**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačního inženýrství**


**Bakalářská práce**

# Automatizace podnikových procesů v MS Office

**Ladislav Marks**

**© 2021 ČZU v Praze**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Ladislav Marks

Systémové inženýrství a informatika
Informatika

Název práce

**Automatizace podnikových procesů v MS Office**

Název anglicky

**Bussiness process automation using MS Office**

---

**Cíle práce**

Cílem bakalářské práce je představení procesu vývoje maker v jazyce VBA, umožňujícím automatizovat úlohy v aplikacích Microsoft Office, které usnadní a zrychlí jejich vykonávání. Tímto procesem dojde k zamezení vykonání chyb uživatelem, které by mohly narušit konzistenci dat anebo by vedly k jejich ztrátě. V práci budou popsány základy jazyka, používané konstrukce, prostředky pro práci s ostatními produkty balíčku Microsoft Office a objektový model produktu Microsoft Excel. V praktické části je pak cílem demonstrace vývoje modelové aplikace, která bude pracovat s prostředky produktů balíčku Microsoft Office.

**Metodika**

Metodikou práce v teoretické části je analýza vybraných technologií studiem odborné literatury. Praktická část se věnuje návrhu a implementaci softwarového řešení s využitím nabytých teoretických východisek k řešení modelové situace. Následně bude zhodnoceno vytvořené řešení a možnosti jeho budoucího rozšíření.

**Doporučený rozsah práce**

35-40 stran

**Klíčová slova**

VBA, Microsoft Office, Automatizace, Podnikové procesy, Microsoft Excel, Tabulkový procesor

---

**Doporučené zdroje informací**

https://docs.microsoft.com/en-us/
KRÁL, M. *Excel VBA : výukový kurz.* Brno: Computer Press, 2010. ISBN 978-80-251-2358-4.
WALKENBACH, John. Microsoft Office Excel 2007: programování ve VBA. Brno: Computer Press, 2008. Programování (Computer Press). ISBN 978-80-251-2011-8.

---

**Předběžný termín obhajoby**

2020/21 LS – PEF

**Vedoucí práce**

Ing. Jiří Brožek, Ph.D.

**Garantující pracoviště**

Katedra informačního inženýrství

Elektronicky schváleno dne 23. 2. 2021

**Ing. Martin Pelikán, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 23. 2. 2021

**Ing. Martin Pelikán, Ph.D.**

Děkan

V Praze dne 05. 03. 2021

**Čestné prohlášení**

Prohlašuji, že svou bakalářskou práci "Automatizace podnikových procesů v MS Office" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 13. března 2021 _____

**Poděkování**

Rád bych touto cestou poděkoval panu Ing. Jiřímu Brožkovi, Ph.D. za vstřícnost, ochotu a poskytnutí cenných rad při zpracování mé bakalářské práce.

# Automatizace podnikových procesů v MS Office

**Abstrakt**

Tato práce se věnuje problematice vývoje maker pro automatizaci procesů v tabulkovém procesoru Excel od společnosti Microsoft. Práce je rozdělena na teoretickou a praktickou část. Cílem je analýza technologií využitelných pro automatizaci a následná demonstrace vývoje modelové aplikace.

V teoretické části je nejprve shrnut historický vývoj tabulkových procesorů. Dále jsou představeny aplikace balíčku Microsoft Office. Následně se práce věnuje analýze jazyka VBA a objektového modelu aplikace Microsoft Excel. Také jsou analyzovány prostředky ostatních aplikací a práce s nimi.

V praktické části je následně demonstrován postup při tvorbě aplikace pro evidenci záznamů výpůjček plošin v podniku.

**Klíčová slova:** VBA, Microsoft Office, Automatizace, Podnikové procesy, Microsoft Excel, Tabulkový procesor

# Business process automation using MS Office

**Abstract**

This thesis is dedicated to issues of macro development for process automation in the Microsoft Excel spreadsheet. The work is separated into a theoretical and into a practical part. Its aim is an analysis of the technologies usable for automation and a subsequent demonstration of a model application development.

The theoretical part contains a summary of the historical progression of spreadsheets, thereafter the applications of the Microsoft Office package are introduced. The work then analyses the VBA language and the object model of the Microsoft Excel application. The means and tools of other applications and are later analyzed as well.

The practical part demonstrates the procedure of developing an application, containing platform loan records in a company.

**Keywords**: VBA, Microsoft Office, Automation, Business processes, Microsoft Excel, Spreadsheet

# Obsah

# Seznam obrázků



# Seznam tabulek



# Seznam kódů

# Seznam použitých zkratek

VBA – Visual Basic for Applications

OLE – Object Linking and Embedding

API – Application Programming Interface

# 1 Úvod

Tabulkové procesory se již řadu let hojně využívají v kancelářském prostředí pro práci s daty. Přitom jsou datové struktury často neodborně vytvořeny, což může následně v případě vzniklých specifických požadavků ztěžovat práci s nimi. Některé operace pak nemusí být kvůli nevhodně navržené struktuře možné vůbec.

Mnohé úkony lze ve spojení s vhodně navrženou datovou strukturou jednoduše automatizovat a tím zabránit chybným operacím, kterých by se uživatel mohl dopustit. V případě manuálně prováděných opakujících se úkonů toto riziko logicky narůstá.

Mezi používané tabulkové procesory patří Microsoft Excel. Ten nabízí možnost vytváření maker pro automatizaci procesů v jazyce Visual Basic for Applications (VBA). Při vývoji těchto maker je možné manipulovat s množstvím vestavěných objektů tabulkového procesoru, mezi které patří buňky, jejich oblasti anebo celé listy. Zároveň je možné vytvářet nejrůznější formuláře pro ovládání sešitu. Jazyk VBA lze využít pro automatizaci i v ostatních aplikacích balíčku Microsoft Office. Díky provázanosti aplikací tohoto balíčku zároveň lze také ovládat více aplikací pomocí jednoho kódu. Tyto aplikace se od sebe liší svým objektovým modelem, ale jazyk se používá stejný.

# 2 Cíl práce a metodika

## 2.1 Cíl

Cílem bakalářské práce je představení procesu vývoje maker v jazyce VBA, umožňujícím automatizovat úlohy v aplikacích Microsoft Office, které usnadní a zrychlí jejich vykonávání. Tímto procesem dojde k zamezení vykonání chyb uživatelem, které by mohly narušit konzistenci dat anebo by vedly k jejich ztrátě. V práci budou popsány základy jazyka, používané konstrukce, prostředky pro práci s ostatními produkty balíčku Microsoft Office a objektový model produktu Microsoft Excel. V praktické části je pak cílem demonstrace vývoje modelové aplikace, která bude pracovat s prostředky produktů balíčku Microsoft Office.

## 2.2 Metodika

Metodikou práce v teoretické části je analýza vybraných technologií studiem odborné literatury. Praktická část se věnuje návrhu a implementaci softwarového řešení s využitím nabytých teoretických východisek k řešení modelové situace. Následně bude zhodnoceno vytvořené řešení a možnosti jeho budoucího rozšíření.

# 3 Teoretická východiska

## 3.1 Tabulkový procesor

„V oblasti účetního žargonu byl a je tabulkový kalkulátor velký list papíru se sloupci a řádky, který organizuje údaje o transakcích, aby je podnikatel mohl prozkoumat. Rozloží nebo zobrazí všechny náklady, příjmy, daně a další související údaje na jeden list papíru, aby je manažer mohl prozkoumat při rozhodování.“ Tabulkový procesor v elektronické podobě ukládá data do buněk definovaných sloupci a řádky. Pomocí vzorců je možné tyto data sumarizovat a poskytovat uživateli souhrnné informace o nich. [1]

## 3.2 Předchůdci Microsoft Excel

### 3.2.1 VisiCalc a SuperCalc

Historie tabulkových procesorů sahá až do 70. let 20. století, kdy díky Danu Bricklinovi a Bobu Frankstonovi spatřil světlo světa VisiCalc, předchůdce všech dalších elektronických tabulkových procesorů. Tento program mohl být součástí počítače Apple II a definoval funkce jeho nástupců, mezi které patřilo již dříve zmiňované používání vzorců a ukládání hodnot do buněk označených řádkem a sloupcem. [2]

Na počátku program obsahoval sešit o velikosti 5 sloupců a 20 řádků. Frankston poté optimalizoval kód programu tak, že zabíral 20 kB paměti a mohl být spuštěn na mikropočítači. VisiCalc hned po uvedení dosáhl velké úspěšnosti a prodán byl asi 1 milion kopií. [1]

Během toho vznikl obdobný program SuperCalc od společnosti Socrim, který byl přímou konkurencí pro VisiCalc. S příchodem IBM PC vznikly nové verze těchto programů určených pro tento počítač, který odstartoval éru nové hardwarové platformy. I přes své nedostatky, mezi které patřil např. limitovaný počet znaků v buňce, byly tyto programy ve své době inovativním řešením pro manipulaci s daty. [2]

### 3.2.2 Lotus 1-2-3

Úspěch VisiCalc vyvolal u skupinky nadšenců ze společnosti Lotus Development Corporation potřebu vytvořit tabulkový procesor s novou koncepcí a vylepšenými funkcemi. Tato společnost vedená Mitchem Kaporem a Jonathanem Sachsem uvedla v roce 1983 na trh produkt 1-2-3. I přes svou vyšší cenu 495 dolarů se stal okamžitě oblíbeným. [2]

Lotus 1-2-3 přinesl všechny předchozí nástroje ve vylepšeném podání. Hlavní změnou od svých předchůdců byla ale podpora maker, díky kterým bylo možné automatizovat opakující se činnosti a tím pádem zvýšit produktivitu uživatelů. I když tvorba maker spočívala pouze v zaznamenávání klávesových úhozů a jejich následným přehráváním, šlo o opravdovou revoluci. [2]

Úspěch Lotus 1-2-3 skončil s příchodem operačního systému Windows 3.0 od společnosti Microsoft v roce 1990. Díky změně používání počítače, které tento systém přinesl, nebyl Lotus 1-2-3 nikdy vážným konkurentem Microsoft Excel. Přicházely i další verze programu, avšak na jeho uživatele již lze narazit jen zřídka. [2]

### 3.2.3 Quattro Pro

V roce 1987 přinesla na trh společnost Borland International svůj tabulkový procesor Quattro. Jednalo se o levnější klon Lotus 1-2-3 s větší nabídkou nástrojů a odlišným systémem nabídek. Hlavní bylo, že zajišťoval kompatibilitu maker obou programů a nabízel stejné příkazy. [2]

V roce 1989 byla vydána výkonnější verze produktu, Quattro Pro. Ta umožňovala např. práci s více listy na jedné obrazovce a měnitelnost velikosti jejich zobrazení i přes absenci grafického rozhraní. Po několika dalších vydaných verzích vypadal Quattro Pro jako nejlepší tabulkový procesor až do příchodu Excel 5. [2]

## 3.3 Microsoft Excel

V této kapitole byly využity zdroje: [2] [3]

V 80. letech minulého století vyvinula společnost Microsoft svůj tabulkový procesor jménem MultiPlan. Byl určen pro počítače s operačním systémem CP/M a později i pro platformy Apple II, Apple III, XENIX a MS-DOS. Kvůli složitému používání a na tu dobu odlišnému uživatelskému rozhraní ale nikdy nepředčil svého hlavního konkurenta, Lotus 1-2-3.

