TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Elektronická příjemka

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Tomáš Erlebach**
Vedoucí práce: Mgr. Jiří Vraný, Ph.D.
Konzultant: Petr Grusman

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Elektronická příjemka

Jméno a příjmení: **Tomáš Erlebach**
Osobní číslo: M16000022
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Seznamte se s formátem XML, jeho aplikacemi a možnosti tvorby odvozených formátů aplikací. Dále prostudujte problematiku evidence při produkci a nakládání s odpady podle aktuálně platné legislativy v ČR.
2. Na základě provedené rešerše navrhněte formát pro elektronickou příjemku, tak aby splňovala platné náležitosti.
3. Navržený formát prakticky implementujte, otestujte a ověřte jeho použitelnost v aplikacích pro zpracování průběžné evidence odpadů v elektronické podobě.

Rozsah grafických prací: dle potřeby
Rozsah pracovní zprávy: 30-40 stran
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

[1] HAROLD, Elliotte Rusty a W. Scott MEANS. XML in a nutshell. 3rd ed. Sebastopol, CA: O'Reilly, c2004. ISBN 05-960-0764-7.
[2] Zákon o odpadech č. 185/2001 Sb. [3] Vyhláška o produkci a nakládání s odpady č. 383/2001 Sb.

Vedoucí práce: Mgr. Jiří Vraný, Ph.D.
Ústav nových technologií a aplikované informatiky

Datum zadání práce: 9. října 2019
Předpokládaný termín odevzdání: 18. května 2020

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 17. října 2019

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

21. května 2020 Tomáš Erlebach

Poděkování

Rád bych poděkoval panu Petru Grusmanovi a dalším zaměstnancům společnosti INISOFT s.r.o. za to, že mi poskytli zázemí a svými připomínkami přispěli ke vzniku tohoto díla. V neposlední řadě bych chtěl poděkovat panu Mgr. Jiřímu Vranému, Ph.D. za vedení této práce.

Abstrakt

Tato bakalářská práce si klade za cíl na základě platné legislativy České republiky navrhnout formát elektronické příjemky tak, aby mohl sloužit jako podklad pro vedení průběžné evidence odpadů. Cíle je dosahováno použitím značkovacího jazyka pro popis dat. Práce se také zabývá tvorbou dalších podpůrných aplikací.

Součástí práce je analýza dané problematiky, navržení identifikátoru příjemek odpadů s důrazem na off-line použití, popsání struktury pomocí schématu a vytvoření aplikace pro převod do PDF souboru, včetně implementace webových služeb a nástrojů pro validaci.

Klíčová slova

.NET aplikace, XML, XSD, příjemka odpadů

Abstract

This bachelor thesis aims to design a digital receipt note format in accordance with valid Czech Republic laws and regulations that could serve as a basis for keeping continuous records of waste. The purpose is achieved by using a markup language to describe the data. The thesis also deals with the development of additional supporting applications.

The thesis includes the analysis of the issue, the design of a waste receipt note identifier, focusing on off-line use, a structure description using a schema, and the development of an application for conversion to a PDF file, including the implementation of web services and validation tools.

Keywords

.NET application, XML, XSD, waste receipt note

Obsah

Seznam tabulek

Seznam obrázků

Seznam zkratek

API	*Application programming interface*
DLL	*Dynamic link library*
DPH	*Daň z přidané hodnoty*
EDI	*Electronic Data Interchange*
HTML	*Hypertext Markup Language*
IČO	*Identifikační číslo osoby*
IČZ	*Identifikační číslo zařízení*
ISDOC	*Information System Document*
PDF	*Portable Document Format*
PSČ	*Poštovní směrovací číslo*
REST	*Representational state transfer*
W3C	*World Wide Web Consortium*
XML	*eXtensible Markup Language*
XSD	*XML Schema Definition*
ZUJ	*Základní územní jednotka*

Úvod

Povinnost vedení průběžné evidence odpadů vyplývá z aktuálně platného zákona o odpadech č. 185/2001 Sb. a platí pro všechny právnické osoby i fyzické osoby oprávněné k podnikání. Pro jednotlivé subjekty představuje značnou administrativní zátěž. S využitím internetu a moderních technologií by bylo možné celý proces vedení evidence značně zjednodušit, a to pomocí elektronické výměny informací o převzetí, respektive předání odpadů. Na tomto základě spočívá myšlenka elektronické příjemky odpadů.

Cílem této práce je vytvořit popis datového standartu tak, aby byl jednoduchý a zároveň uchovával všechny potřebné informace nutné pro automatické zakládání záznamů do průběžné evidence odpadů. Text práce čtenáře v první kapitole seznámí s celou problematikou příjmu odpadů a nabídne srovnání s již existujícím datovým standardem elektronických faktur ISDOC.

V následujících kapitolách je čtenář konfrontován se všemi fázemi vývoje. Celá druhá kapitola se věnuje požadavkům na řešení a specifikuje podmínky a prostředí pro realizaci finálního díla. Třetí kapitola se zabývá návrhem a předkládá možné způsoby, jak k danému problému přistoupit. Čtvrtá kapitola popisuje zvolené řešení, popisuje samotný formát příjemky a také aplikace, které při jeho realizaci vznikly. Poslední kapitola validuje dosažené výsledky a diskutuje možnosti budoucího rozšíření.

1 Elektronická příjemka

V běžném životě se člověk často shledá s potřebou uchovávat informace dokládající nějakou realitu, a to jak pro vlastní přehled či mnohem častěji z potřeby tuto skutečnost někomu prokázat později. Z tohoto důvodu se začaly používat dokumenty, které toto tvrzení zhmotnily. Jistě se každý během svého života setkal s nějakou stvrzenkou, a proto není překvapující, že tato potřeba pronikla i do světa odpadového hospodářství.

Odpadové hospodářství reguluje v České republice v souladu s právem Evropské unie Ministerstvo životního prostředí. V současnosti nakládání s odpady upravuje zákon č. 185/2001 Sb. o odpadech, ve znění pozdějších předpisů a o změně některých dalších zákonů. [1]

Paragraf 39 odst. 1 zákona č. 185/2001 Sb. o odpadech, ukládá původci odpadů a oprávněné osobě, která nakládá s odpady, vést průběžnou evidenci o odpadech a způsobech nakládání s nimi. Přičemž evidence se vede za každou samostatnou provozovnu a za každý druh odpadu. Způsob vedení evidence pro jednotlivé druhy odpadů pak stanoví prováděcí právní předpis. [1]

Představme si tedy situaci z praxe. Právnická osoba v rámci své činnosti vyprodukuje odpad. Je dle § 4 odst. 1 písmena x) zákona č. 185/2001 Sb. o odpadech původcem odpadu. V rámci ochrany životního prostředí a trvale udržitelného rozvoje movitý odpad, na který se nevztahuje výjimka § 2 zákona č. 185/2001 Sb. o odpadech odveze do zařízení, které je dle zákona č. 185/2001 Sb. o odpadech a o změně dalších zákonů v platném znění dle § 14 odst. 1, § 14 odst. 2 a § 33b odst. 1 oprávněné s těmito odpady nakládat [1], například se tedy může jednat o sběrný dvůr, který daný odpad převezme. Tuto skutečnost doloží tím, že vystaví doklad o převzetí, takzvanou příjemku.

Celý proces vedení průběžné evidence odpadů by bylo možné velmi zjednodušit, pokud by příjemce odpadu vytvořil příjemku v elektronické podobě, tak aby si původce odpadu v rámci komunikace s tímto zařízením mohl jednoduše tento doklad „načíst“ do své průběžné evidence. Odpadla by tak starost se správným přepisem informací z fyzické papírové podoby dokumentu, protože digitalizace umožní tento proces automatizovat.

1.1 Podoba příjemky

Pokud by elektronická příjemka měla sloužit jako podklad pro vedení průběžné evidence odpadů, která je dána zákonem č. 185/2001 Sb. o odpadech, je nutné se ještě seznámit s příslušnou vyhláškou, jak je uvedeno v paragrafu 39. V tomto případě to je vyhláška č. 383/2001 Sb. a ve znění pozdějších platných změn, vydaná Ministerstvem životního prostředí, pojednávající o podrobnostech nakládání s odpady. Zde se v příloze 2 nachází informace, že při přejímce je nutné zaznamenat kód odpadu, kategorii, hmotnost, datum dodávky a totožnost dodavatele, a to včetně identifikačního čísla zařízení, které se musí nacházet i na písemném potvrzení. [7]

V dalším bodě se lze dočíst, že dodavatel musí poskytnout oprávněné osobě IČO, bylo-li mu nějaké přiděleno, poté název nebo jméno a příjmení a v některých případech, podle toho, jestli je dodavatelem oprávněná osoba, tak i identifikační číslo zařízení a když je dodavatelem zrovna původce odpadu, tak ještě identifikační číslo provozovny, název, adresu a identifikační číslo základní územní jednotky. [7]

Při bližším prozkoumáním již existujících papírových příjemek, lze zjistit, že krom výše zmíněných údajů, často obsahují navíc ještě celou řadu dalších informací, jako jsou například údaje o platbě, o dopravci, celkové ceně včetně DPH, častokrát rozepsané ještě pro jednotlivé položky zvlášť, datum uskutečnění zdanitelného plnění, poznámky a informace o odpovědných osobách nebo podrobnosti o zakázce či podrobnosti o příjezdu a odjezdu nebo také kontaktní údaje a další informace. Už jen při porovnání dvou různých příjemek z různých systémů zjistíme, že to, co je obsažené na jedné se nemusí vyskytovat na druhé a tyto dodatečné informace žádný zákon pevně nedefinuje a výsledná podoba příjemky je tak do značné míry zhotovena dle vůle každého jednotlivého vystavovatele příjemky a není řízena nějakým obecným vzorem nebo pravidlem, které by bylo možné na všechny aplikovat. Tuto skutečnost je pak nutné při návrhu elektronické příjemky zohlednit.

1.2 Analýza požadavků

Pro výsledný elektronický formát je důležité, aby obsahoval všechny příslušné povinné údaje, tak aby mohl sloužit jako podklad pro průběžnou evidenci odpadů a zároveň umožnil v rámci jedné příjemky v sobě zahrnout informace o několika polož-

kách odpadů ve vztahu 1:N a to proto, aby nebylo nutné kvůli každému odpadu vystavovat příjemku novou.

Hlavním požadavkem je jednoduchost a jednoznačnost, protože navržený systém bude muset dokázat zpracovat až stovky tisíc záznamů napříč celou Českou republikou. Možnou inspirací by v tomto případě mohl být již existující standart ISDOC.

