VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Erik Hudcovský

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

EXTRAKCE INFORMACÍ Z OSOBNÍCH DOKLADŮ
EXTRACTION OF INFORMATION FROM IDENTITY DOCUMENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Erik Hudcovský**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Caha**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Erik Hudcovský ***ID:*** 204182

Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Extrakce informací z osobních dokladů

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s údaji uváděnými na osobních dokladech (občanský průkaz a cestovní pas). Vytvořte aplikaci, která z vybraného typu osobního dokladu dostupného ve formě obrázku extrahuje údaje o jeho drženi do formátu XML a JSON. Pro extrakci informací využijte vhodné volně dostupné nástroje typu OCR. Zaměřte se na strojově čitelné oblasti dokladů. Aplikaci sestavte v programovacím jazyce Python 3 nebo C#/.NET. Vytvořený kód vystavte pod licencí MIT na GitHub. Při použití jazyka Python aplikaci umístěte na repozitář PyPI.

DOPORUČENÁ LITERATURA:

[1] ŘÍHA, Zdeněk. Právní úprava elektronických identifikačních průkazů a cestovních dokladů v ČR [online]. Brno, 2013. Dostupné z: <https://is.muni.cz/th/xr5zc/>. Diplomová práce. Masarykova univerzita, Právnická fakulta. Vedoucí práce Libor Kyncl.

[2] Doc 9303, Machine Readable Travel Documents Part 5 - Specifications for TD1 Size Machine Readable Official Travel Documents (MROTDs). International Civil Aviation Organization, [online] 2015. Dostupné z: < https://www.icao.int/publications/Documents/9303_p5_cons_en.pdf>.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: Ing. Tomáš Caha

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

ABSTRAKT

Táto práca sa zaoberá spracovaním informácií z osobných dokladov (občianskeho preukazu alebo cestovného pasu) do formy, ktorá je ďalej jednoducho spracovateľná pre počítače a celkovo pre IT odvetvie. Toto spracovanie je implementované aplikáciou, ktorú som v rámci mojej bakalárskej práce vypracoval. Aplikácia obsahuje na vstupe naskenovaný doklad, typ dokladu a formu požadovaného výstupu. Na výstupe dostaneme typ dokladu v požadovanom formáte. Celá aplikácia používa ku svojmu fungovaniu externý OCR nástroj (OpticalCharacter Recognition, v preklade Optické rozoznávanie znakov), ktorý je implementovaný tak, aby sa dal čo najjednoduchšie nahradiť iným OCR nástrojom. V mojej aplikácií som použil Tesseract. Tento OCR nástroj je v rámci bezplatných OCR nástrojov ten najjednoduchší a najpresnejší zároveň. Taktiež má stále silnú podporu komunity a je naďalej rozvíjaný. V tejto práci som sa taktiež venoval jeho testovaniu, ako na mnou vytvorených vzorkách textu, tak aj na reálnych skenoch dokladov. Aplikácia je tiež spracovaná ako inštalačný balíček, takže môže byť jednoducho importovaná do iných projektov. Celá aplikácia je vystavená ako OpenSource na GitHube pod slobodnou licenciou MIT.

KĽÚČOVÉ SLOVÁ

extrakcia informácií, identifikačné doklady, optické rozoznávanie znakov, tesseract

ABSTRACT

This thesis is about the processing information from personal documents (ID card or passport) into the form that is further easily to be processed for computers and the IT industry in general. This process is implemented by the application I developed as part of my bachelor's thesis. The application contains the scanned document, the document type and the form of the required output. As the output we get the document type in the required format. The entire application is using in process an external OCR tool (OpticalCharacter Recognition), which is implemented so that it can be easily replaced by another OCR tool. I used Tesseract in my application. This OCR tool is the simpliest and most accurate of the free OCR tools at the same time. It also has strong community support and is still being developed. In this thesis, I also focused on its testing, both on the samples of text I created, and on real scans of documents. The application is also processed as an installation package, so it can be easily imported into other projects. The entire application is displayed as OpenSource on GitHube under the free license of MIT.

KEYWORDS

information extraction, identification cards, optical character recognizion, tesseract

HUDCOVSKÝ, Erik. *Extrakce informací z osobních dokladů*. Brno, 2020, 53 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Tomáš Caha

VYHLÁSENIE

Vyhlasujem, že svoju bakalársku prácu na tému „Extrakce informací z osobních dokladů“ som vypracoval samostatne pod vedením vedúceho bakalárskej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý 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 autora

POĎAKOVANIE

Rád by som poďakoval vedúcemu mojej bakalárskej práce pánovi Ing. Tomášovi Cahovi, za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov

Úvod

Táto bakalárska práca sa venuje problematike a následnému riešeniu spracovania údajov zo strojovo čitateľných častí identifikačných dokladov. V dnešnej dobe obsahuje väčšina identifikačných dokladov strojovo čitateľnú oblasť, ktorá je určená na strojové spracovanie. Za pomoci rôznych softvérov si dokážeme z tejto oblasti získať všetky potrebné informácie. V bakalárskej práci vytvorím aplikáciu, ktorá bude spracovávať údaje z občianskeho preukazu. Hlavné piliere tejto aplikácie sú jej rozšíriteľnosť, dostupnosť a poprípade jednoduchosť. Prácu som rozdelil do niekoľkých kapitol, z ktorých každá rieši inú časť problému. Kapitola 1 sa bude zaoberať jednotlivými dokladmi, ktoré budem zapracovávať do výslednej aplikácie. V tejto kapitole rozoberiem údaje, ktoré musia jednotlivé identifikačné karty obsahovať a taktiež rozoberiem ich strojovo čitateľné oblasti. V kapitole 2 sa zameriam na softvér optického rozoznávania znakov a vyberiem jeden, ktorý použijem pre aplikáciu. Kapitola 3 sa bude zaoberať priamo vývojom danej aplikácie. Poslednú kapitolu 4 chcem venovať príkladom použitia aplikácie s reálnymi dokladmi. Taktiež by som tu rád otestoval limity použitia aplikácie.

1 Doklady použité na spracovanie

Aplikácia, ktorú budem vo svojej bakalárskej práci vytvárať, sa bude zaoberať spracovaním osobných dokladov. V tejto kapitole sa budem venovať dokladom, na ktoré som sa zameral.

1.1 Občiansky preukaz

Je to verejná listina, ktorá slúži ako identifikačný doklad. Vydávaný je štátom, v ktorom má občan uvedené trvalé bydlisko. Každý štát môže mať vlastnú verziu svojho občianskeho preukazu, ale vo všeobecnosti platí, že tento preukaz obsahuje fotku občana a jeho základné informácie.

Česká verzia

Údaje obsiahnuté v českom občianskom preukaze sú definované v zákone č. 328/1999 Sb. verzie 26 Českého zákona.

Existuje viacero platných verzií občianskeho preukazu. Ja som sa zameral na tú najrozšírenejšiu verziu, čo je v dobe písania tejto bakalárskej práce občiansky preukaz vzor 2012. 1. januára 2012 vstúpila v platnosť novela zákona, ktorá prináša niekoľko noviniek, predovšetkým vo formáte preukazu, zapisovania údajov a získavania fotografií občanov. Podľa paragrafu 3 tohto zákona, občiansky preukaz obsahuje:

- meno, poprípade mená
- priezvisko
- pohlavie
- štátne občianstvo
- dátum a miesto narodenia
- rodné číslo
- adresa miesta trvalého pobytu, vrátane označenia tohto údaju ako adresy úradu
- úradné záznamy obsahujúce neskrátenú podobu mena
- dátum skončenia platnosti
- číslo občianskeho preukazu
- dátum vydania občianskeho preukazu
- označenie úradu, ktorý ho vydal[1]

(a) predná strana (b) zadná strana

Obr. 1.1: Český občiansky preukaz[1]

Slovenská verzia

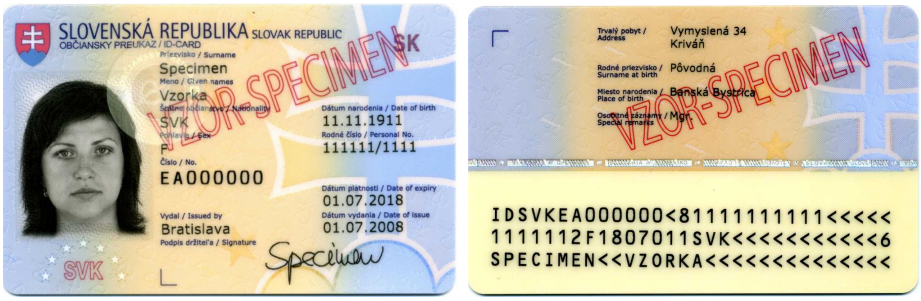
Podľa paragrafu 3 zákona č. 224/2006 Z. z. o občianskych preukazoch Slovenského zákona, občiansky preukaz obsahuje:

- meno
- priezvisko
- rodné priezvisko
- pohlavie
- štátne občianstvo
- dátum a miesto narodenia
- rodné číslo
- adresa trvalého pobytu občana
- dátum vydania
- dátum skončenia platnosti občianskeho preukazu
- miesto vydania
- strojovo čitateľné údaje
- zobrazenie podoby tváre
- zobrazenie podpisu držiteľa
- elektronický čip

Do občianskeho preukazu sa v špeciálnych prípadoch taktiež uvádzajú iné skutočnosti, ak je to potrebné. Napríklad, ak je občan nespôsobilý na právne úkony alebo má uložený trest zákazu pobytu, táto skutočnosť musí byť uvedená taktiež v občianskom preukaze. Občiansky preukaz taktiež obsahuje časť osobitné záznamy, do ktorej sa môžu na žiadosť občana uviesť niektoré z nasledujúcich skutočností:

- údaje o závažných chorobách, ktoré za určitých okolností vyžadujú okamžité podanie lieku alebo poskytnutie špeciálnej prvej pomoci po predložení potvrdenia od lekára
- údaje o krvnej skupine a podskupine po predložení potvrdenia od lekára

- skratka akademického titulu[2]



(a) predná strana (b) zadná strana

Obr. 1.2: Slovenský občiansky preukaz[17]

1.2 Cestovný pas

Cestovný pas je doklad vydávaný tým štátom, ktorý umožňuje vstup a prechádzanie inými krajinami. Každý štát má svoj vlastný vzor cestovného pasu. V niektorých krajinách, ako je napríklad Amerika, kde neexistujú občianske preukazy, môže plniť funkciu identifikačnej karty práve cestovný pas.

