VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO DROBNÉHO ZEMĚDĚLCE

INFORMATION SYSTEM FOR A SMALL FARMER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. ŠTĚPÁN ČERNOCH**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání diplomové práce/19976/2016/xcernо20

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Černoch Štěpán, Bc.**
Obor: Počítačové a vestavěné systémy
Téma: **Informační systém pro drobného zemědělce**
Information System for a Small Farmer
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s principy tvorby informačních systémů na webu a daty, které poskytuje ministerstvo zemědělství ohledně výskytů různých škůdců v ČR.
2. Analyzujte požadavky na informační systém pro drobného zemědělce, který by měl umožňovat správu příjmů a výdajů, zákazníků, objednávek atd. Dále by dle aktuálních informací z www stránek MZeČR načíst informace o výskytu škůdců a zemědělce upozornit na jejich výskyt ve vybraných lokalitách.
3. Navrhněte informační systém dle požadavků. Využijte k tomu vhodných modelovacích technik.
4. Informační systém implementujte a ověřte jeho funkčnost na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky a další možné pokračování tohoto projektu.

Literatura:

- WWW stránky ministerstva zemědělství ČR: http://eagri.cz/public/web/mze/
- Welling, L., Thomsonová, L.: PHP a MySQL: rozvoj webových aplikací. Vyd. 1. Praha: SoftPress, 2003, 910 s. ISBN 80-86497-60-7.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

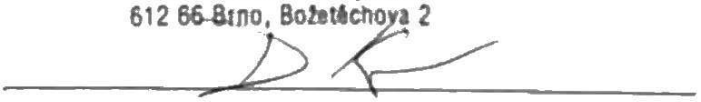
Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2



doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato diplomová práce se zabývá analýzou, návrhem, implementací a testováním informačního systému pro drobného zemědělce. Cílem systému je správa zákazníků, příjmů a výdajů, objednávek, správa zápisků a možnost sledování výskytu škůdců plodin. Práce popisuje možnosti vývoje informačních systémů, detailní specifikaci požadavků a návrh systému. Dále se věnuje implementaci, zejména pak způsobu získávání informací o škůdcích a následně testování aplikace. Informační systém je realizován v jazyce C# jako desktopová aplikace využívající framework .NET a návrhového vzoru MVVM.

Abstract

This diploma thesis deals with the analysis, design, implementation and testing of an information system for a small farmer. The aim of this system is to administer customers, an income and outcome, orders, to manage notes, and possibility to track the occurrence pests of crops. The thesis describes options for the development of information systems, detailed specifications requirements and the design of system. Further is described the implementation, especially then the way to get information about pests and then testing the application. The information system is realized in the language C# as a desktop application using the framework .NET and the design pattern MVVM.

Klíčová slova

Informační systém, C#, MSSQL, .NET, WPF, příjmy, výdaje, zákazníci, výskyt škůdců, WMS server.

Keywords

Information system, C#, MSSQL, .NET, WPF, income, outcome, customers, occurrence of pets, WMS server.

Citace

ČERNOCH, Štěpán. *Informační systém pro drobného zemědělce*. Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Bartík Vladimír.

Informační systém pro drobného zemědělce

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka Ph.D. Uvedl jsem veškeré literární prameny a publikace, ze kterých jsem čerpal.

.....
Štěpán Černoch
22. května 2017

Poděkování

Děkuji mému vedoucímu panu Ing. Vladimíru Bartíkovi Ph.D. za vedení mé diplomové práce.

Obsah

Kapitola 1

Úvod

Nejrůznější informační systémy jsou již neodmyslitelnou součástí dnešního světa. Tato práce je věnována návrhu a implementaci informačního systému pro drobného zemědělce. Tento systém bude zaměřen na malé zemědělské subjekty, pro které není zemědělství hlavním zdrojem finančních příjmů. Dle údajů českého statistického úřadu je počet subjektů, kteří obhospodařují plochy do 10 ha, kolem 8500 [13]. V současné chvíli se takový produkt jeví jako díra na trhu.

Cílem práce je vytvořit aplikaci, která uživateli umožní přehledné zpracování agendy spojené s jeho zemědělskou činností. Neopomenutelnou vlastností je poskytnutí přehledného a efektivního rozhraní, které nebude vyžadovat zdlouhavé školení uživatelů. Aplikace poskytne možnost spravovat příjmy a výdaje, dále pak správu zákazníků a objednávek jednotlivých produktů. Systém bude také nabízet vyobrazení výskytu škůdců vybraných plodin v určeném okolí. Díky tomu získá uživatel cenné informace a bude moci rychleji provést patřičná opatření.

Tato práce postupně seznámí čtenáře s obecnou problematikou informačních systémů. Dále pak přiblíží možnosti informačního systému Ministerstva zemědělství (MZe), který monitoruje výskyt škůdců na území České republiky. Následně je uvedena kompletní specifikace požadavků na výsledný systém, které jsou následně analyzovány. Na základě analýzy požadavků je poté navrhnuta architektura aplikace s využitím dostupných moderních metod. Dále jsou vybrány technologie, které jsou využity pro tvorbu aplikace.

V další části je popsána implementace vybrané vícevrstvé architektury a způsob komunikace mezi jednotlivými vrstvami. Zároveň je také detailněji popsána implementace pro získávání obrazových informací o výskytech škůdců. V této části je uveden způsob řešení grafické rozhraní vytvořené aplikace.

Předposlední kapitola se věnuje testování vytvořené aplikace a v závěrečné kapitole jsou shrnuty dosažené výsledky této práce a také možnosti dalšího pokračování.

Kapitola 2

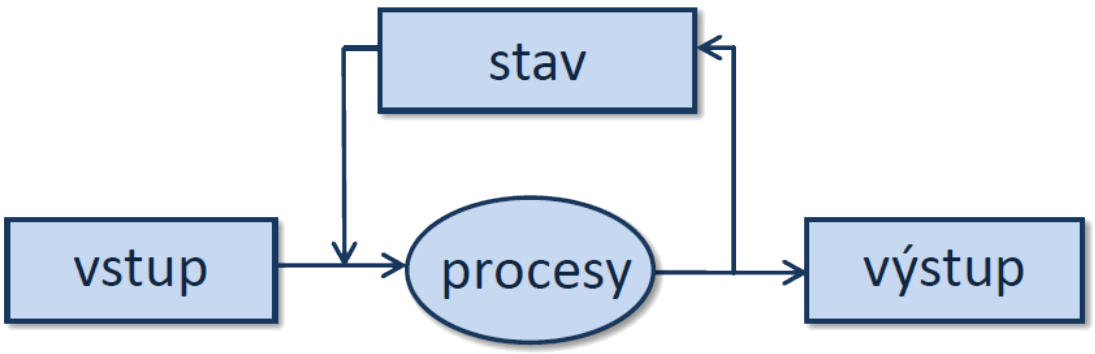
Vývoj informačního systému

Tato kapitola se věnuje teoretickému popisu všech důležitých pojmů a technologií, které jsou potřebné jak pro pochopení následujících částí, ale také pro samotnou tvorbu celé aplikace. V jednotlivých podkapitolách bude vysvětlena definice informačního systému. Dále pak bude popsán systém Ministerstva zemědělství pro monitorování výskytu škůdců a jeho možnosti využití. Nedílnou součástí je také popis použitých technologií jako frameworku .NET, návrhového vzoru MVVM a relačního mapování pomocí Entity Frameworku.

2.1 Definice informačního systému

Obecně je systém definován jako množina prvků a vazeb mezi nimi. V případě informačního systému lze říci, že se jedná o systém pracující s určitými informacemi. Na obrázku 2.1 je uvedeno obecné schéma informačního systému, který se skládá z části pro zadávání vstupů a z části pro získávání výstupů ze systému. Mezi vstupem a výstupem systému probíhá typicky transformace nad vstupními daty pomocí algoritmů. Důležitou částí systému je jeho stav, který je zároveň také vstupem do systému. Na základě tohoto je výstup systému závislý na stavu a vstupu. Stavem systému jsou hodnoty dat.

Pro každý systém je důležité se zabývat persistencí(přetrváváním) a konzistencí (splněním pravidel, které mohou nabývat uchovávané hodnoty). Informační systém je zpravidla modelem (abstrakcí) nějakého fyzického systému.[14]



Obrázek 2.1: Schéma informačního systému

Důležitou částí, která je podstatná pro uživatele, je jak vhodné a přehledné zobrazení výstupů systému, tak i zadávání uživatelských vstupů.

2.2 Webový vs. desktopový systém

Existuje mnoho rozdělení informačních systémů. Ovšem z hlediska spustitelnosti systému lze tyto systémy rozdělit na dvě kategorie, webové a desktopové. Toto rozdělení patří v současné době k nejrozšířenějším. Základním rozdílem mezi nimi je způsob spouštění, resp. způsob přístupu do systému. Zatímco desktopové systémy jsou kompletně spouštěny na koncovém zařízení uživatele, webový systém je zpravidla zpřístupněn přes internetový prohlížeč.

U webovém systému mluvíme o přístupovém bodu jako o tzv. *tenkém klientu* (angl. thin client)[11], kterým může být například již zmíněný webový prohlížeč. Systém se vyznačuje zejména tím, že veškeré výpočty a celá aplikační logika neběží na zařízení uživatele, ale na serveru. K tomuto serveru je uživatel připojen právě skrze zmíněného tenkého klienta. Komunikace se serverem většinou probíhá pomocí protokolu HTTP či HTTPS. Společně s logikou aplikace jsou na serveru uložena také uživatelská, případně i systémová data, zpravidla ve formě databáze. Výhoda takových systémů, která vyplývá z uvedeného popisu, je nezávislost na koncovém řízení. S rozvojem rychlého internetu dnes není ani problém s rychlostí odezvy takových systémů. Na druhou stranu nedostupnost serveru, která může být způsobena třetí stranou, může v nevhodnou dobu omezit či úplně zastavit práci uživatele. Z pohledu bezpečnosti je uložení dat mimo vlastní PC pro některé uživatele nepřípustné. Dalším rozhodujícím faktorem může být i nutnost vyšších, často paušálních, výdajů za poskytování serverového úložiště a výpočetního výkonu.

Desktopový systém je aplikace určena a navržena pro cílové zařízení uživatele, na kterém je spuštěna. Zároveň aplikace využívá dostupných prostředků zařízení jako například výkon procesoru, úložiště na lokálním disku, RAM paměť apod. V případě že aplikace potřebuje větší množství výpočetního výkonu, je možné, že zařízení nebude provádět požadované operace dostatečně rychle.

U obou typů systémů je nutné řešit způsob uložení dat. Webové informační systémy vetšinou ukládají svá data do databáze. Tato databáze může být umístěná buď na stejném, nebo opět vzdáleném serveru. U desktopových systémů, se kterými pracuje jeden uživatel, nemusí být ukládaná data nikam zasílána a jsou uložena přímo v zařízení. Toto ovšem neplatí, pokud je nutné, aby byl desktopový systém, zejména tedy jeho data, přístupný pro více uživatelů současně. V takovém případě je vhodné data ukládat také do databáze umístěné na serveru, ke kterému se jednotlivé aplikace připojí a získají, případně uloží, potřebná data.

2.2.1 Technologie pro webový vývoj

V následující části budou popsány aktuální moderní přístupy při vývoji informačních systémů určených pro webové rozhraní. Pro následující text se předpokládá zároveň nutné použití značkovacího jazyk HTML a případně i CSS (kaskádové styly) a skriptovacího jazyka *Javascript*.

PHP & MySQL

Velmi populární kombinací pro vytváření internetových aplikací je spojení programovacího jazyka *PHP* a volně dostupného databázového systému MySQL. Tento přístup je momentálně velmi rozšířen zejména proto, že není závislý na konkrétní platformě. [9][8]

PHP je široce používaný, volně dostupný skriptovací jazyk, zvláště vhodný pro vývoj webových aplikací. Umožňuje také přímé vložení skriptů do značkovacího jazyk HTML.[7]

Nicméně v současné době není jazyk PHP plně objektový, což může znesnadňovat a prodlužovat celý proces vývoje. Existuje však mnoho typů podpůrných frameworků, které mohou naopak vývoj velmi usnadnit, např. *Nette, Symphony* a další.

Databáze MySQL umožňuje použití pro malé i velké aplikace. Používá standart databázového jazyka SQL. MySQL databáze běží na příslušném serveru. Pomocí PHP je možné se k této databázi připojit a provádět potřebné operace. Samozřejmě firma *Oracle*, která jej momentálně vlastní, nabízí i několik rozšířených a vylepšených verzí, která již jsou placená.

ASP.NET

Dalším možným programovacím jazykem je ASP.NET, který je ale na rozdíl od předchozího spojení PHP a MySQL závislý na platformě, na které je spuštěn. Na serveru musí běžet systém od firmy *Microsoft*. Samotný poskytovatel ASP (Microsoft) umožňuje při návrhu architektury aplikace využít návrhový model MVC (model-view-controller). Velkou výhodou ASP je možnost využití frameworku .NET, který obsahuje mnoho knihoven a funkcí, jež mohou vývojáři velmi pomoci.

Výhodou tohoto jazyka ve spojení s frameworkem .NET je možnost využívat volání stejných funkcí a knihoven. Tato vlastnost najde uplatnění zejména tam, kde je vyvíjena jednak webová a jednak desktopová aplikace.

Bootstrap

Dalším používaným a v současné době také velmi rozšířeným frameworkem pro vývoj webu je *Bootstrap*. Na rozdíl od výše uvedených však neřeší aplikační logiku (tzv. backend aplikace), ale způsob zobrazení uživatelského rozhraní (tzv. frontend aplikace). Tento framework využívá zejména jazyků HTML, Javascript a také kaskádových stylů - CSS.

Jeho klíčovou vlastností je poskytnutí a zajištění responzivního designu aplikace. Responzivním design umožňuje vhodně zobrazit webovou stránku na různých zařízeních s odlišným rozlišením displeje či monitoru. V praxi to znamená, že není třeba vytvářet různá zobrazení pro jednotlivá rozlišení, ale framework zajistí, že jsou jednotlivé elementy na stránce přeuspořádány tak, aby vzhled vyhovoval rozlišení displeje, které používá uživatel. [2]

2.2.2 Technologie pro desktopové aplikace

Desktopová aplikace je definována jako aplikace, která běží samostatně v počítači nebo notebooku. Takové aplikace lze aktuálně vyvíjet například v jazyce *Java*, které jsou díky virtualizaci nezávislé na konkrétní platformě. Další možností je použití frameworku *Qt* a jazyka C++, touto volbou lze také dosáhnout nezávislosti na platformě. Pro systémy *Windows*, které jsou v současnosti nejpoužívanějším operačním systémem, je možné vytvářet aplikace například pomocí platformy .NET a některého z dostupných programovacích jazyků. Samozřejmě možností jak vyvíjet desktopové aplikace je mnohem více.

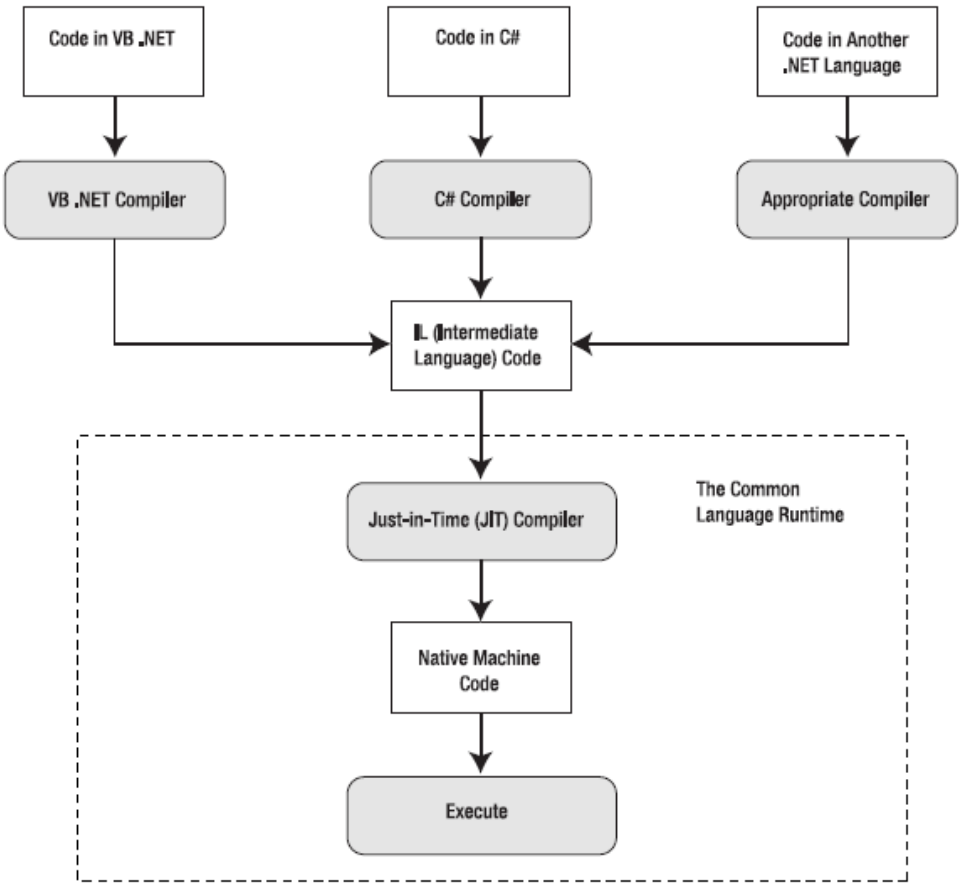
V této podkapitole budou detailněji popsány možnosti vývoje pro operační systém Windows.

Platforma .NET

Framework .NET je rozsáhlá platforma, pomocí které je možné vyvíjet různé druhy softwarových aplikací, zejména mobilní, desktopové nebo webové. Framework je tvořen *Common Language Runtime* (CLR) společně s velkým množstvím knihoven.

Common Language Runtime je prostředí určené pro spuštění managed kódu. Dá se říci, že CLR je obdoba Java Virtual Machine pro platformu .NET. Dále CLR poskytuje několik služeb jako například správu paměti, načítání knihoven, zajištění bezpečnostních služeb, zachytávání vyjímek a další. Pomocí tohoto prostředí je možné vyvíjet aplikace i ve více jazycích zároveň například *C#, F#, Visual Basic* a ještě mnoha dalších.