1.3 Již zavedený formát ISDOC

Formát ISDOC vznikl v minulém desetiletí jako způsob elektronizace faktur. Jeho autorem je sdružením ICT Unie. Předností formátu je automatizovaný import do ekonomických systémů, díky kterému odpadla nutnost ručně přepisovat údaje z papírových faktur, které se běžně zasílaly poštou či v elektronickém formátu PDF, který není uzpůsoben pro automatické zpracování i díky tomu, že každá faktura v tomto formátu může nabývat mnoha podob. Oproti tomu formát ISDOC je založený na jazyku XML a data jsou tak snadno přístupná řadou nástrojů či programovacích jazyků. Pro uživatelské zobrazení lze využít rovněž publikovanou čtečku ISDOCReader, která také umožňuje tisk do papírové podoby. Samotné soubory lze sdílet pomocí e-mailu či externích služeb, které formát podporují. Dobrým vzorem je například datová schránka České pošty. [2][5]

Mezi kritizované nevýhody formátu ISDOC patří lokální zaměření, které neumožňuje provádět fakturaci napříč Evropskou unií, natož celosvětově a jedná se tak o záležitost ryze českých firem a veřejné správy. Tato skutečnost brání masovějšímu rozšíření a mnohé podniky tak dávají přednost jiným řešením, jako jsou například EDI formáty. Další zmiňovanou nevýhodou formátu ISDOC je, že se nijak nezabývá přenosem a následnou archivací, takže se stává, že řada firem v rámci svého vnitřního působení stejně nakonec fakturu používá v papírové podobě a původní myšlenka formátu se tak míjí účinkem. [3]

1.3.1 Technické řešení formátu ISDOC

Jak již bylo nastíněno výše, formát ISDOC se skládá z XML souboru s příponou *.isdoc* a veřejného XSD schématu, které definuje samotnou strukturu dokumentu a obsahuje jednotlivé definice datových typů a jejich podobu včetně přípustných hodnot.[4]

Verze 6.0.1 vydaná 26. 5. 2014 obsahuje dokumentaci standardu v PDF, tutéž dokumentaci v podobě HTML stránky, XSD schéma pro daňový doklad, schéma validující strukturu digitálního podpisu, schéma nebankovního dokladu, schéma XML Signature od W3C a schéma pro manifest ISDOC archivů. Dokument ISDOC totiž může být reprezentován i ve formě ZIP archívu. V takovém případě však má příponu *.isdocx* a používá se zejména pro případy, kdy uživatel nutně potřebuje kromě samotného dokumentu přenášet i přílohy. Nejedná se tady o standardní preferovaný způsob použití formátu.

Samotný dokument musí dodržovat pravidla XML, tak aby byl správně strukturovaný podle W3C a musí být kódován v UTF-8. Validace se liší podle toho, o jaký se jedná dokument, protože každý typ má svá vlastní pravidla. Daňový doklad například nelze validovat jako neplatební dokument. [6]

Modifikace na základě uživatelských požadavků formát ISDOC řeší pomocí uživatelských položek, které se nekontrolují vůči XSD a mohou tedy v sobě zahrnovat jakoukoliv další XML strukturu. [4]

1.4 Katalog odpadů

Z kapitoly 1.1 vyplývá, že je při přejímce nutné zaznamenat kód odpadu. Zařazování odpadů je dáno přímo § 5 a § 6 zákona o odpadech č. 185/2001 Sb. a o změně některých dalších zákonů, který odkazuje na platnou vyhlášku č. 93/2016 Sb. Ministerstva životního prostředí. Tato vyhláška zpracovává směrnice Evropské unie a definuje Katalog odpadů a postup pro zařazování odpadů dle Katalogu odpadů. Jak uvádí § 4, odpady se zařazují pod šestimístná čísla, přičemž první dvojčíslí označuje skupinu odpadů, druhé dvojčíslí podskupinu a třetí druh odpadu. Přesný postup pro určení čísel pak udávají další paragrafy a v případě kdy nelze odpad jednoznačně zařadit, zařadí se dle návrhu obecního úřadu obce s rozšířenou působností. [8]

Pro návrh elektronického formátu příjemky je důležitá pouze podoba kódu odpadu, protože správná kategorizace je záležitost oprávněné osoby, která danou příjemku vystaví.

1.5 Identifikační číslo zařízení

Dle novelizované vyhlášky č. 383/2001 Sb. o podrobnostech nakládání s odpady, každé zařízení oprávněné nakládat s odpady musí být označeno identifikačním číslem zařízení, které je mu přiděleno krajským úřadem. Samotnou podobu formátu pak popisuje § 24a odst. 1 kde je definováno, že na začátku každého identifikačního čísla se vyskytuje údaj CZ následovaným písmenem označující kraj a pětimístným pořadovým číslem. [7]

Tabulka 1: Označení kraje v IČZ

Označení	Kraj
A	Hlavní město Praha
C	Jihočeský
B	Jihomoravský
K	Karlovarský
L	Liberecký
T	Moravskoslezský
M	Olomoucký
E	Pardubický
P	Plzeňský
S	Středočeský
U	Ústecký
J	Vysočina
Z	Zlínský

Například pro CZA00017 je na první pohled patrné, že se podle tabulky 1: Označení kraje v IČZ, jedná o zařízení z Prahy a dále lze v registru dohledat, že se pod tímto číslem skrývá provozovatel Technické služby Praha-Radotín. Samotný registr zařízení je dostupný prostřednictvím informačního systému odpadového hospodářství ISOH, který je provozovaný Ministerstvem životního prostředí.

2 Definice požadavků a jejich specifikace

Kromě vytvoření samotného formátu elektronické příjemky, tak aby se dala používat jako podklad pro vedení průběžné evidence odpadů, je dále nutné pro její praktické použití zajistit možnost snadné distribuce mezi jednotlivými subjekty. Jednou z možností, jak toho dosáhnout je umožnit centralizovanou správu, tak aby se příjemky daly organizovat a dalo se k nim jednoduše přistupovat z jednoho místa. Pro systémové řešení bez nutnosti nějakého složitého vyhledávání a filtrování je výhodné každou vydanou elektronickou příjemku opatřit jednoznačným identifikátorem. Toto ID pak umožní přímý přístup ke konkrétní příjemce, protože bude v rámci celé sítě jedinečné.

Jestliže tedy bude umožněno místo samotné přílohy s daty zasílat jenom toto ID, je pak také nutné zajistit stažení a správné zobrazení dat. Dále by bylo žádoucí poskytnout možnost zhotovení fyzické kopie prostřednictvím tisku.

2.1 Jednoznačný identifikátor

Ze své podstaty by měl být jednoznačný identifikátor unikátní, tak aby bylo možné na základě jeho znalosti provést identifikaci každé jednotlivé příjemky. Vzhledem k tomu, že se předpokládá nasazení v prostředí, kde nelze spolehlivě zaručit internetové připojení a zároveň je očekáváno, že bude umožněno zařízením vydávat příjemky i takzvaně v režimu off-line, nelze jednotlivé identifikátory dávkovat z centrální autority.

Pokud přistoupíme na možnost, že každé zařízení bude generovat pro své příjemky kódy vlastní, je nutné zajistit i bez znalosti ostatních identifikátorů, aby nebylo přiděleno nějaké ID již existující, protože pak by došlo k nejednoznačnosti. Je nutné tedy udržet unikátnost tohoto ID napříč celým systémem.

Zároveň kvůli uživatelskému komfortu by bylo žádoucí, aby identifikátor byl navržen tak, aby byl co nejkratší a nejjednodušší a dal se tak snadno přepisovat bez velkého rizika chyby způsobené přehlédnutím. Ze stejného důvodu by měl obsahovat pouze znaky a číslice, které se nedají zaměnit a lze je pohodlně zapsat na klávesnici.

2.2 Čtečka elektronické příjemky

Čtečka bude sloužit pro zobrazení dat ze souboru elektronické příjemky, tak aby data byla zobrazena v uživatelsky příjemné podobě. Při konečném návrhu je důležité zohledňovat zákaznické požadavky.

2.2.1 Zákaznické požadavky

Aplikace by měla umět pomocí uživatelsky zadaného ID stáhnout soubor elektronické příjemky z webové databáze a pracovat s ním, tak jako s lokálním souborem. Při zpracování bude brán zřetel na Katalog odpadů a zobrazí se tak správný název a kategorie odpadu.

Načtení souboru bude probíhat buď pomocí vložení ID nebo přetažením lokálního souboru. Po vložení XML souboru umožní aplikace zobrazit obsah příjemky a poskytne možnost pro vygenerování PDF dokumentu určeného pro tisk z této elektronické příjemky. Bylo by dobré, kdyby čtečka umožňovala pracovat s QR kódem nebo minimálně při načtení souboru umožnila zobrazit ID i ve formě QR kódu, který by se pak přidal jako další identifikátor při finálním zobrazení. Aplikace by měla také obsahovat informace o účelu aplikace.

2.2.2 Popis produktu

Aplikace bude obsahovat grafické uživatelské rozhraní, logiku provádějící načítání XML dat a modifikování podoby výstupu. Dále modul pro navazování spojení se serverem a metody pro stahování potřebných dat, rozhraní pro export do PDF a v ideálním případě i modul pro práci s QR kódy. Vše může být realizováno formou desktopové aplikace nebo se může jednat o webovou stránku. V tomto případě bude klientovi předán URL odkaz na danou stránku, na které bude daná aplikace umístěna. Přihlašovací údaje nejsou vyžadovány, aplikace bude přístupná pro kohokoliv. Zabezpečení bude realizováno na straně serveru.

2.2.3 Cílové zařízení

Aplikace bude přizpůsobena pro bezproblémový běh na specifikovaném zařízení:

- PC, operační systém Windows 10
- Display (HD 720+)
- Prostředí: C#, framework .NET
- Webové prohlížeče: Chrome, Firefox, Microsoft Edge

2.2.4 Aktualizace

Je žádoucí udržovat aplikaci v průběhu vývoje stále aktuální. Je třeba zajistit v programu aktuálnost katalogu odpadů. V případě nahrání nové verze by bylo vhodné zveřejnit soupis změn oproti verzi předchozí.

2.2.5 Grafické rozhraní

Aplikace bude obsahovat uživatelsky přívětivé grafické rozhraní uzpůsobené pro ovládání myší. Předpokládá se použití na displejích notebooků a počítačů s poměrem stan 16:9 a rozlišením větším než 720px. Grafický styl bude respektovat logo-manuál[1] společnosti INISOFT s.r.o. a obsahovat grafické prvky odkazující k Brandu společnosti INISOFT s.r.o. a to včetně barevného provedení. Aplikace bude obsahovat logo umístěné na viditelném místě. V celé aplikaci bude použit font *TeX Gyre Adventor*, jestliže však technologická či jiná omezení neumožňují použít stanovený font, bude tento font nahrazen písmem alternativním, kterým je v tomto případě font *Arial*.

[1] INISOFT: CORPORATE IDENTITY DESIGN BY FATHER & SONS VISION Květen 2015

3 Návrh řešení

Následující kapitola je věnovaná popisu přístupů a metod, které by se v rámci řešení daly uplatnit.

