VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

WEBOVÁ APLIKACE PRO VIZUALIZACI A ANALÝZU PÍSMA Z DIGITALIZAČNÍHO TABLETU

WEB APPLICATION FOR ONLINE HANDWRITING DATA VISUALIZATION AND ANALYSIS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Vladimíra Šebová

AUTHOR

VEDOUCÍ PRÁCE — Ing. Ján Mucha, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Studentka: Vladimíra Šebová ***ID:*** 222737

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Webová aplikace pro vizualizaci a analýzu písma z digitalizačního tabletu

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je realizace webové aplikace pro vizualizaci a základní analýzu ručně psaného projevu zaznamenaného pomocí digitalizačního tabletu Wacom (online písmo). V rámci práce bude provedena rešerše současných možností implementace webových aplikací a zvolena nejvhodnější varianta k realizaci. Aplikace bude číst data ze souboru předem definovaného formátu (*.svc, *.json) a umožní přehrávaní písma dle časového razítka. Dále bude aplikace implementovat všechny základní vlastnosti video přehrávače (stop, pauza, apod.). Následně bude aplikace poskytovat základní analytické informace o přehrávaných datech jako například rychlost, zrychlení a počet tahů. Analyzovaná data bude možné exportovat jako video soubor.

DOPORUČENÁ LITERATURA:

[1] Wintab: Feel-Touch APIs. WILL Documentation [online]. Japan: Wacom, 2020, 2020 [cit. 2020-10-05]. Dostupné z: https://developer-docs-legacy.wacom.com/pages/viewpage.action?pageId=10422351

[2] DORMANN, Andreas. Ionic 5: Create Awesome Apps for IOS, Android, Desktop and Web. D&D Verlag Bonn, 2020.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: Ing. Ján Mucha, Ph.D.

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

ABSTRAKT

Táto bakalárska práca sa zaoberá rešeršou súčasných možností implementácie webových aplikácií, voľbou najvhodnejšej varianty k realizácií, návrhom a zrealizovaním webovej aplikácie pre vizualizáciu a základnú analýzu ručne písaného prejavu zaznamenaného pomocou digitalizačného tabletu Wacom. V práci je popísaných niekoľko technológií, ktoré boli použité pri návrhu a tvorbe tejto webovej aplikácie. Aplikácia bude schopná čítať dáta zo súboru predom definovaného formátu a prehrávať písmo pomocou časového razítka. Takisto bude poskytovať aj základné analytické informácie o prehrávaných dátach ako napríklad rýchlosť, zrýchlenie a počet ťahov. Analyzované dáta bude možné exportovať ako video súbor.

KĽÚČOVÉ SLOVÁ

Digitalizácia, Django, Python, Virtualizácia, Wacom, Webová aplikácia

ABSTRACT

The topic of this bachelor thesis is to deals with the research of current options for the implementation of web applications, the choice of the most suitable variant for implementation, the design and implementation of a web application for the visualization and basic analysis of handwritten speech recorded using a Wacom digitizing tablet. The work describes several technologies that were used in the design and creation of this web application. The application will be able to read data from a file of a predefined format and play the font using a time stamp. It will also provide basic analytical information about the played data, such as speed, acceleration and number of strokes. The analyzed data will be able to be exported as a video file.

KEYWORDS

Digitization, Django, Python, Virtualization, Wacom, Web application

ŠEBOVÁ, Vladimíra. *Webová aplikácia pre vizualizáciu a analýzu písma z digitalizačného tabletu*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačných technológií, Ústav telekomunikácií, 2023, 64 s. Bakalárska práca. Vedúci práce: Ing. Ján Mucha, Ph.D.

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Vladimíra Šebová
VUT ID autora:	222737
Typ práce:	Bakalárska práca
Akademický rok:	2022/23
Téma závěrečnej práce:	Webová aplikácia pre vizualizáciu a analýzu písma z digitalizačného tabletu

Vyhlasujem, že svoju záverečnú prácu som vypracovala samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

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

Brno ....

podpis autorky

POĎAKOVANIE

Rada by som poďakovala vedúcemu bakalárskej práce pánovi Ing. Jánovi Muchovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci. Jeho cenné rady, návrhy a spätná väzba boli veľkou pomocou pri zlepšovaní a rozvoji mojej práce.

Obsah

Zoznam obrázkov

Zoznam výpisov

Úvod

Táto bakalárska práca sa zaoberá rešeršou súčasných možností tvorby webových aplikácií a zvolením najvhodnejšieho variantu k realizácii. Rovnako sa práca zaoberá aj návrhom samotnej webovej aplikácie, implementáciou tohto návrhu a v konečnom dôsledku vizualizáciou písma získaného pomocou digitalizačného tabletu.

Digitálne technológie stále viac prenikajú do rôznych oblastí ľudských životov, vrátane písania a komunikácie. S rozšírením digitalizačných tabletov a ich schopností sa otvára nová oblasť výskumu a vývoja zameraná na analýzu a spracovanie písma z týchto zariadení na rôzne účely. Jednou z jeho významných výhod je schopnosť uchovávať rôzne parametre. Aj vďaka nim je schopné písmo analyzovať a takisto aj objavovať rôzne grafomotorické poruchy. Nástrojov na vizualizáciu písma je vo všeobecnosti málo, preto je existencia tejto webovej aplikácie dôležitá v tejto oblasti.

Cieľom tejto bakalárskej práce je navrhnúť a implementovať webovú aplikáciu, ktorá umožní vizualizáciu a analýzu písma z digitalizačného tabletu Wacom. Aplikácia bude spracovávať dáta z formátov .svc a .json, takisto ich bude aj vizualizovať do podoby grafu, ktorý bude ovládaný pomocou video prehrávača. Okrem toho bude aplikácia vedieť zobraziť základné premenné, ktoré vznikajú pri písaní, ako napríklad rýchlosť písania, zrýchlenie a počet prerušení medzi ťahmi. Stránka bude schopná uložiť súbor do vopred vybratých formátov.

Prvá kapitola práce objasňuje a vysvetľuje viacero problematík a nástrojov, ktoré boli využité na tvorbu práce. Medzi nich patrí definovanie pojmu webová aplikácia a aké majú tieto aplikácie využitie. Rovnako budú spomenuté technológie prispievajúce vývoju webových aplikácií ako Python, Django framework, HTML, CSS, Javascript, a takisto aj knižnice potrebné pre vizualizáciu ako Plotly, Matplotlib či Handwriting Sample. Nebudú vynechané ani prostredia, v ktorých sa kódy pre aplikáciu tvoria ako Visual Studio Code, Brackets, či program na vytvorenie návrhu aplikácie Figma.

Druhá kapitola je rozdelená do dvoch častí, a to do vizuálnej a funkčnej časti aplikácie. Vizuálna časť zahŕňa návrh webovej aplikácie a jeho implementáciu s porovnaním navrhovaného a skutočného vizuálu. Funkčná časť popisuje jadro aplikácie ako sú jej požiadavky, vlastnosti a takisto aj spôsob komunikácie medzi serverom a klientom. Ďalej sa zaoberá spôsobom, akým aplikácia zobrazuje a manipuluje grafom a možnosťou exportu grafu do rôznych formátov.

1 Teoretická časť študentskej práce

Táto kapitola sa bude venovať webovým aplikáciám. Okrem vysvetlenia ich samotnej podstaty a ich porovnaniu s webovými stránkami budú oboznámené aj technológie používané na ich vývoj, ako aj programy na ich návrh a programovanie.

1.1 Čo sú webové aplikácie

Webové aplikácie sú softvéry alebo programy, ktoré sú prístupné pomocou internetového prehliadača [1]. Ich účelom je prepájať koncového užívateľa s obsahom, ktorý nemusí byť k dispozícii iba na čítanie, ale takisto aj na manipuláciu. Neodmysliteľnou súčasťou webových aplikácií sú servery, na ktorých je postavená komunikácia medzi klientom a webovými aplikáciami ako takými. Klient k nim pristupuje pomocou webovej adresy [2].

Účel webových aplikácii môže byť rôzny. Najčastejšie druhy aplikácii, s ktorými sa človek stretáva pravidelne sú napríklad internetové obchody, cez ktoré si má možnosť objednať tovar či službu, takisto aj sociálne siete, pomocou ktorých je možné komunikovať, či zdieľať rôzny obsah alebo rozličné vzdelávacie aplikácie, ktoré pomáhajú s osobným rozvojom, či obohatením znalostí. V neposlednom rade sa medzi webové aplikácie radia aj finančné či bankové portály, pomocou ktorých sa môžu splácať pohľadávky, či pracovať s investíciami.

Výhod, ktorými disponujú webové aplikácie nie je málo. Jednými z najpraktickejších výhod webových aplikácii je použitie na rôznych platformách od Windowsu cez Linux až po MacOS, rovnako ako aj ľahké udržiavanie vďaka konzistentnému kódu, ktorý je v celej aplikácii nemenný. Prístup z rôznych druhov zariadení nebude takisto problémom. Nezáleží či je aplikácia spustená na telefóne, tablete, či počítači, jej chod a funkčnosť je pre tieto rozličné zariadenia prispôsobená. Nemenej dôležité je aj zjednodušenie každodenných procesov či vyhnutie sa aktualizáciám, ktoré sú pri iných druhoch softvéru nevyhnutné pre ich vývoj a prosperovanie. Vzhľadom na to, že je aplikácia internetová, nie je potrebné žiadne sťahovanie či inštalácia [3].

1.2 Webové aplikácie verzus webové stránky

Pojmy „webové aplikácie“ a „webové stránky“ môžu byť často zamenené. Aj napriek tomu, že tieto dva pojmy môžu vyvolávať v človeku pocit, že sa jedná o totožnú vec, nie je tomu tak.

Webová stránka je zoskupenie obsahu, ktorý je verejne dostupný každému návštevníkovi. Obsahom webových stránok sú väčšinou statické prvky, ako napríklad obrázky, text, audio, video, a tak ďalej. Dáta na nej teda nie je možné spracovávať

či ukladať. Ako už bolo vyššie spomenuté, webová aplikácia je navrhnutá pre koncového užívateľa. Statické prvky webovej stránky slúžia len pre istý vizuál, za to webová aplikácia ponúka koncovému užívateľovi možnosť s aplikáciou do istej miery pracovať a dovoľuje mu s dátami v určitom rozmedzí manipulovať. Stáva sa teda interaktívnou.

Rozhranie aplikácii je zväčša užívateľsky prívetivé a dbá sa na to, aby aplikácia plnila účel, na ktorý bola vytvorená. Keďže s ňou môžu prísť do kontaktu rôzne typy užívateľov, mnoho jej samotných funkcií musí byť ošetrených tak, aby používateľ využil aplikáciu len do takej miery, aby mohla byť v budúcnosti používaná ďalšími užívateľmi. Práve pre kontakt s koncovým užívateľom používajú aplikácie často funkciu autentifikácie, ako napríklad prihlasovacie formuláre, overovanie identity alebo spravovanie užívateľských účtov [4].

Webové stránky sa týmito funkciami nezaoberajú, pretože nie sú na interakciu prispôsobené. Čo sa týka funkčnosti stránky, zväčša býva jednoduchá, za to aplikácie bývajú komplexné a takisto aj väčšieho rozsahu. Ich vývoj býva zložitejší a vyžadujú údržbu v určitých časových intervaloch. Webové stránky sú naopak väčšinou jednoduché nie len na použitie, ale aj na údržbu [3].

1.3 Technológie pre vývoj webových aplikácii

Vývoj webových aplikácií je v súčasnej dobe dynamický a rýchlo sa meniaci proces. Technológie prispievajúce vývoju webových aplikácii sa neustále vyvíjajú a zlepšujú, čo dáva nespočetné množstvo možností vývojárom tvoriť moderné, efektívne a interaktívne webové aplikácie.

1.3.1 Python

Jedným z populárnych programovacích jazykov, ktorý bol vybratý pre zrealizovanie tejto práce je Python. Vytvoril ho počítačový programátor Guido van Rossum a v roku 1991 vydal jeho prvú verziu 0.9.0 [5].

Aj napriek tomu, že tento programovací jazyk nie je ani z ďaleka najstarším, jeho využitie môže byť rôznorodé. Okrem vývoja rôzneho softvéru môže byť takisto nástrojom pre vytváranie pracovných postupov a rovnako je možné ho spojiť aj s databázovými systémami [6]. V procese tvorby tohto jazyka boli vyvinuté rôzne frameworky, pomocou ktorých je v súčasnosti možné s ním vyvíjať aj webové aplikácie [6]. Využitie má rovnako aj pri spracovaní veľkého množstva dát či vykonávaní komplexnej matematiky [6].

Dôvodov výberu tohto konkrétneho jazyka bolo hneď niekoľko. Python disponuje jednoduchou syntaxou, čo je výhoda nie len pre pokročilých programátorov,

ale najmä pre začiatočníkov. Známy je teda pre svoju „zen filozofiu“, čo reprezentuje simplicitu a čitateľnosť kódu. Rovnako je Python známy aj preto, že typy premenných nemusia byť deklarované, ale určujú sa na základe priradených hodnôt, je teda dynamicky typovaný [7].

Jeho štandardná knižnica je dosť veľká na to, aby zjednodušila nie jednému vývojárovi zhotoviť funkčnú, efektívnu a kvalitnú webovú aplikáciu. Nachádza sa v nej množstvo existujúcich modulov, ktoré môžu slúžiť na rôzne účely, ako napríklad na prácu s reťazcami, sieťovým programovaním, programovaním grafického rozhrania a mnoho ďalších. Kód teda nie je potrebné písať od nuly, ale je možné využívať rôzne moduly. Okrem tejto veľkej základnej knižnice sa priaznivci tohto jazyka podieľajú na tom, aby bol použiteľný aj pre konkrétne účely. Z toho dôvodu si veľa vývojárov píše vlastné knižnice, ktoré majú využitie v konkrétnych projektoch a keďže sú tieto knižnice zverejňované, môžu s nimi pracovať aj iní programátori zadarmo. Príkladom knižníc na konkrétne využitie sú napríklad NumPy či Pandas pre prácu s dátami, TensorFlow na strojové učenie, Tkinker, či PyQt pre prácu s grafickým užívateľským rozhraním, rovnako ako aj Django pre webový vývoj, ktorý bol použitý aj v tomto prípade. Komunita tohto programovacieho jazyka rovnako poskytuje aj dokumentáciu, tutoriály či fóra, čo zľahčuje ostatným prácu s týmto jazykom. Pre uľahčenie riadenia rôznych zložitých projektov Python disponuje funkciou objektovo-orientovaného programovania, čo znamená, že kód je možné organizovať do tried a objektov pre ľahší prístup k dátam. Tento jazyk je takisto aj interpretovaný, čo znamená, že sa kód vykonáva bez kompilácie, jeho výsledky je teda možné vidieť okamžite. Kód tohto jazyka je multiplatformový. Jeho prenos je možný medzi rôznymi operačnými systémami ako Windows, macOS, Linux a ďalšie [8].

