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

UŽIVATELSKÉ ROZHRANÍ PRO RYCHLOU OPRAVU AUTOMATICKÝCH PŘEPISŮ TEXTU

USER INTERFACE FOR EFFICIENT CORRECTIONS OF OCR OUTPUT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PAVOL SZEPSI**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL HRADIŠ, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

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

Zadání bakalářské práce

Student: **Szepsi Pavol**
Program: Informační technologie
Název: **Uživatelské rozhraní pro rychlou opravu automatických přepisů textu**
User Interface for Efficient Corrections of OCR Output
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s přístupy používanými pro přepis textu, hlavně pak s jejich výstupy.
2. Vytvořte si přehled o existujících uživatelských rozhraních pro efektivní kontrolu výstupů automatických systémů pro přepis textu.
3. Navrhněte uživatelské rozhraní, které dokáže využít nejistoty a varianty výstupů OCR metod. Do rozhraní můžete přidat prvky gamifikace. Zaměřte se na vhodnost rozhraní pro mobilní a dotyková zařízení.
4. Implementujte uživatelské rozhraní jako webovou aplikaci použitelnou i na mobilních zařízeních.
5. Vyhodnoťte vlastnosti uživatelského rozhraní v uživatelských testech.
6. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

-
- Thorsten Vobl et al.: PoCoTo-an open source system for efficient interactive postcorrection of OCRed historical texts. DATeCH, 2014.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Hradiš Michal, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 29. července 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Cieľom tejto bakalárskej práce je navrhnúť a implementovať webové užívateľské rozhranie pre kontrolu a opravu výstupov OCR, ktoré bude vhodné pre mobilné a dotykové zariadenia. Užívateľské rozhranie využíva varianty výstupu OCR, ktoré môže užívateľ použiť na modifikáciu rozpoznaného textu. Rozhranie je implementované v jazyku JavaScript s využitím frameworku Vue JS. Pre serverovú časť je použitý balíček XAMPP. Pre komunikáciu medzi užívateľským rozhraním a serverom je použitý nástroj Axios. Vytvorené rozhranie umožňuje užívateľom rýchlo a jednoducho opraviť výstupy OCR, či už na počítači alebo na mobilnom zariadení.

Abstract

The aim of the present bachelor thesis was to design and implement a web user interface for checking and correcting OCR outputs which will be suitable for mobile and touchscreen devices. The user interface uses the OCR output variants that the user can use to modify the recognized text. The interface is implemented in JavaScript using the Vue JS framework. XAMPP package is used for the server part. The tool Axios is used for communication between the user interface and the server. The created interface allows users to quickly and easily correct the OCR outputs, either on a computer or on a mobile device.

Kľúčové slová

užívateľské rozhranie, optické rozpoznávanie znakov, OCR, JavaScript, Vue JS, XAMPP, oprava chýb

Keywords

user interface, optical character recognition, OCR, JavaScript, Vue JS, XAMPP, error correction

Citácia

SZEPSI, Pavol. *Uživatelské rozhraní pro rychlou opravu automatických přepisů textu.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

Uživatelské rozhraní pro rychlou opravu automatických přepisů textu

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Michala Hradiša, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Pavol Szepsi
29. júla 2022

Poďakovanie

Chcel by som poďakovať môjmu vedúcemu práce, pánovi Ing. Michalovi Hradišovi, Ph.D., za čas venovaný konzultáciám a za podnety k mojej práci.

Obsah

Kapitola 1

Úvod

V dnešnej dobe sa všetky dokumenty postupne prevádzajú do digitálnej podoby. V spojení s možnosťami Internetu nám to umožňuje rýchlejší prístup k rozličným dokumentom z celého sveta, vyhľadávanie v nich a prácu s nimi. Dokumenty písané ručne alebo na písacom stroji je však potrebné nejakým spôsobom do tejto digitálnej podoby dostať. Na to slúži metóda optického rozpoznávania znakov (optical character recognition — OCR), ktorá jednotlivé písmená a znaky z obrázkov prekladá do editovateľnej textovej formy. Táto metóda však nie je bezchybná a niekedy sa stane, že na výstupe je viacero možností, z ktorých niektorá je správna. V takýchto prípadoch je potrebné výstup OCR skontrolovať manuálne a opraviť prípadné chyby v preklade znakov.

Cieľom tejto práce bolo vytvoriť webovú aplikáciu s užívateľským rozhraním, ktoré umožňuje rýchlu a jednoduchú opravu chýb OCR výstupov na štandardnom počítači, ale taktiež na mobilných zariadeniach.

Na začiatku práce je predstavená metóda optického rozpoznávania znakov a to, ako približne funguje. Ďalej sú predstavené jednotlivé kroky, ktorými v procese rozpoznávania prechádza a tiež niečo o jej výstupoch. Nasleduje zhrnutie niektorých dôležitých oblastí, kde sa dnes technológia OCR využíva, a ako je v nich využitá. Toto všetko sa nachádza v kapitole 2.

V kapitole 3 sa nachádza prehľad niektorých užívateľských rozhraní aplikácií pre rozpoznávanie textu, v ktorých je možné text následne upravovať. Tieto aplikácie boli tiež otestované, pričom dôraz bol kladený hlavne na spôsob a jednoduchosť úpravy textu v týchto aplikáciách a zistenie, aké sú jeho výhody a nevýhody.

V kapitole 4 sú predstavené technológie, ktoré boli použité pri implementácii webovej aplikácie. Najpodrobnejšie je predstavená technológia Vue JS, ktorá tvorí najdôležitejšiu časť vytvorenej aplikácie. Ďalej sú tu popísané technológie Axios, služba EC2 od Amazon Web Services a XAMPP, ktoré boli použité pri implementácii databázovej časti aplikácie.

Kapitola 5 popisuje vizuálny návrh jednotlivých prvkov užívateľského rozhrania a dôvody, prečo boli prvky navrhnuté daným spôsobom.

V kapitole 6 sú popísané kľúčové prvky, pomocou ktorých bola aplikácia implementovaná. Najprv sú tu preberané dáta, s ktorými aplikácia pracuje, a ich forma. Následne je popísaná štruktúra samotnej aplikácie, implementácia funkcií užívateľského rozhrania aplikácie, ale taktiež komunikácia aplikácie s databázou.

Po vytvorení aplikácie boli vykonané užívateľské testy. Kapitola 7 popisuje formu vykonaných testov, s akými užívateľmi boli testy robené, a nakoniec je vo vyhodnotení testov zhrnutý ich výsledok a zmeny, ktoré boli v aplikácii urobené.

V kapitole 8 sa nachádza zhrnutie práce, jej výsledky a možnosti rozšírenia a ďalšieho vývoja aplikácie.

Kapitola 2

Predstavenie optického rozpoznávania znakov

V tejto kapitole sa nachádza stručné prestavenie OCR[1] a základný prehľad jednotlivých krokov, ktorými OCR algoritmus pri rozpoznávaní textu prechádza. Ďalej sú tu spomenuté niektoré významné oblasti použitia OCR technológií.

Optické rozpoznávanie znakov (ďalej len OCR) popisuje proces prevodu tlačeného alebo ručne písaného textu do digitálneho formátu. OCR je významnou oblasťou výskumu v oblasti umelej inteligencie, rozpoznávania vzorov a počítačového videnia.[5]

OCR pomáha konvertovať obrázky alebo fyzické dokumenty do formy, v ktorej je možné v nich vyhľadávať. Príkladmi OCR sú napríklad nástroje na extrakciu textu, konvertory PDF[2] na .txt alebo funkcia vyhľadávania obrázkov od spoločnosti Google.[5]

Dokumenty môžu byť klasifikované na základe rôznych jazykov ako písané na stroji a ručne písané. Rozpoznávanie teda možno rozdeliť na dva spôsoby — rozpoznávanie tlačených znakov (Printed Character Recognition — PCR) a rozpoznávanie ručne písaných znakov (Handwritten Character Recognition — HCR). PCR je v porovnaní s HCR značne jednoduchšie, pretože počet dostupných štýlov písma (fontov) je v prípade PCR oveľa nižší v porovnaní s rôzne diverzifikovanými štýlmi rukopisu pri HCR. HCR možno tiež rozdeliť do dvoch kategórii — offline rozpoznávanie a online rozpoznávanie. Rozpoznávanie v online režime sa vykonáva v reálnom čase, takže v čase, keď užívateľ píše dokument, zatiaľ čo už napísané dokumenty sa rozpoznávajú v režime offline.[18]

Pri vykonávaní HCR je vyššia pravdepodobnosť vzniku chýb ako pri PCR, a teda sa očakáva viacero potrebných ručných opráv chýb rozpoznaného textu.

2.1 Proces optického rozpoznávania znakov

Celý proces OCR zahŕňa sériu krokov, ktoré obsahujú tri ciele: predbežné spracovanie obrazu (pre-processing), rozpoznávanie znakov a následné spracovanie výstupu (post-processing).[5] Konkrétne kroky procesu OCR sa môžu líšiť v závislosti od použitého OCR algoritmu, avšak nasledujúce kroky sú použité vo väčšine algoritmov:

[1] OCR - optical character recognition(optické rozpoznávanie znakov)
[2] PDF - Portable Document Format (prenosný formát dokumentu)

Získanie obrazu (Image Acquisition)

Toto je prvý krok OCR, kedy je dokument skenovaný pomocou skeneru. Skenovanie dokumentu tým, že štandardizuje vstupy, znižuje počet premenných, ktoré je potrebné zohľadniť pri vytváraní OCR softvéru. Tento krok tiež špecificky zvyšuje efektivitu celého procesu tým, že zabezpečuje dokonalé zarovnanie a veľkosť konkrétneho dokumentu.[5]

Predbežné spracovanie (Pre-processing)

Cieľom tohto kroku je, aby bol vstupný súbor použiteľný pre OCR algoritmus.[1] Predbežné spracovanie zahŕňa veľa procesov a môže sa líšiť v závislosti od daného OCR algoritmu. Tu sú niektoré vybrané kroky, ktoré sú dôležitou súčasťou predbežného spracovania dokumentu:

- **Narovnanie** — Ak dokument nebol pri skenovaní správne zarovnaný, je potrebné ho narovnať, aby sa vytvorili úplne vodorovné alebo zvislé riadky textu.[8]
- **Zjemnenie obrazu** — V tomto kroku prebieha vylepšenie niektorých prvkov dokumentu. Akékoľvek nedokonalosti, ako sú napríklad prachové častice viditeľné na skenovanom dokumente, sú odstránené. Okraje a pixely sú vyhladené, aby sa získal čistý a jasný text. Tento krok uľahčuje OCR programu zachytiť a jasne rozpoznať slová v dokumente.[5]
- **Binarizácia** — V tomto kroku prebieha prevod skenovaného dokumentu do čiernobieleho formátu, kde tmavé oblasti predstavujú znaky a biele oblasti pozadie. Tento krok pomáha rozpoznať rôzne štýly písma.[1] Príklad dokumentu, ktorý prešiel binarizáciou je viditeľný na obrázku 2.1.

Obr. 2.1: Na obrázku sa nachádza dokument, ktorý prešiel binarizáciou v OCR softvéri. Vľavo je dokument pred, vpravo po binarizácii. Obrázok bol prevzatý z literatúry [22].

Segmentácia (Segmentation)

Segmentácia je proces rozdelenia textu v obrázkoch na menšie časti. Segmentácia v obrázkoch sa vykonáva po predbežnom spracovaní. Existujú rôzne úrovne segmentácie, podľa toho, na akej úrovni je potrebné vykonať rozpoznanie:

- odseky
- riadky
- slová
- znaky.[18]

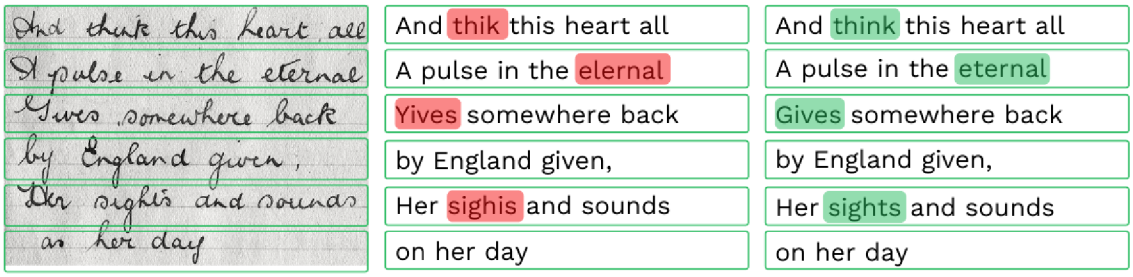
Extrakcia vlastností (Feature Extraction)

V tomto kroku prebieha identifikácia vlastností jednotlivých znakov, a následne rozpoznanie týchto znakov. OCR softvér pracuje naraz s jednou časťou textu, napríklad s jedným znakom alebo riadkom, podľa toho, ako bol text v minulom kroku segmentovaný. Extrakcia vlastností je vykonávaná pomocou jednej z dvoch metód:

- **Rozpoznávanie vzorov** (Pattern recognition) — Algoritmus rozpoznávania vzorov zahŕňa porovnanie každého znaku s knižnicou matíc všetkých známych znakov. OCR porovnáva jeden pixel po druhom, kým nenájde znak, ktorý najviac odpovedá rozpoznávanému znaku, a ten priradí ako výsledný.[9] OCR algoritmy sú trénované na širokej škále štýlov písma, textových formátov a štýlov rukopisu, aby porovnali znaky zo vstupného súboru s tým, čo sa naučili.[1]
- **Rozpoznávanie vlastností** (Feature recognition) — Prostredníctvom algoritmu rozpoznávania vlastností aplikuje OCR softvér pravidlá zohľadňujúce vlastnosti určitého znaku na jeho identifikáciu. Príklady vlastností zahŕňajú počet šikmých čiar, prekrížených čiar alebo kriviek používaných na porovnávanie a identifikáciu znakov.[5]

Konečné spracovanie (Post-processing)

Po úspešnom rozpoznaní znakov sú výsledky porovnané so slovnou zásobou alebo knižnicou znakov, aby sa skontrolovala gramatika a opravili sa prípadné chyby. Na obrázku 2.2 je príklad dokumentu, ktorý prešiel rozpoznaním a následne opravou gramatických chýb v post-processingu.



Obr. 2.2: Na obrázku vľavo sa nachádza text, ktorý je určený pre rozpoznanie. Na strednom obrázku je rozpoznaný text pred opravou chýb. Na obrázku vpravo je rozpoznaný text, ktorý prešiel post-processingom a boli v ňom opravené chyby. Obrázok bol prevzatý z literatúry [9].