Česká verzia

Cestovný pas je zhotovený vo forme knižky o 34 stránkach. Od tretej stránky až po koniec je v dolnej časti perforované číslo pasu. Dátová stránka je umiestená na vyšitej polykarbonátovej karte. Vo vnútri dátovej stránky je vložený bezkontaktný čip, ktorý obsahuje biometrické údaje, ako obraz tváre a otlačky prstov. Neskorší pas, konkrétne verzia vydávaná od 1.1.2012, sa od pôvodnej verzie líšila iba v tom, že v položke „09 PAS VYDAL/AUTHORITY/AUTORITÉ“ sa tiskne (skrátený) názov vydávajúceho úradu mestskej časti, magistrátu alebo mestského úradu.[5]
Český cestovný pas musí zo zákona obsahovať tieto údaje:

- meno, prípadne mená alebo skrátená podoba mena
- priezvisko, rodné priezvisko
- agendový identifikátor fyzickej osoby
- rodné číslo
- dátum, miesto a okres narodenia
- pohlavie
- štátne občianstvo
- rodinný stav alebo partnerstvo, dátum uzavretia manželstva
- adresa miesta trvalého bydliska

- číslo a druh vydaného cestovného dokladu
- dátum vydania cestovného dokladu
- dátum skončenia platnosti cestovného dokladu
- označenie orgánu, ktorý cestovný doklad vydal
- digitálne spracovanie fotografie a podpisu držiteľa

(a) vzor používaný od 1.9.2006 (b) vzor používaný od 1.1.2012

Obr. 1.3: Cestovné pasy ČR

Slovenská verzia

Slovenská republika vo svojej samostatnosti vydávala zatiaľ tri druhy pasov. Platné sú samozrejme momentálne všetky tri, no niektoré z nich môžu byť neplatné pri cestovaní do niektorých krajín. Napríklad, prvé dve verzie sú neplatné pri ceste do USA, pretože USA vyžaduje biometrický pas, ktorý sa vydáva až od 15. januára 2008.[4].
Slovenský cestovný pas musí zo zákona obsahovať tieto údaje:

- meno
- priezvisko
- dátum narodenia
- rodné číslo
- pohlavie
- dátum narodenia
- štátnu príslušnosť(to sa nevzťahuje na cestovný doklad cudzinca)
- diplomatickú hodnosť alebo služobnú hodnosť, ak ide o diplomatický pas alebo služobný pas
- podobu tváre
- podpis
- štátne občianstvo
- biometrické údaje (podpis, otlačok prsta atď)[4]

(a) vzor používaný od 26.11.2014 (b) vzor používaný od 15.1.2008

(c) vzor používaný od 1.4.2005

Obr. 1.4: Cestovné pasy SR

1.3 Strojovo čitateľná oblasť dokladu

Je zóna nachádzajúca sa na väčšine identifikačných dokladov, ako sú pasy, víza, občianske preukazy a pod.. Táto zóna je štandardizovaná a dodržuje pravidlá, ktoré sa nachádzajú v ICAO Document 9303[6]. Keďže tento dokument vydáva organizácia pridružená k OSN, tak tieto pravidlá nemusia platiť v krajinách mimo OSN.

(a) občiansky preukaz[17] (b) cestovný pas[3]

Obr. 1.5: Ukážka strojovo čitateľnej oblasti

1.3.1 Občiansky preukaz

Strojovo čitateľná oblasť občianskeho preukaz,u na ktorý som sa zameral, obsahuje tieto údaje:

- kód dokladu
- kód vydávajúceho štátu
- číslo dokladu
- kontrolná číslica
- rodné číslo (iba v prípade slovenského občianskeho preukazu)
- dátum narodenia
- kontrolná číslica
- pohlavie
- dátum platnosti
- kontrolná číslica
- štátne občianstvo
- celková kontrolná číslica
- priezvisko
- meno

Ako strojovo čitateľná oblasť sa taktiež berie 2D čiarový kód (dvojdimenzionálny čiarový kód s vysokou informačnou hodnotou a schopnosťou detekcie a oprav pri jeho porušení) a kontaktný elektronický čip, ktorý obsahuje číslo občianskeho preukazu a identifikačné údaje držiteľa.[1]

1.3.2 Cestovný pas

Strojovo čitateľná oblasť cestovného pasu, na ktorý som sa zameral, obsahuje tieto údaje:

- kód dokladu
- kód vydávajúceho štátu
- meno
- priezvisko
- rodné číslo (iba v určitých vzoroch CP)
- číslo dokladu
- kontrolná číslica
- štátne občianstvo
- dátum narodenia
- kontrolná číslica
- pohlavie
- dátum platnosti
- kontrolná číslica

- rodné číslo
- celková kontrolná číslica

1.3.3 Validácia strojovo čitateľnej oblasti dokladov

Použitý je systém s váhami 1,3 a 7. Tento systém detekuje všetky jednociferné chyby a približne 90% chýb pri transpozícii. 1,3 a 7 sa používajú preto, že sú nedeliteľné číslom 10, takže zmena akejkoľvek číslice zmení kontrolnú číslicu (preto sa napríklad nemôžu použiť čísla 2 a 5). Tento systém kontroly sa vo veľkom používa v tranzitných číslach bankového smerovania v Spojených štátoch. Každý znak sa nahradí číselnou hodnotou. Cifry sú priamo touto hodnotou, znak '<' má hodnotu 0, znaky A–Z majú hodnoty 10–35.[6]

Príklad:
Platnosť do: 220808

	2		2		0		8		0		8				
×	7		3		1		7		3		1				
=	14	+	6	+	0	+	56	+	0	+	8	=	84	%	10	=	$\underline{\underline{4}}$

Tab. 1.1: Postup výpočtu kontrolnej číslice

Kontrolný znak je 4

2 Optické rozoznávanie znakov

Optické rozoznávanie znakov alebo OCR (z angl. Optical Character Recognition) je metóda umožňujúca preklad obrazu (grafiky) tlačených alebo písaných znakov do textovej, editovateľnej formy napr. do ASCII znakov abecedy. Takto spracovaný text má niekoľko výhod oproti naskenovanému textu. Medzi najhlavnejšie výhody patrí určite možnosť text ďalej editovať v bežnom editore a taktiež zmenšenie veľkosti daného súboru, ktorý už neuchováva pixely, ale iba jednoduché znaky. Na rozpoznávanie znakov z grafických súborov používame takzvané OCR softvéry. Prvý OCR systém bol patentovaný už v roku 1928 Gustavom Tauschekom. Myšlienka je založená na použití svetelných fotobuniek na rozpoznávanie svetelných vzorov na papiery alebo na karte.

2.1 Ako OCR funguje

Predpokladajme, že máme v abecede iba jedno jediné písmeno a to písmeno „A“. Aj pri takto zjednodušených podmienkach narážame na problém rôznorodosti. Každý človek má totižto iný rukopis a preto napíše písmeno „A“ trochu inak. Pri textových editoroch môžeme tento problém prirovnať k rôznym fontom, poprípade odtieňom. Tento problém sa rieši jedným alebo kombináciou nasledujúcich riešení:

a) **Rozpoznávanie vzorov**. Ak by sme donútili každého, aby napísal písmeno „A“ jedným spôsobom, ušetrili by sme hodiny výpočetného výkonu. Ako teda prinútiť všetkých, aby písali rovnakým spôsobom? Už v 60. rokoch 20. storočia bolo vyvinuté písmo s názvom OCR–A, ktoré sa dalo používať pri veciach ako bankové šeky, kreditné karty atď. Každé písmeno malo rovnakú šírku a ťahy boli starostlivo navrhnuté tak, aby bolo možné každé písmeno ľahko odlíšiť od ostatných. Kontrolné tlačiarne boli navrhnuté tak, aby všetky používali dané písmo, a zariadenie OCR bolo navrhnuté tak, aby ho tiež rozpoznávalo. Týmto krokom by bol problém vyriešený, ak by všetko písmo bolo tlačené v tomto fonte. Jediný problém je to, že väčšina z toho, čo svet tlačí, nie je napísaná v OCR–A, navyše nikto nepoužije toto písmo na písanie rukou. Aj keď sa databáza rozpoznávaných fontov rozrástla o ďalšie fonty, problém stále pretrvával.

b) **Detekcia prvkov**, tiež známa ako extrakcia znakov alebo inteligentné rozpoznávanie znakov (ICR), čo je oveľa sofistikovanejší spôsob zisťovania znakov. Predpokladajme, že OCR má vo svojej databázi všetky druhy fontov a fontov písma. Ako vyberieme font, ak by daný text vyzeral trošku inak? Môžeme použiť nasledujúce pravidlo: *Ak vidím dve šikmé čiary, ktoré sa stretávajú v bode hore, a medzi nimi asi v polovici je vodorovná čiara, toto písmeno bude*

pravdepodobne „A“. Použitím tohto pravidla budeme schopní rozoznať väčšinu písmen „A“ bez ohľadu na použitý font. Väčšina moderných všadeprítomných programov OCR (tých, ktoré sú zamerané na rozpoznávanie textu bez ohľadu na font), pracuje skôr s detekciou prvkov než s rozpoznávaním vzoru.

2.2 Čo ovplyvňuje výsledok

Vlastnosti, ktoré najviac vplývajú na výslednú presnosť výsledku, sú:

(1) **Dokument v papierovej forme**.Kvalita originálneho výtlačku výrazne ovplyvňuje presnosť procesu OCR. Špinavé značky, záhyby, škvrny od kávy, škvrny od atramentu a akékoľvek iné túlavé značky, znížia pravdepodobnosť správneho rozpoznávania písmen a slov.

(2) **Spôsob skenovania**.Výtlačok spustíte pomocou optického skenera. Skenery s podávaním listov sú pre OCR lepšie ako skenery s plochým sklom, pretože môžete skenovať strany jeden po druhom. Väčšina moderných programov OCR naskenuje každú stránku, rozpozná text na nej a potom automaticky naskenuje ďalšiu stránku. Ak používate plochý skener, musíte vložiť jednotlivé stránky ručne. Ak máte primerane dobrý digitálny fotoaparát, môžete vytvoriť fotografie svojich stránok pomocou fotografií. Pravdepodobne budete musieť použiť nastavenie makra (close-up), aby ste dostali skutočne ostré písmená, ktoré sú dostatočne jasné na presné OCR.