Při kompilaci zdrojového kódu dochází k jeho převodu do tzv. *managed kódu*. Takto vytvořený kód je zabalen do assembly, která je dvou typů. Buď jako spustitelný kód s příponou *.exe* nebo jako knihovna s příponou *.dll*. Následujícím krokem je reprezentace managed kódu do tzv.*Intermediate language*, který je při čtení CLR konvertován z assembly. Vytvořený Intermediate language je poté pomocí *Just-In-Time kompilátoru* převeden do nativního kódu.



Obrázek 2.2: Schéma Common Language Runtime [15]

Celé schéma kompilace je zobrazeno na obrázku 2.2 na kterém je možné vidět výše popsaný postup kompilace.

Návrhový vzor MVVM

Při návrhu architektury aplikace je vhodné využít některý z dostupných návrhových vzorů. Těchto vzorů architektury je několik, typicky každá technologie doporučuje některou konkrétní.

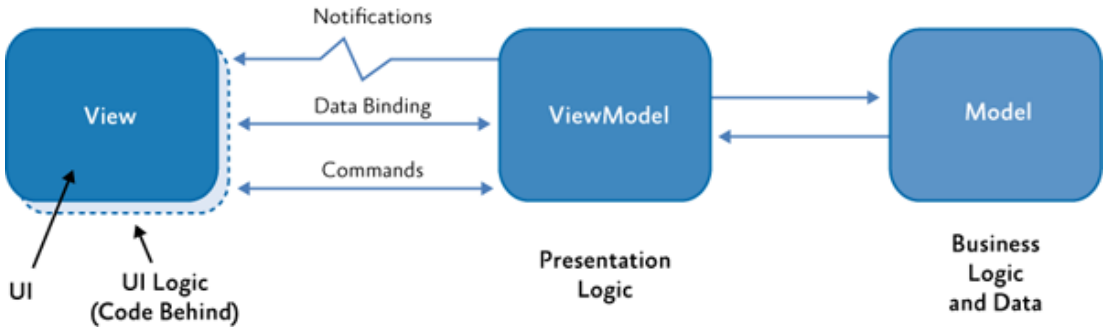
Pro systém Windows je při použití grafického rozhraní WPF (Windows Presentation Foundations), které bude detailněji popsáno níže, doporučeno využít i návrhový vzor MVVM (Model-View-ViewModel). Tento vzor umožňuje oddělení logiky aplikace od uživatelského rozhraní takovým způsobem, aby byly na sobě tyto dvě věci nezávislé.

Model MVVM toto rozdělení umožňuje pomocí tří vrstev (View, ViewModel a Model), které oddělují zobrazení, stav aplikace a data. Zmíněné rozdělení je možné vidět na obrázku 2.3. Vrstva *Model* popisuje veškerá data, se kterými aplikace pracuje. Prakticky vzato zde patří všechny vytvořené třídy s daty a využívané služby, komunikace s dalšími modely apod.

Vrstva *ViewModel* tvoří spojující vrstvu mezi *View* a *Model*. V této vrstvě je uchováván aktuální stav aplikace. Prvky uživatelského rozhraní jsou s touto vrstvou spojeny pomocí *Data Bindingu*, skrze kterého z něj získávají data, nebo naopak ViewModel získává data z těchto prvků.

Poslední vrstvou je vrstva *View*, ve které je veškeré uživatelské rozhraní aplikace a může komunikovat s vrstvou ViewModel.

Tato architektura je podobná architekturám MVC (Model-View-Controller), či MVP (Model-View-Presenter), které také obsahují tři vrstvy. Hlavní výhodou MVVM je ovšem již zmíněná možnost Data Bindingu, který může být obousměrný (mezi View a ViewModel) a také lze zvýšit testovatelnost a rozšiřitelnost takové aplikace. [6]



Obrázek 2.3: Schéma návrhového vzoru MVVM [1]

WPF (Windows Presentation Foundation)

Windows Presentation Foundation je grafický framework pro vytváření formulářových aplikací. Je součástí frameworku .NET a v dnešní době prakticky nahrazuje dřívější technologii *Windows Forms*. K vytváření grafických elementů a modifikaci jejich vlastností využívá značkovací jazyk XAML (Extesible Application Markup Language).

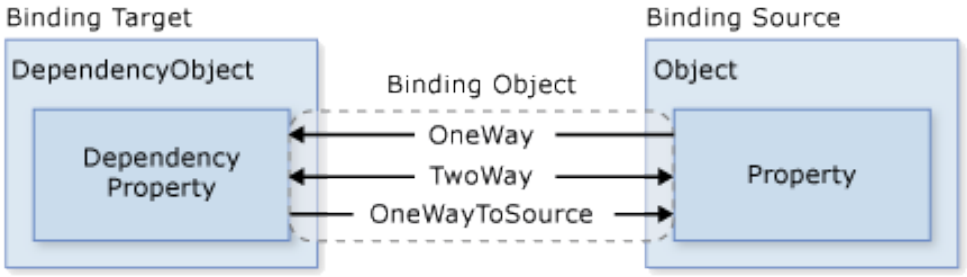
Framework obsahuje širokou škálu připravených grafických prvků. Je však možné si vytvořit i vlastní elementy. Ke změně hodnot a vlastností jednotlivých elementů slouží tzv. *Data binding*. Tento Data binding slouží k propojení vlastností elementů s vlastnostmi objektů. [4]

Na obrázku 2.4 je vidět, že propojení vlastností může být řešeno jedním ze tří možných způsobů. Prvním způsobem je *OneWay* binding, pomocí kterého je možné zdrojovým

objektem změnit vlastnosti grafického elementu, ale vlastnosti zdrojového objektu změnit nelze. Tento způsob je možné použít zejména, pokud chceme data pouze zobrazit.

Druhou možností je *TwoWay* binding, který, jak už anglický název napovídá, umožňuje měnit vlastnosti na obou stranách bindingu. Tento typ lze využít zejména při úpravách uložených hodnot.

Třetím způsob je *OneWayToSource* binding, který je opakem prvního způsobu. Tímto typem je možné změnit vlastnost zdrojového objektu. Typickým příkladem je přepočet hodnot zdrojového objektu na základě akce provedené v uživatelském rozhraní.[3]



Obrázek 2.4: Schéma způsobu data bindingu. [3]

Entity Framework

Entity Framework (EF) je Microsoftem doporučovaná technologie pro přístup k datům. Framework zajišťuje objektově relační mapování, tedy provádí automatický převod objektů do relační databáze a naopak.

Vytváření databáze pomocí toho frameworku je možné třemi způsoby. Prvním přístupem je *Code First* přístup, pomocí kterého se nejprve vytvoří třídy, na jejichž základě jsou následně automaticky vytvořeny tabulky v databázi.

Druhým přístupem je *Model First*, při kterém se nejprve pomocí grafického designeru vytvoří celý model databáze, která je na poté vygenerována. Model databáze je vytvořen v souboru s koncovkou .edmx, který je možné vytvářet v EF Designeru. Třídy k této databázi jsou poté vytvořeny automaticky.

Třetí možností je *Database First*, která se využívá v případě, kdy je potřeba namapovat již existující databázi do tříd v programu. Postup tohoto přístupu spočívá ve výběru existující databáze, které jsou automaticky vytvořeny třídy v programu pro přístup k těmto datům. [5]

Kapitola 3

Sledování výskytu škůdců

V následující kapitole bude popsána možnost získání informací o výskytu škůdců v určeném období. Tyto informace jsou dostupné na webových serverech Ministerstva zemědělství ČR.

3.1 eAgri.cz

Webový portál *eAgri.cz* provozuje Ministerstvo zemědělství ČR. Tento portál poskytuje centrální přístupový bod k informačním zdrojům Ministerstva. Celý web pouze sdružuje další subportály, aby byl přístup pro uživatele co nejjednodušší a měl veškeré informace na jednom místě. Jednotlivé subportály obsahují například web Ministerstva zemědělství, web pozemkového úřadu, informace o lesích, Program rozvoje venkova, informace o dotacích apod.

Nejzajímavějším subportálem je *Portál farmáře*, který může být pro zemědělce velmi přínosný. V této části může uživatel získat informace z nejrůznějších registrů ministerstva, jako například informace z půdního fondu, registru vinic, chmelnic, sadů či registru zvířat. Dále je dostupná evidence přípravků a hnojiv, příjemců dotací apod. Některé z těchto aplikací ovšem vyžadují přihlašovací údaje. Tyto údaje je možné získat registrací zemědělce u Ministerstva zemědělství, což však malí zemědělci, kteří hospodaří bez jakýchkoli dotací, zpravidla nevyužívají. Proto jsou některé aplikace přístupné veřejně.

Pro vytvářenou aplikaci je ovšem nejpodstatnější dostupná aplikace *mapa výskytu škodlivých organismů*, která zobrazuje informace o výskytu škůdců na jednotlivé plodiny v České republice. Tato aplikace bude využita při tvorbě výsledného informačního systému.

3.1.1 Mapa výskytu škůdců

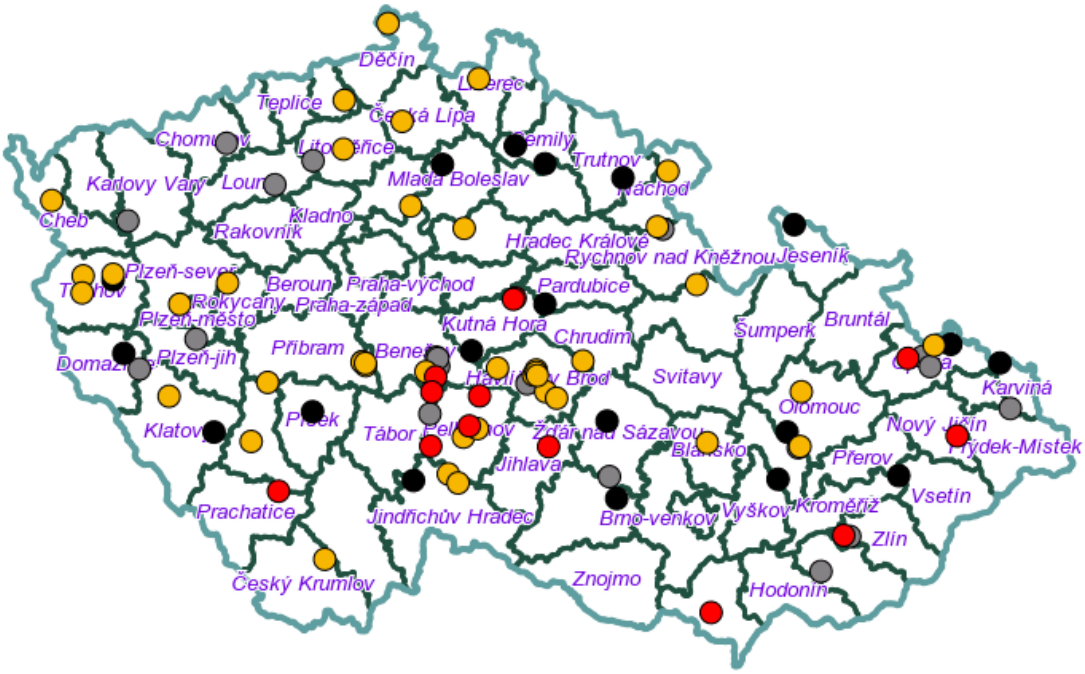
Aplikace pro zobrazení výskytu škůdců je dostupná na webové adrese: http://eagri.cz/public/app/lpisext/public/app/srsmapa/. Pomocí této aplikace eviduje Ministerstvo zemědělství oblasti výskytu jednotlivých škůdců.

Při použití této aplikace nejprve uživatel vybere období, ve kterém požaduje zobrazit škůdce. Zpravidla si uživatel zobrazuje informací z celé zemědělské sezóny, tedy od podzimu do podzimu, proto je tato možnost přednastavena. Po výběru období je nutné vybrat konkrétní plodinu. Na výběr jsou všechny typické plodiny pěstované na území České republiky.

Po tomto výběru a jeho potvrzení je uživatel přesměrován k dalšímu kroku, kdy určí konkrétního škůdce vybrané plodiny. V tomto kroku je možné vybrat i možnost zobrazení pouze pozitivních výsledků. Tento výběr slouží k odfiltrování měření, při kterých nebyl prokázán výskyt daného škůdce.

Na obrázku 3.1 je možné vidět, jakým způsobem je výskyt vybraného škůdce na danou plodinu zobrazen uživateli. Jednotlivé body na obrázku určují místo výskytu. Barva těchto bodů určuje velikost tzv. třídy výskytu, tedy velikost napadení daným škůdcem.

Při kliknutí na dané místo je možné zobrazit i detailnější informace o konkrétním výskytu.



Obrázek 3.1: Ukázka zobrazení výskytu škůdců.

WMS

Celý systém výskytu škůdců běží pomocí služby *OpenGIS® Web Map Service Interface Standard* (zkráceně WMS). Tento standart je open-source a je vyvíjen mezinárodní standardizační organizací *Open Geospatial Consorcium* (OCG).

Tato služba umožňuje zobrazování geografických dat formou rastrových obrázků. Při vykreslování jsou zobrazovány jednotlivé mapové vrstvy, které mohou být při vykreslování překryty přes sebe. Služba umí také pracovat s několika souřadnicovými referenčními systémy angl. *coordinate reference system* (CRS), ke kterému využívá dataset EPSG (Geodetic Parameter Dataset).

Komunikace se službou WMS probíhá pomocí protokolu HTTP a dotazů GET a POST. Tyto požadavky musí mimo jiné obsahovat typ požadavku, formát zobrazované mapy a určení souřadnicového systému. Také je nutné určit tzv. bounding box, což jsou souřadnice, které určují část mapy, která bude zobrazena. [12]

Základní typy požadavků:

- **GetMap** - Tento typ požadavku vrátí obrazová data mapy v určeném grafickém formátu.

- **GetFeatureInfo** - Tento typ vrací ve formátu XML údaje o zadaných souřadnicích. Konkrétně je tento typ použit při získávání detailnějších informací o konkrétním výskytu.

Mapový server pro výskyt škůdců používá **EPSG:102067**, což odpovídá souřadnicovému systému **S-JTSK** (Systém Jednotné trigonometrické sítě katastrální). Tento systém není přímo součástí datasetu, ale v ČR se tento systém využívá, jako by součástí byl. Je to z důvodu, že je tento systém závazným geodetickým referenčním systémem pro Českou republiku. Tento systém používá *Křovákovo zobrazení* a je definován od nultého poledníku Greenwiche.[10]

3.2 Komunikace se serverem

Jelikož je server pro zobrazení výskytu škůdců veřejně dostupný skrze webové rozhraní, je komunikace s tímto serverem rozdělena do několika následujících kroků.

1. Po připojení k serveru je vrácena stránka, na které je seznam dostupných plodin, možnost výběru sezony a upřesnění datumového rozmezí.

2. Nejprve je zvolena plodina a sezóna, respektive rozsah, ve kterém chceme zobrazit výskyt škůdců. Tento rozsah je zpravidla zvolen od podzimu do podzimu.

3. Následně po odeslání dat z předchozího kroku vrátí server další webovou stránku, na které je seznam dostupných škůdců zvolené plodinu a časový rozsah odeslaný v předchozím kroku. V tomto kroku je možné zvolit i možnost zobrazení pouze pozitivních výskytů škůdců.

4. Po odeslání těchto dat dochází na pozadí k získání dat ve formátu JSON, ve kterém server mimo jiné vrátí i identifikační kód *session*, který je potřeba v dalším kroku, aby byl server schopný identifikovat uživatele.

5. Poté je získána mapa ze serveru, na kterém běží služba WMS. Tato mapa je získána pomocí požadavku *GetMap*, který mimo jiné obsahuje i určení formátu, ve kterém má server mapu vrátit. Typicky je mapa získána ve formátu PNG. Dále požadavek určuje, v jakém souřadnicovém systému má server pracovat (v tomto případě EPSG:102067). Díky tomuto určení souřadnicového systému je poté možné pomocí souřadnic určit rohy mapy, kterou chceme získat. Posledními dvěma atributy je určení šířky a výšky obrázku v pixelech, který chceme získat. Je možné také určit vrstvy mapy, díky kterým lze specifikovat podrobnost mapy. V tomto kroku je v požadavku odeslán i identifikační kód *session* pro identifikaci uživatele.

6. Předchozím krokem je získán pouze určený obraz mapy bez zobrazení výskytů škůdce. Proto je zaslán druhý dotaz pro získání obrazu, na kterém jsou barevné body, které určují jednotlivé výskyty. V tomto kroku je nutné odeslání stejných dat jako v předchozím kroku, aby byly získány dva stejné obrazy se stejnými vlastnostmi. Tyto obrazy je poté nutné vzájemně překrýt, aby vznikl obraz s podkladem mapy, na kterém jsou zobrazeny barevné body určující výskyt škůdce.

Ukázku obrazu, který je získán uvedenými kroky je možné vidět na obrázku 3.1. Přiblížení mapy probíhá úpravou souřadnic rohů mapy při odeslání nového požadavku.

Kapitola 4

Analýza požadavků systému

V této kapitole jsou detailně popsány veškeré požadavky na systém pro drobné zemědělce.

Výsledná aplikace je vytvářena pro drobné zemědělce a je určena především pro zefektivnění agendy spojené se zemědělstvím, zejména s objednávkami a s evidencí příjmů a výdajů. V neposlední řadě bude systém umožňovat zobrazení výskytu škůdců daných vybraných plodin v dané lokalitě. Celý systém lze tedy rozdělit do následujících částí.

- **Správa příjmů:** Možnost zadávání a případné editace příjmů ze zemědělské činnosti uživatele.
- **Správa výdajů:** Umožnění vložení a editace výdajů do zemědělské činnosti uživatele.
- **Správa objednávek:** Možnost vkládání a úprav objednávek na určené komodity, které zemědělec prodává.
- **Správa zákazníků:** Správa odběratelů, kteří od zemědělce nakupují komodity nebo jeho služby.
- **Zobrazení škůdců:** Zobrazení aktuálních výskytů škůdců na určené plodiny.
- **Vedení zápisků:** Možnost vedení zápisků k jednotlivým strojům a zařízením.

