PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

# BAKALÁŘSKÁ PRÁCE

Informační systém pro zasílání elektronického nákladního listu v XML formátu

2010 Mário Malicher

Místopříseżně prohlašuji, že jsem celou bakařskou práci včetně příloh vypracoval samostatně.

17. srpna 2010 Mário Malicher

## Anotace

*Cílem bakalářské práce je vytvořit v jazyce C# aplikaci, zajišťující výměnu údajů elektronického nákladního listu (ENL) mezi dopravci. Aplikace bude pracovat jako informační systém zákazníka (ISZ) s rozhraním pro komunikaci s již existujícím informačním systémem provozu (ISP).*

Děkuji za pomoc a trpělivost vedoucímu bakalářské práce RNDr. Petru Sebestyénymu, Ph.D., své přítelkyni, rodině a kolegům.

# Obsah

# Seznam obrázků

# 1. Úvod

Cílem bakalářské práce je vytvořit v jazyce C# aplikaci, zajišťující výměnu údajů elektronického nákladního listu (ENL) mezi dopravci. Zdrojem a inspirací pro naprogramování této aplikace mi bylo v podstatě celé dosavadní studium, pak praxe z práce a internet. V programování mi také hodně pomohly knihy, které jsou zaměřené na prohloubení znalostí v oboru programování. Jedná se o tyto tituly [1], [2], [3], [4], [5]. Navíc jsem měl k dispozici od svého zaměstnavatele a společností s kterými spolupracujeme přesné zadání, XSD schémata a sekvenční diagramy podle kterých jsem aplikaci programoval. Verze pro bakalářskou práci je trochu modifikovaná, jinak by nefungovala. Ve skutečnosti je plně závislá na větším integrovaném systému společnosti OLTIS Group a.s., pro kterou jsem tuto aplikaci vytvořil.

## 1.1. Zadání bakalářské práce

- Aplikace bude pracovat jako informační systém zákazníka (ISZ) s rozhraním pro komunikaci s již existujícím informačním systémem provozu (ISP)
- Součástí ISZ bude knihovna, která vytáhne požadované data s již existující databáze a vytvoří z nich XML zprávu. Tyto data je možné vkládat a modifikovat pomocí jiné aplikace, která není součástí tohoto projektu
- XML zpráva, která bude obsahovat data nákladního listu, musí odpovídat schématům XSD
- Schémata XSD jsou vytvořené pro dopravní společnost využívající ISP k mezinárodní komunikaci. V bakalářské práci jsou tyto schémata mírně modifikovaná oproti realitě
- ENL bude posílán šifrovaný, s heslem známým oběma stranám a v komprimovaném formátu zip

## 1.2. Elektronický nákladní list

- Elektronicky uložené záznamy dat z nákladního listu
- Nákladní list je smlouva o přepravě zboží mezi dopravními společnostmi
- Obsahuje všechny náležitosti týkající se zásilky, která má být doručena z místa A na místo B, v tomhle případě, pomocí železniční dopravy
- Obsahuje, místo nakládky, vykládky, informace o odesílateli, přijímateli, výplatný záznam, informace o vozních, o materiálu a může obsahovat i jiné informace, které se týkají zásilky

# 2. Požadavky a řešení

## 2.1. Požadavky na informační systém

- Vytvořit informační systém pro zpracování ENL
- Zabezpečit komunikaci mezi tímto systémem a informačním systémem provozu
- Komunikace má probíhat pomocí rozhraní těchto systému
- U komunikace je XML zpráva zasílána emailem, jako příloha e-mailu s přesně danou strukturou názvu
- Je přesně daná i struktura názvu předmětu, včetně přesně dané struktury samotných XML zpráv
- V předmětu emailové zprávy se nachází i název služby, kterou bude daná zpráva chtít využit
- Komunikace je asynchronní
- Zprávy musí odpovídat již existujícím XSD schématům, které využívá ISP

## 2.2. Rozhraní

Rozhraní obou systému je přesně dané:

- **Interface ISZ**
  - OznamVyslSpracovania
  - Podaj
  - Dodaj
  - UkonceniePrepravy
  - AvizoPodaja
  - AvizoVstupu
  - AvizoDodaja
  - AvizoUkoncenia
  - avizoVystupu
  - SkoncenieKomunikacie
- **Interface ISP**
  - PredbeznyPodaj
  - OpravaDat
  - PotvrdeniePodaja
  - OznamDorucenia
  - OznamSuhlasuSPrevzatim

## 2.3. Řešení

- Pro vytvoření projektu jsem použil Visual Studio 2005, programovací jazyk C#, Microsoft SQL Server 2005 Express Edition, pomocí které jsem pro školní ukázku vytvořil lokální databázi

- Projekt jsem rozložil na více logických částí – knihoven, kvůli přehlednějším úpravám a opravám kódu

- Nejdřív jsem vytvořil knihovny EmailPop3 a ZipSifra a pak samotné jádro informačního systému ENLKlient

- ENLKlient se stane součástí většího integrovaného systému, který je již využíván v praxi jako logistický systém

- Součástí projektu jsou ještě tyto knihovny: CreateENLXML, IsdlENLKlient, CommonMario a další pomocné knihovny, které jsou zapouzdřené pomocí již vzpomínaných knihoven

- V projektu jsem použil i dvě již existující knihovny, kterých nejsem autorem: smtPop.dll [6], slouží pro komunikaci s poštovním serverem a ICSharpCode.SharpZipLib.dll [7] slouží pro komprimaci a heslování

- GUI (Grafické uživatelské rozhraní)
  Je tvořené dialogy s tabulkami, v kterých se nacházejí základní informace o zásilkách, pak dialogy s komentáři příchozích zpráv, případně volbou rozhraní pro zprávy na odeslání. Rozhraní je tvořené i dialogy s informacemi o případných technických problémech, o nekonzistentních vstupních údajích apod.

- Použité nástroje
  –Microsoft Visual Studio 2005
  –Programovací jazyk C#
  –Microsoft .NET Framework 2.0
  –Microsoft .NET Framework 3.5
  –Microsoft SQL Server 2005 Express Edition
  –MS Windows XP

# 3. Instalace a systémové požadavky

## 3.1. Popis instalace a spuštění programu

Projekt se instaluji pomocí instalačního balíčku InstallENLKlient. Všechny potřebné komponenty včetně lokální databáze se nainstalují do uživatelem zvoleného adresáře. // V realitě je aplikace komponentou, která se spustí zavoláním souboru ENLKlient.exe se vstupními parametry přes příkazový řádek. O tohle vše se postará část integrovaného systému, která využívá tento informační systém jako komponentu. // Pro školní ukázku je tento EXE soubor upraven, tak aby na něj stačilo dva-krát kliknout a aplikace se spustí. Některé vstupní parametry je možné zadat pomocí formuláře, který se zobrazí jako dialog pro komunikaci s uživatelem.

## 3.2. Systémové požadavky

Pro funkčnost aplikace je zapotřebí operační systém Windows XP s .net frameworkem 3.5 a Microsoft SQL Server 2005 Express Edition, kvůli funkčnosti lokální databáze. Navíc procesor o frekvenci minimálně 2 GHz a operační paměť 2 GHz. Aplikace byla testována jenom na takhle výkonném počítači, takže není vyloučené, že bude fungovat i na méně výkonném počítači.

# 4. Uživatelská dokumentace

## 4.1. Popis programu

Program slouží pro komunikaci mezi dvěma informačními systémy – ISProvozu (ISP) a ISZakaznika (ISZ). V případě zákazníka se může jednat o odesílatele zboží, nebo přijímatele.

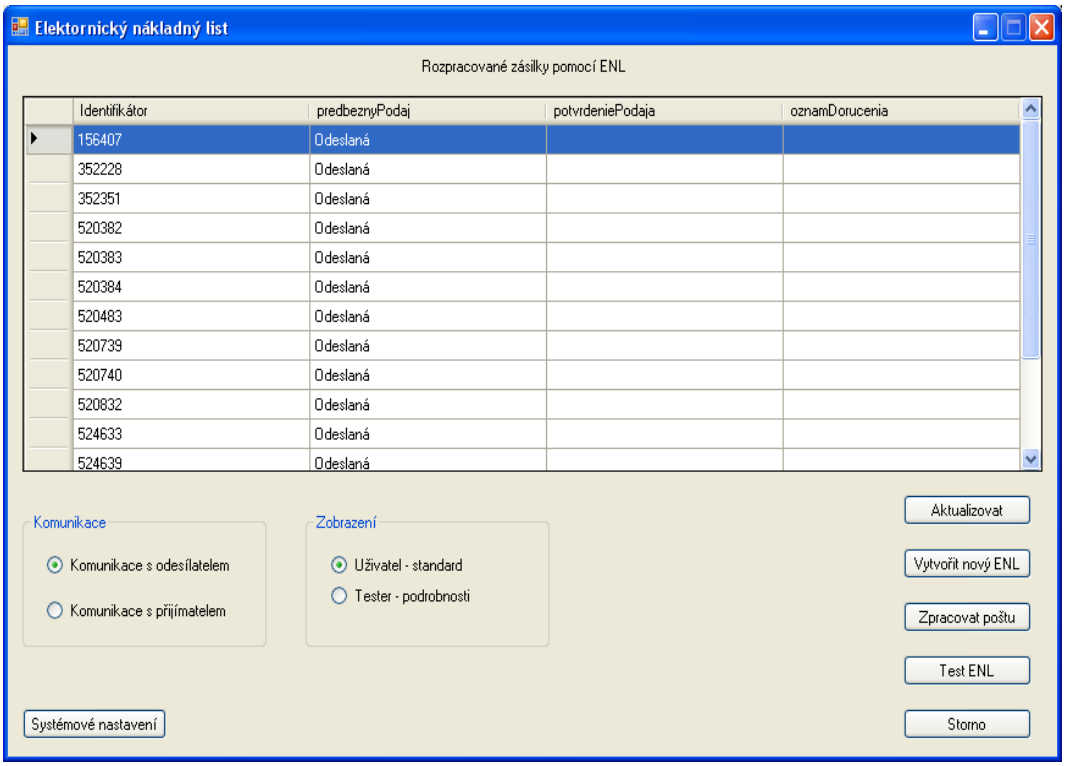
## 4.2. Ovládání Informačního systému zákazníka (ENLKlient)

Před samotným spuštěním aplikace je zapotřebí projít kapitolu o instalaci a požadavcích na konfiguraci počítače. Po úspěšném nainstalování všech potřebných technologií a samotné aplikaci najdete v souboru: readme.txt, popis jak nakonfigurovat tento systém aby fungoval i ve Vašem prostředí. Pak už stačí spustit ENLKlient.exe soubor a aplikace se spustí. Pomocí této aplikace můžete zasílat elektronický nákladní list (ENL) do informačního systému zákazníka (ISZ). Jako první okno se zobrazí hlavní dialog (viz. Obr. 1.). V tomto dialogu se nachází tabulka se seznamem rozpracovaných ENL. Navíc u každé zásilky jsou další sloupce, v kterých je uvedeno, jestli již byl konkrétní druh zprávy odeslán nebo nikoli. Jedná se o zprávy, které jsou určené pro odeslání do ISP. Není tady zobrazena zpráva „opravaDat“. Tato zpráva se může opakovat vícekrát, případně se nemusí vůbec v komunikaci objevit. Z těchto důvodů je vynecháno její zobrazení. Po dvoj-kliknutí na některý řádek tohoto seznamu, Vás aplikace bude informovat o aktuálním stavu, konkrétního nákladního listu. V případě, že se bude v databázi už nacházet nezpracovaná zpráva z ISP určená pro vybranou zásilku, bude možné ji velice jednoduše a rychle zpracovat. V levé části dialogu pod tabulkou se nabízí možnost nastavení komunikace. Záleží na tom, jestli jste odesílatel nebo přijímatel zakázky, na kterou má být vystavený ENL. Vedle tohoto nastavení je možnost zvolit zobrazení. V případě, že se rozhodnete pro zobrazení s podrobnostmi, uvidíte i dialogy, které slouží pro testovací účely. Nemají žáden vliv na funkčnost a můžete je klidně při čtení přeskočit.

V tomto dialogu (viz. Obr. 1.) se nachází níže popsané tlačítka:

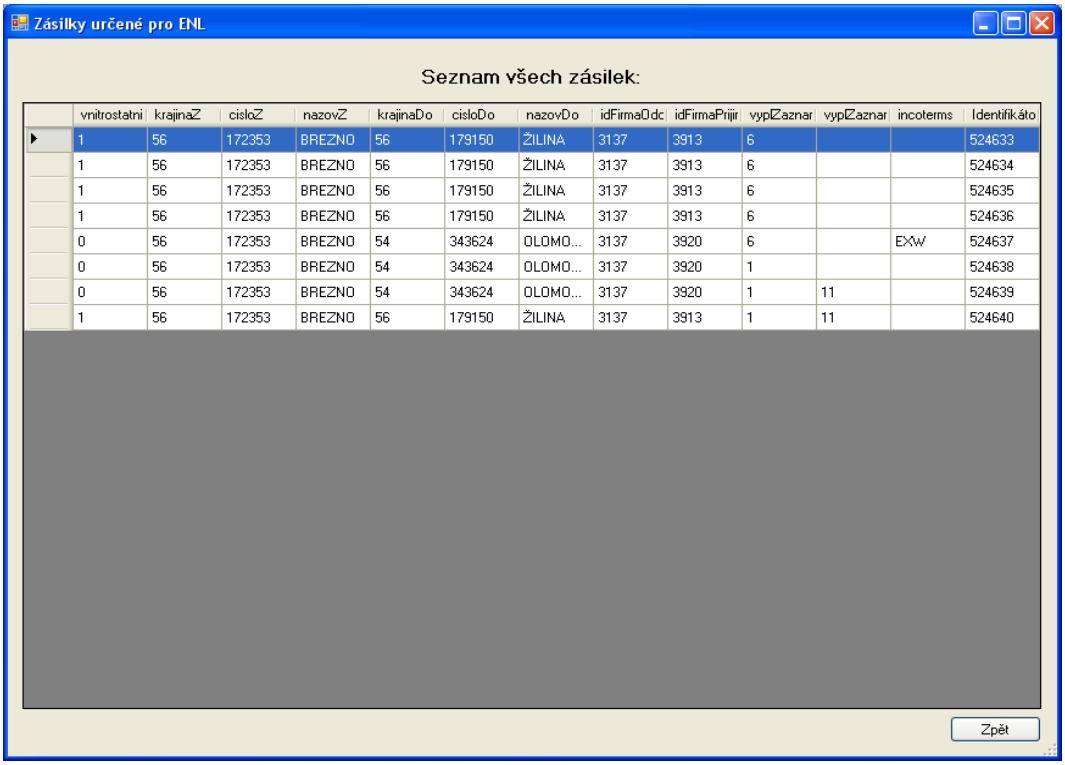
- **Systémové nastavení**
  Pomocí této volby je možné nastavit některé důležité části programu. Jedná se např. o nastavení poštovního serveru, jak pro příchozí tak odchozí poštu. Pak je tady možné nastavit příchozí i odchozí email, jejich hesla, heslo samotné zprávy, adresáře kam se mají ukládat zpracované zprávy. Bližší informace k této konfiguraci najdete v souboru readme.txt, který se nachází na v adresáři Release.

- **Aktualizovat**
  Aktualizuje se tabulka se zásilkami. Načtou se aktuální data z databáze.