(3) **Kontrast dvoch farieb**.OCR je v podstate binárny proces: rozpoznáva veci, ktoré sú alebo nie sú. Ak je pôvodný naskenovaný obrázok dokonalý, akákoľvek čierna farba, ktorú obsahuje, bude súčasťou znaku, ktorý je potrebné rozpoznať, zatiaľ čo akákoľvek biela bude súčasťou pozadia. Redukcia obrázka na čiernu a bielu je preto prvou fázou pri zisťovaní textu, ktorý je potrebné spracovať. Ak máte farebné skenovanie novín s veľkou hnedou škvrnou kávy nad slovami, je ľahké rozoznať text od škvrny; ale ak zredukujete skenovanie na čiernobiely obrázok, škvrna sa zmení aj na čiernobielu a môže skresliť proces OCR.

(4) **Použité OCR**.Všetky programy OCR sa mierne líšia, spravidla však spracovávajú obrázok každej stránky rozpoznaním textového znaku po znakoch, slovo po slove a riadok po riadku.

(5) **Základná oprava chýb**.Sofistikované programy OCR majú špeciálne funkcie na kontrolu chýb, ktoré vám pomôžu spozorovať chyby. Niektorí napríklad používajú to, čo sa nazýva analýza blízkeho susedstva, aby našli slová, ktoré sa pravdepodobne vyskytnú v blízkosti, takže text nesprávne rozpoznaný ako „barking bog“ by sa mohol automaticky zmeniť na „barking dog“ (pretože „barking“ a „dog“ sú dve slová, ktoré sa veľmi často spájajú).

(6) **Analýza rozloženia**.Dobré programy OCR automaticky zisťujú zložité rozloženie stránky, napríklad niekoľko stĺpcov textu, tabuliek, obrázkov atď. Obrázky sa automaticky premenia na grafiku, tabuľky sa (so šťastím) premenia na tabuľky a stĺpce sa rozdelia správne (takže text z prvého riadku prvého stĺpca nie je automaticky spojený s textom z prvého riadku druhého stĺpec).

(7) **Korektúry**.Dokonca ani tie najlepšie programy OCR nie sú dokonalé, najmä ak pracujú z veľmi starých dokumentov alebo tlačeného textu v nízkej kvalite. Preto by posledná etapa v OCR mala byť vždy kontrola dobrým ľudským korektorom![7]

2.3 Tesseract

Aj keď od vedúceho bakalárskej práce mi bol doporučný Google Cloud Vision API, ja som sa v tejto bakalárskej práce rozhodol použiť Tesseract 4 ako OCR, na ktorom budem vyvíjať celú aplikáciu.

Tesseract bol pôvodne vyvinutý v Hewlett-Packard Laboratories Bristol a v Hewlett-Packard Co, Greeley Colorado v rokoch 1985 až 1994. V roku 2005 bol Tesseract zverejnený ako open source od spoločnosti HP. Od roku 2006 je vyvíjaný spoločnosťou Google. V roku 2006 bol tesseract považovaný za jeden z najpresnejších OCR softwérov. Tesseract 4 pridáva nový OCR engine založený na neurálnej sieti (LSTM), ktorý je zameraný na rozpoznávanie liniek, ale stále podporuje aj starý TCR Tesseract OCR engine Tesseract 3, ktorý funguje rozpoznávaním vzorov znakov. Kompatibilita s Tesseract 3 je povolená pomocou režimu Legacy OCR Engine (- 0). Potrebuje tiež školené súbory údajov, ktoré podporujú starý modul, napríklad súbory z úložiska `tessdata`. Hlavným vývojárom je Ray Smith. Správcom je Zdenko Podobny. Tesseract má taktiež podporu unicode (UTF-8) a dokáže rozoznať viac ako 100 jazykov[8]

2.3.1 Jazykové sady

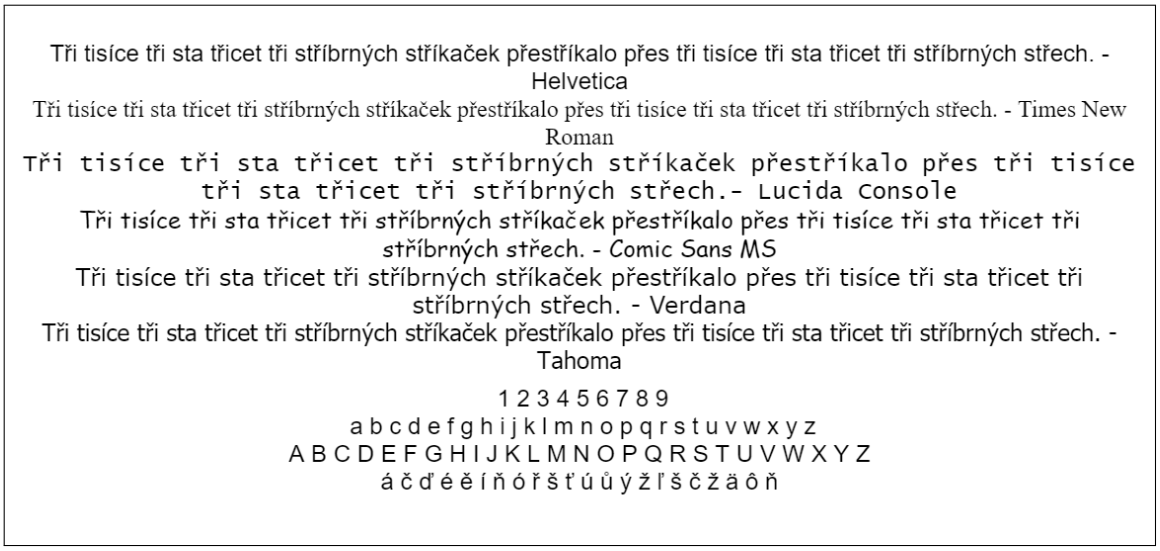
Ako som spomínal v sekcií 2.3 tesseract má podporu pre viac ako 100 jazykov. V praxi ale 100 jazykov väčšinou potrebovať nebudeme a keďže každý jazykový set môže zaberať niekoľko desiatok MB, rozhodol som sa použiť iba český a anglický dataset. Český dataset som sa rozhodol použiť preto, lebo v anglickom jazyku nie je interpunkcia. Zároveň sú v ňom ale takmer všetky písmená, ktoré sú aj v slovenskom jazyku, takže ak by som chcel aplikáciu časom rozšíriť na podrobnejšie čítanie vybraného dokladu, stačilo by zmeniť vybraný dataset na taký, ktorý odpovedá jazyku použitému na doklade. Anglický dataset (ten, ktorý je tam teraz štandardne nastavený) som použil preto, lebo obsahuje menej znakov a jeho dáta sú podrobnejšie

natrénované. Keďže táto bakalárska práca je zameraná hlavne na strojovo čitateľnú oblasť, ktorá nemôže obsahovať iné znaky než tie, ktoré sú definované v anglickej abecede, tento dataset je pre túto prácu dostačujúci.

2.3.2 Porovnanie rôznych použitých jazykových sád

Testovací obrázok

Veľkosť súboru	:	109 kB
Typ súboru	:	PNG
Šírka obrázku	:	1151
Výška obrázku	:	531
Megapixelov	:	0.611



Obr. 2.1: Testovací text

Výstupný text po použití anglickej jazykovej sady

Obr. 2.2: Zprocesovaný testovací text - použitý eng balíček

Výstupný text po použití českej jazykovej sady

Obr. 2.3: Zprocesovaný testovací text - použitý ces balíček

Aj keď sa môže zdať, že ani jeden z datasetov nevedel správne rozoznať daný text, opak je pravdou. Táto ukážka mala slúžiť len na znázornenie rozdielu medzi jednotlivými datasetmi. Kvalitu výstupu avšak určuje viacero faktorov, viz sekcia 2.2. V tomto príklade som zámerne zvolil menej kvalitný obrázok a to z toho dôvodu, aby rozdielnosti medzi jednotlivými datasetmi viac vynikli.

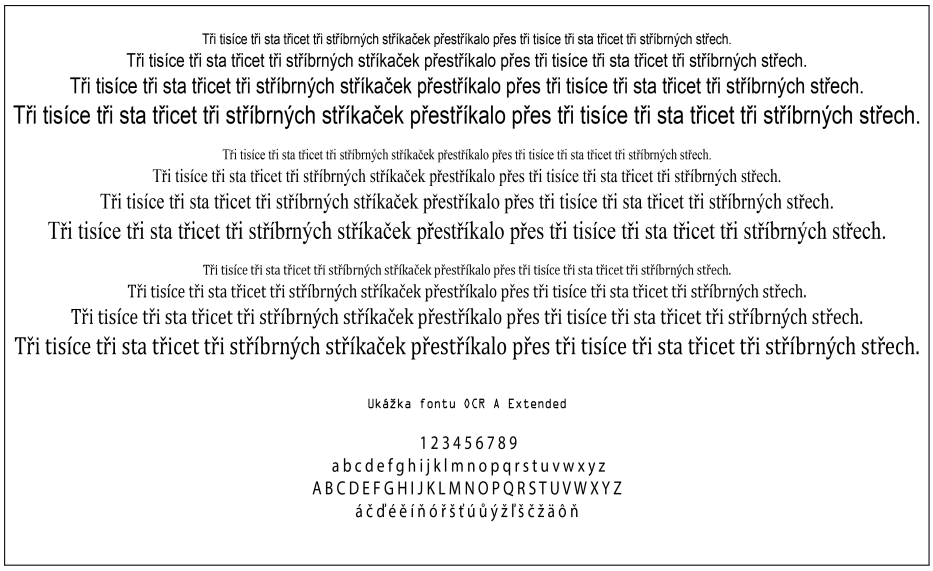
2.3.3 Kvalita obrázkov na vstupe

Ako som spomínal v sekcií 2.2, kvalita obrázkov hrá pri spracovaní obrázku veľkú rolu. Tesseract je softvér určený hlavne na spracovanie čistého textu a tak som ho podrobil skúške, kedy som vytvoril dva identické obrázky, ale exportoval som ich v inej kvalite. Pre oba obrázky som použil češtinu ako jazykový set.

