrace pro monorepo je na Obrázku 5.1. Docker je použit k vytvoření vývojového prostředí odděleně pro obě části a databázi. Zajistí se tak, že aplikace poběží vždy na správné verzi nodejs, vývoj bude nezávislý na systému a databázi bude možné rychle smazat a vytvořit znovu včetně naplnění výchozími daty.

Klientská část aplikace

Klientská část aplikace byla vytvořena pomocí rozhraní příkazového řádku `ng`, jenž je součástí Angularu. Také slouží k vytváření funkčních částí, jako jsou moduly, komponenty a další. Dále se používá ke spouštění samotné aplikace nebo testů, jelikož Angular obsahuje knihovny pro jednotkové a end-2-end testy. Nová aplikace má předem danou jen základní strukturu, která obsahuje především konfiguraci a má jen jednu komponentu `app-root`, jejíž název napovídá, že se jedná pouze o jakýsi „kořen“ aplikace. Proto je třeba definovat vlastní strukturu, která je přehledná a dobře škálovatelná.

V kořenovém adresáři `src/` se nachází vstupní soubory aplikace jako `index.html` a nebo `main.ts`. Také tu najdeme složku pro definování proměnných prostředí aplikace `environments`, složku `assets` se styly a obrázky. Poslední a nejzajímavější je složka `app`, kde se nachází všechny funkční prvky aplikace a její struktura je navržena následovně:

```
app
├── app-root
├── core
│   └── guards
```



Obrázek 5.1: Ukázka použití jednoho repozitáře pro verzování více aplikací najednou. Převzato z [9].

```
    ├── interceptors
    └── services
└── modules
    ├── annotate
    ├── home
    ├── profile
    ├── shared
    ├── tool
    └── video-player
```

Ve složce `core` jsou služby, které jsou dostupné na globální úrovni aplikace. Nachází se zde například prvky, které rozhodují o přístupu k částem aplikace. Definovány jsou ve složce `guards` a lze je využít v kterémkoliv navigačním pravidle, například k zamezení přístupu neregistrovaného uživatele do systému. Další užitečné prvky jsou ze složky `interceptors`, které umožňují zachycení veškeré HTTP komunikace v obou směrech a danou zprávu přečíst, nebo modifikovat. To lze využít při zachytávání chyb, nebo tak vkládat autentizační token do hlavičky zprávy. Poslední jsou tzv. služby (`services`), které jsou základem frameworku pro vkládaní závislostí. Obstarávají sdílení dat mezi komponentami a poskytují rozhraní pro HTTP komunikaci. Mohou být definovány na úrovni celé aplikace, modulu, nebo komponenty. Také je možné všechny definovat na úrovni aplikace a Angular si v čase kompilace sám zjistí, na jaké úrovni, a pro které části, je třeba službu inicializovat.

V Angularu se stránky skládají z komponent. Komponenty je možné logicky uspořádat do modulů, které najdeme definované ve složce `modules`. Moduly jsou dvojího typu: s routováním, nebo bez. Routovací moduly exportují pravidla, která spojují jednotlivé URL routy a komponenty, které se mají zobrazit. Tato pravidla spravuje globální Router Angular aplikace. Moduly neslouží jen k logickému uspořádání komponent, ale umožňují i „lazy loading“. To znamená, že některé moduly stačí načíst později a ušetří se tak čas potřebný

pro zobrazení některých stránek. Například modul pro vytváření anotací a přehrávání videa není pro uživatele relevantní, dokud se do systému nepřihlásí.

Aplikace je rozdělena do několika hierarchicky uspořádaných modulů. Najdeme zde modul pro úvodní stránku s formuláři pro přihlášení a registraci(`home`). Modul pro část nástroje po přihlášení (`tool`) obstarává hlavně navigaci mezi dalšími moduly. Těmi jsou `profile` pro úpravu osobních údajů, modul `annotate` obstarává hlavní obrazovku nástroje pro vytváření anotací. Mimo jiné spravuje potřebná překryvová okna a zahrnuje samostatný modul pro přehrávání videa `video-player`. Jako poslední je speciální modul `shared`, který obsahuje komponenty, které je možné použít na více místech aplikace a ostatní moduly jej importují.

Serverová část aplikace

Serverová část aplikace je oproti klientské méně komplexní. Mezi její hlavní úlohy patří autentizace uživatelů, správa databáze a poskytování REST API rozhraní. Běží v prostředí `nodejs` a aplikace je založena pomocí správce balíčků `npm`. Server je vytvořen pomocí knihovny Express a samotná implementace obsahuje další knihovny jako Mongoose pro práci s databází, `bcrypt` pro šifrování hesel a další. V kořenovém adresáři najdeme soubor `server.js`, který aplikace spouští. Zbylá část aplikace je v adresáři `app`, který vypadá následovně:

```
app
├── models
├── seeders
├── src
└── app.js
```

Jádro serverové části tvoří soubor `app.js`, který importuje routery ze složky `src`. Dále se stará o inicializaci připojení k databázi, aktivuje užitečné middleware, jako JSON parser pro čtení těla zpráv, a definuje CORS pravidla. Koncové body a přijímané metody obsahují routery ve složce `scr`. Zde mimo ně najdeme i implementaci middleware, který kontroluje, zda příchozí požadavky obsahují autentizační token. Pro práci s databází najdeme ve složce `models` definice Mongoose modelů a ve složce `seeders` pak inicializační skript.

Konfigurace Dockeru

V kořenové složce repozitáře se nachází soubor `docker-compose.yml`, který obsahuje konfiguraci Dockeru pro vývoj aplikace. Aplikaci je tak možné začít používat, či editovat, na jakémkoliv stroji s nainstalovaným Dockerem. Spuštěním příkazu `docker-compose up -d` se spustí celkem tři kontejnery, každý pro jednu část aplikace. Kontejnery sdílí části souborového systému obsahující zdrojové kódy projektu. Proměnné prostředí kontejnerů se nachází v souboru `.env`.

5.2 Autentizace a bezpečnost

Veškerá funkcionalita nástroje zůstává nepřihlášenému uživateli skryta. Úvodní obrazovka po pravé straně obsahuje formulář určený k přihlášení. Pokud uživatel ještě vlastní účet nemá, může v horní navigační liště přepnout na registrace a dynamicky změnit obsah i účel formuláře. Formulář před odesláním požadavku na server kontroluje, zda byly zadány všechny požadované údaje a jestli mají správný formát. Pokud uživatel například

zadal nesprávný formát emailové adresy, nebo server vrátil odpověď s chybou, uživateli se zobrazí krátká chybová hláška o nesprávnosti údaje. Pokud se uživatel po předchozí relaci neodhlásil, zůstane v nástroji přihlášený a zobrazí se mu možnost vstoupit do prostředí nástroje, nebo pro odhlášení.

Heslo uživatele je před uložením do databáze nejdříve zahashováno. K tomu je použita knihovna bcrypt[1] a její funkce `hash()`. Funkce přebírá dva parametry: heslo v otevřeném textu a úroveň takzvané „soli“. Aby stejná hesla neměla po hashování stejný výstup, musí se do algoritmu přidat další znaky (v tomto případě „sůl“), aby byl výstup rozdílný. Vyšší hodnota soli zvyšuje odolnost zahashovaného proti útokům, ale je třeba delší čas pro vygenerování. Při přihlašování se kontroluje zadané heslo z formuláře proti heslu z databáze pomocí funkce `compare()`.

Autentizace

Když uživatel zadá správné přihlašovací údaje, je vygenerován JWT token [15], kterým se prokazuje při každém HTTP požadavku. Tento přístup má dva hlavní důvody. REST API je bezstavové, takže nelze spravovat právě přihlášené uživatele pomocí stavu aplikace, jako to dělají ostatní řešení. Dalším důvodem je, že token lze přidat do hlavičky každého odchozího požadavku a na serveru zkontrolovat jeho platnost a příslušnost uživateli bez potřeby cookies.