4.1 Správa příjmů

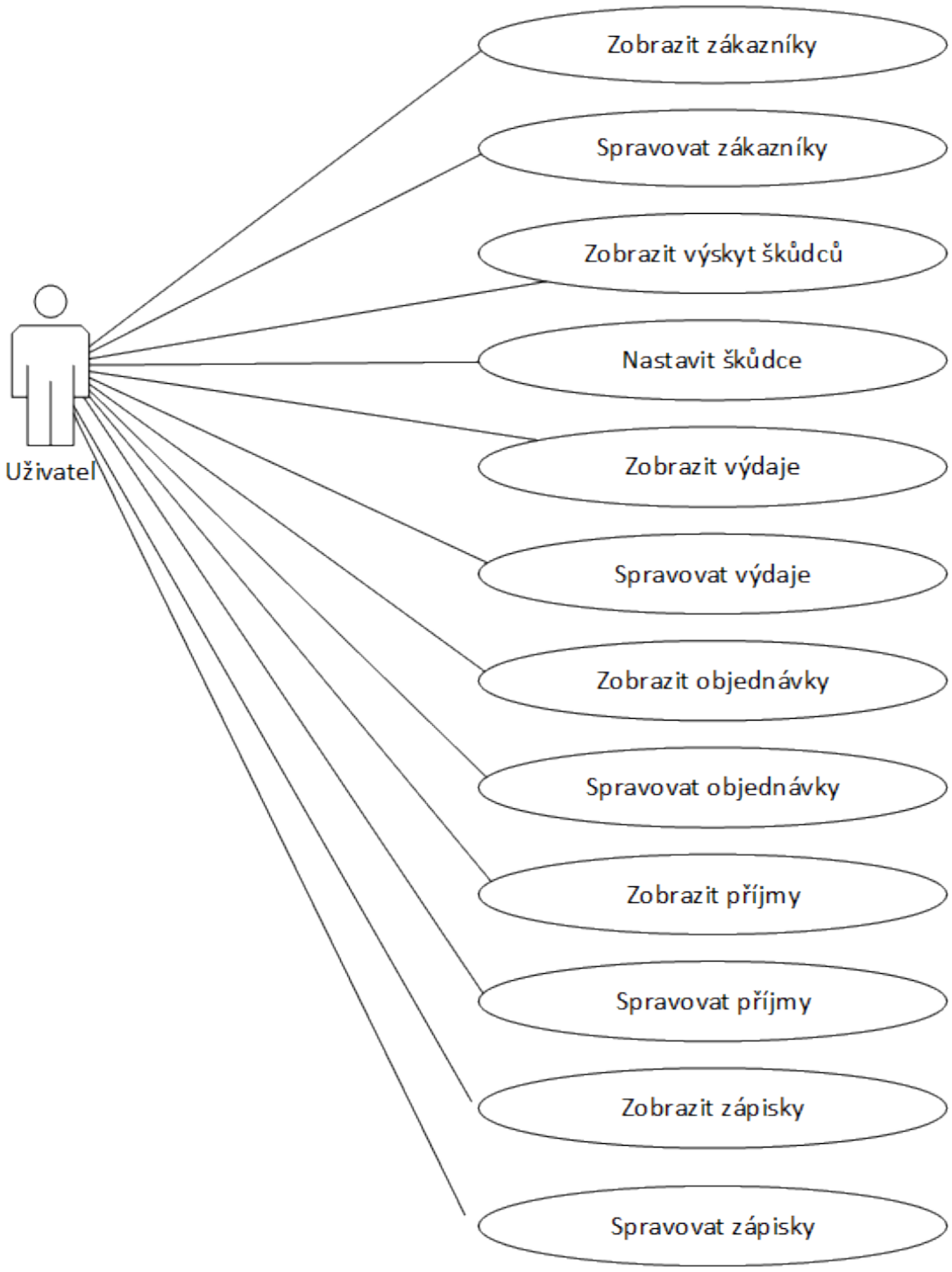
Příjmy jsou hlavním finančním zdrojem každé podnikatelské činnosti. Zemědělská činnost malých zemědělců je také svým způsobem podnikatelská činnost, byť není často nikde oficiálně vedena.

V této sekci musí tedy systém umožňovat zadávání příjmů uživatele. Příjem malého zemědělce lze rozdělit na dva druhy. Příjem za služby a příjem za prodaný produkt.

Zemědělská služba může být například pronajmutí stroje, případně může být v rámci služby využíváno i lidské činnosti zemědělce. Například posekání louky, lisování sena, provedení podzimní orby a podobné činnosti související s obhospodařováním pole.

U každého příjmu je nutné zaevidovat, za jakou činnost byl příjem získán a od koho. Dále je u některých příjmů vhodné uvést poznámku s dodatečnými informacemi. Samozřejmostí je možnost přidání hodnoty příjmu v českých korunách.

Přijatou platbu uživatel zaeviduje v systému v sekci příjmy. Díky tomu získá rychlejší přehled o svých příjmech.



Obrázek 4.1: Diagram případů použití. Pod pojmem "Spravovat"jsou obsaženy CRUD operace.

4.2 Správa výdajů

Nezbytnou součástí každého podnikání jsou i výdaje vztahující se k podnikatelské činnosti. Výdaje v zemědělství jsou například nákup nafty, oprava strojů, nákup nových strojů a podobně.

Při bližším zkoumání výdajů je možné zjistit, že výdaje se z velké části víceméně opakují, což platí hlavně pro spotřební materiál, např. nákup nafty, nákup nových nožů do rotačních sekaček atp.

V systému budou výdaje sjednoceny na jednom místě, kde uživatel při zadávání nového výdaje bude mít možnost z vybrat (zapsat) do jaké kategorie (účelu) výdaj spadá, dále jeho hodnotu a případně další poznámky k zadávanému výdaji.

Stejně jako v předchozím případě je účel této vlastnosti v systému, aby měl uživatel možnost lepšího a rychlejšího přehledu o výdajích.

4.3 Správa objednávek

Evidence objednávek je v kterékoliv oblasti podnikání nutnou záležitostí. Proto bude tuto vlastnost vhodné implementovat i do výsledného systému.

Každá objednávka se vztahuje k určitému druhu zboží. V zemědělství například česnek, cibule, mrkev atp. Objednávka obsahuje jméno zákazníka a množství objednaného zboží. Pro uživatele je vhodné, aby každá objednávka obsahovala i určení, zda byla vyexpedována a v případě, že byla zaplacena, i částku, kterou zákazník zaplatil.

Je nutné, aby uživatel určil, zda je pro něj vhodné prodej služby nebo produktu evidovat formou objednávek, nebo pouze formou jednorázových příjmů.

U každé objednávky je nutné mít možnost objednávku upravit, případně i smazat. Dále je nutné objednávky dělit i podle roku, ke kterému se vztahují.

4.4 Správa zákazníků

Aby byla práce se systémem pohodlná, je velmi vhodné mít veškeré náležitosti na jednom místě. Proto bude implementována i evidence a správa zákazníků. U každého zákazníka je nutné evidovat jméno, příjmení, telefonní číslo, email a případně některé dodatkové informace formou poznámky.

Kontakt na zákazníka je v dnešní době jednou z ceněných informací v podnikání. Díky správné evidenci je možné oslovit každý rok zákazníka, u kterého je velký předpoklad, že bude mít o produkt opětovný zájem.

V systému bude proto možnost zobrazení všech zákazníků a jejich úpravy, případně smazání.

4.5 Zobrazení výskytu škůdců

Mít možnost rychle zjistit, zda a v jakém množství se v okolí vyskytuje některý ze škůdců na plodiny, které zemědělec pěstuje, je velkou konkurenční výhodou. Proto bude systém umožňovat následující vlastnost.

Uživatel bude mít možnost vybrat na mapě místo a tím určit, ve které části republiky (nebo oblasti) chce sledovat škůdce. Současně s tímto výběrem určí také plodinu, na které chce sledovat škůdce.

Následně bude mít uživatel možnost zobrazit okno s grafickými informacemi o výskytu škůdců jednotlivých plodin. Tyto informace se budou zobrazovat pouze na části mapy vybrané uživatelem. Díky tomu se uživateli zobrazí pouze informace, které jsou pro něj potřebné.

K tomuto sledování bude systém využívat služeb Ministerstva zemědělství, zejména jeho systém na sledování škůdců, který je zmíněný v předchozí kapitole. Z tohoto systému budou na základě vložených informací od uživatele systematicky získávány obrazové informace o výskytu škůdců.

Samozřejmostí je umožnění smazání nastaveného záznamu na sledování škůdců.

4.6 Vedení zápisků

Každý dobrý hospodář si vede svůj zápisník, ve kterém má zapsané pro něj podstatné informace. Například kdy byl který stroj zakoupen, jaké vady se u něj projevily, který výrobek od kterého dodavatele je kvalitní, či nikoliv a podobně.

Při bližším zkoumání těchto zápisků lze zjistit, že je možné dělení do kategorií. Například zápisky vztahující se k určitému stroji či k určitým zemědělským událostem nebo činnostem a podobně.

Z těchto důvodů bude systém obsahovat i možnost vedení takových zápisků. Uživatel nejprve vybere nebo vytvoří novou kategorii, do které bude moci vložit textový zápisek.

Cílem této možnosti je přechod od papírové evidence do elektronické tak, aby bylo možné v budoucnu s těmito informacemi lépe pracovat a rychleji požadovanou informaci dohledat.

Kapitola 5

Návrh systému

Dalším krokem po vytyčení hranic systému a jeho kompletní specifikaci následuje v této kapitole návrh databáze pro uložení potřebných dat vycházejících ze specifikace požadavků.

Dále bude v této kapitole popsán grafický návrh aplikace, zejména pak jednotlivé obrazovky, které uživateli umožní práci s tímto systémem.

Nakonec bude popsán a zdůvodněn výběr implementační technologie celého systému a veškeré technologie, které budou při vývoji využity.

5.1 Návrh schématu databáze

Důležitou částí návrhu aplikace je bezpochyby návrh schématu databáze. Proto byly vytvořeny následující konceptuální ER (Entity Relationship) diagramy. ER diagram se skládá z entitních množin, které označují věci, nebo objekty reálného světa, o kterých chceme uchovávat informace. Entitní množina je tedy souhrn entit stejného typu. Každá entita obsahuje atributy. Tyto atributy určují vlastnosti dané entity a jejich hodnoty chceme mít uloženy.

Na základě požadavků je zřejmé, že všechny informace, které chce mít uživatel uloženy, se vždy vztahují k jednomu kalendářnímu roku. Proto je nutné mít tuto vlastnost zanesenou i v databázi, aby došlo k oddělení informací vztahujících se k jednotlivému roku. U této entity je také vhodné mít k dispozici vlastnost, která určuje, který rok je nastaven jako aktuální. Tuto vlastnost bude poté možné vhodně využít při implementaci.

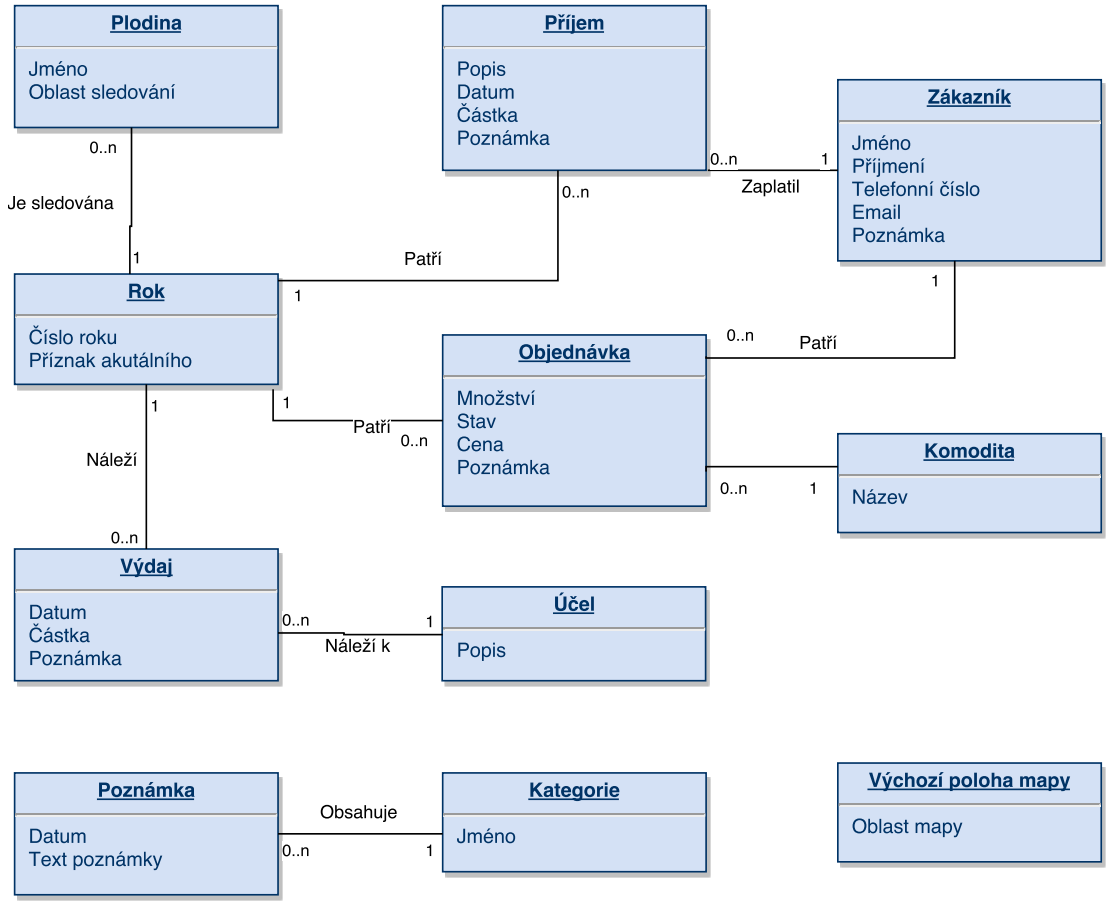
Výsledný konceptuální ER diagram je možné vidět na obrázku 5.1, na kterém jsou znázorněné vztahy mezi jednotlivými entitami. Ke každému konkrétnímu roku může být přiřazeno několik výdajů, příjmů, objednávek a plodin. Tento vztah je na diagramu znázorněn vazbami 1:M k jednotlivým entitním množinám.

Vztah 1:M znázorňuje situaci, kdy k jedné entitě může být vázáno několik entit stejného typu. V případě navrženého konceptuálního ER diagramu je tímto vztahem znázorněna vlastnost, kdy je v rámci jednoho kalendářního roku evidováno několik příjmů, výdajů, objednávek a sledovaných plodin.

Stejný typ vazeb je pak použit i u spojení objednávky a zákazníka, kdy jeden zákazník může mít v systému několik objednávek. Stejně je tomu i ve vztahu Komodita - Objednávka, kde je znázorněno, že na jednu komoditu může být v systému několik objednávek.

Vztah mezi entitními množinami Příjem a Zákazník znázorňuje možnost, kdy je možné od jednoho zákazníka evidovat několik příjmů v daném roce.

Dále vztah mezi entitními množinami Výdaj a Účel, kdy je znázorněno, že k jednomu účelu může být několik výdajů.



Obrázek 5.1: Konceptuální návrh databáze.

Jak bylo zmíněno ve specifikaci požadavků, systém má umožňovat vkládání zápisků vztahujících se k určitým kategoriím. Toto je v ER diagramu vyobrazeno entitami *Poznámka* a *Kategorie* a jejich vztahem 1:M, tedy, že k jedné kategorii může spadat několik poznámek.

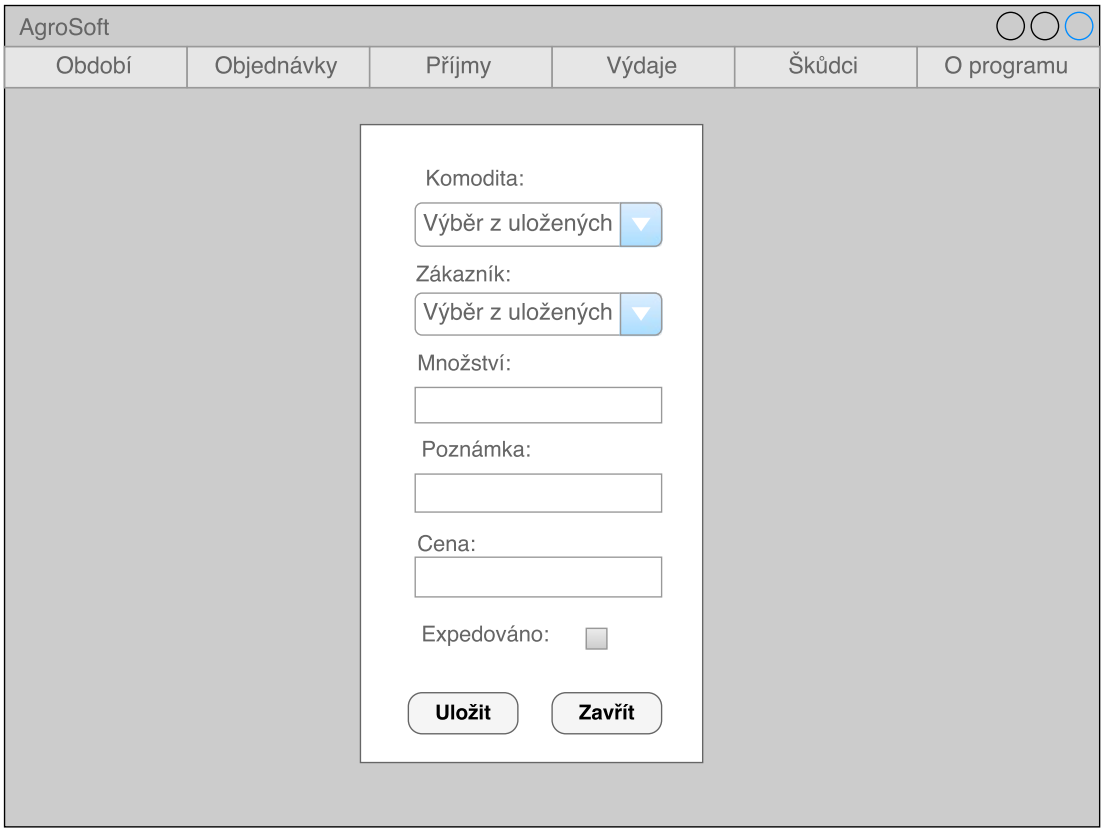
Poslední důležitou entitou v tomto diagramu je entita *Výchozí oblast mapy*, která slouží pro uložení oblasti mapy, kterou si uživatel zvolil jako výchozí. Tato entita je zde zanesena z důvodu, že uživatelé budou chtít zpravidla sledovat stejnou oblast mapy pro různé plodiny.