Meranie presnosti OCR sa vykonáva porovnaním výstupu z vykonanej OCR analýzy s obsahom pôvodnej verzie. Existujú dva typické spôsoby analýzy presnosti OCR softvéru:

- Presnosť na úrovni znakov — koľko percent znakov bolo správne rozpoznaných.
- Presnosť na úrovni slov — koľko percent slov bolo správne rozpoznaných.[5]

Vo väčšine prípadov je presnosť 98-99% prijateľná miera presnosti meraná na úrovni stránky. To znamená, že na stránke s približne 1 000 znakmi by softvér OCR mal presne identifikovať 980 – 990 znakov.[5]

Výstup OCR algoritmu môže byť jednoduchý, ako napríklad reťazec znakov alebo textový súbor. Pokročilejšie OCR nástroje však dokážu vytvoriť súbor PDF, so zachovaním pôvodnej štruktúry stránky, v ktorom je možné text vyhľadávať. Aj keď zatiaľ neexistujú žiadne nástroje, ktoré by zaručili 100% presnosť na rôznych vstupných súboroch, niektoré OCR algoritmy môžu dosiahnuť presnosť až 99,8% na známych textoch s dobre rozpoznateľným štýlom písma. Rozpoznávanie ručne písaných znakov, alebo slabý proces trénovania algoritmu môžu výrazne zhoršiť výsledky a chybovosť rozpoznávania. Preto je potrebné, aby užívatelia neustále monitorovali, kontrolovali a opravovali výstup OCR algoritmov, najmä keď do procesu spracovania vstupujú dokumenty s novým štýlom písma.[1]

2.2 Využitie optického rozpoznávania znakov v praxi

OCR má v dnešnom svete, kedy sa všetko digitalizuje, mnoho praktických využití. Nižšie sú predstavené niektoré významné oblasti, kde sa OCR využíva, a taktiež konkrétne prípady použitia.

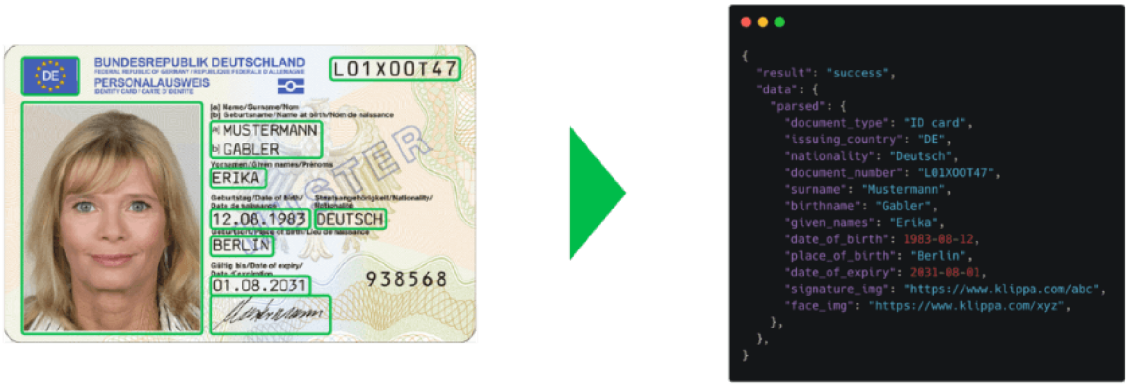
Uchovávanie dokumentov

Starožitné knihy, historické dokumenty, osobné záznamy a mnohé ďalšie dôležité informácie existujú vo forme záznamov napísaných atramentom na papieri. Technológia OCR nám dáva príležitosť digitalizovať takýto obsah, takže aj v prípade zničenia alebo znehodnotenia dokumentov bude obsah stále dostupný.[19] Taktiež to umožňuje jednoduchšie vyhľadávanie v dokumentoch.

Bankovníctvo

Bankový sektor je jedným z najväčších spotrebiteľov OCR technológií, ktoré tu pomáhajú zvyšovať bezpečnosť, zlepšujú správu údajov a optimalizujú riziká.[5] Jedným z príkladov využitia OCR v bankovníctve je manipulácia so šekmi a vkladanie papierových šekov pomocou bankových aplikácii. Tieto aplikácie využívajú OCR algoritmy na identifikáciu relevantných polí v šekoch a podľa toho vykonávajú operácie bez toho, aby musel zamestnanec prepisovať všetky tieto údaje manuálne. Okrem toho môžu tieto aplikácie taktiež vykonávať overenie podpisov voči existujúcej databáze a okamžite vykonať transakciu.[1]

V oblasti bankovníctva sa OCR tiež využíva na extrakciu údajov z občianskeho preukazu. Banky musia overovať totožnosť svojich zákazníkov, aby sa uistili, že títo zákazníci sú tí, za ktorých sa napríklad pri zakladaní nového účtu vydávajú. Po extrakcii údajov možno skontrolovať, či daný človek nemá záznam v databáze podvodov, alebo či nie je na čiernej listine, aby sa odhalili pokusy o podvod.[9] Na obrázku 2.3 sa nachádza občiansky preukaz, z ktorého boli použitím OCR algoritmu extrahované potrebné údaje. Extrakcia údajov z dokladov sa však využíva napríklad aj pri cestovných pasoch, a taktiež na veľa miestach mimo bankového sektoru.



Obr. 2.3: Príklad extrakcie údajov z občianskeho preukazu. Vľavo sa nachádza naskenovaný občiansky preukaz, na ktorom sú vyznačené polia, v ktorých sa rozpoznáva text. Na obrázku vpravo sa nachádzajú extrahované údaje. Obrázok bol prevzatý z literatúry [9].

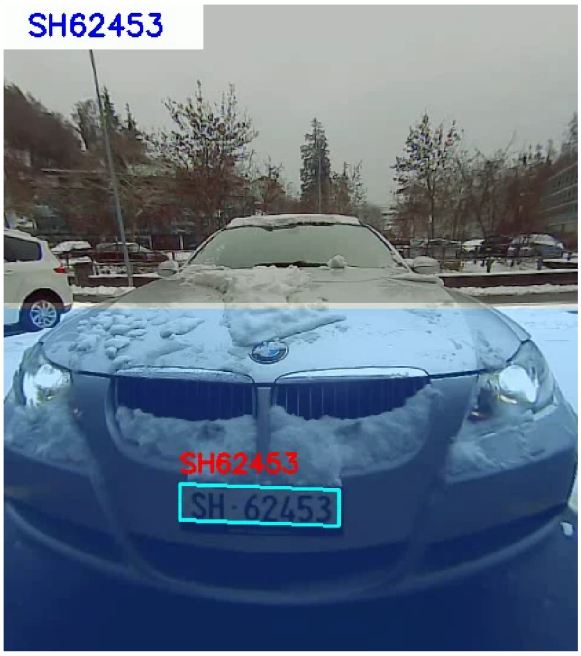
Zdravotníctvo

Podobne ako v bankovom sektore, zdravotnícke organizácie zhromažďujú veľké množstvo dokumentov v listinnej podobe, ako sú napríklad röntgenové snímky, výsledky laboratórnych testov, plány liečby a ďalšie. OCR technológie pomáhajú digitalizovať tieto záznamy, aby sa zabránilo ich strate a znížilo sa úsilie vynaložené na ručnú manipuláciu s dokumentami.[1]

Rozpoznávanie poznávacích značiek

Automatické rozpoznávanie poznávacích značiek (Automatic number-plate recognition — ANTR) je jednou z aplikácií OCR, ktorá je používaná už mnoho rokov. Technológia ANTR sa používa v nespočetnom množstve krajín na zaistenie národnej bezpečnosti od cestných priestupkov až po sledovanie trestnej činnosti. Poznávacie značky sú elektronicky pripojené k údajom vlastníka vozidla, čo zjednodušuje proces jeho identifikácie. Keďže pozná-

vacie značky sú tvorené kombináciou niekoľkých čísel a písmen s ľahko čitateľným štýlom písma, rozpoznávanie poznávacích značiek môže byť vykonávané s vysokou konzistenciou a presnosťou.[19]



Obr. 2.4: Kamera na parkovisku sníma oblasť parkovacieho miesta v reálnom čase. Po zaparkovaní vozidla je rozpoznaná jeho poznávacia značka. Obrázok bol prevzatý z literatúry [5].

Prevod textu na reč

Technológia prevodu textu na reč (Text-to-speech) umožňuje skenovanie textu a následné čítanie nahlas zariadením. Táto technológia veľmi pomáha ľudom so zrakovým postihnutím a ide vôbec o jednu z prvých aplikácií OCR. V dnešnej dobe táto technológia podporuje nespočetne veľa jazykov a dialektov, takže je prístupná pre ešte viacej ľudí.[19]

Kapitola 3

Existujúce rozhrania pre kontrolu a opravu skenovaného textu

V súčasnosti existuje mnoho užívateľských rozhraní, ktoré umožňujú kontrolovať a opravovať chyby v skenovanom texte. Veľa z nich je súčasťou mobilných aplikácií, ktoré slúžia na skenovanie dokumentov, ich ukladanie, úpravu a prácu s nimi. V tejto kapitole sú popísané niektoré vybrané užívateľské rozhrania pre kontrolu a opravu skenovaného textu a ich funkcie, ktoré boli otestované. Pre testovanie boli vybrané také aplikácie, ktoré umožňujú upravovať text a zároveň mať zobrazený skenovaný dokument, aby sa dal text kontrolovať podľa neho.

3.1 Pen to Print

Pen to Print je mobilná aplikácia dostupná pre Android a iOS, určená primárne pre OCR skenovanie rukou písaného textu. Fotografia pre skenovanie môže byť vytvorená priamo v aplikácií, alebo dodaná zo zariadenia. Text je po skenovaní možné skontrolovať po jednotlivých riadkoch a opraviť prípadné chyby. Opravy sa vykonávajú pomocou klávesnice a prepisujú sa celé riadky, nie napríklad iba oddelené slová. Aplikácia v stave opravy textu je zobrazená na obrázku 3.1. Po kontrole textu a oprave chýb sú riadky textu spojené a celý text sa dá uložiť alebo kopírovať a ďalej s ním pracovať.

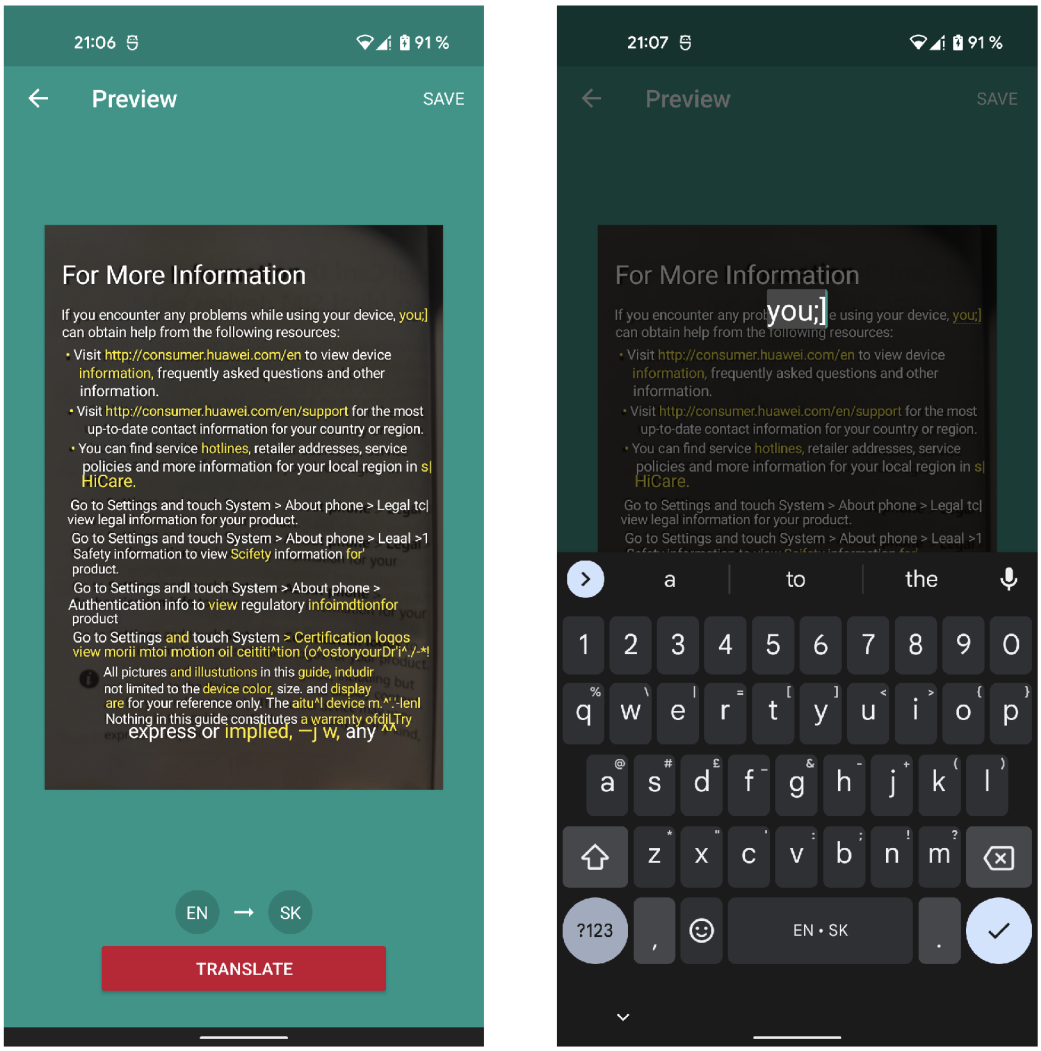
Obr. 3.1: Časť užívateľského rozhrania aplikácie Pen to Print, kde sa upravuje skenovaný text. Skenovaný text je rozdelený na riadky, ktoré je možné samostatne upraviť kliknutím na tlačidlo Edit.

3.2 ABBYY

Mobilná aplikácia FineReader od ABBYY slúži na skenovanie dokumentov a ich konverziu na upraviteľný text. Rozpoznaný text sa dá však otvoriť iba v inej aplikácií pre úpravu textu, takže oprava chýb priamo vo FineReaderi nie je možná.

ABBYY má však aj aplikáciu TextGrabber Scan OCR Translate, ktorá je určená na preklad textu z fotografie, ale taktiež umožňuje tento text pred prekladom upraviť. Úprava prebieha prepisovaním textu pomocou klávesnice. Je zobrazená skenovaná fotografia,

ktorá je však prekrytá rozpoznaným textom tak, že je ťažké na ňu vidieť, a teda úprava je viac-menej robená len opravou chýb v texte podľa vlastného uváženia bez sledovania skenovanej fotografie. Je to viditeľné na obrázku 3.2. Opravovanie chýb prebieha po jednotlivých slovách, nie po celých riadkoch. Keď sú chyby opravené, užívateľ zvolí preklad textu a pôvodný text bude nahradený preloženým textom. V aplikácií sú k dispozícii tri skúšobné skeny, potom je potrebné zaplatiť si za Premium.

