Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vytěžení všech výskytů regulárního výrazu a jejich uložení při těžbě dokumentů v produktu IBM Datacap

Nikola Novotná (roz. Ježková)

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Nikola Ježková

Systémové inženýrství a informatika
Informatika

Název práce

Vytěžení všech výskytů regulárního výrazu a jejich uložení při těžbě dokumentů v produktu IBM Datacap

Název anglicky

Processing all occurrences of a regular expression, and storing them when extracting documents in IBM Datacap

---

Cíle práce

Cílem práce je vytvoření komponenty, která usnadní vytěžování všech výskytů regulárního výrazu na zpracovávané straně dokumentu pomocí vytěžovacího nástroje IBM Datacap. Těžbu dat a jejich uložení ke straně dokumentu provede samotná komponenta implementovaná v software IBM Datacap.

Metodika

Práce sestává ze dvou hlavních částí – přehledu teoretických východisek a praktické části.

Metodika zpracování teoretické části je založena na studiu odborných informačních zdrojů. Na základě syntézy zjištěných poznatků bude popsána problematika související s využitými technologiemi a nástroji IBM.

Praktická část práce spočívá v analýze, návrhu a implementaci modulu, který bude sloužit k vyhledání a uložení dat pomocí nástroje IBM Datacap. Při návrhu bude využito standardních metod a nástrojů softwarového inženýrství, implementace bude provedena primárně pomocí jazyka C#. Vytvořený modul bude nasazen a otestován a na základě zkušeností z provozu budou navrženy případné další možnosti jeho budoucího rozvoje.

Doporučený rozsah práce

35-40 stran

Klíčová slova

datacap, regulární výrazy, import, ibm, c#

Doporučené zdroje informací

C# bez předchozích znalostí, Pavel Bory, ISBN: 978-80-251-4686-6
https://msdn.microsoft.com/en-us/library/t0aew7h6.aspx
https://www.ibm.com/support/knowledgecenter/en/SSZRWV_9.1.7/com.ibm.datacaptoc.doc/datacap_9.1.7.htr
IBM Datacap 9.1.7 Dokumentace

Předběžný termín obhajoby

2020/21 ZS – PEF (únor 2021)

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 11. 1. 2018

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 11. 1. 2018

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 02. 11. 2020

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Vytěžení všech výskytů regulárního výrazu a jejich uložení při těžbě dokumentů v produktu IBM Datacap" jsem vypracovala samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autorka uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušila autorská práva třetích osob.

Licenci pro software IBM Datacap jsem měla k dispozici jako zaměstnanec firmy scanservice a. s., která je také Business Partnerem firmy IBM.

V Praze 29.11.2020 _____

Poděkování

Rád(a) bych touto cestou poděkoval(a) mému vedoucímu práce, panu doktoru Brožkovi a firmě scanservice a. s. za možnost využití licence pro IBM Datacap.

Vytěžení všech výskytů regulárního výrazu a jejich uložení při těžbě dokumentů v produktu IBM Datacap

Abstrakt

Cílem práce je vytvoření komponenty, která usnadní vytěžování všech výskytů regulárního výrazu na zpracovávané straně dokumentu pomocí vytěžovacího nástroje IBM Datacap. Těžbu dat a jejich uložení ke straně dokumentu provede samotná komponenta implementovaná v softwaru IBM Datacap.

Klíčová slova: IBM, Datacap, regulární výrazy, import, c#

Processing all occurrences of a regular expression, and storing them when extracing documents in IBM Datacap

Abstract

The goal of this thesis is to create a component which will help with processing of all occurrences of a regular expression on the processed page with software IBM Datacap. The data extraction and saving of these mined data will be done by this component which will be integrated into the IBM Datacap software.

Keywords: IBM, Datacap, regular expressions,import, c#

Obsah

1. Úvod 12

2. Cíl práce a metodika 13

2.1. Cíl práce 13

2.2. Metodika 13

3. Teoretická východiska 14

3.1. Vytěžování dat 14

3.1.1. Teorie vytěžování dat – textu 14

3.1.2. Jaký je rozdíl mezi digitalizací a vytěžováním dat? 14

3.1.3. Druh dokumentů 14

3.1.3.1. Strukturované dokumenty 14

3.1.3.2. Polostrukturované dokumenty 15

3.1.3.3. Nestrukturované dokumenty 16

3.1.4. Použití vytěžování v praxi 17

3.2. IBM Datacap 19

3.2.1. Stručná historie 19

3.2.2. Současnost 19

3.2.3. Struktura platformy IBM Datacap 19

3.2.4. Aplikace 20

3.2.4.1. Workflow 20

3.2.4.2. Task – dílčí úloha 20

3.2.4.3. Ruleset – set pravidel 21

3.2.4.4. Rule – pravidlo 21

3.2.4.5. Function – funkce 21

3.2.4.6. Action – akce 22
3.2.5. Struktura DCO (10) 22
3.2.5.1. Batch – dávka 23
3.2.5.2. Document – dokument 23
3.2.5.3. Page – strana 24
3.2.5.4. Field – pole 24
3.2.6. Smart Parametry 24
3.2.7. Vývoj aplikace 24
3.2.8. Doprogramování funkcionalit, resp. komponent, resp. custom akcí 25
3.3. Regulární výrazy 25
3.3.1. Co je to regulární výraz 25
3.3.2. Tabulka substitucí 26
3.3.3. Regulární výrazy v Datacapu 27
3.3.3.1. Vytěžování dle regulárních výrazů 27
3.3.3.2. Validace dat regulárním výrazem 28
3.4. C# – programovací jazyk 28
3.4.1. C# v kombinaci s Datacapem 29
4. Vlastní práce 30
4.1. Problematika vytěžování regulárních výrazů IBM Datacap 30
4.1.1. Popis problému? Z čeho vychází práce? 30
4.1.2. Analýza řešení 31
4.1.2.1. Návrh řešení 31
4.1.2.2. Klíčové body 32
4.1.3. Prerekvizity pro implementaci 33
4.1.3.1. Systémové prerekvizity 33
4.1.3.2. Znalostní prerekvizity 33

4.1.4. Použité techniky 33
4.1.4.1. Cyklus foreach 33
4.1.4.2. Načtení parametrů 34
4.1.4.3. Podpora SMART parametrů 35
4.1.4.4. Ošetření existence souborů 36
4.1.4.5. Práce s KEY souborem 37
4.1.4.6. Práce s XML souborem 37
4.1.4.7. Nalezení shody s regulárními výrazy 38
4.1.4.8. Vytvoření pole 38
4.1.4.9. Uložení hodnoty a pozice do pole 39
4.1.4.10. Zápis do logu dílčí úlohy 40
4.1.4.11. Návrat hodnoty – výsledku – akce 40

5. Výsledky a diskuse 41

5.1. Výsledek implementace 41
5.1.1. Testy v aplikaci 41
5.1.1.1. Ruleset s implementovanou akcí 41
5.1.1.2. Přiřazení akce do DCO 42
5.1.1.3. Výsledek akce nad rozpoznanou stranou 42
5.2. Zhodnocení 43

6. Závěr 44

7. Seznam použitých zdrojů 45

8. Přílohy 47

Seznam obrázků

Obr. 1 Strukturovaný dokument (4) 15
Obr. 2 Polostrukturovaný dokument (18) 16
Obr. 3 Nestrukturovaný dokument (17) 17
Obr. 4 DCO Hierarchie (zdroj: autor) 23
Obr. 5 Vzorový regulární výraz (zdroj: autor) 28
Obr. 6 XML c.xml (zdroj: autor) 32
Obr. 7 Okno s dokumentací (zdroj: autor) 34
Obr. 8 Ukázka implementace (zdroj: autor) 41
Obr. 9 Přiřazení do hierarchie (zdroj: autor) 42
Obr. 10 Výsledek z testu (zdroj: autor) 42

Seznam tabulek

Tabulka 1 Regulární substituční tabulka (13) 26

Seznam použitých zkratek

OCR – optical character recognition

Digitalizace – převedení analogu do digitální podoby

IBM – firma, která vlastní produkt Datacap

DataCap – software pro Data Capture – vytěžování dat

Metadata – vytěžovaná pole

Workflow – postup procesu, zde proces např. dávky

Usecase – téma či typ projektu – např. usecase zpracování faktur

Custom akce – akce do Datacapu doprogramovaná třetím subjektem

1. Úvod

V rámci této práce autorka bude řešit jeden z nedostatků funkcionality softwaru pro vytěžování dat IBM Datacap. IBM Datacap v tuto chvíli nemá funkcionalitu vytěžení většího množství výskytu dat, které vyhovují zadanému regulárnímu výrazu resp. výrazům, naráz, a to včetně jeho (jejich) uložení.

IBM Datacap je software, který se využívá pro tvorbu aplikací na vytěžování a zpracování dat. Tento systém přichází s množinou již předimplementovaných funkcionalit, které se při tvorbě aplikací dají použít. Autorka v rámci své praxe narazila na vícero usecasů, které by výše popsanou funkcionalitu využily pro lepší výsledky a taktéž pro snížení času jak na vývoj samotné aplikace, tak při provozu.

Funkcionality do systému IBM Datacap lze doprogramovat například za pomoci C# a tato cesta bude také použita v této práci.

Autorka daný problém v rámci práce zanalyzuje, navrhne řešení a toto řešení implementuje včetně zkušebních testů pro ověření funkčnosti.

2. Cíl práce a metodika

2.1. Cíl práce

Cílem této práce je zanalyzovat, navrhnout, naimplementovat a otestovat novou funkcionalitu – akci do softwaru IBM Datacap, která bude schopna vytěžit a uložit slova na straně, která vyhovují zadaným regulárním výrazům. Regulární výrazy a veškeré uživatelské parametry nutné pro chod komponenty budou zaintegrovány pro použití přímo z vývojového studia produktu IBM Datacap, což umožní programátorovi využít tuto doimplementovanou funkci bez ztráty komfortu známého prostředí.

Cílem této práce je tudíž rozšířit funkcionalitu IBM Datacap o vlastnost, kterou v době práce tento software neoplývá a která je dle autorky využitelná v mnoha praktických situacích.

2.2. Metodika

Jako primární přístup autorka práce zvolila rozdělení na teoretickou část a praktickou.