Obrázok s vysokou kvalitou

Parametre obrázku:

Veľkosť súboru	:	191 kB
Typ súboru	:	PNG
Šírka obrázku	:	3596
Výška obrázku	:	2150
Pixelov na jednotku X	:	11811
Pixelov na jednotku Y	:	11811
Megapixelov	:	7,7



(a) vstupný obrázok

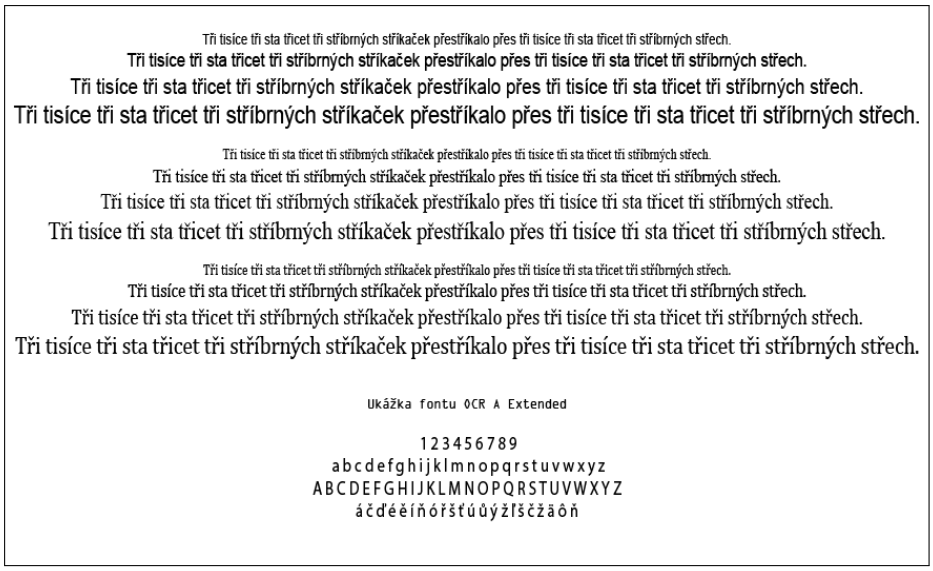
(b) výstupný text

Obr. 2.4: Spracovanie obrázku s vysokou kvalitou

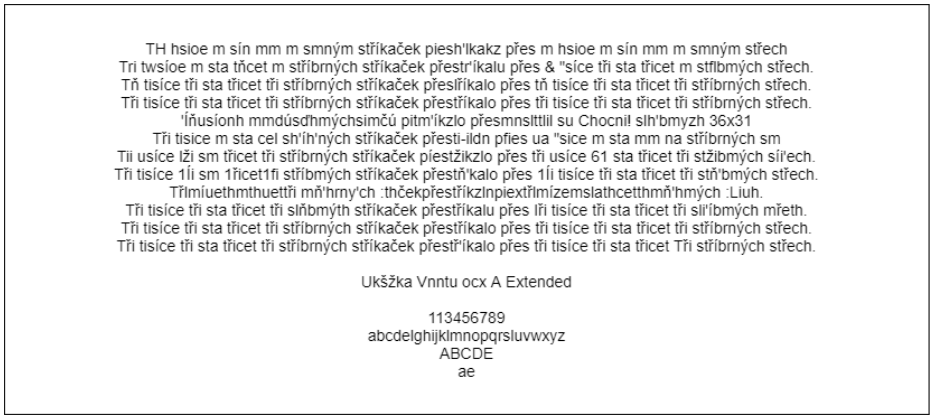
Obrázok s nízkou kvalitou

Parametre obrázku:

Veľkosť súboru	:	46 kB
Typ súboru	:	PNG
Šírka obrázku	:	864
Výška obrázku	:	517
Pixelov na jednotku X	:	2835
Pixelov na jednotku Y	:	2835
Megapixelov	:	0,447



(a) vstupný obrázok



(b) výstupný text

Obr. 2.5: Spracovanie obrázku s nízkou kvalitou

Na obrázku 2.4 môžeme vidieť, že pri vysokej kvalite vstupného obrázku tesseract spracoval takmer všetky znaky bez chyby. Naopak pri obrázku s nízkou kvalitou (obrázok 2.5) mal tesseract problém spracovať aj základné znaky a čísla.

3 Vývoj aplikácie

Zadaním tejto práce bolo vytvoriť aplikáciu, ktorá z vybraného typu osobného dokladu vo forme obrázku extrahuje údaje o jeho držiteľovi do formátu XML alebo JSON. Aplikácia sa hlavne zameriava na spracovanie strojovo čitateľnej oblasti dokladu. Požiadavka bola, aby vytvorený program bol písaný v C#/.NET. Vytvorený kód má byť pod licenciou MIT na GitHub–e. V spolupráci s vedúcim sme stanovili, že výsledná aplikácia:

- musí byť ľahko rozšíriteľná o ďalšie typy dokladov na spracovanie
- musia v nej byť ľahko zameniteľné použité nástroje, hlavne použité OCR
- mala by byť využiteľná aj s inými subjektmi než len s mojou fakultou resp. univerzitou

3.1 Štruktúra programu

Keďže program mal byť voľne prístupný verejnosti, bolo dôležité, aby bol program ľahko upraviteľný a ďalej rozšíriteľný. Pre tento dôvod som sa rozhodol, že vytvorím triedu `OCR.cs`, ktorej jedinou úlohou bude implementovať použité OCR. Každé ďalšie OCR, ktoré si bude chcieť užívateľ pridať, bude musieť mať vlastnú triedu, ktorá bude implementovať rozhranie `IProcess`. Rozhranie bude obsahovať premennú typu string `Text` a funkciu `Process` s návratovým typom string, ktorej parametre sú:

- `dataPath` (cesta k naskenovanému dokladu)
- `userName` (prihlasovacie meno na použitie vybraného API)
- `password` (heslo na použtie vybraného API)

```
interface IProcess
{
    string Text { get; set; }
    string Process(string dataPath,string userName,string password);
}
```

Z dôvodu prehľadnosti som sa rozhodol projekt kategoricky rozdeliť na viacero zložiek:

Zoznamy

- Sú umiestnené v zložke Enums.

a) **Sex** - Obsahuje všetky platné pohlavia podľa normy ISO/IEC 5218:2004 [13]

```
public enum Sex
```

```
{
    NotKnown = 0,
    Male = 1,
    Female = 2,
    NotApplicable = 9
}
```

b) **Nation** - Obsahuje platné krajiny spolu s ich numerickým kódom podľa normy ISO 3166-1.[14]

```
public enum Nationality
    {
        Andorra = 53,
        UnitedArabEmirates = 126,
        Afghanistan = 133,
        .
        .
        .
        CzechRepublic = 50,
        .
        Slovakia = 120,
        .
        .
        Zambia = 131,
        Zimbabwe = 132
    }
```

c) **Country** - Obsahuje platné krajiny spolu s ich kódom podľa normy ISO 3166-1 alpha-2.[14]

```
public enum Nationality
{
    AD,
    AE,
    AF,
    .
    .
    .
    CZ,
    .
    SK,
    .
    .
```

```
    ZM,
    ZW
}
```

d) **CardType** a **CardSubType** - Každý identifikačný preukaz má v strojovo čitateľnej oblasti určené, čo je to za typ preukazu. Tento typ sa obvykle skladá z dvoch písmen, preto som použil dva rozličné enumy, každý určený pre jedno písmeno typu preukazu.

```
public enum CardType
{
    Passport = 'P',
    Visa = 'V',
    IdentityCard = 'I'
}
```

```
public enum CardSubType
{
    IdentityCard = 'D',
    ResidencePermit = 'R'
}
```

Rozhrania

Sú umiestnené v zložke Interfaces. V tejto zložke sa nachádzajú dve rozhrania a to `ICardDataProcess` a `IOcrProcess`. `IOcrProcess` rozhranie je použité pri spracovaní použitých OCR. V skratke ide o to že každé požité OCR musí mať funkciu ktorá vráti všetok text z obrazového súboru definovaného jeho cestou.

```
internal interface IOcrProcess
{
    string Process(string dataPath);
}
```

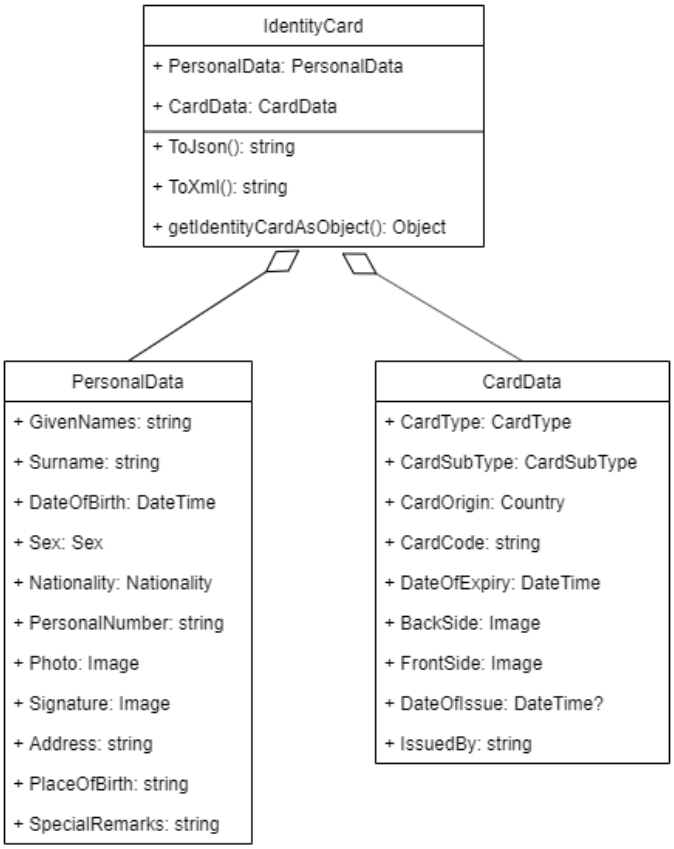
Rozhranie `ICardDataProcess` je použité priamo pre jednotlivé doklady a určuje funkcie ktoré má obsahovať trieda, ktorá sa bude zaoberať spracovaním hociakého dokladu.

```
internal interface ICardDataProcess
{
    IdentityCard getIdentityCard();
```

```
        string ProcessCard(IOcrProcess ocr, string backPageDataPath,
            string frontPageDataPath = null);
    }
```

Modely

Sú umiestnené v zložke Models.