Generování JWT tokenu zajišťuje knihovna `jsonwebtoken`[2]. Obsah tokenu je standardizovaný a skládá se ze tří částí: hlavičky, těla a podpisu. Hlavička obsahuje informace o typu tokenu a jaký typ algoritmu byl pro podpis použit. Tělo tokenu má libovolný obsah. Typicky se do něj ukládají role uživatele, jeho jednoznačný identifikátor a podobné. V případě této implementace je v těle uložen identifikátor uživatele uložený v databázi, který automaticky generuje MongoDB. Aby token nebylo možné podvrhnout, třetí část obsahuje podpis. Token podepisuje server pomocí tajného klíče a symetrické kryptografie.

Po přihlášení si klientská aplikace token uloží do `localStorage` v prohlížeči. Při každém požadavku je přidán do hlavičky požadavku pomocí middleware `token-interceptor`. Když se uživatel odhlásí, token se z `localStorage` smaže. Na serverové aplikaci je implementovaný middleware `verifyToken`, který ověří, zda hlavička token obsahuje a zda je platný. Pokud je vše v pořádku, je dekódován identifikátor uživatele a jeho hodnota předána další funkci, která s požadavkem pracuje. Následující krátká ukázka kódu ilustruje, jak lze použít tento middleware jako ochranu před veřejným, nebo neoprávněným přístupem:

```
router.get('/user', verifyToken, async (req, res) => {
    const id = req.userId;
    ...
});
```

5.3 Vytváření a správa datasetů a videí

Sekce obsahuje ukázky obrazovek uživatelského rozhraní pro správu datasetů a videí. Dále popisuje změny oproti návrhu provedené během implementace a po uživatelském testování.

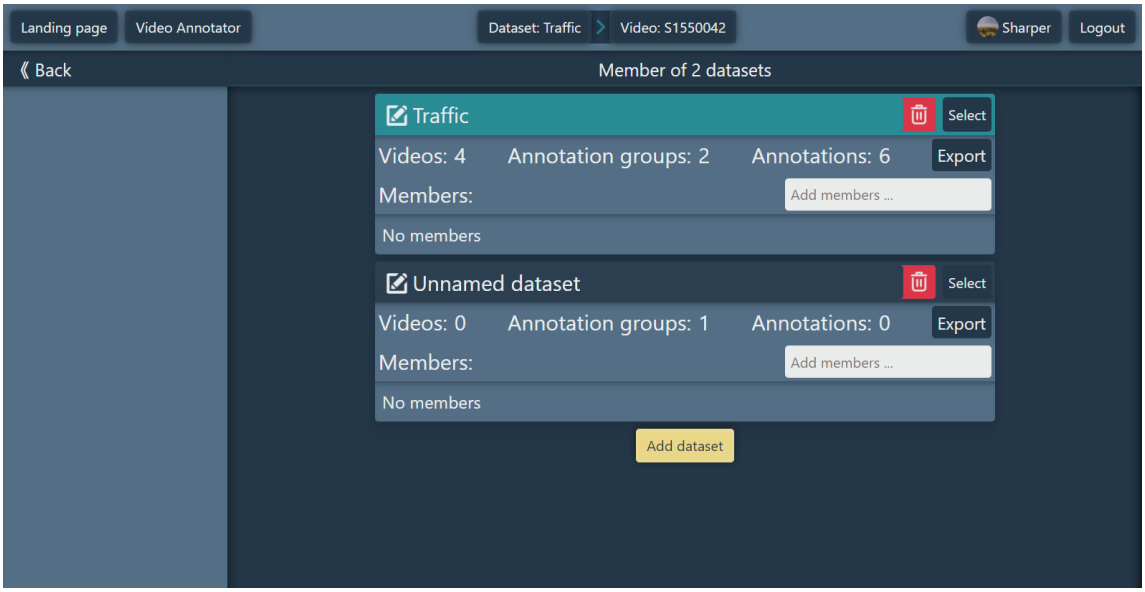
[1] bcrypt - dostupné z: https://www.npmjs.com/package/bcrypt
[2] jsonwebtoken - dostupné z: https://www.npmjs.com/package/jsonwebtoken

Správa datasetů

Pro nového uživatele se po registraci vytvoří dataset s výchozím názvem a poté ho nástroj přesměruje na stránku pro vytváření anotací, kde může přidat první video do datasetu. Pokud by chtěl upravit údaje o datasetu, nebo jeho název, klikne v horní liště nástroje na název datasetu a je přesměrován na stránku se všemi datasety. Ukázka finálního uživatelského rozhraní je na Obrázku 5.2.

Po uživatelském testování bylo provedeno několik změn. První z nich se týká samotného navigačního menu. Před název vybraného datasetu a videa byl přidán popisek, protože původní řešení nebylo dostatečně jasné. Také přibyl odkaz na hlavní anotační stránku v levé horní části menu. Většina uživatelů v první řadě hledala tuto možnost v horním menu a až poté si všimla tlačítka `Back` po levé straně.

Oproti návrhu bylo přidáno tlačítko `Select`, které dataset nejen vybere, ale vzápětí uživatele přesune na stránku s vytvářením anotací na první video. Vybrat dataset je také možné kliknutím přímo na kartu s datasetem. Aby se zabránilo nechtěnému vybrání při interakci s ostatními ovládacími prvky, například kliknutím na `Export`, volají handlery funkci `stopPropagation()` na daném eventu. K vytvoření nového datasetu slouží tlačítko `Add dataset`, které vytvoří nový dataset s výchozím názvem „Unnamed dataset“. Název jde měnit kliknutím přímo na název a je uložena na server po zadání nové hodnoty, nebo tlačítkem „Enter“. Během úpravy se změní pozadí políčka na bílou a změní se i barva ikony.



Obrázek 5.2: Obrazovka pro správu datasetů po změnách z testování aplikace. Byly přidány ovládací prvky jako tlačítko `Select` pro vybrání datasetu a přesměrování na stránku pro vytváření anotací. Zdroj: autor.

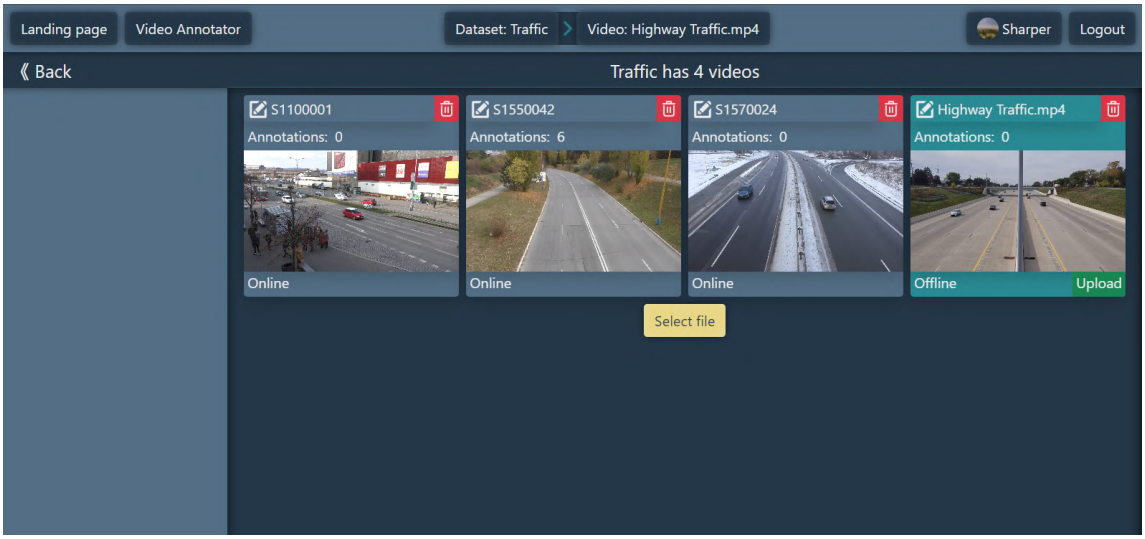
Správa videí

Uživatelé mají možnost přidávat videa k existujícím datasetům. V nástroji je proto vytvořena samostatná stránka, na kterou je možné se dostat pomocí odkazu napravo od datasetu v horním menu. Ukázka rozhraní této stránky je na Obrázku 5.3.