V rámci teoretické části budou studovány odborné zdroje a manuály, které jsou dostupné k produktu IBM Datacap, a to v rozsahu jak informativním – hlubší poznání softwaru jako takového, jeho komponent a struktur, tak i implementačním – rozbor, jak lze doimplementovat požadovanou funkcionalitu a integrovat ji. Důraz bude kladen i na studium zdrojů ohledně regulárních výrazů a rozšiřování znalosti o C#.

V praktické části bude již autorka zaměřena primárně na analýzu konkrétního problému, tudíž návrhu, jak docílit požadovaného výsledku, co je konkrétním cílem, jaká jsou rizika a vlastní use-case daného návrhu. Dalším krokem je jasný návrh, jak by vše šlo zpracovat, vydefinování klíčových bodů, které bude nutné splnit včetně všech nutných prerekvizit a použití konkrétních technik.

Autorka práce danou funkcionalitu bude navrhovat v souladu s vodopádovým přístupem k vývoji SW. Po samotné implementaci tedy proběhne test řešení a řešení bude (po úspěšném testu) možno použít v produkci.

Použití vodopádového přístupu bude vhodné z důvodu jasně definovaných požadavků na začátku a téže z důvodu přehlednosti celého postupu.

3. Teoretická východiska

3.1. Vytěžování dat

3.1.1. Teorie vytěžování dat – textu

Vytěžování dat, hlavně to automatické, je bráno jako zrychlení zpracování dat vyskytujících se jak na papírové, tak překvapivě i elektronické podobě dokumentu. Vytěžovat (a potažmo i digitalizovat) se dá víceméně jakýkoliv typ dokumentu. Avšak mezi jedny z častějších patří formuláře, smlouvy, faktury, daňové doklady. (1) Co se týče digitalizace, tak tam mezi zpracovávané typy dokumentů patří například i knihy. My se v této práci budeme hlavně zaobírat částí vytěžování textu za pomoci OCR.

3.1.2. Jaký je rozdíl mezi digitalizací a vytěžováním dat?

Digitalizací se myslí obecné převedení fyzického (analogového) dokumentu do digitální podoby. Vytěžování dat potřebuje ke své činnosti již tento digitální formát. Proto jdou tyto disciplíny ruku v ruce. Alias – digitalizaci bez vytěžování uděláte, vytěžení nikoliv.

3.1.3. Druh dokumentů

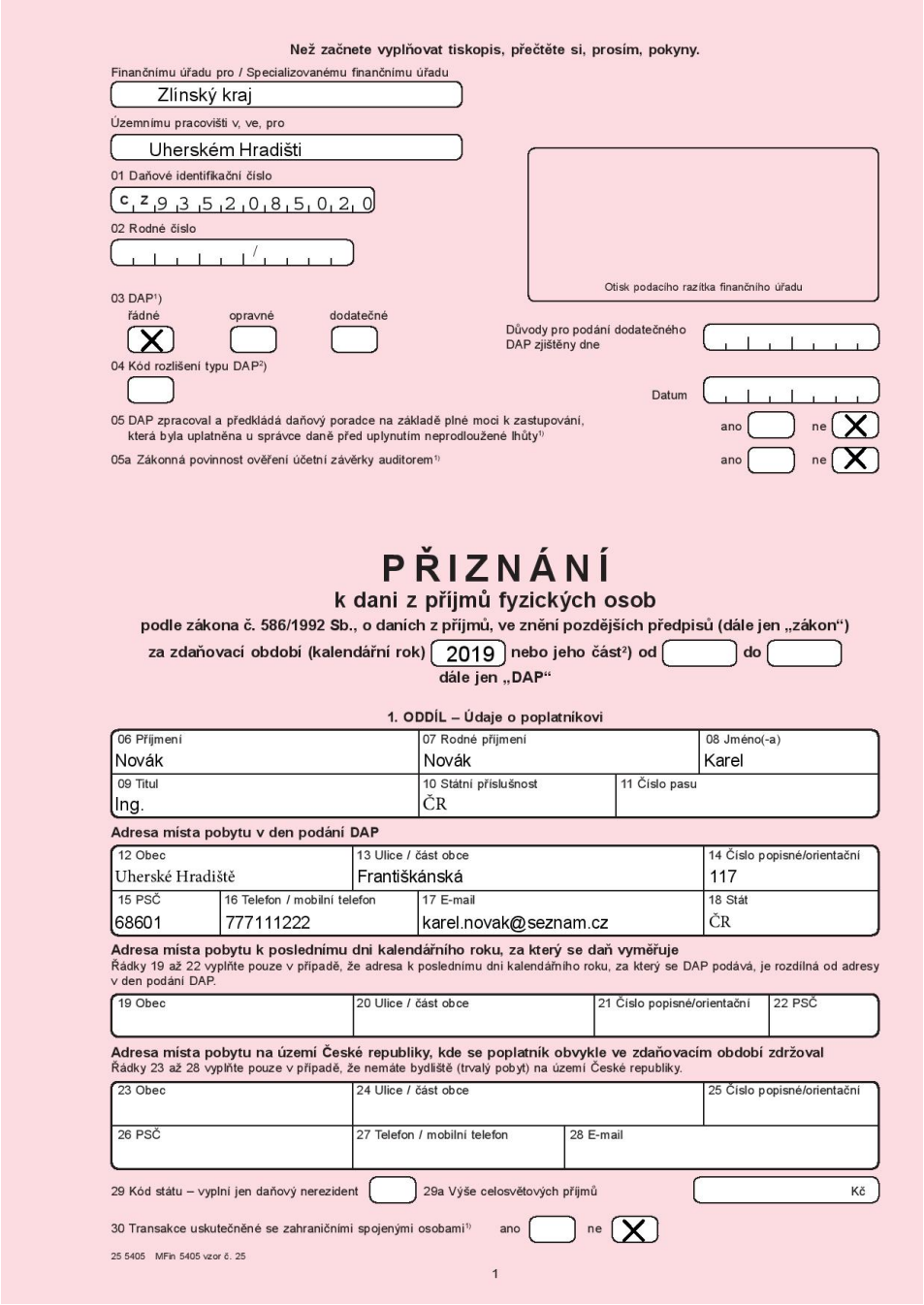
Dokumenty pro vytěžování můžeme rozdělit do tří základních pilířů

a) Strukturované – příkladem je formulář, složenka…
b) Polo-strukturované – například faktura (část je strukturovaná, část nikoliv)
c) Nestrukturované – nejlepší ukázkou je dopis

3.1.3.1.Strukturované dokumenty

Obecně lze říci, že tyto dokumenty mají body zájmu (metadata) na předem známých a konkrétních místech. (2) (3) Jedná se o nejjednodušší typ dokumentu pro automatické zpracování, kde se pravidla zpracování dá plně nastavit, aby korespondovalo s daným dokumentem.

Nejlepšími příklady se zdají být formuláře – například daňové přiznání, složenky...Tyto dokumenty jsou pro vytěžování dat velmi vhodné, jejich zpracování do workflow a případně provozu může být velmi rychlé.



Než začnete vyplňovat tiskopis, přečtěte si, prosím, pokyny.

Finančnímu úřadu pro / Specializovanému finančnímu úřadu: Zlínský kraj

Územnímu pracovišti v, ve, pro: Uherském Hradišti

01 Daňové identifikační číslo: CZ9352085020

02 Rodné číslo:

Otisk podacího razítka finančního úřadu

03 DAP[1]) řádné [X] opravné [ ] dodatečné [ ]

Důvody pro podání dodatečného DAP zjištěny dne

04 Kód rozlišení typu DAP[2])

Datum

05 DAP zpracoval a předkládá daňový poradce na základě plné moci k zastupování, která byla uplatněna u správce daně před uplynutím neprodloužené lhůty[1]) ano [ ] ne [X]

05a Zákonná povinnost ověření účetní závěrky auditorem[1]) ano [ ] ne [X]

PŘIZNÁNÍ

k dani z příjmů fyzických osob

podle zákona č. 586/1992 Sb., o daních z příjmů, ve znění pozdějších předpisů (dále jen „zákon")

za zdaňovací období (kalendářní rok) 2019 nebo jeho část[2]) od [ ] do [ ]

dále jen „DAP"

1. ODDÍL – Údaje o poplatníkovi

06 Příjmení	07 Rodné příjmení	08 Jméno(-a)
Novák	Novák	Karel
09 Titul	**10 Státní příslušnost**	**11 Číslo pasu**
Ing.	ČR	

Adresa místa pobytu v den podání DAP

12 Obec	13 Ulice / část obce	14 Číslo popisné/orientační
Uherské Hradiště	Františkánská	117
15 PSČ	**16 Telefon / mobilní telefon**	**17 E-mail** / **18 Stát**
68601	777111222	karel.novak@seznam.cz / ČR

Adresa místa pobytu k poslednímu dni kalendářního roku, za který se daň vyměřuje

Řádky 19 až 22 vyplňte pouze v případě, že adresa k poslednímu dni kalendářního roku, za který se DAP podává, je rozdílná od adresy v den podání DAP.

19 Obec	20 Ulice / část obce	21 Číslo popisné/orientační	22 PSČ

Adresa místa pobytu na území České republiky, kde se poplatník obvykle ve zdaňovacím období zdržoval

Řádky 23 až 28 vyplňte pouze v případě, že nemáte bydliště (trvalý pobyt) na území České republiky.