Obrázek 1. Hlavní okno aplikace

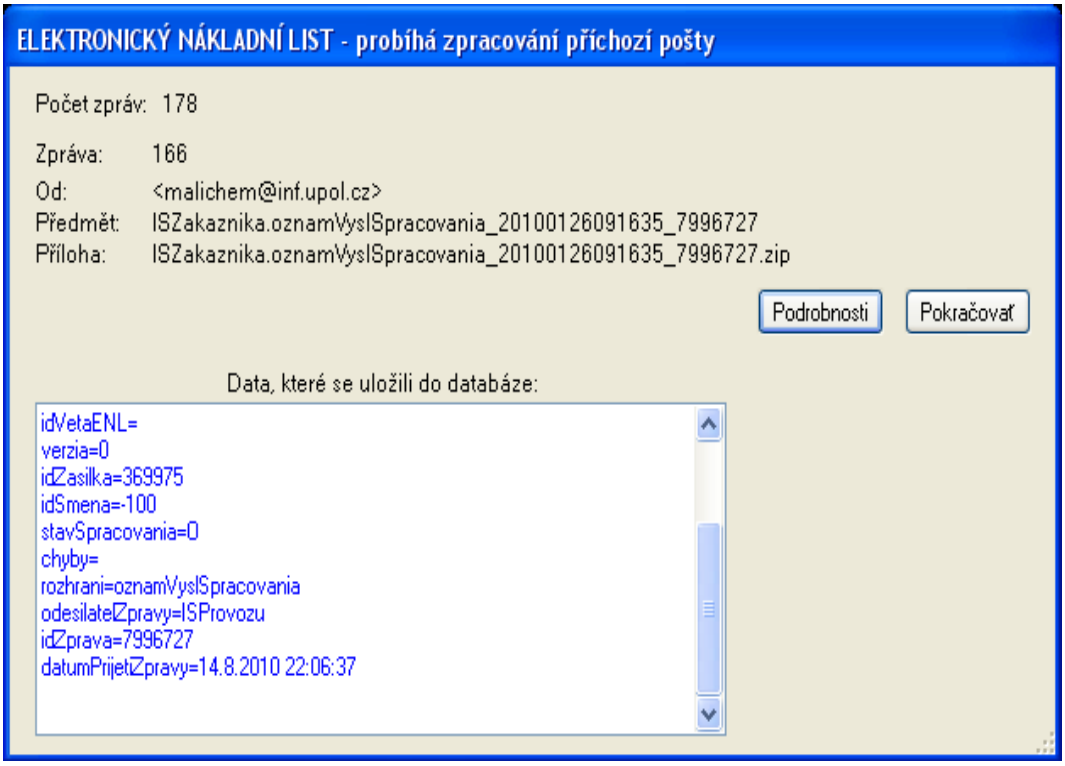
- **Vytvořit nový ENL**
  Otevře se další dialog viz. obr. 2 se zásilkami, z kterých je možné vytvořit zprávu určenou pro komunikaci s ISP. Dvojklikem na levé tlačítko myši se aktivuje vytvoření ENL s následnou možností odeslat tuto zprávu do ISP. Na každou zásilku je možné vytvořit jenom jeden ENL.




Obrázek 2. Výběr zásilky pro komunikaci pomocí ENL

- **Zpracovat poštu**
  Toto tlačítko slouží pro zpracování příchozí pošty. Aplikace projde celý poštovní server určený pro příjem elektronických nákladních listů. V případě, že v něm najde zprávu určenou pro ISZ odeslanou s předem domluveného poštovního serveru IS Provozu (ISP), tak je tahle zpráva zpracovaná. Zpracovaný je předmět zprávy, dále je zpracovaná zašifrovaná a za-heslovaná příloha, která obsahuje XML zprávu, určenou pro konkrétní rozhraní ISZ. Nejdůležitější data pro identifikaci zprávy, které obsahuje XML zpráva, jsou automaticky uložené v databázi do tabulky ENLStatus. Celá zpráva je současně uložená do adresáře, který je určený pro zprávy z ISP. Zpráva je uložena zašifrovaná, za- heslovaná a taky rozšifrovaná a bez hesla. V případě, že se na poštovním serveru nenachází žádný email, zobrazí se dialog s touto informací: „V emailové schránce se nenachází žádný email od ISP“. V opačném případě se zobrazí následující okno (viz. Obr. 3.). Po stisku

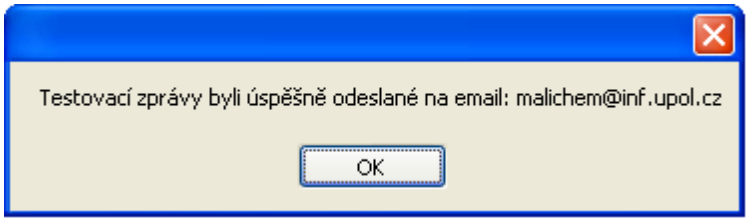
tlačítka Podrobnosti v tomto okně se navíc zobrazí data, která se uložila do databáze, jak je to vidět na obrázku. Pomocí tlačítka Pokračovat se vrátíte do hlavního menu.




Obrázek 3. Zpracování pošty

- **Test ENL**
  Toto tlačítko slouží na prověření funkčnosti celé aplikace. Před kliknutím na toto tlačítko je zapotřebí vytvořit první zprávu pro komunikaci a to „predbeznyPodaj“ pro konkrétní zásilku s identifikátorem = 369975. Postup pro vytvoření této zprávy se nachází v podkapitole 4.3.. Po vytvoření této zprávy je zapotřebí kliknout na tlačítko Test ENL. To způsobí, že budou na poštovní server pro příchozí poštu odeslány testovací zprávy, které již v minulosti byly postupně doručovány z ISP do ISZ jako reakce na posílání testovacích zpráv z ISZ do ISP pro zásilku s id = 369975. Tyto zprávy byly archivovány pro další testování v případě větších změn v systému. Komunikace je ve skutečnosti asynchronní a může klidně trvat více dnů. Do komunikace zasahují lidi jak na jedné tak na druhé straně. Díky tomuto shromážděnému balíčku zpráv je možné celou komunikaci předvést během pár minut. Komunikace neobsahuje dvě zprávy: „avizoVystupu“ a „avizoVstupu“. Tyto dvě zprávy mají jenom informační charakter a v podstatě téměř stejný význam jako zprávy: „avizoUkoncenia“a „avizoPodaja“.

Jsou to zprávy generované z ISP. V případě, že odeslání zpráv proběhne v pořádku, zobrazí se tento informační dialog (viz. Obr. 4.). Jestli má vše probíhat podobně jako ve skutečnosti, tak teď byste měli kliknout na tlačítko „Zpracovat poštu“. Všechny zprávy budou zpracovány a data o nich uložena do databáze. Pak stačí označit v tabulce s rozpracovanými zásilkami ENL testovací zásilku s identifikátorem 369975 a dvojklikem na levé tlačítko myši spustit další zpracování této zásilky. Celá obsluha komunikace je popsána všeobecné dále v návodě.

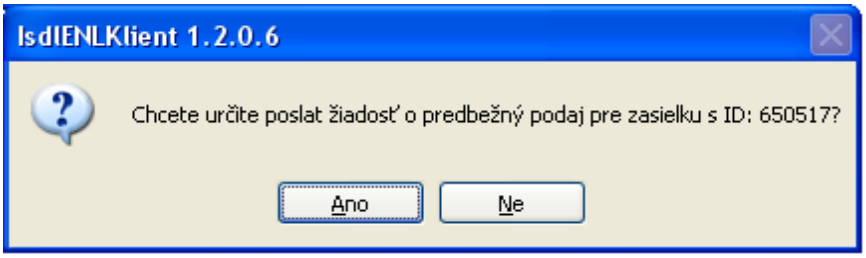



Obrázek 4. Oznámení úspěšného odeslání emailu

- **Storno**
  Po kliknutí na toto tlačítko se aplikace ukončí.

## 4.3. Vytvoření zprávy ”predbeznyPodaj”

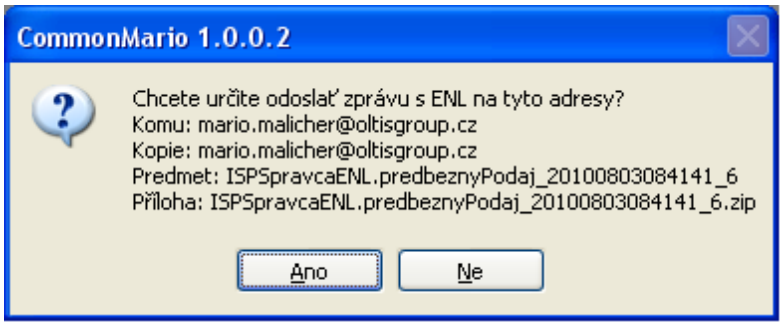
Po dvojkliku na levé tlačítko myši, na jakoukoli zásilku, která se nachází v tabulce se seznamem všech zásilek v dialogu pro vytváření nového ENL (viz. Obr. 2.) se zobrazí dialog (viz. Obr. 5.) s dotazem jestli chcete určitě poslat žádost o předběžný podej pro zásilku s konkrétním ID. V případě souhlasu a existence konkrétní zásilky v databázi se vytvoří první zpráva pro komunikace ENL – “predbeznyPodaj”.




Obrázek 5. Předběžný podej

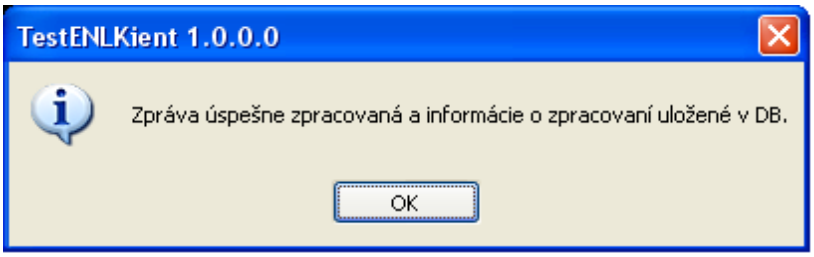
Následně proběhne kontrola tabulky ENLStatus v databázi, jestli už náhodou nedošlo k žádosti o předběžný podej pro konkrétní zásilku. Jestli ne tak se začne vytvářet XML zpráva určená pro předběžný podej.
Následuje dotaz, jestli chcete určitě odeslat email s vytvořenou zprávou (viz. Obr.

6.). Tento dialog se zobrazí vždy těsně před odesláním jakékoli zprávy z ISZ do ISP.




Obrázek 6. Odeslání emailu

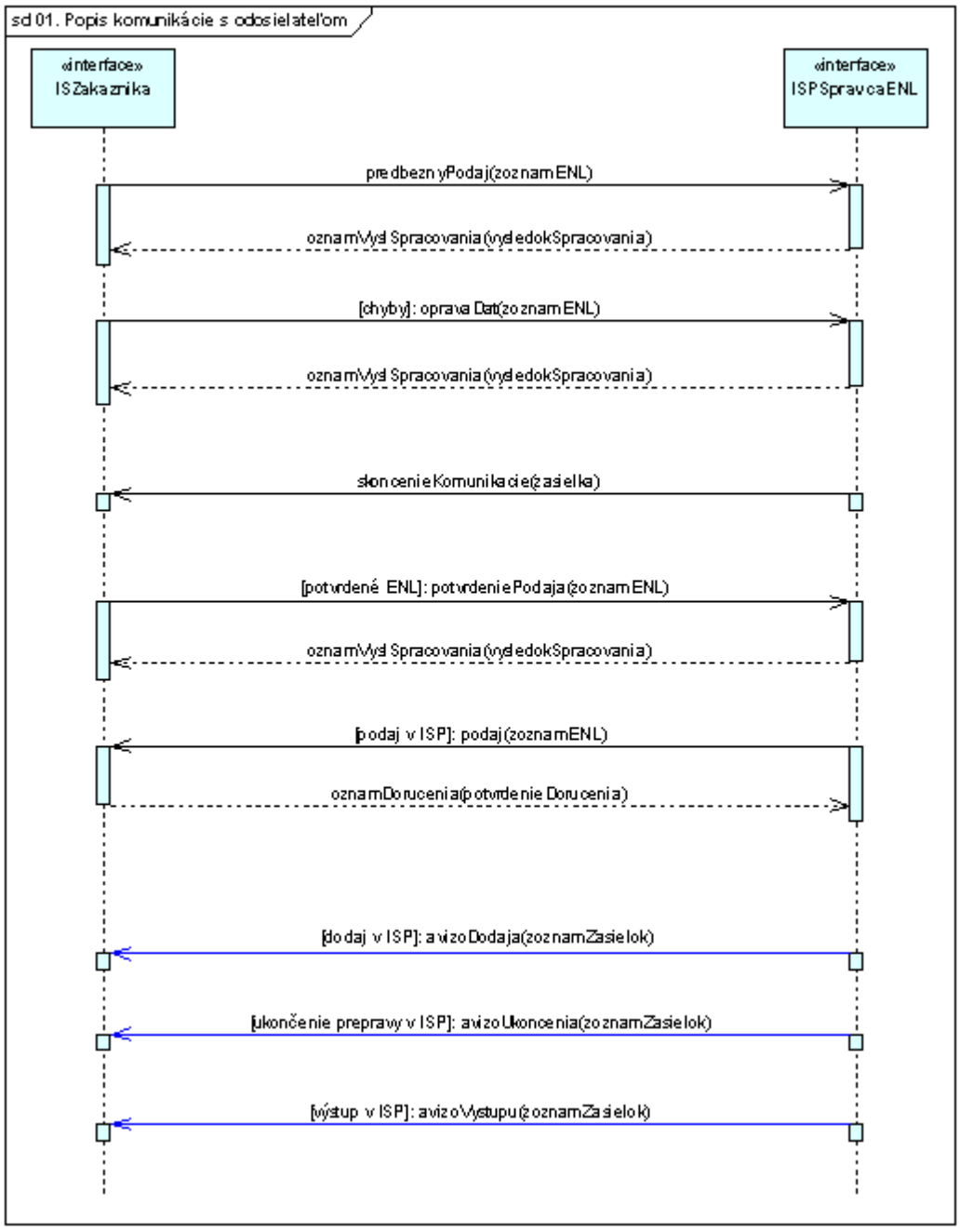
Pak se zobrazí zpráva o úspěšném zpracování a uložení zprávy v DB (viz. Obr. 7.).