5.2 Uživatelské rozhraní

V průběhu navrhování je potřeba určit veškeré obrazovky, které bude výsledná aplikace obsahovat. Jednotlivé obrazovky jsou navrženy tak, aby obsluhovaly pouze jednu konkrétní agendu z celé množiny všech dostupných v aplikaci. Díky tomu je možné lépe omezit případné chyby uživatele, než kdyby bylo více funkcí na jedné obrazovce.

- **Obrazovky pro správu roku:** Tyto obrazovky budou umožňovat jednak přidání (založení) nového roku, a jednak případné smazání celého roku i se všemi daty vztahujícími se k danému roku.

- **Obrazovka pro přidání zákazníka:** Tato obrazovka bude obsahovat formulář pro přidání informací o novém zákazníkovi. Přidání zákazníka bude potvrzeno kliknutím na tlačítko.
- **Obrazovka pro výpis zákazníků:** Obrazovka, na které se zobrazí tabulka všech dostupných zákazníků společně s možností výběru konkrétního zákazníka a přechodu na obrazovku pro správu vybraného zákazníka.
- **Obrazovka pro správu zákazníka:** Obrazovka umožňující úpravu informací vybraného zákazníka, případně i jeho smazání.
- **Obrazovka pro přidání výdaje:** Obrazovka umožní přidat výdaj do systému. Uživateli zobrazí formulář, ve kterém vyplní potřebné údaje a následně vloží záznam do systému kliknutím na tlačítko.
- **Obrazovka pro výpis všech výdajů:** Na této obrazovce bude mít uživatel možnost zobrazení všech výdajů formou tabulky. Stejně jako u zákazníků bude mít uživatel možnost vybrat konkrétní výdaj a přejít na obrazovku pro jeho úpravu či smazání.
- **Obrazovka pro správu výdaje:** Obrazovka umožní uživateli upravit údaje již uloženého výdaje v systému.
- **Obrazovka pro výpis příjmů:** Obrazovka pro zobrazení veškerých příjmů v daném roce formou tabulky.
- **Obrazovka pro přidání příjmu:** Obrazovka zobrazí formulář pro vložení potřebných informací k novému příjmu dle specifikace.
- **Obrazovka pro úpravu příjmu:** Na této obrazovce bude uživateli umožněno změnit údaje již uloženého příjmu.
- **Obrazovka pro přidání nové objednávky:** Obrazovka nabídne uživateli formulář pro přidání nové objednávky. Součástí formuláře je i menu se jmény zákazníků, aby uživatel mohl jednoduše přiřadit objednávku konkrétnímu zákazníku.
- **Obrazovka pro výpis objednávek:** Tato obrazovka nabídne zobrazení všech objednávek na danou komoditu formou tabulky. V této tabulce bude zobrazeno u každé objednávky jméno zákazníka, objednané množství, informace zda byla objednávka expedována a tlačítko pro možnost smazání a úpravy objednávky.
- **Obrazovka pro úpravu objednávky:** Obrazovka nabídne možnost upravení údajů ve vybrané objednávce.
- **Obrazovka pro výběr monitorování škůdců:** Na této obrazovce uživatel vybere z menu konkrétní plodinu a dále na zobrazené mapě České republiky vybere oblast, pro kterou chce sledovat škůdce plodiny.
- **Obrazovka pro zobrazení výskytu škůdců:** Zde budou uživateli zobrazeny náhledy pro každou plodinu ve vybrané oblasti, společně s vyznačením výskytu škůdce dané plodiny. V případě že jedna plodina má více škůdců, bude zvlášť pro každého škůdce zobrazen samostatný náhled.



Obrázek 5.2: Wireframe návrh obrazovky pro přidání nové objednávky.

- **Obrazovka pro výběr kategorie a zobrazení zápisků:** Jelikož může existovat několik kategorií zápisků, uživatel na této obrazovce nejprve vybere konkrétní kategorii a následně dojde k zobrazení zápisků v této kategorii. Na této obrazovce bude také tlačítko pro přidání nového zápisku či kategorie.
- **Obrazovka pro vložení zápisku:** Obrazovka zobrazí jednoduchý formulář pro přidání textového zápisku k dané kategorii.

Uvedené obrazovky obsahují pouze ty nejdůležitější obrazovky pro práci se systémem. Výčet neobsahuje vedlejší obrazovky, například obrazovku pro přidání nového roku či přepnutí aktuálního roku apod.

5.3 Výběr implementační technologie

V následující podkapitole budou uvedeny technologie, které byly vybrány na základě požadavků.

5.3.1 Druh aplikace

Celý systém je vyvíjen na základě požadavků od zadavatele (v sekci testování označen jako Uživatel A) a jeho požadavkem bylo vytvořit aplikaci jako desktopovou. Tento jeho požadavek nebyl na začátku vývoje aplikace znám a byl upřesněn až v průběhu analýzy požadavků

a návrhu celého systému. Při určení druhu aplikace bylo zohledněno i to, že se systémem pracuje zpravidla pouze jeden uživatel. Vytvoření aplikace jako desktopové zadavatel požadoval také proto, že v současné době nevyžaduje přístup k datům přes internet.

5.3.2 Databáze

Aby byla zajištěna možnost budoucího rozšíření výsledné aplikace, byla pro ukládání uživatelských dat zvolena databáze MSSQL. S touto databází je možné pracovat i v případě vývoje webového informačního systému. Díky zvolení této databáze by bylo možné v takovém případě pomocí technologie ASP.NET implementovat systém v prohlížeči nad stávající databází bez nutnosti její úpravy či konverze již uložených dat.

5.3.3 Programovací jazyk

Hlavním programovacím jazykem byl zvolen jazyk C# na platformě .NET. Díky tomu je sice aplikace závislá na platformě Windows, kterou pro svůj běh potřebuje, ale jelikož je to v současnosti nejrozšířenější operační systém, toto omezení nevadí. Pro zlepšení práce s databází bude využit Entity framework, který je popsán ve druhé kapitole.

5.3.4 Uživatelské rozhraní

K tvorbě grafického uživatelského rozhraní bude použita technologie WPF, která byla popsána výše. Celá aplikace pak bude rozdělena do vrstev definovaných návrhovým vzorem MVVM (také zmíněno ve druhé kapitole). Toto rozdělení je přímo doporučováno při využití technologie WPF.

Kapitola 6

Implementace systému

V následující kapitole budou popsány implementační detaily vytvořeného systému. Nejdříve je popsán způsob vytvoření a funkce jednotlivých vrstev aplikace. Dále se kapitola detailně věnuje způsobu získávání dat o výskytech škůdců z portálu Ministerstva zemědělství ČR, konkrétně ukazuje způsob komunikace mezi klientem a serverem. V neposlední řadě se také věnuje způsobu zobrazování dat pomocí WPF.

Implementace probíhala ve vývojovém prostředí Microsoft Visual Studio Enterprise 2015. Pro rychlejší a snadnější vývoj bylo využito i rozšíření Resharper od firmy JetBrains.

6.1 Vrstvy aplikace

Celá aplikace je v prostředí Visual Studia vytvořena jako jedno řešení (Solution), které v případě Visual Studia představuje strukturu pro organizování jednotlivých projektů. Každá vrstva použitého návrhového vzoru MVVM představuje jeden projekt. Součástí řešení jsou samozřejmě i jiné projekty, například pro práci s databází či komunikaci se serverem.

Všechny projekty obsažené v řešení, kromě projektu implementujícího vrstvu View, jsou vytvářené jako knihovny. Tyto projekty jsou při překladu transformovány do DLL (Dynamic Link Library) knihoven, které využívá aplikace při svém spuštění.

Vrstva View je v řešení zahrnuta jako spustitelná aplikace systému Windows. Tato vrstva slouží pro spuštění grafického rozhraní. Blíže bude popsána v některé z následujících podkapitol.

6.1.1 Model

Vrstva Model je v řešení implementována jako knihovna *Agrosoft.Model*. Tato knihovna je zodpovědná za definici dat, která budou uložena v databázi. Třídy obsažené v této knihovně jsou následně využity pomocí Entity Frameworku k tvorbě databázového schématu. Schéma databáze je vytvořeno automaticky pomocí tzv. *code first* přístupu.

Díky tomuto přístupu je možné rychle promítnou případné změny v definici tříd do databázového schématu. Jednotlivé vlastnosti tříd jsou při vytváření databáze transformovány na konkrétní sloupce v tabulkách. Aktualizaci schématu je možné spustit pomocí příkazu v příkazovém řádku Visual Studia. Při této aktualizaci dochází k vytváření tzv. migrací (angl. migrations), což jsou třídy definující změny ve schématu databáze. Díky těmto migracím je možné se při vývoji případně vracet k některým předchozím schématům, což je z programátorského hlediska velká výhoda.

Knihovna *AgroSoft.Model* obsahuje adresář *Base*, který má dva podadresáře *Interface* a *Implementation*. V adresáři Interface je umístěna definice rozhraní *IModel*, které definuje, že každá třída implementující toto rozhraní musí obsahovat vlastnost *Id*, která je typu *Guid*. Zároveň toto rozhraní dědí z rozhraní *INotifyPropertyChanged*, které slouží k upozornění klienta při změně hodnoty vlastnosti. Datový typ Guid (globally unique identifier) reprezentuje globální unikátní identifikátor, který je generován automaticky. Takto generovaný identifikátor je pak možné využít jako primární klíč v jednotlivých tabulkách v databázi.

Adresář *Implementation* obsahuje definici třídy *BaseModel*, která slouží jako bázová třída pro ostatní třídy ve vrstvě Model. Tato třída implementuje výše zmíněné rozhraní *IModel* a také metodu *OnPropertyChanged*, která popisuje chování při změně hodnoty vlastnosti objektu.

V dalších adresářích projektu *Agrosoft.Model* jsou třídy definující vlastnosti jednotlivých entitních množin. Každá z těchto tříd dědí ze třídy *BaseModel*, čímž je zajištěno, že každý objekt bude obsahovat vlastnost *Id* s globálním unikátním identifikačním číslem. Ostatní vlastnosti tříd se odvíjí od navrženého konceptuálního ER diagramu.

Na následující ukázce zdrojového kódu je zobrazena třída definující entitu *Rok* z konceptuálního diagramu.

```
public class YearModel : BaseModel
{
   public int Year { get; set; }
   public bool IsActual { get; set; }

   public virtual ObservableCollection<OutcomeModel> Outcomes { get; set; }
   public virtual ObservableCollection<CropModel> Crops { get; set; }
   public virtual ObservableCollection<IncomeModel> Incomes { get; set; }
   public virtual ObservableCollection<OrderModel> Orders { get; set; }

   public YearModel()
   {
      Year = 0;
      IsActual = false;
   }
}
```

Zdrojový kód 6.1: Ukázka třídy definující entitu

Ve výše uvedené ukázce kódu 6.1 jsou definovány čtyři vztahy 1:M a dvě vlastnosti *Year* a *IsActual*. Samozřejmě ve výsledku je také zahrnuta vlastnost *Id* z bázové třídy *BaseModel*. Vztah 1:M je pro Entity Framework definován jako kolekce entit. V této ukázce je jako kolekce použita *ObservableCollection*. Volba této kolekce je z důvodu automatické notifikace při její změně, což z velké části usnadňuje celý vývoj. Na základě konceptuálního návrhu jsou tedy vytvořeny vazby mezi entitami Rok-Výdaj, Rok-Plodina, Rok-Příjem a Rok-Objednávka.

Obdobně jako na ukázce jsou vytvořeny i ostatní třídy. Způsob ORM (object-oriental mapping) který umožňuje automatické mapování relačních dat do doménových objektů zajišťuje Entity Framework. Popis tohoto mapování a práce s databází bude popsána v podkapitole níže.

Celá vrstva Modelu pouze definuje vlastnosti entit a neobsahuje žádnou logiku pro práci s těmito daty. O logiku se stará vrstva *ViewModel*, která je popsána v následující podkapitole.

6.1.2 ViewModel

Následující vrstva *ViewModel* slouží ke spojení vrstev *Model* a *View*. Tato vrstva je v řešení (Solution) implementována jako projekt *AgroSoft.ViewModel*. Hlavním účelem této vrstvy je udržovat stav aplikace a zpřístupňovat data v určitém formátu pro vrstvu View.

Tento projekt obsahuje dva adresáře. Prvním adresář s názvem *Base*, který zahrnuje adresáře *Command*, *Interface* a *ViewModel*. Tato struktura adresářů bylu zvolena z důvodu lepší orientace ve zdrojových souborech při provádění případných pozdější změn.

Adresář Command obsahuje definici třídy *CommandBase*. Tato třída implementuje rozhraní *ICommand* z frameworku .NET. Dále využívá tzv. genericitu, která slouží k určení datového typu vlastnosti proměnné až při vytváření instance dané třídy. Tímto generickým parametrem je předáván datový typ konkrétního ViewModelu. Způsob použití této třídy bude popsán níže.

Adresář Interface obsahuje definici rozhraní *IObservableViewModel*. Toto rozhraní obsahuje metodu *UpdateActualYear*, která jako parametr přijímá instanci třídy *YearModel*. Toto rozhraní implementují třídy v této vrstvě, které jsou závislé na informaci, který rok je nastaven jako aktuální. Aktuální rok si může zvolit uživatel a v případě změny aktuálního roku na jiný je nutné, aby některé objekty na tuto akci reagovaly patřičným způsobem (např. přepočet hodnot, změna dat apod.). Konkrétní třídy implementující rozhraní budou popsány níže.

Posledním adresářem je ViewModel, ve kterém jsou definovány dvě třídy *ViewModelBase* a *ViewModelCollection*. Třída *ViewModelBase* je definovaná jako abstraktní a slouží jako základní bázová třída. Třída implementuje rozhraní *INotifyPropertyChanged*, které umožňuje notifikaci při změně některé z vlastností, a *IDisposable*, které dovoluje spravovat využívané zdroje při zrušení objektu. Důležitou vlastností třídy je proměnná *Service*, která je datového typu *Repository*, kterému je předán generický parametr. Detailnější informace o tomto datovém typu jsou popsány v podkapitole 6.1.4. Pro pochopení v tom bodě postačí informace, že datový typ *Repository* umožňuje práci s databází. Další třída ViewModelCollection, která je také v tomto adresáři, dědí z uvedené třídy ViewModelBase a jak už její název napovídá, slouží pro práci s kolekcí dat. Její klíčovou vlastností je proměnná *Items* datového typu *ObservableCollection*, která obsahuje načtená data z databáze. Dalšími vlastnostmi jsou také proměnné *SelectedItem*, *NewItem* a *EditedItem*, které slouží pro výběr, vytvoření či editaci některé položky z kolekce.

Druhým adresářem v projektu je adresář *Commands*, kterému je věnována následující sekce.

Commands

V tomto adresáři jsou do dalších adresářů rozděleny třídy implementující tzv. Commands. Command (příkaz) vychází z návrhového vzoru MVVM. Pomocí těchto příkazů komunikují mezi sebou vrstvy *View* a *ViewModel*. Typicky se pomocí nich provádí změna stavu jednotlivých ViewModelů. Klasickým případem je reakce na stisk tlačítka v uživatelském rozhraní.

Každá třída implementující některý z příkazů dědí ze třídy *CommandBase*, která byla popsána výše. Při tom je této bázové třídě předán i typ určující ViewModel, ke kterému se příkaz vztahuje. Na následující ukázce zdrojového kódu 6.2 je zobrazena třída implementující příkaz pro úpravu údajů zákazníka. Zde je vidět předání datového typu *CustomerViewModel* bázové třídě *CommandBase*. Nejdůležitější částí každého příkazu je metoda *Execute*, která je volána při jeho provedení. V této metodě uvedené na ukázce je vybraný zákazník,

```
public class EditCustomerCommand : CommandBase<CustomerViewModel>
{
   public EditCustomerCommand(CustomerViewModel viewModel) : base(viewModel)
   {
   }

   public override void Execute(object parameter)
   {
      try
      {
         ViewModel.SelectedItem.OnPropertyChanged("Name");
         ViewModel.SelectedItem.OnPropertyChanged("Surname");
         ViewModel.SelectedItem.OnPropertyChanged("PhoneNumber");
         ViewModel.SelectedItem.OnPropertyChanged("Email");
         ViewModel.SelectedItem.OnPropertyChanged("Comment");
         ViewModel.SaveData();
         ViewModel.StatusViewModel.Status = "Zakaznik byl upraven!";
      }
      catch (Exception)
      {
         ViewModel.StatusViewModel.Status = "Zakaznika se nepodarilo upravit!";
      }
   }
}
```

Zdrojový kód 6.2: Ukázka třídy definující příkaz

který je upravován, označen v proměnné *SelectedItem*. Aby se změněné informace o zákazníkovi promítly i do uživatelského rozhraní, je potřeba nejprve vynutit notifikaci o změně každé z jeho vlastností. Zde tedy voláním metody *OnPropertyChanged("název proměnné")*. Následně je volána metoda *SaveData*, která zajistí uložení změněných dat do databáze. Nakonec je nastaven text s informací o provedené akci do StatusViewModelu, který slouží k zobrazování informací o stavu aplikace. V případě že provádění některé z těchto činností způsobí vyvolání výjimky (zejména při ukládání dat do databáze), je tato výjimka zachycena v bloku *catch*, ve které je nastaven text, který se zobrazí uživateli.