Společnost se tedy rozhodla vyvinout nový tabulkový procesor pojmenovaný Excel. Narozdíl o svého předchůdce disponoval grafickým rozhraním a ve verzi 2 přinesl výkonný jazyk maker XLM, předchůdce VBA.

Ve verzi 3 vydané roku 1990 přišly další zajímavé doplňky, jako například podpora práce s externími databázemi.

Verze 4 vydaná roku 1992 přinesla uživatelsky přívětivější prostředí, ale kvůli zhoršujícím se vztahům Microsoftu s další společností IBM nebyla nikdy vydána pro operační systém OS/2. Ani následující verze již pro tento systém nebyly vydány.

V roce 1995 byla vydána nová verze Excel 5. Následně byla vydána i verze Excel 7 určená pro nově vydaný operační systém Windows 95, ta se ale od svého předchůdce nijak příliš nelišila. Došlo zde k optimalizaci zdrojového kódu a výsledkem byla znatelně větší rychlost při práci s tímto programem. Byla zajištěna také kompatibilita souborů s předešlou verzí Excel 5, ale u vytvářených aplikací ve VBA nebyla kompatibilita stoprocentní.

Verze 8 vydaná v roce 1997 přinesla, kromě desítek obecných novinek, nové rozhraní pro vývoj uživatelských aplikací. Byla také přidána možnost vytvářet dialogová okna. V dalších verzích, Excel 2000, Excel XP a Excel 2003, byly vydány pouze nevýznamné novinky.

Zlom nastal s vydáním verze Excel 2007, která přinesla kompletně nové, uživatelsky přívětivější rozhraní. Dále přinesla řadu užitečných nástrojů, jako například podmíněné formátování buněk, úpravy vzhledu grafů a další.

Další vydané verze stále přinášely nové funkce, jako například podpora běhu na více jádrech, dynamické doplňování dat v tabulkách, nástroje Power Pivot, Power Query a Power Maps určené pro analýzu velkých objemů dat a mnoho dalších. Microsoft začal nově vydávat verze Excel 365, které jsou uživatelům nabízeny ve formě předplatného.

## 3.4 Ostatní součásti balíku Microsoft Office

Předností produktů v balíku Microsoft Office je schopnost dobře mezi sebou komunikovat. Disponují totožným rozhraním a podporou jazyka VBA. V této oblasti tyto produkty vynikají. [2]

### 3.4.1 Microsoft Access

„Microsoft Access je systém řízení báze dat (SŘBD) od Microsoftu, který kombinuje relační stroj Microsoft Jet Database Engine s grafickým rozhraním a vývojářskými nástroji. Je obsažen ve vyšších edicích balíčku Microsoft Office.“ [4]

Je jedním z produktů pro správu dat od společnosti Microsoft. Slouží k propojení souvisejících informací, stejně jako ostatní relační databáze. Mezi jeho nástroje patří propojení dat z různých zdrojů, jako jsou ostatní oblíbené databázové systémy, sešity

programu Microsoft Excel, nebo XML a textové soubory. Dokáže tedy pracovat s různými datovými formáty. [4]

### 3.4.2 Microsoft Outlook

Microsoft Outlook je poštovní klient, jehož hlavními funkcemi je příjímání a odesílání elektronické pošty. Je obsažen v některých verzích balíčku Microsoft Office. Uveden na trh byl v roce 1997 a byl obsažen v později vydaném operačním systému Windows XP, přičemž první verze byla i poslední, která byla zdarma. [5]

Jedná se o aplikaci, která nabízí spoustu nástrojů. Pokud potřebuje uživatel pouze přijímat a odesílat zprávy, není potřeba si ji pořizovat. K tomu mu poslouží aplikace Pošta, která je k dispozici v operačním systému Windows od verze 8.1. Oproti této aplikaci nabízí Microsoft Outlook možnost synchronizace osobních dat prostřednictvím účtu Microsoft, uspořádání zpráv do složek na základě pravidel, nastavení odesílání automatických odpovědí, označení zpráv a spoustu dalších. Dále nabízí např. adresář kontaktů, kalendář a seznam úkolů. [5]

### 3.4.3 Microsoft Word

Microsoft Word je dalším z produktů balíku Microsoft Office. Slouží k vytváření a úpravě textových dokumentů. Stejně jako u Microsoft Outlook, pokud uživatelovy potřeby nepřesahují rámec běžných úprav textu a základního stylování, nabízí Microsoft alternativu v podobě aplikace WordPad, která je zdarma a je součástí operačního systému Microsoft Windows 7 a vyšší. Microsoft Word nabízí jednoduché a rychlé formátování textu pomocí přednastavených stylů, které zahrnují různé odsazení textu, řádkování apod. Také lze vkládat další netextové součásti dokumentu, jako grafy, tabulky, obrázky a obrazce. Je možné vložit obsah dokumentu, seznam literatury, nastavit číslování stránek atd. [6]

## 3.5 VBA

### 3.5.1 Předchůdci

První předchůdce VBA byl jazyk BASIC (v překladu víceúčelový jazyk symbolických instrukcí pro začátečníky) vyvinutý v 60. letech. Původně byl určen studentům na naučení technik vývoje aplikací. Z tohoto jazyka se následně vyvinuly další programovací jazyky. [2]

V 90. letech byl společností Microsoft vydán Visual Basic pro Windows. Narozdíl od svého předchůdce není interpretovaným jazykem, ale aplikace v něm vyvinuté jsou zkompilovány do binárního kódu a tím je zvýšena rychlost programu. Z tohoto jazyka se dále vyvinul jazyk VBA. [2]

### 3.5.2 **Charakteristika jazyka**

„VBA je počítačový programovací jazyk určený k použití ve spojení s hostitelskou softwarovou aplikací, jako je např. textový procesor. V takové situaci koncový uživatel hostitelské aplikace používá jazyk VBA k zápisu programů, které mají přístup k údajům a funkcím hostitelské aplikace a mohou s nimi manipulovat.“ [7]

Kód aplikace se nachází v modulech. Moduly jsou obsaženy v samotných souborech produktů Microsoft Office. V modulu je možné vytvářet procedury a funkce, což jsou samostatné kusy kódu vykonávající jedinečnou činnost. Procedury po vykonání nevrací žádnou hodnotu, funkce vrací hodnoty předem definovaného typu. Pro manipulaci se součástmi aplikace se používají vestavěné třídy objektů, které mají své vlastnosti a metody. Objekty jsou hierarchicky uspořádány. Zatímco vlastnosti představují jakési nastavení objektu, metody lze chápat jako činnosti, které může objekt vykonávat. Tyto objekty lze také uspořádat do kolekcí. [2] Dále VBA obsahuje většinu elementů a konstrukcí, které jsou součástí dalších programovacích jazyků, jako proměnné a jejich pole, cykly a spoustu dalších. [2] [8]

Jazyk VBA podporují i další produkty balíku Microsoft Office, liší se od sebe pouze podobou objektového modelu. Po nastudování jazyka je pro vývojáře tedy jednoduché vytvářet makra pro všechny tyto produkty. [2]

### 3.5.3 **Budoucnost**

Za alternativu k VBA může být považován jazyk Javascript a s ním spojené aplikační rozhraní. Výhodou tohoto jazyka je možnost vytvořit programy, které mohou potenciálně fungovat jak na klasickém počítači, tak na mobilním zařízení. Oproti tomu má ale VBA ve spoustě aspektech výhodu. Jednou z nich je existence rozsáhlé komunity vývojářů, díku čemuž je velice snadné nalézt řešení problému. Další je skutečnost, že společnosti již používají funkční softwarová řešení, není tedy potřeba je nahrazovat. [9] Podle Walkenbacha je také snazší se naučit programovat v jazyce VBA než v jeho alternativách. [2]

## 3.6 VBA Excel

### 3.6.1 Proměnné

„Proměnná je jednoduše pojmenovaná pozice v paměti vašeho počítače určená pro uložení něčeho (obvykle hodnoty). Proměnné v sobě mohou ukládat širokou paletu různých datových typů – od jednoduchých logických hodnot (*True* nebo *False*) až po velká čísla s dvojitou přesností.“ [2] Do proměnných je také možné ukládat objekty, jako např. sešit, graf nebo třeba externí aplikaci Microsoft Word. [10] V tomto případě hovoříme o objektových proměnných. [11]

Proměnné lze deklarovat na úrovni procedury nebo na úrovni modulu. Zatímco proměnnou deklarovanou na úrovni procedury lze použít pouze v dané proceduře, ve které je deklarována, tak deklarace na úrovni modulu nám umožní ji použít ve kterékoli proceduře v daném modulu. [11]

#### 3.6.1.1 Pojmenovávání proměnných

Podle Walkenbacha [2] je vhodné pojmenovávat proměnné systematicky tak, aby název co nejlépe vyjadřoval obsah či účel proměnné. Usnadní to tak vývoj aplikace. Zásady pro pojmenovávání proměnných jsou:

- Název může kromě písmen obsahovat také číslice a některé z interpunkčních znamének
- Název musí začínat písmenem
- VBA nerozlišuje malá a velká písmena v názvech, tzn. např. UrokovaMira je shodný název s urokovamira
- Nelze používat čárky a tečky v názvech
- Nelze v názvech používat speciální znaky (#, $, %, & nebo !)