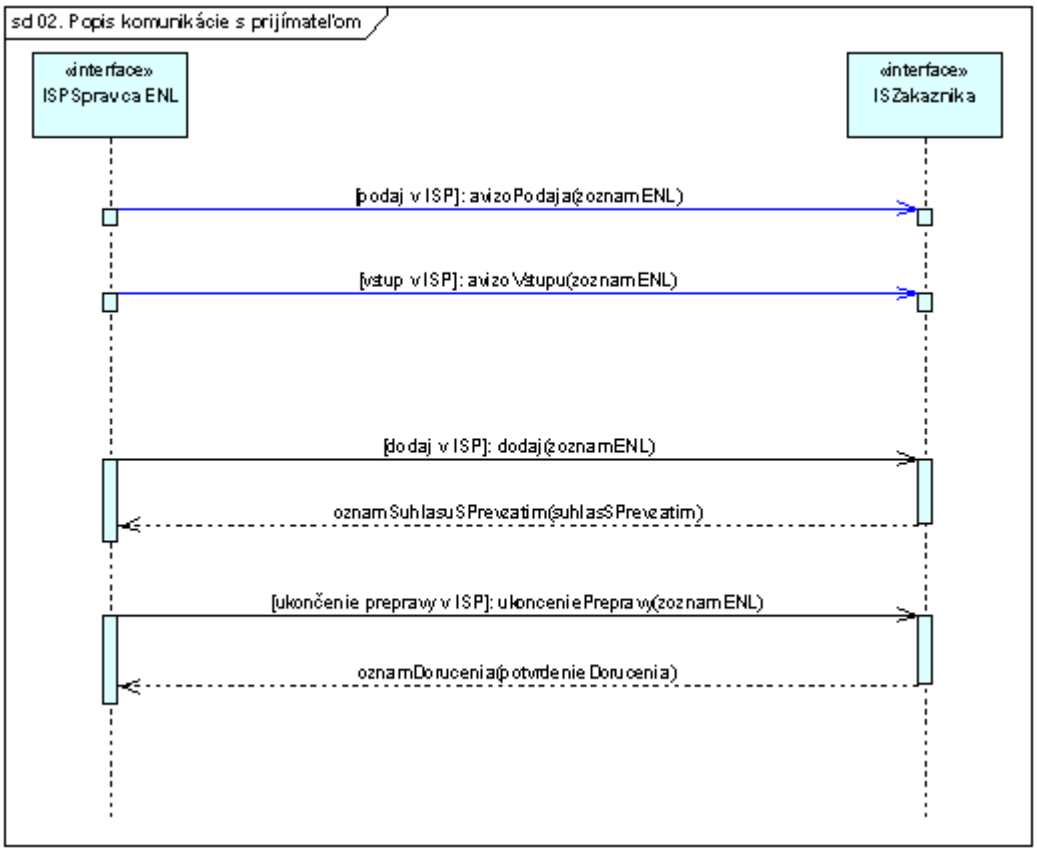

Obrázek 7. Informace o úspěšném zpracování zprávy

Tohle byla první zpráva určená pro začátek komunikace pomocí ENL – „predbeznyPodaj“.
Všechny zprávy, které se používají pro komunikaci, jsou znázorněny na sekvenčních diagramech, které byli součástí zadání (viz. Obr. 8.a 9.).

Zpráva „opravaDat“ a odpověď na ní „oznamVyslSpracovania“ se můžou opakovat, v případě chyb v ENL. V případě, že je vše v pořádku tak tyto dvě zprávy se vůbec nezasílají.
Zpráva „skoncenieKomunikacie“ může nastat kdykoliv. Např. v případě nefunkčnosti IS zákazníka a podobně.




Obrázek 8. Popis komunikace s odesílatelem



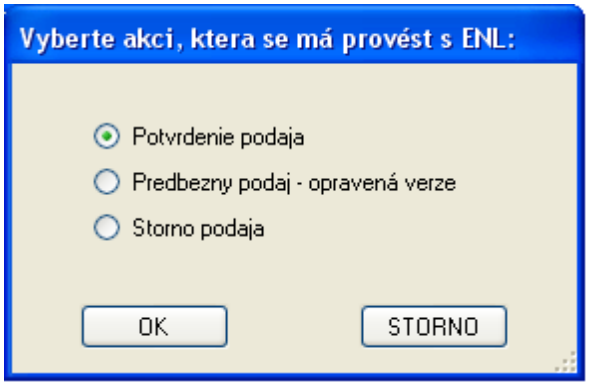

Obrázek 9. Popis komunikace s přijímatelem

## 4.4. Vytvoření dalších zpráv

Při vytváření a odesílání dalších zpráv z ISZ se obsluhují následující jednoduché dialogy.

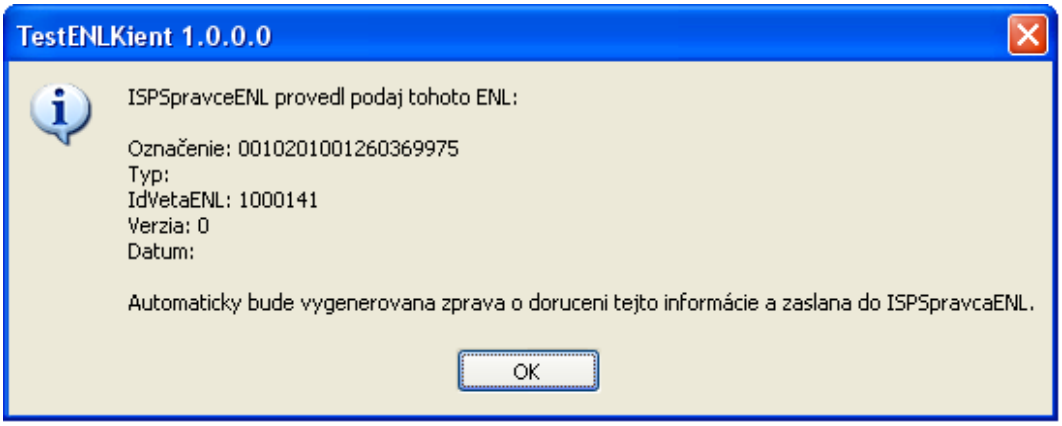
**Při komunikaci, kde je ISZ (ENLKlient) nasazen u odesílatele zakázky:**

Pomocí následujícího dialogu (viz. Obr. 10.) je možné odeslat tyto zprávy: potvrdeniePodaja, opravaDat, případně stornovat předběžný podej. Dialogy, které umožňují vytvoření zprávy určené pro odeslání z ISZ, se zobrazují v pořadí, podle sekvenčních diagramů (viz. Obr. 8.a 9.). Pořadí je tedy závislé na odpovědích z ISP.




Obrázek 10. Výběr vytvoření nové zprávy pro komunikaci pomocí ENL

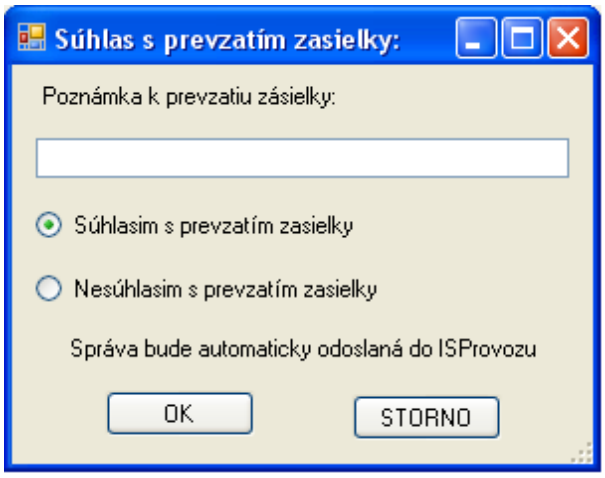
V dalším dialogu (viz. Obr. 11.) je informace z ISP o provedení podeje a informace o tom, že bude automaticky vygenerovaná zpráva „oznamDorucenia“. Což je vlastně i poslední zpráva od odesílatele.




Obrázek 11. Oznámení podeje a automatické odeslání odpovědi

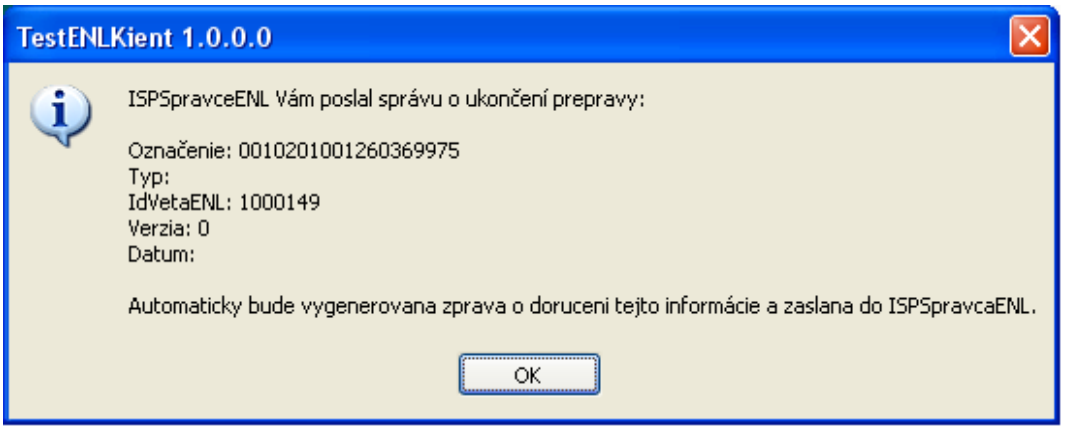
**Při komunikaci, kde je ISZ (ENLKlient) nasazen u přijímatele zakázky:**

Pomocí následujícího dialogu (viz. Obr. 12.) odešlete zprávu „oznamSuhlasuSPrevzatim“. Poznámka v dialogu je nepovinná.



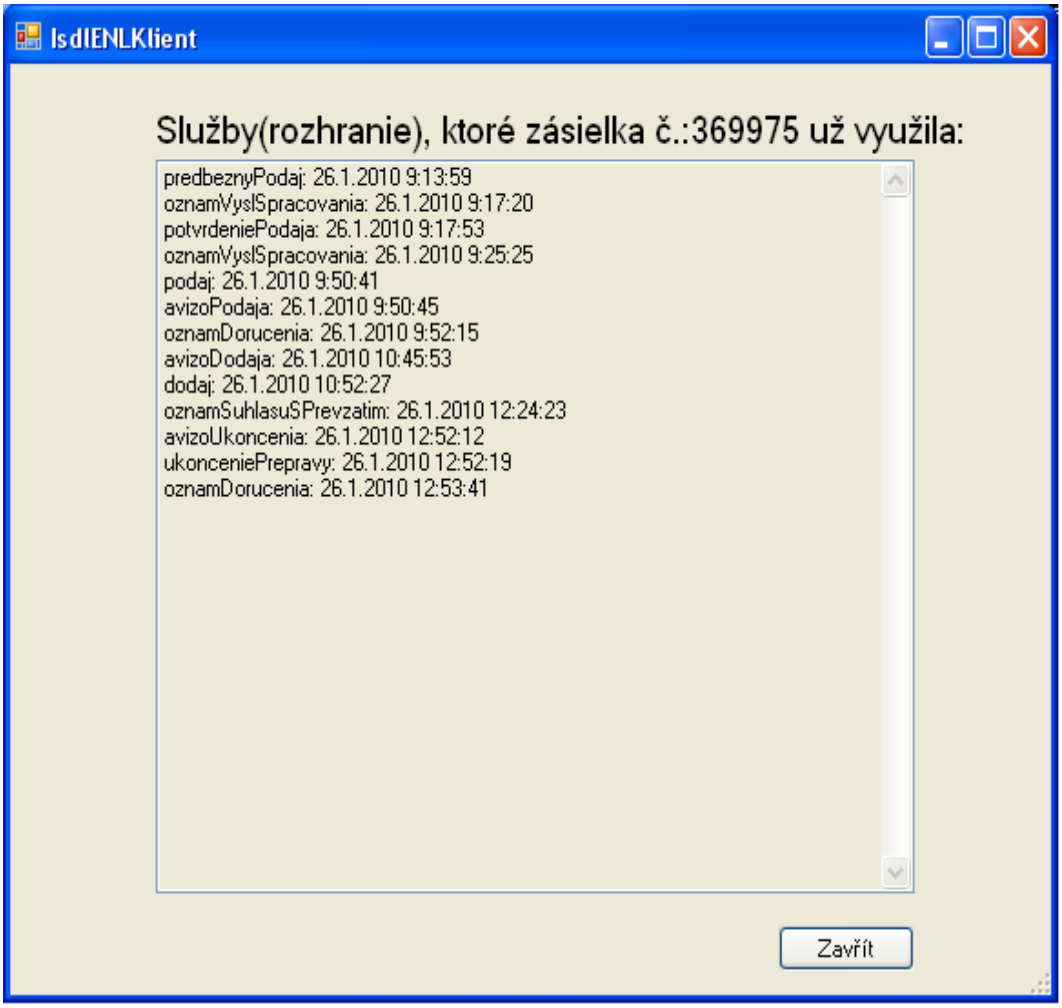

Obrázek 12. Souhlas s převzetím zásilky

A nakonec ještě pomocí dalšího dialogu (viz. Obr. 13.), který informuje o ukončení přepravy se automaticky odešle zpráva o doručení do ISP.




Obrázek 13. Oznámení o ukončení přepravy a automatické odeslání odpovědi

Komunikace ze strany ISZ je tím ukončena.
U obou komunikací, po dvojkliku na levé tlačítko myši na jakoukoli zásilku v okně (viz. Obr. 1.), která ještě nedostala požadovanou odpověď od ISP, se zobrazí následující dialog (viz. Obr. 14.). V okně jsou vypsané všechny zprávy, které konkrétní zásilka již využila.




Obrázek 14. Seznam zpráv, které zásilka již využila

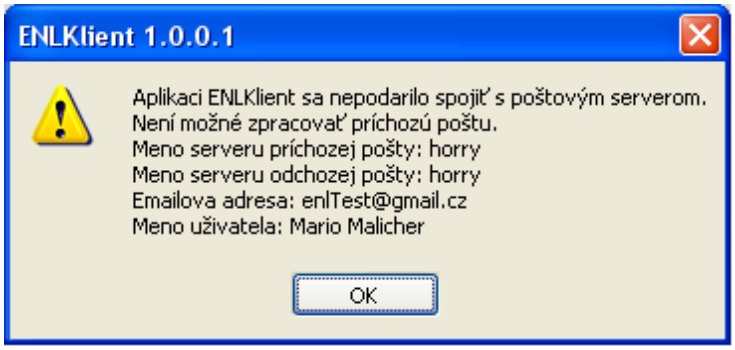
## 4.5. Odpovědi z ISP (Informační systém provozu)

Odpovědi z ISP mají informační charakter. Samozřejmě vše se ukládá do databáze. Avšak obsluha je úplně triviální. To znamená, že si jenom přečtete informaci a kliknete na tlačítko OK. Případnou opravu dat je možné vykonat v aplikaci Zásilky, přesně tak jak se to dělalo do teď. A pak se opět vrátit k ENL, kde bude možné odeslat opravu dat. Aplikace Zásilky je součástí již vzpomínaného integrovaného systému využívaného komerčně již v praxi. Není tedy součástí tohoto projektu.

## 4.6. Další dialogy

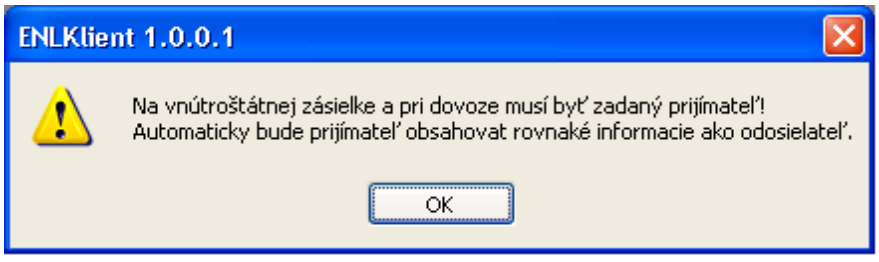
**Tyto dialogy se můžou zobrazit v případě, že by nastal nějaký technický problém:**

Nefunguje poštovní server (viz. Obr. 15.):



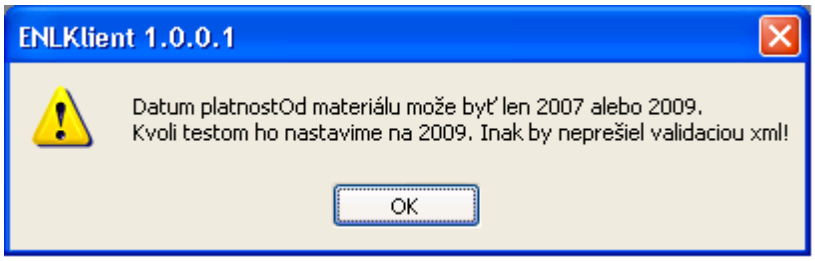

Obrázek 15. Není možné navázat spojení s poštovním serverem

Chybějí data v databázi (viz. Obr. 16.):




Obrázek 16. Není zadaný přijímatel

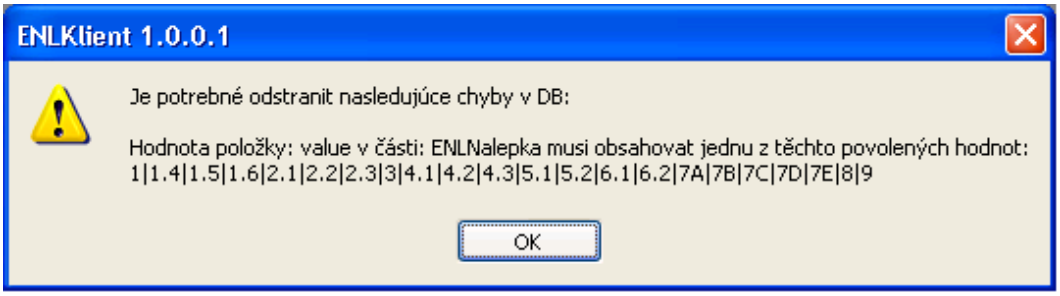
Nepovolená data v databázi (viz. Obr. 17.):
I když nastanou tyto situace, je možné ENL zpracovat a odeslat.