Mimo adresáře jsou v projektu obsaženy třídy definující pro každou třídu z vrstvy *Model* samostatnou třídu spadající do vrstvy *ViewModel*. Pro tyto třídy je bázovou třídou zvolena *ViewModelCollection*, což znamená, že veškerá data z databáze jsou načtena do kolekcí. Zbývající třídy ViewModelů, které je možné v projektu nalézt, mají bázovou třídu *ViewModelBase*.

IObservableViewModel

Některé třídy také implementují rozhraní *IObservableViewModel*, které zajišťuje jednotný způsob komunikace s instancemi těchto tříd. Toto rozhraní implementují následující třídy *IncomeViewModel*, *OutcomeViewModel*, *OrderViewModel* a *CropViewModel*. Tyto třídy slouží k zobrazování dat, které jsou závislé na zvoleném kalendářním roku. Jelikož pro přehlednost aplikace není vhodné zobrazovat veškerá data ze všech kalendářních let, je právě tímto způsobem zajištěno, že při změně kalendářního roku je tento rok předán jednotným způso-

bem všem instancím, které jsou na zvoleném roce závislé. Při vyvolání těchto metod dojde v daných instancích k nastavení (vytřídění) dat, která jsou relevantní pro tento rok.

6.1.3 View

View je poslední vrstvou z návrhového vzoru MVVM. Tato vrstva je na rozdíl od předchozích dvou projektů implementována v projektu *Agrosoft.View* jako WPF aplikace. V této podkapitole bude popsána zejména struktura projektu a komunikace s nižší vrstvou. Tvorbou grafického rozhraní a umístění jednotlivých prvků na obrazovce se věnuje podkapitola 6.3.

Projekt obsahuje čtyři adresáře. *Controls*, *Converters*, *Images* a *Windows*. Dále obsahuje hlavní okno aplikace v souboru *MainWindow.xaml*, ve kterém je definováno základní rozvržení celé aplikace, zejména tedy umístění menu, zobrazovací části a stavového řádku. Každý prvek uživatelského rozhraní (okno, komponenta atd.) může mít nastaven tzv. *DataContext*, kterým je možné definovat zdrojový objekt a následně pomocí tzv. *DataBindingu* získat (a zobrazit) hodnoty jeho vlastností.

Zobrazovací část v aplikaci je řešena formou vytvořených komponent v adresáři *Controls*. Při výběru položky z menu aplikace dojde k zobrazení dané komponenty v zobrazovací části okna. Tyto vytvořené komponenty zpravidla slouží pouze k zobrazování dat z ViewModelů.

Zadávání nových hodnot, úprava stávajících, případně odsouhlasení smazání je řešeno formou dialogových oken. Tato okna jsou umístěna v projektu v adresáři *Windows*. Okna jsou zobrazena při vykonání nějaké akce (např. stisk tlačítka) v aktuálně zobrazené komponentě. Při otevření dialogového okna je vypočítána jeho pozice na střed hlavního okna aplikace. Dále je upravena neprůhlednost (Opacity) hlavního okna, čímž dojde k jeho ztmavení. Při uzavírání okna je neprůhlednost vrácena na původní hodnotu. Tato funkcionalita je implementována z důvodu zajištění lepší uživatelské přehlednosti.

Adresář *Images* obsahuje obrázky, které jsou použity jako ikony v menu aplikace a také ikony pro navigaci při výběru části mapy. V posledním adresáři *Converters* jsou definovány třídy, které implementují tzv. hodnotové konvertory. Tyto konvertory jsou využívány v technologii WPF v případě, že je potřeba změnit zobrazovanou hodnotu, nebo její datový typ. V níže uvedeném příkladu, je zobrazen příklad převodu datového typu *bool* na řetězce obsahující slova “Ano ”nebo “Ne ”. Tyto konvertory jsou používány při *DataBindigu*.

```
public class BoolToCzechStringConverter : IValueConverter
{
   public object Convert(object value, Type targetType, object parameter, CultureInfo
       culture)
   {
      return (bool)value ? "Ano" : "Ne";
   }
}
```

Zdrojový kód 6.3: Ukázka konvertoru

Na další ukázce je zobrazen zobrazen způsob *DataBindingu* i s využitím konvertoru převádějícího hodnotu datového typu *bool* na řetězce.

```
<Run Text="{Binding SelectedItem.IsActual, Converter={StaticResource
    BoolToCzechStringConverter}}"></Run>
```

Zdrojový kód 6.4: Ukázka DataBindingu s parametrem

Na této ukázce je do parametru *Text* pomocí klíčového slova *Binding* vložena vlastnost *IsActual* z vybrané položky z kolekce. Tato vlastnost je již zmíněného datového typu *bool*. Proto je dále uveden parametr *Converter*, který určuje, že vloženou vlastnost *IsActual* požadujeme zpracovat v nějakém konvertoru. V parametru je poté určen konkrétní konvertor. V tomto případě *BoolToCzechStringConverter*. Do něj je konvertovaná vlastnost předána jako parametr. Po vykonání konverze jsou do parametru *Text* uložena data, která jsou určena v návratové hodnotě metody *Convert* ve zvoleném konvertoru.

6.1.4 Services

Dalším projektem v řešení je *AgroSoft.Services*, který by bylo možné popsat jako databázovou vrstvu. V tomto projektu je řešena veškerá práce s databází.

Projekt obsahuje dva adresáře *Migrations* a *Services*. Účel těchto dvou adresářů bude rozebrán níže. Dále obsahuje definice tříd pro nastavení a připojení k databázi. Důležitou třídou je *AgroSoftDbContext*, která má jako bázovou třídu *DbContext*, která je definována v *Entity Frameworku*. V konstruktoru je definován tzv. *Connection string*, kterým je možné nastavit potřebné parametry pro připojení k databázi. V této práci je pro databázi zvolen soubor s koncovkou *.mdf*. Proto je nutné před překladem aplikace nastavit správnou cestu k databázovému souboru.

Zmíněná třída obsahuje metodu *OnModelCreating*, kde dochází k mapování tříd z vrstvy *Model* do konkrétních tabulek v databázi.

Další třídou v projektu je *AgroSoftDbMigrationsConfiguration*, ve které jsou nastaveny dva parametry pro povolení databázových migrací a také, že při migraci může docházet ke ztrátě dat. Toto nastavení je vhodné při vývoji aplikace, kdy při změně modelu jsou změny promítnuty i do databáze. Při migraci je vytvořena nová třída v adresáři *Migrations*, ve které jsou definovány změny oproti stávajícímu schématu databáze. Díky tomu je možné při implementaci vracet jednotlivé změny zpět.

V další třídě *DbContextSingleton* je definován, jak již název třídy napovídá, návrhový vzor *Singleton*. Tento způsob řešení zajišťuje existenci pouze jedné instance databázového kontextu v celé aplikaci.

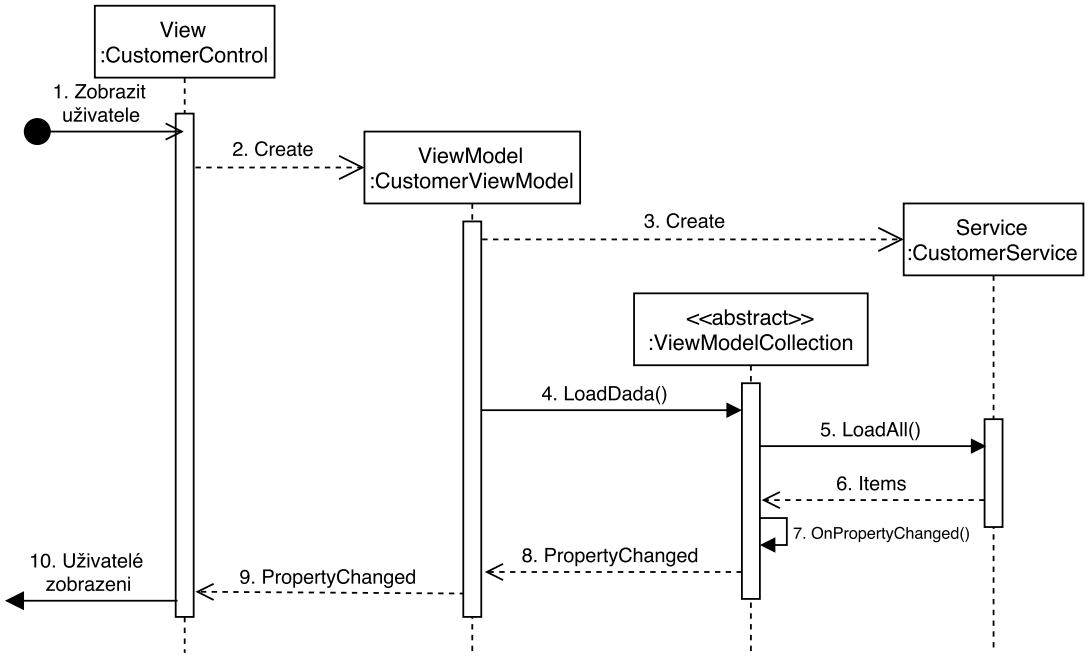
Zbývající adresář *Services* obsahuje definici abstraktní třídy *Repository* (umístěné v podadresáři Repository). V této abstraktní třídě je opět využito generického parametru, kterým se předá konkrétní třída z vrstvy *Model*. V konstruktoru třídy je nastavena vnitřní vlastnost udržující databázový kontext. Implementovaných metod je více, než je skutečně v aplikaci využito, ale metody jsou zde ponechány pro možné použití v budoucnu.

Adresář dále obsahuje definice tříd, které využívají zmíněnou třídu *Repository* jako svou bázovou. Tyto třídy jsou vytvořeny pro každou třídu z vrstvy *Model* a tím zajišťují uložení dat z vrstvy Model do databáze.

6.1.5 Komunikace mezi vrstvami

Nejčastější akcí uživatele je typicky zobrazení uložených dat v databázi. Pro vyobrazení této konkrétní situace byl vybrán případ zobrazení všech uložených zákazníků v systému. Sekvenční diagram, který odpovídá tomuto případu je zobrazen na obrázku 6.1.

Na této ukázce je požadavek od klienta na zobrazení zákazníků. Tento požadavek je proveden při kliknutí myši na položku v menu aplikace. Poté je zobrazena komponenta *CustomerControl* z logické vrstvy *View*, avšak na pozadí aplikace dochází k následujícím akcím.



Obrázek 6.1: Sekvenční diagram pro zobrazení zákazníků.

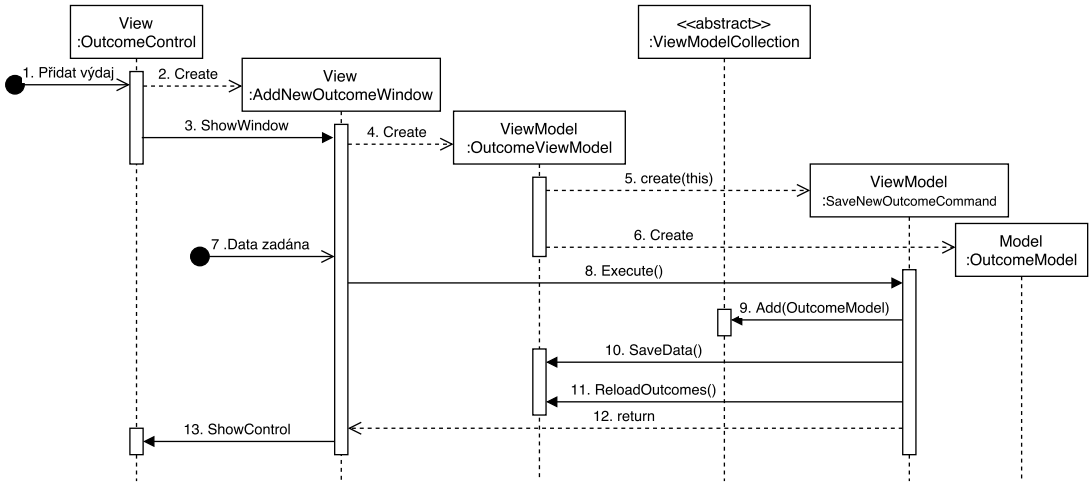
V dalším kroku po zobrazení komponenty je vytvořena instance třídy *CustomerViewModel* z vrstvy *ViewModel*. Vlastnosti této instance jsou pomocí *DataBindingu* zobrazeny v komponentě, ovšem při vytváření instance je potřeba načíst data z databáze. Proto je vytvořena instance třídy *CustomerService* z vrstvy *Service*. Tato instance slouží k vytváření a správě instancí třídy *CustomerModel*.

Následně je z instance *CustomerViewModel* volána metoda *LoadData* bázové třídy *ViewModelCollection*. Z této třídy je poté zaslán požadavek *LoadAll* instanci *CutomerService*. Při tomto požadavku dochází k použití *Entity Frameworku*, kdy dojde k načtení všech dat z příslušné tabulky databáze a vytvoření kolekce instancí třídy *CustomerModel* odpovídající jednotlivým záznamům v tabulce.

Tato kolekce načtených dat je poté načtena do vlastnosti *Items* ve *ViewModelCollection*. Tímto krokem jsou již data k dispozici, avšak pro jejich zobrazení v komponentě je nutné tuto komponentu informovat, že došlo ke změně vlastnosti. Tato notifikace o změně je provedena voláním metody *OnPropertyChanged*. Tímto dojde k přenesení informace o změně a následnému zobrazení načtených dat v komponentě *CustomerControl*.

Další velmi častou akcí v systému je přidání některé nové položky. Proto byla vybrána situace přidání nového výdaje. Tento případ je zanesen opět na sekvenčním diagramu na obrázku 6.2. Z důvodu lepší přehlednosti diagram obsahuje pouze hlavní komunikaci potřebnou pro vytvoření a uložení nového výdaje. Detaily komunikace budou nyní popsány.

Na začátku diagramu vycházíme z předpokladu, že v hlavním oknu aplikace je zobrazena komponenta *OutcomeControl*. První zaslaná zpráva v diagramu představuje kliknutí myši na tlačítko, umístěné v této komponentě, pro přidání výdaje. Po tomto stisku je vytvořena instance nového okna *AddNewOutcomeWindow*, které obsahuje formulář pro zadání potřebných dat k novému výdaji.



Obrázek 6.2: Sekvenční diagram pro přidání výdaje.

Dále je vytvořena instance třídy *OutcomeViewModel* pro správné nastavení *DataBindingu*. V tomto ViewModelu jsou dále také vytvořeny instance tříd *SaveNewOutcomeCommand* a *OutcomeModel*. Třída *SaveNewOutcomeCommand* slouží k uložení nového výdaje do kolekce výdajů a následně také do databáze. Při vytváření je této třídě předána instance tohoto ViewModelu.

Instance *OutcomeModel* ve ViewModelu představuje nově přidávaný výdaj. K vlastnostem této instance jsou *DataBindingem* propojeny jednotlivá formulářová pole v okně. Při *DataBindingu* dochází k automatickému nastavení hodnot těchto vlastností na základě uživatelem zadaných dat. V praxi to znamená, že není nutné řešit čtení jednotlivých dat z formulářů.

Po zadání veškerých potřebných dat iniciuje uživatel samotné uložení kliknutím na tlačítko umístěné v okně. Při tomto stisku je volána metoda *Execute* v instanci třídy *SaveNewCommand*. Následně je objekt *OutcomeModel*, jehož vlastnosti obsahují data zadaná uživatelem, přidán do kolekce všech výdajů. Poté je provedeno samotné uložení dat z modelu do databáze. Proto je volána metoda *SaveData* v *OutcomeViewModel*. V metodě dochází k následnému vyvolání metody *SaveAll* ve vrstvě *Services*, ale pro přehlednost diagramu není tato komunikace zobrazena.

Dalším krokem je vyvolání metody *ReloadOutcomes*, která přepočítá celkovou hodnotu výdajů v daném roce. Bez tohoto přepočtu by se neaktualizovala tato hodnota v okně aplikace. Po návratu z metody je okno pro zadávání nového výdaje uzavřeno a uživateli je opět zobrazena komponenta *OutcomeControl*.

6.2 Získávání informací o škůdcích

V následující podkapitole bude popsán způsob řešení získávání informací o výskytu škůdců ze serveru Ministerstva zemědělství. Základní způsob této komunikace byl popsán v kapitole 3. Tento systém ministerstvo zpřístupňuje skrze webovou aplikaci. Jelikož k tomuto serveru neexistuje žádné dostupné API (Aplication Programming Interface), byla zachytávána veškerá komunikace mezi serverem a webovým klientem. Následně byla tato komunikace analyzována a implementována v aplikaci.

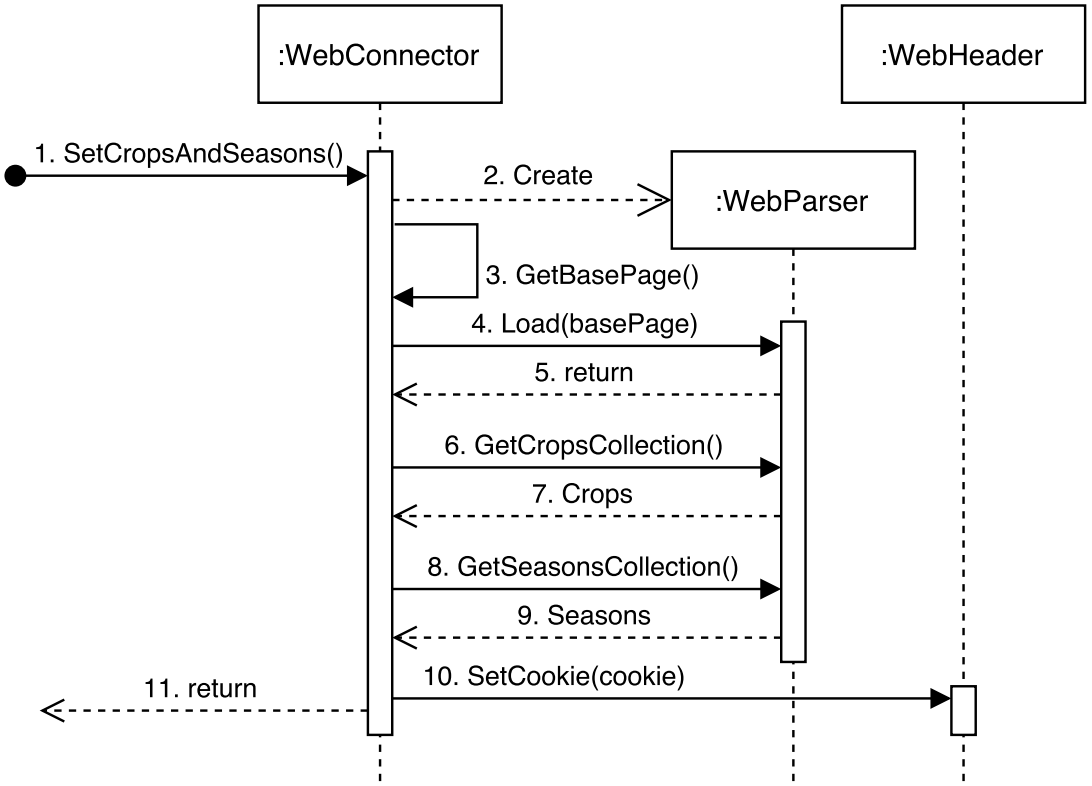
Řešení této komunikace je implementováno v projektu *AgroSoft.MapServer*, který obsahuje dva adresáře *Implementation*, obsahující definice tříd, a *Interface* obsahující definice rozhraní.