23 Obec	24 Ulice / část obce	25 Číslo popisné/orientační
26 PSČ	**27 Telefon / mobilní telefon**	**28 E-mail**

29 Kód státu – vyplní jen daňový nerezident [ ] 29a Výše celosvětových příjmů [ ] Kč

30 Transakce uskutečněné se zahraničními spojenými osobami[1]) ano [ ] ne [X]

25 5405 MFin 5405 vzor č. 25

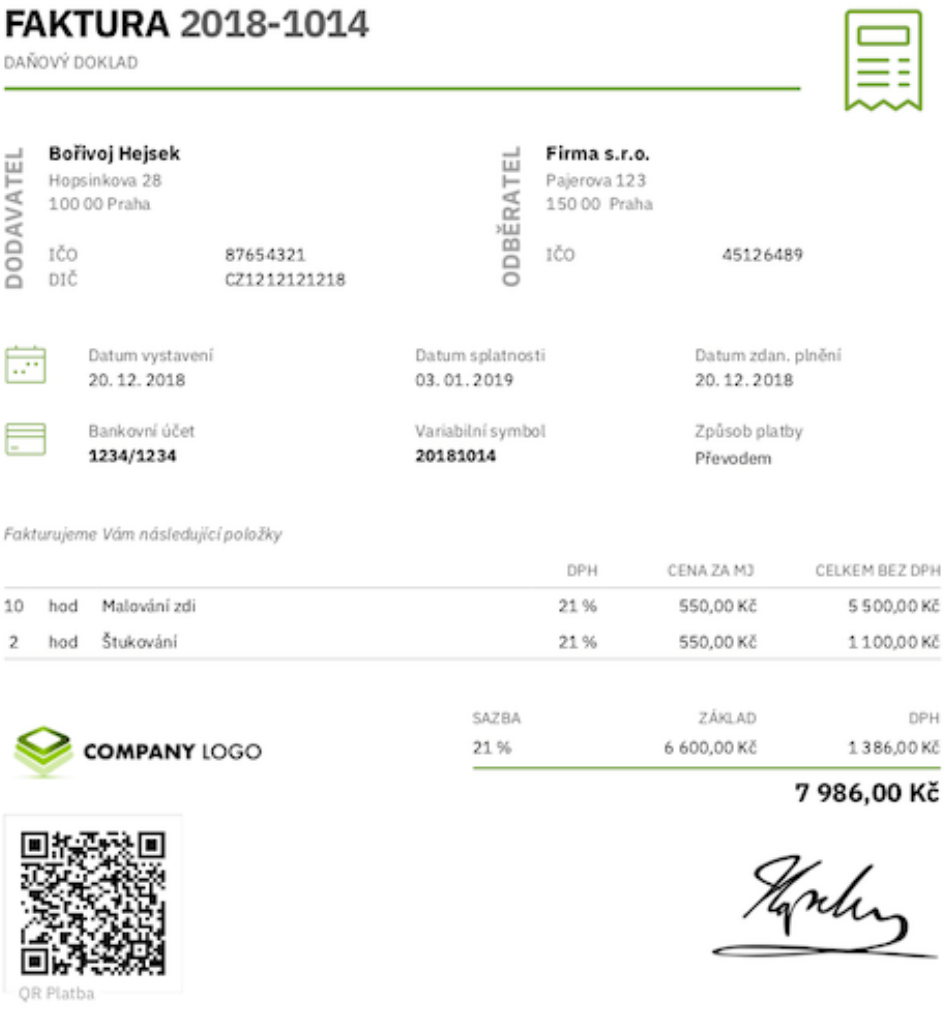
1

Obr. 1 Strukturovaný dokument (4)

3.1.3.2.Polostrukturované dokumenty

Polostrukturované dokumenty nemají vždy stejný layout či rozložení polí, ale nachází se na nich předem známá data a dodržuje se určitý formát. (5) Krásným příkladem

jsou například faktury. Hlavičkové údaje faktury lze považovat za strukturovaný obsah s různým výskytem, řádkové položky jako obsah nestrukturovaný (nikdy nevíme, kolik řádků bude faktura mít). (3)



FAKTURA 2018-1014

DAŇOVÝ DOKLAD

DODAVATEL

Bořivoj Hejsek
Hopsinkova 28
100 00 Praha

IČO 87654321
DIČ CZ1212121218

ODBĚRATEL

Firma s.r.o.
Pajerova 123
150 00 Praha

IČO 45126489

Datum vystavení 20. 12. 2018
Datum splatnosti 03. 01. 2019
Datum zdan. plnění 20. 12. 2018

Bankovní účet 1234/1234
Variabilní symbol 20181014
Způsob platby Převodem

Fakturujeme Vám následující položky

			DPH	CENA ZA MJ	CELKEM BEZ DPH
10	hod	Malování zdi	21 %	550,00 Kč	5 500,00 Kč
2	hod	Štukování	21 %	550,00 Kč	1 100,00 Kč

COMPANY LOGO

SAZBA	ZÁKLAD	DPH
21 %	6 600,00 Kč	1 386,00 Kč

7 986,00 Kč

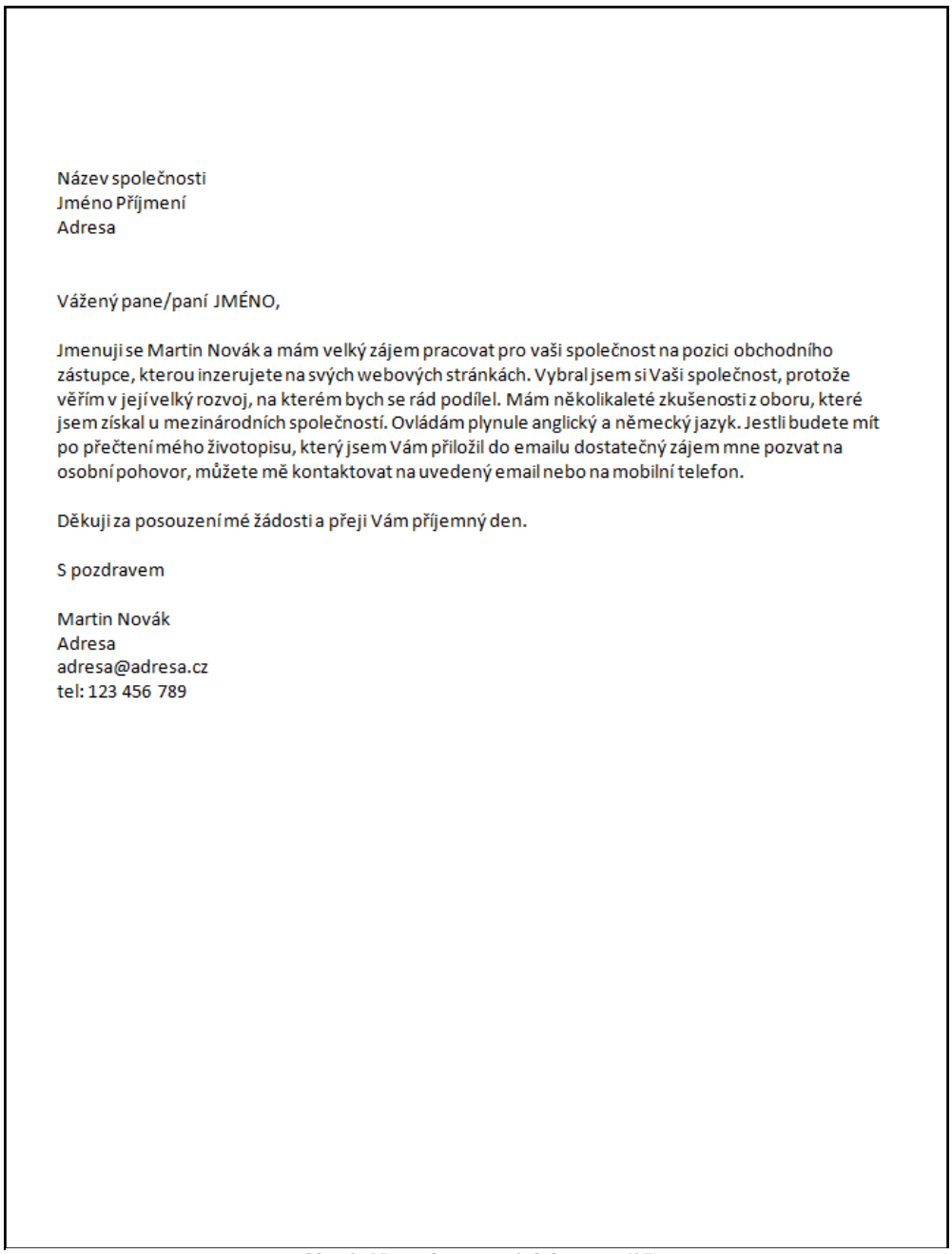
QR Platba

Obr. 2 Polostrukturovaný dokument (18)

3.1.3.3.Nestrukturované dokumenty

Tyto dokumenty jsou obecně nejsložitější a zároveň z pohledu obsahu nejzajímavější. Jejich metadata jsou na náhodných místech, často uprostřed prozaického textu či v blíže neznámém okolí, které se těžko identifikuje. Pro jejich vytěžování musí být použita množina různých technik (například i vyhledávání pomoci regulárních výrazů). (3)

Obecným příkladem nestrukturovaného dokumentu může být například dopis, stížnost či žádost.



Název společnosti
Jméno Příjmení
Adresa

Vážený pane/paní JMÉNO,

Jmenuji se Martin Novák a mám velký zájem pracovat pro vaši společnost na pozici obchodního zástupce, kterou inzerujete na svých webových stránkách. Vybral jsem si Vaši společnost, protože věřím v její velký rozvoj, na kterém bych se rád podílel. Mám několikaleté zkušenosti z oboru, které jsem získal u mezinárodních společností. Ovládám plynule anglický a německý jazyk. Jestli budete mít po přečtení mého životopisu, který jsem Vám přiložil do emailu dostatečný zájem mne pozvat na osobní pohovor, můžete mě kontaktovat na uvedený email nebo na mobilní telefon.

Děkuji za posouzení mé žádosti a přeji Vám příjemný den.

S pozdravem

Martin Novák
Adresa
adresa@adresa.cz
tel: 123 456 789

Obr. 3 Nestrukturovaný dokument (17)

3.1.4. Použití vytěžování v praxi

Automatické vytěžování údajů se v praxi používá primárně pro zrychlení procesu (1), odbourává se tím například nutnost dané údaje vyhledávat a přepisovat manuálně. Je

tudíž i potřeba méně lidských zdrojů a správnou implementací lze dosáhnout nízké chybovosti. Lze zavést i kontrolní mechanismy dat, na které je zákazník upozorněn a tudíž ví, že daný údaj nesplňuje například jeho kritéria.

Častým usecase je vytěžování faktur, formulářů (1) či na základě rozpoznaného textu i odhad sentimentu dokumentu. Daná vytěžovaná metadata se často předávají do dalšího procesu zpracování, ať už je to v případě faktur systém účetní, či bezpečné úložiště podléhající standardům.