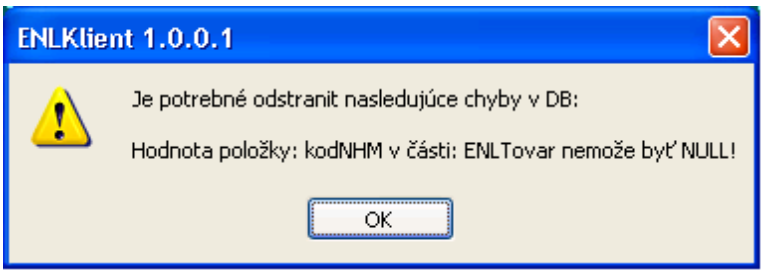

Obrázek 17. Datum platnosti materiálu je mimo povolenou hodnotu

**Dialogy upozorňující na nekonzistentní data určené pro ENL:**

Avšak když se zobrazí následující dialogy (viz. Obr. 18. a 19.), musíte pak provést úpravy v Zásilce u konkrétní položky. Opět se jedná o již vzpomínanou aplikaci Zásilky.




Obrázek 18. Nekonzistentní data v databázi

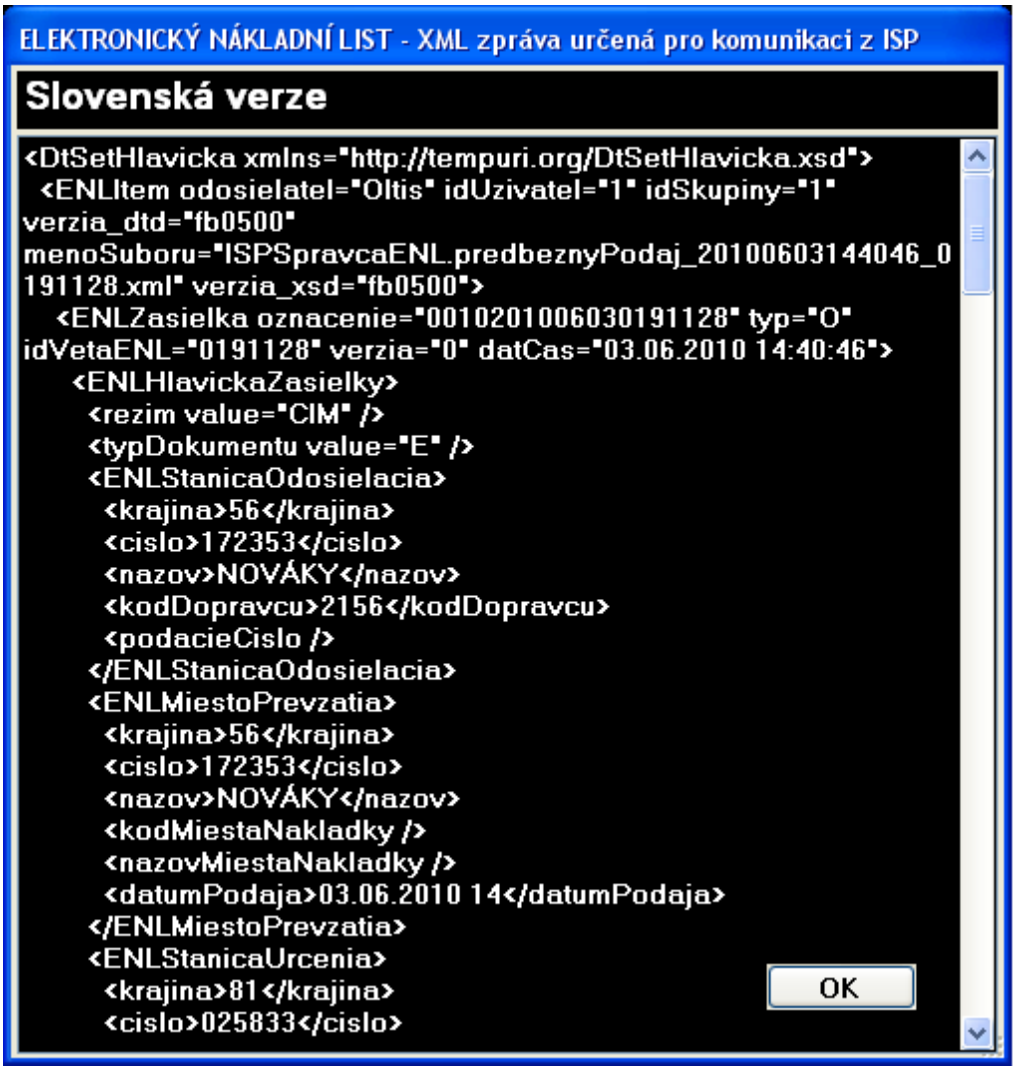



Obrázek 19. Nepovolená vstupní hodnota pro vytvoření ENL

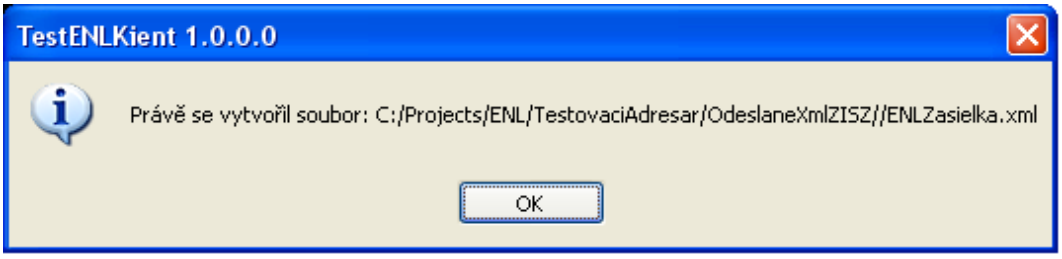
Atd.

**Dialogy s podrobnostmi:**

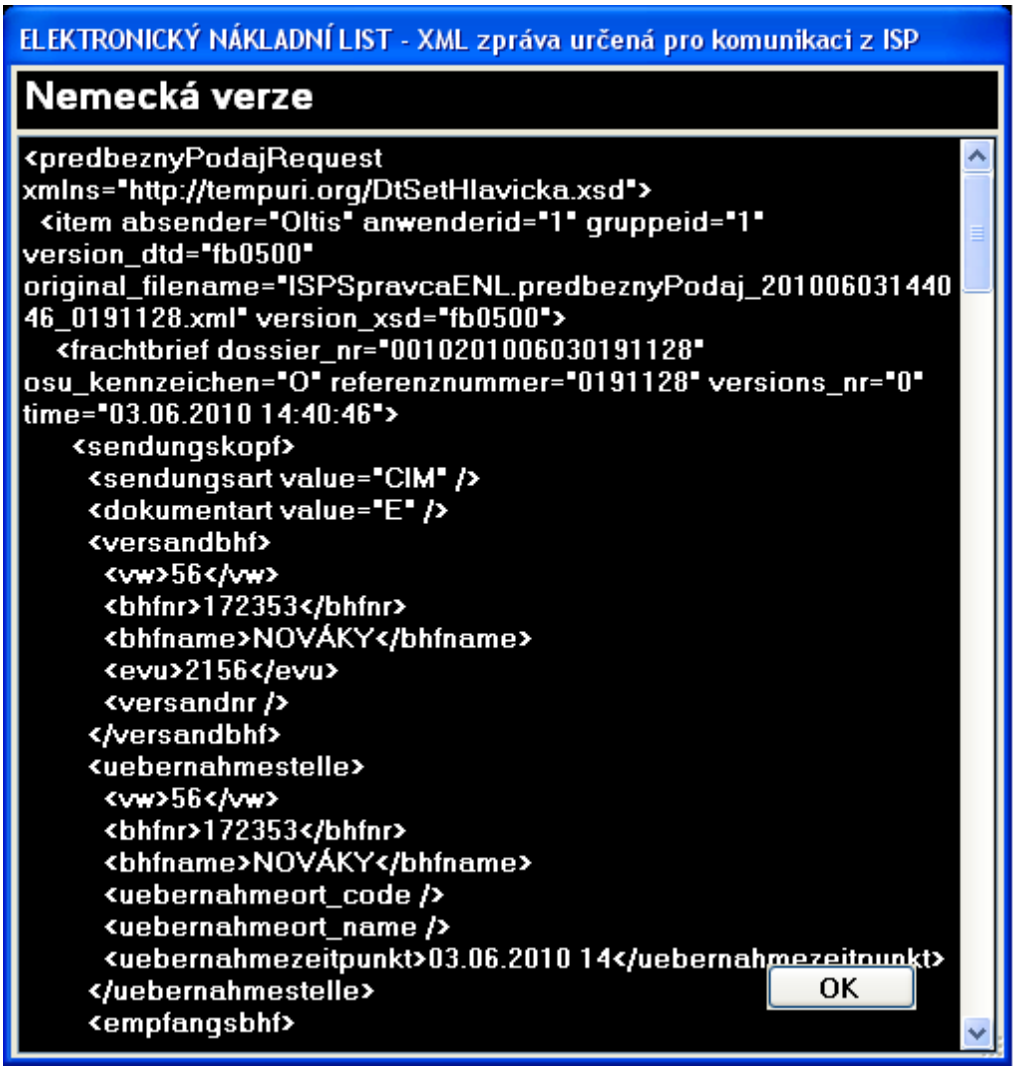
Dialogy (viz. Obr. 20., Obr. 21., Obr. 22., Obr. 23., Obr. 24., Obr. 25., Obr. 26.), které se budou zobrazovat u vytváření všech XML zpráv v případě, že zvolíte v hlavním dialogu [Obrázek 1] zobrazení „Tester – podrobnosti“.



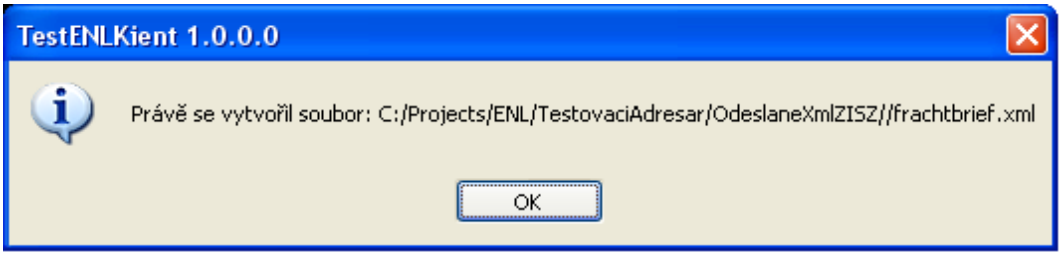

Obrázek 20. XML zpráva v slovenském jazyku



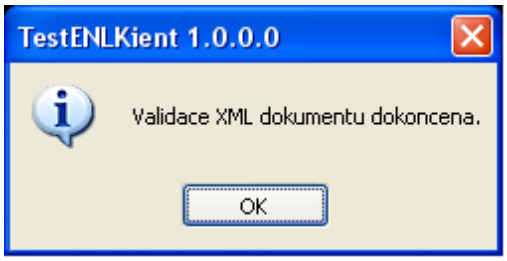

Obrázek 21. Dočasné místo uložení XML zprávy v slovenském jazyku



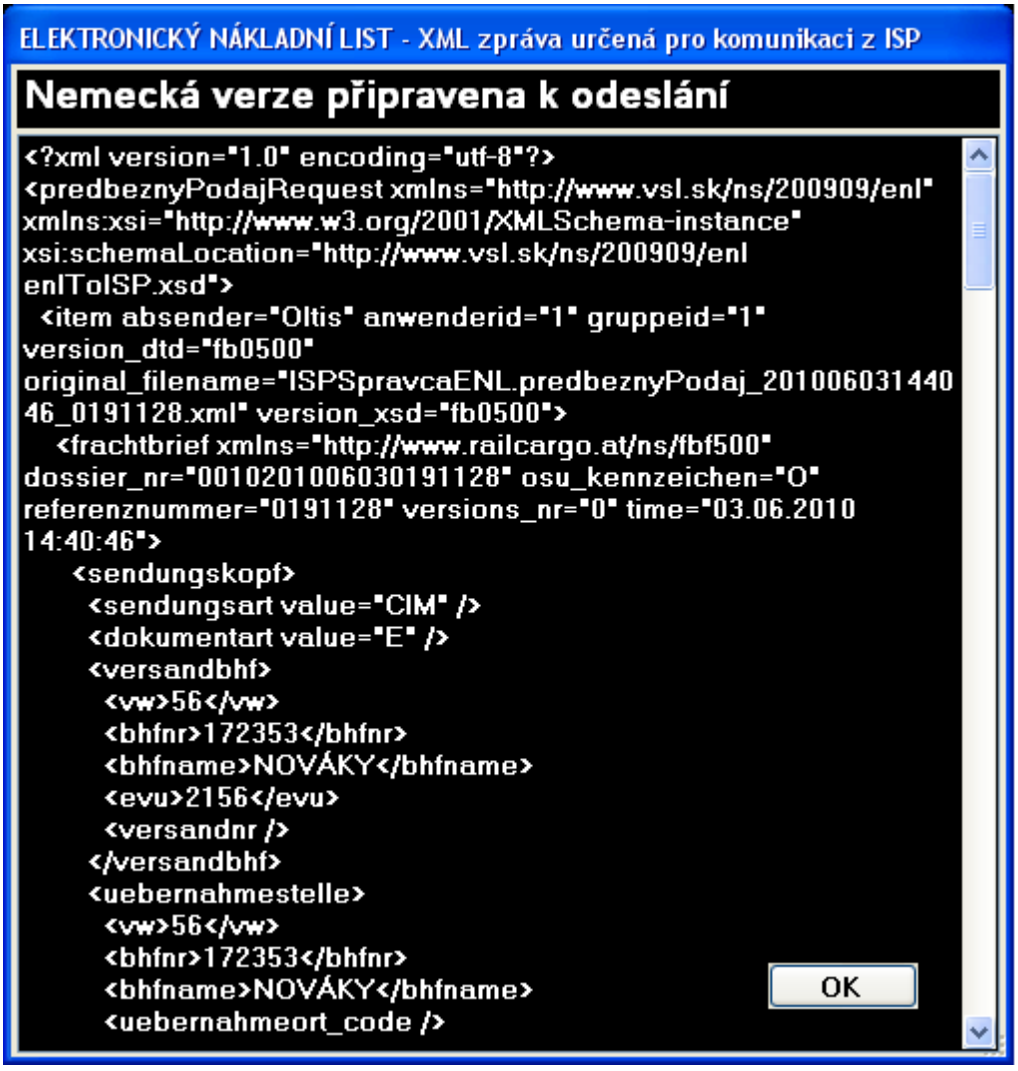

Obrázek 22. XML zpráva v německém jazyku



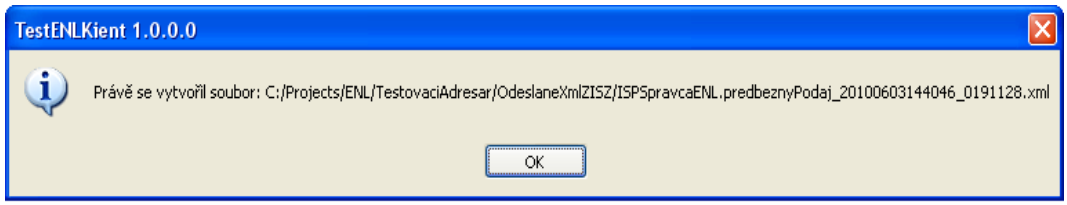

Obrázek 23. Dočasné místo uložení XML zprávy v Německém jazyku




Obrázek 24. Validace XML dokumentu úspěšně dokončena




Obrázek 25. XML zpráva v Německém jazyku připravena k odeslání




Obrázek 26. Trvalé místo uložení XML zprávy připravené k odeslání v Německém jazyku