3.1 Návrh formátu

Jak již bylo řečeno v kapitole 1, klíčová vlastnost elektronické příjemky je schopnost přenášet informace související s průběžnou evidencí odpadů. Jinými slovy, není nutné přenášet všechny informace do druhého zařízení, i když by bylo vhodné formát zvolit tak, aby bylo v případě potřeby možné strukturu dále rozšířit, tak aby přenášela všechny požadované údaje z příjemky. Zároveň navržený formát elektronické příjemky by měl reflektovat skutečnost, že sekce obsahující informace o odpadu může obsahovat více záznamů na jedné příjemce.

Jako dobré řešení se jeví pro realizaci elektronické příjemky použít značkovací jazyk XML, protože tento způsob popisu dat je zcela přenositelný, lidsky dobře čitelný a zároveň má standardizovaný způsob validace, což je v tomto případě značná výhoda.

Strukturu výsledného souboru je možné si představit jako jednotlivé sekce (bloky), kdy samostatný blok obsahuje jednoznačný identifikátor sloužící jako ID příjemky. Další blok je vyhrazen údajům o autorovi evidence (odběrateli), zahrnující informace o názvu, identifikačním čísle osoby, provozu resp. identifikačním čísle zařízení, obci, ulici, PSČ a ZÚJ. Dále soubor bude obsahovat blok o partnerovi (dodavateli) s informacemi o něm jako je identifikační číslo osoby IČO, provoz resp. identifikační číslo provozovny, název obce a ulice, PSČ, ZÚJ... Posledním blokem, který se bude moci v rámci souboru N-krát opakovat, je položka odpad obsahující informace o kódu odpadu dle katalogu odpadů, kategorii, čisté váze v tunách, upřesňující informace o názvu odpadu a datu.

3.2 Distribuce dat

Přidanou hodnotou elektronické příjemky může být možnost centrální distribuce příjemek. Aby toto bylo možné, je nutné provozovat server, který bude jednotlivé příjemky spravovat. Jednou z možností, jak toho dosáhnout, je vytvoření webového

REST API, které bude obsluhovat centrální databázi příjemek. Pokud toto rozhraní budou implementovat aplikace odběratele a dodavatele, bude možné za určitých podmínek pro předávání a zpracování příjemek využívat této digitální platformy.

Samotná databáze může být realizována několika způsoby. První z nich je zachovat na straně serveru příjemku jako soubor uložený v binární podobě, který bude přístupný pomocí jedinečného identifikačního kódu, kterým bude každá příjemka označena, a který zároveň bude sloužit jako primární klíč v databázi. Nevýhoda tohoto řešení spočívá ve skutečnosti, že při použití tohoto řešení může docházet k velké redundanci dat. V databázi se tak může nacházet mnoho stejných položek, které se na všech příjemkách opakují a svou přítomností zabírají místo.

V případě použití relačního modelu, se nebude příjemka ukládat jako samostatný celek, ale využije se existujících vztahů. Tím, že se například uloží údaje o subjektu pouze jednou a dále se bude při výskytu na daný záznam jen odkazovat, bude možné úsporněji uložit všechna data, ovšem za cenu vyšší režie na straně serveru, protože při tomto řešení bude nutné procházet a zpracovávat každou nahranou příjemku zvlášť. Jednotlivé záznamy se transformují do připravených tabulek v databázi a v případě jejich vyžádání bude nutné je opět zase složit dohromady do cílového formátu, ve kterém se odešlou klientovi. Další možná nevýhoda může spočívat v náchylnosti k průběžným změnám ve formátu, kdy taková změna může ovlivnit správné fungování celé databáze. V případě změny či rozšíření formátu by bylo nutné upravit i příslušné schéma databáze.

Některé databáze jako je *Microsoft SQL Server* nebo *PostgreSQL* obsahují přímo datový typ XML, který lze při řešení využít.[2] V každém případě jsou však kladeny vysoké požadavky na zabezpečení databáze.

[2] Více informací o formátu na adrese: https://docs.microsoft.com/en-us/sql/relational-databases/xml případně https://www.postgresql.org/docs/current/datatype-xml.html pro PostgreSQL.

3.3 Generování ID

Tak, aby bylo umožněno pro každou příjemku generovat ID nezávisle na ostatních je třeba zajistit, aby i v případě vygenerování velkého počtů identifikátorů nedošlo v průběhu času ke shodě hodnot těchto kódů.

Na první pohled se nabízí využít například již známý algoritmus pro univerzální unikátní identifikátor definovaný standardem *RFC4122*. Tento standart je navržen pro generování kódů dlouhých 36 znaků. Výsledný kód se pak skládá ze znaků šestnáctkové soustavy a spojovníků. [9]

Pro potřeby elektronické příjemky, kde se předpokládá ruční přepis kódů do systému, se zdá být výhodnější použít pro lepší čitelnost a větší uživatelský komfort kratší sekvenci. Toho lze dosáhnout využitím znalostí prostředí, ve kterém se toto ID bude tvořit. Jako jeden ze vstupních parametrů lze využít aktuální čas a datum, protože ze své podstaty je zaručeno, že se taková kombinace už v budoucnu nezopakuje. Nicméně, aby se zamezilo duplikaci, je nutné ještě zamezit tomu, aby nějaká zařízení ve stejný čas nevygenerovala stejný kód, proto je nutné přidat další parametr.

Platné IČO subjektu je sice údaj známý, protože je obsahem každé elektronické příjemky a jedná se také o údaj nezbytný pro následnou evidenci odpadů, ale může existovat několik zařízení spadající pod jedno IČO a z tohoto důvodu nelze pro generování použít.

Oproti tomu identifikační číslo zařízení, definované novelizovanou vyhláškou č. 383/2001 Sb. Ministerstva životního prostředí o podrobnostech nakládání s odpady využít lze, protože přímo v § 24a odst. 5 této vyhlášky je garantováno, že jednou přidělené identifikační číslo se již znovu nepoužije. [7]

Aby se zredukoval potřebný počet míst pro zakódovaní informace, transformuje se výstup do nového zápisu obsahující alfanumerické znaky.

3.4 Čtečka

Následující kapitola je věnovaná popisu funkcionalit desktopové aplikace. Chování celého systému zobrazuje obrázek 1: Use case diagram.

3.4.1 Načítání XML a interakce s uživatelem

Při startu programu bude mít uživatel možnost skrz grafické uživatelské rozhraní zadat do vyhledávacího pole ID elektronické příjemky nebo přetáhnout XML soubor do aplikace, případně zadat skrz průzkumníka cestu ke zvolenému souboru, přičemž je nezbytné, aby byly odchyceny následující stavy: zvolený soubor neexistuje nebo je špatného formátu, komunikační modul vrací, že zadané ID neodpovídá žádnému souboru nebo k němu nemá uživatel přístup. Grafické rozhraní by prostřednictvím chybové hlášky mělo uživatele o vzniklé situaci informovat, a to i v případě, že není k dispozici internetové připojení. Dále by měl být modul schopen ovládat veškeré ovládací prvky a například zajistit, že se po stisknutí tlačítka spustí příslušná funkce.

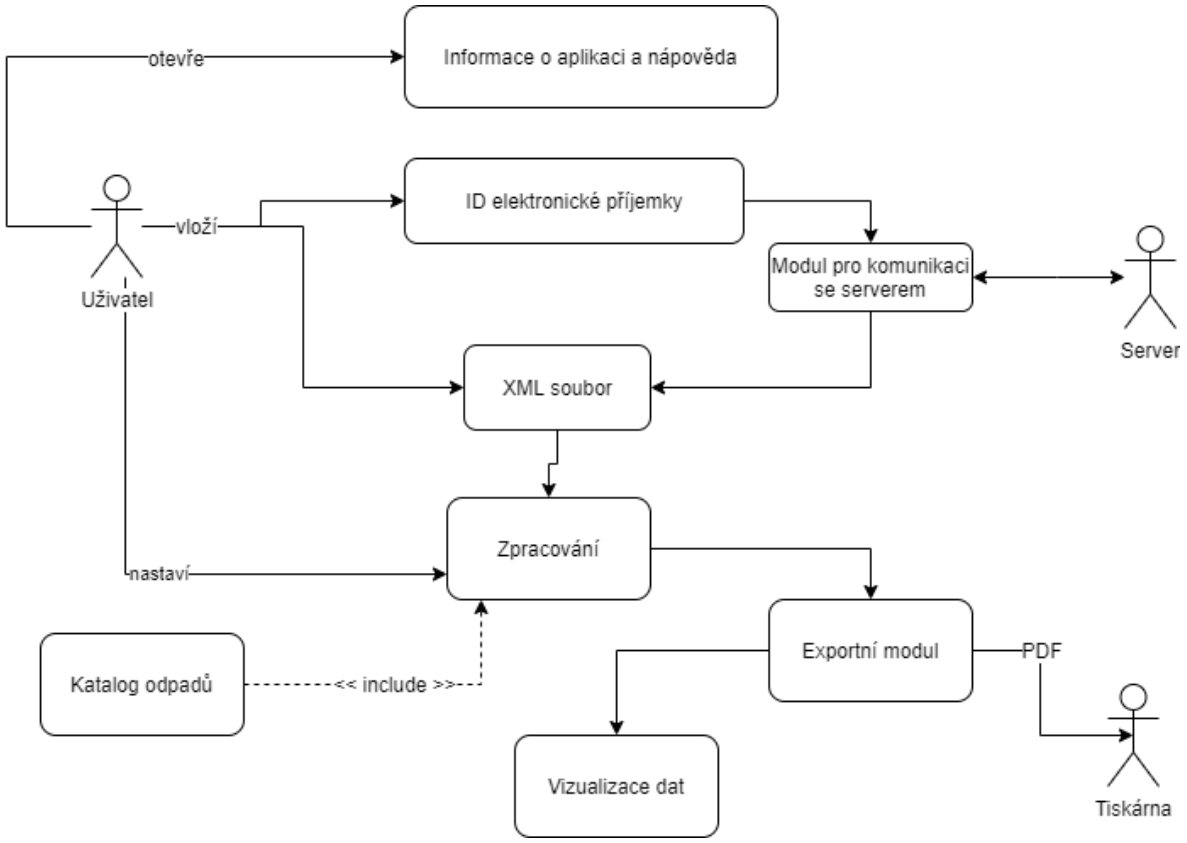
3.4.2 Komunikační modul

Bude zajišťovat veškerý síťový provoz se servery, zejména inicializovat a zprostředkovávat se serverem bezpečné připojení, posílat dotazy a dále také zajistí stažení a předání soboru. Je důležité zamezit při ztrátě internetového připojení pádu celého programu.

3.4.3 Zpracování a vizualizace

Modul bude mít na starosti veškeré zpracování XML souboru, jeho validaci a následnou přípravu pro zobrazení na display. Dále bude zajištěno, že když bude v katalogu odpadů pod kódem odpadu uvedena kategorie O (ostatní odpad), ale v XML souboru bude označení N (kategorie nebezpečný), zobrazí se na výstupu O/N. Obdobně pokud v katalogu bude N a soubor XML bude k tomuto odpadu uvádět O, zapíše se N/O. Další přidanou funkcionalitou může být převedení ID do podoby QR kódu.