Dále uvádí příklady vhodného pojmenování různých typů proměnných:

```
x = 1
UrokovaMira = 0.075
VyskaSplatky = 243089
ZapsanaData = False
x = x + 1
MojeCislo = TvojeCislo * 1.25
Uzivatel = "Josef Novák"
DatumZacatku = #3/14/2006#
```

*Kód 1 - Příklady pojmenovávání proměnných (zdroj: [2])*

### 3.6.1.2 **Datové typy**

„Datový typ proměnné říká, jakým způsobem budou data uložena v paměti – jako celá čísla, reálná čísla, řetězce a tak dále.“ Proměnné různých datových typů zabírají různou velikost paměti. Čím méně paměti proměnné používané v proceduře zabírají, tím rychleji se procedura vykoná. Proto je vhodné využít typ s co nejmenší alokovanou pamětí s ohledem na dostatečný rozsah hodnot. [2] Následující tabulka obsahuje datové typy s rozsahem hodnot i velikostí v paměti, které je možné využít:

*Tabulka 1 - Přehled typů proměnných (zdroj: [12])*

| **Datový typ** | **Velikost v paměti** | **Rozsah hodnot** |
|---|---|---|
| Boolean | 2 byty | True nebo False |
| Byte | 1 byte | 0 až 255 |
| Collection | Neznámé | Neznámé |
| Currency (scaled integer) | 8 bytů | -922 337 203 685 477.5808 až 922 337 203 685 477.5807 |
| Date | 8 bytů | 1. ledna 100 až 31. prosince 9999 |
| Decimal | 14 bytů | +/-79 228 162 514 264 337 593 543 950 335 bez desetinné čárky; +/-7,9228162514264337593543950335 s 28 místy za desetinnou čárkou |
| Dictionary | Neznámé | Neznámé |
| Double (double-precision floating-point) | 8 bytů | -1,79769313486231E308 až -4,94065645841247E-324 pro záporné hodnoty; 4,94065645841247E-324 až 1,79769313486232E308 pro kladné hodnoty |
| Integer | 2 byty | -32 768 až 32 767 |
| Long (Long integer) | 4 byty | -2 147 483 648 až 2 147 483 647 |
| LongLong (LongLong integer) | 8 bytů | -9 223 372 036 854 775 808 až 9 223 372 036 854 775 807, platný pouze na 64-bitových systémech |
| LongPtr (Long integer na 32-bitových systémech LongLong integer na 64-bitových systémech) | 4 byty na 32-bitových systémech, 8 bytů na 64-bitových systémech | -2 147 483 648 až 2 147 483 647 na 32-bitových systémech; -9 223 372 036 854 775 808 až 9 223 372 036 854 775 807 na 64-bitových systémech |

| Object | 4 byty | Odkaz na objekt |
|---|---|---|
| Single (single-precision floating-point) | 4 byty | -3,402823E38 až -1,401298E-45 pro záporné hodnoty, 1,401298E-45 až 3,402823E38 pro kladné hodnoty |
| String (libovolná délka) | 10 bytů + délka řetězce | 0 až přibližně 2 miliardy |
| String (fixní délka) | Délka řetězce | 1 až přibližně 65 400 |
| Variant (s čísly) | 16 bytů | Jakákoli hodnota až do rozsahu Double |
| Variant (se znaky) | 22 bytů + délka řetězce (24 bytů on 64-bit systems) | Stejný rozsah jako pro String s libovolnou délkou |
| Uživatelsky definovaný | Velikost požadovaná prvky | Rozsah každého prvku je shodný s rozsahem jeho datového typu |

#### 3.6.1.3 **Deklarace proměnných**

Tato kapitola byla zpracována dle [2] [11].

U VBA existuje více způsobů deklarace proměnných. První z nich je způsob bez deklarace datového typu. Tímto způsobem deklarované proměnné mají nastaven typ *Variant*. Tyto proměnné ale zabírají více paměti a při jejich používání v kódu můžou nastat chyby, např. když dojde k překlepu v jejich názvu při dalším používání.

```
a = 10
b = "test"
```

*Kód 2 - Deklarace proměnné 1 (zdroj: vlastní)*

Druhým způsobem je výslovné uvedení datového typu proměnné. Tento způsob je oproti předchozímu vhodnější jak kvůli již výše uvedeném riziku překlepů při používání proměnných, tak i kvůli rychlejšímu běhu programu a efektivnějšímu využití paměti. Pro prevenci před deklarováním proměnných prvním způsobem je možné vynutit výslovnou deklaraci příkazem *Option Explicit* na první řádek modulu. Toto lze nastavit i v editoru pro všechny nově vytvořené moduly v nastavení ve volbě *Require Variable Declaration*.

```
Dim a As Long
a = 10
Dim b As String: b = "test"
```

*Kód 3 - Deklarace proměnné 2 (zdroj: vlastní)*

Třetím způsobem je možné deklarovat přístupnost proměnné. Toto se týká pouze deklarace proměnné na úrovni modulu. Přístupnost lze upravit výrazy *Public* nebo *Private* před názvem proměnné. V případě uvedení *Public* může být proměnná použita jakoukoli procedurou v projektu, v případě uvedení výrazu *Private* může být k proměnné přistupováno pouze z modulu, ve kterém byla vytvořena. V případě deklarace proměnné na úrovni modulu s použitím výrazu *Dim* je tato proměnná považována jako privátní.

```
Public a As Long
Private b As String
```

*Kód 4 - Deklarace proměnné 3. (zdroj: vlastní)*

Jako další způsob deklarování proměnné je použití výrazu *Static* před název proměnné. O takové proměnné lze hovořit jako o statické. Lze ji deklarovat pouze na úrovni procedury. V této proměnné je uložena její hodnota i po ukončení procedury a pro ztrátu její hodnoty je nutné, aby byla procedura ukončena příkazem *End*, nikoliv *End Sub*.

VBA dále umožňuje vytvářet tzn. konstanty. Tímto termínem lze označit pojmenovanou hodnotu, která vlastně není proměnnou, protože ji od deklarace není možné změnit. Způsob deklarace konstanty se od stejné operace u proměnné liší nahrazení výrazu *Dim* klíčovým slovem *Const*. Využití konstant je doporučeno v případě, kdy je potřeba použít specifickou hodnotu na více místech v kódu. Tato hodnota se v případě potřeby může změnit pouze na jednom místě.

#### 3.6.1.4 **Objektové proměnné**

Objektovou proměnnou se rozumí proměnná, která obsahuje odkaz na objekt. Pro přiřazení odkazu na objekt se používají klíčová slova *Set* před název proměnné a *New* sloužící k vytvoření objektu. V následujícím kódu se na prvním řádku deklaruje pole, které ale neobsahuje žádné odkazy na objekty. Tyto odkazy se přidávají až v následujících řádcích společně s vytvářením samotných objektů: [13]

```
Dim myChildForms(1 To 4) As Form1
Set myChildForms(1) = New Form1
Set myChildForms(2) = New Form1
Set myChildForms(3) = New Form1
Set myChildForms(4) = New Form1
```

*Kód 5 - Práce s objektovými proměnnými (zdroj: [13])*

Přiřazení hodnoty *Nothing* objektové proměnné způsobí uvolnění systémových a paměťových zdrojů, které objekt, jehož odkaz proměnná obsahovala, zabíral. [13]

V případě použití objektu jiné aplikace Microsoft Office je nutné přidat odkaz na její knihovnu v záložce References v editoru. Po následné deklaraci proměnné a jejího typu je nutné pro vytvoření objektu použít funkci *CreateObject*, která jako parametr přijímá jeden z programových identifikátorů OLE (OLE Programmatic Identifiers). [14]

```
Dim appWD As Word.Application
Set appWD = CreateObject("Word.Application")
```

*Kód 6 - Vytvoření nového objektu aplikace Microsoft Word (zdroj: [14])*

#### 3.6.1.5 **Pole a kolekce**

V této části bylo čerpáno ze zdrojů: [15] [16] [17] [11].

„Kolekce a pole se používají pro seskupení proměnných. Obě entity ukládají sadu podobných položek, např. seznam známek studentů nebo názvy zemí. Použití kolekce nebo pole umožňuje rychle a snadno manipulovat s velkým počtem položek.“ [15]

Při deklaraci pole je nutné uvést maximální počet prvků, tzn. velikost pole. Tyto prvky jsou v poli indexovány. V základu je index prvního elementu 0. Je ale možné první index změnit, a to výrazem *Option Base*.

```
Dim varData(3) As Variant
varData(0) = "Claudia Bendel"
```

*Kód 7 - Deklarace statického pole a přístup k prvkům (zdroj: [16])*

Ve VBA lze vytvořit i pole s dynamickým počtem prvků. Pro vytvoření je nutné nejdříve deklarovat pole pomocí klíčového slova *Dim* s vynecháním počtu prvků. Pro nastavení počáteční velikosti pole se *Dim* nahradí výrazem *ReDim*. Je důležité poznamenat, že při opětovné změně velikosti pole pomocí *ReDim* se obsah pole odstraní. Pro zachování hodnot při této operaci se musí použít výraz *ReDim Preserve*.

```
Dim strNames() As String
ReDim strNames(1 To 3)
ReDim Preserve strNames(1 To 4)
```

*Kód 8 - Dynamicé pole (zdroj: [17])*

Zatímco u pole je potřeba znát jeho potřebnou velikost při jeho deklaraci, u kolekce to nutné není. Nevýhodou kolekce oproti poli je to, že v případě, kdy je deklarován datový typ prvků jedním ze základních datových typů (např. Long, String), tak jsou prvky po přidání pouze ke čtení. To nenastává u kolekcí, jejichž prvky jsou odkazy na objekty.