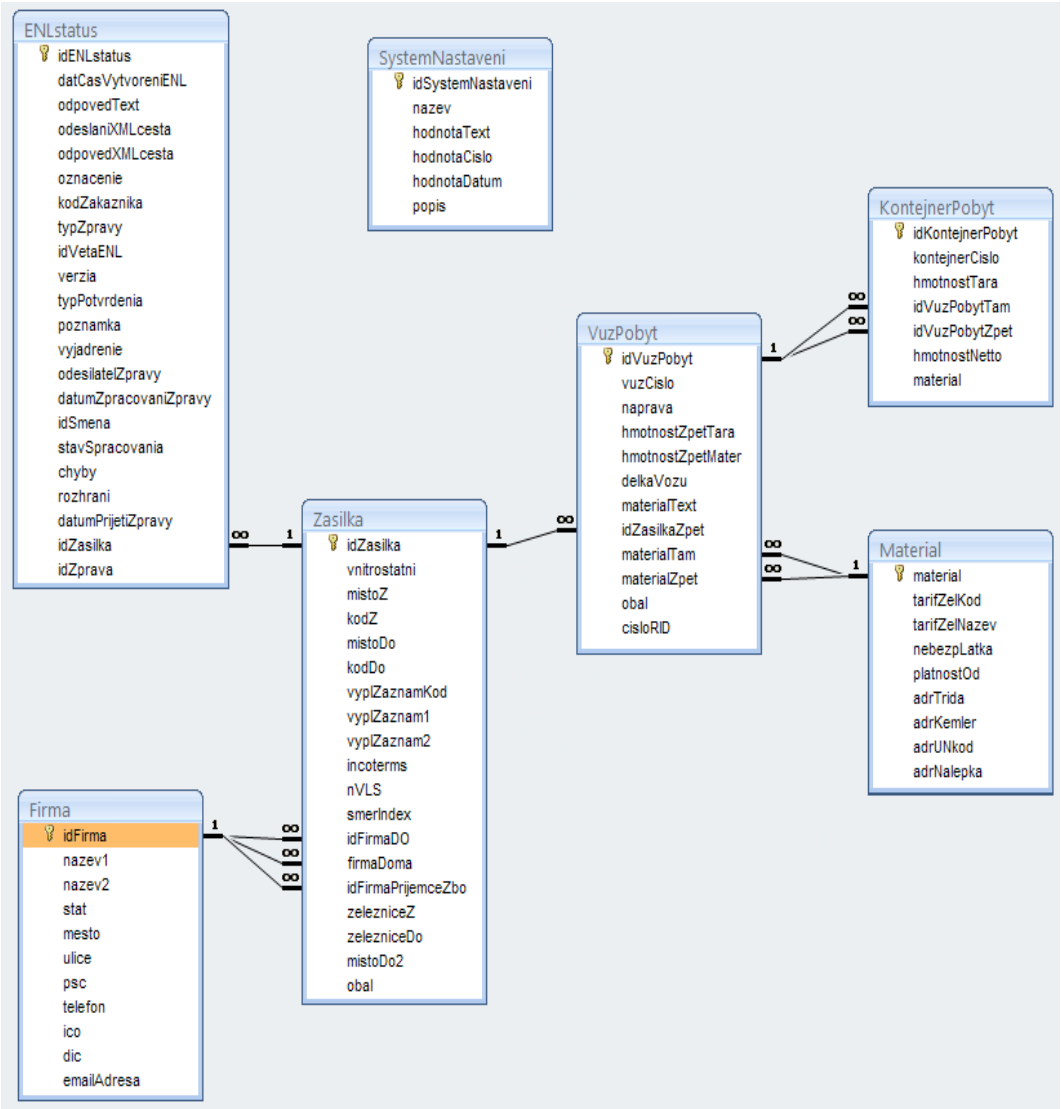
# 5. Programátorská dokumentace

Tato kapitola popisuje podrobně způsob řešení. Je rozdělena do více částí. Aplikace se skládá z více knihoven, které mezi sebou navzájem komunikují a z hlavního souboru typu exe. Jedná se o komponentu, která bude využívaná jako informační systém zákazníka (ISZ) pro komunikaci s již existujícím informačním systémem provozu (ISP). Všechny části aplikace jsou popsány v následujících podkapitolách. Systém již byl několikrát testován. Avšak ještě nebyl nasazen u zákazníka. V kódu jsou některé částí „zapoznámkované“, kvůli možné modifikaci na základě požadavků zákazníka.
V podkapitole 5.1 se nachází popis databáze, kterou využívá aplikace. V podkapitole 5.2 je popsán spouštěcí soubor ENLKlient.exe, který může být volán se vstupními parametry, nebo bez nich. V podkapitole 5.3 je popsána knihovna IsdlENLKlient.dll. Tato knihovna je zapouzdřena souborem ENLKlient.exe. V podkapitole 5.4 je popsána knihovna EmailPop3.dll, která zapouzdřuje již existující knihovnu smtpPOP3.dll [7]. V další podkapitole 5.5 je popis knihovny CreateENLXML.dll, která slouží pro vytvoření, překlad a validaci odchozí XML zprávy. V podkapitole 5.6 je stručný popis knihovny ZipSifra.dll. Zapouzdřuje knihovnu ICSharpCode.SharpZipLib.dll [8], sloužící pro komprimování, zašifrování a „zaheslování“ odchozí XML zprávy a taktéž dekomprimování, rozšifrování a „odheslování“ příchozí XML zprávy. Pomocná knihovna sloužící všem komponentám ISZ CommonMario je probraná v poslední podkapitole 5.7. Aplikace využívá ke komunikaci z databází speciální knihovny UNIMAINMODULELib, UNIUNIMODULELib a Interop.ADODB. Jedná se o DCOM komponenty. První dvě jsem celé přepracoval na .NET knihovny, jinak by nefungovali bez speciálního prostředí, v kterém se používají. Tyto knihovny obsahují i metody, které nemají žádnou funkcionalitu jenom vrací požadovaný typ dat. Je to z toho důvodu aby se kód v aplikaci nemusel na více místech měnit.

## 5.1. Databáze

Aplikace komunikuje s databází typu SQL SERVER 2005 EXPRESS EDITION. Pro školní ukázku je vytvořena lokální databáze, která obsahuje tyto ta-

bulky s těmito relacemi viz. Obr. 27..




Obrázek 27. ER diagram

Tabulky, které jsou na obrázku Obr. 27. včetně relací, kromě tabulky ENLstatus existovali ještě před vznikem této aplikace a jsou používané v již výše vzpomínaném integrovaném systému. Teď je využívá i tato aplikace avšak jenom na čtení a následné vytvoření ENL. Ve skutečnosti, každá z tabulek obsahuje o hodně více atributů. Pro školní ukázku jsem vybral jenom atributy, které jsou využívané projektem ENLKlient. Na obrázku se nachází i tabulka ENLstatus. Tuto tabulku jsem vytvořil pro ukládání dat, určených k identifikaci konkrétní zprávy. Většina atributů z této tabulky jsou součástí posílaných zpráv a byli v zadání. Stručný popis atributů tabulky ENLstatus:

- **idENLstatus** – identifikátor zprávy v tabulce ENLStatus
- **datCasVytvoreniENL** – datum a čas vytvoření ENL
- **odpovedText** – v případě, že zákazník posílá odpověď
- **odeslaniXMLcesta** – tady je uložena cesta, kde se ukládají odesílané XML zprávy
- **odpovedXMLcesta** – tady je uložena cesta, kde se ukládají přijaté XML zprávy
- **oznacenie** – jednoznačný identifikátor konkrétního ENL pro konkrétní zásilku, musí odpovídat regulárnímu výrazu ([0-9]42[0-9]3[0—1][0-9][0-3][0-9]8)
- **kodZakaznika** – kód zákazníka
- **typZpravy** – O=originál, U=update, S=storno
- **idVetaENL** – identifikátor zprávy
- **verzia** – verze je závislá na počtu aktualizací, oprav a stornu
- **typPotvrdenia** – P="doručení podeje", U="doručení ukončení přepravy"
- **poznamka** – poznámka k převzetí
- **vyjadrenie** – vyjádření zákazníka, jestli souhlasí s převzetím, true nebo false
- **odesilatelZpravy** – ISZákazníka nebo ISProvozu
- **datumZpracovaniZpravy** – datum zpracování zprávy pomocí ISZ
- **idSmena** – identifikátor směny
- **idZasilka** – identifikátor zásilky
- **stavSpracovania** – stav zpracování nákladního listu; O="OK-zpracování bez chyb"; E="Error-zpracování s chybou"; W="Warning-zpracování s varovným hlášením"
- **chyby** – popis konkrétní chyby
- **rozhrani** – služba, kterou chce ISP využít
- **datumPrijetiZpravy** – datum přijetí XML zprávy
- **idZprava** – identifikátor zprávy – posledních 7 znaků z označení, existuje kvůli tomu, že někdy může ISP poslat více ENL v jedné XML zprávě

## 5.2. ENLKlient.exe

Soubor ENLKlient je startovací část informačního systému. Jedná se o DCOM soubor datového typu EXE se vstupním i výstupním rozhraním. Je to soubor, který jsem vytvořil jako část sloužící pro větší integrovaný systém, který je celý naprogramovaný ve Visual Studiu 6.0, konkrétně v C++ a proto jsem musel .NET knihovnu upravit na DCOM – kvůli kompatibilitě. V podstatě jsem zapouzdřil knihovnu IsdlENLKlient.dll aby bylo možné komunikovat se staršími technologiemi. V této části projektu jsou vytvořené dva formuláře, pomocí kterých je možné vytvořit, odeslat, přijat požadované XML zprávy a samozřejmě je i zpracovat. Tyto formuláře nebyli součástí zadání. Avšak kvůli přehlednější komunikaci jsem je dodatečně vytvořil. Zadávání dat do databáze a jejich oprava se realizuje v jiné části integrovaného systému, který není součástí tohoto projektu. Jedná se o aplikaci Zásilka z integrovaného systému ISDL. Systém ISDL je komerčně využíván a není možné ho předvést. Aplikace ENLKlient je upravena tak aby byla funkční i bez tohoto integrovaného systému.

### 5.2.1. Popis částí souboru ENLKlient

#### Interface IENL

V tomto rozhraní jsou definovány vstupní parametry s návratovou hodnotou typu object (tzn. jakýkoli datový typ).

```
object VlozeniIdZasilky(int idZasilka, int idSmena, object data1,
object data2, object data3);
```

- **idZasilka** – identifikátor zásilky
- **idSmena** – nemá vliv na funkcionalitu programu, je to identifikátor konkrétního uživatele, který je pak veden jako člověk zodpovědný za práci s konkrétním ENL
- **data1** – momentálně nevyužitý vstupní parametr.
- **data2** – implicitně nastavena "Komunikace s odesilatelem", možnost zvolit "Komunikace s přijímatelem" nebo ponechat původní nastavení.
- **data3** – v případě použití datového typu Dictionary bude možné do objektu vložit asociativní pole, což znamená neomezeny počet atributu s hodnotami, avšak ve skutečnosti takovéto asociativní pole neexistuje v C++ ve Visual Studiu 6, takže při skutečné komunikaci ho nebudeme moci použít v plné míře

### Třída cENL

V této třídě je implementováno rozhraní IENL. Metoda VlozeniIdZasilky, přijímá vstupní parametry od hlavní aplikace, která využívá tento modul a vrací object v podobě výpisu o stavu průběhu. Navíc tady dochází k inicializaci formuláře FrmENL a volání jeho metody AktivaceISZZeZasilek se vstupními parametry.

### Formulář FrmENL

Formulář, který obsahuje hlavní menu, kterého obsahem je tabulka s rozpracovanými zásilkami pomocí ENL. Dále je tady možné pomocí grafického rozhraní nastavit způsob komunikace, zobrazení dialogů, nastavit systémové data. Je tady možnost spustit zpracování emailu pomocí aktivace knihovny EmailPOP3. Případně je možné zvolit volbu Test. Test se postará o odeslání testovacích zpráv na požadovaný email a pak je možné jednoduše otestovat funkčnost aplikace (viz. Test ENL ). Dále jsou popsány nejdůležitější metody tohoto formuláře:

```
public FrmENL(int IdZasilka_, int IdSmena_)
```

Konstruktor třídy se vstupními parametry. Dochází tady k nastavení globálních proměnných IdSmena a IdZasilka.

```
private void FrmENL_Load(object sender, EventArgs e)
```

Metoda, která se spustí vždy při otevření dialogu. Odtud je volána aktualizace tabulky podle databáze a metoda na ověření existence adresářů pro ukládání XML zpráv, případně jejich vytvoření.

```
private void RefreshGridu()
```

Inicializace třídy PraceSGridem, která se stará o načtení dat z databáze do tabulky, neboli do datového typu dataGridView.

```
private void AktivaceISZPrimo(int idZasilka)
```

Metoda, která je volaná v případě přímého spuštění aplikace. To znamená, že aplikace není zpuštěná z integrovaného systému ISDL z části Zásilky. Podle toho se nastaví vstupní parametry.

```
public void AktivaceISZZeZasilek(int idZasilka, int idSmena,
object data1, object data2, object data3)
```

Metoda, která je volaná z integrovaného systému ISDL z části Zásilky. Nastavení vstupních parametrů podle parametrů, které byli zaslané aplikaci Zásilky.

```
private void AktivaceISZ(int idZasilka, int idSmena, object data1,
object data2, object data3)
```

Tato metoda je společná pro obě předchozí varianty. Navíc tady dochází k inicializaci třídy cIsdlENL knihovny IsdlENLKlient a předání těchto vstupních parametrů objektu, který vzniká z třídy cIsdlENL a v kterém je implementované rozhraní IIsdlENL z již vzpomínané knihovny IsdlENLKlient. Nakonec je volán opět aktualizace tabulky se zásilkami.

```
private void btnNovyENL_Click(object sender, EventArgs e)
```