3.4.4 Use case



Obrázek 1: Use case diagram

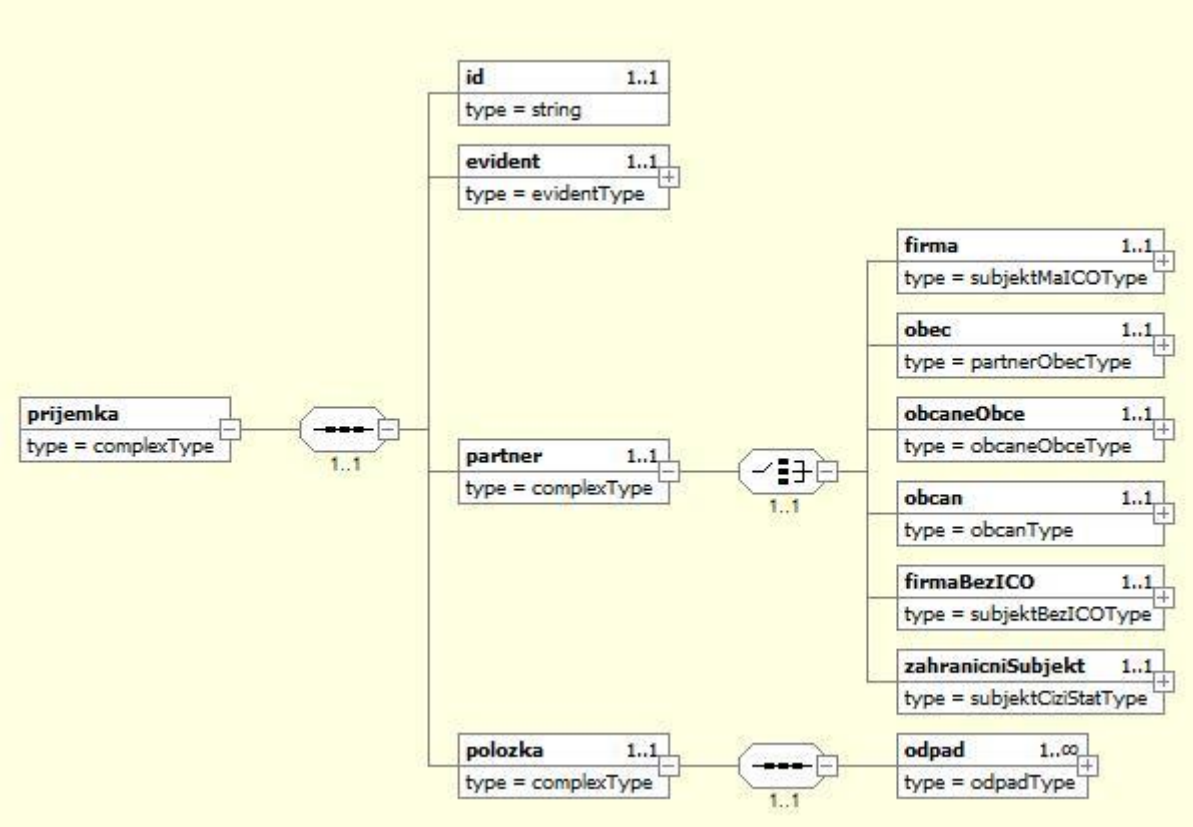
4 Realizace řešení

4.1 Formát elektronické příjemky

Data elektronické příjemky se popisují jazykem XML, který je podobně jako třeba HTML jazykem značkovacím. Oproti HTML, které se používá převážně při strukturování webových stránek, XML nemá žádné pevně nadefinované tagy, ale umožňuje tvorbu značek vlastních pro specifické použití. Veškerý obsah správně strukturovaného dokumentu je vždy uzavřen v jednom kořenovém elementu. [10] V tomto případě se jedná o tag *<prijemka>*.

4.1.1 Popis struktury

Samotná struktura příjemky je pak popsána pomocí vytvořeného XML schématu *schema.xsd.* Jedná se o další XML soubor, který je napsán dle standartu W3C a díky kterému je pak možné provádět validace jednotlivých příjemek. Tento soubor popisuje datové typy a specifikuje výskyt jednotlivých elementů příjemky.



Obrázek 2: Schéma elektronické příjemky

Formát XSD schéma byl zvolen pro popis elementů zejména díky své rozšířenosti a velké podpoře ze strany vývojových nástrojů, i když většina editorů a nástrojů je uzpůsobena pro práci s XML schématy verze 1.0 a nepodporují tak pokročilejší funkce, které byly přidány do XSD ve verzi 1.1 v roce 2012. Možnou nevýhodou tohoto řešení může být rozsáhlost a horší čitelnost oproti jazyku Relax NG, který se vyznačuje úspornější syntaxí. Toto je i přednost jazyka DTD, ovšem zde se již jedná o historické řešení, které je již v dnešní době překonané. [12]

4.1.2 Obsažené elementy

Jak dokazuje obrázek 2: Schéma elektronické příjemky, kořenový element příjemka je komplexní typ, který v sobě zahrnuje několik dalších elementů. První z nich je element *id*, který obsahuje textový řetězec, jenž slouží jako jednoznačný identifikátor každé příjemky.

Další element *evident* je uživatelsky vytvořený typ pro uložení všech informací o subjektu vytvářející příjemku. Obsahuje položku *ICO* pro identifikační číslo osoby v platném tvaru. Aby byl tento požadavek zajištěn, není použit žádný ze základních datových typů XML schématu, ale je pomocí odvození a restrikce vytvořen nový datový typ *icoType.* Tento typ obsahuje regulární výraz, který zajišťuje, že je do tohoto elementu možné vložit pouze 8 číslic. Obdobně je řešen i *pscType*, který je omezen na 5 číslic. Další položky jako je provoz, název, ulice, obec a zuj jsou omezeny pouze délkou řetězce.

Další položka, která je potomkem elementu příjemka je značka partner. Zde je již situace značně složitější. Z kapitoly 1.1 je zřejmé, že se obsah a objem informací značně mění v závislosti na subjektu, který se v dané situaci vyskytuje. Z tohoto důvodu je v definici použita konstrukce *choice*, která umožňuje použití jednoho z následujících elementů:

Element firma je definován jako uživatelský typ *subjektMaICOType* a používá se zejména v případě, kdy je partnerem subjekt, který je registrován v obchodním rejstříku. V tomto případě se vyplňuje do položky ICO platné identifikační číslo osoby, uvádí se provoz a název, dále typ subjektu, zuj a obec. Volitelně pak lze ještě zaznamenat informace tagem pro ulici, poštovní směrovací číslo a kód státu. Toto je umožněno díky použití argumentu *minOccurs*, který je v tomto případě nastaven na hodnotu nula.

Element *obcan* je určen pro případ, kdy jako partner vystupuje fyzická nepodnikající osoba. Je definován jako *obcanType*, kde jsou zahrnuty tagy pro typ subjektu, obec, ulici, jméno, příjmení, datum narození, zuj, typ průkazu a číslo průkazu, přičemž typ průkazu se uvádí zkratkou danou konvencí. V případě cestovního průkazu se uvede písmeno C, občanský průkaz má písmeno O, pro řidičský průkaz pak písmeno R a povolení k trvalému pobytu potom značí písmeno T. Jiné hodnoty nejsou povoleny.

Zvláštním typem je poté element *obcaneObce*. Typ *obcaneObceType* definuje pouze položku typ subjektu, název a zuj. Je to dáno tím, že lze evidovat předání za celou obec.

Pro případy, kdy subjekt nemá IČO se použije typ *subjektBezICOType*. Element s tímto typem pak místo identifikačního čísla osoby použije identifikátor. Zbytek položek je obdobný jako v případě *subjektMaICOType*, který definuje tagy pro typ subjektu, název, ulici, obec, psč, zuj a kód státu.

SubjektCiziStatType je označení pro datový typ elementu *zahranicniSubjekt*, který obsahuje typ subjektu pevně definovaný na Z a název a kód státu, který se uvádí dle ISO 3166-1 alpha-2 dvěma znaky reprezentující zemi původce. Zbytek položek jako je IČO, provoz, ulice, obec, psč a zuj patří mezi volitelné a nemusí se uvádět.

Posledním přímým potomkem definovaným XSD souborem je element *polozka*. Jedná se o komplexní typ, který v sobě obsahuje libovolný počet elementů odpad typu *odpadType*. Tento element má nastavený parametr *maxOccurs* na hodnotu *unbounded*. Díky tomu je zaručeno, že na jednu příjemku lze přidat několik odpadů najednou. Samotný element *odpad* obsahuje element datum, který je definován uživatelským typem *datumType*, který je regulárním výrazem omezen na platné hodnoty data ve formátu DD.MM.RRRR. Tím pádem je možné za DD dosadit pouze čísla 01 až 31, tak aby den odpovídal kalendářní hodnotě, obdobně měsíc v rozsahu 01 až 12 a rok byl uveden čtyřciferným číslem. Dalším tagem je *kodOdpadu* pro zápis kódu odpadu dle Katalogu odpadů. Element kategorie je typu *kategorieType*, který připouští pouze zápis hodnoty N pro nebezpečný odpad a kategorii O pro ostatní. Element množství je omezen typem *mnozstviType* na desetinné číslo větší než nula. Pro poznámky je pak vyhrazen element *upresneni*, který povoluje řetězce o maximální délce 60 znaků.

XSD schéma pro elektronickou příjemku definuje základní strukturu elektronické příjemky tak, aby vyhověla potřebám průběžné evidence odpadů. Nezahrnuje dodatečné informace, které mohou být předmětem dalšího rozvoje. Použití jazyka XML ostatně umožňuje budoucí rozšíření struktury.

4.2 Generátor

Z kapitoly 2.1 vyplývá potřeba unikátního identifikátoru, který se bude používat pro označování každé příjemky. Z tohoto důvodu vznikla aplikace generátoru. Vývoj probíhal agilním způsobem, to znamená, že v průběhu vývoje docházelo ke změnám na základě připomínek společnosti INISOFT s.r.o. Podoba prvotní aplikace je zachycena na obrázku 3: První verze generátoru.

Obrázek 3: První verze generátoru

Postupem času docházelo jak k úpravě kódovacího algoritmu, tak samotné podoby generátoru, tak aby výsledná aplikace co nejlépe odpovídala specifikaci a zároveň byla vhodná i pro off-line použití. Hlavním nedostatkem, který se podařilo v průběhu vývoje odstranit, byla nízká variabilita kódu. Stávalo se, že jedno přihlášené zařízení,

které generovalo identifikátor v krátkých intervalech po sobě vytvářelo výsledný kód velmi podobný kódu předchozímu. Typicky se kód lišil jen v několika málo znacích.