6.2.1 1.krok - Získání plodin a sezónních období

Prvním krokem k získání informací o škůdcích je potřeba získat úvodní stránku z webového serveru a následně ji zpracovat. K tomuto kroku je v adresáři *Implementation* vytvořena třída *WebConnector*, která slouží k veškeré komunikaci se serverem. K tomu využívá instanci třídy *WebClient* z frameworku .NET, která usnadňuje komunikaci s webovými servery. Dále také obsahuje instanci třídy *WebHeader*, která má bázovou třídu *WebHeaderCollection* z frameworku .NET a slouží k definici hlaviček HTTP dotazů. Tato třída se využívá až v následujících krocích.

Třída dále obsahuje proměnné pro uložení seznamu jednotlivých plodin a sezón. Pro ulehčení případné úpravy implementace jsou tyto seznamy definovány jako seznam instancí tříd implementující definované rozhraní. Ve výsledku to znamená, že pro plodiny byla definována třída *Crop* implementující rozhraní *ICrop* a pro období třída *Season* implementující rozhraní *ISeason*.

Samotné vytváření instancí plodin a sezón a nastavení jejich dat je prováděno při vyvolání metody *SetCropsAndSeasons*. K tomu, aby mohly být instance vytvořeny, je nejprve nutné stáhnout úvodní HTML stránku a zpracovat data obsažená na této stránce. Ke zpracování dat je vytvořena třída *WebParser*, která umožňuje pomocí metody *LoadData* načíst staženou webovou stránku a poté jsou vyvoláním metody *GetCropsCollection* a *GetSeasonCollection* vráceny seznamy plodin a sezón.



Obrázek 6.3: Sekvenční diagram načtení plodin a sezón.

Na obrázku 6.3 je znázorněn sekvenční graf tohoto prvního kroku. Po vyvolání metody *SetCropsAndSeasons* je vytvořena instance třídy *WebPaser*. Poté je zavolána privátní metoda *GetBasePage*, která využívá instanci třídy *WebClient*, pomocí její metody *UploadValues* získá HTML stránku ze serveru. Stránka je touto metodou načtena jako pole bajtů. Tato data jsou poté metodou *Load* předána instanci *WebParser*, kde je toto pole bajtů převedeno na datový typ *string* a zároveň zakódováno do formátu UTF-8 kvůli správnému zobrazení české diakritiky.

Po tomto načtení je volána metoda *GetCropsCollection*, ve které je pomocí regulárního výrazu získán obsah HTML elementu *select*. Tento element obsahuje další elementy *option*, které mají atribut *value* s číselnou hodnotou a také samotný název plodiny. Ukázka tohoto elementu je zobrazena na zdrojovém kódu 6.5. Aby bylo možné takto získané elementy dobře zpracovat jsou následně vloženy jako obsah dalšího elementu, tak aby vznikl validní formát XML. Takto vytvořená data jsou načtena do vytvořené instance třídy *XMLDocument*, jelikož tato třída poskytuje vhodné rozhraní k jejich zpracování. Poté jsou cyklicky získávány jednotlivé hodnoty elementu *option* (jméno plodiny) a jejich parametru *value* a na jejich základě vytvářeny instance třídy *Crop* s danými hodnotami a přidávány do vytvořeného seznamu. Třída *Crop* obsahuje vlastnosti pro definování jména a hodnoty atributu value. Nakonec je takto vytvořený seznam vrácen návratovou hodnotou.

```
<option value="5">Cibule</option>
```

Zdrojový kód 6.5: Ukázka elementu option.

Následně je volána metoda *GetSeasonsCollection*, která pracuje obdobně jako předchozí metoda. Z webové stránky jsou získána data o jednotlivých sezónních obdobích. Výsledkem je vytvoření seznamu instancí třídy *Season*, které obsahují vlastnosti pro identifikační číslo sezóny a určení jejího začátku a konce.

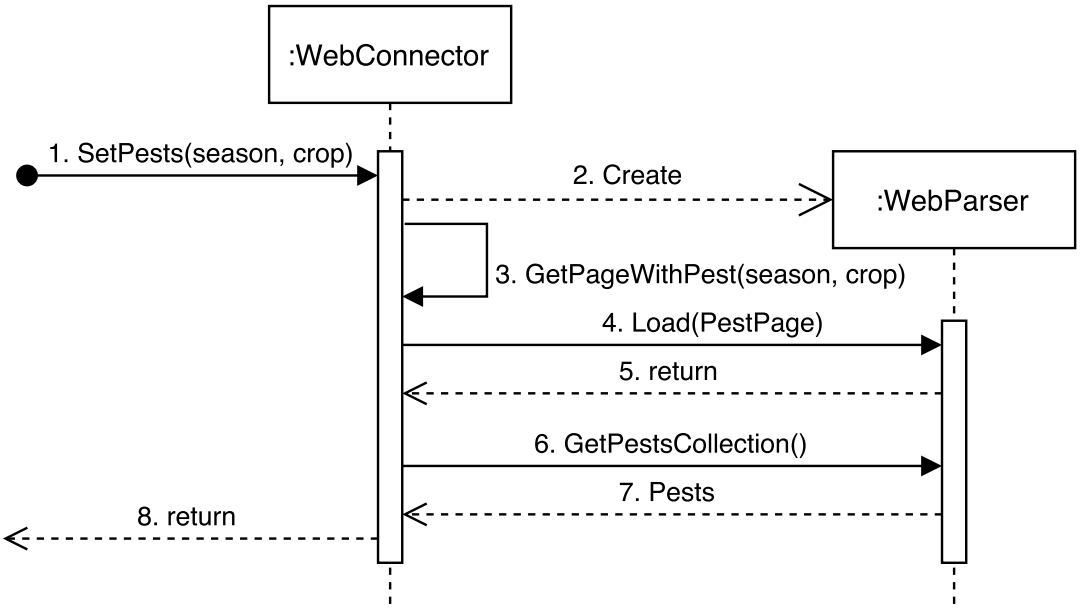
Při získání HTML stránky je získána i hlavička odpovědi obsahující parametr *SetCookie*. Hodnotu toho parametru je nutné v dalších krocích vložit do hlaviček požadavku pro správnou identifikaci na straně serveru. Proto je do instance třídy *WebHeader* tato hodnota vložena pomocí volání metody *SetCookie*.

6.2.2 2.krok - Získání škůdců

V prvním kroku je tedy z úvodní stránky získán seznam plodin a seznam sezón. V následujícím kroku je potřeba vybrat konkrétní plodinu ze seznamu a také období. Tento výběr je odeslán v požadavku na server a je vrácena HTML stránka, která obsahuje výčet dostupných škůdců této plodiny. Dále také nabízí vybrat možnost zobrazení výsledků, jen pokud je výskyt škůdce pozitivní.

Pro tento krok obsahuje *WebConnector* metodu *SetPests*, která vytvoří seznam obsahující instance tříd implementující rozhraní *IPest*. Tato třída obsahuje vlastnosti pro nastavení identifikačního čísla škůdce a jeho jména. Zároveň tato třída implementuje rozhraní *IPest* pro ulehčení možné budoucí změny implementace.

Komunikace mezi klientem a serverem je zachycena na sekvenčním diagramu na obrázku 6.4. Na tomto diagramu je vidět podobnost s předchozím diagramem, ovšem metoda *SetPests* vyžaduje dva parametry. Prvním parametrem je konkrétní instance třídy implementující rozhraní *ISeason* a druhým je instance třídy implementující rozhraní *ICrop*. Těmito dvěma parametry je definována konkrétní sezóna a plodina. Dále je opět vytvořena instance třídy *WebParser*, která bude použita pro následné zpracování HTML stránky.



Obrázek 6.4: Sekvenční diagram pro načtení seznamu škůdců.

Následně je vyvolána privátní metoda *GetPageWithPest*, které jsou parametrem předány instance tříd určující zvolenou sezónu a plodinu. V této metodě je instanci třídy *WebConnector* přiřazena instance třídy *WebHeader*, která obsahuje nastavené hodnoty jednotlivých parametrů hlavičky požadavku, včetně nastaveného parametru *Cookie* z předchozího kroku. Aby mohl být proveden požadavek na server, je nutné k požadavku přidat i hodnoty určující vybranou plodinu a sezónu. Jelikož jsou tyto hodnoty na webové stránce nastavovány do formulářů, je nutné tuto skutečnost přenést do implementace. Proto při vyvolání metody *UploadValues* je pomocí parametru nastavena nejen příslušná URL adresa serveru, ale druhým parametrem jsou nastavena data s hodnotami. Tento druhý parametr je instance třídy *NameValueCollection* z frameworku .NET. Aby byla v implementaci všechna formulářová data pohromadě, byla k tomu účelu vytvořena statická třída *WebFormData*, kde jednotlivé statické metody vracejí na základě parametrů instanci třídy *NameValueCollection*. Na ukázce zdrojového kódu 6.6 je zobrazena metoda pro získání formulářových dat pro tento krok.

```
public static NameValueCollection GetFormWithCropAndSeason(ICrop crop, ISeason season)
{
   return new NameValueCollection()
   {
      { "f1", "f1" },
      { "f1:Plodina", crop.Id },
      { "f1:Sezona", season.Id },
      { "f1:Datum_od", season.From},
      { "f1:Datum_do", season.To },
      { "f1:j_id_jsp_823783899_22", "Pokracovat" },
      { "javax.faces.ViewState", "j_id1:j_id2" }
   };
}
```

Zdrojový kód 6.6: Ukázka formulářových dat.

Po zaslání požadavku je obdržena HTML stránka a obdobným způsobem, jako v předchozím kroku, jsou data zpracována v instanci třídy *WebParser*. Po provedení zpracování jsou výsledná data uložena v seznamu obsahující jednotlivé instance třídy *Pest*.

6.2.3 3.krok - Získání mapy

Posledním krokem je samotné získání mapy z mapového serveru. Protože je pro provedení tohoto kroku potřeba udělat několik akcí, bude následující text rozdělen na jednotlivé části. Mapa je získána zavoláním metody *GetImageWithPests* ve třídě *WebConnector*, která vrací vytvořenou instanci třídy *Image* obsahující obraz získané mapy. Tato metoda očekává dva parametry, prvním je instance třídy implementující rozhraní *IPest*, která určuje konkrétního škůdce, a druhým je instance třídy implementující rozhraní *IBbox*, které určuje pozici mapy. Účel použití tohoto rozhraní bude popsán níže.

Získání HTML stránky

Prvním krokem je zaslání dotazu pro získání poslední HTML stránky. Dotaz je proveden obdobně jako v předchozích případech (pomocí metody *UploadValues*) a obsahuje i data určující hodnoty formulářů. V tomto případě se jedná zejména o určení konkrétního škůdce. Tato stránka však není následně nijak zpracována, neboť neobsahuje žádné potřebné informace pro využití v aplikaci. Tento požadavek je proveden pouze z důvodu zajištění aktualizace *session* na straně serveru.

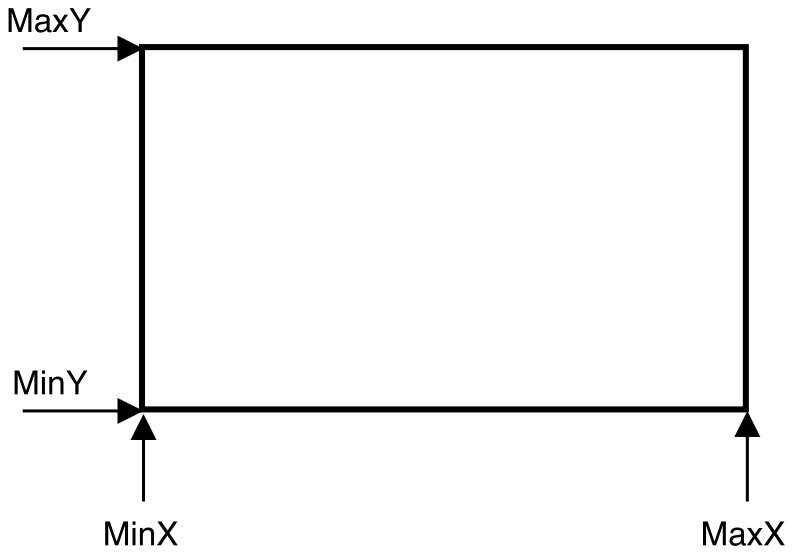
Získání parametru z JSON

Webová aplikace Ministerstva zemědělství následně získá z dalšího serveru data ve formátu JSON, které obsahují parametry pro nastavení jednotlivých elementů na stránce. Mimo jiné je v tomto JSON formátu obsažen také parametr *filter*. Jeho hodnota obsahuje řetězec obsahující identifikační číslo parametru *session* a také, pro účely aplikace nejdůležitější, identifikační číslo určující obrazovou vrstvu na které jsou zobrazeny výskyty vybraného škůdce. Zpracování formátu JSON a získání hodnoty požadovaného parametru je opět provedeno ve vytvořené instanci *WebParser*.

Získání podkladového obrysu mapy

Samotné zobrazení obrazu mapy s jednotlivými výskyty škůdce se sestává ze dvou kroků. V prvním kroku je nejprve získán obrys mapy České republiky s vyznačením jednotlivých krajů a okresů. Pro získání tohoto obrazu se již zasílá požadavek na URL adresu WMS serveru. Ke specifikaci požadavku se na server zasílají data stejným způsobem, jako v předchozích bodech hodnoty formulářů. Zde je již využita instance třídy implementující rozhraní *IBbox*, která hodnotami svých vlastností definuje pozici požadovaného obrazu. Rozhraní *IBbox* definuje čtyři vlastnosti *MinX, MaxX, MinY a MaxY*. Těmito parametry se určují hranice obrazu, jak je možné vidět na obrázku 6.5. Tyto parametry je možné měnit, a tím docílit efektu posunutí obrazu, nebo přiblížení či oddálení.

Na ukázce zdrojového kódu 6.7 je zobrazena metoda ze třídy *WebFormData*, která vytváří instanci třídy *NameValueCollection* obsahující parametry a jejich hodnoty. Tato data jsou poté zaslána v požadavku na WMS server.



Obrázek 6.5: Použití parametrů rozhraní IBbox.

```
public static NameValueCollection GetFormForBackgroundImage(IBbox bbox)
{
   return new NameValueCollection(){
      {"service", "wms"},
      {"version", "1.1.1"},
      {"request", "GetMap"},
      {"bbox", bbox.ToString()},
      {"srs", "EPSG:102067"},
      {"format", "image/png"},
      {"height", "240"},
      {"width", "388"},
      {"layers", "SC_KRAJE_GEN,CR_OKRESY,CR_HRANICE_SRS,CR_HRANICE2_SRS"},
      {"styles", ""}
      };
}
```

Zdrojový kód 6.7: Ukázka zasílaných dat

Jednotlivé parametry a jejich hodnoty jsou v ukázce definovány při vytváření nové instance. Každý parametr je definován ve složených závorkách, ve kterých je uvede název parametru a hodnota oddělená čárkou. Takto definované parametry jsou také odděleny čárkou. Prvním parametrem na ukázce je parametr *service* s hodnotou *wms*. Tento parametr určuje, na kterou běžící službu na serveru je požadavek určen. Dalším parametrem je *version*, kterým je určena konkrétní verze služby. Tento parametr je uveden z důvodu, že na některých serverech může běžet stejná služba v několika verzích a požadavek by nemusel být proveden korektně.

Dále následuje parametr *request*, kterým je určen typ požadavku. V tomto případě tedy *GetMap*. Poté je uveden parametr *bbox*, kterým je definována pozice obrazu na mapě. Jeho hodnota je definována instancí předanou v parametru metody. Jelikož jsou všechny parametry a jejich hodnoty definovány jako řetězce, umožňuje instance pomocí metody *ToString* vrácení hodnot ve formě řetězce v příslušném formátu.

Další parametr v pořadí je *srs*, což je zkratka z anglického názvu *Spatial Reference System* a určuje v jakém referenčním prostorovém systému jsou data zadávána. Hodnota

tohoto parametru je *EPSG:102067*, což určuje, že data jsou zadána v Křovákově souřadnicovém systému. Následuje parametr *format* pro určení obrazového formátu mapy. Další dva parametry určují výšku a šířku výsledného obrazu v pixelech.

Posledními dvěma parametry jsou *layers* a *styles*. Hodnota parametru *styles* není určena, ale parametr musí být uveden, jelikož jej server požaduje. Parametr *layers* udává svou hodnotou, jaké obrazové vrstvy je možné na výsledné mapě zobrazit. Vrstvou se v tomto případě rozumí například vyznačení hranic okresů, zobrazení vodních toků, zobrazení silnic a podobně. Na základě určené pozice obrazu na mapě WMS server sám zvolí, které vrstvy budou zobrazeny a které nikoliv.

Tímto je tedy získán obraz obsahující pouze určenou část mapy a nejsou na něm zobrazeny pozice výskytu vybraného škůdce.