Seznam obsahuje karty pro jednotlivá videa v datasetu. Uživatel má možnost video vybrat kliknutím, nebo dvojklikem vybrat video a zároveň se přemístit na anotační stránku.

Funkce dvojího kliknutí byla přidána na základě zpětné vazby uživatelů. Videa lze přidat kliknutím na tlačítko `Select file`, kdy se zobrazí klasické vyskakovací okno pro daný operační systém.

Po vybrání jednoho, nebo více videí se přečtou základní informace o souborech a nahrají se do databáze. Soubory jako takové zůstávají na straně klienta, dokud neklikne na tlačítko `Upload`. Uživatel má také možnost pracovat se soubory bez nahrání na server pomocí tlačítka `Load offline`. Soubor je přečten pomocí instance `FileReader` a jeho funkce `readAsDataURL()`. Uživatel tuto funkci také může použít pro okamžité zahájení práce, zatímco se soubor nahrává na server. Všechny provedené změny, včetně vytvořených anotací, se automaticky ukládají na serveru. Jelikož prohlížeč nemá přístup k souborovému systému, musí uživatel při opětovné návštěvě nástroje znovu vybrat chybějící soubory.



Obrázek 5.3: Obrazovka pro správu videí obsahuje karty reprezentující jednotlivá video. Zobrazují základní informace jako název, počet anotací a dostupnost videa. Nástroj umožňuje práci přímo s lokálními soubory, nebo je uživatel může nahrát na server a budou tak vždy dostupná i pro ostatní členy datasetu. Zdroj: autor.

Webový video přehrávač

Nástroj pracuje s videi, které je nutné nejen v prohlížeči přehrávat, ale také s nimi manipulovat a aplikovat dočasné transformace obrazu. V rámci řešení byl vytvořen jednoduchý video přehrávač, který vyhovuje všem potřebným kritériím. Je rozdělen do několika komponent, jenž jsou uspořádány do modulu `video-player` Ukázka rozhraní přehrávače je na Obrázku 5.4.

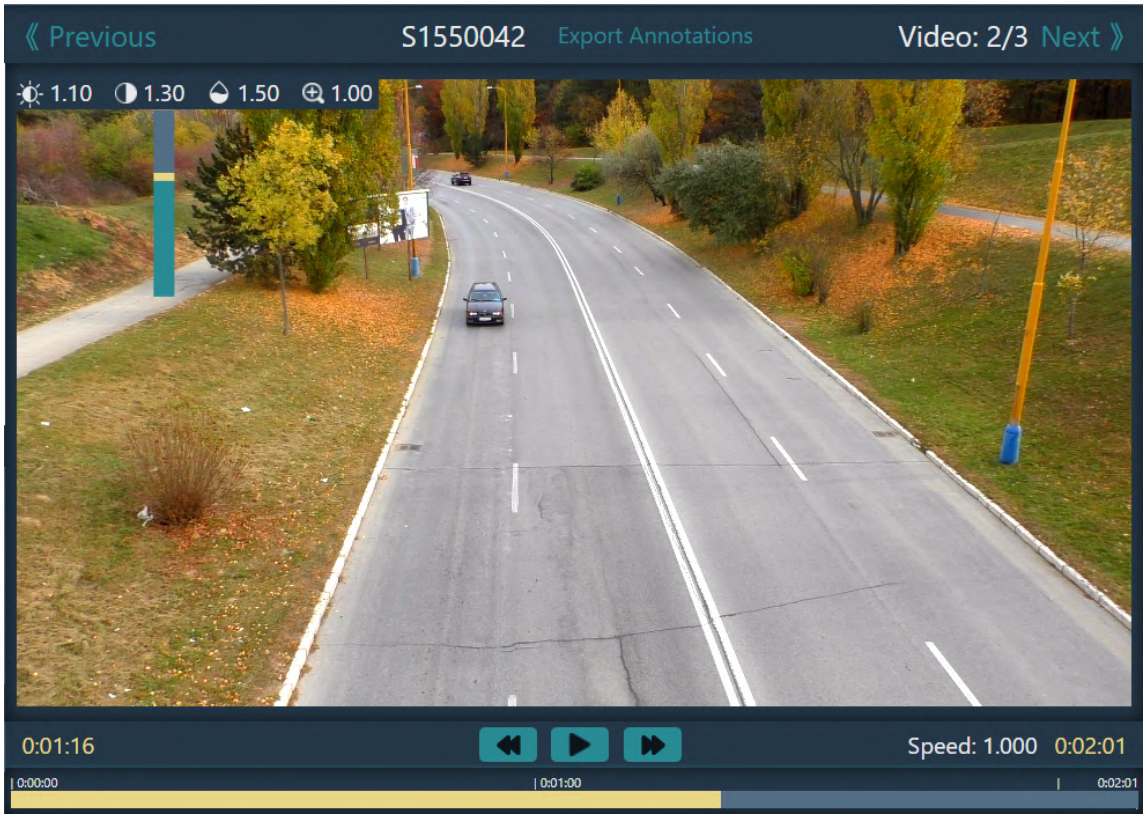
Implementace je založená na HTML elementu `<video>` a jeho aplikačním rozhraní. Element umí přehrávat videa ve formátech MP4, WEBM a OGG. Ovládací prvky videa nejsou součástí jeho elementu, protože s obrazem se nechá libovolně manipulovat a mohly by se tak dostat mimo dosah uživatele. Video je také možné ovládat pomocí klávesnice. Stiskem mezerníku se video začne přehrávat/pozastaví a pomocí šipek dopředu a dozadu skočí v čase. Aplikace listenery pro tyto eventy registruje na úrovni celého okna v momentu otevření přehrávače a odstraní je při jeho zavření. Tímto přístupem je možné video ovládat nezávisle na tom, který prvek stránky má zrovna „focus“.

Manipulovat s obrazem se dá pomocí myši. Kliknutí a posun posouvá pozici obrazu v rámečku, scrollování zase obraz přibližuje/oddaluje. K manipulaci je použita CSS property `transform` s funkcemi `translate()` pro posun a `scale()` pro přiblížení. Při kliknutí si přehrávač uloží pozici myši a při tažení počítá rozdíl v souřadnicovém systému. Ten přičítá k aktuálnímu posunu až do momentu uvolnění kliknutí.

V horní části obrazovky se nacházejí ovládací prvky pro transformaci obrazu. Upravovat jde jas, kontrast a nasycení barev obrazu. Prvky doplňuje i možnost přiblížení. Původní návrh obsahovat tlačítka +/- pro úpravu hodnot. Po testování byl tento způsob nahrazen posuvníkem, který se zobrazí při najetí kurzoru na hodnotu. Měnit hodnoty lze i pomocí scrollováním na dané hodnotě a kliknutím se vrátí na výchozí hodnotu. K transformaci slouží CSS property `filter` a funkce `brightness()`, `contrast()` a `saturate()`. Stejný styl ovládání, včetně posuvníku, je použit pro úpravu rychlosti přehrávání v pravé dolní části obrazovky.

Základní velikost videa je plně responzivní, takže se mění s velikostí okna a dostupného okolního místa. Velikost nelze nastavit na automatickou, protože obraz videa překrývá 1:1 HTML `<canvas>` pro vykreslení anotací a ten musí mít přesně danou velikost. Při eventu okna `resize` se každých několik desítek milisekund počítá velikost dostupného místa na obrazovce a přepočítává se velikost videa na základě toho, zda je obraz limitovaný šířkou, nebo výškou. K limitování, jak často se tento výpočet provádí, je použita metoda `throttleTime()` z knihovny RxJS[3].