1.3.2 Django framework

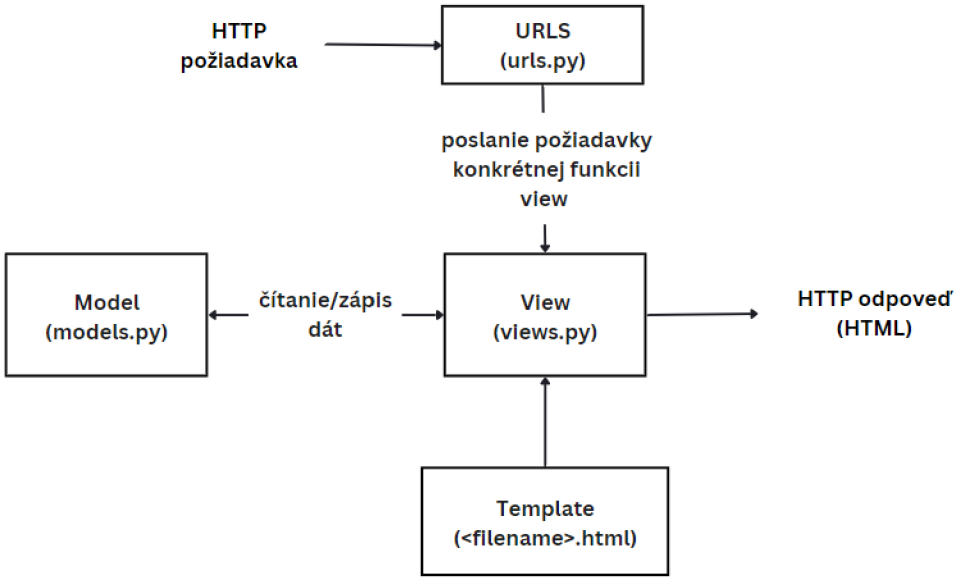
Django je populárny webový framework pre Python, navrhnutý na rýchle a jednoduché vytváranie webových aplikácií. Poskytuje vysokoúrovňové nástroje a automaticky rieši množstvo bežných potrieb, ako je správa databázy, bezpečnosť a administratívne rozhranie. Jeho dobre zdokumentované prostredie umožňuje rýchle začatie projektu. Django je ideálny pre vývoj databázových aplikácií a poskytuje objektovo-relačné mapovanie, automatickú generáciu formulárov a jednoduchý prístup k dátam cez Python API. S jeho pomocou je možné vytvárať dynamické webové stránky s čistým a jednoduchým mapovaním URL adries. Vďaka širokému spektru funkcií a flexibilnému administratívnemu rozhraniu je Django vhodný pre rýchle vytváranie aplikácií s databázovým backendom [9].

Medzi jeho hlavné výhody patrí najmä jeho rýchlosť, robustnosť, bezpečnosť,

škálovateľnosť a všestrannosť, tieto vlastnosti sú popísané podrobnejšie v nasledujúcich odstavcoch [10].

Filozofiou tohto frameworku je „Batteries included“ a tá sa snaží poskytovať všetko, čo by vývojári chceli robiť takzvane „out of the box“, teda nad klasické možnosti. Všetko potrebné pre rôzne projekty sa nachádza na jednom mieste, všetko spolu bezproblémovo funguje a dodržiava isté princípy dizajnu, má taktiež aj rozsiahlu a aktuálnu dokumentáciu. Multifunkčnosť tohto frameworku nie je jedinou z jeho silných stránok. Je s ním možné vytvoriť webovú stránku takmer akéhokoľvek využitia, napríklad redakčné systémy, sociálne siete či spravodajské weby. Svoj obsah môže poskytovať v akomkoľvek formáte, ako napríklad HTML, JSON a XML. Okrem týchto rôznych funkčností sa dá ešte aj rozšíriť o ďalšie rôzne komponenty. Bezpečnosť tohto frameworku nie je vôbec problém, nakoľko bol vyvinutý tak, aby vývojárom pomohol vyhnúť sa bežným bezpečnostným chybám veľmi rafinovaným spôsobom, a to pomocou automatickej ochrany webovej stránky. Poskytuje bezpečný spôsob správy používateľských účtov a hesiel, pričom sa vyhýba bežným chybám, ako je vkladanie informácií o reláciách do súborov cookie tam, kde je to zraniteľné alebo priame ukladanie hesiel namiesto hash hesla. Hash hesla je totiž hodnota s pevnou dĺžkou vytvorená odoslaním hesla prostredníctvom kryptografickej funkcie hash. Django má schopnosť kontrolovať či je zadané heslo správne, jeho spustením cez hašovaciu funkciu a porovnaním výstupu s uloženou hašovou hodnotou. Táto funkcia je ale „jednosmerná“, takže aj keby sa útočník dozvedel hash hodnotu, spätné zistenie pôvodného hesla je preňho ťažké. Štandardná ochrana pred mnohými zraniteľnosťami, ako napríklad skriptovanie medzi stránkami, falšovanie žiadostí medzi stránkami či clickjacking je zabezpečená [11].

Architektúra tohto frameworku je založená na komponentoch, čo znamená, že jej každá časť je nezávislá od ostatných, v prípade potreby je teda možné ju vymeniť. Toto oddelenie dáva možno vývojárom škálovať pre zvýšenú prevádzku pridaním hardvéru na akejkoľvek úrovni. Celý koncept Djanga je písaný pomocou vzorcov tak, aby bolo možné ďalej kód opakovane používať. Nedochádza k zbytočnej duplicite a rovnako podporuje aj zoskupovanie súvisiacich funkcií do opätovne použiteľných celkov, na nižšej úrovni zoskupuje súvisiaci kód do modulov. Keďže je Django písaný v jazyku Python, ktorý beží na mnohých platformách, vývojári nie sú viazaní na žiadnu konkrétnu serverovú platformu a svoje aplikácie je možné spúšťať na viacerých verziách systémov Linux, Windows a macOS. Django ako taký dostal veľkú podporu aj od mnohých poskytovateľov webhostingu, ktorí často poskytujú špecifickú infraštruktúru a dokumentáciu pre hosťovanie stránok tomuto frameworku [11].



Obr. 1.1: Štruktúra frameworku Django [11]

Na obrázku 1.1 je vidieť štruktúru tohto frameworku. HTTP požiadavka sa najprv dostavé na adresu urls.py a potom sa pošle na konkrétnu funkciu zhody vo views.py. Views.py obsahuje funguje Pythonu, ktoré prevezmú webovú požiadavku z urls.py a poskytnú HTTP požiadavku do templates. Podľa dotazov sa môže požiadavka dotazovať aj k údajom v models.py [12].

1.3.3 Matplotlib

Podľa koncovky „lib“ je zjavné, že Matplotlib je knižnica, ktorá je výkonná a veľmi populárna na vizualizáciu údajov v Pythone. Je schopná vytvárať rôzne čiarové, stĺpcové grafy a takisto aj rozptylové grafy, rovnako aj histogramy, a to všetko v 2D zobrazení. Takisto je veľmi flexibilná a multiplatformová. Je postavená na poliach NumPy a navrhnutá na prácu so širším zásobníkom SciPy. Predstavil ju John Hunter v roku 2002. Jej najväčšou výhodou je, že umožňuje vizuálny prístup k obrovskému množstvu údajov v jednoduchých a prehľadných vizuáloch [13].

1.3.4 JavaScript

Podstatnou časťou pre zobrazovanie dynamického obsahu na webovej aplikácii sa zaoberá JavaScript. Je to totiž skriptovací jazyk na vytváranie dynamického obsahu webových stránok či aplikácii. Vytvára prvky, ktoré zlepšujú a celkovo umožňujú

používateľovi aplikácie či stránky interagovať s ňou. Bol vynájdený v roku 1995 americkým programátorom Brendanom Eichom. Pôvodne bol vyvinutý pre webový prehliadač Netscape 2 a neskôr z neho vznikol štandard ECMA-262. Postupne bol spoločnosťou Mozilla dovyvíjaný pre ich webový prehliadač Firefox [14].

Jedno využitie naberá pri weboch či mobilných aplikáciách. Vývoj frameworkov JavaScriptu, ako napríklad jQuery, ReactJS, Node.js, umožňuje vývojárom používať vo svojich projektoch vopred napísaný kód JavaScript. Frameworky JavaScriptu pozostávajú z funkcií, ktoré majú na starosti zjednodušiť proces vývoja a ladenia. Jednou zo základných funkcií JavaScriptu je pridávanie dynamiky webovým stránkam, čo zahŕňa už funkcie, ako napríklad zobrazovanie animácií, úpravu viditeľnosti textu a vytváranie rozbaľovacích ponúk. Podstatné časti webových stránok, ako sú HTML a CSS, dodávajú webom iba statické zobrazenie. JavaScript im dodá dynamickosť a používateľom dá možnosť interagovať s takýmito webovými stránkami [15].

JavaScript má veľké množstvo výhod, ktoré z neho robia lepšiu voľbu ako jeho konkurenti. Vďaka jednoduchosti sa JavaScript ľahšie učí, implementuje. Rýchlosť tohto jazyka je takisto výhodou, keďže spúšťa skripty priamo vo webovom prehliadači. Tento jazyk takisto znižuje požiadavky odosielané na server. Jeho aktualizácie sa vzťahujú len na určité časti webovej stránky. Okrem výhod má tak ako každý jazyk aj tento svoje nevýhody. Rôzne webové prehliadače môžu interpretovať JavaScript kód odlišne, čo spôsobuje nekonzistentnosť. Bezpečnosť kódu tohto jazyka je tiež nestabilná vzhľadom na to, že beží na strane klienta, takže je zraniteľný voči zneužitiu nezodpovednými používateľmi. Ladenie JavaScript kódu môže byť ťažšie preto, že niektoré editory podporujúce toto ladenie môžu byť menej efektívne ako iné [15].

1.3.5 Plotly

Plotly je grafická bezplatná open-source knižnica interaktívnych grafov. Jej uplatnenie je možné nájsť v nie jednom programovacom jazyku, nakoľko má plnú podporu vo viacerých jazykoch, ako sú napríklad Python či JavaScript. Vie vytvoriť rôzne typy grafov od jednoduchých, príkladom sú čiarové, bodové, plošné, stĺpcové či chybové, až po komplexnejšie ako histogramy, tepelné mapy, polárne, či bublinové grafy alebo aj grafy s viacerými osami. Táto knižnica nie je len o vytváraní vizuálizácií, vie takisto aj vyzdvihnúť dátam potenciál a veľmi dobre sa ňou vypichuje podstata zobrazených údajov. Spadá pod MIT licenciu a jej grafy je možné vykresľovať v Jupyter Notebooks, samotných HTML súboroch či v Dash aplikáciách [16].

Súčasťou knižnice Plotly je aj knižnica Plotly.js. Ako je zrejmé z prípony, ide o JavaScriptovú knižnicu, ktorá sa zameriava primárne na vyhotovenie grafov a in-

terakciu s nimi pomocou JavaScriptu na webovej stránke. Výhodou tejto funkcie je to, že užívateľ interaguje na stránke s grafom bez nutnosti odosielania požiadaviek na server, ako to býva pri exportovaných grafoch Pythonu [17].

1.3.6 NumPy

NumPy je knižnica v programovacom jazyku Python, ktorá obsahuje množstvo nástrojov na prácu s multidimenzionálnymi poliami. Pomocou nej sa dajú nad týmito poliami prevádzať matematické operácie. Hlavným princípom NumPy je efektívne a rýchlo spracovať dáta. Pre manipuláciu s veľkými dátovými sadami poskytuje jednoduché rozhranie a používa vysoko optimalizované operácie, ako napríklad sčítanie, odčítanie, násobenie, delenie, a tak ďalej. Nad poliami sa tieto operácie vykonávajú naraz, vďaka čomu je rýchlosť a efektivita o dosť vyššia v porovnaní s postupným spracovávaním jednotlivých prvkov. Takisto táto knižnica poskytuje rôzne metódy na manipuláciu s poliami, ako je rezanie, indexovanie a zlučovanie, čo umožňuje flexibilné a efektívne spracovanie dát. Pomocou týchto nástrojov je možné vykonávať štatistické výpočty, lineárnu algebru, Fourierovu transformáciu, náhodné generovanie čísel a mnoho ďalších operácií. V spojitosti s ďalšími knižnicami Pythonu, ako napríklad SciPy, Pandas alebo Matplotlib, vie Numpy efektívne spracovávať, analyzovať a vizualizovať dáta vo vedeckých a analytických aplikáciách [18].

1.3.7 Handwriting Sample

Pre vizualizáciu písma z grafického tabletu je vhodná knižnica, ktorá bude obsahovať všetky funkcie na správne zobrazenie písma vo webovej aplikácii. V tomto prípade bol vybratý balík Handwriting Sample, ktorý vyvinuli členovia laboratória pre analýzu mozgových chorôb BDALab. Tento balík obsahuje PyPi inštalovateľný modul, ktorý slúži na manipuláciu s online dátami rukopisu, ktoré boli získané prostredníctvom digitalizačných tabletov spoločnosti Wacom. Tento balík takisto používa triedu Handwriting Sample, ktorá vie jednoducho manipulovať s dátami písanými rukou. Tieto dáta pozostávajú z časových údajov, ktoré majú takúto štruktúru: pozícia na osi x a y, časové razítko, poloha pera, azimut, sklon a tlak. Hlavné funkcie knižnice sú načítanie dát zo súborov formátov .svc a .json, načítanie NumPy polí súboru, rovnako ako aj dátové rámce Pandas. Knižnica takisto umožňuje aj transformáciu jednotiek ako pozície na osi x a y do milimetrov, uhly na stupne a čas na sekundy či jednoduchý prístup a manipuláciu s časovými údajmi, ale aj ukladanie dát v rôznych formátoch. Metadáta súborov sa dajú spracovať rovnako jednoducho. Balík sa dá použiť aj pre dáta získané z iných zariadení ak vyhovujú skupine vyššie uvedeného zoznamu časových údajov [19].

1.3.8 HTML

Jazyk HTML je spolu s CSS neodmysliteľnou súčasťou nástrojov pre tvorbu webových stránok či aplikácii. Je to takzvaný značkovací jazyk, pretože sa skladá zo značiek, respektíve tagov. Tieto značky slúžia k tomu, aby sa pomocou nich dal takzvane „obalovať text“, a tak ho členiť a dávať každej časti iný určitý význam. Je mylne označovaný ako programovací aj napriek tomu, že neumožňuje predvádzať žiadne logické operácie. Vďaka programovacím jazykom je ale HTML ďalej upravovateľný [20].