Za pomoci správně zvoleného přístupu při vývoji aplikace pro vytěžování a za použití vhodného softwaru pro daný usecase, je úspora nejen časová, ale i finanční (1), protože v odděleních nemusí cirkulovat větší množství papíru – obsah je zdigitalizován – a dokument je dostupný takřka stále, co to interní systémy daného zákazníka dovolí.

3.2. IBM Datacap

IBM Datacap je SW platforma, sdružující vícero aplikací vč. vývojářského studia, ve kterém se dají vytvářet vytěžovací aplikace. V této práci uvažujeme verzi IBM Datacap 9.1.7.

3.2.1. Stručná historie

Produkt (a vlastně celá společnost) IBM Datacap nepatřil vždy do "rodiny" IBM. Touto společností byl akvizován až v roce 2010 (6) (7), kdy proběhlo zakoupení celé společnosti Datacap, Inc. za blíže neurčenou sumu.

Historie vlastního vytěžovacího softwaru sahá až do roku 1989, kdy vznikl první předchůdce současné verze – Paper Keyboard (8). IBM si uvědomovala potřebu mít produkt, který by pomohl zákazníkům zpracovávat jejich listinné dokumenty, a proto rozšířila své portfolio o tyto služby. V roce 2010 akvizovala nejen Datacap, ale i další společnosti (7).

3.2.2. Současnost

V současné době je produkt Datacap stále vyvíjen, vychází pro něj pravidelné opravné balíčky a aktualizace. Je součástí takzvané "data capture" fáze, která se při zpracování dokumentů, jak papírových, tak digitálních, nachází naprosto na začátku flow. Produkt je zatížen licencí, tj. bez zakoupené licence jej nelze využívat.

3.2.3. Struktura platformy IBM Datacap

Platforma IBM Datacap se primárně skládá z vícero komponent, částí a služeb, které jsou využívány dle charakteru řešení, prostředí a případně dle zakoupené licence.
Důležité části Datacapu pro tuto práci jsou komponenty pro vývoj:

1. Datacap Studio – vývojové studio (prostředí)
2. Datacap Server – služba, bez které nefunguje komunikace mezi databázemi aplikace a definičními soubory

A prezentační vrstva, ve které si lze prohlédnout výsledek vytěžení dokumentu z pohledu zákazníka, a to při verifikaci dokumentu:

1. Tenký klient – IBM Content Navigator
2. Tlustý klient – Datacap Desktop

3.2.4. Aplikace

Samotná aplikace běžící na této platformě se dá rozdělit na několik jednotlivých úrovní detailnosti, které popisují její funkčnost. Nicméně reálné provedení (a jeho pořadí v sekvenci) popsaných vlastností probíhá ještě v závislosti na struktuře DCO (popsáno níže)

Tyto úrovně jsou (9):

a. Workflow
b. Task – dílčí úloha
c. Ruleset – set pravidel
d. Rule – pravidlo
e. Function – funkce
f. Action – akce

3.2.4.1.Workflow

Prozaicky řečeno – workflow je složeno z jednotlivých Tasků, tasky jsou složené ze za sebou jdoucích rulesetů a rulesety se skládají z jednoho až n rules. Rule obsahuje minimálně jednu funkci (opět jich může být až x) a funkce obsahují také minimálně jednu akci, maximum není stanoveno.

3.2.4.2.Task – dílčí úloha

Primárně ohraničuje jeden krok ve worfklow. Je to spíše sémantické rozdělení, není nikde stanoveno kolik dílčích úloh ve workflow musí být. Na úrovni dílčích úloh se ale rozděluje, zdali je úloha automatická (zpracovává ji stroj), nebo manuální (zpracovává ji člověk).

Klasický příklad dílčích úloh v Datacapu a jeho aplikaci je VScan, který je používán pro import obrazů či Verify, používané pro manuální opravu dat operátorem. Dílčí úlohy

mohou být pojmenovány jakkoliv, doporučuje se ale, aby název vypovídal o charakteru úlohy.

Rulesety jsou v tasku seřazeny se závislosti na pořadí. První v pořadí je první vykonán.

3.2.4.3.Ruleset – set pravidel

Set pravidel je jedna jednotka v dílčí úloze. Dílčí úloha musí mít minimálně jeden ruleset, obvykle jich ale má vícero. Pojmenování rulesetů (kromě OOTB rulesetů) je volné a závisí na vývojáři aplikace. Doporučuje se jej ale pojmenovat tak, aby bylo zřejmé, co ruleset dělá. Ruleset může být přiřazen vícero dílčím úlohám. (9)

Pravidla v rulesetech mohou být jakkoliv seřazena, nezáleží na pořadí. Pravidla z konkrétního rulesetu jsou spouštěna dle hierarchie, tudíž první je spuštěno pravidlo na pozici batch open, poté na pozici open první dokument, open první strana…

Jedno pravidlo může být použito na n polích, ale na jednom poli může být použito maximálně právě jedno pravidlo z daného rulesetu.

3.2.4.4.Rule – pravidlo

Pravidlo je logická jednotka, která obsahuje minimálně jednu, většinou však vícero funkcí. Právě tato jednotka je přiřazena k DCO hierarchii a pojí tím dohromady umístění ve workflow s umístněním v hierarchii.

3.2.4.5.Function – funkce

Záleží na pořadí funkcí v pravidle. Funkce vrací návratovou hodnotu true či false. Dokud akce v rámci funkce vrací true, tak se pokračuje ve zpracování dané funkce. Pokud všechny akce v rámci funkce vrátí true, vrací true i funkce a zpracování pravidla je dokončeno. Pokud vrátí akce ve funkci false, přerušuje se zpracování dané funkce nehledě na to, jestli se jedná o první, poslední či jakoukoliv akci ve funkci a zahajuje se na poli zpracování funkce druhé v pořadí. Pokud není další funkce k dispozici, pravidlo se ukončí.

Důležité je zde zmínit, že pokud v rámci jakékoliv akce, resp. funkci dojde k úpravě např. dat v poli a až pak následně další akce skončí false, **nejsou** data vrácena do původní

podoby. S tím je nutné počítat při návrhu algoritmu pravidla. (9)

3.2.4.6.Action – akce

Akce jsou stavební bloky jednotlivých funkcí, lze je přirovnat k jednotlivým funkcím v objektovém programování obecně. Jejich složitost se liší, mohou být ekvivalentem k jednomu řádku kódu, i ke stovkám.

Akcím lze dle jejich charakteru přiřadit jak vstupní, tak výstupní parametry. Pokud chceme rozšířit funkcionalitu aplikace o novou vlastnost, doprogramováváme tento typ komponenty, nicméně je nutné řídit se náležitostmi, které se liší dle verzí systému.

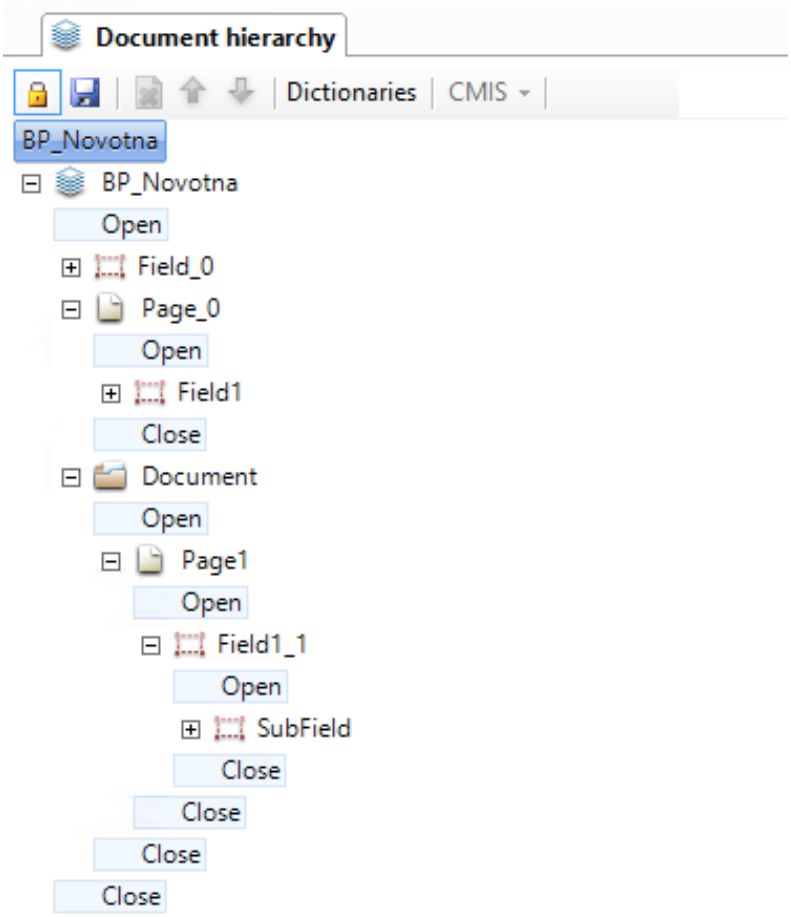
V rámci jednotlivých druhů licencí Datacapu jsou ve vývojovém studiu přístupné knihovny akcí, které představují předem použitelné bloky pro vývoj aplikace. Tyto akce jsou psané buďto v C# či historicky dříve ve VBScript. Autorka práce použije C#.

3.2.5. Struktura DCO (10)

V originále *hierarchy of DCO* (Datacap Application Objects). Definuje strukturu objektů v aplikaci.

Druhy objektů jsou:

a. Batch – dávka
b. Document – dokument
c. Page – strana
d. Field – pole



Obr. 4 DCO Hierarchie (zdroj: autor)

3.2.5.1.Batch – dávka

Nejvyšším bodem této hierarchie je dávka (Batch). Je nejvyšším uzlem, kořenem datového stromu. Objekt dávky je jediný bez rodiče a v aplikaci je pouze jeden.

Dávka může mít jako potomky objektové typy Dokument, Strana, Pole, a to v množství 0 až n.

3.2.5.2.Document – dokument

Dokument se může skládat ze Stran(y) a polí. Ojediněle téže z dokumentů, nicméně v praxi se toto nastavení moc nepoužívá.

3.2.5.3.Page – strana

Strana má jako povolené potomky Strany a pole. Fakticky se chová jako strana listinného dokumentu.