V této události, která nastane při kliknutí na tlačítko „Vytvořit nový ENL“, dochází ke vzniku formuláře frmNovaZasilka a následné otevření tohoto formuláře, kde se nachází seznam všech zásilek i těch, které nejsou rozpracované.

```
private void dtGrdVwZasilkyENL_DataSourceChanged(object sender,
EventArgs e)
```

Tato událost nastane u jakékoli změny obsahu tabulky s rozpracovanými zásilkami. To znamená, že i při prvním naplnění tabulky. Je tady kontrola jestli nedošlo ke změně volby zobrazení a pak tady dochází k zjišťování informací ohledně konkrétních zásilek v tabulce. Jestli už došlo k odeslání zpráv. V případě, že ano změna se projeví v dialogu, konkrétně v tabulce.

```
private void rdBtnOdesilatel_CheckedChanged(object sender,
EventArgs e)
```

Událost nastane v případě, že klient změní způsob zobrazení v tomto formuláři. Pak tady vznikne instance třídy PraceSGridem a pomocí konstruktoru je této instanci poslán celý aktuální formulář.

```
private void btnAktualizovat_Click(object sender, EventArgs e)
```

Aktualizace tabulky se zásilkami pomocí události, která je volaná v případě, že se klikne na tlačítko „Aktualizovat“.

```
private  void dtGrdVwZasilkyENL_CellMouseDoubleClick(object sender,
DataGridViewCellMouseEventArgs e)
```

Tato událost nastane po dvojkliku na jakoukoli rozpracovanou zásilku. A pak se volá již výše popsaná metoda AktivaceISZPrimo.

```
private void btnZpracovatPostu_Click(object sender, EventArgs e)
```

Událost nastane při kliknutí na tlačítko „Zpracovat poštu“. Dochází tady k aktivaci knihovny EmailPOP3, která je odpovědná za komunikaci s poštovním serverem.

```
private void rdBtnUzivatel_CheckedChanged(object sender, EventArgs e)
```

Tato událost nastane u změny zobrazení. Pak se nastaví globální proměnná pro všechny knihovny na požadovanou hodnotu.

```
private void btnTestENL_Click(object sender, EventArgs e)
```

Pomocí této události, která nastane při kliknutí na tlačítko „Test ENL“, vznikne objekt testENL, který se postará o zaslání testovacích zpráv na poštovní server pro příchozí poštu.

```
 private void btnSysNastaveni_Click(object sender, EventArgs e)
```

Událost, která nastane při kliknutí na tlačítko „Systémové nastavení“. Dochází tady ke vzniku instance třídy FrmSysNastaveni. A následně otevření modálního dialogu této instance.

```
private void OvereniExistenceAdresareXMLPripadneVytvoreni()
```

Metoda na ověření existence adresářů, které jsou využívané, nebo budou pro archivování XML zpráv, jak odchozích, tak příchozích. V případě neexistence těchto adresářů, jsou pomocí této metody vytvořeny.

### Třída PraceSGridem

Tato třída je odpovědná za práci s datovým typem dataGridView. Tady je popis nejdůležitějších metod této třídy:

```
public PraceSGridem(FrmNovaZasilka frmNovaZasilka_)
public PraceSGridem(FrmENL frmENL_)
```

Toto jsou přetížené konstruktory se vstupním parametrem konkrétního formuláře.

```
public string NacteniSQLZasilka()
public string NacteniSQLZasilkyENL()
```

Metody, které vrací řetězec naplněný požadovaným sql dotazem.

```
public void NaplneniGridu(string sql, ref DataGridView gridView)
```

Naplnění datového typu dataGridView daty z databáze.

#### Formulář FrmNovaZasilka

Formulář slouží pro zobrazení tabulky všech zásilek i těch, které nejsou rozpracované pomocí ENL.

#### Třída TestENL

Instance této třídy je odpovědna za odeslání testovacích zpráv na poštovní server pro příchozí poštu pomocí knihovny EmailPOP3.

```
public TestENL(int idSmena_)
```

Konstruktor s parametrem volá metodu OdeslaniTestovacichZprav();

```
private void OdeslaniTestovacichZprav()
```

Metoda, která se postará o odeslání testovacích zpráv na poštovní server příchozí pošty.

#### Třída Vyjimky

Obsahuje statickou metodu Log s parametrem datového typu Exception. Tato metoda vypíše na obrazovku chybu, která je poslaná jako parametr do této metody.

## 5.3. Knihovna IsdlENLKlient.dll

Tato knihovna zjišťuje, jestli jsou v databázi, nějaké nezpracované příchozí zprávy. Navíc tady dochází ke komunikaci s ostatními knihovnami projektu. Podle rozhodnutí této knihovny jsou vytvářený instance tříd ostatních modulů, potřebných pro vyřízení požadavků kladených na ISZ. Dochází tady i k zápisu některých důležitých dat do databáze, konkrétně do tabulky ENLStatus. ENLStatus je určená pro ukládání nejdůležitější informací o XML zprávě. 5.3.1

### 5.3.1. Popis částí knihovny IsdlENLKlient

Následuje popis jednotlivých částí knihovny.

#### Interface IIsdlENL

Interface pro komunikaci s touto knihovnou.

### Třída cIsdlENL

Třída, v které je implementované rozhraní IIsdlENL. Probíhá tady inicializace základních tříd této knihovny. Následuje popis nejdůležitějších metod:

```
public object VlozeniIdZasilky(int idZasilka, int idSmena,
object data1, object data2, object data3)
```

Metoda přijímá vstupní parametry od hlavní aplikace, která využívá tuhle knihovnu a vrací datový typ object v podobě výpisu o stavu průběhu. Navíc tady dochází k inicializaci třídy cStart a předání vstupních parametrů objektu, který vzniká z třídy cStart

```
private void ZruseniSessienAModulu()
```

Metoda nejdřív zruší reference na objekty a pak explicitně volá garbage collector, kvůli komunikaci s dcom moduly, které používáme v práci.

```
private void SpecialniLogProEmail()
```

Momentálně nepoužívaná metoda, avšak plně funkční. Je možné ji využít pro zaslání chyb pomocí emailu

### Třída cStart

Startovací třída této knihovny. Popis nejdůležitějších metod:

```
public cStart(int idZasilka_, int idSmena_, object data1_,
object data2_, object data3_)
```

Konstruktor třídy, obsahuje inicializaci objektů tříd dataSet, probíhá tady kontrola vstupních parametrů, v případě, že mají hodnotu null, změníme tyto hodnoty na prázdné řetězce

```
private void InicializaceTrid()
```

Metoda slouží pro inicializaci dalších tříd aplikace. Rozhoduje se tady i o tom jak se bude aplikace chovat na základě vstupních parametrů. Pomocí metody se získají některé systémové nastavení, uložené v databázi.

```
public void InitcENL()
```

Metoda slouží pro vytvoření nové instance třídy cStart.ENL, navíc pomoci zásobníku stENL je zachovaná hodnota atributu „info“

### Třída cCteniDB

Třída se statickými metodami sloužící pro čtení dat z databáze.

```
public static bool NacteniDatPrvniNezpracovaneZpravyZISP
(bool komunikaceSOdesilatelem_)
```

Metoda ukládá data z databáze do objektu třídy cENL, který je vytvořen jako statický objekt v třídě cStart s názvem ENL.

```
public static bool BylJizProvedenPredbeznyPodejNeboAvizoPodeje
(string rozhrani_)
```

Metoda zjišťuje podle vstupního parametru, jestli již byla provedena daná akce. Když (rozhraní = 'avizoPodaja') tak metoda odpoví na dotaz, jestli již bylo provedeno avízo podeje. Když (rozhraní = 'predbeznyPodaj') tak metoda odpoví na dotaz, jestli již byl proveden předběžný podej.

```
public static bool NahledNaVsechnyOperaceENL
(bool zobrazitVsechnyZasilky, bool zobrazitNahled)
```

Metoda se chová podle vstupních parametrů, buď se zobrazí všechny operace všech zásilek, nebo se zobrazí všechny operace konkrétní zásilky a samotné zobrazeni je navíc závislé na parametru zobrazitNahled, kvůli případnému využiti metody, při voláni službou

```
public static bool PotvrdeniePodajaJeSchvaleneZISP()
```

Metoda slouží ke zjištění jestli ISZ poslal zprávu „potvrzeniPodeje“ do ISP.

```
public static string OznamVyslSpracovaniaSeTyka()
```

Metoda přesně zjistí, ke které XML zprávě patří tato zpráva „oznamVyslSpracovania“.

### Třída cKontrolaDatVDB

Instance této třídy se stará o prohlížení dat v databázi v tabulce ENLStatus, kde jsou veškeré potřebné data pro základní analýzu příchozích i odchozích XML zpráv. Na základě těchto informaci se tady rozhodne jak má aplikace reagovat.

```
public cKontrolaDatVDB(int idZasilka_, string adresarXMLOdeslane_,
string adresarXMLOdpoved_)
```

Konstruktor, který se postará při inicializaci objektu této třídy o spuštění operací, které přináleží této třídě. Rozhoduje se tady na základě vstupního parametru, který má vliv na funkcionalitu a je uložený v objektu cStart.ENL.Data2 , jestli se jedná o komunikaci s odesílatelem nebo přijímatelem. Tady se provede kontrola, jestli byl již na tuto konkrétní zásilku poslán předběžný podej nebo ještě ne. V případě, že ne, tak je možné zaslat předběžný podej do ISP. Musí se samozřejmě jednat o komunikaci s odesílatelem.

```
private bool VyhledaniPrvniNezpracovaneZpravyVDB()
```

Metoda vyhledá v databázi, v tabulce ENLStatus první nezpracovanou zprávu. Případně vrátí informaci o tom, že v databázi momentálně není žádná nezpracovaná zpráva.

```
public bool PredbeznyPodej()
```

Metoda slouží k ověření, jestli chce klient poslat předběžný podej. Jestli jo, pak tady vytvoříme instanci třídy cVyberRozhrani. Když je konstruktor třídy cVyberRozhrani bez parametrů, tak se jedná o rozhraní, které je v názvu této metody, tzn.: „předběžný podej“

### Třída cVyberRozhrani

Instance této třídy slouží ke kontrole posledního využitého rozhraní ISZ a následné nastaveni pro využití nového rozhraní z ISZ do ISP nebo jenom nastaveni pro využití nového rozhraní zprávy z ISZ do ISP

```
public cVyberRozhrani(bool odpoved)
```

Přetížený konstruktor třídy. Použitý v případě, že v databázi, v tabulce ENLStatus je nezpracovaná XML zpráva určená pro tuto konkrétní komunikaci. Pak je volána metoda VyberRozhraniISZ.

```
public cVyberRozhrani()
```

Přetížený konstruktor třídy, použitý v případě, že se jedná o první zprávu pro zásilky, tzn.: „predbeznyPodej“.

```
private void Inicializace()
```

Slouží k volání dalších metod, v kterých dochází k inicializaci objektu typu dataSet, dále k volání operací pro nastavení atributů nové zprávy a vytvoření XML zprávy, včetně kontroly hodnot XML.

```
public void VyberRozhraniISZ()
```

Operace na základě jména příchozí XML zprávy vybere rozhraní, které tato zpráva chce využít. Jedná se o zprávu odeslanou z ISP do ISZ.

```
private void NastaveniAtributuNoveZpravy()
```

Metoda na základě přirazeného rozhraní, na které se bude posílat nová zpráva, nastaví některé důležité atributy této zprávy

```
private void VytvoreniXMLAKontrolaHodnot()
```

Metoda obsahuje inicializaci třídy, která se postará o vytvoření konkrétní XML zprávy v českém i v německém jazyku, včetně kontroly atributů, pomocí knihovny CreateENLXML.dll.

### Třída cAktivaceCreateENLXML

Třída je zodpovědná za komunikaci s knihovnou CreateENLXML.dll, která se stará o vytvoření a zpracování nové XML zprávy, určené pro odeslání do ISP

```
public void VytvoreniXML(string StrVyberRozhrani_)
```

Metoda se postará o vytvoření XML zprávy pomocí knihovny CreateENLXML.dll. Díky objektu InitCreateENLXML, který je instancí třídy cInitCreateENLXML z knihovny CreateENLXML se v konstruktoru této třídy pošlou důležité data do této knihovny, kde jsou následně zpracované. Díky statické referenci na objekt ENL, která je vytvořena v třídě cStart a předaná jako parametr objektu InitCreateENLXML, kde jsou pak doplněny data, případně některé upravené, je pak možné pokračovat v práci s tímto objektem a novými daty v knihovně IsdlENLKlient.

```
private static void PrehledAktualnichHodnotENL
(ref StringBuilder atributySHodnotami)
```

Metoda slouží pro naplnění proměnných objektu cStart.ENL podle toho jestli je použitá před nebo po práci s objektem typu CreateENLXML.cInitCreateENLXML. V případě, že ji voláme před vytvořením objektu, tak se naplní kolekce arLstAktualniHodnotyENL atributy s hodnotami, které se posílají do knihovny CreateENLXML. A v případě, že ji voláme po vytvoření objektu, tak se tato kolekce naplní atributy s hodnotami, které přichází z knihovny CreateENLXML

### Interface InterfaceISZakaznika

Interface reprezentuje rozhraní pro komunikaci elektronického nákladního listu na straně informačního systému zákazníka (ISZ).

### Třída cIISZakaznika : InterfaceISZakaznika

Ve třídě je implementované rozhraní InterfaceISZakaznika. Jednotlivé metody této třídy vytvářejí reakce podle rozhraní, které chce informační systém provozu (ISP) využít. Jména metod odpovídají vstupnímu rozhraní informačního systému zákazníka (ISZ = aplikace ENLKlient se všemi pomocnými knihovnami). Pak je tady ještě následující metoda:

```
private void ReakceNaAviza(string vstupniRozhrani_)
```

V této metodě vytvoříme instanci třídy cReakceNaZpravu a pomocí konstruktoru této třídy předáme instanci vstupní parametr: „vstupniRozhrani“.

### Třída cReakceNaZpravuAkce

Tahle třída slouží pro případ, že je zapotřebí nějakým způsobem reagovat na příchozí zprávu z ISP.

```
public cReakceNaZpravuAkce(string vstupniRozhrani_)
```

Konstruktor se vstupním parametrem. Tady se rozhodně, o jaký druh odpovědi se bude jednat.

```
private void InfoProKlientaOObsahuPrichoziZpravy()
```

Metoda vypíše v dialogu, zákazníkovy, informace o zprávě, která byla doručená z ISP.

```
VytvoreniXMLZpravyOznamDorucenia()
```

Metoda vytvoří instanci třídy cAktivaceCreateENLXML, v které vznikne instance třídy cInitCreateENLXML z knihovny CreateENLXML pomocí které pak vytvoříme XML zprávu, určenou pro tuto konkrétní odpověď.

```
private void VytvoreniXMLOznamSuhlasuSPrevzatimRequest()
```

Metoda se chová stejně jako předchozí, avšak posílá jiný argument důležitý pro vytvoření nové XML zprávy. Později se pak proto vytvoří jiná XML zpráva než pomocí předchozí metody. Zpráva, která se vytvoří, odpovídá konkrétní odpovědi.

### Třída cReakceNaZpravu

Tahle třída poskytuje informace o příchozí zprávě pro klienta. Zpráva nevyžaduje odpověď. Jedná se o avíza, které jsou po přečtení zákazníkem označené v databázi jako zpracované.

```
public cReakceNaZpravu(string vstupniRozhrani_)
```

Konstruktor s parametrem vstupního rozhraní. Tento vstupní parametr je rozhraní, které chce využít ISP v ISZ.

```
private void InfoProKlientaOObsahuPrichoziZpravy()
```

Metoda vypíše zákazníkovy informace týkající se příchozí zprávy pro konkrétní zásilku, s kterou právě pracujeme. A pak poznačí do DB datum přečtení neboli zpracování.