Získání pozic výskytů škůdce

V dalším kroku je nutné získat obraz s vyznačením pozic výskytů daného škůdce. Tento obraz je získán z příslušné URL adresy WMS serveru. Zasílání požadavku probíhá stejným způsobem jako požadavek pro získání obrazu s obrysem mapy v předchozím bodu. Požadavek musí obsahovat parametry a hodnoty, které jsou shodné s daty v předchozím bodu. To proto, aby byly získány dva obrazy, které jsou pro stejnou část mapy a zároveň mají stejné rozlišení. Musí být ovšem změněna hodnota parametru *layers* a musí být také přidán parametr *F_SRS_MAPA_MP*, jehož hodnota musí být nastavena na hodnotu parametru *filter* získaného ze souboru JSON uvedeného výše. Díky tomuto parametru služba WMS pozná, pro kterého škůdce a v jakém období má vrátit obraz s vyznačením výskytů.

Oba obrazy, jak obrys mapy, tak pozice škůdců, jsou ze serveru získány jako pole bajtů. Z těchto polí jsou poté vytvořeny instance třídy *Image*, které reprezentují tyto obrazy.

Obrázek 6.6: Ukázka spojení obrazů.

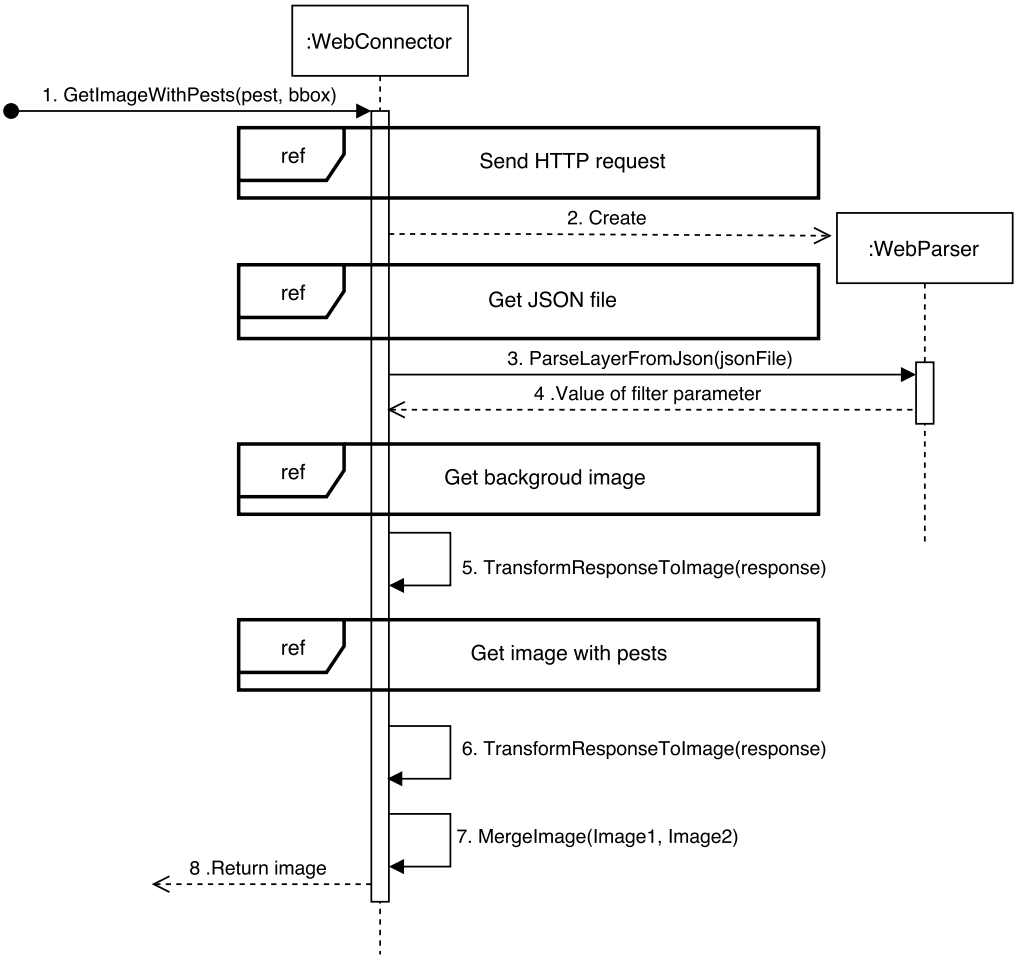
Spojení obrazů

Posledním krokem je spojení obou získaných obrazů. Výsledkem tohoto spojení je jeden obraz obsahující obrys mapy společně s vyznačením výskytů. Ukázka výsledného spojení je zobrazena na obrázku 6.6.

Na tomto obrázku je vidět jednak obrys části mapy (levý horní obraz), jednak obraz určující místa výskytů (pravý horní obraz). Cílem tohoto spojení je tedy vytvořit obraz, který je na ukázce zobrazen dole.

Ke sloučení těchto dvou obrazů slouží metoda *MergeImage*, které jsou parametrem předány obě instance třídy *Image*. Jelikož z těchto instancí není možné přímo vytvořit výsledný obraz, musí být nejprve na základě instancí třídy *Image* vytvořeny instance třídy *Bitmap*. Následně tyto instance využívá třída *Graphics*, která umožňuje vytvoření požadovaného obrazu.

Sekvenční diagram



Obrázek 6.7: Sekvenční diagram pro získání výsledného obrazu.

Na následujícím sekvenčním diagramu 6.7 je zobrazen výše popsaný způsob komunikace při získávání výsledného obrazu. Jako první je zaslán požadavek na server, kvůli identifikaci vybraného škůdce. Následně vytvořena instance třídy *WebParser*. Poté je získán ze

serveru soubor ve formátu JSON. Ten je zpracován ve třídě *WebParser* pro získání hodnoty parametru *filter*.

Poté je ze serveru získán obraz ve formě pole bajtů obsahující část mapy určenou instancí třídy *Bbox*. Tato data jsou předána privátní metodě *TransformResponseToImage*, která na jejich základě vytvoří instanci třídy *Image*. Dále je získán obraz určující pozice výskytů škůdce a stejným způsobem je předán výše zmíněné metodě pro převod na instanci třídy *Image*. Nakonec jsou tyto dva obrazy spojeny do jednoho a výsledná instance třídy *Image* je vrácena návratovou hodnotou.

6.3 Grafické rozhraní

V následující podkapitole bude popsáno, jakým způsobem je vytvořeno uživatelské rozhraní celé aplikace. Při tvorbě rozvržení jednotlivých ovládacích prvků v okně aplikace byl kladen důraz jednoduchost a intuitivnost ovládání. Jelikož je aplikace vytvořena jako desktopová a drtivá většina takových aplikací má horizontální navigační menu umístěné v horní části okna, je také tímto způsobem řešena navigace v aplikaci.

Celá aplikace je rozdělena na tři části. Horizontální menu, zobrazovací část a stavový řádek. Toto rozdělení aplikace bylo zvoleno z důvodu, že spousta aplikací, které běžný uživatel používá, je právě takto rozdělena. Díky tomu je poté pro uživatele orientace v aplikacích snadnější.

Horizontální menu obsahuje následující jednotlivé položky.

- **Období** - Tato položka obsahuje položku *Spravovat období*, při jejímž výběru jsou uživateli zobrazeny dostupné roky uložené v systému a zároveň je může mazat, vytvářet nové a nastavovat vybraný rok na aktuální. Podle tohoto určení se poté v ostatních sekcích zobrazují data vztahující se k tomuto vybranému roku.

- **Příjmy** - Tato položka obsahuje položku *Spravovat příjmy*, která zobrazí uživateli všechny uložené příjmy v roce, který je nastaven jako aktuální. Zde má uživatel také možnost přidávat, upravovat a mazat příjmy.

- **Výdaje** - V této položce je možné vybrat ze dvou dalších položek. A to *Správa výdajů*, která zobrazí uložené údaje v aktuálním roce a umožní jejich administrativu, a *Správa účelů výdajů*, která zobrazí uložené názvy účelů výdajů. I zde je samozřejmě umožněna editace těchto dat.

- **Zákazníci** - Tato část obsahuje položku *Spravovat zákazníky*, která, stejně jako výše zmíněné položky, zobrazí uživateli uložené informace o zákaznících a umožní jejich editaci.

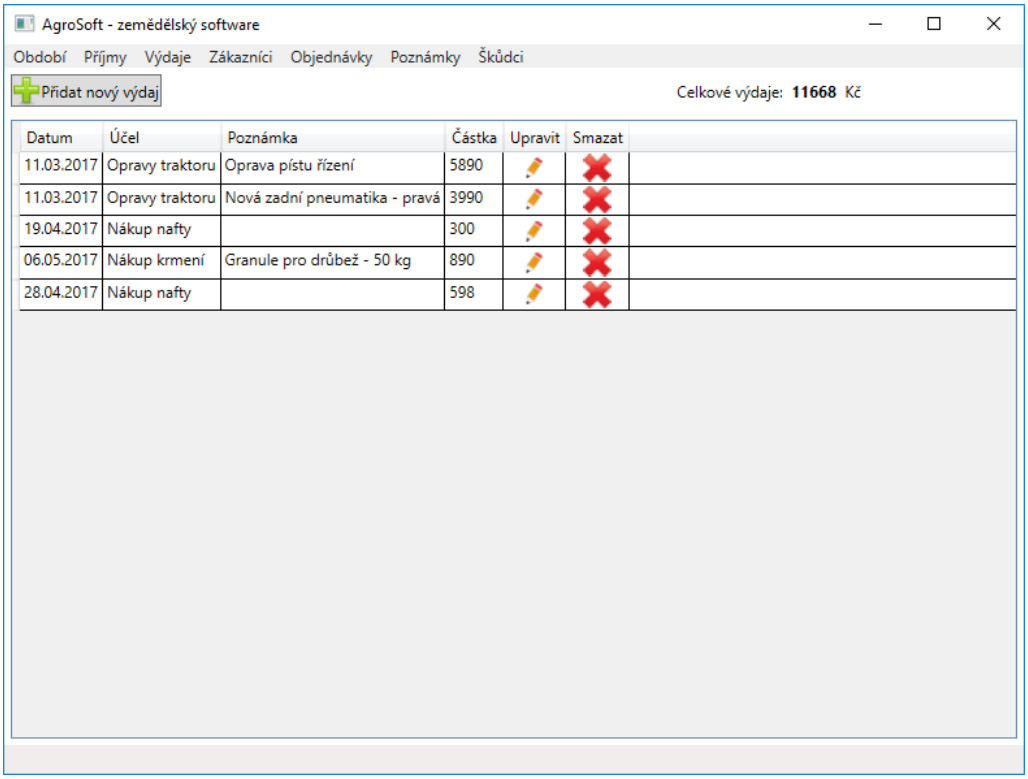
- **Objednávky** - Tato kategorie zahrnuje další dvě položky. První položkou je *Správa objednávek*, umožňující administraci objednávek vztahující se ke zvolenému roku. Druhou položkou je *Správa komodit*, která zobrazí uložené názvy komodit a samozřejmě také nabídne možnost jejich úprav.

- **Poznámky** - V této části je také obsažena položka *Spravovat poznámky*, ovšem při jejím výběru jsou zobrazeny na začátku pouze kategorie poznámek a až při výběru konkrétní kategorie jsou následně zobrazeny na téže obrazovce poznámky, které do

této kategorie patří. V této části je možné spravovat jednak kategorie poznámek a jednak i samotné poznámky.

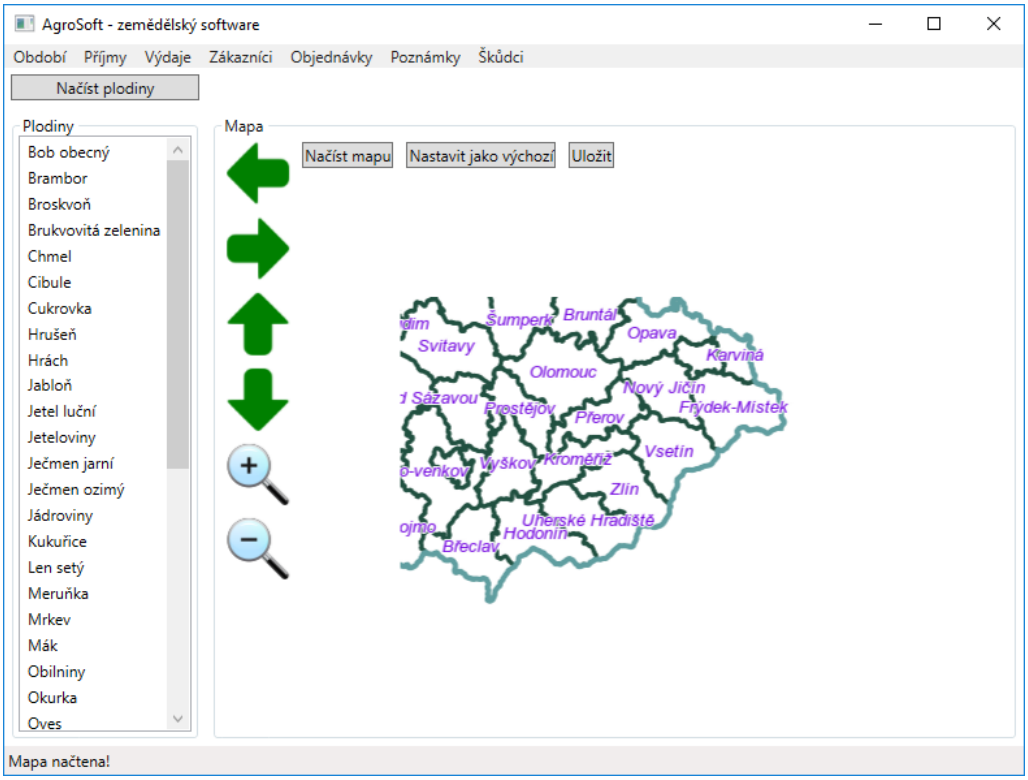
- **Škůdci** - V této poslední položce je možné vybrat ze dvou položek. První položka *Sledované plodiny* umožní zobrazení názvů sledovaných plodin v daném roce a možnost získání obrazových informací o výskytech jednotlivých škůdců této vybrané plodiny. Druhou položkou je *Nové sledování*, při jejímž výběru je možné vybrat část mapy a určit, kterého škůdce v dané oblasti zahrnout do sledování.

Zobrazovací část se mění na základě vybraného odkazu v horizontálním menu. Na obrázku 6.8 je zobrazen vzhled obrazovky pro výpis všech výdajů. Jak je vidět, výdaje jsou zobrazeny v tabulce. Uživatel má možnost seřazení položek na základě vybraného sloupce. Poslední dva sloupce slouží k možnosti úpravy a smazání konkrétního řádku(záznamu). V levém horním rohu je umístěno tlačítko pro přidání nového výdaje. Při kliknutí na tyto ovládací prvky je uživateli zobrazeno modální okno ve kterém uživatel může vkládat data k novému záznamu, případně upravovat, nebo potvrdit smazání. Obdobným způsobem jsou řešeny i obrazovky pro správu příjmů, zákazníků, účelů výdajů, objednávek a komodit.



Obrázek 6.8: Obrazovka pro výpis výdajů.

Mírně odlišným způsobem jsou zobrazeny obrazovky pro správu roků, poznámek a výskytu škůdců. U těchto sekcí nejsou data zobrazovaná v tabulce jako ve výše zmíněných případech. U správy roků a poznámek je zobrazovací část okna rozdělena na dvě části. V levé části jsou zobrazeny buď dostupné roky, nebo kategorie poznámek ve formě seznamu. Při výběru konkrétní položky v tomto seznamu se ve druhé části zobrazí další informace k této vybrané položce.



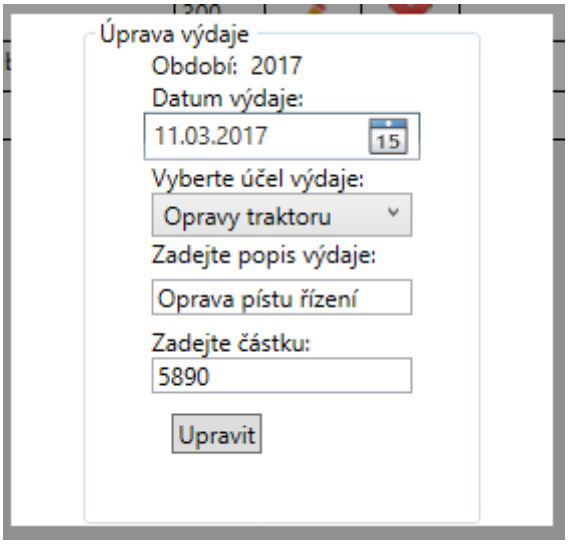
Obrázek 6.9: Obrazovka pro nastavení nového sledování.

Obdobným způsobem, tedy rozdělením na dvě části je řešena i sekce pro sledování výskytu škůdců. Grafické rozhraní pro nastavení nového sledování je vidět na obrázku 6.9. V levé části jsou po kliku na tlačítko *Načíst plodiny* zobrazeny dostupné plodiny, které jsou staženy z webové aplikace Ministerstva zemědělství. Následně v pravé části je možné zobrazit mapu ČR. K prvotnímu stažení mapy dojde po kliku na tlačítko *Načíst mapu*. Pozici mapy, která bude při tomto prvotním načtení zobrazena může uživatel měnit pomocí tlačítka *Nastavit jako výchozí*. Tato funkce usnadňuje vytváření nových sledování škůdců, neboť se předpokládá, že oblast, kterou chce uživatel sledovat je pro všechny jeho plodiny víceméně stejná. Tedy, že pole na kterých plodiny pěstuje jsou poblíž sebe.

Dále má uživatel možnost mapu posunovat pomocí zelených šipek umístěných vedle zobrazené mapy. Je také umožněno mapu přibližovat a oddalovat. Těmito ovládacími prvky může uživatel měnit oblast pro sledování. Při každé z těchto akcí probíhá stažení a zobrazení nového obrazu mapy ze serveru.

Po nastavení oblasti vybere uživatel konkrétní plodinu v levé části a stiskne tlačítko *Uložit*. Všechna tlačítka na této obrazovce komunikují s vrstvou *ViewModel* pomocí příslušných *Commands*.