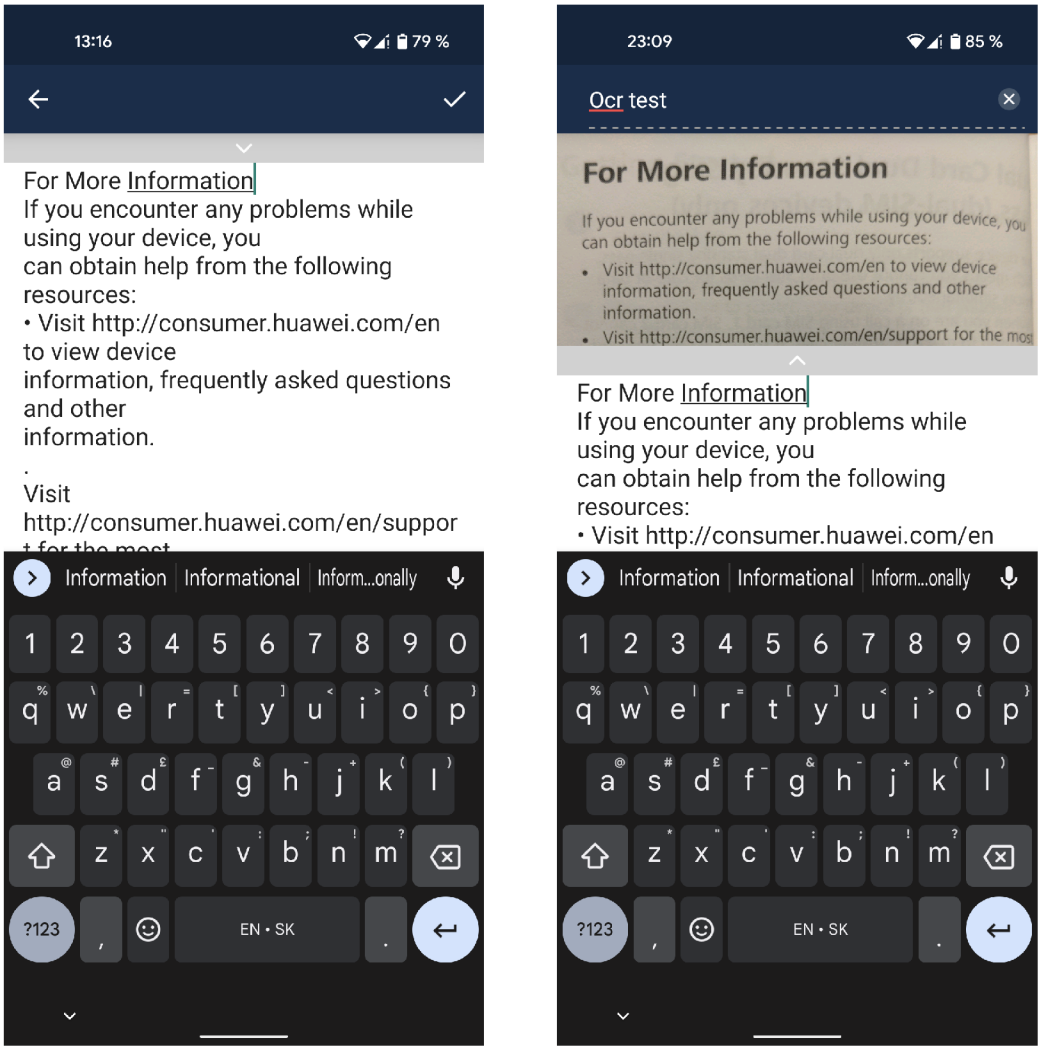


Obr. 3.2: V tejto časti aplikácie ABBYY TextGrabber Scan OCR Translate je možné opraviť prípadné chyby skenovania. Na snímke obrazovky vľavo sa nachádza skenovaný dokument prekrytý rozpoznaným textom. Snímka obrazovky vpravo ukazuje úpravu nesprávne rozpoznaného slova.

3.3 Tiny Scanner

Tiny Scanner je aplikácia pre iOS a Android umožňujúca skenovať dokumenty, fotografie, účtenky, knihy alebo čokoľvek iné do PDF.[2] Snímku s textom na rozpoznanie je možné dodať buď vytvorením fotografie priamo z aplikácie, alebo pridaním existujúcej fotografie.

Po skenovaní textu na fotografii je celý rozpoznaný text v textovom poli, ktoré je možné upravovať klávesnicou. Popritom si užívateľ môže zvoliť, či chce pri úprave vidieť originálnu snímku, ktorú je možné priblížiť a posúvať sa v nej. Aplikácia v stave úpravy rozpoznaného textu je zobrazená na obrázku 3.3. OCR funkcie aplikácie sú po využití dvoch bezplatných rozpoznaní textu platené.

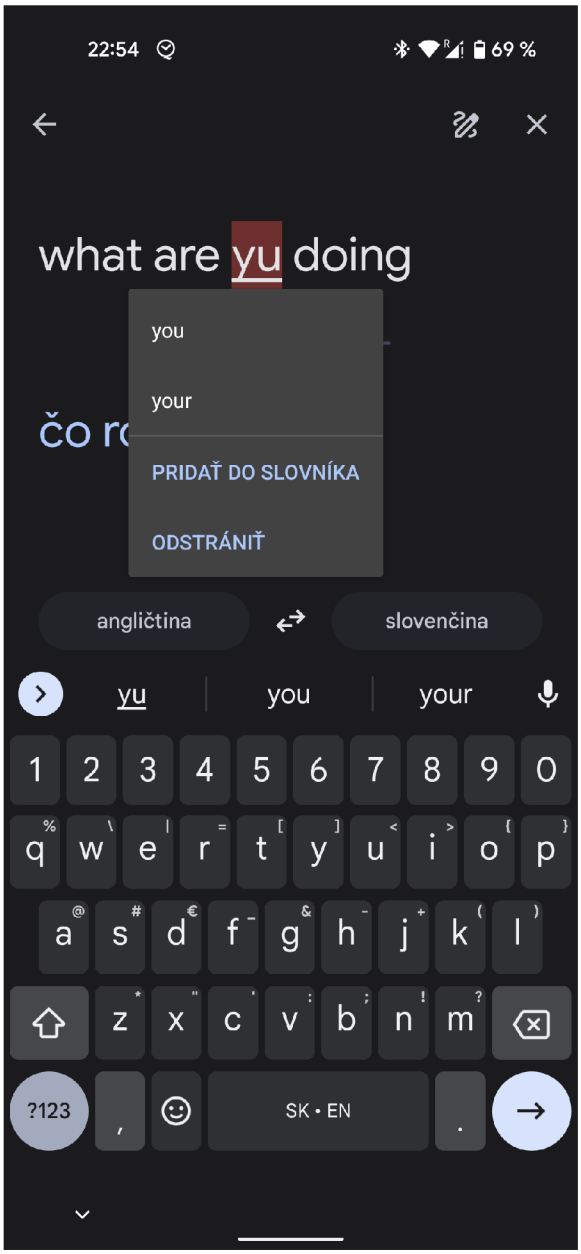


Obr. 3.3: Časť užívateľského rozhrania aplikácie Tiny Scanner určená na úpravu rozpoznaného textu z fotografie. Na obrázku vľavo sa nachádza náhľad úpravy textu bez zobrazenia originálnej fotografie. Kliknutím na šípku v sivom poli nad textom je možné rozbaliť alebo zbaliť náhľad fotografie. Na obrázku vpravo je zobrazená úprava textu naraz súčasne s náhľadom skenovanej fotografie.

3.4 Alternatívne možnosti úpravy textu

Žiadna z testovaných aplikácií neponúka alternatívu úpravy textu klávesnicou, ako napríklad výber z možných variantov slov, čo je bežné napríklad pri nástrojoch pre kontrolu pravopisu. Príklad takejto úpravy je možné vidieť na obrázku 3.4, kde je zobrazená aplikácia

cia Google Translate a je tu použitá aj kontrola pravopisu. Užívateľ si môže pri nesprávne napísanom slove vybrať z ponúkaných možností náhrady slova. Takáto možnosť v OCR aplikáciách by mohla užívateľovi urýchliť a zjednodušiť opravu chýb.



Obr. 3.4: Na obrázku sa nachádza mobilná aplikácia Google Translate s kontrolou pravopisu a možnosťou náhrady nesprávne napísaného slova jedným z ponúkaných variantov.

Kapitola 4

Použité technológie pre tvorbu webovej aplikácie

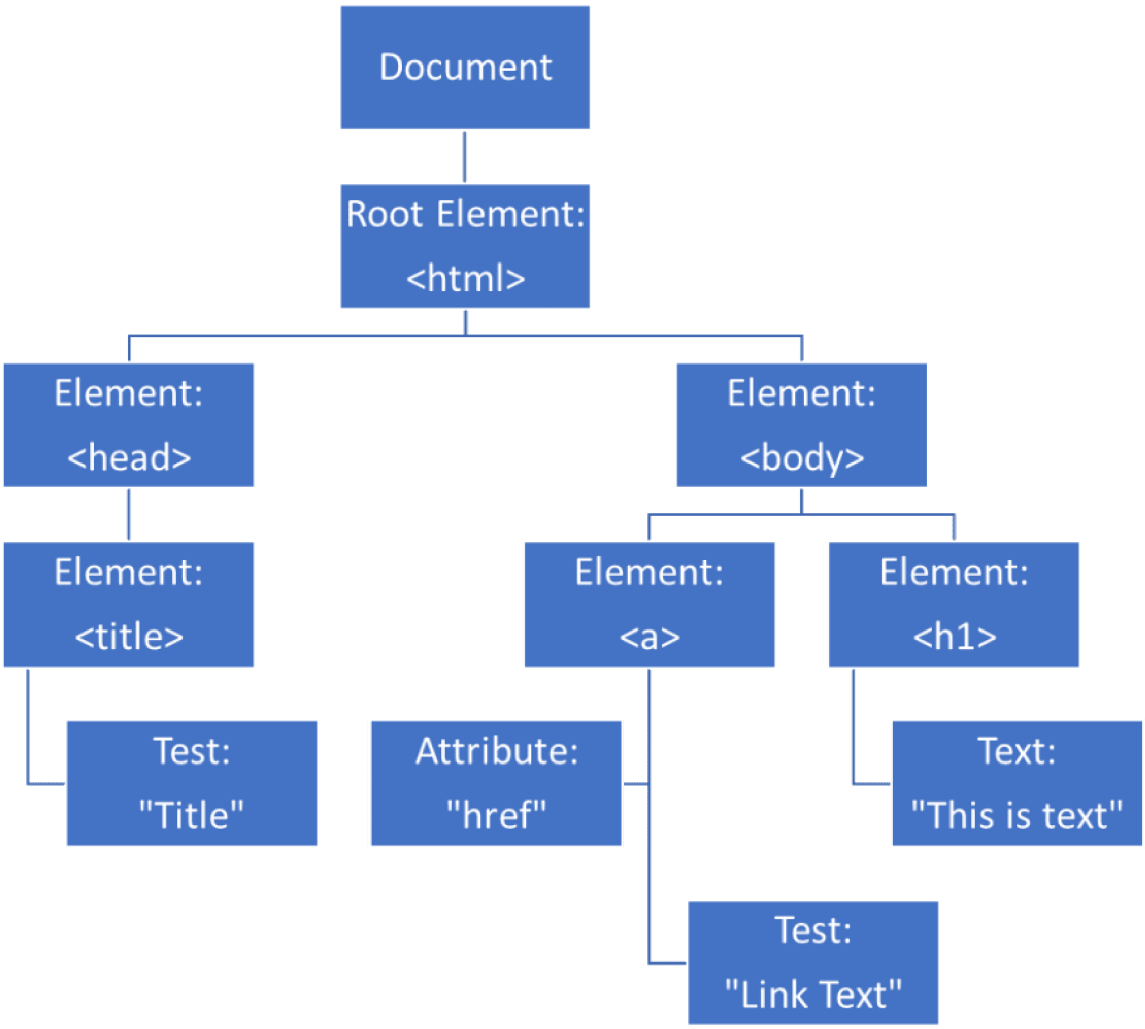
V tejto kapitole sú popísané technológie, ktoré boli použité pri implementácií webovej aplikácie, databázy, a spustenia webového a databázového servera. Pre pochopenie fungovania jednotlivých technológií je potrebné vysvetliť význam niekoľkých použitých pojmov.

- **Framework** je nástroj, ktorý poskytuje hotové komponenty alebo riešenia, ktoré sú prispôsobené s cieľom urýchliť vývoj. Účelom frameworku je pomáhať pri vývoji a poskytovať nízko-úrovňovú funkcionalitu, aby sa vývojár mohol zamerať na prvky, ktoré robia projekt jedinečným.[15]

- **SPA**[1] je implementácia webovej aplikácie, ktorá načíta iba jeden webový dokument a potom aktualizuje obsah tohto dokumentu prostredníctvom rozhraní JavaScript API, keď sa má zobraziť iný obsah. To umožňuje užívateľom používať webové stránky bez načítania celých nových stránok zo servera, čo môže viesť k zvýšeniu výkonu a väčšej dynamickosti.[13]

- **DOM**[2] je dátová reprezentácia objektov, ktoré tvoria štruktúru a obsah dokumentu na webe. Reprezentuje stránku tak, aby bolo možné meniť štruktúru, štýl a obsah dokumentu. DOM predstavuje dokument ako uzly a objekty prepojené do stromovej štruktúry. Týmto spôsobom môžu programovacie jazyky komunikovať so stránkou.[13] Príklad jednoduchého DOM stromu je zobrazený na obrázku 4.1.

[1] SPA - Single-page application (jednostránková aplikácia)
[2] DOM - Document Object Model (objektový model dokumentu)



Obr. 4.1: Na obrázku sa nachádza príklad jednoduchého DOM stromu webovej stránky. Obrázok bol prevzatý z [10].