Obrázek 4: Uživatelské rozhraní finální aplikace

Na základě zkušeností s vývojem předchozích verzí generátorů je současný generátor navržen tak, aby umožnil pružnou reakci na budoucí uživatelské požadavky. Například lze zvolit povolené znaky, formát výstupu, složení kódu pro generování, a to včetně možnosti implementování dalších regionů, pro které přirozeně neplatí pravidla typická pro česká zařízení. Výsledku je dosaženo obecným objektovým návrhem, který oproti předchozím verzím obsahuje pokročilejší funkce. Díky bezztrátové konverzi je možné pomocí inverzních funkcí výsledný kód zpátky dekódovat.

4.2.1 Popis aplikace

Výsledná aplikace generátoru je napsaná v jazyce C# s využitím technologie .NET Framework 4.7.2. Uživatelské grafické rozhraní slouží zejména k obsluze třídy *GeneratorID*. Tato třída obsahuje v nejnovější verzi navíc ještě metody pro dekódování a formátování, které byly do aplikace přidány později a nejsou přes toto rozhraní přístupné, protože se předpokládá užití metod tříd generátoru hlavně v podobě importované DLL knihovny nebo voláním uvnitř projektu.

Samotné generování kódu s defaultním nastavením se provede pomocí výchozího objektu generátoru a voláním jeho funkcí *GenerateCode* či *GenerateAutoFormattedCode*.

Obě metody, jak *GenerateCode* tak *GenerateAutoFormattedCode* přijímají na vstupu parametr *icz*, který je typu string a při tomto nastavení musí odpovídat platnému identifikačnímu číslu zařízení, tak jak je popsán v kapitole 1.5, jinak je výsledkem výjimka *System.ArgumentException*. V případě grafického rozhraní je správný formát vstupu kontrolován pomocí regulárního výrazu.

4.2.2 Postup generování kódu

Zpracování probíhá v privátní metodě *CodeGenerator*, která má kromě parametru *icz* ještě druhý parametr pro specifikování času pomocí objektu *DateTime.* V případě, že se nepoužije přetížená metoda pro generování kódu, použije se aktuální čas získaný pomocí vlastnosti *DateTime.Now*. *CodeGenerator* provede ve svém těle volání metody *Serialization* objektu *Core*, což je interface, které slouží k definování obslužných funkcí generátoru. Ve výchozím nastavení je pro zpracování nastavena třída *CZParser*, která toto rozhraní implementuje.

Funkce *Serialization* slouží k úpravě vstupu pro potřeby generátoru. V první řadě pomocí funkce *ConvertICZ* převede znaky z IČZ na jejich číselnou reprezentaci a zarovná na požadovaný počet míst. Převod je ovlivněn nastavením atributu *iczPattern* a ve výchozím nastavení se kód kraje nahrazuje dle tabulky 2: Převod znaků na číselnou hodnotu.

Tabulka 2: Převod znaků na číselnou hodnotu

Kód	Hodnota	Kraj
CZA	01	Hlavní město Praha
CZS	02	Středočeský
CZU	03	Vysočina
CZL	04	Olomoucký
CZK	05	Moravskoslezský
CZH	06	Královéhradecký
CZE	07	Karlovarský
CZP	08	Plzeňský
CZC	09	Jihomoravský
CZJ	10	Liberecký
CZB	11	Jihočeský
CZM	12	Pardubický
CZT	13	Ústecký
CZZ	14	Zlínský

Následně se převede *DateTime* na objekt *DateTimeString,* což je speciálně upravená třída, která umožňuje přistupovat k hodnotám jako k souvislému textovému řetězci bez interpunkce a zároveň zaručí správnou interpretaci dat. Toto je nezbytné zejména z toho důvodu, aby nedocházelo k nejednoznačnosti, kdy jsou do řetězce doplněné nuly, tak aby se dal odlišit první listopad od jedenáctého ledna a podobně.

Poté co jsou všechny vstupní parametry převedeny na řetězce čísel dochází k vytvoření výsledného kódu sloučením a promícháním hodnot v proměnných. Tato operace je ovlivněna nastavením atributu šablony *codePattern*. Změna šablony má zcela zásadní vliv na konečnou podobu kódu a jen s její znalostí lze provést správné dekódování. Šablona se skládá ze značek, které reprezentují jednotlivé znaky případně operace, které se nad kódem provedou. Výchozí podoba šablony je:

{ "!" ,"s1", "d1", "d2", "F1", "F2", "ICZP1", "ICZP2", "Y4" , "ICZN3", "ICZN4", "H2", "s2", "#", "!", "M1", "Y1", "F3", "M2", "Y2", "ICZN5", "ICZN6", "ICZN7", "Y3", "H1", "m1", "m2" }

Znak „!“ říká, že v další iteraci bude aplikována na aktuální hodnotu funkce označená vlaječkou. V tomto případě to znamená, že se změní o 1 její hodnota. Symbol „#“ zase znamená, že je ukončena aktuální sekvence a začne se tvořit sekvence nová. Další znaky slouží pro pozicování a jejich význam udává tabulka 3: Význam značek v kódovací šabloně. Výsledkem funkce *Serialization* je list longů.

Metoda *CodeGenerator* veškerý výstup z funkce *Serialization* pošle do metody *Encoder,* což je metoda kodéru a pak z výsledku sestaví konečný kód.

4.2.3 Metoda kodéru

Kodér funguje na principu převodu desítkového zápisu číselné hodnoty do jiné soustavy. Cílová soustava je definována atributem *BaseAlphabet*, který obsahuje ve formě stringu všechny povolené symboly, které se mohou v kódu vyskytovat. Ve výchozím bezparametrickém konstruktoru se do abecedy přiřadí pouze čísla a verzálky anglické abecedy s tím, že jsou odebrány znaky, u kterých by mohlo dojít snadno při ručním opisu k záměně. Typickým příkladem je například symbol 0, O a Q, kdy při použití specifických fontů je těžké je vzájemně rozeznat. Podobně matoucí je také znak I a číslice 1.

Kódování probíhá uvnitř funkce *Encoder* z 64bitového čísla datového typu long za předpokladu, že je vstup větší než nula tak, že je z tohoto čísla vypočítán zbytek po dělení počtem symbolů v abecedě. Na základě jeho hodnoty je do výsledného řetězce přiřazen odpovídající znak z abecedy, který se nachází na daném indexu. Následně je vstup vydělen velikostí abecedy a pokud je stále toto číslo větší než nula, tak se proces opakuje, dokud se tak nestane. Nakonec je řetězec vrácen pozpátku pomocí privátní statické funkce *ReverseString,* která převede string na pole znaků a převrátí v novém řetězci jejich posloupnost, takže poslední znak pole se stane prvním a první znak se stane posledním. Výsledkem je reprezentace čísla v nové soustavě.

Tabulka 3: Význam značek v kódovací šabloně

Značka	Údaj
s1	desítky vteřin
s2	jednotky vteřin
d1	desítky dní
d2	jednotky dní
F1	stovky milisekund
F2	desítky milisekund
F3	jednotky milisekund
H1	desítky hodin
H2	jednotky hodin
ICZN3	třetí znak icz
ICZN4	čtvrtý znak
ICZN5	pátý znak icz
ICZN6	šestý znak icz
ICZN7	sedmý znak icz
ICZP1	první znak perfixu icz
ICZP2	druhý znak perfixu icz
M1	desítky měsíce
M2	jednotky minut
m1	desítky minut
m2	jednotky minut
Y1	tisíce let
Y2	stovky let
Y3	desítky let
Y4	jednotky let

4.2.4 Výsledná podoba kódu

Výstup kodéru je nutné v některých případech ještě před publikováním upravit. Klíčovou roli v konečné podobě kódu sehrává zvolená implementace rozhraní *Igenerator-Core.* Modifikace třídy *GeneralParser* umožňuje tvorbu různých kódů. Jediná limitace

je dána možnostmi kodéru, které jsou omezeny maximální velikostí datového typu long, kterou nelze při návrhu překročit. Pokud je nutné kódovat větší číslo je nutné ho rozdělit do několika sekvencí jako to je v případě výchozí implementace pomocí odvozené třídy *CZParser*. Zde je nutné při modifikaci šablony pro kódování počítat s tím, že se řetězcová reprezentace v následujícím kroku převede na číslo, a tedy pokud je prvním znakem nula, tak se dál nepřenáší a výsledek je o řád menší. Pakliže se jedná o několik sekvencí jednoho kódu rozdělených na části, nejlepšího výsledku se dosáhne, pokud budou všechny sekvence stejně dlouhé. V opačném případě lze pro potřeby dekódování přidat dělící znak v podobě atributu Joker, který se dá v generátoru také definovat.

Metoda *GenerateCode* po zavolání funkce *CreateResultString,* která spojí všechny části do jednoho řetězce, vrací finální hodnotu kódu. Oproti tomu metoda *Generate-Auto-FormattedCode* provede ještě naformátovaní kódu rozdělením znaků do skupin oddělených pomlčkou pomocí funkce *Autoformat.* Velikost jednotlivých skupin je ovlivněna celkovou délkou kódu. Pro elektronickou příjemku se za standartních okolností generují kódy se čtyřmi skupinami o čtyřech znacích. Například pro příjemku generovanou 26. března 2020 v 17:30 zařízením s identifikačním číslem CZL00012 by formátovaný kód měl hodnotu EU9D-3Z4T-EMUA-RPB9.

Formátování kódu lze také provést kdykoliv později voláním statické funkce *Format-Code.* Pomocí parametru *ngroups* se definuje počet písmen ve skupině a parametr x slouží pro vložení oddělovacího znaku, pokud nelze použít výchozí pomlčku.

Generátor lze v budoucnu rozšířit tak, aby umožňoval kódování i podle jiného zadání. Jestliže pro kódování nelze využít třídu *CZParser,* například pro kódování identifikátorů v jiné zemi, kde nemají identifikační číslo zařízení, je možné přidat třídu zcela vlastní, podle vlastních pravidel. Pro správné fungování generátoru je pouze nutné, aby implementovala rozhraní *IgeneratorCore.* To obsahuje dvě funkce, *Serialization* slouží pro příjem dat do *Encoderu* jako list longů, má dva parametry, první icz je datového typu string a obvykle se používá jako identifikátor, druhý parametr je datového typu DateTime. Inverzní funkce je *Deserialization,* která slouží pro dekódování, kterému je věnovaná další kapitola.