Jak již bylo zmíněno výše, při vytváření nových záznamů (mimo nastavení sekce sledování škůdců), jejich úpravě a smazání je zobrazeno modální okno. Příklad modálního okna je vyobrazen na obrázku 6.10, které je zobrazeno při úpravě vybraného výdaje. Při vytváření tohoto okna je hlavní okno aplikace ztmaveno. Tlačítka v těchto modálních oknech, které slouží k vytvoření, uložení úpravy, či potvrzení smazání, komunikují s vrstvou *ViewModel* také pomocí *Commands*. Na rozdíl od tlačítek, které pouze otevřou nebo uzavřou modální okno. Tyto tlačítka nijak s vrstvou *ViewModel* komunikovat nemusí.



Obrázek 6.10: Ukázka modálního okna.

Posledním prvkem v okně aplikace je stavový řádek umístěný ve spodní části okna. Tento řádek slouží pouze k informování uživatele o stavu provedené akce. Například zdárné vytvoření záznamu apod.

Kapitola 7

Testování

V následující kapitole bude popsán způsob a výsledky testování vytvořené aplikace. Testováno bylo splnění požadavků pomocí *akceptačních testů* a následně proběhlo i uživatelské testování, při kterém pracovali s aplikací sami uživatelé.

7.1 Akceptační testování

V následujících tabulkách jsou uvedeny případy, které byly v jednotlivých sekcích testovány. V těchto tabulkách jsou také uvedeny očekávané výsledky jednotlivých případů a zda byl výsledek testu úspěšný či nikoliv.

V následující tabulce 7.1 jsou uvedeny testy pro sekci zajišťující správu období v programu.

Testovaný scénář	**Očekávaný výsledek**	**Stav**
Vytvoření roku	Nový rok je vytvořen.	OK
Smazání roku	Rok je vymazán a zároveň všechny související příjmy, výdaje, objednávky a sledování škůdců.	OK
Změna aktuálního roku	Aktuální rok je změněn a současně jsou změněny i obsahy sekcí výdaje, příjmy, objednávky a sledování škůdců.	OK

Tabulka 7.1: Tabulka akceptačních testů správu období.

Další tabulky 7.2 a 7.3 obsahují testy operací s výdaji a účely výdajů.

Testovaný scénář	Očekávaný výsledek	Stav
Vytvoření výdaje	Nový výdaj je vytvořen a celkový součet výdajů je přepočítán.	OK
Upravení výdaje	Vybraný výdaj je upraven a celkový součet výdajů je přepočítán.	OK
Smazání výdaje	Vybraný výdaj je smazán a celkový součet výdajů je přepočítán.	OK

Tabulka 7.2: Tabulka akceptačních testů pro výdaje.

Testovaný scénář	Očekávaný výsledek	Stav
Vytvoření účelu výdaje	Nový účel je vytvořen.	OK
Úprava účelu	Název účelu je upraven.	OK
Smazání účelu	Účel je smazán i se všemi výdaji, které se k němu vztahují.	OK

Tabulka 7.3: Tabulka akceptačních testů pro účely výdajů.

V další tabulce 7.4 jsou uvedeny testy pro sekce obsahující objednávky a komodity.

Testovaný scénář	Očekávaný výsledek	Stav
Vytvoření objednávky	Nová objednávka vytvořena.	OK
Úprava objednávky	Objednávka je upravena.	OK
Smazání objednávky	Objednávka je smazána.	OK
Vytvoření komodity	Nová komodita je vytvořena.	OK
Úprava komodity	Název komodity je upraven.	OK
Smazání komodity	Vybraná komodita a všechny objednávky vybrané komodity jsou smazány.	OK

Tabulka 7.4: Tabulka akceptačních testů pro objednávky a komodity.

Předposlední tabulka 7.5 zobrazuje testy pro sekci příjmů a také sekci zákazníků.

Testovaný scénář	Očekávaný výsledek	Stav
Vytvoření příjmu	Nový příjem je vytvořen a celkový součet příjmů přepočítán.	OK
Úprava příjmu	Příjem je upraven a součet příjmů přepočítán.	OK
Smazání příjmu	Příjem je smazán a součet příjmů přepočítán	OK
Vytvoření zákazníka	Nový zákazník je vytvořen.	OK
Úprava údajů zákazníka	Údaje vybraného zákazníka jsou upraveny.	OK
Smazání zákazníka	Údaje vybraného zákazníka jsou smazány a současně i všechny jeho objednávky a přijaté příjmy.	OK

Tabulka 7.5: Tabulka akceptačních testů pro příjmy a zákazníky.

V poslední tabulce 7.6 jsou uvedeny akceptační testy pro sekci sledování výskytu škůdců.

Testovaný scénář	Očekávaný výsledek	Stav
Načtení mapy	Mapa je zobrazena.	OK
Uložení výchozí pozice	Současné zobrazení pozice je uloženo jako výchozí.	OK
Posun mapy nahoru	Zobrazení mapy je posunuto směrem nahoru.	OK
Posun mapy dolů	Zobrazení mapy je posunuto směrem dolů.	OK
Posun mapy doprava	Zobrazení mapy je posunuto směrem doprava.	OK
Posun mapy doleva	Zobrazení mapy je posunuto směrem doleva.	OK
Přiblížení mapy	Zobrazená část mapy je přiblížena.	OK
Oddálení mapy	Zobrazená část mapy je oddálena	OK
Vytvoření nového sledování	Vybraná plodina a oblast mapy je uložena.	OK
Smazání sledování	Vybrané sledování je smazáno.	OK
Zobrazení škůdců	Výskyty škůdců vybrané plodiny jsou zobrazeny.	OK

Tabulka 7.6: Tabulka akceptačních testů pro sledování škůdců.

7.2 Uživatelské testování

Po dokončení celé aplikace bylo provedeno uživatelské testování. Tohoto testování se zúčastnili tři uživatelé, kterým byla aplikace nainstalována a zároveň jim byl zaslán dotazník s několika otázkami. V této podkapitole budou vypsány jak jednotlivé otázky, tak konkrétní odpovědi o těchto uživatelů.

7.2.1 Charakteristiky uživatelů

Uživatel A : Pro tohoto uživatele byla celá aplikace primárně určena a výše popsané požadavky byly přizpůsobeny jeho potřebám. Jedná se 32letého zemědělce, který obhospodařuje přibližně 4,5 ha. Tato zemědělská činnost není jeho hlavním zdrojem příjmů. Každoročně pěstuje asi osm druhů zemědělských komodit a některé z nich i prodává. Veškeré záznamy si až dosud vede papírovou formou a částečně také pomocí tabulkového procesoru.

Uživatel B : Druhým uživatelem je 43letý muž, který obdělává asi 6 ha. Na této rozloze ovšem zpravidla pěstuje pouze dvě komodity (pšenice a brambory). Nadbytek úrody prodává. V budoucnu zamýšlí rozšířit rozlohu a počet komodit. V současné době také pracuje v lese. Obě tyto činnosti také nejsou jeho hlavním zdrojem příjmů. Žádné záznamy si doposud nevedl.

Uživatel C : Posledním uživatelem, který byl zapojen do testování aplikace, je 53letý muž, jehož znalosti práce s počítačem jsou mírně nad základní znalostí. Obdělává cca 2,5 ha, mimo to nabízí zemědělské služby (orba, setí atd.) ostatním malým zemědělcům v okolí. Dále chová několik včelstev. Každoročně prodá přes 100 kg medu a také některé další včelí produkty. Příjem z těchto činností není jeho hlavní zdroj příjmů a záznamy si prakticky nevede.

7.2.2 Získané odpovědi

V této podkapitole budou popsány jednotlivé otázky a konkrétní odpovědi od uživatelů.

Otázka č. 1 - Je pro Vás menu v aplikaci srozumitelné?

Uživatel A : Ano, jednotlivé položky v menu aplikace jsou dle mého názoru přehledné a navigace v nich mi nečinila problém.

Uživatel B : Každý odkaz v menu je pojmenován podle kategorie, kterou obsahuje. Díky tomu nebyl z tohoto hlediska žádný problém s orientací v programu.

Uživatel C : Trvalo mi delší dobu, než jsem se v programu zorientoval. Po proklikání všech odkazů v menu jsem už věděl, kde co je.

Otázka č. 2 - Je pro Vás orientace v jednotlivých sekcích srozumitelná? (vytvoření výdaje, objednávky, jejich úpravy atd.)

Uživatel A : Ano, líbí se mi jednotný styl zobrazování informací v jednotlivých odkazech v menu (příjmy, výdaje atd.). Přestože je volba roku a správa po známek vytvořena jiným stylem, jsou data zobrazena srozumitelně.

Uživatel B : Ačkoliv nemám s podobnými aplikacemi velké zkušenosti, nebyl pro mě problém se v programu zorientovat a pracovat s ním.

Uživatel C : Orientace ani tak nebyla problém, jako to, že jsem zprvu nevěděl, co přesně aplikace po vytvoření nových údajů udělá. Poté, co jsem si to nejprve vyzkoušel, pochopil jsem, jak a kde se data zobrazí a jak je dále použít.

Otázka č. 3 - V položce menu (Nové sledování) vyberte na mapě ČR oblast Vašich polí a nastavte některou z Vámi pěstovaných plodin. Zhodnoťte složitost tohoto úkolu.

Uživatel A : Po zobrazení části programu, kde se nastavuje nové sledování, nebyla zobrazena mapa ČR. Z možností, které byly zobrazeny jsem pochopil, že je nejprve nutné zvolit načtení mapy a poté i plodin. Poté, co se objevila mapa, již nebyl problém úkol dokončit úspěšně. Složitost považuji za minimální.

Uživatel B : Po přechodu tuto kategorii jsem si všiml tlačítka pro načtení mapy. Jelikož jsem zvyklý pro manipulaci s mapou pomocí myši, byl pro mě nezvyk posouvat mapu pomocí šipek. Napoprvé se mi jevil úkol jako těžký, ale při opětovném použití bych ho označil za lehký.

Uživatel C : Zprvu jsem vůbec nevěděl jak to udělat, jelikož na obrazovce nebyla mapa ani plodiny. Poté jsem zkusil kliknout na tlačítka pojmenované jako načíst a objevily se jak názvy plodin, tak mapa. Když jsem pochopil způsob práce s mapou, dokončit úkol již nebyl problém.

Otázka č. 4 - Zobrazte si výskyty škůdců plodiny zvolené v předchozím kroku. Jsou pro Vás zobrazené informace přínosné?

Uživatel A : Ano, jelikož jsou zobrazeny informace o všech škůdcích dané ke každé plodině, beru tuto možnost jako výraznou úsporu času oproti manuálnímu hledání na webových stránkách.

Uživatel B : Ano, informace jsou přínosné. Pro informaci v jaké míře v okolí škůdce řádí to bohatě stačí.

Uživatel C : Ano, vůbec jsem netušil, že takovéto informace o výskytu škůdců jsou dostupné. Ačkoliv jsou tyto informace pro mě přínosné, nemyslím si, že bych při pravidelném používání programu tuto funkci využíval.

Otázka č. 5 - Postrádali jste v aplikaci nějakou funkci?

Uživatel A : Ne, aplikace splňuje vše co bych potřeboval.

Uživatel B : Ne, v aplikaci mi nic zásadního nechybělo.

Uživatel C : Chyběla mi možnost zvětšení obrázku, na kterém je zobrazena mapa při výběru nového sledování.

Otázka č. 6 - Používali byste takto vytvořenou aplikaci?

Uživatel A : Ano, takto vytvořená aplikace by se mi velmi hodila a myslím si, že by dokonale pokryla veškerou moji agendu spojenou se zemědělskou činností.

Uživatel B : Ano, i když na používání podobných aplikací nejsem zvyklý. Ani si prozatím nevedu žádnou evidenci, ale myslím si, že bych mohl v budoucnu podobnou aplikaci

potřebovat.

Uživatel C : Pro mě osobně by bylo používání aplikace spíše zatěžující, protože si tak rozsáhlou evidenci nevedu. Nicméně si myslím, že pro někoho může být aplikace velkým přínosem.

7.2.3 Zhodnocení výsledků

Z výše uvedených odpovědí lze vyvodit, že navrhovaná aplikace je uživateli vnímána převážně jako poměrně přehledná a srozumitelná. Žádný z uživatelů neměl problém se s aplikaci naučit pracovat. Součástí testování bylo také zadání úkolu, který byl zaměřen na ověření použitelnosti části aplikace pro sledování škůdců, konkrétně jak si uživatelé poradí s nastavením nového sledování. Z odpovědí k tomuto úkolu lze konstatovat, že pro uživatele nebyl problém toto nové sledování nastavit již při prvním používání aplikace. Příznivě vyznívá také celkové zhodnocení aplikace, kterou dva uživatelé hodnotili jako přínosnou pro ně samé a pro posledního uživatele by byla aplikace spíše zátěží, ovšem poznamenal, že aplikace má určitý potenciál.

Kapitola 8

Závěr

Cílem této diplomové práce bylo navrhnout, vytvořit a otestovat informační systém pro drobného zemědělce. Nejprve byly uvedeny současné možnosti vývoje informačních systémů. Dále byl popsán systém Ministerstva zemědělství pro zobrazování výskytu škůdců na území České republiky.

V další části práce byly popsány požadavky na výsledný systém a na jejich základě byla provedena analýza těchto požadavků. Z vytvořené analýzy byl poté proveden návrh celého systému. Dále byla vybrána implementační technologie pro vývoj.

Systém byl implementován jako desktopová aplikace běžící na platformě .NET. Pro tuto implementaci byl zvolen návrhový vzor MVVM. Celý systém a způsob implementace jednotlivých vrstev zvoleného návrhového systému byl popsán společně s ukázkou komunikace mezi jednotlivými vrstvami.

V této části popisu implementace byl také detailně ukázán způsob získávání informací o škůdcích ze serveru Ministerstva zemědělství. Jelikož je systém na sledování škůdců dostupný pouze jako webová aplikace, musel být způsob získávání informací zjištěn z analýzy komunikace mezi webovým klientem a serverem.

Po implementaci byl celý systém otestován. Při testování byly provedeny akceptační testy, které hodnotí, zda byly splněny všechny požadavky systému. Dále bylo také provedeno uživatelské testování, při kterém tři uživatelé odpovídali na otázky v dotazníku.

V budoucnu je možné aplikaci rozšířit o možnost zobrazování informací z dalších serverů Ministerstva zemědělství. Dále by bylo možné vytvořit i webovou verzi systému. Toto vytvoření by nemělo být implementačně náročné, neboť by bylo možné využít open-source framework *DOTVVM*, který by využil stávající implementaci. Při dalším vývoji by systém mohl uživateli poskytovat možnost generování různých tiskových sestav, které by uživatel mohl využít.

Implementovaný systém splňuje veškeré definované požadavky a jeho hlavním přínosem je možnost zobrazení výskytu škůdců, díky kterým může uživatel rychle zjistit aktuální výskyty v okolí a případně provést preventivní opatření (postřik, odpuzovače apod.). Z výsledků uživatelského testování lze říci, že systém je navržen přehledně a má potenciál.

Literatura

[1] *5: Implementing the MVVM Pattern Using the Prism Library 5.0 for WPF*. Microsoft, [Online; navštíveno 1.12.2016].
URL https://msdn.microsoft.com/en-us/library/gg405484(v=pandp.40).aspx

[2] *Bootstrap - The world's most popular mobile-first and responsive front-end framework.* . Bootstrap, [Online; navštíveno 28.11.2016].
URL http://www.getbootstrap.com

[3] *Data Binding Overview*. Microsoft, [Online; navštíveno 16.12.2016].
URL https://msdn.microsoft.com/en-us/library/ms752347(v=vs.110).aspx

[4] *Getting Started (WPF)*. Microsoft, [Online; navštíveno 16.12.2016].
URL https://msdn.microsoft.com/en-us/library/ms742119(v=vs.110).aspx

[5] *Introduction to Entity Framework*. Microsoft, [Online; navštíveno 17.12.2016].
URL https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx

[6] *MVVM: Model-View-ViewModel*. Tomáš Herceg, Tomáš Jecha, [Online; navštíveno 1.12.2016].
URL http://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel

[7] *PHP 5 Introduction*. Refsnes Data, [Online; navštíveno 29.11.2016].
URL http://www.w3schools.com/php/php_intro.asp

[8] *PHP: Hypertex preprocessor*. PHP, [Online; navštíveno 30.11.2016].
URL http://www.php.net

[9] *PHP MySQL Database*. Refsnes Data, [Online; navštíveno 29.11.2016].
URL http://www.w3schools.com/php/php_mysql_intro.asp

[10] *Souřadnicové systémy*. ČÚZK, [Online; navštíveno 20.12.2016].
URL http://geoportal.cuzk.cz/(S(xlsxeqrmpjwkofzbejurjuid))/Default.aspx?mode=TextMeta&side=sit.trans&text=souradsystemy

[11] *What is a Thin Client* . Devon IT, [Online; navštíveno 15.11.2016].
URL http://www.devonit.com/thin-client-education

[12] *WMS reference*. Open Source Geospatial Foundation, [Online; navštíveno 20.12.2016].
URL http://docs.geoserver.org/stable/en/user/services/wms/reference.html

[13] Český statistický úřad: *Zemědělství* . Statistická ročenka České republiky - 2014, [Online; navštíveno 10.11.2016].
URL https://www.czso.cz/csu/czso/320198-14-r_2014-1300

[14] Hruška T., K. Z.: *Informační systémy (IIS,PIS) Studijní opora*. 2012, [Online; navštíveno 15.11.2016].
URL https://www.fit.vutbr.cz/study/courses/WAP/private/opory/

[15] Pluskal J., G. R.: *Přednášky k předmětu IW5*. FIT VUT, [Online; navštíveno 20.11.2016].
URL http://www.fit.vutbr.cz/study/courses/IW1/public/

Kapitola 9

Obsah přiloženého CD

Přiložené CD obsahuje následující adresáře:

- **src** - Zdrojové soubory
- **doc** - Elektronická verze dokumentace
- **latex** - Zdrojové soubory dokumentace
- **screens** - Snímky hlavních obrazovek aplikace