3.2.5.4.Field – pole

Pole může mít za potomky pouze pole. Využívá se například v jednom typu vytěžování checkboxů, anebo ve vytěžování řádkových položek.
Jedno pole lze použít u vícero druhů stran, nicméně jakmile je u jakékoliv strany změněno, změní se ve všech výskytech. Proto je leckdy nutností dodržovat unikátnost názvů polí.

3.2.6. Smart Parametry

Samostatnou malou, ale neméně důležitou kapitolou při propojování funkcí a DCO jsou tzv. Smart parametry.

Smart parametry jsou vlastně výrazy, které vrací zpět buď hodnotu, nebo daný objekt. (11) Využívají se primárně k tomu, aby nemusela být uvedena doslovná hodnota, kterou programátor v rámci implementace ani nemusí znát. Smart parametr je uvozen symbolem @. Existují různé typy smart parametrů, jako obecný Smart Parametr lze uvést například parametr @ID, který vrací ID objektu, na kterém je spuštěn, tento parametr je tzv. „runtime“ - běhový. Jiný smart parametr - @APPPATH(VAR) je konfigurační a získává cestu.

3.2.7. Vývoj aplikace

Vývoj jednotlivé aplikace by bylo záhodno rozdělit do vícero kroků:

1) Analýza zadání
2) Návrh aplikace
3) Algoritmizace jednotlivých úrovní
4) Naprogramování
5) Test

3.2.8. Doprogramování funkcionalit, resp. komponent, resp. custom akcí

Obecně pro vývoj aplikace v produktu IBM Datacap není nutná aktivní znalost programování v žádném jazyce. V každém případě je však možné znalost objektového programování považovat za výhodu. Stejně důležitá je pro návrh aplikace i analytická znalost či znalost algoritmizace.

Nicméně samotná znalost C# či VBScriptu není pro programování custom akcí dostačující. Neméně důležitá je také znalost toho, jak vůbec aplikace jako taková funguje, znalost DCO hierarchie, jednotlivých vlastností objektů a případná reference již existujících knihoven, aby nedošlo ke zbytečné duplicitě kódu. (2)

Custom akce se do platformy přidávají ve tvaru .dll a to do RRS složky Datacapu, která tyto knihovny obsahuje.

3.3. Regulární výrazy

Jak již bylo zmíněno, vytěžování dat lze (alespoň v rámci IBM Datacap) naimplementovat několika různými způsoby.

Ten zřejmě nejčastější v rámci nestrukturovaného obsahu je vyhledávání za pomoci regulárních výrazů.

Níže bude popsáno, co to vlastně regulární výraz je, jak se skládá a jak jej lze v rámci vytěžování v IBM Datacap využít.

3.3.1. Co je to regulární výraz

Regulární výraz, alias v angličtině Regular expression či zkráceně RegEx, je stručně řečeno vzor. Vzor, který je použit pro porovnání se vstupním zpracovávaným textem (12), v případě vytěžovací aplikace s textem, který se nachází na dokumentu.

Regulární výrazy lze samozřejmě implementovat i v klasickém programování, a to v různých jazycích, nicméně v této práci jej uvažujeme primárně za účelem vytěžování dat v IBM Datacap a pro vytvoření custom akce, která je předmětem této práce, a to v jazyce C#.

3.3.2. Tabulka substitucí

Znak	Význam	Příklad
*	Nula až více předchozího znaku	Ah* porovnává Ahhh či A
?	Nula až jeden předchozího	Ah? Porovnává Ah či A
+	Jeden či více předchozího	Ah porovnává Ah nebo Ahhh, ale ne A
\	Použití pro „escape“ speciálního znaku	Hungry\? Porovnává řetězec Hungry?
.	Jakýkoliv jeden znak	Do. Porovná Dog,Dot..
()	Závorky – sjednocení	Použití jako v matematice
[]	Porovná jakýkoliv z vypsaných znaků	[cbf]ar porovná car,far nebo bar
|	OR	Pondělí|Úterý – porovná jedno z nich
{}	Porovná uvedený počet výskytů v závorce předchozího znaku	[0-9]{3} porovná 315, ale ne 31
^	Začátek řetězce	
$	Konec řetezce	

Tabulka 1 Regulární substituční tabulka (13)

3.3.3. Regulární výrazy v Datacapu

3.3.3.1. Vytěžování dle regulárních výrazů

Vytěžování s pomocí regulárních výrazů lze v rámci Datacapu rozdělit na dvě různé kategorie, a to:

1. Kvantitové vytěžování
2. Typové vytěžování

1. Kvantitové vytěžování

Při vývoji aplikace se lze rozhodnout, zdali u daného pole programátor použije akce podporující pouze jeden regulární výraz či slovník regulárních výrazů. V případě slovníku regulárních výrazů bude dokument prohledáván postupně prvním až posledním výrazem, shora dolů, dokud nevrátí shodu. V tom případě akce končí a dále neprohledává. Pokud shoda s výrazem není nalezena, vrací akce false. Totéž i v případě použití pouze jednoho regulárního výrazu.

2. Typové vytěžování

Toto vytěžování rozdělujeme na dva způsoby. První je, že vytěžujeme tu danou konkrétní hodnotu, která nás zajímá. Například SPZ – tam lze předem říci, že dokážeme vytvořit regulární výraz takových vlastností, že pravděpodobnost shody s jiným řetězcem než SPZ na dokumentu je spíše nulová.

Druhým způsobem je to, že si pomocí regulárního výrazu můžeme najít oblast, kde se metadata, která nás zajímají, pravděpodobně vyskytují. Příklad lze uvést třeba následující – Příjmení: Po tomto řetězci zpravidla bude směrem vpravo následovat řetězec obsahující příjmení. Použití regulárního výrazu nám pomůže eliminovat případné chyby OCR, náhrady znaků a jiné.

3.3.3.2.Validace dat regulárním výrazem

Regulární výraz lze kromě použití ve vytěžování využít v Datacapu i k ověření validity dat. Opět můžeme použít příklad z praxe – lze ověřit validitu datumu, jestli dané datum odpovídá a hlavně jestli existuje, či jestli je řetězec dostatečně dlouhý, a jiné…

Obr. 5 Vzorový regulární výraz (zdroj: autor)

3.4. C# – programovací jazyk

„C# (vyslovit "viz Sharp") je moderní, objektově orientovaný programovací jazyk, který je typově bezpečný. Jazyk C# má své kořeny v řadě jazyků C a bude okamžitě známý programátorům jazyka C, C++, Java a JavaScriptu.

C# je objektově orientovaný programovací jazyk ***orientovaný na součásti****. Jazyk C# poskytuje jazykové konstrukce pro přímou podporu těchto konceptů, což C# přirozený jazyk pro vytváření a používání softwarových komponent. Vzhledem k tomu, že v jazyce C# byly přidány funkce pro podporu nových úloh a vznikajících postupů pro návrh softwaru.“* (14)

3.4.1. C# v kombinaci s Datacapem

Programovací jazyk C# je používaným nástrojem pro doprogramování funkcionalit v Datacapu.

Pro implementaci je použit software Microsoft Visual Studio 2019 a pro založení projektu IBM nabízí využití template ve dvou různých verzích. Lze stáhnout buď na webu IBM nebo na verzovacím nástroji GitHub.

První verze se tvoří s odděleným RRX souborem, který je nositelem informací pro Datacap – obsahuje nápovědu k akci, příklad použití, a pak samotné DLL. Dle autorky má tato verze nevýhodu vícero souborů, a navíc musí být dané DLL registrováno.

Druhou verzí je verze All-in-one, znamená to, že RRX již je „zabudováno“ v rámci samotného DLL, DLL nemusí mít registrováno a nositelem všech informací je pouze DLL. Dle autorky má toto řešení výhodu, že se člověk stará pouze o jeden soubor a neřeší registraci v systému.

V rámci tohoto řešení je pro založení projektu, a tudíž pro vytvoření custom akce, použita verze All-in-one.

4. Vlastní práce

4.1. Problematika vytěžování regulárních výrazů IBM Datacap

V době, kdy autorka psala tuto práci se v rámci softwaru IBM Datacap nenacházela funkcionalita, resp. akce, která by dovolovala naráz v jednom kroku vytěžit a uložit veškeré výskyty slov (textu), vyhovující zadanému listu regulárních výrazů. Neexistovala ani pro jeden konkrétní regulární výraz.

V rámci autorčiny pracovní praxe se ukázalo, že tato funkcionalita je v určitých usecasech velmi potřebná a ušetřila by jak čas při zpracování jednotlivých dávek, tak čas při vývoji aplikace a při snaze obejít tuto chybějící akci.

4.1.1. Popis problému? Z čeho vychází práce?

Naprosto prvotní myšlenka na doimplementování této funkcionality autorku napadla při vývoji aplikace, která měla ze stránky dokumentu vytěžit a uložit veškerá rodná čísla, která se na dané straně nacházela. Při vstupu dokumentu nebylo jasné, kolik a jestli vůbec, se rodná čísla na straně nachází.

Dokumenty byly primárně nestrukturované, různých druhů vzhledů, neměly většinou ani společného vydavatele.

Dokumenty byly veskrze odlišné.

Pokud by byl tento úkol těžen běžným způsobem, musel by se zvolit konečný počet polí, které by se na dané straně nacházely, a po jednom je plnit v těchto krocích:

1. Najít vyhovující text regulárního výrazu
2. Uložit
3. Porovnat, jestli již náhodou hodnota není existující např. v předchozím poli
4. Opakovat, dokud nebude dosaženo konečný počet polí.

Nevýhody jsou zřejmé skoro na první pohled – vždy by se kontroloval tento počet polí – například i kdyby byl na stránce jen jeden výskyt, vždy by se jich pokusilo hledat n, dále by nastal i opačný problém – co kdyby těchto výskytů bylo více než je nadefinované n? Potom by tato data byla ignorována a zákazník by je při předávce výstupu neobdržel.

Nehledě na to, že by toto prohledávání zabralo výrazně více času než jednotné načtení, což by se mohlo ukázati kritické při větším objemu dat.