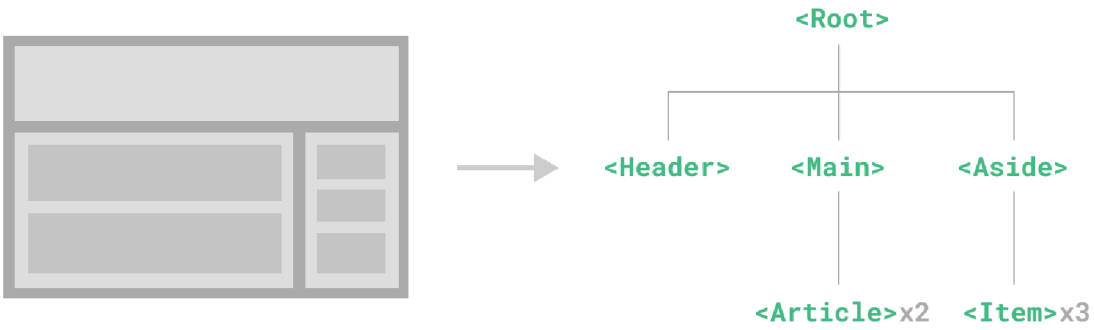
4.1 Vue JS

Vue JS je open-source framework jazyka JavaScript určený pre tvorbu interaktívnych užívateľských rozhraní a SPA. Vytvoril ho Evan You, prvá verzia vyšla v roku 2014.[20] Vue, React a Angular sú tri najpoužívanejšie JavaScript frameworky. Veľká výhoda Vue spočíva v tom, že je minimalistický a rýchly.[16]

Komponenty

Komponenty sú základné stavebné jednotky vo Vue aplikácii. Umožňujú rozdeliť užívateľské rozhranie do nezávislých a znovu-použiteľných častí. Pre Vue aplikácie je bežné, že sú organizované do stromu previazaných komponentov.[21] Jednoduchá schéma komponentov

je zobrazená na obrázku 4.2. Je to veľmi podobné previazaniu natívnych HTML[3] prvkov V DOM-e, avšak Vue implementuje svoj vlastný komponentový model, ktorý umožňuje zapúzdriť obsah a logiku (premenné, metódy, atď.) do každého komponentu.[21]



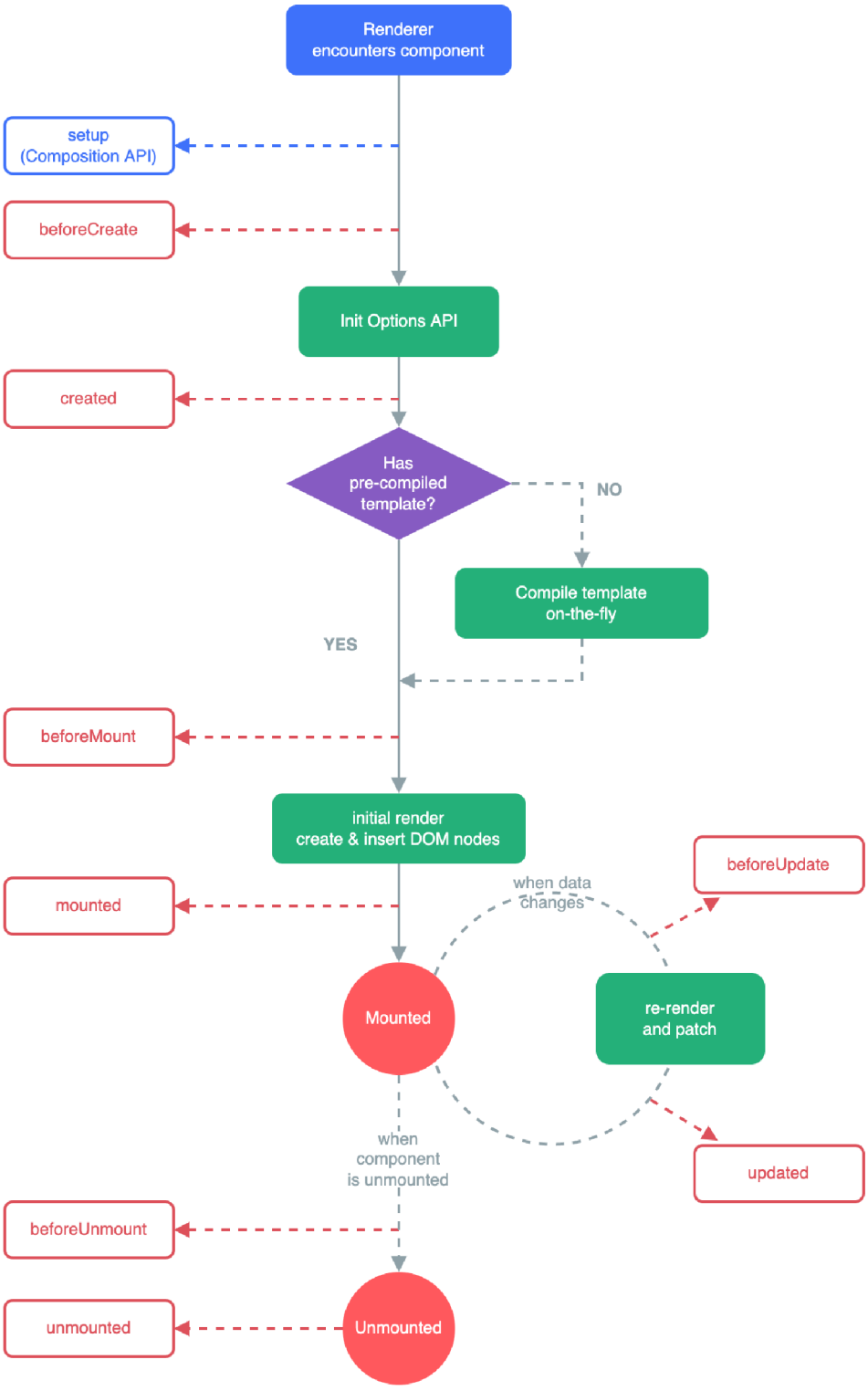
Obr. 4.2: Príklad schémy stromu komponentov vo Vue JS. Vľavo sa nachádza náčrt webovej aplikácie s vyznačením komponentov. Vpravo sa nachádza schéma stromu komponentov a ich previazanie. Obrázok bol prevzatý z literatúry [21].

Komponent má svoj vlastný stav a vlastnosti. Stav komponentu je vo Vue je označovaný ako data. Stav si ovláda a mení samotný komponent. Zmeny hodnôt premenných v stave je možné sledovať cez watchery. Watcher sa nastaví na konkrétnu premennú v data a následne sa táto funkcia zavolá po každej zmene hodnoty danej premennej. To môže byť užitočné napríklad vtedy, keď sa majú v prípade zmeny hodnôt vykonať nejaké animácie.[16]

Každá inštancia Vue komponentu pri svojom vytvorení prechádza sériou inicializačných krokov — napríklad potrebuje nastaviť pozorovanie dát, zostaviť šablónu, pripojiť inštanciu komponentu k DOM-u a aktualizovať DOM pri zmene údajov. Zároveň počas tohto procesu spúšťa funkcie nazývané lifecycle hooks, ktoré užívateľom poskytujú možnosť pridať svoj vlastný kód v konkrétnych fázach.[21] Existuje osem fáz životného cyklu komponentu, ktoré sú zobrazené v diagrame na obrázku 4.3:

1. beforeCreate
2. created
3. beforeMount
4. mounted
5. beforeUpdate
6. updated
7. beforeUnmount
8. unmounted

[3]HTML - Hypertext markup language (hypertextový značkovací jazyk)



Obr. 4.3: Diagram životného cyklu komponentu vo Vue JS. Na vrchu je stav, v ktorom je komponent rozpoznaný, a začína sa jeho vytváranie. Komponent prechádza sériou inicializačných krokov až po stav Mounted. V tomto stave sa komponent aktualizuje. Po odpojení z DOM-u sa dostane do stavu Unmounted. Obrázok bol prevzatý z literatúry [21].

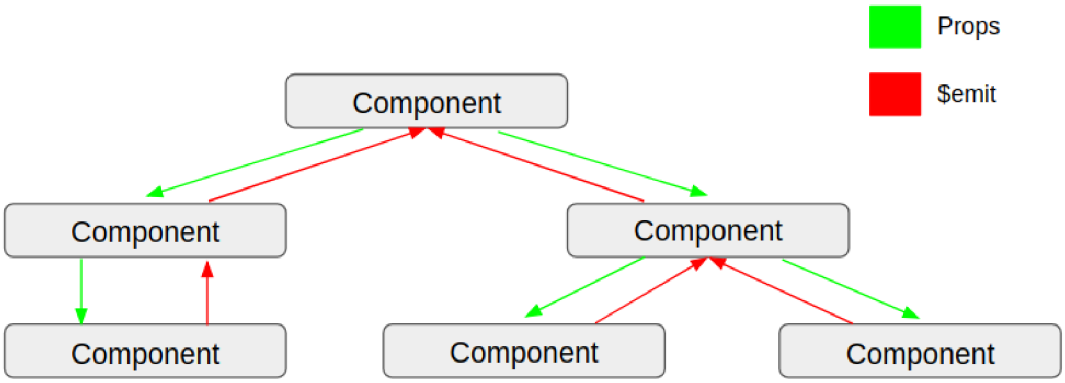
Jednou z výhod komponentov je ich schopnosť zapúzdriť dáta, metódy a HTML prvky, použiť ich na viacerých miestach aplikácie a tým pádom sa vyhnúť opakovanému písaniu rovnakého kódu. Avšak často sa stáva, že komponenty potrebujú medzi sebou komunikovať a posielať si dáta. Toto je zabezpečené pomocou tzv. props a emits.

Props a Emits

Props je špeciálne kľúčové slovo (keyword), odvodené zo slova properties (vlastnosti). Props môžu byť registrované v komponente a slúžia na predávanie dát z rodičovského komponentu do jedného z jeho synovských komponentov. Dáta cez props môžu prúdiť iba týmto jedným smerom, to znamená, že pomocou props nie je možné posielať dáta zo synovského do rodičovského komponentu.[14] Každá property môže mať definovaný typ (boolean, string, array, function a pod.).[16]

Emit je funkcia ktorá dovoľuje posielať signály zo synovského do rodičovského komponentu.[11] Tieto signály slúžia na oznámenie rodičovskému komponentu, že nastala nejaká udalosť (event), napríklad klik (click event). Rodičovský komponent potom typicky vykoná nejakú akciu, ako napríklad zavolanie nejakej funkcie, alebo aktualizácia dát komponentu.[17]

Dáta môžu byť posielané aj v rámci viacerých úrovní komponentov. Na obrázku 4.4 je zobrazená jednoduchá schéma použitia props a emits medzi komponentami.



Obr. 4.4: Diagram posielania dát medzi komponentami vo Vue JS. Obrázok bol prevzatý z literatúry [7]

Refs

Refs (odvodené od references) sú vlastnosti inštancie, ktoré sa používajú na registráciu alebo označenie odkazu na HTML prvky alebo podradené prvky v šablóne aplikácie.[12] Umožňujú získať priamy odkaz na konkrétny DOM prvok alebo inštanciu synovského komponentu po jeho pripojení.[21]

Ak sa do HTML prvku vo Vue šablóne pridá atribút ref, bude možné sa odkazovať na tento prvok alebo dokonca na podradený prvok.[12] Toto je užitočné napríklad v prípade, že potrebujeme zavolať nejakú funkciu v synovskom komponente.

Podmienené vykresľovanie

Na vykreslenie nejakého HTML prvku alebo Vue komponentu a jeho pridanie do DOM stromu iba za určitej podmienky je možné použiť Vue príkaz v-if. Blok, kde je v-if aplikovaný sa vykreslí iba vtedy, ak má výraz za v-if pravdivú hodnotu.

V-if môže byť doplnený príkazmi v-else-if a v-else pre doplnenie podmienok a na vykreslenie iného bloku v prípade nepravdivých hodnôt v predchádzajúcich blokoch. Jednoduchý príklad použitia podmieneného vykresľovania, ktorý bol prevzatý z literatúry [21]:

```
<div v-if="type === 'A'">
  A~</div>
<div v-else-if="type === 'B'">
  B
</div>
<div v-else-if="type === 'C'">
  C
</div>
<div v-else>
  Not A/B/C
</div>
```

Vykresľovanie zoznamu

Príkaz v-for môže byť použitý na vykreslenie zoznamu položiek, ktoré sa nachádzajú v poli. v-for vyžaduje špeciálnu syntax vo forme ïtem in items", kde items je pole zdrojových údajov a item je alias pre prvok poľa, cez ktoré sa iteruje.[21] V-for môže byť použitý aj zreťazene, ako je to zobrazené v nasledujúcom príklade, ktorý bol prevzatý z literatúry [21]:

```
<li v-for="item in items">
  <span v-for="childItem in item.children">
    {{ item.message }} {{ childItem }}
  </span>
</li>
```

Naviazanie atribútov k prvkom

Na naviazanie jedného alebo viacerých atribútov alebo podporných komponentov k HTML prvku alebo Vue komponentu sa používa príkaz v-bind. Ak je tento atribút naviazaný na údaje definované vo Vue inštancii, potom je možné pozorovať dynamické zmeny prvku pri zmenách údajov. Používa sa napríklad pri posúvaní tried, pre priradenie props k synovským komponentom, alebo pre priradenie CSS[4] štýlov.[6] Syntax príkazu v-bind je nasledujúca, príklady boli prevzaté z literatúry [6]:

```
<div v-bind:attribute="expression"></div>;
```

Avšak pre jeho časté používanie bola vytvorená skrátená syntax:

```
<div :attribute="expression"></div;
```

[4]CSS - Cascading style sheets (Kaskádové štýly)

4.2 Axios

Axios je HTTP klient pre JavaScript založený na objektoch typu Promise. Má schopnosť vytvárať HTTP žiadosti z prehliadača a spracovávať transformáciu údajov žiadostí a odpovedí.[4]

Objekt Promise predstavuje eventuálne dokončenie (alebo zlyhanie) asynchrónnej operácie a jej výslednú hodnotu, ktorá nie je známa pri začatí operácie. Umožňuje priradiť obslužné akcie k hodnote eventuálneho úspechu alebo zlyhania asynchrónnej akcie. To umožňuje asynchrónnym metódam vracať hodnoty ako synchrónne metódy: namiesto okamžitého vrátenia konečnej hodnoty vráti asynchrónna metóda prísľub (promise) dodať hodnotu v určitom bode v budúcnosti.[13]

4.3 EC2 od Amazon Web Services

Webový a databázový server musia byť niekde spustené. Na to bola využitá služba Amazon Elastic Compute Cloud (EC2) od Amazon Web Services (AWS). Amazon EC2 poskytuje škálovateľnú výpočtovú kapacitu v cloude AWS. Používanie EC2 eliminuje potrebu investovať do hardvéru vopred, takže je ideálny na vývoj a testovanie aplikácií. EC2 poskytuje virtuálne počítačové prostredia, známe ako inštancie (instances).[3] EC2 je spoplatnené podľa toho, koľko výpočtového času je mesačne využitého.

Vytvorená inštancia beží na platforme Windows 10. Po vytvorení jej bola priradená verejná IPv4 adresa 18.156.173.102. Po pripojení na túto adresu cez webový prehliadač je možné aplikáciu vyskúšať.