[3] RxJS - dostupné z: https://rxjs.dev/



Obrázek 5.4: Ukázka vlastního webového přehrávače videa postaveného na HTML <video> elementu. Podporuje ovládání pomocí klávesnice a umožňuje manipulaci z obrazem. Kromě přiblížení a posunutí umí také změnit jas, kontrast a nasycení barev obrazu. Zdroj: autor.

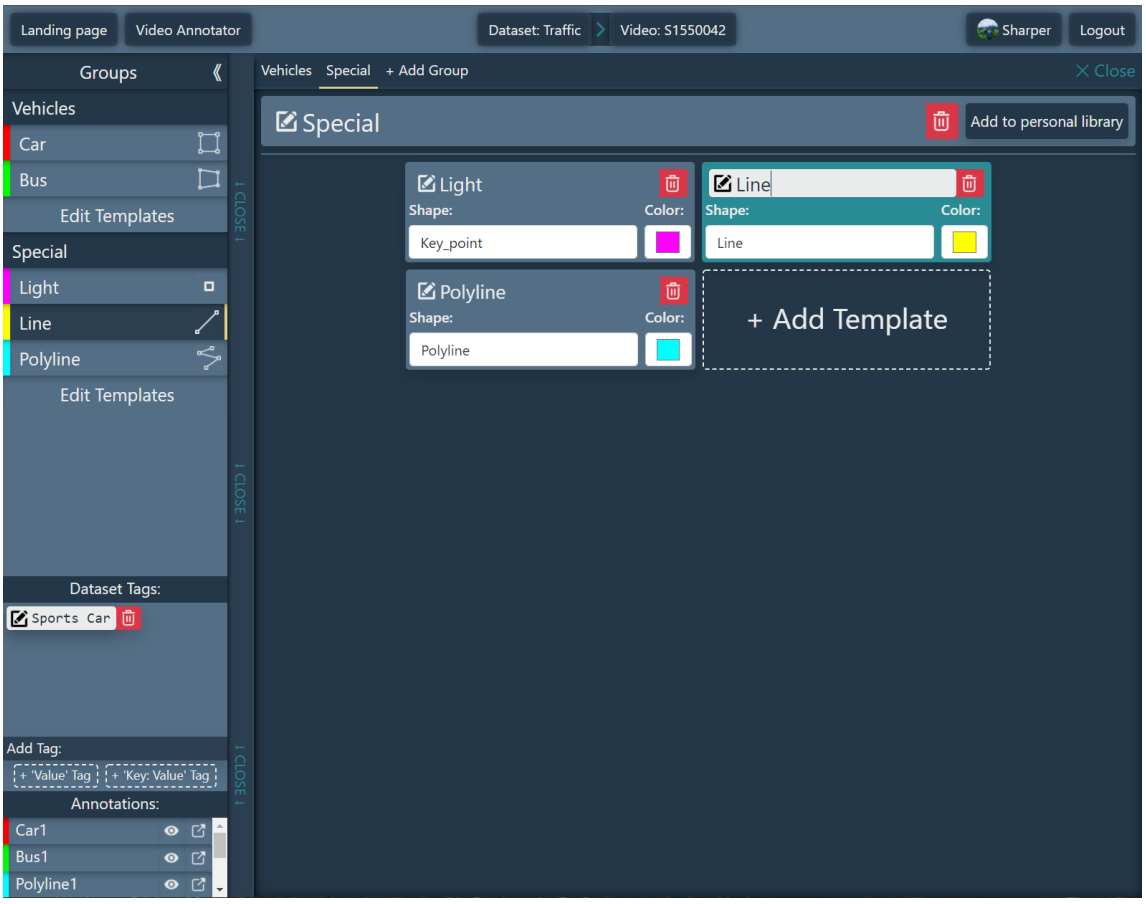
5.4 Šablony anotací a knihovna uživatele

Nástroj modularitu anotací řeší několika způsoby. První z nich staví na anotačních šablonách, které obsahují pouze základní údaje. Podle nic uživatel vytváří konkrétní anotace, které fungují jako samostatné objekty a lze je libovolně upravovat. Také k nim lze přiřadit dodatečné informace v podobě tagů. Šablony tvoří skupiny, které jsou součástí datasetu a jsou sdíleny mezi členy. Druhý způsob řešení modularity umožňuje skupiny datasetu zkopírovat do uživatelské knihovny a naopak. Tím je možné skupiny znovu používat. Uživatel má možnost svoji knihovnu nezávisle upravovat. Ukázka vyskakovacího okna je na Obrázku 5.5.

Okno překrývá video a časovou osu s anotacemi. Otevřít lze několika způsoby, jako kliknutím na `Groups`, kliknutím na danou skupinu nebo dvojklikem na šablonu, která se nachází po levé straně. Oproti návrhu byl text `+ Add template` nahrazen `Edit Templates` a změnila se i funkce. Původní verze, kromě otevření okna, rovnou vytvořila novou šablonu a to uživatele zmátlo. Nyní jen otevře okno. Také byla přidána možnost okno zavřít pomocí tlačítka `Esc`. Také bylo přidáno vybírání šablony pomocí kláves 1-9 počítáno od shora napříč skupinami. Vybraná šablona je vizuálně odlišena od ostatních.

Šablona definuje tvar anotace, její barvu a název. Konkrétní anotace hodnoty dědí a k názvu je přidána pořadová číslovka. Po vytvoření není šablona s anotací dále spojena a tak jde samostatně upravovat. V horní části obrazovky se nachází tlačítko `+ Add Group`, které umožňuje přidat prázdnou skupinu, nebo skupinu z uživatelské knihovny. Ze seznamu

jsou vyřazeny dříve přidané skupiny, pokud se shodují svým názvem a obsahem. Knihovna vypadá velmi podobně. Nachází se v profilu uživatele a poskytuje stejnou úroveň editování.



Obrázek 5.5: Vyskakovací okno se skupinami obsahující šablony. Vybraná šablona se pozadím karty odlišuje od ostatních. Na obrázku je vidět efekt při změně jména šablony. Stejné jádro okna je použito i v uživatelské knihovně. Zdroj: autor.

5.5 Vytváření a správa anotací

Samotným jádrem aplikace je vytváření anotací ve videu. Hlavní strana se skládá z několika částí. Největší část pokrývá video přehrávač popsaný v sekci 5.3. Dále zde najdeme skupiny anotačních šablon a v dolní části seznam již vytvořených anotací. Vedle seznamu se nachází časová osa zobrazující dobu trvání anotace ve videu. Ukázka tohoto rozhraní je na obrázku 5.6.

Přehrávač překrývá HTML `<canvas>`. Kopíruje transformace aplikované na přehrávač a také responzivně mění svoji výchozí velikost vzhledem k velikosti okna. Kreslení tvarů a manipulaci s objekty řeší vlastní malá knihovna, popsaná v sekci dále v textu. Tvar anotace udává zvolená šablona a vytváří se typicky tažením myši, nebo klikáním do obrazu. Tvary typu polygon a tah lze také dokončit stisknutím klávesy `Enter`. Vytvořená anotace má ve videu základní dobu trvání 10 s.

Délku trvání je možné upravit tažením koncových bodů intervalu, který je vykreslen v dolní části obrazu. Tažení mění i pozici v aktuálně přehrávaném videu, aby bylo možné

dobře určit, kdy sledovaný objekt opustí obrazovku, nebo se objeví. Po testování byla přidána akce, kdy kliknutím v kombinaci s klávesou `Ctrl` na kraj intervalu dojde k zarovnání na poslední/první keyframe. Také bylo přidáno posunutí času videa kliknutím na libovolnou část intervalu.

Keyframe drží informaci o pozici anotace v obraze pro daný čas ve videu na dvě desetinná místa. Vytvoření anotace vytvoří první keyframe. Konec i začátek intervalu může mít explicitní keyframe, nebo je dá implicitně podle posledního/prvního známého keyframe. Jelikož video má spojitý čas a nástroj s ním nepracuje jen jako se sekvencí snímků, mezi dvojicí keyframe se provádí lineární interpolace. Pokud uživatel není spokojený s pozicí anotace, může ji přesunout a upravit. Tím se vytvoří nový keyframe. Jejich pozice se zobrazují v časové ose anotace jako značky ve tvaru diamantu a kliknutím na ně se posune čas videa pro jejich úpravu.