Obr. 3.1: Model

V tejto zložke sa nachádza iba jeden model a tým je model `IdentityCard`, ktorý sa dalej rozdeluje na `CardData` a `PersonalData`. Ako je z názvu patrné, model `CardData` obsahuje všetky dáta týkajúce sa dokladu (dátum vydania, dátum platnosti, atď) a model `PersonalData` obsahuje všetky dáta týkajúce sa osoby ako takej (meno, priezvisko, dátum narodenia atď). Model `IdentityCard` taktiež obsahuje funkcie `ToJson()` a `ToXml()`, ktoré majú ako návratový typ string a ich úlohou je vrciať daný model buď v JSON formáte alebo v XML formáte

Pomocníci

Sú umiestnený v zložke Helpers. Pomocníci sú funkcie použité v mnohých projektoch na uľahčenie práce. Tieto funkcie sa väčšinou používajú v celom projekte a nie sú na nič viazané. Správne použitie týchto pomocníkov vyžaduje aby trieda a všetky jej funcie boli `public static`. Týmto spôsobom zaručíme že trieda sa bude dať volať odkiaľkolvek z projektu a jej použitie bude vyzerať nejak takto:

```
IDCard.CardData.BackSide =
    IdentityCardHelper.processCardImage(backPageDataPath);
```

V mojom projekte sa nachádza iba jeden helper a to `IdentityCardHelper`. Tento helper obsahuje všetky funkcie spojené zo spracovaním hociakého dokladu. Jeho obsahom sú napríklad Dictionaries, ktoré slúžia na rozlíšenie slovenských a českých dokladov, ako aj na rozlíšenie pohlaví a národností.

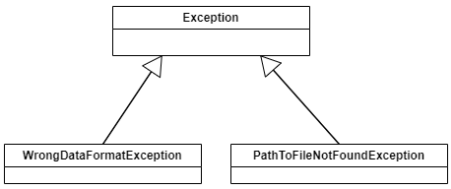
```
public static Dictionary<string, Nationality> Nations = new
    Dictionary<string, Nationality>()
{
    {"SVK",Nationality.Slovakia},
    {"CZE",Nationality.CzechRepublic},
};

public static Dictionary<string, Country> Countries = new
    Dictionary<string, Country>() {
    {"SVK",Country.SK},
    {"CZE",Country.CZ},
};

public static Dictionary<string, Sex> Genders = new
    Dictionary<string, Sex>()
{
    {"M",Sex.Male},
    {"F",Sex.Female},
    {"<",Sex.NotApplicable}
};
```

Ďalšou dôležitou funkciou ktorú `IdentityCardHelper` obsahuje je aj kontrola správnosti strojovo čitateľnej oblasti.

```
public static bool Validate(string stringOnValidate, int? validationValue)
        {
            var count = 0;
            var index = 7;
            for (int i = 0; i < stringOnValidate.Length; i++)
            {
                var letter = stringOnValidate[i];
                switch (index)
                {
                    case 7:
                        count = count + parseLetterToIntValue(letter) * 7;
                        index = 3;
                        break;

                    case 3:
                        count = count + parseLetterToIntValue(letter) * 3;
                        index = 1;
                        break;

                    case 1:
                        count = count + parseLetterToIntValue(letter) * 1;
                        index = 7;
                        break;

                    default:
                        break;
                }
            }

            return count % 10 == validationValue;
        }
```

Výnimky

Sú umiestnené v zložke Exceptions.



Obr. 3.2: Výnimky

a) **PathToFileNotFoundException** - Cesta k vybranému súboru nebola nájdená.
b) **WrongDataFormatException** - Dáta určené na spracovanie nie sú v správnom formáte.

Ukážka implementácie **WrongDataFormatException**. Táto výnimka je použitá, keď som podľa niektorých parametrov získaných informácií vyhodnotil, že daná informácia nie je valídna. Napríklad, ak som zistil že na mieste kde by malo byť rodné číslo, sa nenachádzajú iba číslice, viem s istotou povedať že nastala chyba.

```
/* Implementácia WrongDataFormatException */
[Serializable]
internal class WrongDataFormatException : Exception
{
    public WrongDataFormatException()
    {
    }

    public WrongDataFormatException(string errorMessge)
       : base(String.Format("Message: {0}", errorMessge))
    {
    }
}
```

Podobná implementácia je použitá aj pri **PathToFileNotFoundException**. Táto výnimka je použitá ak cesta k súboru s naskenovaným dokladom nie je valídna.

Použité OCR

Sú v zložke OCRs. Zložka obsahuje zatiaľ iba triedu `Tesseract4Process`. V tejto triede je implemetované Tesseract podľa rozhrania `IOcrProcess`.

```
/* Implementácia rozhrania IPocess */
internal class Tesseract4Process : IOcrProcess
```

```
{
    public float MeanConfidence { get; set; }
    public string Text { get; set; }
    private string userName { get; set; }
    private string password { get; set; }


    public Tesseract4Process(string userName = null, string password =
        null)
    {
        this.userName = userName;
        this.password = password;
    }

    public string Process(string dataPath)
    {
        try
        {
            // vytvorenie instancie použitého Tesseract enginu
            using (var engine = new TesseractEngine("", "ces",
                EngineMode.Default))
            {
                // spracovanie použitého obrázku
                using (var img = Pix.LoadFromFile(dataPath))
                {
                    // zprocesovanie načítaného obrázku vytvoreným
                        enginom
                    using (var page = engine.Process(img))
                    {
                        Text = page.GetText();
                        MeanConfidence = page.GetMeanConfidence();
                        return page.GetText();
                    }
                }
            }
        }
        catch (Exception e)
        {
            return "Unexpected Error: " + e.Message;
        }
    }
```

```
}
```

3.2 Implementácia

Funkcionalita celého programu je riadená triedou `Extractor`. V tejto triede sa nachádzaj[tri funkcie ktoré spájajú celý program dohromady.

Proccess()

Funkcia `Proccess`, ktorá spája celý program dohromady, a jej výstupom sú spracované doklady vo výslednom formáte.

```
public static string Process(string cardType, string backSidePath, string
    format = "JSON", string frontSidePath = null)
        {
            try
            {
                // inicializácia použitého OCR
                var ocr = new Tesseract4Process();

                switch (cardType)
                {
                    case "IC":
                        var identificationCard = new
                            IdentificationCardProcess(backSidePath,
                            frontSidePath, ocr);
                        return chooseOutput(format, identificationCard);

                    case "P":
                        var passport = new
                            PassportProcess(backSidePath,ocr);
                        return chooseOutput(format, passport);

                    default:
                        return "Card type was not recognized";
                }
            }
            catch (PathToFileNotFoundException ex)
            {
                throw ex;
            }
```

```
}
```

Ako si môžeme všimnúť, prvá vec ktorá sa v tejto funkcii vykoná, je inicializácia použitého OCR. Keďže ja som používal v celom programe iba jedno OCR, bolo pre mňa zbytočné tam pridávať nejakú funkciu ktorá by sa starala o použitie rôznych OCR. Tento spôsob implementácie je použitý práve preto, že väčšina OCR softwarov je platená, a teda obsahujú nejaké prihlasovanie údaje poprípade SSH kľúč na autorizáciu. Týmto spôsobom implementácie je možné dostiahnuť to, že prihlasovacie údaje k použitému OCR sa vyplnia len raz, a to pri inicializácii. Keďže Tesseract je voľne dostupný, a nepotrebuje žiadnu autorizáciu, jeho implementácia vyzerá nasledovne.

```
private string userName { get; set; }
private string password { get; set; }

// Username aj Password sú defaultne nastavené na null
public Tesseract4Process(string userName = null, string password =
    null)
{
    this.userName = userName;
    this.password = password;
}
```

Aj keď tieto prihlasovacie údaje nie sú nikde použité, ich implementácia je tam pridaná ako príklad budúceho použitia.

chooseOutput()

Ďalšou funkciou použitou v triede `Extractor` je funkcia `chooseOutput()`, ktorej vstupnými parametrami sú formát do ktorého má byť model prevedný a interface typu `ICardDataProcess`, ktorý zaistí že doklad ktorý chcem spracovať má implementovanú funkciu `getIdentityCard()`. Funkcia `chooseOutput()` má za úlohu previesť objekt `IdentityCard` do výslednej podoby XML alebo JSON. Defaultná implementácia je nastavená na prevedenie do JSON formátu a to z toho dôvodu, že je rozšírenejší pri webových aplikáciach.

Print()

Poslednou funkciou použitou v triede `Extractor` je funkcia `Print()`. Táto funkcia má za úlohu vypísať text spracovaný vybraným OCR. Táto funkcia mala pôvodne slúžiť čisto pre moje debugovacie účely a po dokončení programu som mal v pláne

ju odstrániť v rámci refactoringu, ale keďže sa overila ako dobrý pomocník pri debugovaní rozhodol som sa ju tam nechať. Funkcia ako taká dokáže dve základné veci. Dokáže vypísať čistý text tak ako ho vybrané OCR prečítalo a dokáže vypísať text ktorý prešiel všetkými kontrolami v rámci kontroli strojovo čitateľnej oblasti. Týmto spôsobom sa dá jednoducho rozlíšiť, či nastala chyba v kode, alebo pri spracovní obrázku.

3.3 Inštalácia vyvinutej aplikácie

Projekt je umiestnený v repozitári GitHub [1]. Projekt ako taký stačí stiahnuť a po kompilácií sa vytvorí výsledný CLI program ktorý je umiestnený v zložke bin. Program ako taký ale nebude fungovať bez zadaných parametrov.