4.1.2. Analýza řešení

Autorka si dala za cíl analyzovat a vymyslet řešení, které by dovolilo ošetřit tyto problémy:

a. Vznikla by jednotná a přehledná struktura vytěžených dat
b. Bylo by možné při vyhledání použít více než jeden regulární výraz
c. Byla by ošetřena případná duplicita dat na stejné pozici
d. Byl by povolen duplicitní výskyt stejného textu na různých pozicích na straně
e. Vše by bylo přehledně zdokumentováno v provozním logu dílčí úlohy
f. Veškeré vstupní parametry by podporovaly smart parametry a byly modifikovatelné

4.1.2.1.Návrh řešení

Hlavním nositelem informace a úrovní, na které se bude akce spouštět je úroveň DCO pole. Pole bylo zvoleno primárně proto, že v rámci využití technik, které se používají při těžbě nestrukturovaného obsahu, se používají techniky právě volané na poli. Proto autorka chtěla být konzistentní. Dále je důvodem také zobrazení v prezentační vrstvě.

Pro samotnou implementaci bude nutné téže načíst soubor KEY obsahující regulární výrazy a načíst cestu ke složce, kde se nachází definiční soubor, který obsahuje rozpoznaný text na straně. Pro načtení do pole, kde jeden řádek ve vstupním souboru se rovná jedné položce v poli, lze použít třídu Systém.IO.File a metodu ReadAllLines.

Tento soubor vzniká po celostránkovém rozpoznání OCR enginem a po normalizaci strany. Jedná se o XML soubor s defaultním pojmenováním ve tvaru (ID strany)c.xml. V rámci tohoto XML lze najít strukturu slov, linek a pozic, ze kterých autorka chce vycházet při práci s textem. Pro načtení bude nutné vyřešit podmínku, aby se načetly právě nody typu w – slovo. Tuto schopnost třída Systém.Xml.XmlNodeList se svou metodou SelectNodes podporuje.

```xml
<?xml version="1.0" encoding="utf-16"?>
<CCO path="c:\datacap\nazev_aplikace\batches\20201102.000000\tm000001.cco">
  <L l="1782" t="224" b="255" r="2197" i="1">
    <W l="1782" t="224" r="1921" b="255" x="1" y="1">Text</W>
    <W l="1937" t="224" r="1999" b="255" x="2" y="1">Vzorovy</W>
    <W l="2038" t="224" r="2197" b="255" x="3" y="1">Dokument</W>
  </L>
```

Obr. 6 XML c.xml (zdroj: autor)

Jelikož pro založení projektu používáme šablonu od IBM právě pro vývoj custom akcí, tak jsou v projektu již potřebné reference přidány defaultně.

Nejméně časově náročným řešením z hlediska případných zbytečných průchodů textem a porovnání s již vytěženými vhodnými slovy by tudíž bylo použití dvou zanořených cyklů, kde obalová úroveň cyklu by procházela jednotlivé načtené nody.

Ve chvíli, co by byla splněna podmínka shody s regulárním výrazem, by tato daná hodnota byla uložena do vytvořené struktury pole Rodičx/Potomekx, kde jak pro rodiče tak potomka by byla uložena pozice, která se využívá v rámci prezentační vrstvy, tak v případě potomka i hodnota, se kterou by se dále v aplikaci pak mohlo pracovat.

Jak bylo zmíněno v teoretické části, pole musí mít unikátní název, jinak ukazují na stejné vlastnosti objektu, tudíž i zachování unikátnosti názvů musí být v rámci implementace vyřešeno.

4.1.2.2.Klíčové body

a. Správně načíst vstupní parametry, které se definují v Datacap Studiu
b. Podpora SMART parametrů
c. Ošetřit, jestli existují nutné soubory
d. Načíst soubory
e. Najít slova vyhovující regulárnímu výrazu
f. Ošetření duplicitního porovnávání u již vyhledaného slova
g. Uložit tato slova do nových polí vč. textu a pozice
h. Zachovat unikátnost názvu polí
i. Vodicí texty v logu dílčí úlohy
j. Návrat výsledku akce

Použité techniky jsou více popsány v kapitole 4.1.4.

4.1.3. Prerekvizity pro implementaci

4.1.3.1.Systémové prerekvizity

i) Microsoft Visual Studio 2019
ii) IBM Datacap 9.1.7 nainstalovaný na stejném prostředí jako Visual Studio
iii) Aplikace pro test s celostránkovým rozpoznáním
iv) KEY soubor – textový soubor obsahující množinu regulárních výrazů

4.1.3.2.Znalostní prerekvizity

Pro možnost implementace vlastní akce musí mít programátor pochopení minimálně základu teorie vývoje aplikací v Datacapu obecně, pochopení funkčnosti hierarchie a workflow. V rámci implementace akce lze využívat také DCO API Datacapu, které umožňuje práci s objekty. Další nutnou znalostí je znalost C#, který se pro vytvoření akcí používá.

4.1.4. Použité techniky

Níže jsou rozebrané použité jednotlivé nejdůležitější techniky pro implementaci řešení. Jako ukázky jsou vybrány části zdrojového kódu z akce zpracovávané pro tuto BP.

4.1.4.1.Cyklus foreach

„Cyklus foreach funguje ve spolupráci s datovým typem pole. Tento cyklus funguje tak, že postupně projde jednotlivé prvky pole a umožní s nimi pracovat v těle cyklu. Důležité je, že cyklus projde vždy všechny prvky pole a umožní s každým jednotlivým prvkem pracovat v těle cyklu. Ukažme si syntaxi tohoto cyklu.

```
foreach (promennaProJedenPrvek in poleKtereProchazime)
{
// Příkazy v těle cyklu
}
```

Použijeme tedy klíčové slovo foreach. V závorce následuje definice proměnné, která bude v každém běhu cyklu obsahovat právě jeden prvek z pole. Za touto proměnnou následuje klíčové slovo in a za ním proměnná obsahující pole, jehož prvky chceme procházet.“ (15)

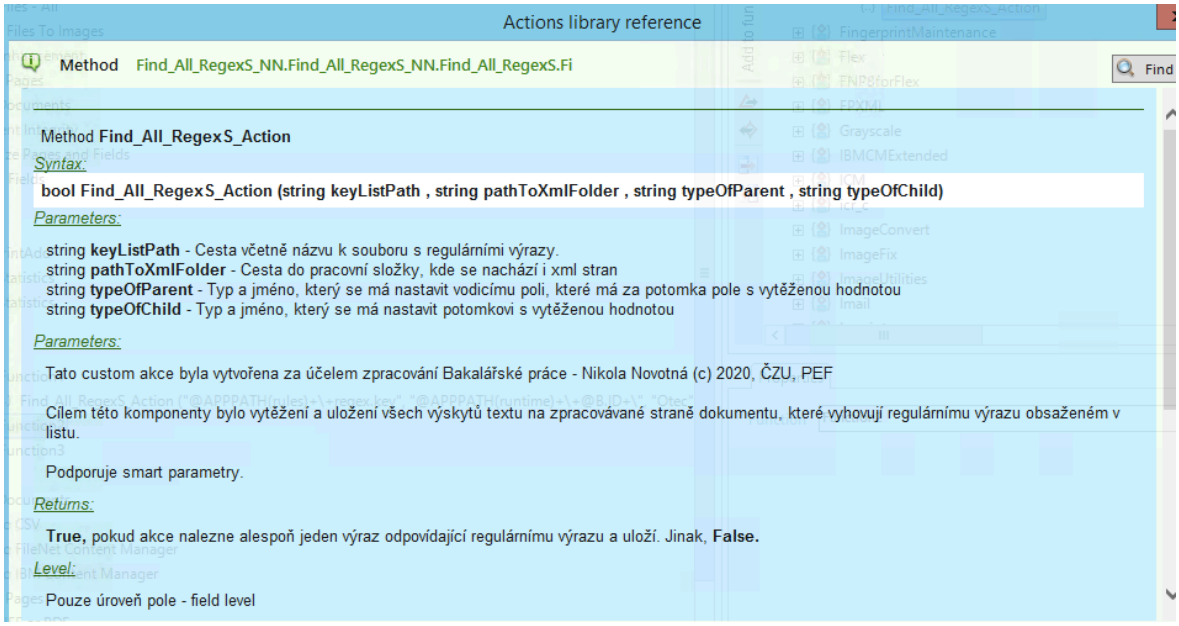
4.1.4.2.Načtení parametrů

Definice pro načtení parametrů ve Studiu se nachází v souboru TheRRX.rrx, které má strukturu XML.

Právě zde se definuje typ parametrů akce a také dokumentace celé akce, která je ze studia přístupná.

```
4142
<method name="Find_All_RegexS_Action">
    <p name="keyListPath" type="string" qi="Cesta včetně názvu k souboru s regulárními výrazy."/>
    <p name="pathToXmlFolder" type="string" qi="Cesta do pracovní složky, kde se nachází i xml stran"/>
    <p name="typeOfParent" type="string" qi="Typ a jméno, který se má nastavit vodicímu poli, které má za potomka pole s vytěženou hodnotou"/>
    <p name="typeOfChild" type="string" qi="Typ a jméno, který se má nastavit potomkovi s vytěženou hodnotou"/>
    <ap>
```

Náhled dokumentace, jak vypadá v Datacap Studiu pro tuto implementovanou akci.



Obr. 7 Okno s dokumentací (zdroj: autor)

4.1.4.3.Podpora SMART parametrů

Používání smart parametrů může velmi zjednodušit migraci aplikací na prostředí zákazníka či produkci, proto i v rámci této akce bude autorka používat techniky pro podporu smart parametrů.