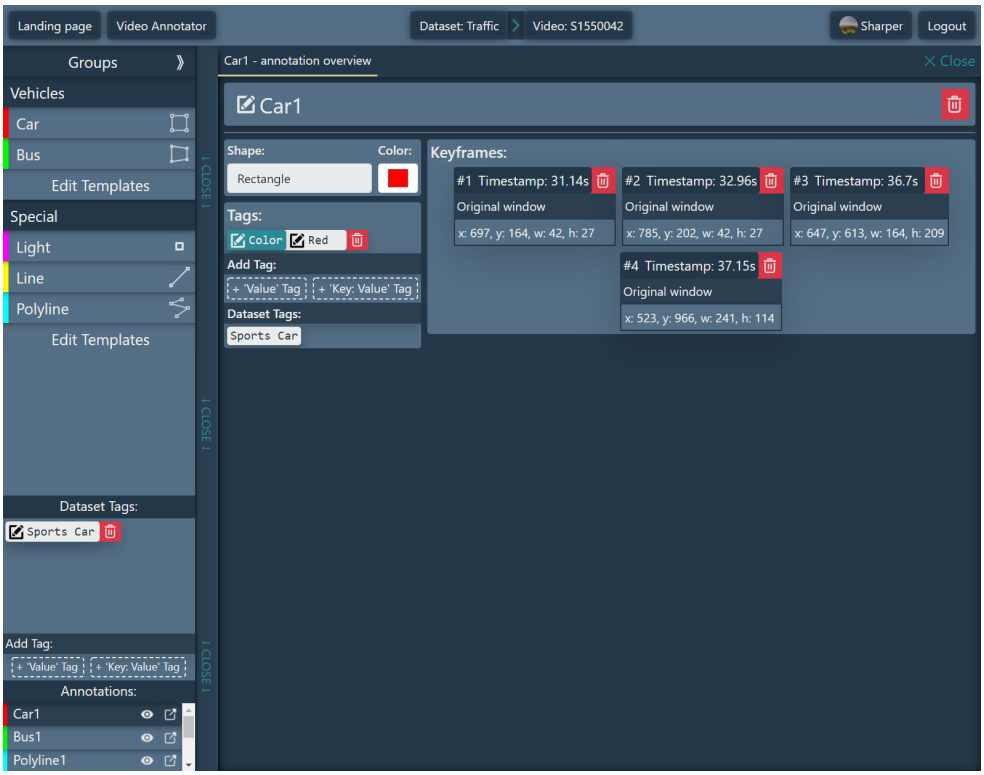
Obrázek 5.6: Hlavní obrazovka pro vytváření anotací. Tvar anotace udává zvolená šablona ze seznamu v levé části. Anotace se vytváří tahem (čtverec) nebo klikáním myši (polygon). Dolní část obsahuje seznam vytvořených anotací a časovou osu trvání ve videu. Zdroj: autor.

Správa anotací

K úpravě informací o anotaci slouží dvě okna. Jedno malé, které je na Obrázku 5.6, a druhé pro pokročilejší úpravy je na Obrázku 5.7. Na úrovni datasetu můžou být definované různé tagy, které uživatel chce libovolně přiřazovat k anotacím.

Malé vyskakovací okno se otevírá dvojím kliknutím do prostoru anotace. Aby nedošlo k nechtěnému posunutí, nástroj s anotací nehýbe, dokud se jeho pozice od té původní nezmění alespoň o několik pixelů. Malé okno slouží k přiřazování tagů buď z datasetu, nebo uživatel vytvoří tag úplně nový. Také lze anotaci dočasně skrýt, nebo otevřít větší editační okno. Po testování byly přidány klávesové zkratky pro mazání (tl. `Delete`) a zavření okna (tl. Esc).

Větší vyskakovací okno umožňuje stejné úpravy, ale navíc umí měnit název anotace a jeho barvu. Tvar anotace je pevně dán a nelze změnit. Rozhraní zobrazuje přehlednější seznam tagů a také seznam všech keyframe. Keyframe zobrazuje informace o pozici anotace v obrazu a čas ve videu. Hlavním účelem je možnost keyframe smazat.



Obrázek 5.7: Správa anotace pomocí velkého vyskakovacího okna. K anotaci jdou přiřadit tagy definované na úrovni datasetu, nebo uživatel vytvoří zcela nové. Smazat keyframe anotace lze v seznamu na pravé straně okna. Zdroj: autor.

Knihovna pro vykreslení a interakci s anotacemi

Aby byla práce s nástrojem rychlá, knihovna pro vykreslení tvarů musí splňovat určitá kritéria. Měla by umět tvary vytvářet pomocí tahů a kliknutí myší a průběh vhodně prezentovat uživateli. Tvary musí nejen umět vytvořit, ale také upravovat jejich pozici a tvar. Vykreslení musí být dostatečně rychlé, jelikož pozice anotací se v přehrávaném videu mění v řádu desítek za vteřinu.

Žádná z knihoven, se kterými bylo experimentováno, nesplnila všechna kritéria. Programová nadstavba, která jejich nedostatky doplňovala, byla ve výsledku natolik objemná, že stačilo jen přidat vykreslování tvarů pomocí HTML elementu `<canvas>` a detekci kurzoru v anotacích, aby vzniklo vlastní řešení.

Knihovna používá pro kreslení tvarů 2D context z `<canvas>` elementu. Nástroj nejprve přepočítá pozice všech anotací ve videu. Pokud pro daný čas existuje keyframe, je použit ten. Jinak se aktuální pozice dopočítává pomocí lineární interpolace. Poté se zavolá funkce pro vykreslení anotací. Ta vždy začíná smazáním předchozího plátna funkcí `clearRect()` a poté vykreslí všechny anotace. Vetšinou k tomu stačí funkce pro vytváření čtverců a přímek. Také se vykresluje aktuálně vytvářený tvar, o kterém si udržuje informace stav nástroje.

Implementaci bylo třeba doplnit o detekci kurzoru v anotaci. Nástroj podporuje celkem pět tvarů, ale některé části kódu lze použít pro detekci opětovně. Například rovnice pro výpočet vzdálenosti bodu (v tomto případě kurzoru) od přímky je možné použít i pro výpočet vzdálenosti od lomené čáry, iterováním výpočtu vždy mezi dvěma následujícími body. Mezi netriviální výpočty patřil jen problém bodu v polygonu. Pro ten byl použit Ray-casting algoritmus [14].

Kromě detekce kurzoru v anotaci bylo třeba i detekovat, jestli se kurzor nachází nedaleko některého vrcholu anotace. Určuje se tak, zda se hýbe celou anotací, nebo objekt mění tvar posunutím jednoho z vrcholů, nebo stran. Aby detekce nebyla zcela přesná na jednotlivé pixely, počítá se s určitým prahem tolerance v okolí vrcholů a stran.

Správa stavu nástroje

Při vytváření anotací a manipulaci s nimi hraje velkou roli stav aplikace. Vzhled, možné akce a dostupné klávesové zkratky se mění podle toho, zda je vybraná anotační šablona, nebo je vybrána konkrétní anotace a nebo jestli uživatel právě interaguje s kanvasem pro kreslení anotací.

Pro správu stavu aplikace jsou použity algebraické datové typy z funkcionálních programovacích jazyků, které TypeScript podporuje. Výhodou je, že se aplikace nemůže dostat do nesprávného stavu a vždy má k dispozici požadované hodnoty uložené přímo ve stavu. Například kanvas pro kreslení anotací se může dostat do stavů: posouvání videa, posouvání anotace, změna velikosti anotace a vytváření anotace. Ukázka kódu pro správu stavu kanvasu doplněná o výchozí stav:

```
type VideoMoving = { state: 'VIDEO_MOVING', startX: number, startY: number, ... };
type Moving = { state: 'MOVING', ... };
type Resizing = { state: 'RESIZING', ... };
type Creating = { state: 'CREATING', ... };
type Default = { state: 'DEFAULT' };

type CanvasState = Default | Moving | Resizing | Creating | VideoMoving;
```

Výpis 5.1: Každý ze stavů musí mít definovanou `state` property.