4.4 XAMPP

Pre účely vytvorenia databázového a webového servera bol použitý softvérový balíček XAMPP, ktorý bol nainštalovaný na EC2 inštancii popísanej v predošlej sekcii. XAMPP obsahuje webový server Apache a databázový server typu MariaDB, ktorý používa jazyk MySQL. Taktiež obsahuje grafické užívateľské rozhranie phpMyAdmin pre SQL[5], pomocou ktorého je možné upravovať vytvorenú databázu a sledovať jej stav.

[5]SQL - Structured query language (Štruktúrovaný jazyk pre dopyty)

Kapitola 5

Vizuálny návrh užívateľského rozhrania

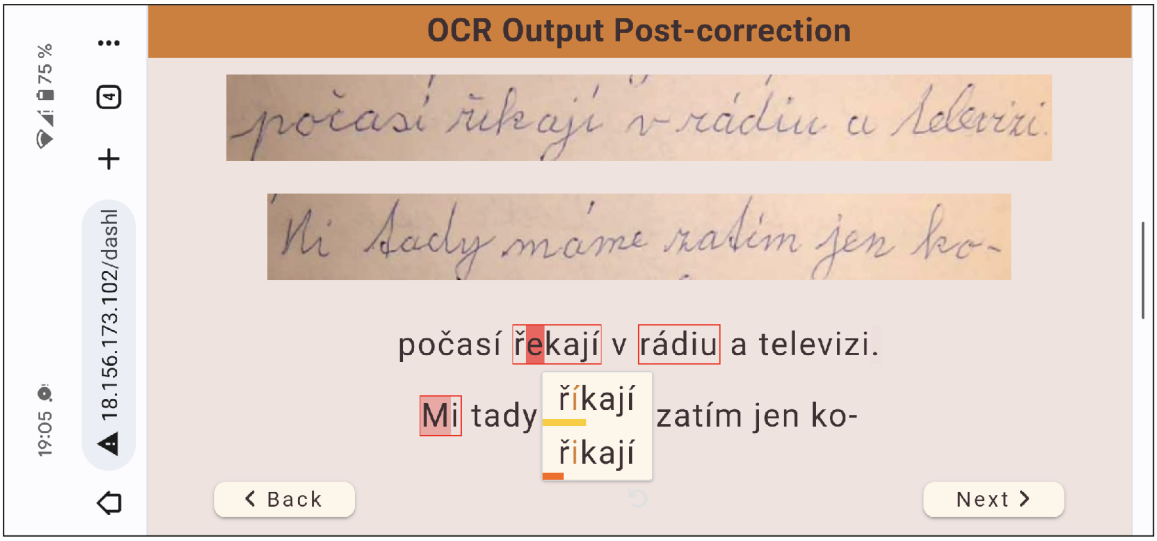
Táto kapitola obsahuje popis návrhu funkcionalít a vzhľadu užívateľského rozhrania aplikácie, a tiež odôvodnenia ich zvolenia.

Pri návrhu užívateľského rozhrania bolo potrebné myslieť na to, že bude určené primárne pre dotykové zariadenia, čo sú väčšinou mobilné telefóny s malým displejom. Z tohto dôvodu bolo pri návrhu potrebné dbať na rozmiestnenie a veľkosť jednotlivých prvkov tak, aby sa všetko potrebné, čo užívateľ potrebuje vidieť, vošlo na malú obrazovku mobilného telefónu.

5.1 Zobrazenie na mobilných zariadeniach

Pre mobilné zariadenia bol zvolený spôsob zobrazenia na šírku, čo je možné vidieť obrázku 5.1. Je to tak z dôvodu rozmerov riadkov, ktoré sú nízke a široké, či už sú to obrázky vystrihnutých riadkov z fotografie dokumentu alebo riadky rozpoznaného textu. Riadky by v prípade zobrazenia na výšku museli byť značne zmenšené, aby sa vošli na obrazovku telefónu, alebo by bez ich zmenšenia užívateľ musel riadky posúvať, čo by mohlo zhoršiť jednoduchosť použitia aplikácia. Zobrazenie na šírku bolo teda zvolené preto, aby sa na obrazovku zmestili celé riadky a aby zároveň boli pre užívateľa čitateľné bez nutnosti ich približovania a posúvania.

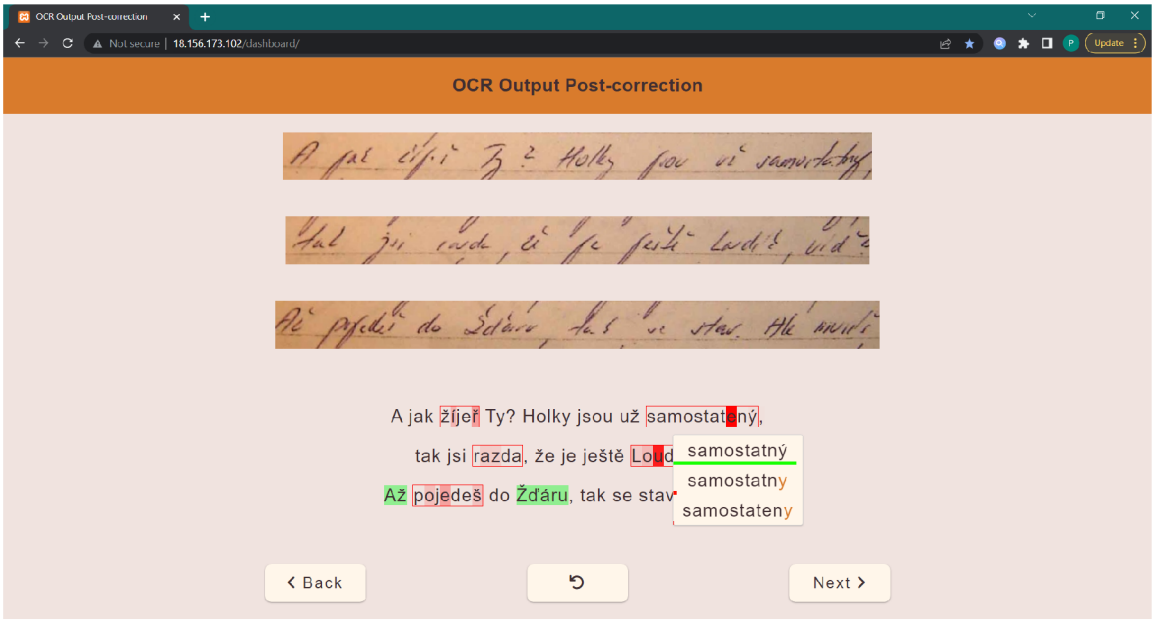
Na mobilných zariadeniach sa zobrazujú naraz dva riadky pre úpravu. Pôvodne bolo rozhranie testované s troma, ale ukázalo sa, že takto sú vystrihnuté riadky z fotografie príliš blízko seba, a na niektorých zariadeniach sa môžu aj prekrývať, kvôli nedostatočnej výške (resp. šírke) zariadenia. Jeden riadok by bol na druhej strane z môjho pohľadu nevhodný preto, že užívateľ by musel príliš často prepínať medzi zobrazenými riadkami. Preto bolo zvolené práve súčasné zobrazenie dvoch riadkov.



Obr. 5.1: Na obrázku sa nachádza užívateľské rozhranie aplikácie zobrazené na mobilnom telefóne. Zobrazenie obsahu na šírku necháva dostatok priestoru pre zobrazenie celých riadkov a taktiež ostatných prvkov rozhrania.

5.2 Úprava rozpoznaného textu a využitie variantov výstupu

Užívateľ pri opravách rozpoznaného textu potrebuje vidieť obrázok originálneho dokumentu, a zároveň aj rozpoznaný text, ktorý ďalej kontroluje a opravuje. Z tohto hľadiska nie je použitie klávesnice na úpravu textu najvhodnejšia voľba, pretože na obrazovke zaberie príliš veľa miesta a nezostal by priestor pre náhľad týchto dvoch nevyhnutných súčastí. Ako spôsob úpravy textu bol teda zvolený výber jednotlivých znakov alebo celých slov z možných variantov OCR výstupu. Toto riešenie necháva dostatok priestoru pre všetko potrebné, čo užívateľ potrebuje na obrazovke vidieť. Riešenie výberu slov z variantov je zobrazené na obrázkoch 5.1 a 5.2.



Obr. 5.2: Na obrázku sa nachádza užívateľské rozhranie aplikácie zobrazené v desktopovom prehliadači. Jedno zo slov má otvorené varianty, z ktorých si užívateľ môže niektorú vybrať, a dané slovo bude nahradené vybraným variantom. Prvé slovo je z troch prítomných najviac pravdepodobné, čomu zodpovedá lišta pod ním, ktorá siaha skoro úplne k pravému okraju ponuky. Druhé slovo je málo pravdepodobné, takže jeho lišta je veľmi krátka. Tretie slovo v ponuke je tak málo pravdepodobné, že lištu pod ním takmer nevidno.

Aby užívateľ nemusel v texte hľadať nesprávne rozpoznané znaky, je potrebné ich označiť. Tiež vďaka tomu užívateľ hneď vie, na ktoré slová a znaky môže kliknúť a zobraziť ich varianty. Znaky, ktoré nie sú jednoznačne správne a majú viacero možných variantov sú v texte vyznačené červeným farebným pozadím. Intenzita priehľadnosti pozadia určuje pravdepodobnosť, s akou je momentálne vybraný znak správny. Čím je červená farba pozadia menej priehľadná, tým je väčšia pravdepodobnosť, že znak je nesprávny. Taktiež slová, ktoré obsahujú minimálne jeden nejednoznačný znak, a teda majú viacero variantov, sú ohraničené rámčekom. Tieto označenia sú viditeľné na obrázku 5.3. Aby užívateľ počas úprav slov a znakov vedel, ktoré už opravil, opravené slová a znaky sa vyznačia zeleným pozadím, čo je zobrazené na obrázku 5.2.

Obr. 5.3: Na obrázku sa nachádza užívateľské rozhranie aplikácie bez otvorených variantov slova alebo znaku. Slová, ktoré je potrebné skontrolovať, sú vyznačené červeným rámčekom. Samostatné znaky sú vyznačené červeným pozadím.

5.3 Označenie pravdepodobnosti návrhov

Pravdepodobnosť správnosti jednotlivých variantov slov a znakov je informácia, ktorá môže byť pre užívateľov užitočná pri rozhodovaní sa nad správnym variantom. Avšak jej označenie číselne by nemuselo byť pre užívateľa veľmi prívetivé. Rozhodol som sa preto pravdepodobnosť návrhov označiť graficky, pretože som sa domnieval, že tento spôsob označenia bude pre užívateľa jednoduchší a rýchlejší na spracovanie.

Po otvorení ponuky návrhov je pod každým návrhom lišta, ktorej dĺžka a farba označujú pravdepodobnosť správnosti daného návrhu. Dĺžkou sa myslí to, po kade lišta siaha on ľavého okraja ponuky. Čím je bližšie k pravému okraju ponuky, tým je vyššia pravdepodobnosť správnosti variantu. Farebná škála lišty siaha od úplne červenej po úplne zelenú. Čím sa farba viac približuje zelenej, tým je pravdepodobnosť správnosti variantu vyššia. Označenia pravdepodobností sú viditeľné na obrázkoch 5.1 a 5.2.

5.4 Tlačidlá

Rozhranie obsahuje tri tlačidlá — tlačidlo na posúvanie vpred (Next), vzad (Back), a tlačidlo na vrátenie vykonaných zmien. Stlačením tlačidiel Next a Back môže užívateľ prepínať medzi zobrazenými riadkami vpred, resp. vzad.

Vrátenie zmien je možné vykonať aj opätovným otvorením návrhov daného slova a zvolením pôvodného návrhu. Tento spôsob je však zdĺhavý, a v niektorých prípadoch aj nezrealizovateľný, napríklad v nasledovnej situácii: V texte sa nachádza znak, ktorý má niekoľko variantov. Jeden z nich je, že na tomto mieste sa nemá nachádzať žiaden znak. Užívateľ si myslí, že tento variant je správny, a teda ho zvolí. Pôvodný znak zmizne, a zvyšok riadku sa posunie o jeden znak doľava. O chvíľu však užívateľ z pohľadu na náhľad

dokumentu zistí, že predsa tam nejaký znak je. Nevie však už otvoriť jeho varianty, pretože tam, kde bol pôvodný znak, nie je momentálne žiaden — užívateľ to pred chvíľou zvolil. V tomto prípade ale môže stlačiť tlačidlo na návrat zmien, vrátiť pôvodný znak a znova si vybrať z jeho dostupných variantov.

Kapitola 6

Implementácia webovej aplikácie a jej súčastí

V tejto kapitole sa nachádza popis niektorých dôležitých prvkov implementácie aplikácie, jej jednotlivých komponentov, databáze a back-endu. Pre implementáciu samotného užívateľského rozhrania bol použitý JavaScript framework Vue JS verzie 3.2.26., ktorý bol načítaný pomocou CDN[1]. Pre komunikáciu rozhrania s back-endom bola použitá JavaScriptová knižnica Axios, ktorá bola tak isto načitaná pomocou CDN. V back-ende bol pre komunikáciu s databázou použitý jazyk PHP. Pre implementáciu webového a databázového servera bol použitý nástroj XAMPP, ktorý bol nainštalovaný na vzdialenej inštancii služby EC2 od Amazon Web Services.

6.1 Vstupné dáta

Použité dáta boli poskytnuté vedúcim práce. Tieto dáta boli získané pomocou nástroja PERO OCR, ktorý vykonal rozpoznanie textu z obrázkov dokumentov. Taktiež boli obrázky dokumentov rozdelené na jednotlivé riadky.

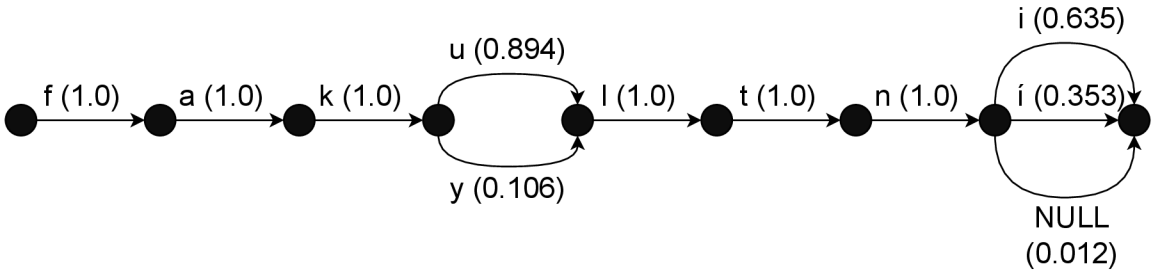
Vstupné dáta, s ktorými aplikácia pracuje sa skladajú z dvoch častí, a to obrazové a textové. Obrazové dáta sú súbory vo formáte JPG obsahujúce vystrihnuté riadky z dokumentov. Tieto obrázky sú potrebné pre zobrazovanie jednotlivých riadkov namiesto celého dokumentu. Textové dáta sú súbory vo formáte Pickle, ktoré obsahujú Confusion network rozpoznaného dokumentu.