### Třída cOznamVyslSpracovania

Instance teto třídy je volaná v případě, že z ISP přišla zpráva, která chce využít rozhraní „OznamVyslSpracovania“.

```
public cOznamVyslSpracovania(string XMLZISP_)
```

Konstruktor vola metodu AnalyzaHodnotAtributuZpravy().

```
private void AnalyzaHodnotAtributuZpravy()
```

Metoda provede analýzu hodnot atributů příchozí XML zprávy z ISP využívající rozhraní „oznamVyslSpracovania“. V dialogu se zobrazí výsledek analýzy.

```
private static void PoznaceniZpracovaniZpravyVDB()
```

Metoda poznačí v databázi zpracování konkrétní zprávy a nastaví proměnnou cStart.ENL.ZpravaSeMaOdesilat na hodnotu „false“. To znamená, že je v celé aplikaci zakázáno odeslání jakékoli zprávy.

### Třída cSkoncenieKomunikacie

Třída obsluhuje zpracování zprávy s informací o skončení komunikace z ISP do IS zákazníka - odesílatele.

```
public cSkoncenieKomunikacie()
```

Konstruktor volající jedinou metodu třídy InfoProKlientaOSkonceniKomunikace(). Navíc se tady zakáže odeslání jakékoli zprávy z ISZ do ISP.

```
private void InfoProKlientaOSkonceniKomunikace()
```

Metoda provede zobrazení informací týkající se ukončení komunikace z ISP.

### Třída cDataDoDB

Tahle třída ukládá data do databáze.

```
public cDataDoDB(string odesilatelZpravy_)
```

Konstruktor v sobě obsahuje volání metody, sloužící k poznačení data a času zpracování konkrétní zprávy do DB. Dále obsahuje i volání metody, která se stará o naplnění hodnot atributů určených do databáze. Tyto data se týkají nové zprávy, připravené na odeslání z ISZ do ISP. Proto je důležitý správný vstupní parametr: „ISProvozu“ nebo „ISZakaznika“.

```
private void NaplneniHodnotAtributuUrcenychDoDB()
```

Plnění dat určených do databáze.

```
public bool VlozeniDatDoDB()
```

Vložení konkrétních dat přímo do databáze pomocí speciálních modulů UNIMAINMODULELib a UNIUNIMODELELib.

```
public bool PoznacitDoDBZeZpravaJeZpracovana()
```

Procedura poznačí do databáze datum a čas zpracování zprávy.

### Třída cGlobal

Pomocná globální třída se statickými proměnnými. Je možné ji dále rozšiřovat v budoucnu.

### Třída CommonIsdlENLKlient

Pomocná třída pro získání jména, verze aplikace a vytvoření proměnné pro speciální logování v případě potřeby. Je možné ji dále rozšiřovat v budoucnu.

### Interface InterfaceISPSpravcaENL

Tento interface je informační kvůli přehledu, jaké zprávy se zasílají do ISP. Není nikde implementován. Bližší popis je přímo v kódu.

### Formulář FrmOznamSuhlasuSPrevzatim

Dialog, kde je možné vložit poznámku k převzetí zásilky. A vyjádřit souhlas, nebo nesouhlas s převzetím zásilky. Data jsou pak pomocí logické vrstvy aplikace vložena do XML zprávy a odeslané do ISP.

#### Formulář FrmPodajOpravaStorno (Prezentační vrstva)

Dialog, kde je možné se rozhodnout, jestli se má provézt potvrzení podeje, předběžný opravený podej nebo storno podeje. Případně žádná z těchto voleb.

#### Formulář FrmVyuziteSluzby

Dialog zobrazí seznam využitých služeb (rozhraní) všech zásilek, nebo jenom jedné zásilky. Výběr závisí na vstupních parametrech integrovaného systému, který využívá tuto aplikaci jako komponentu.

#### Zdrojový soubor Zdroje

Tento soubor obsahu SQL dotazy, které jsou pak volané v aplikaci. Obsahuje tyto dotazy: SQLAvizoPodaja, SQLDatCasVytvoreniENL, SQLENLAll, SQLENLAllZasilka, SQLENLStatusOdeslane, SQLENLStatusOdpoved, SQLENLStatusPredbeznyPodej, SQLENLVerzia, SQLPotvrdeniePodaja.

## 5.4. Knihovna EmailPop3.dll

Knihovna EmailPop3. Tuhle knihovnu, jsem vytvořil zapouzdřením již existující knihovny smtpPop3, která pracuje s poštovním protokolem pop3. Zapouzdřením a implementací svého kódu jsem vytvořil, knihovnu, která vyhledává na určeném poštovním serveru emaily určené pro komunikaci a následně je zpracuje. Obsah emailu dešifruje a extrahuje do adresáře určeného pro skladování XML zpráv z ISP určených do ISZ. Základní informace o zprávě, které se nacházejí v XML kódu, jsou pak uloženy do databáze do tabulky ENLStatus. Tady je uložen i čas přijetí zprávy a po zpracování i čas zpracování zprávy. Navíc jsem do této knihovny implementoval i třídu na odesílání emailu, která pracuje s poštovním protokolem SMTP.

### 5.4.1. Popis částí knihovny EmailPop3

Tato knihovna obsahuje následující třídy, formuláře.

#### Třída cStartPOP

Táto třída obsahuje inicializaci třídy odpovědné za příjem pošty, jsou tady načítané zkladní data pro práci s poštou.

```
public cStartPOP(Dictionary<object, object> dictDataCastENL)
```

Konstruktor volá metodu NastaveniAutentizace(), pak vytvoří instanci třídy cZpracovaniZpravy, využije některé metody této třídy. Taky je tady vytvořena instance třídy cDBConnect, která se stará o uložení dat do databáze. Jako vstupní parametr této metody je použité asociativní pole generického typu.

`private void AktivaceDialogu()`

Metoda slouží k zobrazení informačního dialogu.

`private void DeAktivaceDialogu()`

Metoda slouží k zavření informačního dialogu.

`public static void Init(int idSmena_)`

Statická metoda pro vytvoření objektu, který komunikuje z databází a objektu, který obsahuje veškeré důležité atributy určené pro identifikaci XML zprávy.

`private void SmazatHodnotyGlobalnichPromennych()`

Mazáni globálních proměnných třídy cGlobal, kvůli opětovnému volání některých metod této knihovny.

`private void LogovaniUlozenychDat()`

Logování uložených dat.

`private void OznacitSouborJakoZpracovany()`

Metoda zkopíruje zpracovávaný soubor, který nám poslal ISP, doplní před datový typ „.XML“ znaménko +, což bude znamenat, ze tahle zpráva je již vyřízena a plně zpracovaná. A nakonec smaže původní XML zprávu bez znaménka +.

`public static void NastaveniAutentizace()`

Nastaveni všech relevantních dat pro komunikaci pomocí emailu.

`private static void ZruseniSessienAModulu()`

Metoda ruší sessiony a moduly.

### Třída cZpracovaniEmailu

Instance této třídy slouží pro zpracování emailové zprávy.

```
public cZpracovaniEmailu(FrmDialog frmDialog_)
```

Konstruktor vola metodu PrijetiEMailu().

```
private void PrijetiEMailu()
```

Metoda pomocí instance knihovny SmtPop, třídy POP3Client komunikuje s poštovním serverem přes protokol pop3. Navíc kontroluje jednotlivé emailové příchozí zprávy, jestli se jedna o zprávu určenou do ISZakaznika nebo ne.

```
private void SmazaniObsahuDialogu()
```

Metoda smaže obsah hodnot zobrazených v dialogu.

### Třída cZpracovaniZpravy

Instance této třídy slouží k zpracování zprávy, která se nachází v emailové zprávě jako příloha, jedna se o XML zprávu určenou pro komunikaci s ENL.

```
public void ZjisteniCestyKAdresaruNaUlozeniOdpovedi()
```

Pomocí této metody zjistíme cestu k adresáři, který nám má sloužit pro ukládání příchozích zpráv z ISProvozu určených do ISZakaznika

```
public void NacteniZakladnichDatZOdpovedi(string innerXML)
```

Procedura volá metody pomoci, kterých dojde k načtení základních dat z příchozí XML zprávy.

```
private void UlozeniZpravyDoAdresareSCislemOznaceni()
```

Pomocí této metody uložíme zprávu do adresáře s číslem označení, které se nachází uvnitř zprávy.

```
private void AnalyzaHodnotAtributuOznacenie()
```

Metoda rozloží hodnotu atributu oznacenie a její částí přiradí konkrétním určeným proměnným.

```
public void RozzipovaniARozsifrovaniZpravy()
```

Rozzipuje a rozšifruje zprávu, zazipovaný originál označíme: ...+.zip a navíc v této metodě inicializujeme globální proměnné, v kterých budou uložené další důležité data, jako cesta, jméno a cesta se jménem souboru. Na rozzipovani a rozšifrování je použitá knihovna ZipSifra.dll.

```
private void ZjisteniJmenaRozhraniKtereChceZpravaZISPVyuzit
(string innerXML_)
```

Metoda zjišťuje, jaké rozhraní chce zprava využít. A díky tomu pak pomocí dalších operací zjistí aplikace strukturu XML zprávy a bude schopna načíst požadované data.

```
private void NacteniHodnotZXMLZpravy()
```

Metoda slouží pro zpracování hodnot z příchozí XML zprávy a jejich uložení do objektu ENL, který je vytvořený v třídě cStartPOP.

### Třída Autentizace

Tahle statická třída slouží k ukládání a používání dat tykajících se emailového serveru. Nejsou tady žádné operace, kromě vlastností proměnných.

### Třída cDBConnect

Instance této třídy zabezpečuje ukládání základních dat tykajících se příchozí zprávy ENL z ISProvozu.

```
public cDBConnect()
```

Konstruktor třídy volá metodu na naplnění hodnot atributu určených do databáze a následně metodu pro uložení základních informaci do databáze. V případě, že se nepovedou z jakéhokoli důvodu uložit úspěšně data do databáze je tahle informace uložena v logu.

```
private void NaplneniHodnotAtributuUrcenychDoDB()
```

Metoda plní hodnoty atributů určených do databáze.

```
public bool UlozeniZakladnichInformaciDoDB()
```

Metoda slouží pro ukládání základních dat do databáze.

### Třída cOdeslaniPosty

Třída se stará o odesílání emailů pomocí serveru určeného v databázi v systémových nastaveních (tabulka systemoveNastaveni).

```
public cOdeslaniPosty(bool jednaSeOLog_, Dictionary<object, object>
dictDataCastENL_)
```

Konstruktor volá všechny potřebné metody potřebné pro odeslání pošty pomocí poštovního protokolu smtp.

```
private void NastaveniPrijimatelaPosty()
```

V metodě se nastaví globální proměnné této třídy, jsou to data tykající se přijímatele pošty.

```
private void NastaveniPrijimatelaPostyProLog()
```

Tahle metoda slouží pro případ, že chceme využít odeslání logu emailovou poštou.

```
private void DefiniceDatProOdeslaniEmailu()
```

V této metodě se nastaví jméno serveru a jméno emailové adresy uživatele.

```
public void OdeslaniPostyPomociSMTP()
```

Metoda se stara o odeslání pošty pomocí protokolu SMTP.

### Třída CommonEmailPOP3

Pomocná třída pro knihovnu EmailPOP3.

```
public static string JmenoAVerzeApp
```

Tahle statická vlastnost vrací jméno a verzi aktuální knihovny v datovém typu string.

### Formulář FrmDialog

Dialog pro zobrazení průběhu zpracování emailové schránky.

## 5.5. Knihovna CreateENLXML.dll

Tato knihovna je určena pro vytvoření XML zprávy, která má být odeslaná do ISP (ISProvozu). Dochází tady i k přeložení vytvořené zprávy z českého jazyka na německý. Překlad se týká elementů a atributů, takže všech tagů XML zprávy. Hodnoty zůstávají původní. Pak je v této knihovně provedena validace XML zprávy, pomocí xsd schémat, které byli dodané z ISP. V případě, že zpráva projde validací, jsou základní identifikační data týkající se ENL uložené do databáze, do tabulky ENLStatus. Nakonec je možné zprávu odeslat do ISP pomocí knihovny EmailPOP3.

### 5.5.1. Popis částí knihovny CreateENLXML

V této podkapitole jsou popsány následující části.

#### Třída cInitCreateENLXML

Tady se načítají vstupní data a pak se rozhodne, jaký typ XML zprávy bude vytvořen.

```
public cInitCreateENLXML(CommonMario.cENL dataProENL_)
```

Konstruktor s parametrem obsahujícím základní data ENL.

```
public void VyberTypuXML(string typXML_)
```

Metoda slouží pro výběr typu XML zprávy na základě vstupního parametru typu string. Navíc obsahuje instanci objektu třídy cFinalUpravy, která se postará už při inicializaci o všechny potřebné finální úpravy XML zprávy.

```
private static void VytvoreniIDVetaAIDZprava()
```

Pomocná metoda pro vytvoření instance třídy cDBZasilka a pak využití jedné z metod této instance pro tvorbu identifikátorů idVeta a idZprava.

#### Třída cStartCreate

Třída, kde se inicializují velice důležité statické objekty, které jsou hodně používané v cele aplikaci.

```
public cStartCreate(CommonMario.cENL dataProENL_)
```

Konstruktor třídy s parametrem provádí inicializaci statických objektů projektu.

### Třída cVytvoreniXMLZoznamENL

Instance této třídy slouží k vytvoření XML zprávy typu ZoznamENL, která chce využít jedno z těchto tři rozhraní v ISProvozu: „predbeznyPodajRequest“, „opravaDatRequest“, „potvrdeniePodajaRequest“

```
public cVytvoreniXMLZoznamENL()
```

Konstruktor třídy se stará o inicializaci datasetů v kterých je uložená struktura XML zprávy, dále volá metodu na vytvoření souboru XML v českém jazyku a pak se tady inicializuje třída odpovědna za překlad této zprávy do německého jazyka.

```
public void VytvoreniSouboruFormatuXML()
```

Metoda vytváří soubor formátu XML, pomocí objektů, které vznikají v této metodě. Jedná se o objekty – hlavickaENL a vozneAUO.

```
private void NastaveniHodnotAtributuAElementu
(DataSet objektDataSet_, ArrayList arrayListObjektyENL_)
```

Metoda nastavuje hodnoty atributu a elementu datasetů. Vzniká tady objekt, pomocí kterého dochází ke kontrole hodnot, které se nachází v nové zprávě.

### Třída cHlavickaENL

Instance této třídy slouží k zpracování dat týkajících se hlavičky ENL. Obsahuje objekt, sloužící pro načtení dat z databáze do objektů datového typu ADODB.RecordSet. Dále tady dochází k předání obsahu objektů typu ADODB.RecordSet do objektů typu dataSet s názvem DtSetHlavicka.

```
public cHlavickaENL(cDBZasilka DBZasilka_)
```

Konstruktor třídy s parametrem používá metodu na načtení dat z databáze pomoci objektu DBZasilka.

```
private void PrirazeniRecordSetu()
```

Metoda přesouvá data z objektu DBZasilka do lokálních objektů datového typu ADODB.RecordSet.

```
public void PresouvaniDatZRSDoDataSetu()
```

Metoda přesouvá data z lokálních objektů typu ADODB.RecordSet do objektů typu DataSet. Navíc tady dochází k vytvoření některých důležitých hodnot určených pro identifikaci XML zprávy.

```
private static string VypocetHodnotyPriznaku(int priznakInt,
string priznak)
```

Pomocná metoda pro určení hodnoty jednoho z atributů zprávy, konkrétně se jedná o atribut s názvem příznak.

```
private void KontrolaCislaStanice(ref string cisloZ, ref string
cisloDo)
```

Metoda provede kontrolu počtu řetězců čísla stanic, v případě, že je počet řetězců menší než 6, tak doplní na začátek tolik nul, aby se počet řetězců rovnal 6.

```
private void DoplneniNul(ref string cisloZNeboDo, int pocetRetezcu)
```

Pomocná metoda předchozí metody. V případě potřeby slouží pro doplnění nul před řetězec.