Jelikož se v případě kolekce jedná o objektovou proměnnou, při její deklaraci je nutné klíčové slovo *New*.

```
Dim coll As New Collection ' deklarace
coll.Add "text" ' přidání prvku
Debug.Print coll(1) ' přístup k prvku
Set coll = New Collection ' odstranění všech prvků
```

*Kód 9 - Operace s kolekcemi (zdroj: [15])*

### 3.6.2 **Základní konstrukce**

#### 3.6.2.1 **Podmíněné větvení kódu**

Díky konstrukcím pro větvení kódu je možné, podle určitého kritéria, provádět různé operace. Ve VBA mezi tyto konstrukce patří:

- *If ... Else ... End If*

- *Select Case* [2]

Při specifikaci podmínky se používají operátory. Mezi tyto operátory patří:

- = je rovno
- <> je jiné než
- < je menší než
- <= je menší nebo rovno než
- > je větší než
- >= je vetší nebo rovno než
- *AND* – obě podmínky musí být pravdivé
- *OR* – alespoň jedna ze dvou podmínek musí být pravdivá
- *NOT* – podmínka je nepravda [18]

Konstrukce *If ... Else ... End If* patří k nejčastěji používaným. Slouží k podmíněnému provedení jedné či více operací. Část *Else*, která je nepovinná, se používá k provedení příkazů v případě, že podmínka není splněna. Vynecháním této části se při nesplnění podmínky neprovede nic. [2]

```
If [PODMÍNKA ZDE] Then ' => Když je podmínka splněna, tak
'Instrukce v případě splnění
Else ' => Jinak
'Instrukce v případě splnění
End If
```

*Kód 10 - Konstrukce If ... Else ... End If (zdroj: [18])*

Konstrukce *If ... Else ... End If* může být rozšířena o blok *ElseIf*, pomocí kterého je možné přidat více podmínek, které se postupně vyhodnocují. Jestliže je jedna z podmínek pravdivá, podmíněné operace se provedou a další podmínky se již nevyhodnocují. [2]

```
If [PODMÍNKA 1] Then ' => Jestli je podmínka 1 pravdivá, tak
' Instrukce 1
ElseIf [PODMÍNKA 2] Then ' => Když je podmínka 1 nepravda, ale podmínka 2
je pravda, tak
' Instrukce 2
Else ' => Když jsou obě nepravdivé, tak
' Instrukce 3
End If
```

*Kód 11 - Použití ElseIf (zdroj: [18])*

Blok *Select Case* je v dobrou alternativou k předchozí konstrukci v situacích, kdy je potřeba rozhodnout mezi třemi a více možnostmi. [2]

```
Select Case Time
    Case Is < 0.5
        Msg = "Dobré dopoledne"
    Case 0.5 To 0.75
        Msg = "Dobré odpoledne"
```

```
    Case Else
        Msg = "Dobrý večer"
End Select
```

*Kód 12 - Konstrukce Select Case (zdroj: [2])*

#### 3.6.2.2 **Cykly**

Pro případ potřeby vykonat část kódu opakovaně se využívají cykly. Některé cykly se opakují podle pravdivosti podmínky a některé podle počtu opakovaní, který se předem určí. Ve VBA jsou k použití tyto konstrukce:

- *Do … Loop*
- *For … Next*
- *For Each … Next* [19]
- *While … Wend* [20]

„Konstrukci *Do...Loop* lze použít pro spuštění bloku příkazů s neurčitým počtem opakování. Příkazy se opakují buď, dokud je podmínka splněna, anebo dokud se podmínka nestane splněnou.“ [21] V případě uvedení podmínky u výrazu *Do* je podmínka ukončení cyklu vyhodnocována při začátku každého kroku, zatímco při uvedení u výrazu *Loop* je podmínka vyhodnocována na konci každého kroku. Ke specifikaci podmínky se používají klíčová slova *While*, nebo *Until*. [21]

```
counter = 0
myNum = 20
Do While myNum > 10
    myNum = myNum - 1
    counter = counter + 1
Loop
```

*Kód 13 - Cyklus Do ... Loop 1 (zdroj: [21])*

```
counter = 0
myNum = 9
Do
    myNum = myNum - 1
    counter = counter + 1
Loop While myNum > 10
```

*Kód 14 - Cyklus Do ... Loop 2 (zdroj: [21])*

Cyklus *For … Next* se používá v případě potřeby přesně určit počet opakování. Na začátku je deklarována proměnná, ke které je po provedení příkazů přičtena určitá hodnota. Tato hodnota může být modifikována pomocí klíčového slova *Step*. Jakmile je proměnná rovna hodnotě definované po výrazu *To*, další opakování cyklu se nekoná a cyklus je ukončen. [22]

```
For j = 2 To 10 Step 2
    Total = Total + j
```

```
Next j
MsgBox "Celková hodnota je: " & Total
```

*Kód 15 - Cyklus For ... Next (zdroj: [22])*

Cyklus *For Each ... Next* slouží k provedení operací s poli nebo kolekcemi. Počet opakování je určen automaticky podle počtu prvků. [23]

```
For Each polozka In MojePolozky
    MsgBox polozka
Next polozka
```

*Kód 16 - Cyklus For Each ... Next. (zdroj: vlastní)*

Poslední z cyklů, které je možné použít ve VBA, je *While ... Wend*. Testování podmínky se zde provádí na začátku každého opakování. [20]

```
Dim counter
counter = 0 ' Inicializace proměnné
While counter < 20 ' Testování hodnoty counter
    counter = counter + 1 ' Inkrementace hodnoty Counter
Wend ' Konec cyklu While když Counter > 19
Debug.Print counter ' Výpis 20 v běhovém okně
```

*Kód 17 - Cyklus While ... Wend (zdroj: [20])*

#### 3.6.2.3 With … End With

Tato konstrukce je užitečná v případě, kdy je potřeba vykonat více operací s jedním objektem. Pří provádění jednotlivých operací není nutné se pokaždé odkazovat na ten samý objekt, odkázání se v tomto případě provede pouze jednou. [24] Tímto způsobem může být dosaženo vyšší rychlosti provedení operací s objektem. [2]

```
With MyObject
    .Height = 100 ' Stejné jako MyObject.Height = 100.
    .Caption = "Hello World" ' Stejné jako MyObject.Caption = "Hello
World".
    With .Font
        .Color = Red ' Stejné jako MyObject.Font.Color = Red.
        .Bold = True ' Stejné jako MyObject.Font.Bold = True.
    End With
End With
```

*Kód 18 - Konstrukce With ... End With (zdroj: [24])*

### 3.6.3 Procedury a funkce

Pojem procedura označuje blok kódu, který vykonává nějakou činnost. Funkce je navíc schopna vracet určitou hodnotu. Stejně jako u proměnných lze upravovat jejich přístupnost výrazy *Public* a *Private*. [10] V případě neuvedení výrazu specifikujícím přístupnost je procedura veřejná, tzn. přístupná i z dalších modulů, ale použitím příkazu *Option Private Module* lze zajistit opak. Privátní procedury mj. nelze spouštět v dialogovém okně maker. Použitím výrazu *Static* při deklaraci procedury zajistíme uchovávání hodnot proměnných

v proceduře i po vykonání operací. „Až na několik výjimek musí být všechny příkazy VBA zapsány v procedurách. Výjimky tvoří deklarace proměnných na úrovni modulu, definice uživatelských datových typů a některé další volby s globální platností (např. příkaz *Option Explicit*).“ [2]

Procedury a funkce mohou mít specifikované parametry, což jsou názvy proměnných, se kterými pracují. Těmto proměnným se v okamžiku volání procedury předávají hodnoty, tzn. argumenty. [25]

```
Public Sub Pozdrav()
    Vypis "Dobrý den"
End Sub
Public Sub Vypis(zprava As String)
    MsgBox zprava
End Sub
```

*Kód 19 - Práce s procedurami. (zdroj: vlastní)*

Pro volání procedur existuje ve VBA více způsobů. První z nich je způsob bez závorek uzavírajících předávané argumenty, druhý spočívá v použití klíčového slova *Call* před názvem procedury a uzavření argumentů závorkami. [26] Třetím způsobem je spouštění pomocí metody *Application.Run()*. [2] Při volání funkce za účelem přiřazení její návratové hodnoty do proměnné je nutné uzavřít argumenty závorkami. V případě potřeby je ve VBA možné procedurám a funkcím předávat hodnoty jmenovitě jen některým argumentům. [26]

```
MsgBox Title:="Task Box", Prompt:="Task Completed!"
```

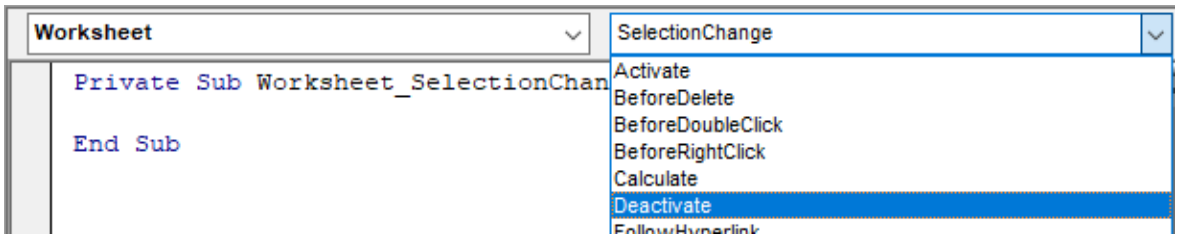
*Kód 20 - Jmenovité předávání argumentů (zdroj: [26])*

### 3.6.4 **Události**

V této kapitole bylo čerpáno z těchto zdrojů: [2] [27]

Pro vykonání příkazů v momentě, kdy nastane nějaká událost, se využívají událostní procedury. V Excelu lze zachytit události různých objektů, jako např. události sešitu, listu, grafu a formuláře. Události *OnTime* a *OnKey* nejsou přidruženy žádnému objektu. Dále je možné sledovat události na úrovni celé aplikace Excel, v tomto případě je ale nutné vytvořit tyto události v modulu třídy.