Confusion network

Confusion network je jednoduchý lineárny orientovaný acyklický graf, kde každá cesta od počiatočného ku koncovému uzlu prechádza cez všetky ostatné uzly.[23]

V prípade vytvorenej aplikácie sa confusion network skladá zo znakov, kde každému znaku prislúcha pravdepodobnosť jeho správnosti. To znamená, že keď má znak iba jednu možnosť, má pravdepodobnosť 1, ale pri viacerých možnostiach sa už pravdepodobnosť rozdeľuje medzi nich. Niekoľko takýchto objektov so znakmi potom tvorí jeden riadok dokumentu. Príklad jednoduchej confusion network je zobrazený na obrázku 6.1.

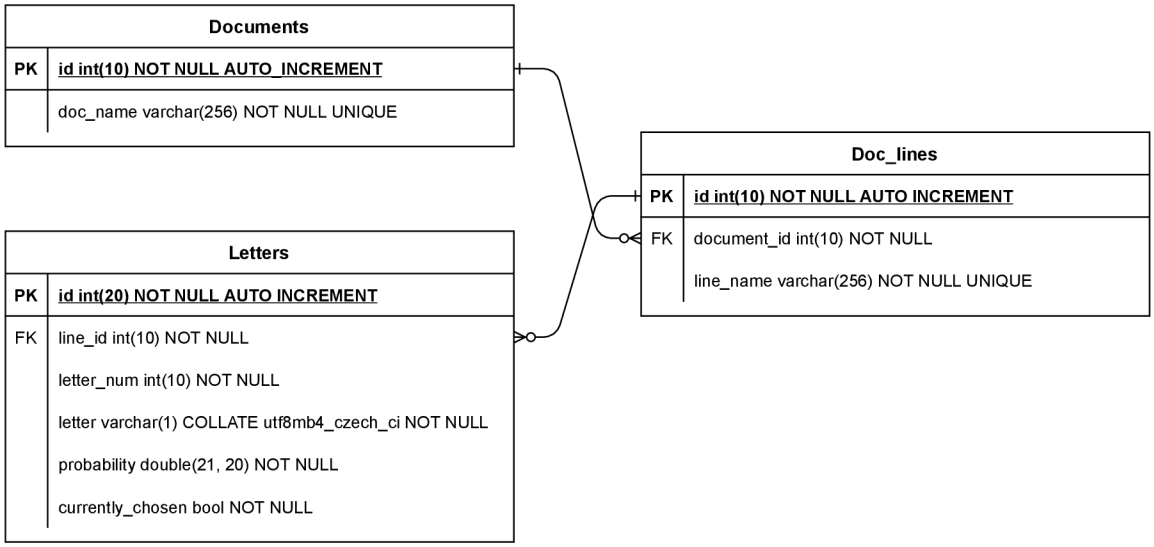
[1] CDN - Content distribution network (sieť distribúcie obsahu)



Obr. 6.1: Na obrázku sa nachádza príklad Confusion Network, ktorá reprezentuje jedno slovo. Za každým písmenom je v zátvorke uvedená jeho pravdepodobnosť správnosti.

6.2 Uloženie dát v databáze

Pre účely uchovania dát o dokumentoch a znakoch v nich bola vytvorená MySQL databáza s troma tabuľkami — `documents`, `doc_lines` a `letters`. Jej relačný diagram je zobrazený na obrázku 6.2.



Obr. 6.2: Relačný diagram databáze vytvorenej databáze, ktorá má tri tabuľky — `documents`, `doc_lines` a `letters`.

Tabuľka `documents` obsahuje iba názvy dokumentov a ich ID. Tabuľka `doc_lines` obsahuje údaje o všetkých riadkoch — ich ID, názvy (názov dokumentu + číselné označenie riadku) a ID dokumentu, ku ktorému riadok patrí. ID dokumentu je v tejto tabuľke cudzím kľúčom (Foreign key), ktorý prislúcha primárnemu kľúču tabuľky `documents`. V tabuľke `letters` sa nachádzajú údaje o všetkých znakoch — ID, číslo znaku v rámci riadku (`letter_num`), konkrétny znak (`letter`), pravdepodobnosť správnosti znaku (`probability`), hodnota typu boolean či znak momentálne je alebo nie je zvolený v upravovanom texte (`currently_chosen`) a nakoniec ID riadku, do ktorého znak patrí (`line_id`). V tomto prípade je cudzím kľúčom ID riadku, ktorý prislúcha primárnemu kľúču tabuľky `doc_lines`.

Tabuľka `documents` bola pri vytváraní manuálne naplnená názvami dokumentov. Tabuľky `doc_lines` a `letters` boli naplnené spustením skriptov insert_lines.py, resp. in-

sert_letters.py. Skripty je potrebné spustiť v tomto poradí. Skript insert_letters.py postupne prechádza súbor, v ktorom sa nachádza confusion network dokumentov, číta z neho dáta o znakoch a následne ich zapíše do databázy. Ak má nejaký znak viacero možností, do databázy sa všetky tieto možnosti zapíšu s rovnakým číslom znaku (`letter_num`), čo bude neskôr potrebné pri skladaní znakov do textu.

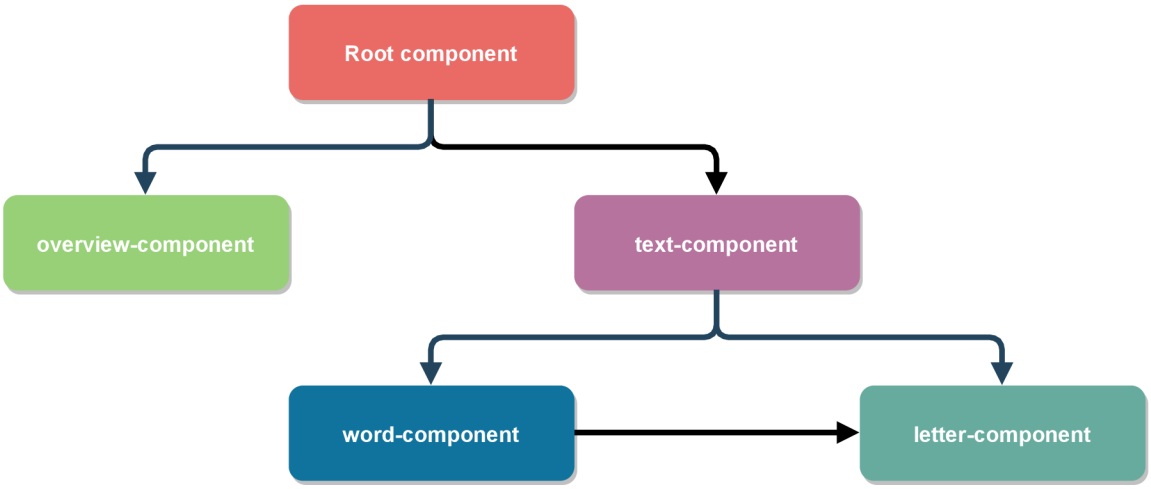
6.3 Rozdelenie aplikácie na komponenty

Implementácia webovej aplikácie začína vytvorením Vue aplikácie `app` v súbore main.js. Aplikácia je v súbore index.html pripojená k hlavnému prvku HTML tela, `<div>` s `id=app`:

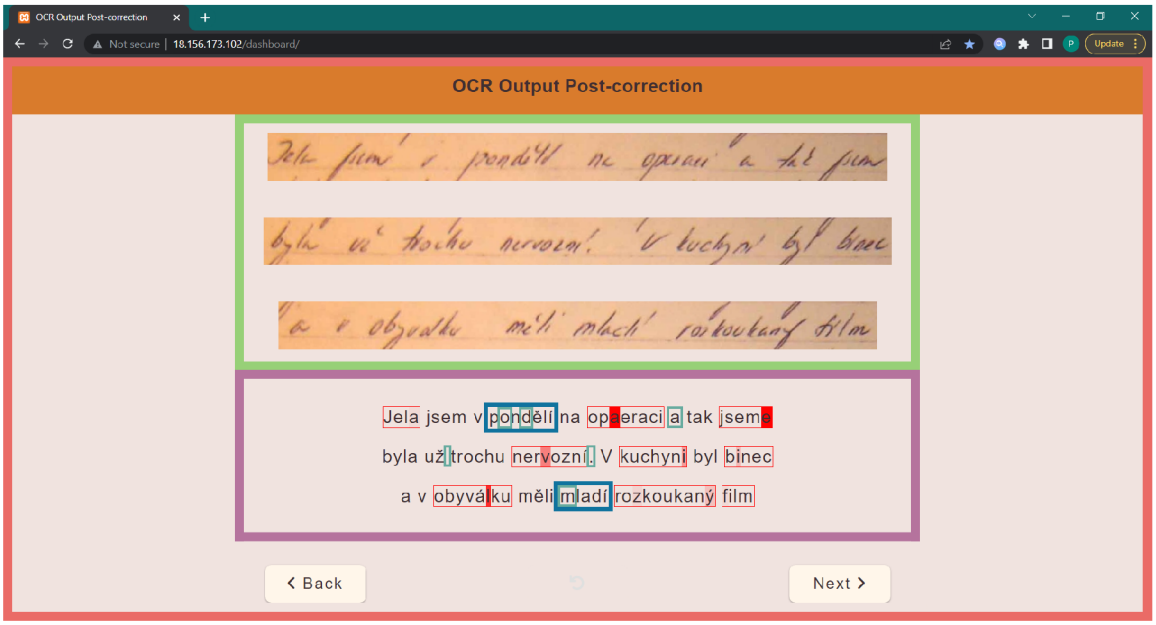
```
const mountedApp = app.mount('#app')
```

V rámci tohto prvku sa nachádza všetok obsah aplikácie. Na vrchu stránky sa nachádza hlavička, ktorá je zobrazená vždy. Pod ňou je zobrazený buď obsah stránky, alebo v prípade načítania nových dát zo serveru je zobrazený tvz. loader-spinner s oznámením o načítaní dokumentu. Aplikácia je rozdelená na niekoľko zreťazených komponentov, ktoré sú zobrazené v strome komponentov na obrázku 6.3. Na obrázku 6.4 sú tieto komponenty na snímke obrazovky aplikácie ohraničené rovnakými farbami ako na predchádzajúcom obrázku. Jednotlivé komponenty sú nasledovné:

- **Základný komponent aplikácie** nachádzajúci sa v súbore main.js. Z tohto komponentu vychádzajú nasledujúce komponenty, ktorých súbory sa nachádzajú v podpriečinku components a sú pomenované podľa názvov jednotlivých komponentov.
- **overview-component** zapúzdruje náhľad jednotlivých riadkov originálneho dokumentu, podľa ktorého užívateľ kontroluje správnosť rozpoznaného textu. V rámci rozhrania je situovaný na vrchu stránky, ihneď pod hlavičkou.
- **text-component** má na starosti všetko to, čo sa týka textu, ktorý sa ďalej upravuje. Je to rodič podkomponentov word a letter.
- **word-component** je komponent, ktorý je použitý vždy v prípade, keď sa v texte nachádza slovo dlhšie ako jeden znak. Každé písmeno v slove je potom samostatný podkomponent letter-component.
- **letter-component** zapúzdruje každý samostatný znak. Ak je znak šúčasťou slova, tak je tento komponent podkomponentom word-componentu. V opačnom prípade je podkomponentom text-componentu.



Obr. 6.3: Schéma komponentov tvoriacich aplikáciu, a ich prepojenie. `letter-component` môže byť dieťaťom `text-componentu` alebo `word-componentu` podľa toho, či sa znak nachádza v slove alebo samostatne na riadku.



Obr. 6.4: Vyznačenie jednotlivých komponentov farebne v užívateľskom rozhraní, podľa farieb, ktoré korešpondujú s obrázkom 6.3.

6.4 Aktualizácia obsahu stránky

Získavanie dát z databázy prebieha v asynchrónnej funkcii `getLinesData()` v hlavnom komponente aplikácie. Počas načítania dát a aktualizácie synovských komponentov sa zobrazí loader-spinner, ktorý dáva užívateľovi vedieť, že musí počkať na aktualizáciu dát. Loader-spinner je aktivovaný na začiatku funkcie nastavením premennej `loaderActive` na true. Následne je použitá HTTP žiadosť GET knižnice Axios:

```
axios.get('get_lines.php', { params: { number_of_lines: numberOfLines,
    line_id: lineID, } });
```

Funkcia spustí skript v súbore get_lines.php a pošle mu parametre upresňujúce koľko riadkov s dátami o nich má byť získaných (`number_of_lines`), a prípadne ID prvého riadku tohto setu riadkov (`line_id`). Pod setom riadkov sa rozumie skupina niekoľkých riadkov s dátami o nich, ktoré sú získané z databáze. ID riadku slúži na to, aby bolo možné listovať medzi setmi riadkov pomocou tlačidiel `Next` a `Back`, a aby boli pri listovaní načítané stále tie isté sety riadkov, ktoré už boli predtým zobrazené. ID prvých riadkov zo setov riadkov sú ukladané v poli v hlavnom komponente. Ak nie je upresnené ID riadku, z databázy budú získané náhodné riadky z náhodne vybraného dokumentu. Tento náhodný výber prebieha taktiež v skripte get_lines.php.

Po získaní dát z databázy funkcia `getLinesData()` ďalej tieto dáta spracuje a aktualizuje nimi premenné komponentu, ktoré sú posielané cez props komponentom `overview-component` a `text-component`. Keď sa tieto komponenty aktualizujú (nastane lifecycle hook `updated`), tak pomocou emitov vyšlú signál rodičovskému komponentu o ich aktualizácií. Emity nastavia hodnoty premenných `overviewUpdated` resp. `textUpdated` na true. Watchery kontrolujú, či už sa zmenili hodnoty oboch premenných, a ak áno, zobrazí sa nový obsah:

```
watch: {
    overviewUpdated(newValue) {
        if (newValue && this.textUpdated) {
            this.loaderActive = false;
        }
    },

    textUpdated(newValue) {
        if (newValue && this.overviewUpdated) {
            this.loaderActive = false;
        }
    }
}
```

Zobrazenie náhľadu nových riadkov

Zobrazenie a aktualizáciu nových náhľadov riadkov zabezpečuje `overview-component`, ktorý dostáva cez prop zoznam názvov riadkov, ktoré majú byť zobrazené. Následne pre každý názov zobrazí prislúchajúci obrázok riadku, pričom súbory obrázkov hľadá v podpriečinku /img/lines.