Stránka napísaná v HTML jazyku je pre prehliadač veľmi dôležitá. Vývojári používajú HTML ako nástroj, pomocou ktorého vie webový prehliadač určiť, akým spôsobom má zobrazovať určité prvky webovej stránky, ako napríklad obyčajný text, hypertextové odkazy či mediálne súbory. Takisto sa dá prostredníctvom HTML kódu vytvoriť webovú dokumentáciu. Umožňuje totiž organizovať a formátovať dokumenty podobne ako iné známe editory. V súčasnosti je považovaný za webový štandard. Je takisto pravidelne aktualizované vďaka World Wide Web Consortium, ktoré udržiava a vyvíja špecifické HTML [20].

HTML dokument sa člení na celky. Pre prehliadač je najprv potrebné definovať, o aký druh dokumentu sa jedná. Neskôr sa musí definovať, že sa jedná o HTML dokument. Potom sa dokument rozdelí na takzvanú hlavičku a telo. V hlavičke sa nachádzajú informácie pre bližšie definovanie tohto dokumentu. Nachádzajú sa v nej rovnako aj takzvané meta dáta, ako napríklad titulka stránky či definovanie skupiny znakov, ktoré sa budú v dokumente používať. V tele HTML dokumentu sa nachádza obsah, ktorý sa zobrazuje na webovej stránke, ktorý má v konečnom dôsledku upútať používateľovu pozornosť [21].

Všetky už vyššie spomenuté HTML stránky, z ktorých sa môže jedna webová stránka skladať obsahujú množstvo prvkov, ktoré pozostávajú zo značiek a atribútov. Tieto značky a atribúty sú základom pre tvorbu HTML stránok a aplikácií, každá z nich predáva prehliadaču určité informácie. Značka informuje webový prehliadač o tom, kde sa prvok začína a kde naopak končí. Ohraničuje teda jeho výskyt. Atribút naopak popisuje rôzne vlastnosti prvku. Prvok sa skladá z troch častí, a to z otváracej značky, obsahu a záverečnej značky. Otváracia značka sa používa na označenie, kde prvok začína. HTML značka je obalená do špicatých zátvoriek. Obsah, ktorý sa medzi značkami nachádza je práve to, čo používateľ uvidí na stránke. Záverečná značka je veľmi podobná tej otváracej s tým rozdielom, že pred názvom tejto značky sa nachádza lomka. Ďalšou dôležitou súčasťou je už vyššie spomínaný atribút, ktorý sa takisto skladá z dvoch častí, a to z jeho názvu a hodnoty. Jeho názov identifikuje informácie, ktoré chce používateľ pridať, naopak jeho hodnota poskytuje ďalšie špecifikácie. Pre programovanie je veľmi dôležitý ďalší HTML atribút, a to

je takzvaná trieda. Atribút trieda pridáva informácie o štýle, ktorý môže fungovať na rôznych prvkoch s rovnakým názvom triedy. Veľké množstvo elementov má rovnako ako otváraciu, tak aj zatváraciu značku. Niektoré sú ale výnimkou z dôvodu, že značka nemá obsah, ktorý by zobrazovala. Môže sa teda definovať sama len pomocou atribútov, ako napríklad značka pre čiaru alebo zobrazuje obsah, ktorý už má vlastné parametre, ktoré sa rovnako dajú meniť dodatočnými atribútmi, napríklad značka pre obrázok [21].

1.3.9 CSS

Stránky tvorené pomocou HTML by síce spĺňali funkčnosť, používateľov by ale takéto holé stránky bez dizajnu nemuseli zaujať. Či už je stránka vytvorená za účelom vzdelávacím alebo má práve spĺňať nejakú konkrétnu funkciu, jej dizajn môže veľakrát zavážiť či na nej používateľ strávi nejaký svoj čas. Vyššie boli spomenuté HTML značky, ktoré obsah stránky členia a takzvane obaľujú. Jazyk CSS robí ďalej to, že takto obalenému obsahu dáva dizajn, nie len aby spĺňal funkciu predaja informácii používateľovi, ale aby aj zaujal vizuálom, či aby bol text prehľadne upravený a vhodný na čítanie. Jazyk CSS pomáha teda webovým stránkam zmeniť napríklad farby, samotné rozloženie alebo fonty písma. Vývojárom dáva možnosti prispôsobiť stránku pre rôzne typy zariadení, ako napríklad veľké či malé obrazovky alebo tlačiarne. CSS je od HTML nezávislé, a je teda možné ho použiť s rôznymi inými značkovacími jazykmi. Oddelenie HTML od CSS uľahčuje údržbu stránok, zdieľanie štýlov na stránkach a prispôsobenie stránok rôznym prostrediam. Tieto CSS štýly sú uložené v samostatnom súbore, ktorý je potrebné pripnúť do HTML hlavičky pre správne prepojenie a následné premietanie CSS štýlov na HTML stránku [22].

Hlavnou výhodou používania CSS štýlov je šetrenie času, keďže CSS štýly je možné napísať raz a následne použiť rovnaké štýly pre viac HTML stránok. Je možné definovať štýly pre každý prvok a použiť ho na veľmi veľa iných stránkach. Rýchlosť načítania stránok je takisto iná s používaním CSS štýlov a bez nich. Keďže sa odkaz na tieto štýly prikladá do hlavičky HTML stránky a nepíšu sa štýly ku každému atribútu stránky zvlášť, zmenšuje to veľkosť a rovnako aj počet riadkov v HTML súbore, čo dodáva sťahovaniu a následnému načítaniu stránky väčšiu rýchlosť. Údržba takýchto stránok je rovnako jednoduchšia, pretože pri vykonaní veľkej zmeny stačí jednoducho zmeniť štýl a všetky prvky na všetkých webových stránkach sa automaticky aktualizujú. CSS má takisto oveľa širšiu škálu atribútov ako HTML, takže je možné pre webové stránku poskytnúť lepší vzhľad než len s HTML atribútmi. Šablóny týchto štýlov rovnako umožňujú aj optimalizáciu obsahu pre viac ako jeden typ zariadenia. S použitím rovnakého HTML dokumentu je možné prezentovať rôzne verzie webovej stránky pre rôzne vreckové zariadenia od mobilných telefónov

cez tablety, až po zariadenia väčších rozsahov ako tlačiareň. Pre prevádzkovanie webových stránok aj do budúcna je veľmi vhodné používať CSS štýly práve preto, aby boli súčasné HTML stránky kompatibilné aj s budúcimi prehliadačmi [22].

1.4 Programy na tvorbu webových aplikácií

Čo sa týka samotnej tvorby webových aplikácií, v teoretickej rovine by žiadne dodatočné programy neboli nutné. Samotný program by bolo možné písať aj v obyčajnom textovom editore, rovnako ako aj dizajn by bolo možné nakresliť rukou na papier. V praxi by sa webová aplikácia takýmto spôsobom navrhovala a písala veľmi ťažko. Programy na tvorbu takýchto webových aplikácií neponúkajú len prostredie, ktoré je prispôsobiteľné pre používateľa, čo môže veľmi uľahčiť prácu s programom. Ponúkajú ale omnoho viac funkcií, vďaka ktorým je tvorba týchto programov jednoduchšia.

1.4.1 Figma

Pre návrh vizuálu webovej stránky bola vybratá Figma. Figma je výkonný web dizajnový nástroj pre tvorbu rôzneho obsahu od webových aplikácií cez webové stránky po logá a ešte oveľa viac. Je to platforma, ktorá spája výkonné dizajnové funkcie s efektívnejším pracovným postupom. Bola vytvorená pre budúcnosť webu práve vďaka svojím funkciám, ktoré nie sú až tak bežné. Použiť sa v nej dá napríklad funkcia moderného pera, pomocou ktorého sa dajú vykresliť rôzne vektorové siete. Nie je potrebné zlučovanie či pripojenie pera k pôvodnému bodu cesty. Takisto sa v nej dajú jednoduchým spôsobom navrhovať oblúky pre rôzne prvky, ako napríklad hodiny, obrazovky hodiniek či koláčové grafy. Obsahuje pokročilé funkcie písma, ktoré pomôžu vyjadriť svoju značku nie jednému dizajnérovi [23].

Pre jednoduchšiu prácu s objektami existuje funkcia pre manuálnu zmenu veľkosti, ktorá umožňuje napríklad zmeniť veľkosť tlačidiel spolu s textom alebo takisto funkcia pre automatické rozloženie komponentov pre jednoduchšiu prácu s dizajnom ako takým. Figma obsahuje takisto aj moduly, ktoré automatizujú úlohy a pomáhajú zvýšiť pracovný výkon. Pomocou widgetov je možné zvýšiť produktivitu svojho celého tímu. Do vyhľadávania stačí zadať položku, ktorú je momentálne potrebné pridať a keďže Figma disponuje množstvom dizajnov, každý projekt si nájde to svoje. Zákazníci figmy môžu takisto vytvárať či disponovať vlastnými súkromnými doplnkami a miniaplikáciami v rámci svojej spoločnosti. Pre jednoduchú prácu s väčším množstvom obsahu je možné vytvoriť si konzistentné štýly týkajúce sa farby, textu, mriežky či efektu, ktoré pomáhajú udržiavať obsah v jednotnom štýle. V spojení s dostupnými knižnicami je veľmi jednoduché pridať na svoje plátno svoje či cudzie výtvory. Projekty Figmy sú prístupné na prezeranie pre neobmedzený počet divákov,

čo umožňuje množstvu spolupracovníkom prezrieť si konkrétny projekt či dokonca ho aj okomentovať. Export pre ďalšie množstvo klientov je jednoducho možný v rôznych formátoch. Statické súbory sa môžu veľmi rýchlo stať interaktivnými vďaka ďalším možnostiam Figmy. Prepojenie prvkov vie byť veľmi jednoduché a pri pridaní rôznych animácií vedia produkty, ako napríklad vizuály eshopov ukázať svoju funkčnosť už v samotnom dizajne projektu. Figma disponuje aj aplikáciou, pomocou ktorej je možné si prezerať svoje návrhy aj mimo času stráveného na počítači [23].

1.4.2 Brackets

Pre programovanie vizuálu webovej aplikácie bol vybratý editor Brackets. Brackets je ľahký, ale za to výkonný a moderný textový editor, ktorý je od základov vytvorený pre webových dizajnérov a front-end vývojárov. Boli doňho zakomponované vizuálne nástroje, ktoré mali pomôcť správnym spôsobom bez toho, aby prekážali kreatívnemu procesu vývojára. Je to open-source projekt, ktorý je podporovaný aktívnou a zanietenou komunitou [24].

Prostredie programu Brackets je prispôsobiteľné pre jeho používateľa. Nie je vôbec potrebné preskakovať medzi kartami súborov. Namiesto toho je možné okno programu rozdeliť na viac častí, pre ľahšiu prácu medzi dvoma súbormi. S týmto editorom je možné získať prepojenie ku svojmu prehliadaču v reálnom čase. Zmeny vykonané v HTML či CSS súboroch je možné vidieť okamžite v živom náhľade tohto programu. Podporované sú rovnako aj preprocesory, ktoré pomáhajú aktívne napredovať vývojárom v ich pracovnom postupe. Tento editor má rovnako aj množstvo rozšírení či pluginov, ktoré umožňujú spoluprácu v rôznych oblastiach programovania. Takisto vedia pomôcť aj pri zmene vizuálu pracovného prostredia, takže je možné si vybrať zo širokého množstva šablón, ktoré vedia spraviť toto prostredie prehľadnejšie [24].

1.4.3 Visual Studio Code

Pre programovanie back-endu webovej aplikácie bol vybratý program Visual Studio Code. Tento program inak nazývaný aj ako VS Code je open-source editor pre písanie či upravovanie zdrojového kódu programu. Bol vytvorený spoločnosťou Microsoft v roku 2015. Je možné ho použiť na kódovanie v akomkoľvek programovacom jazyku bez prepínania editorov. Má podporu pre rôzne programovacie jazyky vrátane Pythonu, Java, C++, JavaScript a mnoho ďalších [25].

Visual Studio Code obsahuje mnoho rozšírení pre ľahšiu prácu so zdrojovým kódom. Jednou z populárnych funkcií je napríklad rozšírenie Live Share, ktoré ponúka zdieľať svoj kód na diaľku, a teda je možné pomocou neho spolupracovať s ostatnými

vývojármi. Kód je možné upravovať v reálnom čase a pre komunikáciu medzi účastníkmi je možné použiť funkciu četu a hovoru. Vhodné je to pri veľkých projektoch pre rôzne skupiny vývojárov, ale napríklad aj pre skupinové projekty v škole. Pre nováčikov v programovaní je v tomto programe funkcia na zvýrazňovanie kľúčových slov, čo pomôže ľahko identifikovať vzory kódovania pre jednoduchšie učenie. Pre podobnú pomoc tento program disponuje funkciami ako IntelliSense či Peek Definition pre jednoduché pochopenie rôznych funkcií a súvislosti kódov medzi sebou. Takisto pri kódovaní ponúka Visual Studio Code návrhy na dokončenie riadkov kódu a rýchle opravy bežných chýb. Je možné tiež použiť aj debugger, ktorý prejde každým riadkom kódu a pomôže užívateľovi pochopiť, čo sa v kóde deje. Rovnako je možné aj prispôsobovať si pracovné prostredie pomocou výberu svojho obľúbeného písma a ikon či témy pracovného prostredia ako takej. Pomocou vstavaného ovládania zdroja je možné si uložiť prácu v priebehu času tak, aby nedošlo ku strate pokroku. Rovnako sa dá porovnať si dve verzie rôznych časových bodov pomocou grafického zobrazenia vedľa seba. Pre zložitejšie projekty napríklad v oblasti dátovej vedy či vizualizácie je k dispozícií aj Jupyter Notebook v rámci VS Code. Pomocou neho je možné krok za krokom vizualizovať a integrovať s údajmi v programe, premennými či grafmi [25].

1.5 Online rukopis

Nie je to až tak dávno, kedy sa technológie vyvinuli takým spôsobom, že sa aj bežné činnosti ako práve obyčajné písanie začali meniť. Online rukopis je príkladom toho, ako sa tradičné písanie mení a prispôsobuje sa digitálnym nástrojom a novým formám komunikácie. Táto podkapitola sa bude zaoberať tým, čo je online rukopis, ako sa dá získať, ako sa líši od tradičného písania a aké výhody prináša.

Online rukopis, známy aj ako digitálne písanie rukou, je proces písania na digitálnom zariadení pomocou špeciálneho pera alebo stylusu. Tento text môže byť ďalej uložený a elektronicky zdieľaný s inými ľuďmi. Môže byť použitý na rôzne účely, ako napríklad tvorba poznámok, podpisov, kresieb, dokumentov alebo aj na analýzu písma ako takého [26].