Objekty Excelu, jako list nebo sešit, mají vždy přiřazen svůj modul, kam lze umístit událostní procedury, např. modul *ThisWorkbook* náležící sešitu aplikace nebo modul *List1* náležící konkrétnímu listu. V těchto modulech je možné vybrat potřebnou událostní proceduru v menu v horní části textového editoru.




*Obrázek 1- Výběr událostí v editoru (zdroj: vlastní)*

Názvy událostí nelze měnit a z pohledu přístupnosti jsou privátní. Některé událostní procedury také přijímají parametry, se kterými je možné poté pracovat, jako např. u události sešitu *SheetActivate*, která je spuštěna při přepnutí z jiného listu. Jako parametr zde je přijímán aktivovaný sešit.

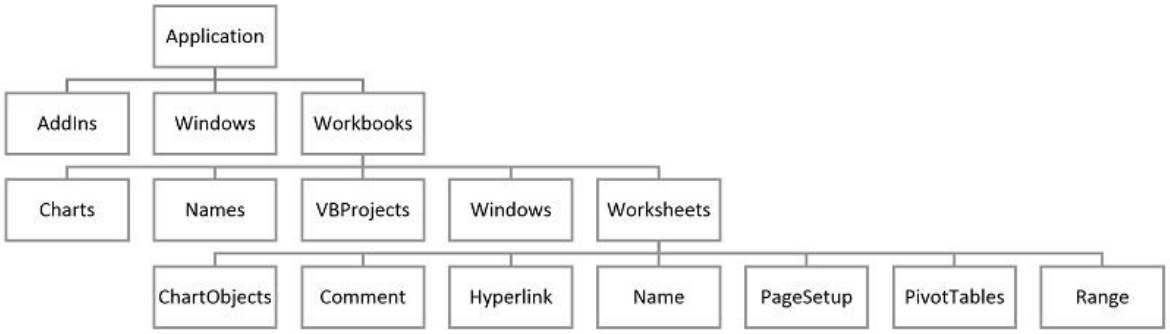
```
Private Sub Workbook_Open()
    For Each polozka In Me.Worksheets
        polozka.Protect "heslo"
    Next polozka
End Sub
```

*Kód 21 - Uzamčení všech listů při otevření sešitu (zdroj: vlastní)*

### 3.6.5 Objektový model

V této kapitole byly využity zdroje: [10] [28] [29]

Objektovým modelem rozumíme hierarchicky uspořádanou skupinu objektů. Například aplikace Word je objektem nejvyšší úrovně a ten obsahuje objekt *Document*. Objekt *Document* zase obsahuje objekt *Paragraph* představující odstavec. Objekty jsou definovány třídou, což je v podstatě šablona pro vytvoření objektu. Každý objekt má své vlastnosti, které je možné nastavovat a metody, které lze volat. Změnou vlastnosti lze docílit určitého chování či vzhledu objektu a volání metody způsobí určitou akci objektu. K podřazenému objektu (vlastnosti) je možné přistoupit s uvedením mateřského objektu. Objekty třídy <*globals*> (např. *ActiveSheet* nebo *ActiveWorkbook*) jsou přístupné přímo bez uvedení mateřského objektu.




*Obrázek 2 - Zjednodušený objektový model Microsoft Excel (zdroj: [29])*

V Microsoft Excel je nejvýše postaveným objektem objekt *Application*. Tento objekt obsahuje všechny ostatní objekty. Užitečnou vlastností je *ScreenUpdating*. Po nastavení této vlastnosti na *False* přestane Excel překreslovat obrazovku a výsledkem je vyšší rychlost provedení maker, která manipulují s ostatními objekty Excelu, např. při zapisování hodnot do buněk. Pro obnovení vykreslování je nutné tuto vlastnost nastavit zpět na *True* nebo je tak učiněno při ukončení běhu makra automaticky. Další užitečnou vlastností objektu *Application* je *DisplayAlerts*. Pomocí této vlastnosti lze zakázat zobrazování upozornění např. při odstraňování listu nebo při ukládání souboru do nového umístění. Mezi často používané vlastnosti lze ještě zařadit např. *EnableEvents* potlačující události, *StatusBar* umožňující vypsat informace o běhu programu do stavové lišty Excelu anebo objekt *WorksheetFunction* sloužící k spuštění vzorců (při zápisu je nutné dodržet anglické znění). Mezi používané metody objektu *Application* patří např. *InputBox*. Tato metoda zobrazí dialogové okno s jedním vstupem. V případě vyplnění vrací text/číslo, v případě kliknutí na tlačítko Cancel vrací logickou hodnotu *False*.

Součástí objektu *Application* je také kolekce sešitů *Workbooks*. Vytvoření nového sešitu se provede metodou *Add* a otevření již existujícího je možné pomocí metody *Open*. Pro zadání cesty k souboru uživatelem je vhodné použít metodu objektu *Application GetOpenFileName*.

Mezi vlastnosti objektu *Workbook* patří mj. kolekce *Worksheets* a *Charts* obsahující pracovní listy a listy s grafy. Tyto druhy objektů jsou pak obsahem univerzální kolekce *Sheets*. K listům v kolekcích lze přistupovat pomocí jména nebo indexu. Metodou *Add* lze přidat nový list a pomocí parametru *Before* nebo *After* určit pozici listu v seznamu.

Objekt *Range* je podle Krále pravděpodobně nejpoužívanějším objektem při programování ve VBA pro Excel. „Oblast, kterou definuje objekt *Range*, může být buňka, souvislá i nesouvislá část buněk, řádek, sloupec nebo jejich libovolná kombinace.“ [10] Oblast buněk je možné určit výběrem jedné buňky, rozsahem řádků či sloupců oddělených středníkem, použitím dvou argumentů označujících levý horní a pravý dolní roh oblasti nebo zápisem více nesouvislých oblastí oddělených čárkami. Další způsob výběru buněk spočívá v použití vlastnosti *Cells*. Umístění buňky je možné specifikovat buď indexy nebo názvy řádku a sloupce.

### 3.6.6 **Ošetření chyb**

V případě, kdy se procedura chystá např. manipulovat s již neexistujícím sešitem nebo zapisovat do zamčené buňky, není možné tuto operaci provést a program již není schopen dále pokračovat. Pro ošetření takovýchto chyb slouží příkaz *On Error*, který se umisťuje před rizikovou část kódu a je schopen stornovat příkaz, při kterém by došlo k havárii.

První varianta příkazu, *On Error Resume Next*, se používá pro ignorování následujících příkazů, u kterých dojde k chybě. Druhou variantou je *On Error GoTo <návěstí>*. Při nastání chyby u jedné z instrukcí program tuto instrukci přeskočí a pokračuje na místě, kde se nachází návěstí. Tato operace umožňuje zamezení zobrazení chybové hlášky a přerušení běhu programu. [27] Pro obnovení standardního zpracování chyb se používá příkaz *On Error Goto 0*. [2]

```
Sub DemoChyba()
    On Error GoTo Handler
    Selection.Value = 123
    Exit Sub
Handler:
    MsgBox "Výběru nelze přiřadit hodnotu"
End Sub
```

*Kód 22 - Použití návěstí při ošetření chyb (zdroj: [2])*

### 3.6.7 **Práce s jinými aplikacemi**

V této kapitole bylo čerpáno z následujících zdrojů: [2].

Jazyk VBA disponuje několika prostředky pro práci s ostatními programy. Pomocí těchto prostředků je možné spouštět např. jinou aplikaci Microsoft Office nebo dávkový soubor systému DOS. Je možné také spouštět mnoho aplikací OS Windows podporující technologii Automation.

Prvním takovým prostředkem je funkce *Shell*. Tato funkce přijímá jako první parametr text s názvem programu a jako druhý modifikátor zobrazení, kdy argument 1 způsobí, že aplikace bude spuštěna v normální velikosti a bude aktivována. Jako návratovou hodnotu tato funkce vrací číslo úlohy aplikace, které lze později využít pro aktivaci okna. Aplikace spuštěna pomocí funkce *Shell* je zpracovávána asynchronně, tzn. následující instrukce jsou dále vykonávány bez čekání na výsledek běhu spuštěné aplikace. Pozastavení kódu do doby, než bude externí aplikace spuštěna, lze provést pomocí cyklu. Ten se opakuje, zatímco funkce *GetExitCodeProcess*, přidružená ke konkrétnímu procesu pomocí funkce *OpenProcess*, vrací hodnotu *&H103*. V těle cyklu se volá příkaz *DoEvents*, který zpracovává události ve Windows.

```
Sub SpustitKalkulacku()
    Dim Program As String
    Dim TaskID As Double
    On Error Resume Next
    Program = "calc.exe"
    TaskID = Shell(Program, 1)
    If Err <> 0 Then
        MsgBox "Nelze spustit " & Program, vbCritical, "Chyba"
    End If
End Sub
```

*Kód 23 - Spuštění kalkulačky pomocí funkce Shell (zdroj: [2])*

Druhým prostředkem je funkce *ShellExecute*. „*ShellExecute* je funkce API (Application Programming Interface) systému Windows, která umí spouštět aplikace. … Tato funkce umí spustit aplikaci i v případě, kdy známe jen název souboru, který v ní má být otevřen (za předpokladu, že daný typ souborů je ve Windows zaregistrován).“ Na začátek modulu je nutné deklarovat API.

Třetím prostředkem je příkaz *AppActivate*. Příkaz *AppActivate* program nespouští, ale aktivuje jeho již spuštěnou instanci, pokud taková instance existuje. Jako parametr přijímá text s titulkem aplikace uvedený v horní liště okna.