(1) **Typ dokladu** - povinný, možné vstupy sú IC (Identification Card) a P (Passport)
(2) **Cesta k obrázku dokladu zo strojovo čitateěným polom** - povinný
(3) **Výstupný formát** - nepovinný (defaultne nastavený na JSON), možné parametre sú JSON alebo XML
(4) **Cesta k opačnej strane dokladu** - nepovinný

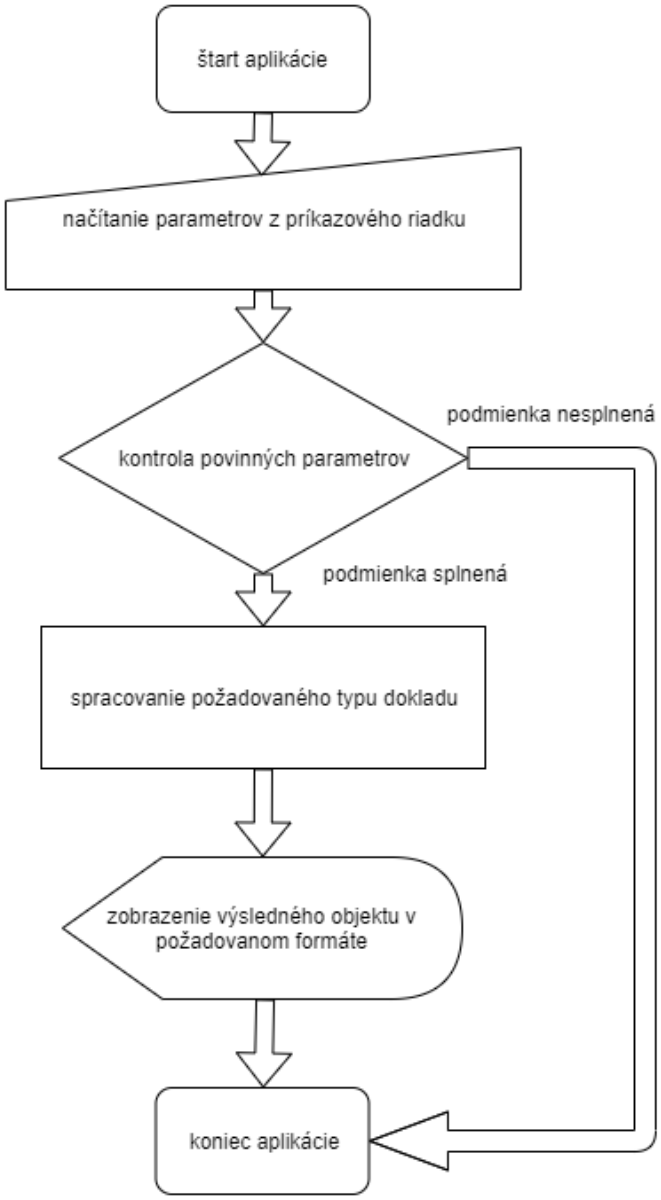
```
.\IdentityCardInformationExtractor.exe
P
"C:\Users\StrojovoCitatelnaOblastCP.png"
JSON
"C:\Users\PrednaStranaCP.png"
```

Obr. 3.3: Spustenie aplikácie s parametrami

3.3.1 Konzolová aplikácia s jednoduchým GUI

Na obrázku si môžeme všimnúť vývojový diagram mojej mojej aplikácie. Môj projekt je konzolová aplikácia, takže po štarte programu okamžite dochádza ku načítaní a kontrole parametrov. Podľa toho či sú parametre správne môj program buď pokračuje v spracovaní, alebo sa ukončí okamžite.

[1] https://github.com/erzik987/IdentityCardInformationExtractor



Obr. 3.4: Vývojový diagram konzolovej aplikácie

3.4 Inštalačný balíček

Súčasťou vypracovania mojej aplikácie bolo aj následné vytvorenie inštalačného balíčka, takzvaného nugetu, ktorý bude poskytovať funkcionalitu celého môjho programu. Tento balíček je voľne prístupný na stránke www.nuget.org [2]. Inštalácia tohto balíčka je možná viacerými spôsobmi. Najjednoduhší spôsob inštalácie je priamo z Visual Studia. Postup tejto inštalácie môžme nájsť na officiálnej stránke Microsoftu

[2]https://www.nuget.org/packages/IdentityCardInformationExtractor

www.docs.microsoft.com[3]. Po inštalácii poslednej verzie balíčka stačí použiť triedu `Extractor` a jej funkciu `process`.

Príklad použitia balíčka:

```
 /* Príklad použitia nuget balíčka IdentityCardInformationExtractor */
using IdentityCardInformationExtractor;
using System;

namespace IdCardInformationExtractorNugetTest
{
    class Program
    {
        static void Main(string[] args)
        {
            var result = Extractor.process("P",
                "C:\\Users\\erik.hudcovsky\\OneDrive - Solarwinds\\BP
                materials\\mojPas.png", "JSON");
            Console.WriteLine(result);
            Console.ReadLine();
        }
    }
}
```

[3]https://docs.microsoft.com/en-us/nuget/quickstart/install-and-use-a-package-in-visual-studio

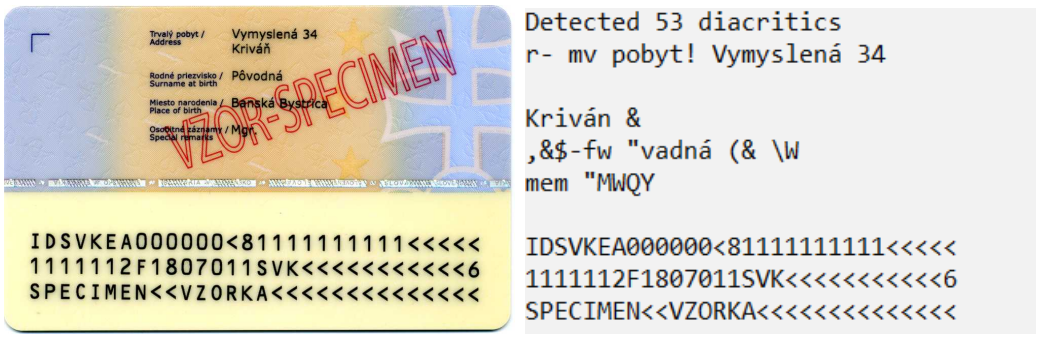
4 Vyhodnotenie

V tejto kapitole sa budem venovať hlavne testovaniu už dokončenej aplikácie. Pripravil som si pár najčastejších scenárov ktoré môžu nastať pri skenovaní dokumentu. Ku každému scenáru som dalej pridal návrh ako naskenovaný dokument bol v čo najlepšej kvalite pre spracovanie.

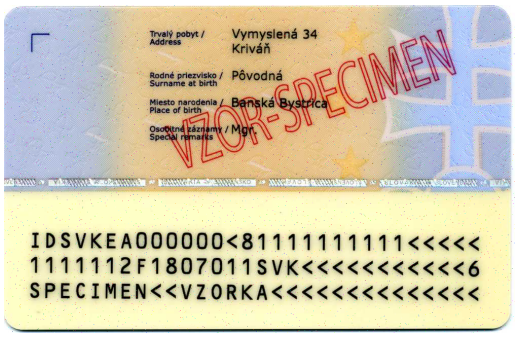
V tejto sekcii som si pripravil niekoľko reálnych ukážok mojej aplikácie a nejaké jej testy ktoré budú testovať okrajové situácie mojej aplikácie aby som zistil či je skutočne pripravená na reálne použitie. Na testovanie som použil slovenskú verziu občianskeho preukazu vzor 2013, a to z toho dôvodu, že som na oficiálnych stránkach Ministerstva vnútra Slovenskej Republiky našiel tento doklad v najlepšej kvalite. Na moje sklamanie som nemohol správne otestovať cestovný pas, pretože všetky obrázky ktoré som na internete našiel boli v príliš nízkej kvalite na to, aby mnou vybrané OCR dokázalo s istotou rozpoznať všetky znaky v strojovo čitateľnej oblasti. Vzhľadom na okolnosti v dobe písania tejto bakalárskej práce (COVID-19) som nebol schopný vyskúšať skenovanie na profesionálnej tlačiarni a preto som bol nútený si strojovo čitateľnú oblasť v cestovnom pase simulovať.

4.1 Zvýšený digitálny šum

Jednou z bežných nedokonalostí pri skenovaní hocijakého dokumentu je výskyt takzvaného digitálneho šumu. Čo to vlastne digitálny šum je? Pojem digitálny šum sa vzťahuje na vizuálne skreslenie. Šum na fotografii alebo oskenovanom dokumente vyzerá ako malé farebné pixely alebo škvrny a niekedy sa podobá na zrno.[15] https://www.colesclassroom.com/digital-noise-correct/

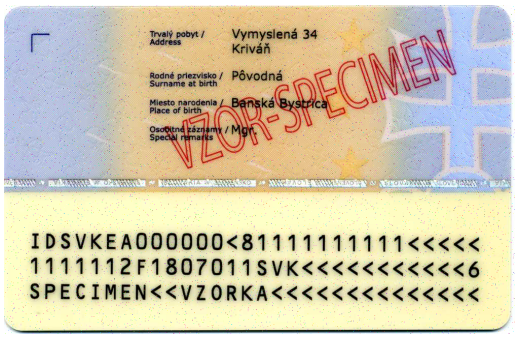


Obr. 4.1: Spracovanie OP v plnej kvalite




```
Warning. Invalid resolution 0 dpi. Using 70 instead.
Detected 61 diacritics
r" - mnobvt/ Vymyslená 34

* mx.. É" %QWÉ&
LW?

IDSVKEAD00000<81111111111<<<<<
1111112F1807011SVK<<<<<<<<<<<6
SPECIMEN<<VZORKA<<<<<<<<<<<<<

\_--_---/
```

Obr. 4.2: Spracovanie OP s 1% šumom




```
Warning. Invalid resolution 0 dpi. Using 70 instead.
Detected 83 diacritics
F- ,   " mvpouvt/ Vymyslená 34 _

' Kriván - "&
Sumlrmntbírml Puvodne QQ \W

mem WW
Hmofrs

'IDSVKEA000000<811111111111<<<<<
_1111112F1807011SVK<<<<<<<<<<<6_
_SPECIMEN<<VZORKA<<<<<<<<<<<<<

___-___-
```

Obr. 4.3: Spracovanie OP s 2% šumom

Z výsledkov môjho testovania je zrejmé že OCR, ktoré som použil je veľmi náchylné na šum v obrázku, a preto už pri 2% šumu s plnou silou šumu (šum bol úplne nepriehľadný), sa strojovo čitateľná oblasť deformuje do podoby, ktorú môj program vyhodnotí ako nevalídnu a preto vyhodí výnimku.