Za pomoci smart objektu a metody MetaWord je vstupní hodnota převedena. Pokud byla hodnota smart parametrem, je uložena návratová hodnota smart parametru. V jiném případě je zachována původní hodnota string.

```
4143

// Pokud chceme použít jako vstup smart parametr, musí být typu string.

// Převod a uložení hodnot vytažených ze smart parametru pomocí smart
objektu.
//Pokud daná vstupní hodnota nebyla smartparametrem, je vrácen původní
string
localSmartObj = new dcSmart.SmartNav(this);

//proměnná obsahující cestu ke klíči vč. klíče
string uncodedKeyListPath = localSmartObj.MetaWord(keyListPath);
//proměnná obsahující cestu do runtime složky dávky, kde se vyskytují
definiční soubory dávky vč. xml
string uncodedPathToXmlFolder = localSmartObj.MetaWord(pathToXmlFolder);
//název typu, který má být přiřazen vodícímu poli v Datacapu
string uncodedTypeOfParent = localSmartObj.MetaWord(typeOfParent);
//název typu, který má být přiřazen potomkovi vodícího pole v Datacapu
string uncodedTypeOfChild = localSmartObj.MetaWord(typeOfChild);
```

4.1.4.4.Ošetření existence souborů

Ošetření existence souborů autorka provádí ze začátku akce z toho důvodu, že pokud by alespoň jeden ze souborů nebyl k dispozici, tak nelze akci provést a je rovnou vrácena návratová hodnota false, která přerušuje nejen zpracování akce, ale i funkce jako takové.

```
4144

// Zjišťujeme, jestli pole má nějakého předchůdce.
TDCOLib.IDCO objParentPage = CurrentDCO.Parent();
if (objParentPage == null)
{
WriteLog("Neexistuje předchůdce");
return false;
}

// Jelikož pole nemusí mít za svého předchůdce stranu, na které je však
provedeno celostránkové rozpoznání textu, zjišťujeme, jestli parent
objekt je typu strana / Page /
if (!(objParentPage.ObjectType() == Level.Page))
{
WriteLog("Předchůdce není typu strana");
return false;
}
WriteLog("Rodič v podobě stránky existuje.");

// Jednotlivý vytěžený celostránkový text můžeme nalézt v definičním
souboru strany, označeného jmennou konvencí @P.ID+c.xml
string pathToXmlC =
Path.Combine(uncodedPathToXmlFolder,objParentPage.ID+"c.xml");
// Zjišťujeme, jestli tento soubor pro danou vrácenou stranu existuje
if (!File.Exists(pathToXmlC))
{
WriteLog("Definiční stránka nebyla nalezena");
return false;
}
WriteLog($"Název rodiče je {objParentPage.ID}, tudíž path pro c.xml je
{pathToXmlC}");
```

```
// Zjišťování existence keylistu - souboru s výčtem regulárních výrazů
if (!File.Exists(uncodedKeyListPath))
{
WriteLog("List s regulárními výrazy nebyl nalezen");
return false;
}
WriteLog($"Path pro keylist  je {uncodedKeyListPath}");
```

4.1.4.5.Práce s KEY souborem

Key soubor je textový soubor, pro jehož načtení je použita třída System.IO.File a metoda ReadAllLines.Tato metoda „otevře textový soubor, přečte všechny řádky souboru do pole řetězců a pak soubor zavře“ (16).

Dále je pro jistotu kontrola na velikost pole, jestli náhodou není naprosto prázdné. V tomto případě by nemělo smysl pokračovat.

```
4145
//Jelikož máme potvrzenou existenci souboru, načítáme tento soubor do
pole stringů a to po jednotlivých řádcích
string[] regexList = File.ReadAllLines(uncodedKeyListPath);
WriteLog($"Keylist  je načten ");
// Byl v souboru alespoň jeden řádek - klíč ?
if (regexList.Length == 0)
{
WriteLog("List klíčů byl prázdný");
return false;
}
```

4.1.4.6.Práce s XML souborem

Práce s XML souborem je v rámci této akce rozdělena do dvou částí. V první za pomoci třídy System.Xml.XmlDocument a metody Load(parametr cesty k souboru) načteme XML jako celek.

Poté z XML „filtrujeme“ jenom uzly, které nás zajímají, tudíž vyhovují výrazu XPath v metodě SelectNodes z třídy System.Xml.XmlNodeList. Pro tento konkrétní typ xml chceme načíst pouze W elementy, které jsou potomkem CCOa poté potomkem L.

```
4146
//Načítáme XML pro zpracovávanou stránku
XmlDocument pageXml = new XmlDocument();
pageXml.Load(pathToXmlC);

// Dle struktury tohoto souboru nás zajímají nody pro slova, které se
nachází zanořené pod CCO/L/W
XmlNodeList nodesWords = pageXml.SelectNodes("/CCO/L/W");
```

4.1.4.7.Nalezení shody s regulárními výrazy

```
4147
// Načtení konkrétního stringu a vytvoření regulárního výrazu z daného
stringu, který je vstupním parametrem
Regex regex = new Regex(regexItem);
//Výpis konkrétního regulárního výrazu, který se zpracovává v dané
iteraci
WriteLog($"Regex - současný - {regexItem}");
// Podmínka, která je splněná, pokud text slova vyhovuje regulárnímu
výrazu
if (regex.Match(node.InnerText).Success)
{
....
}
```

4.1.4.8.Vytvoření pole

Pro vytvoření pole se používá metoda AddChild, která se volá na objektu, pod kterým daný potomek bude vytvořen. Tato metoda je obecná, dá se použít i pro vytvoření stran či dokumentů.

První parametr metody udává typ objektu – úroveň v DCO – v tuto chvíli parametr 3, což znamená pole.

Druhý parametr nastavuje objektovou proměnnou name (jméno) na danou hodnotu v parametru.

Třetí parametr udává pozici vytvořeného potomka. Může nabývat hodnoty 0, což znamená, že se pole dá na začátek dané struktury, nebo -1 což značí konec struktury. Zde autorka používá parametr -1.

```
4148
// Vytvoření vodicího pole – typu 3 (to znamená pole), druhý parametr je
jméno pole (musí být unikátní, proto přidáváme count),
//poslední parametr určuje lokaci, kam dané pole bude uloženo - 0 znamená
na začátek, -1 na konec.
TDCOLib.IDCO lineItem = CurrentDCO.AddChild(3, uncodedTypeOfParent+count,
-1);
```

4.1.4.9.Uložení hodnoty a pozice do pole

Uložení hodnoty a pozice jsou v rámci metod nad objekty dva různé na sobě nezávislé kroky.

Pro uložení pozice se využívá metoda SetPosition, jejíž parametrem jsou 4 parametry typu int, obsahující jednotlivé pozice pro Left, Top, Right a Bottom souřadnice.

Pro uložení hodnoty se používá vlastnost objektu .Text, do které se přiřazuje stringová hodnota, v tomto případě InnerText vyhovujícího node XML souboru.

```
/4149
// Ukládáme pozici nově vytvořenému poli.
item.SetPosition(leftPosition, topPosition, rightPosition,
bottomPosition);
// Dáváme typ.
item.Variable["TYPE"] = uncodedTypeOfChild;
// Status je opět PASS = 0
item.Status = 0;
// Změna – zde ukládáme vytěženou hodnotu, tudíž cílovou hodnotu, která
vyhověla regulárnímu výrazu
item.Text = node.InnerText;
```

4.1.4.10.Zápis do logu dílčí úlohy

Pro zápis do logu dílčí úlohy je použita metoda WriteLog(string message), která je předpřipravena v šabloně projektu.

4.1.4.11.Návrat hodnoty – výsledku – akce

Jako každá akce v Datacapu, tak i tato akce je s návratovou hodnotou true a nebo false. Dle této návratové hodnoty Datacap pozná, jak se má zachovat dále v implementovaném algoritmu funkcí, jestli má pokračovat dále v současné funkci, či zpracování přerušit a pokračovat s funkcí následující.

V rámci této akce je návratová hodnota true právě tehdy, pokud je uloženo alespoň jedno pole s vyhledaným slovem a pozicí, které odpovídalo regulárnímu výrazu.

Jinak vrací false. Tato skutečnost je popsaná i v dokumentaci akce, která je přístupná z vývojového studia Datacapu.

5. Výsledky a diskuse

5.1. Výsledek implementace

V rámci analýzy a následně pak během testů bylo ověřeno, že danou problematiku lze do IBM Datacapu za pomoci custom akce naprogramovat a vytěžené údaje následně dále zpracovávat v závislosti na charakteru aplikace, ve kterém je akce použita.

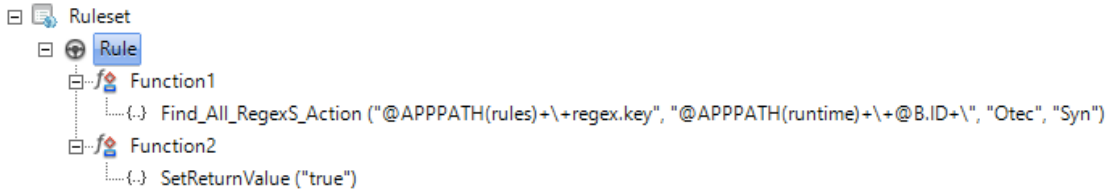
5.1.1. Testy v aplikaci

Testy probíhaly na prostředí serveru, který měl nainstalovanou verzi IBM Datacap 9.1.7 opravný balíček 003. Pro ověření funkčnosti byla použita aplikace, která splňovala prerekvizity pro použití akce (celostránkové rozpoznání s normalizací textu).

V rámci testů bylo ověřeno, že:

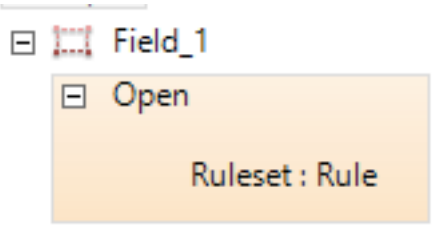
1. Datacap Studio, a tudíž i další komponenty tuto akci vidí a mohou ji použít
2. Lze zadat parametry akce, se kterými je dále pracováno, a to v plném rozsahu, tj:
    a. Keylist parametr
    b. Parametr složky obsahující definiční xml s vytěženým textem
    c. Parametr se jménem / typem vodicího pole
    d. Parametr se jménem / typem pole, do kterého je ukládána hodnota
3. Pozice polí je v rámci testovacích dávek viditelná a odpovídající poloze textu
4. Hodnota ze strany je do polí uložena přesně ve tvaru slova, které se nachází v definiční stránce.