Spuštění jiné aplikace Microsoft Office bez záměru ji dále ovládat je možné pomocí metody *ActivateMicrosoftApp*. Tento čtvrtý prostředek pro spouštění aplikací je součástí objektu *Application*. Pokud je již program spuštěn, tak proběhne pouze jeho aktivace. Jako parametr tato metoda přijímá jednu z těchto konstant označujících samotné aplikace:

- *xlMicrosoftWord*
- *xlMicrosoftPowerPoint*
- *xlMicrosoftMail*
- *xlMicrosoftAccess*
- *xlMicrosoftFoxPro*
- *xlMicrosoftProject*
- *xlMicrosoftSchedulePlus*

Technologie Automation, jinak označovaná jako OLE, slouží k přistupování k jiným aplikacím a práci s nimi. Tuto technologii lze využít u těch programů, které vystavují své objekty Automation, tj. např. u aplikací Microsoft Office. Před použitím externí aplikace v kódu je nezbytné vytvořit její instanci. Vytvoření instance lze provést dvěma způsoby:

- Časná vazba
- Pozdní vazba

„Slovo vazba (binding) v tomto případě vyjadřuje vazbu mezi voláním funkcí a skutečným kódem, který tuto funkci implementuje.“ Při použití časné vazby je nejdříve potřeba v položce References v editoru vybrat potřebnou knihovnu a poté je možné vytvořit instanci objektu. Mezi výhody tohoto způsobu patří podle Walkenbacha lepší výkon při samotném vytváření objektů, možnost používat konstanty definované v externí aplikaci a pohodlnější vývoj samotné aplikace, jelikož má vývojář k dispozici prohlížeč objektů a editor mu dokáže doplňovat názvy členů externího objektu.

```
Dim AplikaceWord As New Word.Application
```

*Kód 24 - Vytvoření instance aplikace Word při použití časné vazby (zdroj: [2])*

Použití pozdní vazby spočívá v deklarování proměnné s datovým typem *Object* a následným přiřazením instance objektu pomocí funkce *CreateObject* nebo *GetObject*. Zatímco funkce *CreateObject* vrací odkaz na nově vytvořenou instanci objektu, funkce *GetObject* vrací odkaz na instanci již spuštěné aplikace. Při vytvoření nové instance s využitím pozdní vazby je použita vždy nejnovější verze konkrétní aplikace, ale je i možné specifikovat požadovanou verzi.

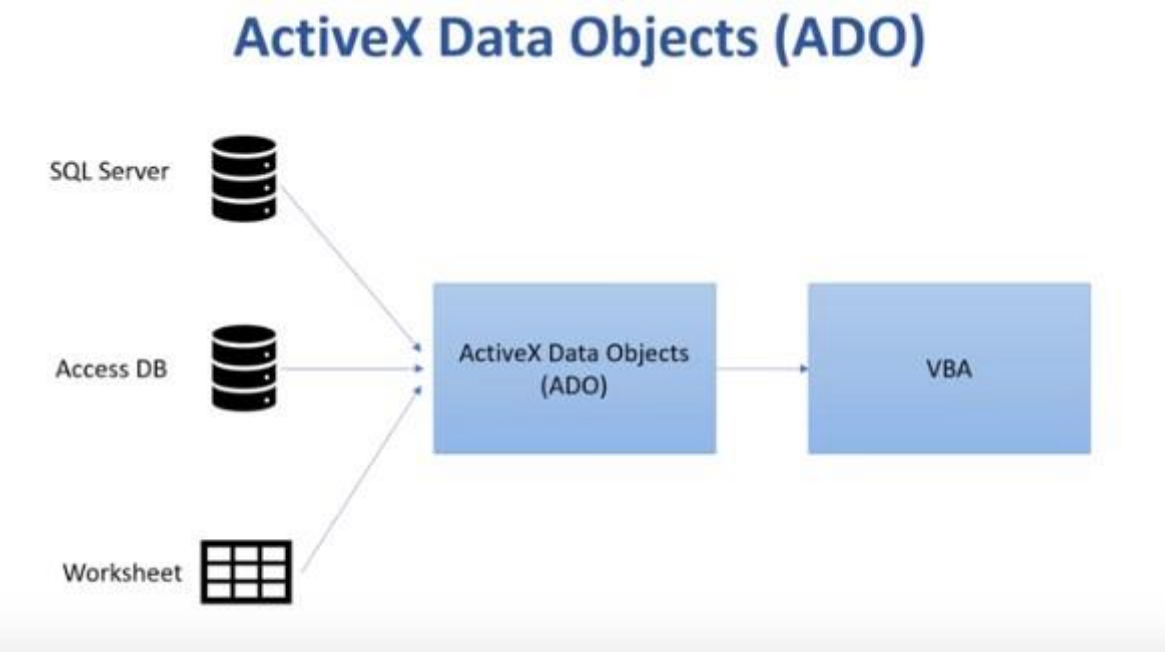
```
Dim AplikaceWord As Object
Set AplikaceWord = CreateObject("Word.Application")
```

*Kód 25 - Vytvoření instance aplikace Word při použití pozdní vazby (zdroj: [2])*

Aplikace, které nepodporují Automation, je možné ovládat pomocí příkazu *SendKeys*, který dokáže simulovat stisky kláves prováděné uživatelem. Tento příkaz ale nedokáže rozpoznat verzi spouštěného programu, tudíž zde může nastat problém s dlouhodobou funkčností kódu vzhledem k aktualizacím těchto programů.

#### 3.6.7.1 **Microsoft Access**

Přístup k datům uchovaným v aplikaci Access je umožněn prostřednictvím objektového modelu ADO (ActiveX Data Objects). Tento objektový model umožňuje také přístup k datům různých databázových formátů. Oproti staršímu modelu DAO (Data Access Objects) je ADO upřednostňovaná varianta pro komunikaci s databázemi. [2]




*Obrázek 3 – Schéma komunikace objektového modelu ADO (zdroj: https://www.p2w2.com/blog/accessing-multiple-data-sources-using-ado-and-vba-in-excel/)*

### 3.6.7.2 Microsoft Outlook

Mezi aplikace podporující technologii Automation patří také Microsoft Outlook. Aplikace disponuje objektovým modelem, díky kterému je možné manipulovat s daty ve složkách a uživatelským prostředím aplikace. Pro přístup k složkám s daty je nutné vytvořit odkaz na novou instanci objektu *NameSpace*, který odkazuje na rozhraní MAPI. MAPI (Messaging API) je sada rozhraní určená k přistupování k různým poštovním aplikacím a systémům. [30]

```
Set objOL = New Outlook.Application
Set objNS = objOL.GetNameSpace("MAPI")
Set objFolder = objNS.GetDefaultFolder(olFolderContacts)
```

*Kód 26 - Přístup ke složce s kontakty aplikace Outlook (zdroj: [31])*

Pro vytvoření základních typů položek je možné použít metodu *CreateItem*:

```
Set NewTask = objOLApp.CreateItem(olTaskItem)
```

*Kód 27 - Vytvoření úkolu v aplikaci Outlook (zdroj: [31])*

### 3.6.7.3 Microsoft Word

Ve VBA lze použít objekt *Word.Application*. Podobně jako u Microsoft Excel, kde objekt *Application* má jako vlastnost kolekci *Workbooks*, má Word kolekci *Documents* reprezentující otevřené dokumenty. Založit nový dokument je možné pomocí metody *Add*. Metoda *Add* dále může přijímat parametr s cestou k jinému dokumentu, který má být

otevřen. Zápis do dokumentu se provádí pomocí metody *TypeText* nebo *TypeParagraph*. [27]

```
app.Selection.TypeText Range("B12")
```

*Kód 28 - Zápis hodnoty v buňce B12 do dokumentu Word (zdroj: [27])*

### 3.6.8 Ovládací prvky v listu

Do listů aplikace Excel lze vkládat různé ovládací prvky, které mohou sloužit k provádění různých operací. Mezi tyto prvky se řadí např. Tlačítko, Zaškrtávací pole, Rozevírací pole, Přepínač a různé seznamy. [10]

Možnost přidávat ovládací prvky do listu přišla s vydáním verze Excel 5.0 v roce 1993. Důsledkem toho bylo získání dominantní pozice na trhu na úkor produktů společností Lotus a Borland. Tato skupina prvků se nazývá Form Controls. I když je Excelu možné použít i novější prvky skupiny ActiveX, prvky Form Controls mohou stále najít své uplatnění díky zpětné kompatibilitě a jednoduššímu používání. Oproti prvkům ActiveX jsou Form Controls schopny reagovat pouze na jednu událost (např. kliknutí), mají menší počet vlastností a je možné je použít v listu s grafem. Proceduru, která se má spustit při interakci s ovládacím prvkem, lze vybrat z rozevíracího seznamu po vytvoření prvku. Prvky lze také přidávat do listu i programově: [10]

```
Dim lb As ListBox
Dim rg As Range
Set rg = Range("E2:F4")
With rg
Set lb = .Parent.ListBoxes.Add _
(.Left, .Top, .Width, .Height)
End With
```

*Kód 29 - Přidání prvku ListBox do listu (zdroj: [10])*

Prvky ActiveX mohou oproti svému předchůdci reagovat na více událostí a mají také více vlastností, díky kterým lze prvek modifikovat. Přiřazení makra je nutné provést ručně v kódu. [10] Všechny tyto prvky jsou objekty typu *OLEObject* a jsou součástí kolekce *OLEObjects*. [32]

```
ActiveSheet.OLEObjects.Add "Forms.CommandButton.1", _
Left:=10, Top:=10, Height:=20, Width:=100
```

*Kód 30 - Přidání tlačítka ActiveX do listu (zdroj: [32])*

### 3.6.9 Uživatelské formuláře

Kromě předdefinovaných formulářů (např. dialogy Otevřít nebo Uložit jako) lze ve VBA vytvářet také vlastní uživatelské formuláře. S vydáním verze Excel 97 tyto formuláře nahradily listy typu dialog. [2]