Spracovanie nových textových dát

Po tom, ako `text-component` dostane nové dáta o riadkoch, musí tieto dáta spracovať, aby s nimi mohol ďalej pracovať. Na to slúži funkcia `parseText()`, ktorá je volaná ako pri vytvorení komponentu, tak aj pred aktualizáciou jeho DOM stromu. Táto funkcia vytvorí pole riadkov `parsedText`, kde každý prvok je pole objektov znakov a slov (objekt slov tiež obsahuje pole objektov znakov). Objekty znakov obsahujú mimo iného informácie o variantoch znaku s ich pravdepodobnosťami, ďalej to, ktorý variant je práve zvolený, a taktiež čísla riadku, slova, prípadne pozície v slove, kde sa daný znak nachádza. Od teraz sa v tomto komponente pracuje s dátami v tomto novovytvorenom poli. Objekty slov a samostatných znakov obsahujú tiež atribút `suggestionsOpened` typu boolean, ktorý označuje, či má dané slovo alebo znak práve teraz otvorenú ponuku variantov.

`parseText()` taktiež oddeľuje slová od ostatných znakov v texte, a to pomocou kontroly, či sa znak rovná niektorému znaku z oddeľovačov slov. V prípade, že nie, je znak pridaný do slova, inak je pridaný do poľa `parsedText` ako samostatný znak.

Po spracovaní textu je aktualizovaný DOM strom tohto komponentu. Ten pomocou príkazu v-for a v-if postupne vykresľuje DOM, podľa aktuálneho obsahu poľa `parsedText`. V prípade, že je položka v poli objekt slova, pridá do DOM-u komponent slova (`word-component`), v opačnom prípade pridá komponent znaku (`letter-component`). To, ktorý z komponentov má pridať zisťuje podľa existencie premennej `id` v objekte, keďže túto premennú obsahujú iba objekty znakov:

```
<template v-for="(lineData, index) in this.parsedText" :key="index">
    <div class="line">
        <template v-for="(data, index) in lineData" :key="index">
            <word-component
                v-if="typeof data.id === 'undefined'"
                :wordObj="data"
                @open-suggestions="this.handleOpening"
                @select-suggestion="this.handleSelection"
                @save-current-word="this.saveCurrentWord"
                ref="wordComponent">
            </word-component>
            <letter-component
                v-else
                :letterObj="data"
                @open-suggestions="this.handleOpening"
                @select-suggestion="this.handleSelection"
                @save-current-word="this.saveCurrentWord">
            </letter-component>
        </template>
    </div>
</template>
```

6.5 Implementácia práce s ponukou variantov

Otvorenie ponuky variantov

Ponuku variantov je možné otvoriť kliknutím na slovo alebo znak, ktorý obsahuje varianty. Click event je zaznamenávaný v komponente `letter-component` pomocou v-bind `@click` na HTML prvku `<span>` daného znaku. Následne prebehne kontrola, či má znak viac ako jeden variant, a či je znak súčasťou slova. Ak je znak súčasťou slova, jeho varianty sa neotvoria, pretože v tomto prípade sa otvárajú varianty slova v komponente `word-component`.

Po kliknutí sa pošle signál pomocou emitu do rodičovského komponentu. To je buď `word-component` alebo `text-component`. Oba tieto komponenty zachytávajú signál rovnakým spôsobom. V prípade `word-componentu` prebehne kontrola, či má slovo viac ako jeden variant, a ak áno, ponuka sa otvorí. Emit taktiež ako parameter pošle objekt udalosti `event`, ktorý obsahuje informácie o udalosti kliku. Je to potrebné kvôli súradniciam kliku, aby `word-component` vedel, kde sa má zobraziť ponuka variantov.

V `text-componente` sa pri otváraní ponuky variantov ukladajú informácie o tom, ktoré slovo alebo znak otvára svoje varianty. Ukladá sa číslo riadku, číslo slova a prípadne číslo písmena v slove. Tieto dáta budú neskôr potrebné pre zatvorenie variantov daného slova.

Selekcia variantu z ponuky a aktualizácia databázy

Po vybraní variantu z ponuky sa najprv lokálne v komponente aktualizuje súčasne zvolené slovo alebo znak. Je však potrebné tieto údaje aktualizovať aj v databáze. Na to je použitá HTTP žiadosť POST knižnice Axios:

```
axios.post('update_letters.php', newLetters);
```

Táto funkcia spustí skript v súbore update_letters.php. Parameter funkcie je pole s dátami o aktualizácií znakov. O každom aktualizovanom znaku sa posielajú dva údaje — ID znaku a to, či je znak aktuálne zvolený alebo nie (hodnota 1 alebo 0). V databáze je teda aktualizovaná iba táto hodnota v tabuľke znakov.

Zatvorenie ponuky variantov

Ponuka variantov môže byť zatvorená troma spôsobmi:

1. zvolením niektorého z variantov z aktuálne otvorenej ponuky
2. otvorením ponuky variantov iného znaku alebo slova
3. kliknutím na ľubovoľné miesto v aplikácií

Pri prvej možnosti sa všetko deje v samotnom komponente slova alebo znaku, kde sa nastaví, že ponuka má byť zatvorená. Do `text-componentu` sa len emituje signál o tom, že údaje o aktuálne otvorenej ponuke majú byť vynulované.

V prípade druhej možnosti sa v `text-componente` najprv zatvorí aktuálne otvorená ponuka. Deje sa to tak, že v poli `parsedText`, ktoré obsahuje dáta o texte, s ktorým sa aktuálne pracuje, sa nastaví premenná `suggestionsOpened` objektu, ktorý má aktuálne otvorenú ponuku, na false. Tu sú čísla riadku a slova objektu s aktuálne otvorenou ponukou využité na indexovanie v tomto poli. Až potom môže byť otvorená nová ponuka variantov.

Kliknutia na ľubovoľné miesto v aplikácií sú zaznamenávané v hlavnom komponente aplikácie pomocou event listenera, ktorý sleduje click event. Po kliknutí sa zavolá funkcia

`onClickAnywhere()`, ktorá pomocou $refs zavolá funkciu v `text-componente` a pošle jej objekt udalosti:

```
onClickAnywhere(event) {
    this.$refs.textComponent.closeSuggestionsClick(event);
}
```

V `text-componente` musí najprv prebehnúť kontrola, či má byť volaná funkcia zatvorenia ponuky variantov `closeSuggestions()`. Táto kontrola je tu kvôli situáciíz druhého bodu, kedy je funkcia `closeSuggestions()` volaná zároveň signálom z hlavného komponentu a zároveň z `word-componentu` alebo `letter-componentu` (kvôli otváraniu novej ponuky). Keby bola funkcia volaná dvakrát, čerstvo nastavené údaje o práve otvorenej ponuke by boli vynulované. Na vykonanie kontroly bola použitá časová značka event objektu — `timeStamp`:

```
closeSuggestionsClick(event) {
    if (event.timeStamp - this.clickTimeStampWord > 1) {
        this.closeSuggestions();
    }
}
```

Keďže volania z dvoch komponentov prebehnú v rovnaký čas (je to ten istý klik), ich časové značky budú rovnaké, alebo sa budú líšiť maximálne v mikrosekundách. Pri kontrole sa teda porovnajú časové značky z dvoch komponentov. Ak je časová značka hlavného komponentu (`event.timeStamp`) o jednu alebo viac milisekúnd novšia ako tá z podkomponentu (`this.clickTimeStampWord`), je volaná funkcia `closeSuggestions()`. V tomto prípade ide o tretiu možnosť. V opačnom prípade ide o možnosť dva a funkcia `closeSuggestions()` je volaná podkomponentom vo funkcii `handleOpening()`.

Kapitola 7

Testovanie webovej aplikácie na užívateľoch

Testovanie s užívateľmi prebiehalo naživo, pri osobnom stretnutí s nimi. Najprv bola užívateľom predstavená aplikácia, jej jednotlivé prvky a funkcie. Užívatelia mali potom za úlohu v aplikácii opravovať chyby v rozpoznanom texte a následne zodpovedať niekoľko otázok. Počas používania aplikácie som sledoval, ako s ňou užívatelia pracujú, či majú problémy s ovládaním alebo je pre nich ovládanie intuitívne.

Otázky zodpovedané užívateľmi po skončení testovania

Otázky, ktoré užívatelia po skončení testovania zodpovedali sú nasledovné:

- Čo sa vám na aplikácii páči?
- Čo sa vám na aplikácii nepáči?
- Páči sa vám na mobilnom telefóne zobrazenie stránky na šírku, alebo by ste preferovali zobrazenie na výšku?
- Je nejaký prvok, ktorý by vám pri opravách chýb pomohol, a ktorý sa v aplikácii nenachádza? Čo by ste v aplikácii zmenili pre jednoduchšiu opravu?
- Ako hodnotíte prínos náhľadu riadkov skenovaného dokumentu?
- Ako vám pri opravách pomohla vizualizácia pravdepodobnosti jednotlivých návrhov slov a znakov?
- Ako by ste celkovo zhodnotili aplikáciu na stupnici 1-5, pričom 1 je najlepšie a 5 najhoršie?

Predstavenie užívateľov, ktorí aplikáciu testovali

Testovania sa zúčastnili traja užívatelia. Prvý užívateľ (ďalej len U1) má 23 rokov, je študent ekonomickej univerzity a aktuálne má bakalárske vzdelanie. S informačnými technológiami sa stretáva každý deň na úrovni bežného užívateľa. V rámci podobných aplikácií má predchádzajúcu skúsenosť s mobilnou aplikáciou na skenovanie a rozpoznávanie textu, kde sa

text následnej opravoval prepisovaním, avšak na jej názov si nespomenul. U1 testoval aplikáciu na dvoch zariadeniach — MacBook Air 13 a iPhone 7 — kde na oboch bola aplikácia spustená cez prehliadač Safari.

Druhý užívateľ (ďalej len U2) má tiež 23 rokov a má stredoškolské vzdelanie s maturitou. Pracuje v oblasti IT ako junior DevOps inžinier, takže s informačnými technológiami sa stretáva aj na profesionálnej úrovni. S podobnými aplikáciami nemá predchádzajúcu skúsenosť, iba s aplikáciou Grammarly, ktorá pomáha opravovať gramatické chyby v texte. Aplikáciu testoval na zariadení iPhone XR v prehliadači Safari.

Tretia užívateľka (ďalej len U3) má 53 rokov, inžinierske vzdelanie v oblasti strojníctva a pracovné skúsenosti v oblasti ekonomiky a obchodu. S informačnými technológiami sa stretáva každý deň na užívateľskej úrovni. Skúsenosť s podobnými aplikáciami nemá. Aplikáciu testovala na mobilnom zariadení Google Pixel 3 v prehliadači Google Chrome.

Odpovede užívateľov na otázky a vyhodnotenie otázok

Užívateľ U1 sa chvíľu s užívateľským rozhraním zoznamoval, ale po chvíli už mu bolo jasné, ako ho používať. Na rozhraní sa mu páčilo to, ako sú rozmiestnené riadky, že na vrchu obrazovky sú náhľady riadkov dokumentu a pod nimi riadky na úpravu. Prišlo mu to logické a dobre sa mu text podľa toho opravoval. Taktiež sa mu páčilo umiestnenie tlačidiel na spodku obrazovky, sú podľa neho ľahko dostupné. Užívateľovi U2 sa páčilo označenie rámčekom tých slov, ktoré je potrebné skontrolovať. Taktiež sa mu páčilo riešenie vizualizácie pravdepodobnosti návrhov. Podľa užívateľky U3 je rozhranie intuitívne a jednoduché na použitie.

Užívateľovi U1 sa na rozhraní nepáčilo to, že slová a samostatné znaky majú iný spôsob označenia možnej nesprávnosti — slová majú rámčeky a znaky majú červenú farbu pozadia. Rámčeky by odstránil. Slová, ktoré sú určite správne by tiež označil, napríklad zelenou farbou, aby bolo jasné, že tieto slová nemusí užívateľ kontrolovať. Užívateľ U2 by vizuálne upravil náhľady riadkov — pridal by zaoblenie okrajov a tieň. Rozpoznaný text by taktiež ohraničil a pridal tieň. Tiež by uvítal v rozhraní animácie a skrytie UI[1] prehliadača, prípadne režim celej obrazovky. Užívateľke U3 nenapadla žiadna vec, ktorá sa jej na rozhraní nepáči.

Čo sa týka zobrazenia na šírku na mobilnom telefóne, ani jednému užívateľovi to nevadilo a všetci traja to preferovali kvôli lepšiemu rozloženiu a zobrazeniu prvkov rozhrania. Podľa U3 je to vhodnejšie ako zobrazenie na šírku z dôvodu toho, že naraz môže sledovať viac textu bez nutnosti posúvania riadkov. U1 však poznamenal, že niektorým užívateľom by zobrazenie na šírku mohlo vadiť, pretože neradi držia telefón takto otočený.

U1 by ako nový prvok aplikácie pridal legendu, ktorá by vysvetľovala význam grafického označenia pravdepodobnosti návrhov a označenie možnej nesprávnosti slov a znakov v texte. Uvítal by tiež možnosť prepisovania textu klávesnicou. Ako rozšírenie aplikácie do budúcnosti by pridal prepojenie so slovníkom, aby sa dalo zobraziť vysvetlenie slov v texte. Užívateľ U2 navrhol posunutie škály priehľadnosti pozadia znakov, ktoré sú označené ako možné nesprávne. Posunul by ju tak, aby pozadie nemohlo byť len máličko priehľadné, aby bolo pre užívateľov jednoduchšie lokalizovať tieto znaky. U3 by v aplikácii tiež uvítala možnosť prepisovania textu klávesnicou.

Všetci užívatelia zhodnotili náhľad riadkov originálneho dokumentu ako prínosný a text opravovali vo väčšine prípadov podľa neho. V niektorých prípadoch však bol rukopis v dokumente veľmi ťažko čitateľný, vtedy museli užívatelia opravovať text zväčša podľa vlastného

[1] UI - User interface (užívateľské rozhranie)

uváženia. Užívateľke U3 náhľad riadkov pomohol aj v tom, že podľa rukopisu autora textu môže skúsiť identifikovať, čo chcel autor napísať.

Vizualizácia pravdepodobnosti návrhov bola vo všetkých troch prípadoch hodnotená pozitívne. Užívateľ U1 by k nej však privítal vysvetlivky, ako už spomínal v odpovedi na jednu z minulých otázok. Užívateľ U2 ihneď pochopil význam vizualizácie, k čomu možno prispeli aj jeho odborné skúsenosti s IT. Poznamenal však, že pravdepodobnosti návrhov môžu byť niekedy zavádzajúce, ak napríklad návrh č. 1 má oveľa vyššiu pravdepodobnosť správnosti ako návrh č. 2, ale návrh č. 2 je správny. U3 poznamenala, že zobrazenie pravdepodobnosti môže byť tiež nápomocné pre ľudí, ktorí neovládajú gramatiku veľmi dobre.