4.2 Zmena veľkosti

Ďalšou z vlastností, ktorá ovplyvňuje výsledok spracovania je veľkosť obrázka. V tomto teste som si pripravil obrázok z minulého testu v pôvodnej veľkosti a porovnal som jeho výstup s obrázkom v menšej veľkosti.

Obrázok v plnej veľkosti

Veľkosť súboru	:	79 kB
Šírka obrázku	:	1008 px
Výška obrázku	:	647 px


```
Detected 53 diacritics
r- mv pobyt! Vymyslená 34

Kriván &
,&$-fw "vadná (& \W
mem "MWQY

IDSVKEA000000<81111111111<<<<<
1111112F1807011SVK<<<<<<<<<<<6
SPECIMEN<<VZORKA<<<<<<<<<<<<<<
```

Obr. 4.4: Spracovanie OP v pôvodnej veľkosti

Obrázok v zmenšenej veľkosti

Veľkosť súboru	:	52,3 kB
Šírka obrázku	:	624 px
Výška obrázku	:	400 px


```
IDSVKEAODUOOO<81111111111<<<<<
1111112F1807011SVK<<<<<<<<<<<6
SPECIMEN<<VZORKA<<<<<<<<<<<<<<
```

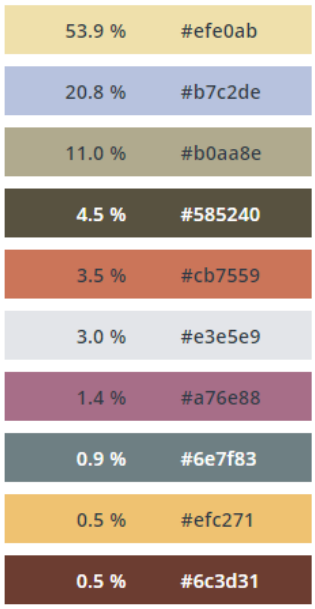
Obr. 4.5: Spracovanie OP v zmenšenej veľkosti

Z porovnania si môžeme všimnúť, že zatiaľ čo pri pôvodnej veľkosti bol pokus aj o spracovanie malého textu nad strojovo čitateľnou oblasťou, pri zmenšenej veľkosti bol tento text úplne odignorovaný. Ďalej môžeme vidieť že zmena veľkosti mala podiel na zmene vnímania znaku "0"v strojovo čitateľnej oblasti. Ako som už som spomínal vyššie, pri takto zmenenej štruktúre strojovo čitateľnej oblasti môj program zistí na základe verifikačných čísiel a formátov daných hodnôt že nastala chyba pravdepodobne pri transformácií obrazu na text a vyhodí výnimku.

4.3 Zmena farby

V tomto teste sa budem venovať rôznymi farebnými variáciami, ktoré môžu mať dopad na rozpoznávanie textu. Zameral som sa hlavne na variácie čiernej a bielej, pretože pri rozpoznávaní textu je táto kombinácia najideálnejšia, navyše je jednoduchšie previesť farebný obrázok na čiernobiely než meniť farby v obrázku. Na editáciu obrázkov som použil stránku `https://www5.lunapic.com/`.

Obrázok s pôvodnými farbami

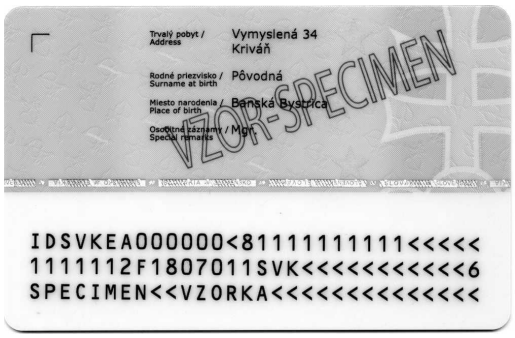



```
Detected 53 diacritics
r- mv pobyt! Vymyslená 34

Kriván &
,&$-fw "vadná (& \W
mem "MWQY

IDSVKEA000000<81111111111<<<<<
1111112F1807011SVK<<<<<<<<<<<6
SPECIMEN<<VZORKA<<<<<<<<<<<<<<
```

Obr. 4.6: Spracovanie v pôvodnej farbe

Čiernobiely obrázok

53.8 %	#c2c2c2
32.7 %	#f3f3f3
11.0 %	#797878
2.5 %	#424242

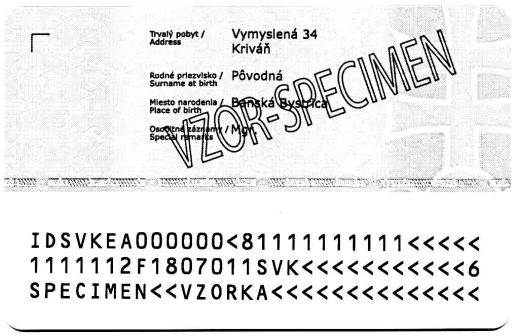



```
Detected 41 diacritics
f- mnv pobyt/ Vymyslená 34

Kriván &
,&...TM' Puvodne gg"
%$?
IDSVKEA000000<81111111111<<<<<

1111112F1807011SVK<<<<<<<<<<<6
SPECIMEN<<VZORKA<<<<<<<<<<<<<<
```

Obr. 4.7: Spracovanie o povodnej farbe

Obrázok v podobe harsh

86.6 %	#f5f5f5
9.9 %	#8d8d8d
3.5 %	#505050




```
Detected 149 diacritics
!- mgscwt/ Vymyslená 34

Kriván &
Rodu & pmmm ! Puvod ná \W
Su unum" C(S
Mleseo narodnn
71% ,c

IDSVKEADDOOOO<81111111111<<<<<
1111112F1807011SVK<<<<<<<<<<6
SPECIMEN<<VZORKA<<<<<<<<<<<<<<

\\\\\\
```

Obr. 4.8: Spracovanie o povodnej farbe

Z porovnania si hneď môžeme všimnúť, že zmena farby má skutočne vplyv na výsledok spracovania. Ako si môžeme všimnúť, spracovanie farebného obrázka a spracovanie čiernobieleho obrázka dopadlo vcelku podobne. To sa dá objasniť tým, že samotné OCR si pred spracovaním upraví obrázok do čiernobielej podoby kvôli spracovaniu. Tento fakt som objasňoval už v kapitole 2.2. Ja som sa rozhodol skúsiť upraviť obrázok ešte o trochu viac a konvertoval som ho do takzvaného modu "harsh". Ak sa bližšie pozrieme na rozdiely medzi obrázkom v tomto mode a čiernobielym obrázkom, zistíme, že tento mod znižuje počet šedých farieb na minimum. Tým pádom nám odstráni šedé okraje okolo písmen a rozpoznávanie textu je oveľa presnejšie. To si môžeme všimnúť aj na výstupe daného obrázka, ktorý spracoval omnoho viac písmen.

4.4 Rotácia

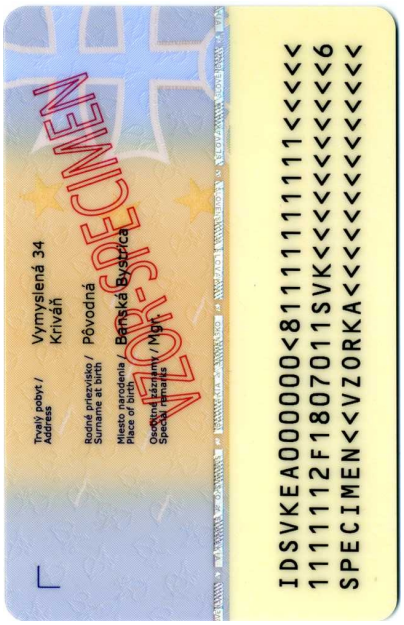
Taktiež rotácia môže pri niektorých OCR zohrávať dôležitú úlohu. V tomto teste si vyskúšame či rotácia má vplyv na výsledný text, a ak áno, aký veľký. Testovať budem tri najčastejšie scenáre, východiskový obrázok bude položený vodorovne so smerom textu. Prvý kontrolný obrázok bude pootočený o 90 stupňov a druhý o 180 stupňov.

Pôvodný obrázok (pootočenie 0 stupňov)


```
Detected 53 diacritics
r- mv pobyt! Vymyslená 34

Kriván &
,&$-fw "vadná (& \W
mem "MWQY

IDSVKEA000000<81111111111<<<<<
1111112F1807011SVK<<<<<<<<<<<6
SPECIMEN<<VZORKA<<<<<<<<<<<<<<
```

Obr. 4.9: Spracovanie obrázku otočeného v smere textu

Pôvodný obrázok (pootočenie 90 stupňov)




```
Detected 53 diacritics
r- mv pobyt! Vymyslená 34

Kriván &
,&$-fw "vadná (& \W
mem "MWQY

IDSVKEA000000<81111111111<<<<<
1111112F1807011SVK<<<<<<<<<<<6
SPECIMEN<<VZORKA<<<<<<<<<<<<<<
```

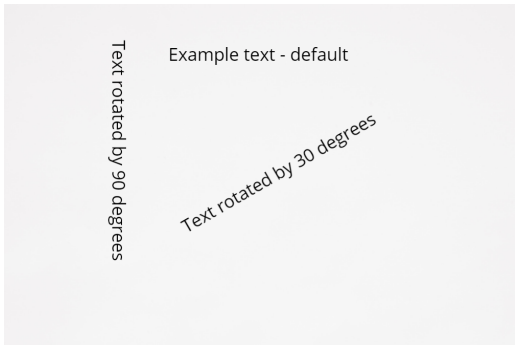
Obr. 4.10: Spracovanie obrázku otočeného o 90 stupňov

Pôvodný obrázok (pootočenie 180 stupňov)


```
Detected 53 diacritics
r- mv pobyt! Vymyslená 34

Kriván &
,&$-fw "vadná (& \W
mem "MWQY

IDSVKEA000000<81111111111<<<<<
1111112F1807011SVK<<<<<<<<<<<6
SPECIMEN<<VZORKA<<<<<<<<<<<<<<
```

Obr. 4.11: Spracovanie obrázku otočeného o 180°

Výsledky tohto testu ukázali že rotácia nemá žiaden vplyv na výsledný text. Po menšom prieskume som zistil že je to preto že tesseract rozpoznáva smer textu a sám si obrázok otočí do vodorovnej polohy. Tesseract si ale otáča celý obrázok a preto teoreticky nedokáže rozpoznať viacero pootočených textov v jednom obrázku. Tento fakt som sa rozhodol hneď aj overiť na mnou vytvorenom obrázku.




```
seejzep 06 Aq pelelw 1xe_|_

Example text - default
```

Obr. 4.12: Spracovanie obrázku s rôzne otočeným textom

Výsledok tohto testu dopadol podľa očakávaní. Tesseract skutočne nedokáže rozpoznať viacero úsekov textu, ktoré sú vzájomne pootáčané. Jedným zo spôsobov ako predísť tejto chybe pri spracovaní obrázku je upraviť si obrázok tak aby všetok text bol otočený jedným smerom alebo spracovávať úseky textu postupne. Pri spracovaní strojovo čitateľných oblastí našťastie tento scenár nemusím riešiť, ale je dobré vedieť že k niečomu takému môže dôjsť.