4.2.5 Dekódování

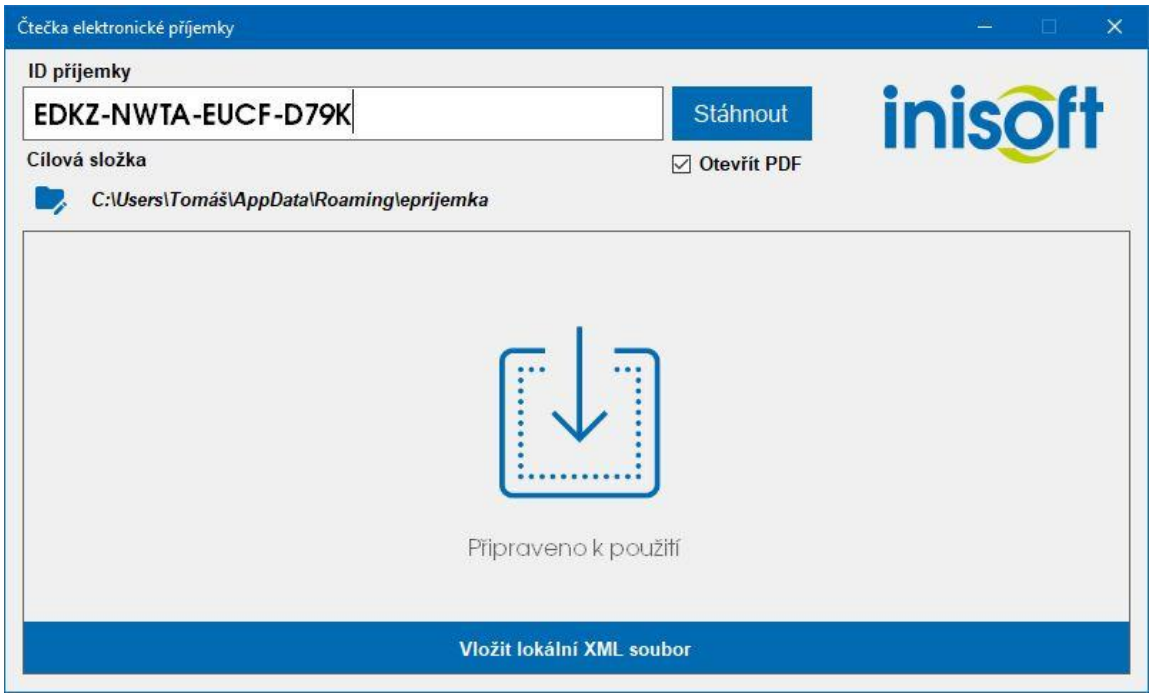
Vygenerovaný kód je možné kdykoliv zpátky dekódovat. Slouží k tomu funkce *Decode*, která přijímá jako parametr *code* textový řetězec obsahující kód z generátoru. Výsledkem funkce je datový typ Tuple, který jako první položku obsahuje textový řetězec s identifikátorem a jako druhý prvek je obsažen objekt DateTime s časem původního zakódování. Pomocí dalších parametrů se dá dodefinovat způsob dělení sekvencí v dekodéru. Ve výchozím případě se dělí kód na dvě části, které se dekódují zvlášť. Pro jejich dekódování se používá privátní funkce *Decoder*, která má jediný parametr datového typu string a vrací číslo ve formátu long. Převod znaků na jejich číselnou reprezentaci probíhá postupným vynásobením indexu příslušného znaku v dané abecedě s umocněnou velikostí abecedy řádem.

Posledním krokem k získání dekódovaných parametrů je na veškerý sloučený výstup z dekodéru aplikování funkce *Deserialization* z *IgeneratorCore* interface, která převede řetězec zpět na identifikátor a DataTime. Funkci *Deserialization* ve výchozím nastavení implementuje třída *CZParser.* Správné sestavení je podmíněno použitím vhodného nastavení šablony *codePattern.*

4.3 Čtečka elektronického formátu

Čtečka elektronické příjemky je jednoduchá Windows Forms aplikace vyvinutá na technologii .NET Framework 4.7.2 za účelem vizualizace dat z příjemky. Uživatelské rozhraní zachycené na obrázku 5: Aplikace čtečky, obsahuje textové pole pro vkládání identifikátoru příjemky, tlačítko pro stažení příslušné příjemky, tlačítko pro změnu cílového adresáře pro export, volbu pro automatické otevření vygenerovaného PDF souboru, oblast pro přetažení souboru příjemky do aplikace případně tlačítko pro nastavení cesty k lokálnímu souboru.

Čtečka vznikla ve dvou verzích, v jedné s podporou webových služeb, kdy je možné rovnou stahovat příjemky ze serveru pomocí identifikátorů a následně v jednodušší freeware variantě, kdy slouží čtečka pouze jako validátor a konvertor XML dat do PDF souboru.



Obrázek 5: Aplikace čtečky

4.3.1 Komponenty aplikace

Součástí obou verzí čtečky je vložený Katalog odpadů. Jedná se o zkompilovaný XML dokument obsahující informace o názvu odpadu, kódu odpadu, kategorii a vztahu N/O a O/N. Načtení objektu katalogu probíhá přímo v konstruktoru třídy *CteckaEP.*

Za účelem validace příjemek je v projektu vytvořená třída *XSDValidator,* která slouží ke kontrole struktury a obsahu oproti předepsanému XSD schématu. Slouží k tomu funkce *Validate*, která v parametru přijímá dané XML, které zkontroluje. Schéma, jenž je popsáno v kapitole 4.1.2 je součástí konstruktoru a je do aplikace zakomponováno rovnou už jako vestavěný zdroj, ke kterému se přistupuje přes Stream sestavení. Pro případ budoucí potřeby je ve třídě připravena i funkce *ValidateXMLfileAgainstXSDfile*, která kromě souboru XML v parametru přebírá také cestu k XSD souboru, který se pak následně použije pro validaci. Veškeré validační události zachytává privátní funkce *ValidationEventHandler* a veškeré výstupy jsou logovány do listu *Errors.* Validace vrátí stav *true* pouze v případě, že je list *Errors* prázdný. Při každém volání validace se list resetuje.

V případě čtečky umožňující stahování příjemek je pro komunikaci se serverem přítomna třída *APiClient.*

4.3.2 Stahování příjemek

Stahování probíhá ze serveru společnosti INISOFT s.r.o., kde je umístěno REST API poskytující přístup k databázi. Zabezpečení je realizováno pomocí OAuth 2.0. *HttpClient* si vyžádá na základě přihlašovacího jména a hesla bezpečnostní token, přes který jsou následně realizovány všechny GET požadavky. Pro získání příjemky slouží asynchronní metoda *getEprijemka,* která má jediný parametr pro identifikátor příjemky. V případě zapsaní špatného ID je vrácena chyba *InvalidIDException.*

4.3.3 Vizualizace

Vizualizace elektronické příjemky je realizována pomocí knihovny MigraDoc PDFsharp, která je distribuovaná pod MIT licencí, která umožňuje i komerční použití s uzavřeným kódem. Pro potřeby čtečky byla vytvořena třída *PDFGeneratror*, která tuto knihovnu integruje. V této třídě je definován vzhled a podoba všech položek elektronické příjemky a do pravého horního rohu je vložen QR kód s identifikátorem, jak je patrné z obrázku 6: Vizualizace elektronické příjemky.

EDKZ-NWTA-EUCF-D79K

Evident
IČO: 25417657
Inisoft odpady s.r.o.
Rumjancevova 696/3
46001 Liberec
ZUJ: 556904

Partner
IČO: 45678902
Metal Produkt - Železná Drahomíra
Žižkova 3b
33202 Starý Plzenec
ZUJ: 558371

Položky

Datum	Kód	Ktg.	Název	Množství	Upřesnění
11.02.2020	170405	O	Železo a ocel	2,23 t	Železo a ocel
11.02.2020	170402	O	Hliník	0,7 t	Hliník
12.02.2020	170402	O	Hliník	0,7 t	Hliník

Obrázek 6: Vizualizace elektronické příjemky

5 Vyhodnocení řešení

Ověřování praktické využitelnosti formátu pro průběžnou evidenci odpadů probíhalo v kooperaci s IT společností INISOFT s.r.o., která má dvacetiletou praxi s vývojem a distribucí programů v odpadovém hospodářství. Díky tomu bylo možné vytvořit formát elektronické příjemky tak, aby byl v plném rozsahu použitelný v rámci softwarových produktů této firmy. Společnost INISOFT s.r.o. tak bude moci jako první navržený formát elektronické příjemky ve svých aplikacích využít. Kontrola funkčních požadavků pak probíhala pomocí testů.

5.1 Testování

5.1.1 Unit testy

Ověření funkčnosti bylo prováděno pomocí sady testů. V první řadě unit testy prověřovaly správnou funkčnost metod v izolované části kódu. Třída generátoru byla testována komplexně pomocí třídy *GeneratorIDTests,* která na několika vzorcích prováděla kódování a zpětné dekódování identifikátorů včetně zobrazení ve správném formátu kódu. Všechny testy byly úspěšné. Testování probíhalo i pro maximální a minimální hodnoty vstupů DateTime, přestože se jejich reálné použití v praxi nepředpokládá. S nadsázkou lze prohlásit, že generování bude funkční ještě několik tisíc let.

5.1.2 Systémové testy

Stěžejní částí testování bylo použití systémových testů. Testování v tomto případě probíhalo na předem připravených DEMO datech. Ověřovala se tak funkčnost aplikací jako celku. Byla použita sada různých dat, která simulovala možné situace. V případě čtečky to například bylo prováděno načítáním příjemek s různými typy partnerů a následnou vizuální kontrolou výstupu.

5.1.3 Akceptační testy

Akceptační testy byly prováděné klientem v sídle společnosti INISOFT s.r.o. a sloužily jako důležitá zpětná vazba. Díky ní se například podařilo odhalit problém, kdy se

při instalaci aplikace objevila hláška o nepodporované verzi .NET Frameworku, což bylo způsobeno nekompatibilní verzí operačního systému.

5.2 Možnosti rozšíření

5.2.1 XSLT transformace

Vzhledem k tomu, že jazyk XML řeší pouze data a nikoliv vzhled a výslednou vizuální podobu dokumentu, je nutné pro dosažení cílového vzhledu užít dalších prostředků. Jednou z dalších možností, jak vizualizovat data z elektronické příjemky, je použití jazyka XSL, který umožňuje transformovat původní XML soubor elektronické příjemky do jiného formátu, kterým může být třeba soubor HTML, který lze pak dále nastylovat pomocí CSS do požadované podoby. Pro připojení XSLT ke XML je třeba do XML dopsat tag *<?xml-stylesheet?>,* který má dva povinné atributy a sice parametr *href,* do kterého se zapíše adresa umístění souboru XSLT a atribut type, který bude nabývat hodnoty *text/xsl.* Samotné zpracování probíhá pomocí XML parseru, který postupně aplikuje šablony zapisované do tagu *<xsl:template>.* Atribut *match* určuje, na které elementy se příkazy použijí, parametr *name* umožňuje šablonu pojmenovat pro budoucí volání, dále pomocí atributu priority můžeme nastavit prioritu, která ovlivňuje, která šablona se v danou chvíli použije na konkrétní uzel. V těle šablony lze použít velké množství instrukcí, které mají vliv na výstupní uzly. [11]

5.2.2 Kompatibilita s formátem InisoftMini

InisoftMini je importní formát vyvinutý společností INISOFT s.r.o. Formát elektronické příjemky by bylo možné za určitých podmínek a s jistým omezením převést do tohoto formátu vhodným nástrojem. Příjemka neobsahuje všechny položky formátu InisoftMini, například "kód nakládání dle číselníku ve vyhlášce" není ve formátu obsažen. Toto je věc, kterou si každý uživatel určí sám v rámci role, kterou v daném případě zastává. Formát elektronické příjemky nese pouze informaci o tom, kdo danou příjemku vystavuje. Vztah mezi jednotlivými subjekty, kdy zařízení přijímá odpad od původce, který zároveň v tomto případě eviduje předání odpadu by musel být definován uživatelským zásahem.