Celkovo U1 ohodnotil rozhranie na stupnici 1-5 známkou 2,2. U2 ohodnotil rozhranie známkou 2. Rozhranie bolo pre neho intuitívne na ovládanie a pochopil aj farebné označenia slov, písmen a pravdepodobností. U3 hodnotila rozhranie známkou 1, páčila sa jej jednoduchosť použitia a intuitívnosť.

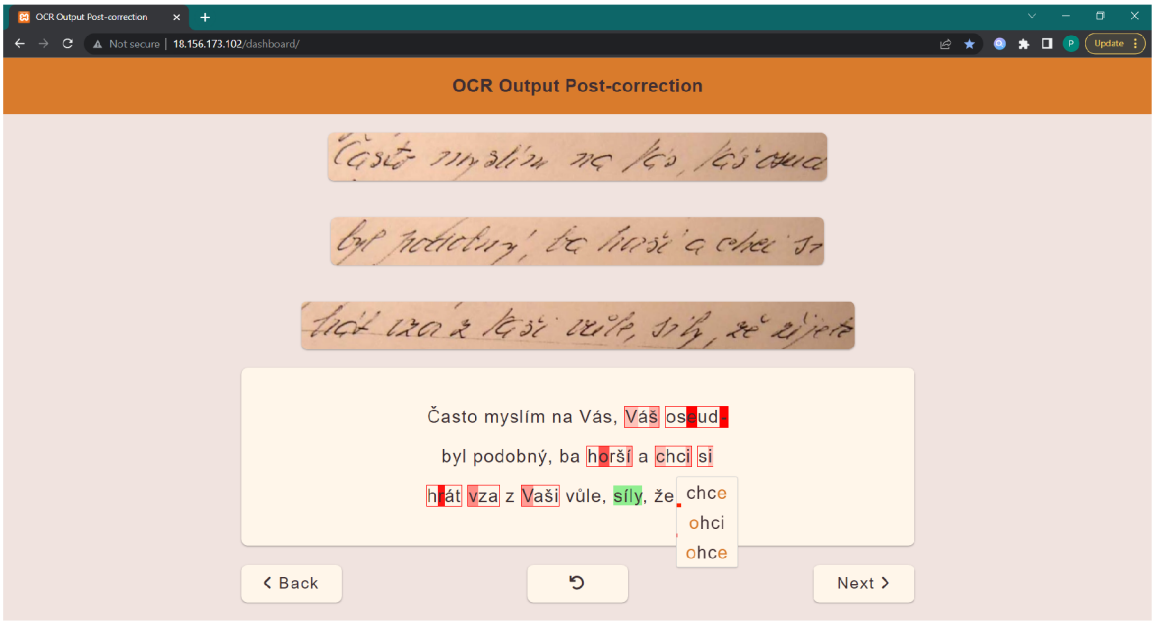
Vyhodnotenie testov a zmeny rozhrania po testovaní

Testy ukázali, že aplikácia a jej užívateľské rozhranie sa užívateľom páči, je intuitívne a jednoduché na použitie. So zámenou slov a písmen pomocou otvárateľnej ponuky variantov nemal žiaden z užívateľov problém. S týmto typom zámeny slov je možné sa stretnúť v mnohých ďalších aplikáciách, takže pre užívateľov bol tento koncept dobre známy. Rozmiestnenie prvkov rozhrania a zobrazenie na šírku sa užívateľom taktiež páčilo. Vizualizácia pravdepodobnosti návrhov všetkým užívateľom pomohla, ale jej vysvetlenie v legende by bolo nápomocné.

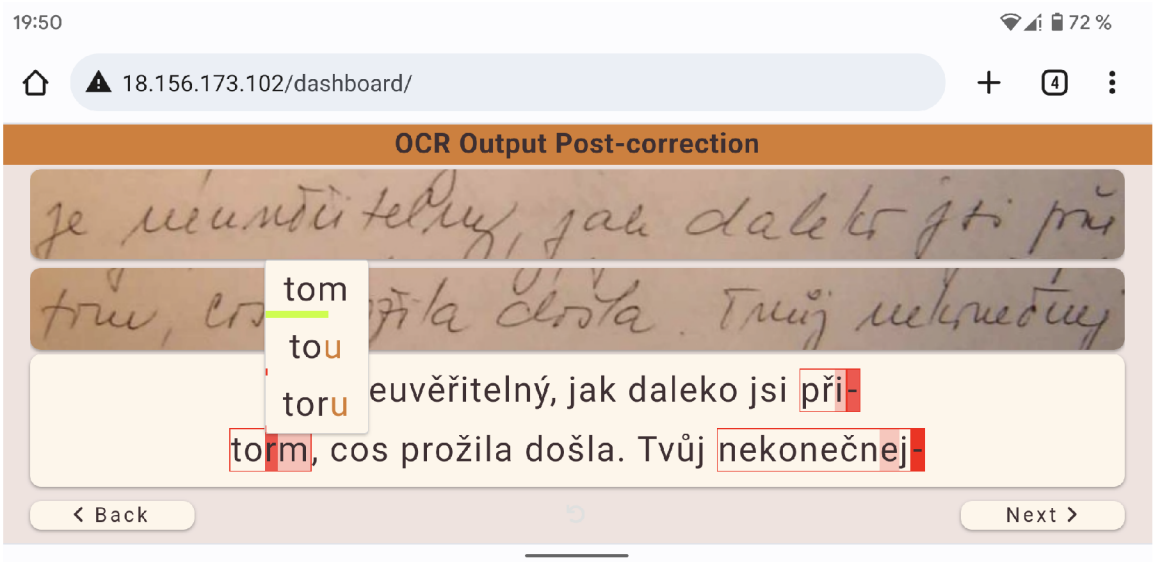
Niektorý užívatelia mali určité výhrady k označeniu slov a znakov, ktoré je potrebné skontrolovať. Preto boli v rozhraní po skončení testov vykonané zmeny v tejto oblasti. Škála priehľadnosti červenej farby pozadia znakov, ktoré sú označené ako možné nesprávne bola posunutá z 0-1 na 0,2-1. Vďaka tomu bude pre budúcich užívateľov jednoduchšie spozorovať v texte znaky, ktorým je potrebné venovať pozornosť. Tiež boli vykonané kozmetické úpravy — zaoblenie hrán obrázkov s náhľadom riadkov, pridanie ohraničenia poľa s rozpoznaným textom a zmena farby pozadia, a taktiež pridanie tieňov. Takto do seba prvky viac zapadajú a vyzerajú jednotnejšie. Rozhranie po vykonaní popísaných zmien je zobrazené na obrázkoch 7.1 a 7.2.

Užívatelia by privítali možnosť prepisovania slov. Táto funkcia môže byť implementovaná pri budúcom vývoji aplikácie. Tiež môže byť pridaná legenda s vysvetlivkami zobrazení označenia slov, znakov, a ich pravdepodobností.

Priemerné hodnotenie aplikácie užívateľmi, ktorí ju testovali je 1,73. Aplikácia sa užívateľom páči a je podľa nich intuitívna.



Obr. 7.1: Snímka obrazovky aplikácie s implementovanými zmenami, ktoré boli navrhnuté užívateľmi počas testovania.



Obr. 7.2: Snímka obrazovky aplikácie s implementovanými zmenami, ktoré boli navrhnuté užívateľmi počas testovania. Tentokrát na mobilnom telefóne s povoleným pretáčaním obsahu.

Kapitola 8

Záver

Cieľom tejto práce bolo vytvoriť webovú aplikáciu s takým užívateľským rozhraním, ktoré bude umožňovať rýchlu a jednoduchú kontrolu a opravu textu rozpoznaného OCR nástrojom. Táto aplikácia mala byť tiež ľahko použiteľná na dotykových zariadeniach.

Vytvorená webová aplikácia bola implementovaná s použitím frameworku Vue JS. Aplikácia umožňuje jednoducho skontrolovať jednotlivé riadky rozpoznaného textu podľa náhľadu riadkov originálneho skenovaného dokumentu. Užívateľom pomáha rýchlo lokalizovať chyby v texte a následne ich opraviť výberom z možných variantov slov a znakov. Taktiež môže užívateľ ako pomôcku pri opravách využiť vizualizáciu pravdepodobnosti chýb a variantov. K aplikácii bola vytvorená aj databázová časť, ktorá zabezpečuje uloženie textových dát a ich aktualizáciu počas úprav textu.

Pred samotnou tvorbou aplikácie bol vytvorený prehľad existujúcich riešení tohto problému, z ktorých boli získané niektoré podnety pre návrh. Tiež bolo potrebné zoznámiť sa s použitím frameworku Vue a ďalšími použitými technológiami ako AWS EC2 alebo XAMPP.

Následne bol vytvorený prvotný návrh, podľa ktorého začala implementácia aplikácie. Návrh sa počas implementácie menil kvôli novým nápadom a podnetom, a podľa neho pokračovala implementácia. Po dokončení implementácie prebehlo testovanie s niekoľkými užívateľmi. Testovanie ukázalo, že aplikácia sa užívateľom páči, je jednoduchá a intuitívna na použitie, a celkovo sa im s ňou pracovalo dobre. Užívatelia mali po odskúšaní aplikácie tiež navrhnúť možné vylepšenia alebo úpravy aplikácie, ktoré by sa im v nej páčili. Niektoré drobné úpravy, ako napríklad zmena priehľadnosti pozadia znakov podľa ich pravdepodobnosti, alebo úprava vzhľadu riadkov boli implementované. Na väčšie úpravy aplikácie, ako napríklad možnosť úplného prepisovania textu však už nezostal čas.

V rámci budúceho vývoja aplikácie by do aplikácie mohla byť pridaná úvodná nápoveda, ktorá by užívateľom vysvetlila, ako s aplikáciou pracovať, a čo jednotlivé farebné označenia znamenajú. Tiež by mohla byť pridaná možnosť úplného prepisovania rozpoznaného textu. Prípadne by mohli byť neisté slová označené aj v náhľadoch riadkov, čo by však mohlo zhoršiť ich čitateľnosť, takže by bolo potrebné túto funkciu otestovať.

Literatúra

[1] Alkhaldi, N. *How optical character recognition algorithms redefine business processes?* [online]. April 2022 [cit. 2022-07-02]. Dostupné z: https://itrexgroup.com/blog/how-ocr-algorithms-redefine-business-processes/.

[2] App Weekly iOS. *How to scan documents to PDF using Tiny Scanner on iPhone?* [online]. October 2022 [cit. 2022-07-04]. Dostupné z: https://www.iosappweekly.com/scan-documents-to-pdf-tiny-scanner-iphone/.

[3] AWS. *What is Amazon EC2?* [online]. 2022 [cit. 2022-07-09]. Dostupné z: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html.

[4] Bates, S. *What is Axios?* [online]. December 2020 [cit. 2022-07-09]. Dostupné z: https://codebots.com/docs/what-is-axios.

[5] Boesch, G. *Optical Character Recognition (OCR) – Overview and Use Cases* [online]. 2022 [cit. 2022-07-02]. Dostupné z: https://viso.ai/computer-vision/optical-character-recognition-ocr/.

[6] GeeksforGeeks. *V-bind Directive in Vue.js* [online]. June 2022 [cit. 2022-07-08]. Dostupné z: https://www.geeksforgeeks.org/v-bind-directive-in-vue-js/.

[7] JavaScript, M. *Vue Props Tutorial* [online]. June 2019 [cit. 2022-07-08]. Dostupné z: https://masteringjs.io/tutorials/vue/props.

[8] Karandish, F. *The Comprehensive Guide to Optical Character Recognition (OCR)* [online]. 2022 [cit. 2022-07-03]. Dostupné z: https://moov.ai/en/blog/optical-character-recognition-ocr/.

[9] Klippa. *What is OCR? The Ultimate Guide to OCR 2022* [online]. April 2022 [cit. 2022-07-04]. Dostupné z: https://www.klippa.com/en/blog/information/what-is-ocr/.

[10] Kumbhar, M. F. *JavaScript HTML DOM* [online]. February 2022 [cit. 2022-07-07]. Dostupné z: https://dev.to/mursalfk/javascript-html-dom-177a.

[11] Lawson, E. *What is Vue Emit?* [online]. January 2022 [cit. 2022-07-08]. Dostupné z: https://javascript.works-hub.com/learn/what-is-vue-emit-47bc7.

[12] Lotanna, N. *Vue refs tutorial: Accessing DOM elements in a Vue.js app* [online]. December 2020 [cit. 2022-07-08]. Dostupné z: https://blog.logrocket.com/vue-refs-accessing-dom-elements/.

[13] Mozilla. *Web Docs MDN* [online]. 2022 [cit. 2022-07-07]. Dostupné z: https://developer.mozilla.org/.

[14] Olawanle, J. *How to Use Props in Vue.js* [online]. August 2021 [cit. 2022-07-08]. Dostupné z: https://www.freecodecamp.org/news/how-to-use-props-in-vuejs/.

[15] Ranjan, R. *What is a Framework in Programming & Why You Should Use One* [online]. October 2021 [cit. 2022-07-07]. Dostupné z: https://www.netsolutions.com/insights/what-is-a-framework-in-programming/.

[16] Šafo. *Úvod do Vue.js* [online]. 2019 [cit. 2022-07-07]. Dostupné z: https://blog.bart.sk/uvod-vue-js/.

[17] Sandhu, S. *How to Emit Data in Vue: Beyond the Vue.js Documentation* [online]. September 2018 [cit. 2022-07-08]. Dostupné z: https://www.telerik.com/blogs/how-to-emit-data-in-vue-beyond-the-vuejs-documentation.

[18] Srivastava, S., Verma, A. a Sharma, S. Optical Character Recognition Techniques: A Review. In: *2022 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*. 2022, s. 1–6. DOI: 10.1109/SCEECS54111.2022.9740911.

[19] SuperAnnotate. *What is Optical Character Recognition (OCR): Overview and use cases* [online]. October 2021 [cit. 2022-07-04]. Dostupné z: https://blog.superannotate.com/ocr-overview-and-use-cases/.

[20] TutorialsPoint. *VueJS - Overview* [online]. 2022 [cit. 2022-07-07]. Dostupné z: https://www.tutorialspoint.com/vuejs/vuejs_overview.htm.

[21] VueJS. *Vue JS Guide* [online]. 2022 [cit. 2022-07-07]. Dostupné z: https://vuejs.org/guide/introduction.html.

[22] Vynckier, I. *B Is for Binarize* [online]. 2022 [cit. 2022-07-03]. Dostupné z: https://how-ocr-works.com/OCR/binarization.html.

[23] Wikipedia. *Confusion network* [online]. 2021 [cit. 2022-07-10]. Dostupné z: https://en.wikipedia.org/wiki/Confusion_network.