Záver

Cieľom mojej bakalárskej práce bolo zoznámenie sa s niektorými typmi identifikačných dokladov a následným vytvorením aplikácie, ktorá bude schopná tieto doklady spracovať do mnou vybraného formátu. Aplikácia mala byť vystavená na GitHub pod slobodnou licenciou MIT a následne z nej mal byť vytvorený nuget publikovaný na stránke `https://www.nuget.org`. Podrobnosti o detailoch jednotlivých identifikačných dokladov som zhrnul v kapitole 1. Môj hlavný zdroj informácií boli v tomto prípade zákony jednotlivých krajín z ktorých doklad pochádza. Pre zjednodušenie som sa zameral iba na doklady českej a slovenskej republiky. Keďže mojou úlohou je zamerať sa hlavne na strojovo čitateľnú oblasť, tak v sekcii 1.3 som sa zameral na jej problematiku. Popísal som tu aký formát majú strojovo čitateľné oblasti jednotlivých dokladov a taktiež som tu ukázal príklad kontrolných číslic použitých v týchto dokladoch. Mojou ďalšou úlohou bolo zvoliť si správny nástroj na čítanie týchto informácií z daného dokladu. Tejto problematike sa venujem v kapitole 2. Mojím cieľom bolo zvoliť si správny software na rozpoznávanie znakov. Po prieskume rôznych softwarov som sa rozhodol pre Tesseract 4.0 ktorý som bližšie opísal v sekcii 2.3. Samotným vývojom aplikácie sa zaoberám v kapitole 3. Venoval som sa tu ako inštalácii aplikácie (sekcia 3.3) tak aj použitým postupom a štruktúre programu (sekcia 3.1). Navrhnutým modelom sa mi podarilo docieliť ľahkú rozšíriteľnosť aplikácie. Tento fakt zdôrazňuje to, že ja sám som aplikáciu rozširoval o ďalší typ dokladu. V mojej poslednej kapitole, v kapitole 4, som sa venoval použitiu aplikácie v reálnom svete. Otestoval som aplikáciu nad rôznymi scenármi ktoré môžu nastať a zdôraznil som jej silné a slabé stránky. Celkovo by som zhodnotil moju aplikáciu veľmi pozitívne o čom svedčí aj fakt že inštalačný balíček môjho projektu má už niekoľko stoviek stiahnutí.

Literatúra

[1] Ministerstvo vnitra České republiky: *Občanský průkaz ČR* [online]. zákon č. 328/1999 Sb., o občanských průkazech. Česká republika: 30. 11. 1999, posledná aktualizácia 1. 7. 2018 [cit. 20. 11. 2019]. Dostupné z URL: <https://www.zakonyprolidi.cz/cs/1999-328>.

[2] Ministerstvo vnútra SR: *Občiansky preukaz SR* [online]. zákon č. 224/2006 Z. z. o občianskych preukazoch a o zmene a doplnení niektorých zákonov. Slovensko: 2013, posledná aktualizácia 7. 2. 2017 [cit. 17. 11. 2019]. Dostupné z URL: <https://www.slovensko.sk/sk/agendy/agenda/_obciansky-preukaz/>.

[3] Ministerstvo vnútra SR: *Cestovný pas SR* [online]. Slovensko: 2013, posledná aktualizácia 7. 2. 2017 [cit. 09. 12. 2019]. Dostupné z URL: <http://www.minv.sk/?vzory-dokladov-cestovne-pasy>.

[4] Ministerstvo vnútra SR: *Cestovný pas SR* [online]. Slovensko: 2007, posledná aktualizácia 10. 12. 2019 [cit. 10. 12. 2019]. Dostupné z URL: <https://www.slov-lex.sk/pravne-predpisy/SK/ZZ/2007/647/>.

[5] Ministerstvo vnútra ČR: *Cestovný pas ČR* [online]. Česká republika: posledná aktualizácia 17. 6. 2017 [cit. 25. 4. 2020]. Dostupné z URL: <https://www.mvcr.cz/clanek/rady-a-sluzby-dokumenty-platne-typy-cestovnich-d aspx>.

[6] ICAO Headquarters: *Strojovo čitateľná oblasť dokladov* [online]. Definícia pravidiel strojovo čitateľnej oblasti Slovensko: [cit. 20. 11. 2019]. Dostupné z URL: <https://www.icao.int/publications/pages/publication.aspx?docnum=9303>.

[7] Woodford, Chris: *Ako funguje OCR* [online]. Obecné vysvetlenie fungovania OCR Slovensko: 2010/2018 [cit. 20. 11. 2019]. Dostupné z URL: <https://www.explainthatstuff.com/how-ocr-works.html>.

[8] Ray Smith: *Tesseract* [online]. Tesseract 4 2019 [cit. 08. 12. 2019]. Dostupné z URL: <https://github.com/tesseract-ocr/tesseract/blob/master/README.md>.

[9] VUT v Brně: *Úprava, odevzdávání a zveřejňování vysokoškolských kvalifikačních prací na VUT v Brně* [online]. Směrnice rektora č. 2/2009. Brno: 2009, poslední aktualizace 24. 3. 2009 [cit. 23. 10. 2015]. Dostupné z URL:

<https://www.vutbr.cz/uredni-deska/vnitrni-predpisy-a-dokumenty/smernice-rektora-f34920/>.

[10] Microsoft: *Úvod do C#* [online]. Úvod do C# Unknown: 2017 [cit. 27. 11. 2019]. Dostupné z URL:
<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction>.

[11] Microsoft: *.NET architektúra* [online]. .NET architektúra Unknown: 2015 [cit. 27. 11. 2019]. Dostupné z URL:
<https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework>.

[12] Microsoft: *Úvod do .NET* [online]. Úvod do .NET Unknown: 2017 [cit. 27. 11. 2019]. Dostupné z URL:
<https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview>.

[13] ISO/IEC 5218: *Codes for the representation of human sexes* [online]. Codes for the representation of human sexes Unknown: 2004 [cit. 21. 12. 2019]. Dostupné z URL:
<https://www.iso.org/standard/36266.html>.

[14] ISO 3166-2: *Country subdivision code* [online]. Country subdivision code 2013 [cit. 21. 12. 2019]. Dostupné z URL:
<https://www.iso.org/standard/63546.html>.

[15] ICAO Headquarters: *Digitálny šum* [online]. Čo je to digitálny šum Unknown: [cit. 25. 4. 2020]. Dostupné z URL:
<https://www.colesclassroom.com/digital-noise-correct>.

[16] Ecma International 2017; What is JSON? In *IETF RFC 8259 The JavaScript Object Notation (JSON) Data Interchange Format*, Brno: VUT Brno, 2019

[17] Ministerstvo vnútra SR: *Občiansky preukaz SR* [online]. Občiansky preukaz SR s čipom Bratislava: 2014 [cit. 27. 11. 2019]. Dostupné z URL:
<https://portal.minv.sk/wps/wcm/connect/sk/site/main/obciansky-preukaz-s-cipom>.

Zoznam symbolov, veličín a skratiek

OCR	optické rozoznávanie znakov – Optical Character Recognition
ICR	inteligentné rozoznávanie znakov – Inteligent Character Recognition
CP	cestovný pas
OP	občiansky preukaz
GUI	grafické rozhranie - Graphic User Interface
SR	Slovenská Republika
ČR	Česká Republika
ICAO	Medzinárodná organizácia pre civilné letectvo - International Civil Aviation Organization
OSN	Organizácia Spojených národov
USA	Spojené štáty americké - United States of America
ASCII	American Standard Code for Information Interchange
XML	eXtensible Markup Language
JSON	JavaScript Object Notation
MIT	Massachusetts Institute of Technology
API	Application Programming Interface

Zoznam príloh

A Projekt umiestnený na GitHube

Projekt je umiestnený v GitHub repozitári na adrese
https://github.com/erzik987/IdentityCardInformationExtractor

```
IdentityCardInformationExtractor
├── Cards
│   ├── IdentificationCardProcess.cs
│   └── PassportProcess.cs
├── Enums
│   ├── CardSubType.cs
│   ├── CardType.cs
│   ├── Country.cs
│   ├── Nation.cs
│   └── Sex.cs
├── Exceptions
│   ├── PathToFileNotFoundException.cs
│   └── WrongDataFormatException.cs
├── Helpers
│   └── IdentityCardHelper.cs
├── Interfaces
│   ├── ICardDataProcess.cs
│   └── IOcrProcess.cs
├── Models
│   ├── CardData.cs
│   ├── IdentityCard.cs
│   └── PersonalData.cs
├── OCRs
│   └── Tesseract4Process.cs
├── Properties
│   └── launchSettings.json
├── tessdata
│   ├── ces.traineddata
│   └── eng.traineddata
├── Extractor.cs
├── IdentityCardInformationExtractor.csproj
├── IdentityCardInformationExtractor.nuspec
├── IdentityCardInformationExtractor.sln
└── Program.cs
```