5.2.3 Zabezpečení

Pro zvýšení důvěryhodnosti elektronické příjemky by bylo vhodné opatřit ji možností elektronického podpisu případně k příjemce připojit kontrolní součet, který by zaručil, že během přenosu nedošlo k manipulaci s daty. Minimalizovala by se tak možnost podvržení, protože příjemce by si mohl sám ověřit po obdržení příjemky, že se jedná o autentický soubor.

5.2.4 Přenášení dalších informací

Současný formát elektronické příjemky neobsahuje všechny informace, které se mohou na příjemce odpadů vyskytovat, případně které by byly žádoucí, ačkoliv aktuálně nejsou pro vedení průběžné evidence nutné. Strukturu XML dokumentu by bylo možné v budoucnu rozšířit dalšími tagy. Přidání dalšího bloku by umožnilo například pro potřeby dodavatele nebo odběratele zaznamenat informace o ceně, kterou by pak následně mohl importovat nějaký účetní program.

Závěr

Elektronická příjemka je dalším příkladem procesu převodu fyzických dokumentů do digitální podoby. Aby toto bylo možné, bylo nutné se seznámit s formátem XML a následně navrhnout vyhovující popis struktury dat. Dále bylo nezbytné prostudovat problematiku evidence při nakládání s odpady, a to podle aktuálně platné legislativy České republiky a tyto znalosti pak vzít v úvahu při celkovém návrhu.

Praktickou implementaci bylo možné realizovat díky jednoznačné definici požadavků a vymezení specifik prostředí finálního produktu. V rámci procesu návrhu byly tyto okolnosti zohledněny, a i díky agilnímu způsobu vývoje softwaru se podařilo odladit funkčnost a otestovat výsledné podpůrné aplikace.

Výsledkem bakalářské práce je popsání struktury elektronické příjemky s využitím standartu XSD, navržení identifikátoru příjemek odpadů tak, aby bylo možné i off-line použití pomocí aplikace generátoru. Klíčovou součástí je i vytvoření aplikace sloužící k převodu příjemky do tisknutelného PDF souboru včetně nástroje pro validaci elektronických příjemek. Přidanou funkcionalitou je pak umožnění prostřednictvím této aplikace přímé stahování existujících příjemek na základě platného identifikátoru ze serveru pomocí webových služeb. Praktické výstupy této práce využije ve svých produktech partner TU Liberec, IT společnost INISOFT s.r.o. Elektronickou příjemku je tak možné v budoucnu dále rozvíjet a rozšiřovat její funkce a využití.

Použitá literatura

[1] ČESKO. Zákon č. 185/2001 Sb., o odpadech a o změně některých dalších zákonů. In: *Zákony pro lidi.cz* [online]. © AION CS 2010-2020 [cit. 13. 4. 2020]. Dostupné z: https://www.zakonyprolidi.cz/cs/2001-185

[2] FERSCHMANN, Petr. Místo papírového dokladu přichází elektronická faktura ISDOC. Lupa.cz [online]. Praha 6: Internet Info, 2009 [cit. 2020-04-04]. Dostupné z: https://www.lupa.cz/clanky/elektronicka-faktura-isdoc-uvod

[3] LICHÝ, Alexander. Český formát elektronických faktur ISDOC zatím vize nenaplnil. CIO Business World on-line [online]. Praha 6: Internet Info DG, 2013 [cit. 2020-04-04]. Dostupné z: https://businessworld.cz/aplikace/cesky-format-elektronickych-faktur-isdoc-zatim-vize-nenaplnil-11328

[4] KUCHAŘ, Petr. E-fakturace po Česku aneb ISDOC přichází. *Computerworld: Deník pro IT profesionály* [online]. Praha 6: Internet Info DG, 2009, 31.07.2009 [cit. 2020-04-05]. Dostupné z: https://computerworld.cz/software/e-fakturace-po-cesku-aneb-isdoc-prichazi-4435

[5] ISDOC. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-03-21]. Dostupné z: https://cs.wikipedia.org/wiki/ISDOC

[6] ISDOC 6.0.1: Národní standard pro elektronickou fakturaci. Isdoc.cz [online]. Praha 1: ICT UNIE, 2014, 26. května 2014 [cit. 2020-04-06]. Dostupné z: http://www.isdoc.cz/6.0/doc/isdoc.html

[7] ČESKO. Vyhláška č. 383/2001 Sb., Ministerstva životního prostředí o podrobnostech nakládání s odpady. In: Zákony pro lidi.cz [online]. © AION CS 2010-2020 [cit. 13. 4. 2020]. Dostupné z: https://www.zakonyprolidi.cz/cs/2001-383

[8] ČESKO. Vyhláška č. 93/2016 Sb., o Katalogu odpadů. In: *Zákony pro lidi.cz* [online]. © AION CS 2010-2020 [cit. 16. 4. 2020]. Dostupné z: https://www.zakonyprolidi.cz/cs/2016-93

[9] LEACH, P., M. MEALLING a R. SALZ. MICROSOFT. RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace. LEVINE, John a Adrian FARREL. ASSOCIATION MANAGEMENT SOLUTIONS, LLC. RFC Editor [online]. USA: The Internet Society, 2005 [cit. 2020-04-19]. Dostupné z: https://www.rfc-editor.org/info/rfc4122. DOI: 10.17487/RFC4122.

[10] KOSEK, Jiří. XML pro každého: podrobný průvodce. Praha: Grada, 2000. ISBN 80-716-9860-1.

[11] ŽÁK, Miroslav. XML – začínáme programovat: podrobný průvodce začínajícího uživatele. 1. Praha: Grada, 2003. ISBN 80-247-0565-6.

[12] KOSEK, Jiří. XML schémata. In: Kosek.cz [online]. 2013 [cit. 2019-12-15]. Dostupné z: https://www.kosek.cz/xml/schema/xmlschema.pdf