Další výhodou je, že lze použít takzvanou „kontrolu vyčerpání“. Jedná se o jednoduchou funkci s jedním parametrem typu `never`. Když část kódu reaguje na stav aplikace pomocí `switch(state)`, je tato funkce použita ve výchozí větvi s parametrem `state`. Pokud v kódu chybí jedna z možností, překladač odmítne kód zpracovat a upozorní tak na chybu v kódu. Bez kontroly vyčerpání by se chyba projevila až za běhu aplikace. Dají se tak kontrolovat kritické části, jako je resetování výchozího stavu.

Kapitola 6

Uživatelské a automatizované testování

Kapitola popisuje knihovny a nástroje použité k testování. Dále popisuje, jak byla testována klientská a serverová část aplikace v průběhu implementace. Krátce je zmíněn způsob automatizování testů. Druhá část kapitoly se zabývá uživatelským testováním. Obsahuje průběh testování a jak výsledky ovlivnily následující iteraci implementace.

6.1 Průběžné testování aplikace

Klientská i serverová část aplikace byla průběžně testována během implementace. K testování byly použity nástroje poskytované platformou Angular, ale i externí programy. Některé testovací procedury jsou zautomatizované, jiné musely být prováděny manuálně. Vetšinou se jednalo o funkční testy, zajišťující konzistenci dříve vytvořených částí aplikace v průběhu vývoje.

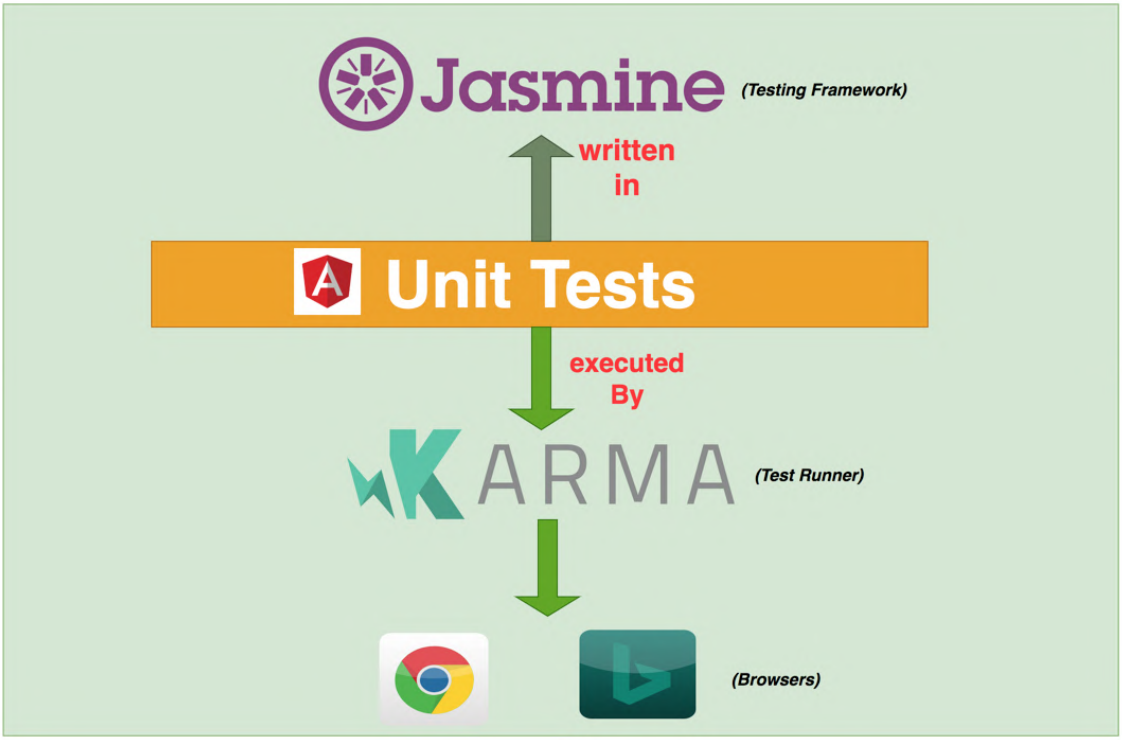
Testování klientské části

Součástí platformy Angular je knihovna pro testování Jasmine[1]. Jasmine najdeme i mimo prostředí Angularu. Její testovací procedury lze obecně použít pro jakýkoliv JavaScriptový projekt, včetně serverových aplikací. Umožňuje vytvářet jednotkové testy a prostředí, ve kterém bude objekt testován. Samotné spouštění testů obstarává knihovna Karma[2]. Karma je taktéž součástí Angularu a stará se o režii spouštění jednotkových testů a jejich vyhodnocení. Testování se spouští pomocí rozhraní příkazového řádku Angularu příkazem `ng test`. Ilustrace testovacího ekosystému je na Obrázku 6.1.

Aby bylo možné testovat, nejdříve je třeba testy definovat. Angular automaticky generuje soubory s příponou `*.spec.ts` při generování komponent pomocí příkazu `ng generate`. Generuje je pro všechny funkční části aplikace jako jsou komponenty, služby, guardi, a další. Testování je zaměřeno hlavně na komponenty a služby. Soubor s testem vytváří danou komponentu a také okolí, ve kterém je test prováděn. Zmíněné okolí je důležité zejména pro komponenty, které zobrazují informace, které jim předává jejich rodičovský element. Některé komponenty také emitují eventy, na které rodičovský element reaguje.

[1] Jasmine - dostupné z: https://jasmine.github.io/
[2] Karma - dostupné z: https://karma-runner.github.io/latest/index.html

Test každé komponenty začíná jejím vytvořením a vykreslením v prohlížeči. Poté se typicky mění vnitřní stav komponenty a sleduje se, jak na změnu komponenta reaguje. Kontroluje se, zda zobrazuje správně data a nezpůsobuje chyby. Toho lze využít při destruktivním testování, kdy jsou komponentě předány buď prázdné parametry, nebo parametry jiného typu, než očekává.



Obrázek 6.1: Platforma Angular obsahuje knihovny pro testování Jasmine a Karma. Jasmine slouží k definování jednotkových testů a scénářů testování. Testy spouští a vyhodnocuje Karma v prostředí webového prohlížeče. Převzato z [19].

Testování serverové části

Serverová část aplikace má menší rozsah implementace, a tak byla zvolena metoda ručního testování. Testovány byly především koncové body rozhraní REST API. K testování byl použit populární nástroj Postman[3].

Každý významný objekt databáze má svoje koncové body rozhraní, které umožňují provádět operace vytvoření, čtení, mazání a aktualizování. Mezi tyto objekty patří uživatelé, datasety, videa, anotace a další. Po vytvoření každého nového objektu a implementování obsluhy koncových bodů, byla vytvořena sada testovacích HTTP požadavků v prostředí aplikace Postman.

Jednotlivé testy ze sady poté byly ručně spouštěny. Kontrolovalo se, zda obsah a kód zprávy odpovídá očekávaným hodnotám. Také bylo třeba dohlédnout, zda se provedly i očekávané změny v databázi. V případě odstranění videa bylo také třeba dodatečně zkontrolovat, zda se na serveru smaže příslušný video soubor. Kromě správného chování se také testovaly různé chybové stavy, jako nevyplněné hodnoty v těle požadavku. V tom případě

[3]Postman - dostupné z: https://www.postman.com/

odpověď musela obsahovat příslušný kód chyby a vhodnou zprávu, která bude uživateli zobrazena.