Tento online rukopis je možné získať pomocou rôznych nástrojov a technológií. Niektoré z nich sú napríklad zahrnuté priamo v operačnom systéme, patria medzi nich Windows Ink alebo Apple Pencil. Ďalšími aplikáciami môžu byť Evernote, OneNote, Google Keep alebo GoodNotes, ktoré majú vlastné špecifiká a funkcie prispôsobené potrebám konkrétneho užívateľa. Tieto aplikácie používajú rôzne technológie vrátane dotykových displejov, digitálneho pera alebo stylusu, ktoré zachytávajú pohyby ruky užívateľa a konvertujú ich na digitálny text. Tento digitálny text môže byť následne upravovaný, zdieľaný a ukladaný v rôznych formátoch.

V prvom rade je online rukopis veľmi pohodlný, keďže umožňuje užívateľom písať na digitálnom zariadení s pohodlným perom alebo stylusom, ktorého používanie môže byť menej namáhavé a unavujúce ako písanie na papier. Okrem toho je online písmo ľahko upravovateľné bez fyzickej potreby dokument vyhodiť, respektíve upravovať text na papieri. Netýka sa to len úpravy gramatiky, ale takisto je možné prispôsobovať si veľkosti písma či farby, čo môže následne pomôcť pri čítaní textu. Keďže je online rukopis možné ukladať a rovnako aj svoju prácu zdieľať ďalej práve v digitálnej podobe, zvyšuje to prístupnosť a mobilitu týchto dokumentov, pretože je veľmi jednoduché k nim pristupovať z rôznych miest a rôznych typov zariadení. Rovnako je možné tento online rukopis konvertovať na digitálny text, čo umožňuje rýchle a ľahké vyhľadávanie, editovanie a spracovanie informácií v digitálnej podobe. Okrem týchto prostých dôvodov digitalizácie sa pomocou nej dá online písmo aj analyzovať, čo otvára možnosti vo veľa odvetviach, napríklad pri diagnostike rôznych grafomotorických chorôb u detí. Takisto je takéto písanie šetrnejšie k životnému prostrediu, keďže nie je potrebné kupovať papier pre zapisovanie informácií. Výhodu to má najmä pri ľuďoch či spoločnostiach, ktoré denne vyprodukujú veľké množstvo dokumentov. Počiatočná investícia môže byť väčšia než pri klasickom písaní perom na papier, z dlhodobého hľadiska vie ale online písanie ušetriť aj peniaze za rôzne doplnky potrebné pri bežnom písaní, teda pri nástrojoch na korektúru či zvyrazňovanie textu.

1.6 Digitalizačný tablet Wacom Cintiq 16

Čo sa týka samotnej digitalizácie, existuje na ňu už veľké množstvo nástrojov, ktoré ju pre používateľa robia veľmi jednoduchou, praktickou a efektívnou. Rôzne formy produktov si žiadajú rôzne nástroje, ktoré pomôžu dostať produkty do digitálnej podoby. Pri online písme je najvhodnejšou a najefektívnejšou formou digitalizačný tablet [27].

Tablet značky Wacom je určený pre kreatívne použitie, ktorý umožňuje používateľom kresliť a maľovať priamo na obrazovke pomocou štýlového pera. Konkrétny model Wacom Cintiq 16 má 15,6-palcovú obrazovku s rozlíšením 1920 x 1080 pixelov a 72% pokrytie farebného priestoru NTSC, čo znamená, že farebná škála tohto tabletu je presná a výrazná. Jeho obrazovka má antireflexnú fóliu, ktorá zabraňuje rušivým odrazom. Vďaka technológií EMR bude pero k tabletu fungovať po celú dobu práce s tabletom. Je totiž napájané elektromagnetickým poľom, ktoré generuje EMR senzor pod displejom. Táto technológia umožňuje bezdrôtovo a bezbatériovo používať pero s veľkou voľnosťou. Pero zdieľa rovnaký snímač a ovládač pre dotyk perom aj prstom a vďaka technológii ES zase dokáže zabezpečiť kompatibilitu s inými

protokolmi pera, ktoré používajú rovnakú technológiu kapacitného pera, ako je napríklad pero Surface od spoločnosti Microsoft. Navyše, pero bolo precízne navrhnuté tak, aby vyváženie hmotnosti, pogumovaný úchop a poloha tlačidiel umožňovali pohodlné používanie. Pero rovnako reaguje na naklonenie ruky používateľa a okamžite reaguje pri pohybe, pričom zachytí každý ťah pera. Znížená paralaxa tiež poskytuje plnú kontrolu, takže kurzor je tam, kde sa očakáva, že sa bude nachádzať. Je rovnako citlivé aj na tlak a vďaka 8192 úrovniam citlivosti rozpozná aj ľahké ťahy. Spolu s perom je možné pomocou tohto tabletu vytvoriť rôzne vizuálne kreácie, ako napríklad 3D modely či maľby alebo rôzne iné grafické práce. Okrem tvorby grafického obsahu je tento tablet prispôsobený aj na úpravu fotografií či strih videí. Použitie vie nadobudnúť aj čo sa týka praktickosti, ako napríklad diár či zápisník, do ktorého je možné si písať ručne písané poznámky [27].

Povrch tabletu, ktorý je citlivý na tlak je dôvodom fungovania tohto chytrého pomocníka. Práve pomocou tohto systému citlivého na tlak umožňuje tabletu Wacom rozpoznať hrúbku tvaru. Pomocou neho odosiela elektronické signály z digitizéra do počítača alebo notebooku. Rovnako používa aj ovládače a softvér Wacom na interpretáciu nakreslených čiar z dotykového pera. Takýmto spôsobom pero komunikuje priamo s tabletom. Tablety tejto značky umožňujú nastaviť úpravy na kreslenie aj v rôznych aplikáciách, ako napríklad Adobe Photoshop alebo AutoCAD. Rovnako disponujú aj funkciou viacdotykového ovládania, čo umožňuje vykonávať úlohy, ako je približovanie a oddaľovanie alebo otáčanie rýchlo a efektívne [27].

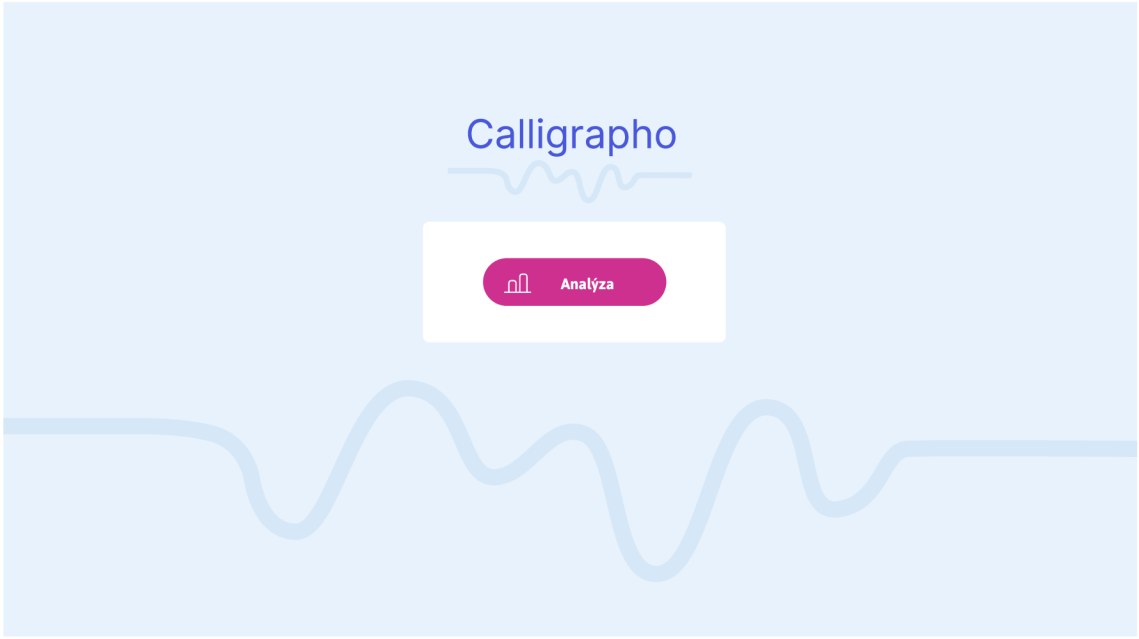
2 Praktická časť študentskej práce

Táto kapitola sa bude venovať tvorbe webovej aplikácie. Okrem spôsobu jej navrhovania sa zrealizuje programovanie jej vizuálu a takisto aj programové riešenie jej funkčnosti s prihliadnutím na princípy jej fungovania.

2.1 Vizuálny návrh webovej aplikácie

Pred procesom tvorby samotnej aplikácie bolo najprv potrebné vytvoriť návrh, na základe ktorého bola potom aplikácia programovaná. Tento návrh bol tvorený v programe Figma a bol inšpirovaný aplikáciou DiaGraMo, ktorej autorom sú členovia výskumnej skupiny BDALab, ktorí sa zameriavajú na výskum a vývoj digitálnych biomarkerov.

Táto aplikácia bola pomenovaná Calligrapho, čo je pretvorený názov pre umenie súvisiace s písaným písmom. Skladá sa z dvoch gréckych slov, a to kallos ako krása a graphein ako písanie. Jej návrh sa skladá z jednej úvodnej stránky a druhej podstránky. Úvodnú stránku je možné vidieť na obrázku 2.1. Okrem jej už vyššie vysvetleného názvu sa na úvodnej stránke nachádza aj logo v tvare vlnky, ktoré symbolizuje jednoduchý znak, ktorý sa deti učia písať pri prvej príležitosti na základnej škole. Takisto sa na úvodnej stránke nachádza tlačidlo s názvom „Analýza“, ktoré užívateľa poprosí o nahranie súboru, ktorý by chcel vizualizovať. Posledný element úvodnej stránky je väčšia vlnka, ktorá sa nachádza v spodnej časti aplikácie.



Obr. 2.1: Návrh aplikácie: Úvodná stránka

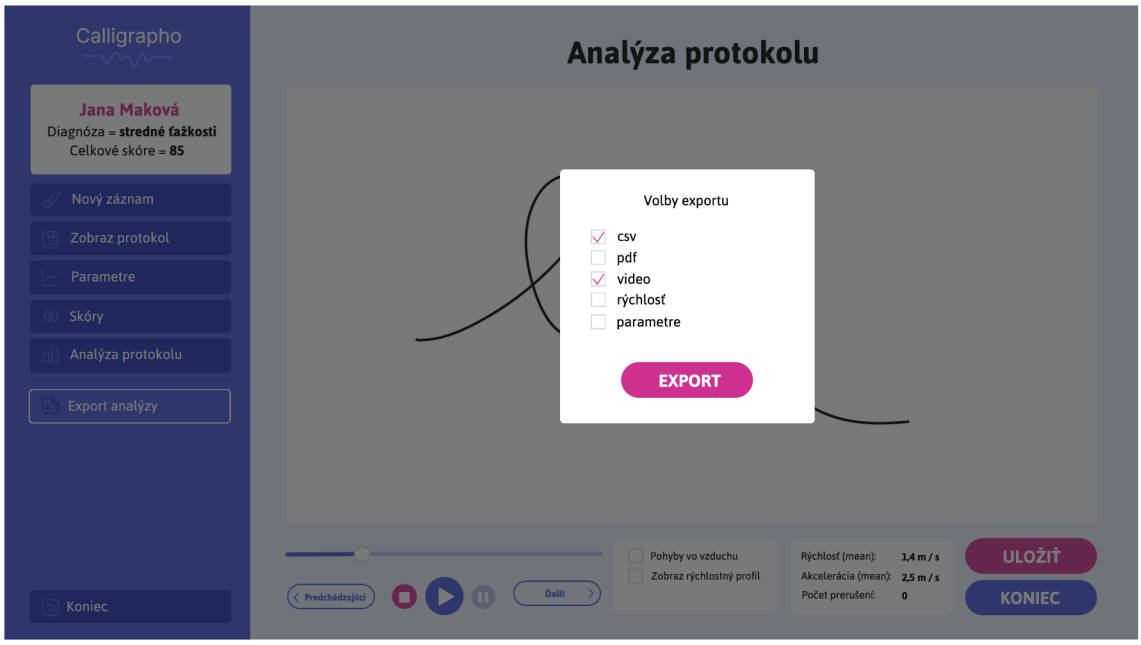
Na obrázku 2.2 je možné vidieť podstránku. Na tejto podstránke sa zobrazuje už samotná vizualizácia. Podstránka sa delí na dve časti. V ľavej časti sa nachádza menu a v pravej sa nachádza plocha spolu s nástrojmi na vizualizáciu. Menu sa rovnako delí na dve časti, prvá časť je biely obdlžník, v ktorom sa nachádzajú základné údaje o osobe, ktorá písmo písala. V druhej časti sa nachádzajú tlačidlá. Celý tento projekt je súčasťou väčšieho systému výskumnej skupiny BDALab. Z toho dôvodu sa v aplikácií nachádzajú aj tlačidlá, ktoré sa v tomto projekte využívať nebudú. Na pravej strane sa nachádza plocha, na ktorú sa bude písmo vizualizovať. V jej spodnej časti sa nachádza prehrávač, pomocou ktorého bude možné písmo zobrazovať po časových stopách.

Obr. 2.2: Návrh aplikácie: Zobrazenie dát nad povrchom tabletu

Takisto sa v spodnej časti nachádzajú aj zaškrtávacie možnosti, ktoré umožňujú zobraziť napríklad pohyb pera nad plochou tabletu, ktorý je vo vizualizácii zobrazený oranžovou farbou. Stránka obsahuje okrem iného aj informácie o priemerných hodnotách rýchlosti písania, zrýchlenia a rovnako aj počtu prerušení. V neposlednom rade sa tam nachádzajú aj tlačidlá, pomocou ktorých je súbor možné uložiť, či aplikáciu zatvoriť. Možnosti exportu, či uloženia sa po kliknutí na tlačidlá zobrazia vo vyskakovacom okne, ktoré je možné vidieť na obrázku 2.3.