Přílohy

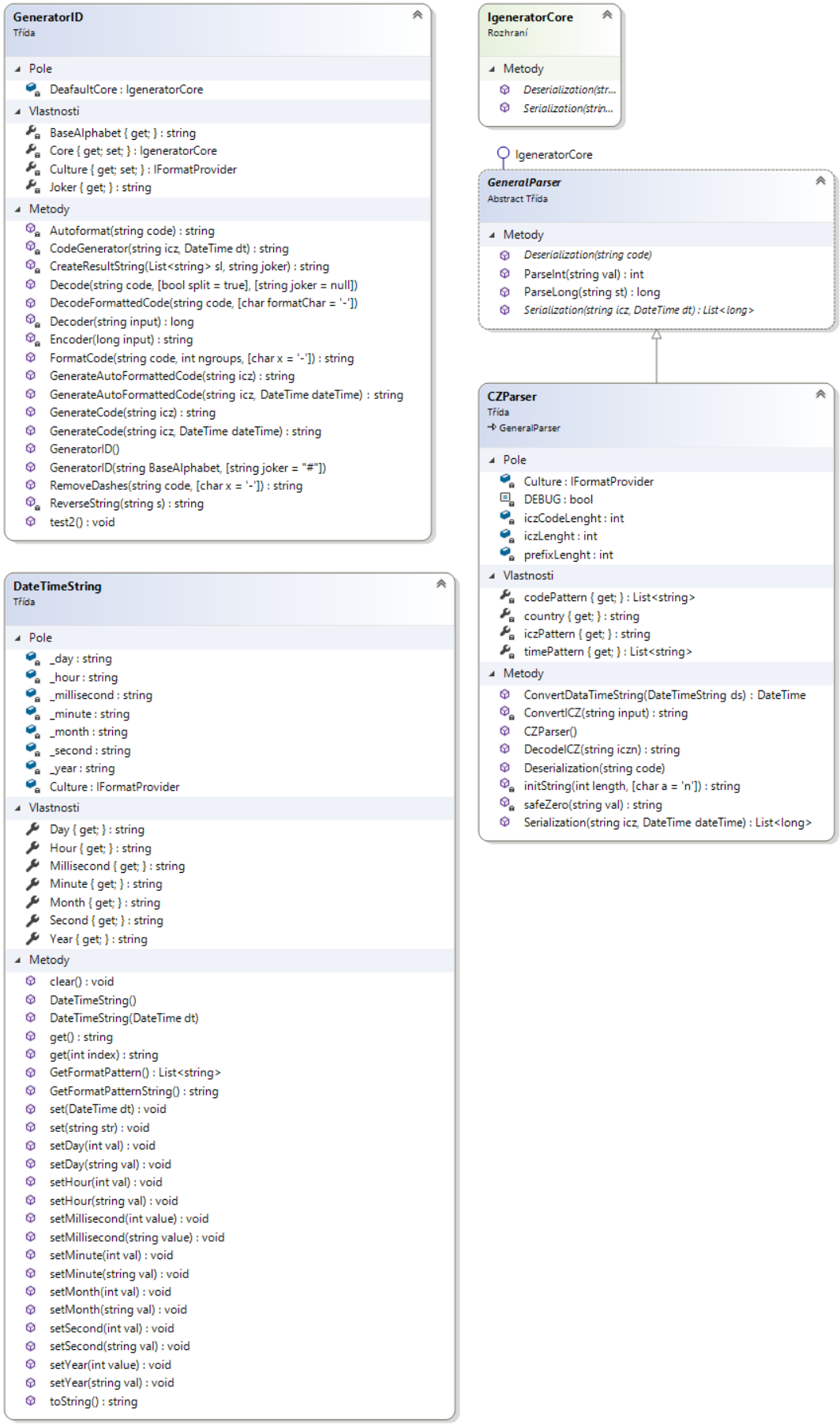
A. Class diagram generátor.exe

B. Class diagram čtečka.exe

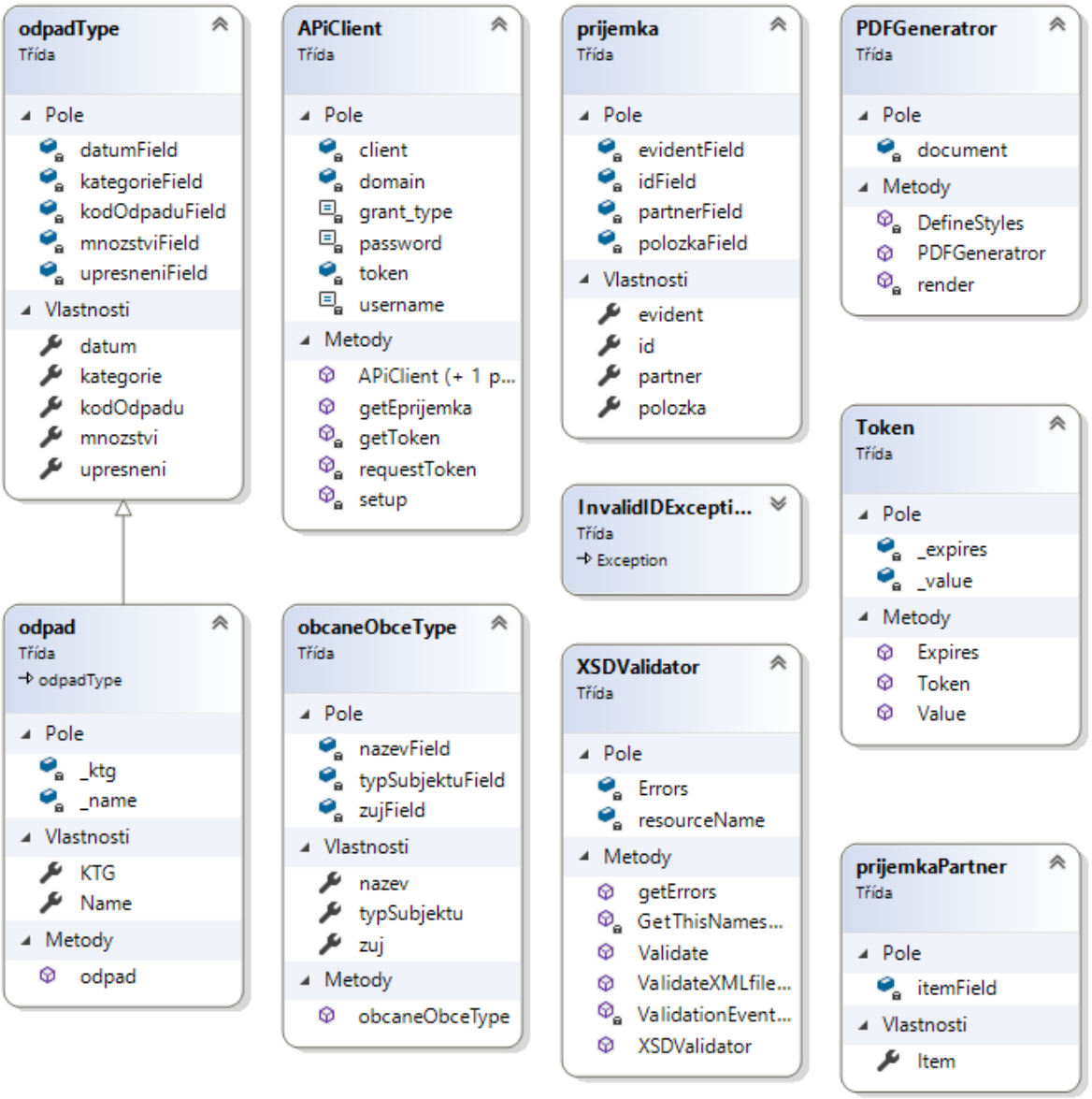
C. Vzorový XML soubor příjemky

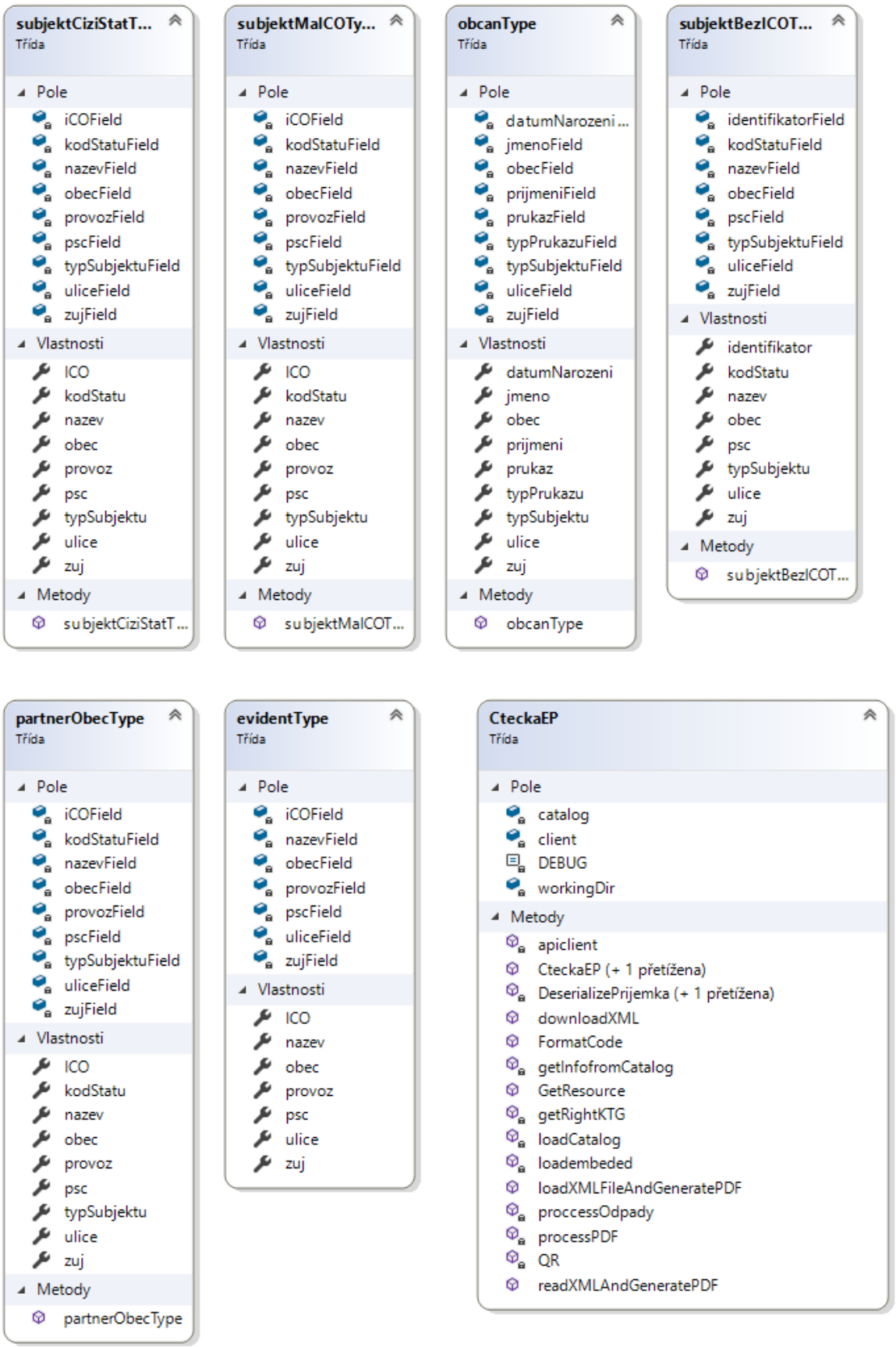
D. Volně vložené DVD se zdrojovými kódy

A. Příloha č. 1: Class diagram generátor.exe



B. Příloha č. 2: Class diagram čtečka.exe




subjektCiziStatT...
Třída
Pole
iCOField
kodStatuField
nazevField
obecField
provozField
pscField
typSubjektuField
uliceField
zujField
Vlastnosti
ICO
kodStatu
nazev
obec
provoz
psc
typSubjektu
ulice
zuj
Metody
subjektCiziStatT...
subjektMalCOTy...
Třída
Pole
iCOField
kodStatuField
nazevField
obecField
provozField
pscField
typSubjektuField
uliceField
zujField
Vlastnosti
ICO
kodStatu
nazev
obec
provoz
psc
typSubjektu
ulice
zuj
Metody
subjektMalCOT...
obcanType
Třída
Pole
datumNarozeni ...
jmenoField
obecField
prijmeniField
prukazField
typPrukazuField
typSubjektuField
uliceField
zujField
Vlastnosti
datumNarozeni
jmeno
obec
prijmeni
prukaz
typPrukazu
typSubjektu
ulice
zuj
Metody
obcanType
subjektBezICOT...
Třída
Pole
identifikatorField
kodStatuField
nazevField
obecField
pscField
typSubjektuField
uliceField
zujField
Vlastnosti
identifikator
kodStatu
nazev
obec
psc
typSubjektu
ulice
zuj
Metody
subjektBezICOT...
partnerObecType
Třída
Pole
iCOField
kodStatuField
nazevField
obecField
provozField
pscField
typSubjektuField
uliceField
zujField
Vlastnosti
ICO
kodStatu
nazev
obec
provoz
psc
typSubjektu
ulice
zuj
Metody
partnerObecType
evidentType
Třída
Pole
iCOField
nazevField
obecField
provozField
pscField
uliceField
zujField
Vlastnosti
ICO
nazev
obec
provoz
psc
ulice
zuj
CteckaEP
Třída
Pole
catalog
client
DEBUG
workingDir
Metody
apiclient
CteckaEP (+ 1 přetížena)
DeserializePrijemka (+ 1 přetížena)
downloadXML
FormatCode
getInfofromCatalog
GetResource
getRightKTG
loadCatalog
loadembeded
loadXMLFileAndGeneratePDF
proccessOdpady
processPDF
QR
readXMLAndGeneratePDF

C. Příloha č. 3: Vzorový XML soubor příjemky

```
<?xml version="1.0" encoding="UTF-8">
<prijemka xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <id>EDKZNWTAEUCFD79K</id>

    <evident>
        <ICO>25417657</ICO>
        <provoz>0</provoz>
        <nazev>Inisoft odpady s.r.o.</nazev>
        <ulice>Rumjancevova 696/3</ulice>
        <obec>Liberec</obec>
        <psc>46001</psc>
        <zuj>556904</zuj>
    </evident>

    <partner>
        <firma>
            <ICO>45678902</ICO>
            <provoz>0</provoz>
            <typSubjektu>F</typSubjektu>
            <nazev>Metal Produkt – Železná Drahomíra</nazev>
            <ulice>Žižkova 3b</ulice>
            <obec>Starý Plzenec</obec>
            <psc>33202</psc>
            <zuj>558371</zuj>
        </firma>
    </partner>

    <polozka>
        <odpad>
            <datum>11.02.2020</datum>
            <kodOdpadu>170405</kodOdpadu>
            <kategorie>O</kategorie>
            <mnozstvi>2.23</mnozstvi>
            <upresneni>Železo a ocel</upresneni>
        </odpad>

        <odpad>
            <datum>11.02.2020</datum>
            <kodOdpadu>170402</kodOdpadu>
            <kategorie>O</kategorie>
            <mnozstvi>0.7</mnozstvi>
            <upresneni>Hliník</upresneni>
        </odpad>
        <odpad>
            <datum>12.02.2020</datum>
            <kodOdpadu>170402</kodOdpadu>
            <kategorie>N</kategorie>
            <mnozstvi>0.7</mnozstvi>
            <upresneni>Hliník</upresneni>
        </odpad>
    </polozka>

</prijemka>
```