Automatizace testování

Testování klientské části se provádí pouze spuštěním příkazu `ng test`. Proto bylo využito nástroje pro automatizaci testování GitLab CI/CD. GitLab[4] je webová služba poskytující vzdálené repozitáře, které slouží k verzování softwaru pomocí nástroje Git. Kromě verzování poskytuje i různé užitečné funkce.

Funkce CI/CD lze konfigurovat pomocí souboru `.gitlab-ci.yml`. Zde jsou definovány jednotlivé fáze testování a jejich procedury. Testy se spouští automaticky při každém nahrání nové verze aplikace do repozitáře. Pokud některý z testů skončí chybou, pošle GitLab správci repozitáře email s upozorněním. Při vývoji se tak ušetří čas, který by jinak zabralo manuálním spouštění.

V projektu jsou definovány dvě fáze testování. V první z nich se provede překlad a build aplikace. Akce se provádí paralelně a nezávisle na sobě. Překladem aplikace se rozumí spuštění emulátoru webového serveru, určeného pro fázi vývoje, pomocí příkazu `ng serve`. Samotný build kompiluje aplikaci ve verzi pro nasazení do produkce. Zahrnuje pokročilou analýzu kódu a optimalizace. Dokáže tak odhalit chyby, které by překlad nenašel. Pokud obě části proběhnou v pořádku, následně se provede spuštění unit testů.

6.2 Uživatelské testování

Po první iteraci vývoje nástroje proběhlo uživatelské testování. Testování se zúčastnili členové akademických kruhů fakulty, kteří se pohybují v oblasti umělé inteligence. Až na výjimky měli všichni předchozí zkušenosti s nástroji pro vytváření anotací v obrazových datech. Scénář testování byl inspirován knihou *Rocket surgery made easy* [17].

Průběh testování

Pro účely testování byla aplikace spuštěna na laptopu pomocí nástroje Docker. Během testování se ale používal externí monitor, klávesnice a myš. Cílem bylo co nejlépe vystihnout běžné prostředí, ve kterém bude aplikace používána. Na každý běh testování byl vymezen časový úsek 20 minut a celkem se testování zúčastnilo šest lidí.

Na začátku každého běhu byl uživatel tázán na své zkušenosti s anotačními nástroji a zda je mu proces vytváření anotací v obrazových datech znám. Až na jednoho byli všichni z dotázaných zkušení uživatelé. Dále bylo uživateli vysvětleno, že cílem testování je samotná aplikace a že on sám nemůže udělat nic špatně. Také byl požádán, aby během užívání nástroje přemýšlel nahlas a říkal, na co se dívá a co se snaží provést za akci.

Poté mu byl představen nástroj, jeho uživatelské rozhraní a k čemu slouží. Nástroj obsahoval dva předem připravené datasety, které byly automaticky obnoveny před každým během. Jeden obsahoval videa z provozu na silnici a druhý záznamy palubní kamery. Uživatelé se postupně s nástrojem seznamovali, a poté se pustili do vytváření anotací, jak byli sami zvyklí. Vyzkoušeli si všechny funkcionality od přidání datasetu a videí, až po vytváření tagů pro anotace.

[4] GitLab - dostupné z: https://gitlab.com/

Výsledky testování

V průběhu každého běhu byly zaznamenávány poznámky. Pokaždé, když uživatel tápal, jak má v akci pokračovat, byly zapsány kroky, které ho do situace dostaly a jak se ze situace snažil dostat. Občas se stávalo, že se uživatel pokoušel interagovat s prvky na obrazovce, které podle návrhu sloužily pouze k zobrazování informací. Poznámky dále obsahovaly připomínky uživatele, jaké funkce by v nástroji ocenil. Na závěr byl uživatel dotázán, jak se mu s nástrojem pracovalo a jaké činnosti mu dělaly největší potíže.

Z poznámek byl vypracován seznam nedostatků a možných vylepšení. Nedostatky se týkaly hlavně slabé podpory ovládání klávesnicí, například chybělo zavírání vyskakovacích oken pomocí `Ecs`, dokončení polygonu pomocí `Enter` a další. Projevily se nedostatečné popisky v horním navigačním menu a některé nevhodné popisky použité na tlačítkách rozhraní. Potíže také působilo vytváření anotací, kdy rozdělaná anotace nešla zrušit, ale musela být nejdříve dokončena a poté smazána.

Valnou většinu nedostatků se podařilo včas do práce zapracovat a některé z nich jsou popsány v kapitole 5. Ze seznamu možných vylepšení je zapracována jen část připomínek a to podle náročnosti na implementaci. Testování přineslo znatelné vylepšení přívětivosti a přehlednosti uživatelského rozhraní. Ovládání nástroje se povedlo rozšířit o několik nových způsobů.

Kapitola 7

Závěr

V rámci řešení této práce byla nastudována problematika vytváření anotací v obrazových datech. Dále bylo třeba nastudovat a analyzovat moderní technologie pro tvorbu webových stránek a tvorbu uživatelského rozhraní. Následně byla navržena webová aplikace pro přehrávání video souborů a vytváření anotací různých tvarů. Návrh stavěl na definovaných požadavcích a rozboru existujících řešení.

Výsledkem práce je nový webový nástroj postavený na architektuře klient-server. Klientská část poskytuje moderní uživatelské rozhraní, které bylo implementováno pomocí populární vývojové platformy Angular. Velká část implementace serverové části zahrnuje knihovnu Express. Další knihovny byly použity zejména k zajištění bezpečnosti systému. Obě části používají pro vzájemnou komunikaci rozhraní REST API.

Nástroj umožňuje snadno vytvářet datasety a přidávat videa. Volitelné sdílení datasetů s dalšími lidmi přináší možnost spolupráce při vytváření anotací. Dále nabízí modularitu anotací díky anotačním šablonám. Šablony tvoří skupiny, které jsou dostupné pro všechny členy datasetu. Uživatelé si je mohou přidat do osobní knihovny a poté přenést do jiného datasetu.

K přehrávání videí slouží jednoduchý přehrávač, který umožňuje manipulaci s obrazem a dokáže dočasně upravit jas, kontrast a saturaci. Součástí řešení je i malá knihovna pro vytváření a editaci základních tvarů anotací. Chybějící pozice anotací ve videu nástroj dopočítává pomocí interpolace. Výsledné anotace umí nástroj exportovat ve formátu JSON.

První kolo uživatelského testování přineslo mnoho užitečných poznatků. Připomínky se týkaly uživatelské přívětivosti a hlavně rozšíření ovládání nástroje pomocí zkratek a nových ovládacích prvků. Většinu z nich se podařilo do práce zapracovat a vylepšit tak interakci s rozhraním.

Další vývoj bude určitě zahrnovat vylepšení algoritmu pro interpolaci anotací. Budoucí vývoj by také mohl přinést modularitu videí, aby bylo možné je sdílet mezi datasety. Ušetřilo by se tak místo na serveru a přeskočil by se i krok nahrávání souborů.

Literatura

[1] *Angular* [online]. Google, 2022 [cit. 15. 1. 2022]. Dostupné z: https://angular.io/.

[2] Create API using Express JS and Sequelize. *Dev IT Journal* [online]. Rutvik Thakkar, 2022 [cit. 15. 1. 2022]. Dostupné z: https://www.blog.devitpl.com/sequelize/.

[3] *Docker* [online]. Docker Inc., 2022 [cit. 15. 1. 2022]. Dostupné z: https://www.docker.com/.

[4] Get Started. *Docker Docs* [online]. Docker Inc., 2022 [cit. 15. 1. 2022]. Dostupné z: https://docs.docker.com/get-started/.

[5] Fast, unopinionated, minimalist web framework for Node.js. *Express* [online]. 2022 [cit. 15. 1. 2022]. Dostupné z: https://expressjs.com/.

[6] *Express Middlewares, Demystified* [online]. Viral Shah, 2022 [cit. 15. 1. 2022]. Dostupné z: https://medium.com/@viral_shah/express-middlewares-demystified-f0c2c37ea6a1.