#### 3.6.9.1 **Ovládací prvky**

K použití ve formulářích jsou k dispozici tyto základní prvky:

- Zaškrtávací políčko (*CheckBox*)
- Pole se seznamem (*ComboBox*) – rozevírací seznam
- Příkazové tlačítko (*CommandButton*)
- Rámeček (*Frame*) – pro seskupení více prvků
- Obrázek (*Image*)
- Popisek (*Label*)
- Seznam (*ListBox*)
- Vícenásobná stránka (*MultiPage*) – pro vytvoření více záložek
- Přepínač (*OptionButton*) – výběr jedné z několika možností
- *RefEdit* – výběr oblasti buněk v listu
- Posuvník (*ScrollBar*) – posuvník pro výběr hodnoty z intervalu možných hodnot
- Číselník (*SpinButton*) – přepínání hodnot šipkami o 1
- Karty (*TabStrip*) – alternativa MultiPage
- Textové pole (*TextBox*)
- Přepínací tlačítko (*ToggleButton*) – pro výběr hodnoty pravda/nepravda

[2]

Do panelu Toolbox lze ještě přidat další specifické prvky, jako např. Windows Media Player nebo Adobe PDF Reader. Toto lze provést výběrem prvku ze seznamu Additional Controls přístupném ze záložky Tools. [33]

### 3.6.10 **Ochrana projektu**

Jelikož je editor s kódem součástí souboru aplikace Microsoft Excel, tak v případě distribuce souboru ostatním uživatelům je vhodné zamezit zobrazení či úpravě tohoto kódu. Excel nabízí zabezpečení kódu heslem v položce *VBAProject Properties* v editoru kódu. Zde na kartě *Protection* lze označit volbu *Lock project for viewing* a zadat heslo. [27]

# 4 Vlastní práce

Tato část práce se věnuje popisu tvorby ukázkové aplikace vyvíjené ve VBA. Při tvorbě bylo využito teoretických východisek obsažených v této práci. Hlavními požadavky pro aplikaci jsou: validace vstupních dat, zabezpečení před nesprávnou manipulací a celkové zefektivnění práce s evidencí.

Data jsou uložena v databázovém souboru aplikace Microsoft Access a veškerá činnost se provádí v programu Microsoft Excel. Program byl vytvořen a otestován na lokálním počítači s operačním systémem Windows 10 v balíčku programů Microsoft Office verze 365.

## 4.1 Charakteristika agendy

V podniku se vedou evidence pronajímaných pracovních plošin a jejich revizí. Tyto evidence jsou uchovávány ve dvou souborech aplikace Excel a nejsou tedy nijak propojeny. Navíc záznamy obsahují duplicitní údaje a nejsou nijak chráněny před nepovolenou úpravou. Při nabízení plošin zákazníkům je vždy nutné zjistit, jestli není plošina vypůjčena nebo jestli nemá naplánovanou revizi. Tento proces je prováděn manuálně a kvůli množství záznamů může být časově náročný.

Je nutné pracovníkům podniku umožnit manipulovat se záznamy plošin, záznamy jejich výpůjček a záznamy zákazníků. Tyto entity by od sebe měly být důsledně odděleny. Dále je potřeba exportovat tyto data dalším složkám společnosti v různých formátech. Aplikaci mohou využít jak pracovníci, jejichž agendou je evidence kontraktů se zákazníky, tak i pracovníci provádějící údržby plošin. V případě zadání chybného údaje je nutné umožnit uživateli tento údaj změnit anebo odstranit celý záznam, aniž by došlo k porušení integrity dat.

### 4.1.1 Výpůjčky

V aplikaci se zaznamenávají tyto údaje o výpůjčkách:

- Plošina, která byla vypůjčena
- Zákazník, kterému byla plošina vypůjčena
- Období, ve kterém byla plošina vypůjčena
- Cena vypůjčení
- Poznámky k výpůjčce

### 4.1.2 **Plošiny**

Záznamy plošin jsou vázány na záznamy výpůjček. Do evidence se zaznamenává:

- Sériové číslo plošiny
- Typ plošiny
- Pohon
- Zdvih v metrech
- Datum revize
- Interval revize ve dnech
- Poznámky
- Vyřazení plošiny

Údaje o typu, pohonu a zdvihu plošiny využijí především pracovníci, kteří sjednávají kontrakty se zákazníky a ostatní údaje využijí pracovníci, kteří vykonávají údržby plošin.

### 4.1.3 **Zákazníci**

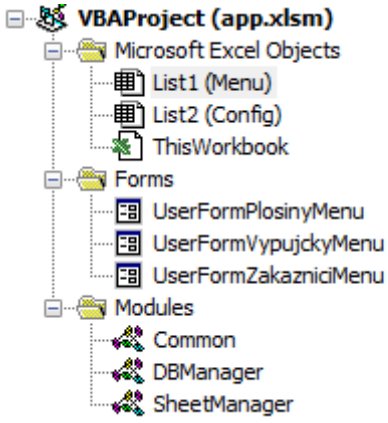
Záznamy o zákaznících jsou také vázány na záznamy výpůjček. Údaje, které je nutné zaznamenávat, jsou:

- IČO
- Název společnosti
- Kontakt na zákazníka ve formě telefonu a emailové adresy

## 4.2 **Struktura projektu**

### 4.2.1 **Klient**

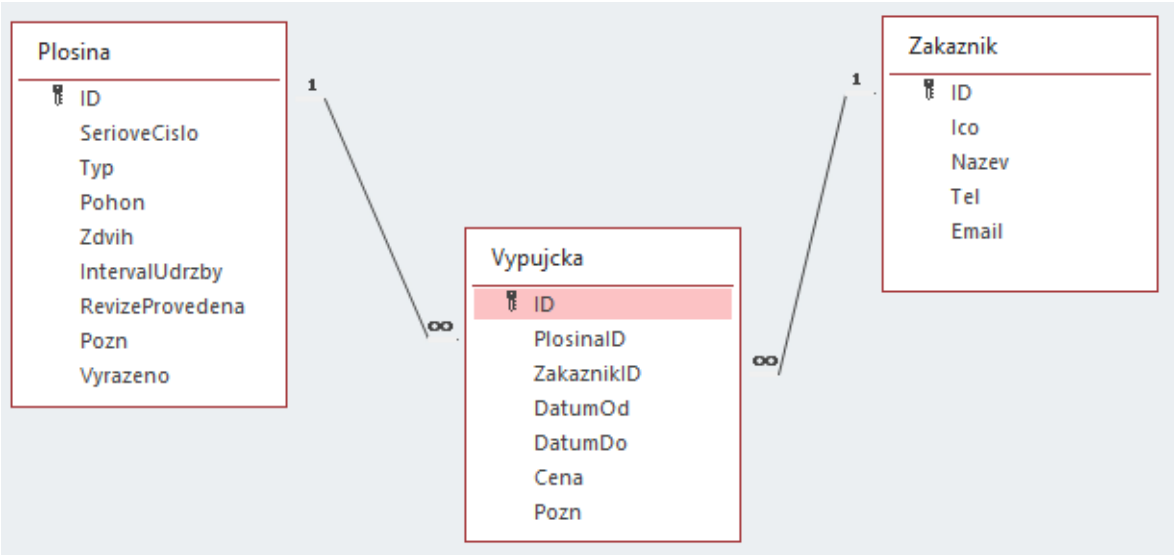
Aplikace má dvě části: klient a server. Jako klient zde vystupuje aplikace Microsoft Excel, ve které jsou všechna makra automatizující práci s daty. Tato aplikace je distribuována uživatelům a jejím prostřednictvím se zajišťuje přístup k datům. Kód aplikace je uspořádán do několika samostatných celků podle druhu činnosti. Moduly *Common*, *DBManager* a *SheetManager* obsahují procedury s obecnější funkcí. Ve stálých listech (Menu) jsou procedury reagující pouze na ovládací prvky, které se v nich nacházejí. Ve formulářích jsou umístěny procedury zajišťující různou funkcionalitu prvků a události reagující na uživatelovu činnost. V projektu je ještě soubor s kódem *ThisWorkbook* reprezentující samotný sešit, který obsahuje události spouštěné při jeho otevření a zavření.




*Obrázek 4 - Struktura projektu (zdroj: vlastní)*

### 4.2.2 Server

Data jsou uložena v databázovém souboru aplikace Microsoft Access, který je umístěn na sdíleném disku přístupném pracovníkům, jejichž agenda se týká evidence výše uvedených dat. V případě absence sdíleného disku nebo používání aplikace pouze jednou osobou je možné datový soubor také uchovávat na lokálním počítači. Tento soubor je možné chránit heslem před zobrazením či manipulací s daty a je k němu přístup pouze prostřednictvím vytvořené aplikace v Microsoft Excel. K záznamům v databázi aplikace přistupuje pomocí objektového modelu ADO a SQL dotazů.




*Obrázek 5 - Struktura databáze (zdroj: vlastní)*

#### 4.2.2.1 Tabulky

Tabulka *Plosina*, ve které jsou uchovávány záznamy o vlastněných plošinách, obsahuje číselný údaj *ID* jako primární klíč, díky kterému je každý záznam jednoznačně identifikován a díky kterému lze nastavit vazbu na záznamy v tabulce *Vypujcka*. Tento údaj je nastaven

automaticky při vytvoření nového záznamu. Dále obsahuje povinný unikátní číselný údaj *SerioveCislo*. Tento údaj je u každého záznamu také povinný a unikátní. Do databáze se ale ukládá jako text kvůli možnosti, kdy začíná nulou. Dále obsahuje číselné údaje *Zdvih* a *IntervalUdrzby*, *Vyrazeno* typu Ano/Ne, *RevizeProvedena* typu datum a textové údaje *Typ*, *Pohon* a *Pozn*. Kromě údaje *Pozn* jsou všechny údaje povinné. Údaj *Vyrazeno* má po nepřiřazení hodnoty automaticky hodnotu Ne.