### Třída cVozneAUO

Instance této třídy slouží ke zpracování dat tykajících se vozů ENL. Dále je tady realizované pomocí objektu DBZasilka načtení dat z databáze do objektů typu ADODB.RecordSet a předání těchto objektů do objektů datového typu dataset se jménem DtSetVozneAUO.

```
public cVozneAUO(cDBZasilka DBZasilka_)
```

Konstruktor třídy s parametrem používá jednu z metod objektu DBZasilka pro načtení dat z databáze.

```
private void PrirazeniRecordSetu()
```

Metoda přesouvá data z objektu DBZasilka do lokálních objektů datového typu ADODB.RecordSet.

```
private void PresouvaniDatZRSDoDataSetu()
```

Metoda přesouvá data z lokálních objektů typu ADODB.RecordSet do objektů typu DataSet.

```
private void NaplneniENLTovar(int pocitadloVoznu, ref int idENLRID,
string cisloVozna, ref int idTovar)
```

Pomocná metoda předchozí metody se zaměřením na objekt ENLTovar a jeho součásti.

```
private void NaplneniENLKontajner(int idTovar_, string materialVozen)
```

Pomocná metoda předchozí metody se zaměřením na objekt ENLKontajner.

### Třída cValidaceDtSet

Instance této třídy kontroluje hodnoty ENL, které jsou uloženy v objektech typu dataset, ve vlastnosti caption - délku řetězce, jestli může byt hodnota null, jestli se nejedná o výčtový typ nějakých prvku atp.

```
public cValidaceDtSet(string jmenoTabulky_, string jmenoAtributu_,
string hodnota_, string popisHodnoty_)
```

Pomocí konstruktoru se nastaví základní proměnné potřebné pro práci s hodnotou daného atributu.

```
public void KontrolaRetezcuVDataSetu(ref DataColumn sloupec)
```

Metoda zjišťuje omezení atributu a kontroluje, jestli jsou dodržená.

```
private void KontrolaDelkyRetezce(string omezeni_)
```

Pomocná metoda na kontrolu délky vstupního parametru.

```
private void ValidaceDecimal(string omezeni_)
```

Pomocná metoda na kontrolu datového typu vstupního parametru, jestli se jedna o datový typ decimal a jestli není překročen počet znaků před nebo po desetinné čárce.

```
public void KontrolaHodnotyPodleCiselniku(string popisHodnoty_)
```

Funkce se stará o validaci hodnot, které můžou nabít jenom určitý výčet hodnot.

### Třída cUpravyTextu

Instance třídy slouží pro úpravy textu XML zpráv.

```
public cUpravyTextu(string puvodniText_)
```

Konstruktor s parametrem textu, který bude objekt této třídy upravovat.

```
public string UpravText()
```

Metoda upraví text XML zprávy - vnitřní hodnoty elementu, které mají být atributy těchto elementů, se pomocí této metody upraví na atributy těchto elementu

### Třída cPreklad

Instance této třídy slouží k překládání XML zprávy z češtiny do němčiny. Překládají se všechny tagy XML zprávy.

```
public cPreklad(DtSetHlavicka hlavickaENL_, DtSetVozneAUO vozneAUOENL_)
```

Konstruktor třídy s parametry.

```
public void prekladSlovencinyNaNemcinu(string puvodniInnerXML_,
string jmenoSouboruSlovnikuNaPreklad_, string jmenoXMLSouboruPrelozeneho_)
```

Metoda překládá názvy slovenských atributu na německé.

```
private void PrelozeniNazvuAtributuAElementuTabulky(DataSet
objektDataSet, string jmenoTabulky_, ArrayList seznamHodnot)
```

Pomocná metoda předchozí metody. Metoda překládá jména atributu a elementu XML zprávy ze slovenského jazyka do českého.

### Třída cValidaceXML

Instance teto třídy se stará o validaci XML zprávy podle XSD schématu.

```
public cValidaceXML(string jmenoXMLSCestou_)
```

Přetíženy konstruktor s jedním parametrem. Aktuálně nepoužívaný.

```
public cValidaceXML(string jmenoXMLSCestou_, string jmenoSchemaSCestou_)
```

Přetížený konstruktor se dvěma parametry. Aktuálně používaný.

```
public void ValidaceDokumentuUrcenimSchemat()
```

Metoda provede validaci XML podle daných schémat. Aktuálně používaná.

```
private void lValReader_ValidationEventHandler(object sender,
ValidationEventArgs e)
```

Tenhle Handler je volaný v případě jakékoli nesrovnalosti vytvořeného XML s XSD schématem.

```
public void ValidaceDokumentu()
```

Metodu je možné použít v případě, že v XML zprávě je odkaz na XSD schéma, kterému má odpovídat. Aktuálně nepoužívaná.

### Třída cOznam

Instance této třídy slouží k vytvoření XML zprávy typu: „PotvrdenieDorucenia“ nebo „SuhlasSPrevzatim“, která chce využít jedno z těchto dvou rozhraní ISProvozu: „oznamDoruceniaRequest“ nebo „oznamSuhlasuSPrevzatimRequest“.

```
public cOznam(string rozhrani_)
```

Konstruktor zavolá metodu VytvoreniXMLOznam s parametrem „rozhrani“, na základě, kterého se rozhodne, jestli vytvoří XML zprávu pro „oznamDorucenia“ nebo pro „oznamSuhlasuSPrevzatim“.

```
private void VytvoreniXMLOznam(string rozhrani_)
```

Metoda vytvoří XML zprávu a uloží ji do globální proměnné StartCreate.ENL.InnerXMLNaOdeslani.

### Třída cFinalUpravy

Instance této třídy zabezpečuje uložení odesílací zprávy do určeného adresáře. Probíhá tady inicializace třídy cValidace, která je odpovědná za validaci XML dokumentu a jsou tady i textové úpravy XML.

```
public cFinalUpravy()
```

Konstruktor třídy zabezpečuje volání metod této třídy.

```
private void FinalniUpravyElementuAParametru()
```

Operace upraví kořenový element - jeho jméno a doplní informace ohledne XSD schémat, kterým má XML zpráva odpovídat.

```
private void UlozeniOdesilaciZpravyDoAdresare()
```

Operace uloží zprávu do adresáře určeného pro ukládání zpráv na odeslání. Jedná se o zprávu, kterou chceme odeslat z ISZakaznika do ISProvozu.

```
public static void UpravyXML(string XMLHlavicka_, string
XMLVozneAUO_, bool nemeckaVerze_)
```

Metoda slouží na úpravy XML zprávy jak české tak německé verzi, podle parametru "nemeckaVerze_". Vzniká tady instance již popisované třídy cUpravyTextu. A taky je tady použita tato instance.

```
private static void NahledNaVytvorenouXMLZpravu(string
jmenoZpravySCestou_, string verzeXML_)
```

Metoda slouží pro zobrazení dialogového okna s obsahem XML zprávy, která se nachází v souboru, ke kterému se dostaneme i pomocí parametru „jmenoZpravySCestou_“.

### Třída cDBZasilka

Třída slouží pro čtení dat z databáze. Tyto data se týkají zásilky.

```
public cDBZasilka(int idZasilka_)
```

Konstruktor s parametrem, pro nastavení proměnné idZasilka v této třídě.

```
public int NacteniDatZDBDoHlavicky()
```

Metoda je určená pro načítání dat z databáze do hlavičky ENL.

```
public int NacteniDatZDBDoVoznu()
```

Metoda je určená pro načítání dat z databáze pro vozy ENL.

```
public int NacteniDatZDBDoIDVetaAIDZprava()
```

Metoda je určená pro načtení a zpracování dat z databáze pro tyto dva identifikátory zprávy idVetaENL a idZpravaENL.

### Třída CommonCreate

Pomocná třída pro knihovnu CreateENLXML.

```
public static string JmenoAVerzeApp
```

Tahle statická metoda vrací jméno a verzi aktuální knihovny v datovém typu string.

```
public static string JmenoAVerzeSpoustecihoExe
```

Tahle statická metoda vrací jméno a verzi souboru typu EXE, který volá tuhle knihovnu.

```
public static void VytvoreniJmenaOdesilaciZpravy()
```

Metoda zabezpečuje vytvoření jména odesílací zprávy, které má přesnou strukturu.

### DataSet Objekty: DtSetEnlToISP, DtSetHlavicka, DtSetVozneAUO

V těchto objektech datového typu DataSet jsou uložené struktury XML zpráv, které je možné v této knihovně vytvořit. Tyto objekty obsahují i omezení kladené na hodnoty atributů zprávy.

### Formulář FrmDialogCreate

Informační dialog zobrazující vytvořenou XML zprávu. Dialog určený hlavně pro testování.

### Zdrojový soubor Zdroje

Tento soubor obsahu SQL dotazy, případně pomocné textové soubory, které jsou pak volané v aplikaci. Obsahuje tyto dotazy: SQLIdVetaENLNew, SQLIdZpravaENLNew, SQLJeridCNci, SQLjeridnlzr, SQLKontajner, SQLPlatnostMaterialu, SQLPotvrdeniePodaja, SQLTovar, SQLVozen, SQLZakaznik, SQLZasilka. Obsahuje tyto pomocné textové soubory: elementFrachtbrief, KorenovyElement, SlovnikSlovenskoNemecky, SlovnikSNasobnosti. Vzhledem k tomu, že požadavky na některé části aplikace se ještě mění a upravují, tak se může stát, že ne všechny části kódu jsou využité nebo plně optimalizované. Naopak můžou být připravené na alternativy.

## 5.6. Knihovna ZipSifra.dll

Tato knihovna slouží na komprimaci, dekomprimaci, šifrování a dešifrování jak příchozích, tak i odchozích zpráv. Využívá ji knihovna EmailPop3 i knihovna IsdlENLKlient. Je v ní zapouzdřena již existující knihovna ICSharpCode.SharpZipLib.dll [8].

### 5.6.1. Popis částí knihovny ZipSifra

Následuje popis nejdůležitějších tříd této knihovny.

#### Třída cZipSifra

Třída umožňuje komprimaci, dekomprimaci, šifrování, dešifrování, heslování, roz-heslování souboru. Na komprimaci a dekomprimaci je použitý formát zip.

```
public cZipSifra(string cestaDoAdresare_, int idSmena_)
```

Konstruktor třídy se dvěma vstupními parametry.

```
public void SifrovatSeZipovanim(string jmenoSouboru, string heslo)
```

Metoda komprimuje a šifruje, navíc používá heslo.

```
public void DesifrovatSRozzipovanim(string jmenoSouboru, string heslo)
```

Metoda dekomprimuje a dešifruje soubor s heslem.

```
private void VyvolanaVyjimka(Exception ex, string jmenoMetody)
```

V případě vyvolání výjimky v této třídě je pak volána tato metoda.

```
private bool JeEmail(string Retezec)
```

Metoda na kontrolu řetězce jestli se jedna o e-mail nebo nikoli, pomoci regulárního výrazu.

```
private string ZjisteniDatovehoTypu(string jmenoSouboruSDatTypem)
```

Metoda vrací datový typ souboru.

#### Třída VlastnostiSouboru

Třída je použita na uložení některých dat týkajících se souboru, s kterými pracuje knihovna.

#### Třída CommonZipSifra

Pomocná třída, pro zjištění jména a verze aplikace.

## 5.7. Knihovna CommonMario

Knihovna, kterou využívají všechny předchozí knihovny. Pro tento projekt jsou použity nesledující třídy z této knihovny.

### 5.7.1. Popis částí knihovny CommonMario

Následuje velice stručný popis tříd, které obsahuje tato knihovna.

#### Třída cDBManager

Třída slouží pro komunikaci s databází. Navíc obsahuje i metody pro logování chybných stavů, případně jakýchkoli poznámek.

#### Třída cENL

Instance této třídy slouží pro přístup k datům určeným pro komunikaci ENL, pro jejich inicializaci i modifikaci.

#### Třída Common

Tato třída obsahuje statické pomocné metody, hlavně pro konverzi z jednoho datového typu do druhého.

# Závěr

Výsledkem této bakalářské práce je aplikace, která představuje informační systém zákazníka, sloužící pro zpracování elektronického nákladního listu. Systém umožňuje využít všechny druhy rozhraní podle zadaného požadavku. Díky testům, které již proběhly, byl systém doladěn. Je vytvořen pro využití v praxi. Systém je možné dále rozšiřovat. V budoucnu by bylo možné vytvořit přívětivější grafické rozhraní, pro jednodušší obsluhu klientem. Navíc je možné rozšířit systém o další atributy a hodnoty XML zprávy, v případě, že by o to byl zájem ze strany klienta, případně ze strany provozu. Při popisu některých metod jsem znovu procházel některé operace, které bych dnes již vytvořil trochu jinak. Takže když bude čas, rád zoptimalizuji i vnitřní kód.

# Conclusions

The result of this work is an application that is serving the customer information system for processing an electronic bill of lading. The system allows you to use all kinds of interfaces according to specified requirements. Thanks to the tests already conducted system was tuned. It is designed for use in practice. The system can be further extended. In the future it would be possible to create a friendly graphical user interface for easier service. Moreover, it is possible to extend the system to additional attributes and values of XML messages in the event that it was the interest of the client or by the operation. In describing some of the methods I again went through some operations that I would today create a bit differently. So when it's time I will optimize the internal code.

# Reference


[1] Bayer, Jůrgen. *C# 2005: Velká kniha řešení.* Computer Press, Brno, 2007.

[2] Mareš, Amadeo. *1001 tipů a triků pro C#.* Computer Press, Brno, 2008.

[3] Pirkl, Josef. *Řešené příklady v C# : C# skutečně prakticky.* KOPP, České Budějovice, 2005.

[4] Petzold, Charles. *Programování Microsoft Windows Forms v jazyce C#* Computer Press, 2008.

[5] Nagel, Christian. *C# 2005: Programujeme profesionálně.* Computer Press, 2008.

[6] SOURCEFORGE [online]. *SmtPop Library 0.4, verze 0.5.1.33667. http://sourceforge.net/projects/dotnetctrlext/files/*

[7] SOURCEFORGE [online]. *SharpZipLib for .NET Framework 2.0, verze 0.86.0.518.http://www.icsharpcode.net/opensource/sharpziplib/Download.aspx*

# A. Popis obsahu přiloženého CD

Součástí této práce je i CD. Jeho obsahem jsou tři adresáře a jeden soubor.

- Adresář `bin` obsahuje instalátor InstallENLKlient.msi. Obsahuje taky spustitelný program včetně všech knihoven, které jsou potřebné pro přímé spuštění programu z CD. Navíc jsou tady i instalátory pro Microsoft .NET Framework 3.5 a Microsoft SQL Server 2005 Express Edition, které jsou zapotřebí pro spuštění aplikace.
- Adresář `doc` obsahuje dokumentaci této práce ve spustitelné podobě a zdrojové kódy dokumentace.
- Třetí adresář `src` obsahuje zdrojové kódy programu.
- V souboru `readme.txt` je popsáno, jak lze aplikaci spustit.