[7] *Leaflet.Editable* [online]. GitHub, Inc., 2022 [cit. 26. 1. 2022]. Dostupné z: https://github.com/Leaflet/Leaflet.Editable.

[8] Guides. *Mongoose ODM* [online]. Mongoose, 2022 [cit. 10. 4. 2022]. Dostupné z: https://mongoosejs.com/docs/guides.html.

[9] *Monorepo.tools* [online]. 2022 [cit. 15. 4. 2022]. Dostupné z: https://monorepo.tools/.

[10] About npm. *Npm Documentation* [online]. 2022 [cit. 15. 1. 2022]. Dostupné z: https://docs.npmjs.com/about-npm/.

[11] *TypeScript* [online]. Microsoft, 2022 [cit. 15. 1. 2022]. Dostupné z: https://www.typescriptlang.org/.

[12] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Irvine, 2000. Disertační práce. University of California. Dostupné z: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[13] Gorton, I. *Essential software architecture*. 1. vyd. Berlin: Springer, 2006. ISBN 3-540-28713-2.

[14] Hormann, K. a Agathos, A. The point in polygon problem for arbitrary polygons. *Computational Geometry*. 2001, sv. 20, č. 3, s. 131–144. DOI: https://doi.org/10.1016/S0925-7721(01)00012-8. ISSN 0925-7721. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0925772101000128.

[15] Jones, M., Bradley, J. a Sakimura, N. *JSON Web Token (JWT)* [RFC 7519]. RFC Editor, květen 2015 [cit. 15.4.2022]. DOI: 10.17487/RFC7519. Dostupné z: https://www.rfc-editor.org/info/rfc7519.

[16] Krishnan, H., Elayidom, M. a Santhanakrishnan, T. MongoDB – a comparison with NoSQL databases. *International Journal of Scientific and Engineering Research*. Květen 2016, sv. 7, s. 1035–1037.

[17] Krug, S. *Rocket surgery made easy: the do-it-yourself guide to finding and fixing usability problems*. 1. vyd. Berkeley, CA: New riders, 2010. ISBN 978-0321657299.

[18] Mozilla. JavaScript. *MDN web docs* [online]. 2022 [cit. 15. 1. 2022]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[19] Prasad, A. *Karma-Jasmine* [online]. 2022 [cit. 17. 4. 2022]. Dostupné z: https://akshayprasad199503.medium.com/karma-jasmine-64ed403c5491.

[20] Roboflow. *Give your software the sense of sight* [online]. 2022 [cit. 17. 1. 2022]. Dostupné z: https://roboflow.com/.

[21] V7. *Image & Video Annotation* [online]. 2022 [cit. 15. 1. 2022]. Dostupné z: https://www.v7labs.com/annotation.

Příloha A

Obsah přiloženého paměťového média

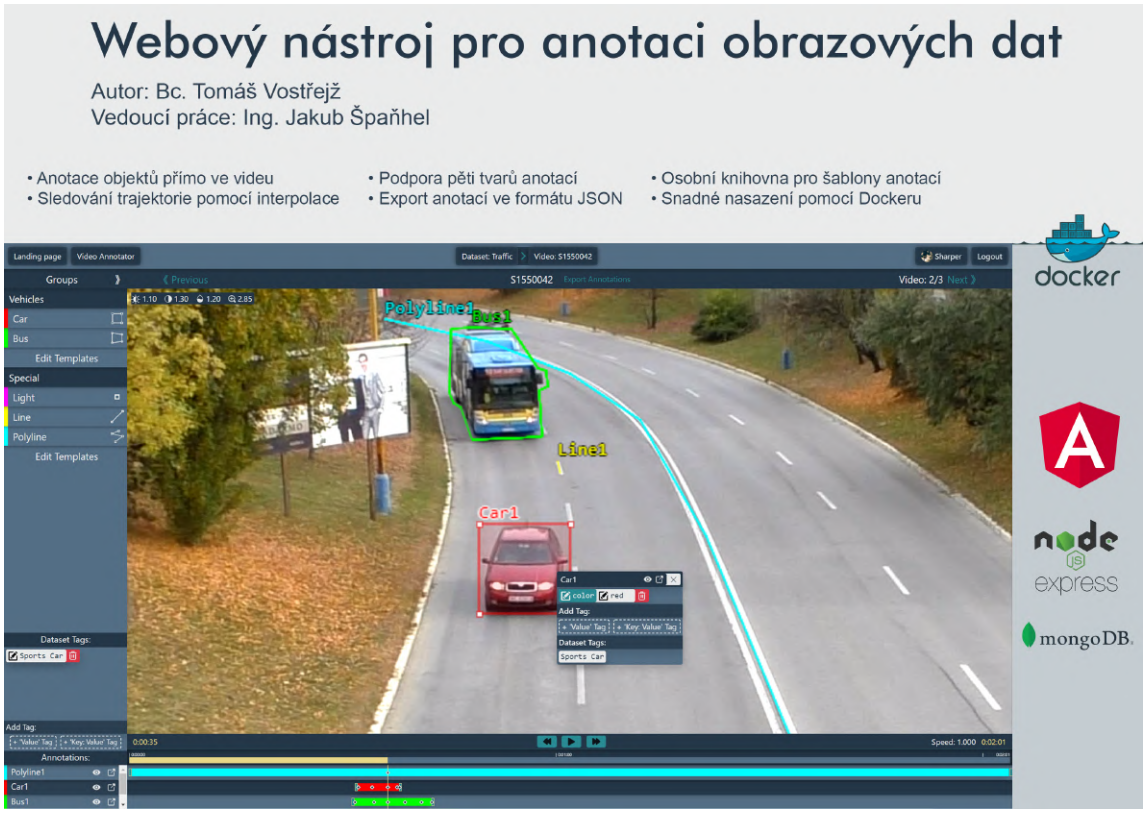
thesis/ - Zdrojové soubory textu práce s včetně výsledného PDF.

videoannotator/ - Složka se zdrojovými kódy obou částí aplikace.

docs/ - Složka s plakátem a videem.

Příloha B

Plakát



Příloha C

Manuál

Pro spuštění obou částí aplikace musí být na systému nainstalován Node.js s aktuální verzí alespoň 16.4.

Klientská aplikace

Nachází se ve složce `frontend` a po spuštění je dostupná na portu 4200. Zde je třeba doinstalovat knihovny pomocí příkazu:

```
npm install
```

Spuštění vyžaduje Angular CLI. Pro nainstalování globálně stačí příkaz:

```
npm install -g @angular/cli
```

Aplikace se spouští příkazem:

```
ng serve
```

Alternativně lze aplikace spustit bez nutnosti globální instalace Angular CLI:

```
node_modules/.bin/ng serve
```

Serverová aplikace

Nachází se ve složce `backend-javascript` a po spuštění je dostupná na portu 3000. Zde je třeba doinstalovat knihovny pomocí příkazu:

```
npm install
```

Aplikace se spouští příkazem:

```
npm start
```

Alternativně lze spustit pomocí:

```
node server.js
```

Na stroji také musí běžet databáze MongoDB. Je třeba přidat řetězec pro připojení do souboru `.env`.

Příloha D

Docker manuál

Pro spuštění celého aplikačního rámce je možné použít Docker.

V kořenovém adresáři obou aplikací je třeba zkopírovat soubor `.env-example` a přejmenovat na `.env`.

Poté stačí spustit:

```
docker-compose up -d
```

Vytvoří se tak tři kontejnery - klientská a serverová část + databáze. Klientská aplikace běží na portu 4200. Knihovna `bcrypt` během instalace také kompiluje některé soubory přímo pro daný operační systém a procesor. Konfigurace Dockeru obsahuje automatickou instalaci knihoven. Pokud jsou ale nainstalovány před spuštěním Dockeru, spuštění aplikace skončí chybou. Klientská část aplikace se v prostředí Dockeru může kompilovat i několik minut před samotným spuštěním.