2.1.1 Implementácia vizuálnej časti aplikácie

Realizácia vyššie spomínaného návrhu bola tvorená pomocou HTML a CSS v programe Brackets. Ako už bolo vyššie spomenuté, táto webová aplikácia sa delí na dve

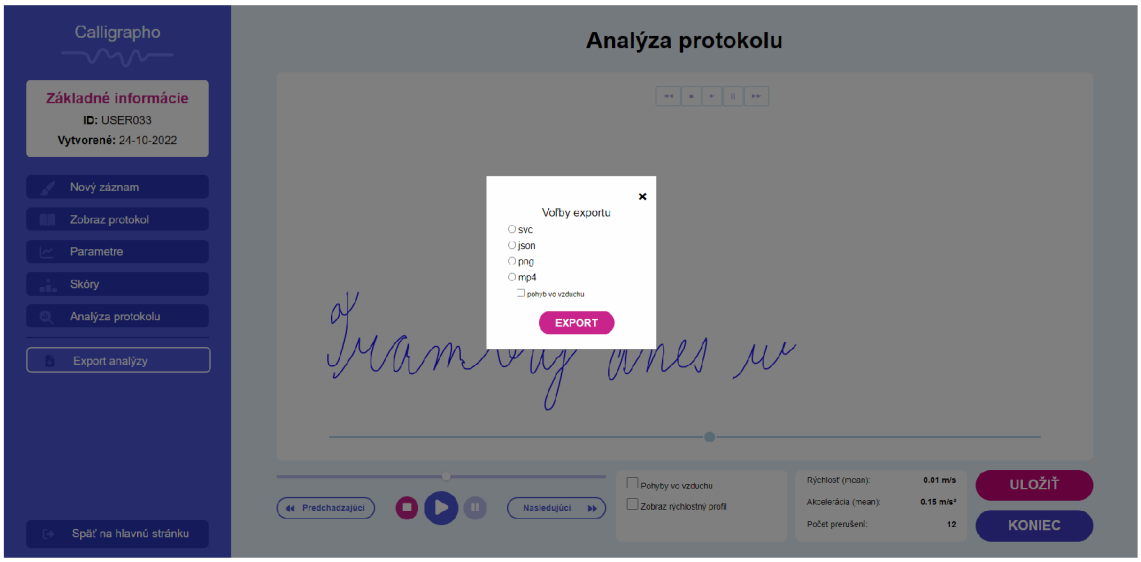


Obr. 2.3: Návrh aplikácie: Spôsoby exportu dát

stránky, teda dva HTML súbory. Prvou z nich je úvodná stránka, na ktorej sa nachádzajú už zmienené prvky ako názov stránky, logo v tvare vlnky, tlačidlo, ktoré sa v konečnom dôsledku nazýva „Vyberte súbor pre analýzu“, ako aj väčšia vlnka na spodnej časti stránky. Táto úvodná stránka bola tvorená s jednoduchým vizuálom z toho dôvodu, aby bolo uživateľovi jasné, aké kroky musí spraviť na to, aby mohol dáta vizualizovať.

Na druhej stránke, na ktorej sa vykonáva samotná vizualizácia, sa pochopiteľne nachádza viac prvkov. Vzhľadom na metadáta, ktoré priložené súbory obsahujú, sa musela zmeniť funkcia bieleho obdlžníka, ktorý sa nachádza v menu časti vľavo hore. Nenachádzajú sa v ňom už informácie o osobe, ktorá písmo písala, ale informácie o súbore ako jeho identifikačné číslo a dátum jeho vytvorenia. Celé menu vľavo nie je tvorené pomocou tlačidiel ako by sa to na prvý pohľad mohlo zdať. Keďže je táto práca súčasťou väčšieho systému, funkcie, ktoré sú v tejto práci využívané, sú vo forme tlačidiel, za to funkcie, ktoré v tejto práci využívané nie sú, sú tvorené pomocou div tagov. Aby neboli tlačidlá dizajnom príliš jednoduché, spolu s názvom tlačidiel sa v nich nachádzajú aj ikony, ktoré sú z externého servera. V pravej časti sa okrem plochy na samotnú vizulizáciu nachádza funkcia prehrávača, ktorú doplňajú okná obsahujúce tlačidlá na zobrazovanie doplnkových kriviek grafu a informácie o vlastnostiach rukopisu. Doplnkové krivky ako zobrazenie pohybov vo vzduchu sú riešené pomocou zaškrtávacieho poľa. V procese tvorby funkčnej časti stránky bolo zistené, že existujúce prvky prehrávača na manipuláciu grafu tvorené v jazyku HTML nevykonávali funkčnosť kódu. Z toho dôvodu boli tieto tlačidlá doplnené

tlačidlami jazyka Python, ktoré sa nachádzajú vo vnútri vizualizovaného písma. Pôvodné tlačidlá boli na stránke ponechané, keďže v budúcnosti by ich funkcionalita mohla využiť plný potenciál. Takisto sa v spodnej časti nachádzajú tlačidlá na uloženie súboru či skončenie aplikácie.



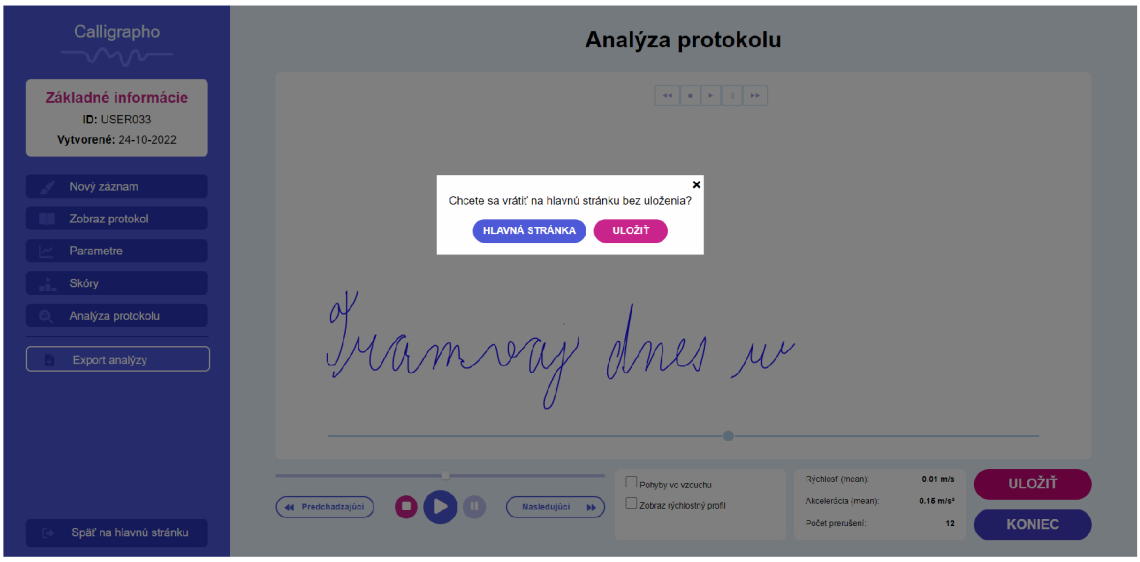
Obr. 2.4: Webová aplikácia: Pop up okno pri exporte

Keďže má uživateľ možnosť uložiť či exportovať súbor do rôznych formátov ako svc, json, png a mp4, po kliknutí na tlačidlá uloženia či exportu sa mu zobrazí vyskakovacie okno, ktoré je možné vidieť na obrázku 2.4. Užívateľ má takisto možnosť výberu, v ktorom z týchto formátov chce mať zobrazený aj pohyb pera nad plochou tabletu. Pre použitie aplikácie v zariadeniach ako počítače, notebooky či tablety, ktoré majú rôzne rozlíšenia, bola aplikácia tvorená s reponzívnym dizajnom, ktorý bol definovaný pomocou media dotazov.

Pri voľbe vrátiť sa späť na úvodnú stránku dostane užívateľ vďaka vyskakovaciemu oknu na obrázku 2.5 možnosť výberu, či by chcel súbor pred odchodom uložiť alebo sa na hlavnú stránku vráti bez uloženia.

2.2 Návrh a implementácia funkčnej časti aplikácie

Po úspešnom navrhnutí a zostrojení vizuálu aplikácie bolo potrebné tomuto vizuálu dodať funkčnosť a teda premyslieť, ktoré súčasti aplikácie spolupracujú. Ako už bolo vyššie spomenuté, vstupné dáta, ktoré aplikácia vizualizuje, boli vyprodukované digitalizačným tabletom Wacom. Aplikácia bola síce vytváraná podľa návrhu BDALab skupiny, obsahuje však iba základné funkcie, ktoré majú potenciál byť v budúcnosti dotvorené do väčších projektov. Na celkovú vizualizáciu bola využitá kľúčová



Obr. 2.5: Webová aplikácia: Pop up okno pri vrátení sa na úvodnú stránku

knižnica HandwritingSample, ktorá svojimi funkciami umožnila transformovať dáta do takej podoby, aby bolo možné ich zobraziť na ploche aplikácie. Na vizualizáciu dát bola potrebná aj knižnica Plotly, ktorá sa používa v rámci kódu na serveri, ale aj v JavaScriptovej forme Plotly.js na strane užívateľa. Plotly.js sa využíva na manipuláciu grafu v reálnom čase bez potreby posielania požiadaviek na server. Plotly rovnako zabezpečilo responzivitu vizualizovaného grafu, keďže HTML a CSS príkazy na responzivitu aplikácie sa na dynamický graf nevzťahujú. Okrem samotného vizualizovania by mala aplikácia podporovať možnosť exportu, ktorý je umožnený prostredníctvom knižnice Matplotlib na konvertovanie dát do videosúboru vo formáte mp4 a obrázka vo formáte png. Aby bol užívateľ schopný interagovať aj s dynamickými časťami aplikácie ako sú formuláre či okná na výber súboru, bolo potrebné zakomponovať aj JavaScript. Požiadavky posielané medzi jednotlivými časťami kódu sú vo forme GET a POST, napríklad pri dátach z formulárov. Keďže je aplikácia funkčná len na lokálnom serveri, na jej fungovanie bol použitý framework Django, ktorý v sebe zahŕňa aj vývojový server. Samotný Django framework združuje všetky komponenty potrebné na vývoj aplikácie v prehľadnej adresárovej štruktúre, čo je hlavným dôvodom výberu tohto frameworku. Aplikácia bola teda vyhotovená tak, aby sa súbor na vizualizáciu poslal do aplikácie len raz a užívateľ s ním bude vedieť manipulovať priamo v nej bez nutnosti komunikácie so serverom. Toto riešenie ušetrí čas a takisto aj plynulosť aplikácie. Aplikácia bohužiaľ pri veľkom objeme dát načítava dáta na vizualizáciu veľmi dlho, čo je ošetrené podmienkou obmedzenej veľkosti súboru na 500 kB. Toto obmedzenie nie je záväzné pre používanie aplikácie v budúcnosti, v tejto práci je limit nastavený z praktického hľadiska, aby sa aplikácia nenačítavala veľmi dlhú dobu.

Pri vývoji aplikácie bolo nevyhnutné stanoviť, aké funkcionality a vlastnosti aplikácia bude obsahovať. V rámci tohto procesu sa podrobne analyzovali požiadavky a potreby cieľovej skupiny používateľov, ako aj technické možnosti a obmedzenia vývojového prostredia. Jedným z hlavných kritérií pri výbere funkcionalít bolo zabezpečiť, aby aplikácia spĺňala požiadavky a potreby BDALab. V rámci výberu funkcionalít bol kladený dôraz na spracovanie a vizualizáciu dát z digitalizačných tabletov, najmä tabletov spoločnosti Wacom, a to v podobe, ktorá umožní ich následnú analýzu.

Funkcionalita a vlastnosti spomenuté v bodoch:

- **Uloženie vstupných dát na server** - užívateľ po návšteve úvodnej stránky zvolí vo vyskakovacom okne súbor, ktorý by chcel vizualizovať a následne analyzovať, tento súbor sa pošle na server a uloží sa na ňom pre ďalšie spracovanie,
- **Poznávanie chýb v súboroch** - v prípade, že užívateľ chce nahrať súbor, ktorý nie je v podporovanom formáte (svc, json), aplikácia ho na to upozorní a rovnaká správa existuje aj v prípade, že chce nahrať súbor nevyhovujúcej veľkosti,
- **Konvertovanie dát súboru** - aplikácia reaguje na to, v akom formáte jej užívateľ poskytol, v prípade, že dáta ostali v pôvodnom formáte nepoužiteľnom pre spracovanie, server to zistí a konvertuje dáta na použiteľné údaje,
- **Vizualizácia grafu v aplikácii** - implementácia funkcií, ktoré sú zahrnuté v hlavnom kóde a exportovanie grafu do aplikácie, taktiež ak sú prítomné metadáta (autor, dátum vytvorenia), sú zobrazené v osobitnej časti aplikácie,
- **Manipulácia grafu** - aplikácia v sebe zahŕňa video prehrávač a jeho prvky, ktorými vie manipulovať vykresľovanie grafu v závislosti od času, taktiež obsahuje aj možnosť zobrazovania doplnkovej krivky z dát „vo vzduchu“, čiže vtedy, keď pero nebolo prítomné na ploche tabletu,
- **Zobrazovanie hodnôt z dát** - na základe vstupných dát kód na serveri vypočítava priemernú rýchlosť, akceleráciu a počet prerušení, ktoré zobrazuje v osobitnej časti aplikácie
- **Exportovanie grafu** - ako poslednú funkcionalitu aplikácia ponúka možnosť exportovania grafu v štyroch rôznych podobách a to vo formátoch svc, json, png a mp4, vysvetlené to bude v ďalšej časti práce

2.2.1 Server a nástroje na spustenie

Ako prvé je potrebné vysvetliť, ako aplikácia funguje. Využitie nástrojov spomenuté bolo, avšak je potrebné spomenúť aj ako sa ku konkrétnym krokom dostať. Prvým krokom je mať spustený Django server, ktorý sa inštaluje pomocou balíčka „pip“. Django funguje formou projektov, ktoré majú svoju adresárovú štruktúru. Za pred-

pokladu, že užívateľ stiahne celú aplikáciu vo forme projektu, je potrebné nastaviť nastavenia projektu na svoje zariadenie, aby aplikácia fungovala. Konkrétny prístup je spomenutý v prílohe pri adresárovej štruktúre elektronickej prílohy. Na to, aby mohol užívateľ spustiť server a aplikáciu samotnú je potrebné napísať príkaz „python manage.py runserver“ v príkazovom riadku na takom mieste, kde má súbor manage.py, čo je vo väčšine prípadov samotný priečinok projektu. Po zapnutí servera by sa mala zobraziť takáto správa:

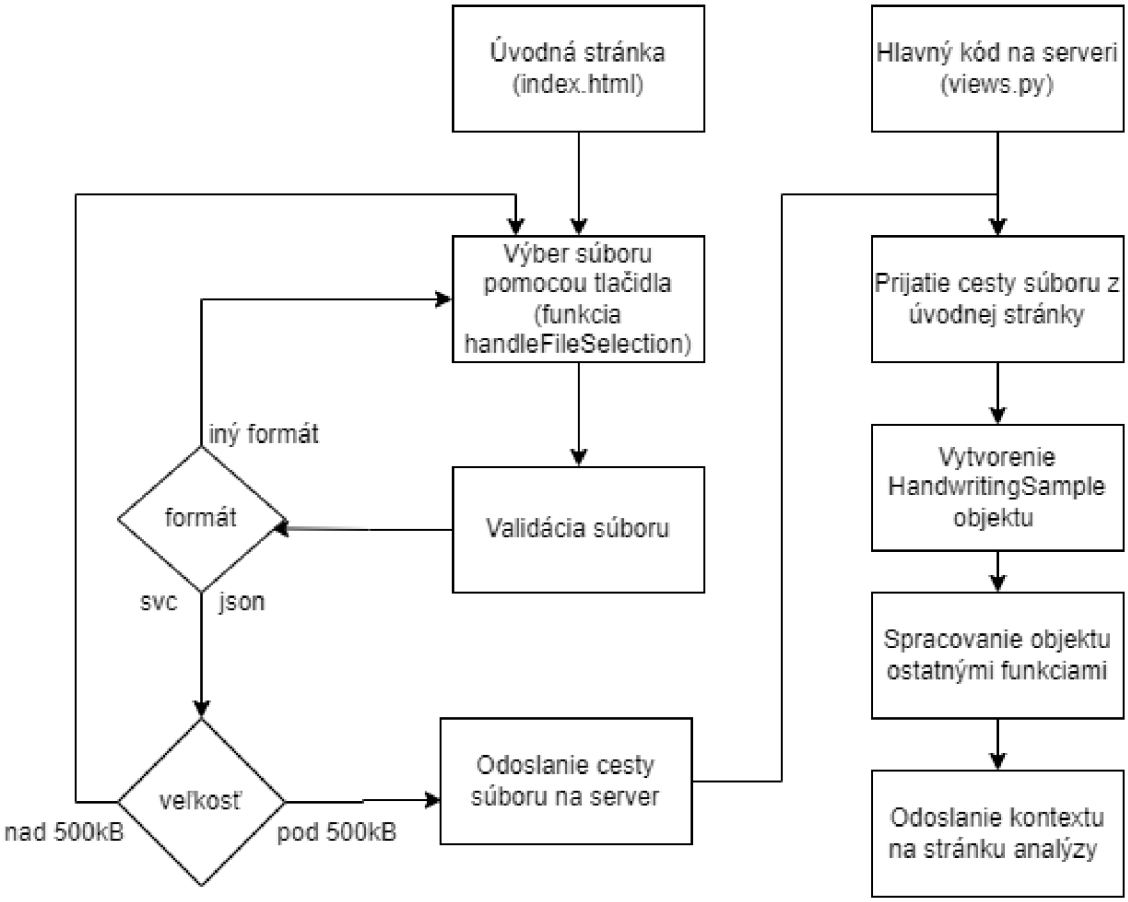
```
System check identified no issues (0 silenced).
May 22, 2023 - 20:36:10
Django version 4.2, using settings 'stranka.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Ak sa server nespustil, znamená to chybu buď v adresárovej štruktúre alebo v kóde Python. Ako je možné vidieť aj na obrázku, server sa nachádza na adrese 127.0.0.1:8000, čo dokazuje, že je to lokálny server. V tomto konkrétnom prípade by sa po načítaní tejto adresy mala zobraziť úvodná stránka index.html. Okrem servera je potrebné mať aj nainštalované knižnice a samotný jazyk Python na spustenie programu. Knižnice, ktoré boli primárne využité sú NumPy, Plotly a Matplotlib, rovnako ako aj podstatná knižnica HandrwritingSample, ktorá prevádza dáta zo súboru na spracovanie.

2.2.2 Komunikácia medzi časťami aplikácie

Aplikácia obsahuje niekoľko stavov z hľadiska komunikácie. Odosielanie požiadavky s cestou vybratého súboru z úvodnej stránky na server je prvou požiadavkou, ktorú vykoná užívateľ po tom, ako príde na stránku a vyberie si súbor na analýzu. Tento krok je potrebný preto, že funkcie na spracovanie dát z knižnice HandrwritingSample majú ako vstupný argument práve túto cestu. V HTML kóde na úvodnej stránke index.html je definované tlačidlo, ktoré je prepojené s JavaScript udalosťou onclick. Táto udalosť spúšťa metódu handleFileSelection(form). V JavaScript metóde handleFileSelection(form) sa získa odkaz na vybraný súbor z formulára pomocou form.elements['file']. Po získaní odkazu na súbor sa vykonáva validácia súboru, ako je kontrola veľkosti a prípony. Ak je súbor platný, zavolá sa metóda form.submit(), ktorá odosiela formulár so súborom na server a po úspešnom odoslaní je užívateľ presmerovaný na stránku analýzy templates.html. Prijatie súboru na serveri a odoslanie požiadavky na stránku je dôležitá súčasť aplikácie, nakoľko sa tam vykonáva väčšina funkcionality. V serverovej časti je definovaný pohľad (view) s názvom my_view, ktorý sa volá pri príchode požiadavky. Ak požiadavka obsahuje sú-

bor a je typu POST, za pomoci request.FILES sa získa prítomný súbor s kľúčom „file“ pomocou request.FILES['file']. Získaný súbor sa spracuje na serveri. Používa sa open(file_path, 'wb') na otvorenie a zápis súboru na disk. Následne sa vykonáva spracovanie súboru na základe jeho prípony (.svc alebo .json). Spracované dáta sa ukladajú do premennej data, ktorú už ďalej spracovávajú jednotlivé funkcie v kóde na serveri. Táto komunikácia je zobrazená na obrázku 2.6.



Obr. 2.6: Komunikácia medzi úvodnou stránkou a serverom

Na konci metódy my_view je použitá funkcia render, ktorá slúži na vygenerovanie odpovede na zadaný template (šablónu) s poskytnutým kontextom. V kontexte sú definované premenné, ktoré budú dostupné v šablóne templates.html pre zobrazenie spracovaných dát. Existujú rôzne premenné, ktoré sú zobrazené na základe jednotlivých podmienok.

Základný kontext, ktorý sa posiela na stránku templates.html je definovaný vo výpise 2.2.2.

```
context = {
    'graph': graph,
    'average_speed': average_speed,
    'average_acceleration': average_acceleration,
    'interruptions_count': interruptions_count,
}
```

Kontext obsahuje premennú graph, ktorá bude vyexportovaná pomocou knižnice plotly.io na stránku, premenné average_speed a average_acceleration obsahujú dáta z výpočtov vo funkcii calculate_speeds, ktoré budú zobrazené v stránke templates.html vo vyhradených priestoroch. Posledná premenná interruptions_count je výsledkom funkcie count_interceptions, ktorá zdieľa miesto na stránke s premennými súvisiacimi s rýchlosťou. Tento kontext však nie je konečný, nakoľko sa k nemu pridávajú metadáta zo súboru, ktoré budú zobrazené v navigačnom paneli na stránke. V prípade, že vstupný súbor obsahuje metadáta, skontroluje sa podmienka, či obsahujú položku „participant“, ktorá reprezentuje meno osoby, ktorej písmo sa skúmalo a „created on“, teda dátum vytvorenia vzorky.

Výpis 2.1: Pripájanie metadát do kontextu

```
meta = data.meta
if 'participant' in meta and 'created_on' in meta:
    ide = meta['participant']['id']
    created_on = meta['created_on']
    context['ide'] = ide
    context['created_on'] = created_on
```

V prípade, že metadáta neobsahujú tieto prvky sa v súbore templates.html vypíše správa o neprítomnosti metadát. Funkcia my_view() je zaslaná späť klientovi v podobe HTML stránky, na ktorej môže klient interagovať s grafom. Stránka je lokalizovaná na adrese http://127.0.0.1:8000/my-view/, kde /my-view/ je atribút pripojený k základnej localhost adrese, ktorá zobrazuje úvodnú stránku. Je potrebné aby bola funkcia posielajúca render zapísaná v súbore urls.py, pretože ak by definovaná nebola, zobrazila by sa chyba práve o tom, že URL nie je definovaná.

2.2.3 Spracovanie a validácia vstupných dát

Dôležitou súčasťou aplikácie je spracovanie vstupných dát, nakoľko sa dáta používajú skoro v každej funkcii aplikácie. Knižnica ponúka viaceré funkcie, ktoré vedia dáta segregovať, vizualizovať, validovať, transformovať, vie ich uložiť do nového súboru a mnoho iných funkcionalít. Spracovanie samotných dát sa vykonáva vo funkcii

my_view(), ktorá prijíma POST požiadavku z úvodnej stránky a z požiadavky použije absolútnu cestu súboru, ktorú potrebujú funkcie triedy HandwritingSample na spracovanie dát, tými funkciami sú:

```
from_svc(cls, path, columns=None, validate=True):
from_json(cls, path, columns=None, validate=True):
```

Tieto funkcie z dát vo formáte svc/json vytvoria tzv. „HandwritingSample“. Tento objekt má svoje charakteristiky, vstupné dáta premieňa na jednotlivé zoznamy, konkrétne ide o :

- x: os X,
- y: os Y,
- time: časové razítko od počiatku epochy UNIX,
- pen_status: stav pera (0 = vo vzduchu, 1 = na povrchu),
- azimuth: azimut hrotu pera,
- tilt: sklon pera vzhľadom k povrchu tabletu,
- pressure: tlak.

Toto začlenenie do zoznamov uľahčí prácu s dátami, keďže sa dajú selektovať, napríklad vo formáte svc_sample.x, čo reprezentuje zoznam bodov na osi x pre objekt HandwritingSample. Pre prácu s ostatnými funkciami bola vytvorená funkcia prepare_data(), ktorá vytvára 4 zoznamy pre každý stav pera a os, teda x_true, y_true, x_false a y_false.

Popri spracovávaní je nutné spomenúť, že aplikácia dáta validuje a to takým spôsobom, že kontroluje veľkosť a formát vstupného súboru. V prípade, že sa na stránku chce nahrať súbor v inom formáte ako svc/json a s veľkosťou menšou ako 0kB, respektíve väčšou ako 500kB, úvodná stránka spustí vyskakovacie okno s chybovou hláškou. Okrem tejto validácie na úvodnej stránke prebieha validácia aj po prijatí dát v hlavnom kóde na strane servera, ktorá kontroluje v akej forme dáta prišli. Pre prácu s dátami je dôležité aby boli v správnej forme a na to sa využíva funkcia transform_all_units() z knižnice HandwritingSample, ktorá zmení hodnoty osí x,y na milimetre, čas na sekundy a uhly na stupne. Dôležité je, aby sa nespustila validácia už validovaných dát a preto je v kóde zahrnutá táto podmienka:

```
if svc_sample.time[1] >= 10**8:
    svc_sample.transform_all_units()
```

Ona zaručuje, že sa transformácia dát vykoná len vtedy, ak je časový atribút vstupných dát uvedený v UNIX čase. Predpoklad, že by ostatné vstupné dáta boli už prekonvertované a čas bol v UNIXe je pomerne nereálny a preto sa validácia dát vykonáva na základe času, keďže je to najjednoduchší spôsob implementácie.

2.2.4 Výpočty súvisiace s dátami

Keďže aplikácia zobrazuje atribúty súvisiace s grafom, bolo potrebné vytvoriť funkcie, aby sa atribúty ako priemerná rýchlosť, priemerná akcelerácia a počet prerušení dali zobraziť. Slúžia na to funkcie calculate_speeds() a count_interceptions(), ktoré vypočítajú rýchlosť, akceleráciu a počet prerušení. Funkcia calculate_speeds vychádzala z predpokladu, že vzorec na výpočet priemernej rýchlosti je

$$v_p = \frac{\Delta s}{\Delta t}, \tag{2.1}$$

kde priemerná rýchlosť (označená Vp) je definovaná ako pomer zmeny vzdialenosti (označenej Δs) k zmenšeniu času (označenej Δt). Vzorec vyjadruje, aká veľká vzdialenosť je prekonaná za jednotku času. Pre výpočet priemernej akcelerácie sa používa vzorec

$$a = \frac{\Delta v}{\Delta t}, \tag{2.2}$$

kde priemerná akcelerácia (označená a) je definovaná ako pomer zmeny rýchlosti (označenej Δv) k zmenšeniu času (označenej Δt). Vzorec vyjadruje, aká veľká zmena rýchlosti nastane za jednotku času.

Z hľadiska kódu to vyzerá takto:

```
def calculate_speeds(data):
    # Výpočet časového intervalu
    time = data.time[-1] - data.time[0]

    # Výpočet celkovej vzdialenosti
    distances = np.sqrt(np.diff(data.x)**2
    + np.diff(data.y)**2)
    total_distance = np.sum(np.abs(distances)) / 1000

    # Výpočet rýchlosti
    average_speed = round(total_distance / time,2)

    # Výpočet počiatočnej a konečnej rýchlosti
    initial_speed = distances[0] /
    (data.time[1] - data.time[0])
    final_speed = distances[-1] /
    (data.time[-1] - data.time[-2])
    average_acceleration =
    round((final_speed - initial_speed) / time,2)
```

```
    return average_speed, average_acceleration
```

2.2.5 Vizualizácia dát

Funkcia count_interceptions vypočíta počet prerušení, teda kedy pero prerušilo kontakt s tabletom. Táto funkcia bola vypočítaná pomocou cyklu, ktorý prechádza cez celú dĺžku súboru na základe atribútu pen_status a tam, kde nastane sekvencia údajov True a False sa pripočíta inštancia do premennej count. Premenná count a výsledné premenné v oblasti rýchlosti sú prostredníctvom funkcie my_view() posielané na stránku analýzy, kde sú zobrazené v časti pod grafom.

Funkcia, ktorá je zodpovedná primárne za vizualizáciu sa nazýva plot_data(). Využíva knižnicu Plotly a jej triedu graph_objects, ktorá vytvára samotný graf na základe použitých funkcií:

```
x_true, y_true, x_false, y_false = prepare_data(data)
frame_count = len(data.x) // 20 # Počet framov
num_frames = len(data.x)
frame_length = num_frames // frame_count
frames = []
for i in range(frame_count):
    end_index = min((i + 1) * frame_length, num_frames)
    scatter_true = go.Scatter(x=x_true[:end_index],
    y=y_true[:end_index], mode='lines', line=dict(color=
       color_true,smoothing=1,shape='spline'))
    scatter_false = go.Scatter(x=x_false[:end_index],
    y=y_false[:end_index], mode='lines', line=dict(color=
       color_false,smoothing=1,shape='spline'))
    # Nastavenie viditelnosti krivky v závislosti od
       indexu
    if i == 0 and i < 1:
        scatter_true.visible = False
    else:
        scatter_true.visible = True

    frame_data = [scatter_true, scatter_false]
    frame = go.Frame(data=frame_data, name=f"Frame {i+1}"
       )
    frames.append(frame)
```

Táto časť kódu je zodpovedná za vytvorenie kriviek pomocou príkazu scatter triedy graph_objects, ktorý prechádza cez všetky inštancie bodov v zoznamoch pre osi x a y. Implementácia je taká, že krivka, ktorá zobrazuje dáta na povrchu bude zobrazená vždy a krivka, ktorá zobrazuje dáta vo vzduchu je skrytá. V uvedenom kóde podmienka o viditeľnosti krivky chýba, avšak je zahrnutá v JavaScript časti kódu, keďže zobrazovanie, respektíve skrývanie krivky prebieha len na klientskej strane. Okrem kriviek sa vytvárajú časové rámce, ktoré budú pomáhať manipulovať graf. Tieto rámce sú dôležité najmä pre tlačidlá na prehrávanie grafu a pre posúvač, ktorým sa dá taktiež zobrazovať rôzne časti grafu podľa času. Tlačidlá a posúvač mali byť pôvodne elementmi HTML, ktoré na stránke aj sú prítomné, avšak to, aby fungovali vo vzťahu s grafom sa nepodarilo uskutočniť. Na základe toho sú vytvorené tlačidlá a posúvač pomocou Plotly, ktoré priamo manipulujú graf a sú súčasťou exportovanej časti kódu na stránke analýzy. Funkcia vytvára nakoniec figúru pomocou príkazu figure triedy graph_objects a posiela ho funkcii my_view ako argument na export do stránky analýzy.

V rámci vizualizácie boli použité aj funkcie, ktoré slúžia na vyhotovenie exportovaných dát vo formátoch mp4 alebo png. Na tie bola použitá knižnica matplotlib. Pre video vo formáte mp4 je využívaná trieda „Matplotlib.animation“, ktorá je ľahko použiteľná a aj nastaviteľná pre vlastné potreby. Pri vytváraní výstupného videa sa používa kodek „ffmpeg“, pri ktorom je dôležité spomenúť, že tento kodek je potrebné nainštalovať, aby fungoval export dát v užívateľskom počítači. Export png formátu je bezproblémový, keďže tento formát je podporovaný univerzálne a nevyžaduje inštaláciu žiadneho externého modulu.

V neposlednom rade funkcia my_view() zodpovedá za presun grafu a častí súvisiacich s ním na stránku analýzy. Konkrétne sa na to používa funkcia knižnice plotly.io, ktorá vytvorí z grafu HTML string, ktorý je následne na stránke prečítaný a zobrazený ako graf.

2.2.6 Manipulácia grafu na stránke

Na to, aby vedel užívateľ manipulovať s grafom sa používajú viaceré funkcie, respektíve časti funkcií, ktoré poskytujú možnosť interagovať s vykresleným grafom na stránke analýzy. Aby sa nespomínalo slovo manipulácia len vo všeobecnej rovine, aplikácia v tomto pojme zahŕňa v prvom rade možnosť upravovať pohyb grafu pomocou posúvača. Ako bolo možné vidieť v implementačnej časti vizualizácie aplikácie, na stránke analýzy sa vyskytujú dva posúvače, na posúvanie kriviek sa používa ten, ktorý je súčasťou bielej plochy, kde je graf vykreslený, posúvač pod tým je zahrnutý pre budúce využitie. Posúvač prítomný v stránke analýzy je výstupom funkcie plot_data(), ktorá už bola spomenutá pri vizualizácii dát. Pre fungovanie posúvača

bolo potrebné využiť funkciu, ktorá vytvorila z grafu rámce, ktoré boli následne v posúvači spracované a vyhotovené pomocou metódy „animate“, ktorá z týchto rámcov vytvorí súvislú animáciu. V nasledujúcom kóde je možné sledovať využitie spomínanej metódy animate s ďalšími atribútmi, ako napríklad mód „immediate“, ktorý zabezpečí okamžitú aktualizáciu kriviek pri pohybe posúvača bez nejakého zdržania.

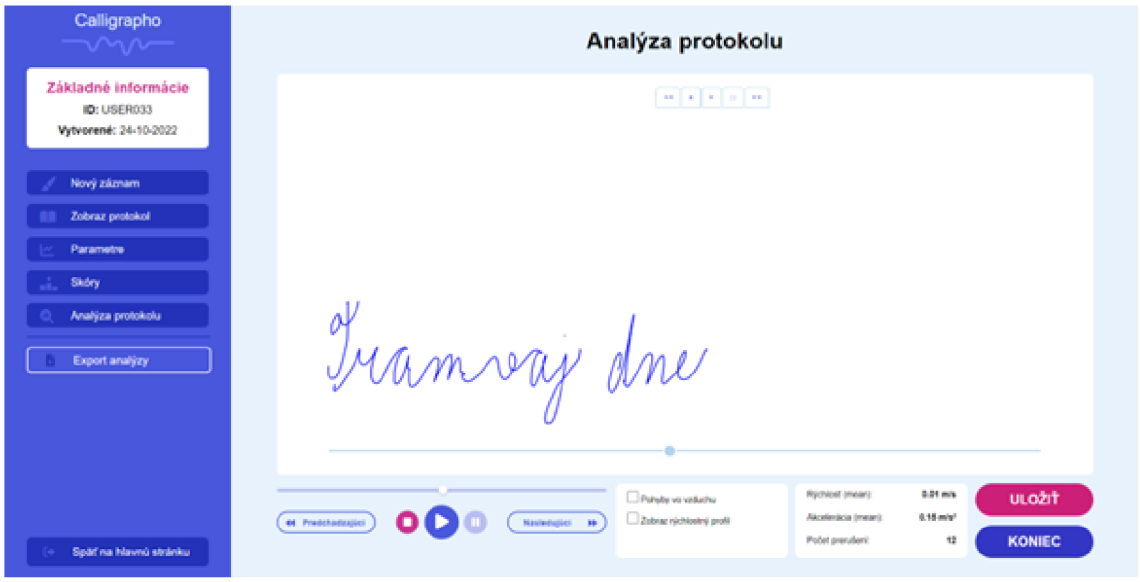
Výpis 2.2: Metóda animate

```
slider = dict(
    steps=[{
        'method': 'animate',
        'label': '',
        'args': [[frame['name']], {'frame': {'duration':
           0, 'redraw': False}, 'mode': 'immediate'}],
    } for i, frame in enumerate(frames)],
```

Okrem posúvača existujú ďalšie prvky, ktoré umožňujú prácu s grafom. Týmito prvkami je myslený samotný média prehrávač vo forme tlačidiel „play“, „stop“, „pause“, „rewind“ a „forward“, ktoré ponúkajú možnosť animáciu spustiť, skončiť, zastaviť, pretočiť späť a pretočiť dopredu. Funkcionalita týchto tlačidiel je založená na podobnej logike ako posúvač, všetky používajú metódu animate, rozdiel je v spôsobe, akým nakladajú s časovými rámcami, ako príklad je uvedený kód pre tlačidlo „pause“:

```
pause_button = dict(
        label="||",
        method="animate",
        args=[[None], {'frame': {'duration': 0, 'redraw':
            False}, 'mode': 'immediate', 'transition': {'
           duration': 0}}]
    )
```

V tomto prípade nie je definovaný rozsah, v akom sa má tlačidlo spustiť, keďže aplikácia chce zaručiť zastavenie prehrávania v bode, v ktorom si to užívateľ želá. Preto je definovaná najmä rýchlosť prehrávania v atribúte „duration“ a dôležitý argument „fromcurrent=True“, ktorý dovolí prerušiť animáciu v bode, v ktorom sa práve posúvač, respektíve animácia nachádza. Na obrázku 2.7 je zobrazená interakcia posúvača s grafom v stránke analýzy, graf reaguje na požiadavku okamžite, čo je veľkou výhodou najmä v prípade náročných používateľov, ktorí budú stránku testovať do svojho limitu.



Obr. 2.7: Webová aplikácia: Interakcia posúvača s grafom

Okrem používania posúvača a tlačidiel aplikácia poskytuje možnosť zobrazovania doplnkových kriviek. V implementácii sa to limitovalo na jednu doplnkovú krivku „Pohyby vo vzduchu“. Na to sa využil nasledovný kód, ktorý implementuje funkcie z knižnice plotly.js, tým pádom nie je zahrnutý do súboru views.py, kde prebieha hlavná logická časť aplikácie, ale v samotnom súbore templates.html, ktorý je šablónou pre stránku analýzy. Kód je vytvorený v krátkom rozsahu, nakoľko logika zobrazovania kriviek nevyžaduje komplexnejšie riešenie.

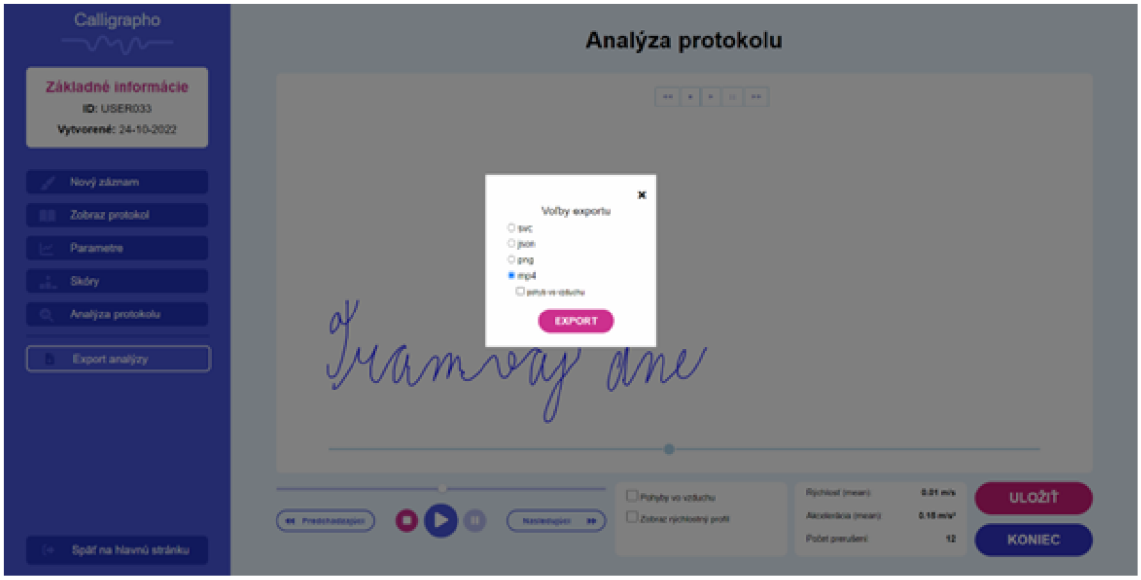
```
document.addEventListener("DOMContentLoaded", function()
   {
    var toggleCheckbox = document.getElementById("
       pohyb_vo_vzduchu");
    var fig = document.getElementById("biela_plocha");
    var traces = fig.data;
    traces[1].visible = false // Skryť krivku pri načítan
       í
    toggleCheckbox.addEventListener("change", function()
       {
      if (toggleCheckbox.checked) {
        traces[1].visible = true;
      } else {
        traces[1].visible = false;
      }
      Plotly.redraw(fig);
```

```
    });
});
```

V praktickej rovine kód nastavuje viditeľnosť krivky, ktorá je pôvodne zo servera vykreslená spolu s prvou, avšak stránka tento stav mení a nechá na užívateľovi, či ju chce mať zahrnutú v zobrazení. Do manipulácie sa dá zahrnúť aj zmena veľkosti okna, graf na zmenu veľkosti reaguje natívne, čo je funkcionalita knižnice Plotly.

2.2.7 Exportovanie grafu

Aplikácia ponúka návštevníkovi možnosť exportovania grafu do ním zvoleného formátu. K zobrazeniu exportového formulára je potrebné kliknúť na tlačidlo „uložiť“, respektíve na tlačidlo „export analýzy“ v navigačnom menu. Ako už bolo vyššie spomenuté, na výber sú 4 formáty a to svc, json, png a mp4, ktoré sú bežne používanými formátmi v tejto oblasti. Pre funkcie png a mp4 navyše existuje možnosť si vybrať, či majú byť dáta exportované v celej forme, alebo len jednotlivé krivky, čo je vidieť na obrázku 2.8.



Obr. 2.8: Webová aplikácia: Formáty exportu

Po zvolení formátu vo formulári stránka žiada meno, pod ktorým sa súbor má uložiť. Ak sa meno nezadá, názov súboru bude prázdny. Po úspešnom odoslaní formulára s názvom a formátom súboru posiela stránka analýzy požiadavku do funkcie my_view, ktorá po prijatí požiadavky vykoná funkciu process_request(), uloží súbor do počítača a pošle požiadavku späť na stránku analýzy o tom, kde sa súbor uložil.

Výpis 2.3: Funkcia na vytvorenie kódu v žiadanom formáte

```
def process_request(request):
    global data
    formaty = request.POST.get('formaty')
    file_name = request.POST.get('file_name')
    if os.name == "nt":  # windows
        file_path = f"{os.getenv('USERPROFILE')}\\
           Downloads"
    else:  # mac/linux
        file_path = f"{os.getenv('HOME')}/Downloads"
    if formaty == 'svc':
        data.to_svc(file_path, file_name, False)
    elif formaty == 'json':
        data.to_json(file_path, file_name, False)
    elif formaty == 'mp4':
        animate_data(data, f"{file_path}\\{file_name}.mp4
           ", request)
    elif formaty == 'png':
        image(data, f"{file_path}\\{file_name}.png",
           request)
    context = {
        'file_path': file_path,
        'redirect': True
    }
    return render(request, 'templates/templates.html',
       context)
```

Pôvodným zámerom bolo použiť HTTP hlavičku „Content-Disposition“ s hodnotou „attachment“, ktorá by prinútila stránku stiahnuť súbor pred očami užívateľa tak, ako je pri iných aplikáciách zaužívané. V prípade tejto aplikácie to však nebolo úplne realizovateľné z toho dôvodu, že sa súbor uložil dvakrát, raz pri vykonaní funkcie na pozadí a raz pri stiahnutí pomocou HTTP požiadavky. Ostal teda spôsob so sťahovaním na pozadí a informovaním užívateľa o lokalite súboru vo forme, ktorá je zobrazená v tomto výpise:

```
Súbor bol uspešne stiahnutý.
Umiestnenie: C:\Users\Vladka\Downloads
```

Závěr

Práca sa zaoberala návrhom a vytvorením webovej aplikácie, vizualizáciou písma, ktoré bolo zaznamenaná digitalizačným tabletom Wacom a jeho jednoduchou analýzou.

V prvej kapitole sa práca venovala definovaniu pojmov súvisiacich s webovými aplikáciami. Rovnako bola v kapitole vyčlenená časť pre technológie súvisiace s vývojom webových aplikácií, z nich bol pre túto prácu najdôležitejší jazyk Python a jeho knižnice, framework Django a taktiež knižnica HandwritingSample, bez ktorej by nebolo možné vizualizáciu zrealizovať. V neposlednom rade boli spomenuté aj programy a jazyky, ktoré boli použité na vytvorenie grafického rozhrania. Nakoniec kapitola charakterizovala pojem online rukopis a digitalizačný tablet Wacom Cintiq 16, ktorý slúžil na zaznamenanie písma.

Druhá kapitola bola rozdelená na dve časti, vizuálnu a funkčnú. Vizuálna časť aplikácie pojednávala o návrhu aplikácie, ktorý bol doplnený jeho implementáciou. Z hľadiska vizualizácie sa dá konštatovať, že väčšinová časť bola vyhotovená podľa pôvodného návrhu, hlavnými rozdielmi je existencia duálneho prehrávača a mierne štylistické zmeny.

Funkčná časť aplikácie bola vytvorená s prihliadnutím na požiadavky, ktoré mala spĺňať. Na začiatok bolo definované, akými vlastnosťami a funkcionalitami má aplikácia disponovať. Na základe toho boli vytvorené jednotlivé funkcie v jazyku Python, ktoré tvoria jadro aplikácie. Medzi hlavné požiadavky aplikácie patrila najmä vizualizácia dát v podobe grafu a jeho manipulácia, spracovávanie dát z úvodnej stránky potrebných na analýzu a taktiež umožnenie exportu grafu do rozličných formátov. Vymenované požiadavky boli splnené v plnom, respektíve mierne obmedzenom rozsahu. Obmedzený rozsah sa týka najmä manipulácie s grafom, nakoľko boli v konečnom dôsledku vytvorené tlačidlá v jazyku Python, aj keď pôvodným zámerom bolo umožniť užívateľovi manipulovať s grafom pomocou HTML tlačidiel. Tento spôsob však nebolo možné zrealizovať. Napriek tomu je funkčnosť manipulácie nezmenená a užívateľ môže graf prehrávať ako video, môže s ním posúvať v smere či proti smeru prehrávania a taktiež je mu umožnené zobrazovať doplnkovú krivku.

Vo všeobecnosti boli ciele práce splnené, aplikácia je responzívna, zobrazuje graf na stránke analýzy vo vyhradenej časti, s grafom sa dá manipulovať v reálnom čase bez potreby komunikácie so serverom, mierne nedostatky je možné badať v rýchlosti načítania grafu, nakoľko logika za tým vyžaduje čas na načítanie a preto je v kóde zahrnuté aj obmedzenie veľkosti. Okrem toho aplikácia obsahuje možnosť exportovania dát a poskytuje informácie o ich analýze. Aplikácia je z praktického hľadiska pre potreby práce úplná, zároveň vytvára priestor na jej vylepšenie do budúcna.

Literatúra

[1] SARHAN, Qusay a Idrees GAWDAN. *Web Applications and Web Services: A Comparative Study*. Science Journal of University of Zakho [online]. Posledná aktualizácia marec 2018 [cit. 2022-10-23]. Dostupné z URL: <https://www.researchgate.net/publication/324106370_Web_Applications_and_Web_Services_A_Comparative_Study>

[2] BRUNO, Vince, Audrey TAM a James THOM. *Characteristics of Web applications that affect usability: A review*. [online]. Posledná aktualizácia 1. január 2005 [cit. 2022-10-23]. Dostupné z URL: <https://www.researchgate.net/publication/221332130_Characteristics_of_Web_applications_that_affect_usability_A_review>

[3] MARTIN, Matthew. *Difference between Website and Web Application (Web App)*. [online]. Posledná aktualizácia 1. apríl 2023 [cit. 2022-10-23]. Dostupné z URL: <https://www.guru99.com/difference-web-application-website.html#:~:text=Summary%3A,accessible%20using%20any%20web%20browser.>

[4] TUMIN, Sharil a Sylvia ENCHEVA. *A Closer Look at Authentication and Authorization Mechanisms for Web-based Applications*. [online]. Posledná aktualizácia 1. január 2012 [cit. 2022-10-23]. Dostupné z URL: <https://www.researchgate.net/publication/250310860_A_Closer_Look_at_Authentication_and_Authorization_Mechanisms_for_Web-based_Applications>

[5] VAN ROSSUM, Guido. *A Brief Timeline of Python*. [online]. Posledná aktualizácia 20. január 2009 [cit. 2022-10-23]. Dostupné z URL: <https://python-history.blogspot.com/2009/01/brief-timeline-of-python.html>

[6] *Python*. [online]. Posledná aktualizácia 2022 [cit. 2022-10-23]. Dostupné z URL: <https://www.python.org/about/apps/>

[7] Tutorials Point. *Python Tutorial*. [online]. Posledná aktualizácia 2017 [cit. 2022-10-23]. Dostupné z URL: <https://www.tutorialspoint.com/python/python_tutorial.pdf>

[8] HALVORSEN, Hans-Petter. *Python Programming*. [online]. Posledná aktualizácia 12. august 2020 [cit. 2022-10-23]. ISBN 978-82-691106-4-7. Dostupné z URL: <https://www.halvorsen.blog/documents/programming/python/resources/Python%20Programming.pdf>

[9] DE LA GUARDIA, Carlos. *Python Web Frameworks*. [online]. Posledná aktualizácia 2016 [cit. 2022-12-23]. ISBN 978-1-491-93810-2. Dostupné z URL: <https://theswissbay.ch/pdf/Books/Computer%20science/O%27Reilly/python-web-frameworks.pdf>

[10] Django Software Foundation. *Django*. [online]. Posledná aktualizácia 2023 [cit. 2022-12-23]. Dostupné z URL: <https://www.djangoproject.com/start/overview/>

[11] MDN contributors. *Django introduction*. [online]. Posledná aktualizácia 24. február 2023 [cit. 2022-12-23]. Dostupné z URL: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>

[12] *Django URLs And Views*. [online]. [cit. 2022-12-23]. Dostupné z URL: <https://www.programink.com/django-tutorial/django-urls-views.html>

[13] Datacamp. *Introduction to Plotting with Matplotlib in Python* . [online]. Posledná aktualizácia marec 2023 [cit. 2022-12-23]. Dostupné z URL: <https://www.datacamp.com/tutorial/matplotlib-tutorial-python>

[14] HAVERBEKE, Marijn. *Eloquent JavaScript: a modern introduction to programming, Third edition.*. [online]. Posledná aktualizácia 2019 [cit. 2022-12-23]. ISBN 978-1593279509. Dostupné z URL: <https://eloquentjavascript.net/Eloquent_JavaScript_small.pdf>

[15] A, Jordana. *What Is JavaScript? A Basic Introduction to JS for Beginners*. [online]. Posledná aktualizácia 31. január 2023 [cit. 2023-01-17]. Dostupné z URL: <https://www.hostinger.com/tutorials/what-is-javascript#What_Is_JavaScript>

[16] GitHub. *Plotly.py*. [online]. [cit. 2022-12-23]. Dostupné z URL: <https://github.com/plotly/plotly.py>

[17] Plotly. *Plotly JavaScript Open Source Graphing Library*. [online]. Posledná aktualizácia 2023 [cit. 2023-01-17]. Dostupné z URL: <https://plotly.com/javascript/>

[18] HARRIS, C.R., K.J. MILLMAN a S.J. VAN DER WALT. *Array programming with NumPy*. [online]. Posledná aktualizácia 16. september 2020 [cit. 2023-01-17]. Dostupné z URL: <https://www.nature.com/articles/s41586-020-2649-2>

[19] Brain Diseases Analysis Laboratory. *Handwriting Sample*. [online]. Posledná aktualizácia 19. november 2022 [cit. 2023-03-17]. Dostupné z URL: <https://github.com/BDALab/handwriting-sample>

[20] POWELL, Thomas A. *HTML & CSS: The Complete Reference, Fifth Edition*. [online]. Posledná aktualizácia 2010 [cit. 2023-03-17]. ISBN 978-0-07-174170-5. Dostupné z URL: <https://www.dcpehvpm.org/E-Content/BCA/BCA-II/Web%20Technology/the-complete-reference-html-css-fifth-edition.pdf>

[21] WILLARD, Wendy. *HTML: A Beginner's Guide, Fourth Edition*. [online]. Posledná aktualizácia 2009 [cit. 2023-03-17]. ISBN 978-0-07-161144-2. Dostupné z URL: <https://www.snggdcg.ac.in/pdf/study-material/computer-science/0071611436%20HTML.pdf>

[22] GOODING, Paul. *Introduction to HTML and CSS*. [online]. Po-sled-ná aktualizácia 2018 [cit. 2023-03-17]. Dostupné z URL: https://chasedigitalage.files.wordpress.com/2018/04/htmlcss.pdf

[23] Figma [online]. [cit. 2023-04-03]. Dostupné z URL: <https://www.figma.com/>

[24] Brackets [online]. [cit. 2023-04-03]. Dostupné z URL: <https://brackets.io/>

[25] Visual Studio Code [online]. [cit. 2023-04-03]. Dostupné z URL: <https://code.visualstudio.com/>

[26] C. C. Tappert, C. Y. Suen and T. Wakahara. *The state of the art in online handwriting recognition*. [online]. Posledná aktualizácia 1990 [cit. 2023-03-17]. ISBN 978-0-07-174170-5. Dostupné z URL: <https://ieeexplore.ieee.org/document/57669>

[27] Wacom [online]. [cit. 2023-04-03]. Dostupné z URL: <https://www.wacom.com/en-us>

Zoznam symbolov a skratiek

Skratky:

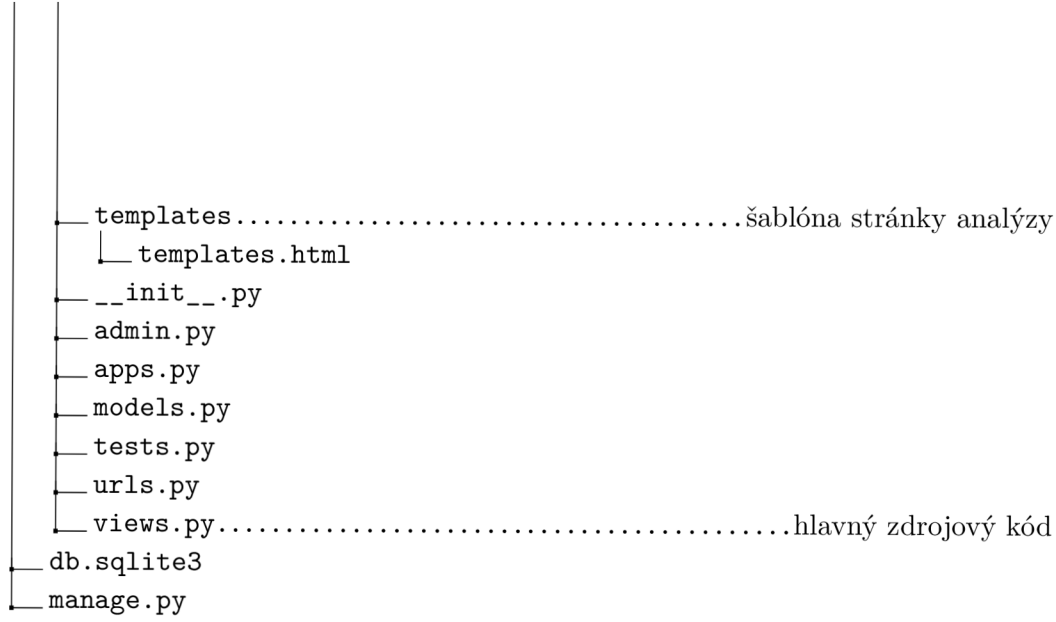
API	Rozhranie pre programovanie aplikácií– Application programming interface
CSS	Kaskádové štýly – Cascading Style Sheets
DOM	Objektový model dokumentu – Document Object Model
ECMA	Európska asociácia výrobcov počítačov – European Computer Manufacturers Association
EMR	Elektromagnetické žiarenie – Electromagnetic radiation
HTML	Hypertextový značkovací jazyk – Hyper Text Markup Language
HTTP	Hypertextový prenosový protokol – Hypertext Transfer Protocol
JSON	JavaScriptový objektový zápis – JavaScript Object Notation
MIT	Massachusettský technologický institut v Cambridge – Massachusetts Institute of Technology
NTSC	Štandard kódovania analógového televízneho signálu – National Television System(s) Committee
URL	Jednotné označovanie objektov – Uniform Resource Locator
XML	Rozšíriteľný značkovací jazyk – Extensible Markup Language

Zoznam príloh

A Obsah elektronickej prílohy

Elektronická príloha obsahuje zdrojové súbory webovej aplikácie pre vizualizáciu dát z digitalizačného tabletu Wacom. Záložka Calligrapho obsahuje zdrojové kódy webovej aplikácie, ktoré môže byť použité pre budúci vývoj. Dôležité nastavenia sa nachádzajú v záložke Calligrapho súbor settings.py a v záložke myapp súbor urls.py. Pre správne spustenie aplikácie je potrebné v súbore settings.py zmeniť 58. riadok na správnu cestu k aplikácii. Samotná aplikácia sa spúšťa z adresára Calligrapho príkazom „python manage.py runserver“ a je dostupná cez localhost 127.0.0.1:8000.

```
/
└── Calligrapho ..... názov projektu
    ├── Calligrapho ..... hlavný priečinok projektu
    │   ├── __pycache__
    │   │   ├── __init__.cpython-311.pyc
    │   │   ├── settings.cpython-311.pyc
    │   │   ├── urls.cpython-311.pyc
    │   │   └── wsgi.cpython-311.pyc
    │   ├── __init__.py
    │   ├── asgi.py
    │   ├── db.sqlite3
    │   ├── settings.py
    │   ├── urls.py
    │   └── wsgi.py
    └── myapp ..... názov aplikácie
        ├── __pycache__
        │   ├── __init__.cpython-311.pyc
        │   ├── admin.cpython-311.pyc
        │   ├── apps.cpython-311.pyc
        │   ├── models.cpython-311.pyc
        │   ├── urls.cpython-311.pyc
        │   └── views.cpython-311.pyc
        ├── migrations
        │   ├── __pycache__
        │   │   └── __init__.cpython-311.pyc
        │   └── __init__.py
        └── static ..... priečinok pre statické súbory
            ├── css
            │   └── main.css
            ├── html
            │   └── index.html
            └── Images
                ├── analazy.svg
                ├── Vector.png
                └── vlnkamala_analyza.svg
```


templates
šablóna stránky analýzy
templates.html
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
hlavný zdrojový kód
db.sqlite3
manage.py