5.1.1.1.Ruleset s implementovanou akcí



Obr. 8 Ukázka implementace (zdroj: autor)

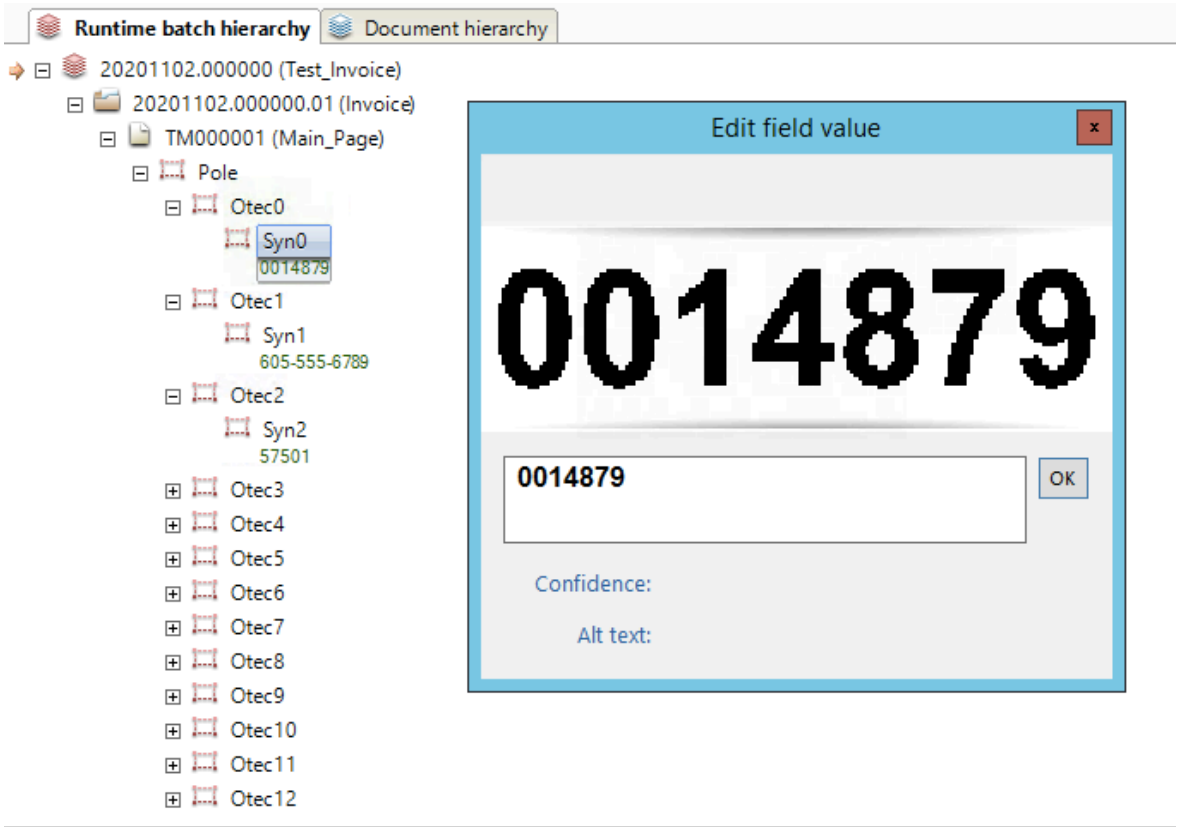
5.1.1.2.Přiřazení akce do DCO



Obr. 9 Přiřazení do hierarchie (zdroj: autor)

5.1.1.3.Výsledek akce nad rozpoznanou stranou

V obrázku níže lze vidět výstřižek (screenshot) z vývojového studia, kde je vidět vytvořená struktura akcí, vzorové pole s hodnotou a uloženou pozicí. Uloženou pozici potvrzuje i přítomnost výřezu z obrazu přesně v pozici, ve kterém se dané slovo původně nacházelo.



Obr. 10 Výsledek z testu (zdroj: autor)

5.2. Zhodnocení

Akce splnila očekávání výsledku, v rámci testovaných stran fiktivních faktur našla veškerá slova, která vyhovovala regulárním výrazům. Primární zaměření této akce je na vyhledání patternových slov. Díky práci s jednotlivými slovy je i ve studiu při simulaci rychlejší než při prohledávání prozaického textu za pomoci vestavěných funkcionalit Datacapu.

6. Závěr

V rámci této práce chtěla autorka naimplementovat výrazně chybějící funkcionalitu v softwaru IBM Datacap, která by umožnila vytěžování všech výskytů textu, který vyhovuje listu regulárních výrazů. Tato funkcionalita by výrazně zlepšila a urychlila vytěžování většího množství dat na zpracovávané straně. Primárním podnětem byl pro autorku usecase vytěžování blíže neurčitého množství rodných čísel na straně dokumentu.

Při analýze byly vydefinovány důležité klíčové body, které bylo nutno splnit a uvažovat o nich při implementaci. Pro implementaci samotnou byla zvolena metoda All-in-one DLL a programovací jazyk C#. Nápověda a veškeré doprovodné texty, které se zobrazují uživateli (programátorovi) ve Studiu byly implementovány v českém jazyce.

V rámci implementace aplikace v Datacapu nemůže programátor na první pohled odlišit akci naimplementovanou a otestovanou v rámci této práce jiným způsobem než dle jazyka použitého v nápovědě. Akce byla plně integrována, splňuje požadovanou funkcionalitu – při použití v aplikaci s celostránkovým rozpoznáním textu vytěží veškerá slova, která vyhovují zadaným regulárním výrazům. Následně tato extrahovaná metadata uloží do dynamicky se vytvářející struktury polí, která odpovídá množství vytěžených dat. Jména a typy této nově vytvořené struktury jsou plně v gesci programátora, který je specifikuje ve formě vstupních parametrů do akce.

V rámci testů bylo ověřeno, že akce plně vytěžuje text / slova na zpracovávané straně, korektně k nim ukládá jejich pozice, tudíž i obrazové náhledy nad stranou jsou plně funkčí v obou prezentačích vrstvách. Náhled lze vidět v kapitole 5.

V tuto chvíli je daná funkcionalita připravena v rozsahu takovém, že by ji bylo možno použít pro vývoj aplikace pro koncového zákazníka a pro další produkční provoz.

Autorku práce již nyní napadají další funkcionality, o které by tuto akci šlo rozšířit, a to například řešení výskytu na celých dokumentech zároveň, nikoli pouze na stranách či práce s vyhledáním volitelně jak na úrovni slov, tak třeba linek.

Naimplementovaná komponenta Find_All_RegexS by v původním usecase o vytěžování rodných čísel splnila zadání, ošetřila duplicitu čísel na stejných pozicích a autorka původní usecase ještě rozšířila o vyhledávání seznamu regulárních výrazů, ne jen jedním.

7. Seznam použitých zdrojů

1. What is OCR? *DocAcquire.* [Online] [Citace: 25. 11 2020.] https://www.docacquire.com/resources/blog/what-is-ocr/.

2. IBM Datacap DDK custom actions. *ibm.com.* [Online] 18. 12 2014. [Citace: 25. 11 2020.] https://www.ibm.com/developerworks/data/library/techarticle/dm-1412datacap-developer-kit/DevWorks-Datacap-9.0-CustomActions.html.

3. Myers, Carley. Structured versus unstructured form data capture. *OCR Solutions.* [Online] 22. 01 2020. [Citace: 25. 11 2020.] https://ocrsolutions.com/structured-versus-unstructured-form-data-capture/.

4. Jak si doma vyplnit daňové přiznání za rok 2019 + vzor. *Virtuální účetní.* [Online] [Citace: 02. 11 2020.] https://virtualniucetni.com/wp-content/uploads/2019/05/1-vyplnena-page-001.jpg.

5. OCR SEMI-STRUCTURED DOCUMENTS. *SoftWorks AI.* [Online] [Citace: 02. 11 2020.] https://www.softworksai.com/reference/document-automation/ocr-semi-structured-documents.

6. ARMONK. IBM Acquired Datacap. *IBM.* [Online] 10. 08 2010. [Citace: 01. 11 2020.] https://www-03.ibm.com/press/us/en/pressrelease/32253.wss.

7. Rao, Lena. IBM Acquires Data And Document Capture Software Company Datacap. *TechCrunch.* [Online] 10. 08 2010. https://techcrunch.com/2010/08/10/ibm-acquires-data-and-document-capture-software-company-datacap/?guccounter=1.

8. Datacap. *Wikipedia.org.* [Online] [Citace: 04. 11 2020.] https://en.wikipedia.org/wiki/Datacap.

9. RuleRunner Logic. *ibm.com.* [Online] [Citace: 02. 11 2020.] https://www.ibm.com/support/knowledgecenter/SSZRWV_9.1.7/com.ibm.dc.reference.doc/dcacb590.htm.

10. Application Objects. *ibm.com.* [Online] [Citace: 10. 11 2020.] https://www.ibm.com/support/knowledgecenter/SSZRWV_9.1.7/com.ibm.dc.reference.doc/dcacb467.htm .

11. Smart Parameter special variable reference. *IBM.* [Online] [Citace: 03. 11 2020.] https://www.ibm.com/support/knowledgecenter/SSZRWV_9.1.7/com.ibm.dc.reference.doc/dcvar074.htm.

12. Jazyk regulárních výrazů – stručná referenční dokumentace. *Microsoft Docs.* [Online] [Citace: 08. 11 2020.] https://docs.microsoft.com/cs-cz/dotnet/standard/base-types/regular-expression-language-quick-reference.
13. Freitag, Peter. Regex Cheat Sheet. *Computer Science Wiki.* [Online] [Citace: 02. 11 2020.] https://computersciencewiki.org/images/thumb/d/d7/Regex_Cheat_Sheet.png/600px-Regex_Cheat_Sheet.png.
14. Prohlídka průvodce C#. *Microsoft Docs.* [Online] [Citace: 29. 10 2020.] https://docs.microsoft.com/cs-cz/dotnet/csharp/tour-of-csharp/.
15. Bory, Pavel. *C# bez předchozích znalostí - e-kniha.* Brno : Computer Press, Albatros Media a.s., 2016. str. 75. ISBN 978-80-251-4686-6.
16. File.ReadAllLines Metoda. *Microsoft Docs.* [Online] [Citace: 01. 11 2020.] https://docs.microsoft.com/cs-cz/dotnet/api/system.io.file.readalllines?view=net-5.0.
17. Motivační dopis. *Můj-životopis.cz.* [Online] [Citace: 03. 11 2020.] http://www.muj-zivotopis.cz/image/motivacni-dopis-nahled-vzor1.png.
18. Vzor faktury. *fakturoid.cz.* [Online] https://www.fakturoid.cz/images/screenshots/invoice-template/fa-lyra.pdf.

8. Přílohy

Zdrojový kód je přiložen k tištené práci na CD, v elektronickém systému ve formátu .zip.