Pro uchování záznamů o zákaznících je určena tabulka *Zakaznici*. Záznam v této tabulce obsahuje, stejně jako u tabulky *Plosina*, primární klíč *ID*. Dále obsahuje jedinečný číselný údaj *ICO* (uchovávaný v databázi jako text ze stejného důvodu, jako u údaje *SerioveCislo*), číslo *Tel* a textové údaje *Nazev* a *Email*. U této tabulky při zadávání záznamu všechny údaje povinné.

Poslední tabulkou v databázi je *Vypujcka*. Ta obsahuje, obdobně jako u předchozích tabulek, identifikátor ID. Pro propojení s předchozími tabulkami obsahuje cizí klíče *PlosinaID* a *ZakaznikID*. Údaje *DatumOd* a *DatumDo* jsou typu datum, *Cena* číslo a *Pozn* text. V této tabulce jsou všechny údaje kromě *Pozn* povinné.

#### 4.2.2.2 **Vazby mezi tabulkami**

Jak již bylo zmíněno, tabulka *Vypujcka* obsahuje cizí klíče *PlosinaID* a *ZakaznikID*, díky kterým je možné propojit záznamy v tabulkách *Plosina* a *Zakaznik*. U těchto vazeb je nastavena referenční integrita. Ta zajišťuje nemožnost odstranit záznamy z tabulek *Plosina* a *Zakaznik*, jestliže v tabulce *Vypujcka* existuje záznam s jejich identifikátory. Jinak řečeno, nelze odstranit záznam plošiny, která již byla někdy vypůjčena a zároveň nelze odstranit záznam zákazníka, kterému byla plošina někdy vypůjčena. Pro provedení těchto operací je nutné nejdříve odstranit konkrétní záznam z tabulky *Vypujcka*.

## 4.3 Připojení k databázi

Připojení k souboru s daty je realizováno pomocí objektu *ADODB*. Tento objekt je používán ve všech procedurách a funkcích v modulu *DBManager* (tj. *TestConnection*, *PullData* a *ExecuteQuery*). V tomto modulu se nachází proměnná *connString*, jejíž hodnota je tzn. *Connection string*. Hodnota této proměnné specifikuje zprostředkovatele připojení, cestu k souboru a další údaje.

```
connString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" _
        & dbPath & ";Persist Security Info=False;"
```

*Kód 31 - Connection string v proceduře TestConnection (zdroj: vlastní)*

V případě ochrany databáze heslem lze do connection string připojit i přístupové heslo: *„Jet OLEDB:Database Password=MyDbPassword;“*. [34]

Poté je nutné vytvořit novou instanci objektu *ADODB.Connection* a otevřít připojení metodou *Open*, která přijímá tzn. *Connection string*. Po vykonání operací s databází je nutné připojení zavřít metodou *Close*.

```
Set cn = New ADODB.Connection
cn.Open connString
'Operace s databází
cn.Close
```

*Kód 32 - Připojení k databázi pomocí objektu ADODB (zdroj: vlastní)*

### 4.3.1 PullData

Procedura *PullData* slouží k výběru dat z databáze pomocí SQL dotazu a importování jej do nového listu. Nejprve se otestuje, zda je možné se k databázi připojit pomocí procedury *TestConnection*. Poté se vytvoří nový prázdný list, otevře připojení a *Recordset*, v cyklu se zkopíruje hlavička tabulky a výsledek dotazu pomocí metodoy *CopyFromRecordset*. Následně se připojení uzavře a uvolní místo v paměti, které zabíraly proměnné *Connection* a *Recordset*. V případě havárie se taktéž uvolní místo v paměti a funkce se ukončí.

```
Public Sub PullData(query As String, sheetName As String)
    On Error GoTo errhandler
    Application.DisplayAlerts = False
    Application.ScreenUpdating = False
    TestConnection False
    SheetManager.ResetSheet sheetName
    Set cn = New ADODB.Connection
    Set rs = New ADODB.Recordset
    cn.Open connString
    rs.Open query, cn
    Dim col As Long
    With ThisWorkbook.Worksheets(sheetName)
        For col = 0 To rs.Fields.Count - 1
            .Cells(.Rows.End(xlUp).row, col + 1).Value =
rs.Fields(col).name
        Next col
        .Range("A2").CopyFromRecordset rs
    End With
    cn.Close
    Set rs = Nothing
    Set cn = Nothing
    Application.DisplayAlerts = True
    Application.ScreenUpdating = True
    Exit Sub
errhandler:
    cn.Close
    Set cn = Nothing
    Set rs = Nothing
    SheetManager.DeleteSheet sheetName
    Application.DisplayAlerts = True
    Application.ScreenUpdating = True
```

```
    Common.ShowErrorMessage
End Sub
```

*Kód 33 – Kód funkce PullData (zdroj: vlastní)*

### 4.3.2 ExecuteQuery

Kód procedury *ExecuteQuery* je o poznání jednodušší než u *PullData*. Nejprve se, stejně jako u *PullData*, připojení otestuje a následně otevře. Dalším příkazem je vykonání dotazu metodou *Execute*. Nakonec se připojení uzavře a uvolní místo v paměti po proměnné *Connection*.

```
Public Sub ExecuteQuery(query As String)
    TestConnection False
    On Error GoTo errhandler
    Set cn = New ADODB.Connection
    cn.Open connString
    cn.Execute query
    cn.Close
    Set cn = Nothing
    Exit Sub
errhandler:
    cn.Close
    Set cn = Nothing
    Common.ShowErrorMessage
End Sub
```

*Kód 34 - Kód procedury ExecuteQuery (zdroj: vlastní)*

## 4.4 Spuštění aplikace

Nejprve se při samotném otevření programu Microsoft Excel spustí událost *Workbook_Open*. V té se pro uživatele uzamknou sešity Menu a Config. Nastavení vlastnosti *UserInterFaceOnly* umožní manipulovat s těmito sešity pouze prostřednictvím maker. Dále je sešit Config skryt před uživateli nastavením vlastnosti *Visible*. Vestavěná konstanta *XlSheetVeryHidden* znamená skrytí listu, který je možné znovu zobrazit pouze prostřednictvím makra. Následně se zavolá procedura *DeleteAllTempSheets*, která odstraní všechny listy kromě Menu a Config, pokud nějaké existují, a nakonec zavolá proceduru *TestConnection*, která testuje, jestli je v listu Config uvedena platná cesta k souboru s daty. V případě předání argumentu *True*, absencí cesty nebo absencí souboru se uživateli zobrazí vestavěné dialogové okno *GetOpenFilename* pro výběr souboru z adresáře. Pokud uživatel přesto nevybere soubor s databází, zavolá se procedura *CloseApp*, která ukončí aplikaci.

```
Private Sub Workbook_Open()
    With Me.Worksheets("Menu")
        .Protect _
        Password:="heslo", UserInterfaceOnly:=True
        .ScrollArea = "$A$1"
    End With
    With Me.Worksheets("Config")
```

```
        .Protect _
            Password:="heslo", UserInterfaceOnly:=True
        .Visible = xlSheetVeryHidden
    End With
    SheetManager.DeleteAllTempSheets
    dbManager.TestConnection False
End Sub
```

*Kód 35 - Událost Workbook_Open (zdroj: vlastní)*

```
If IsEmpty(dbPath) Or changeConn = True Then
        dbPath = Application.GetOpenFilename( _
            FileFilter:="Access Files (*.accdb),*.accdb", _
            MultiSelect:=False, _
            Title:="Vyberte soubor s databází:")
        If dbPath <> False Then
            wsConfig.Range("B1").Value = dbPath
        End If
End If
```

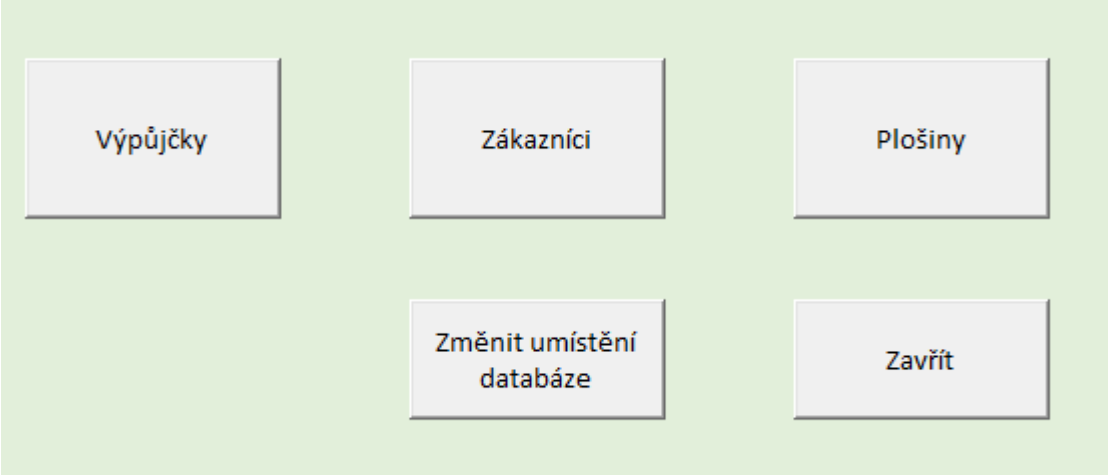
*Kód 36 - Použití dialogového okna GetOpenFilename v proceduře TestConnection (zdroj: vlastní)*

## 4.5 Stálé listy

Sešit obsahuje dva stálé listy, které nikdy nesmí být odstraněny. Tyto listy jsou pojmenovány Menu a Config. Ostatní dočasné listy jsou generovány za běhu programu a slouží pro uložení vybraných záznamů z databáze.

### 4.5.1 Menu

Po spuštění aplikace je uživateli zobrazen list Menu s tlačítky typu ActiveX pro otevření uživatelských formulářů, změnu cesty k souboru s daty a zavření aplikace. Při kliknutí na tlačítko určené pro zobrazení formuláře se zavolá událost, která odstraní všechny dočasné listy a pomocí metody *Show* objektu formuláře daný formulář zobrazí. Metodě *Show* je předáván argument *vbModeless*, který umožňuje práci se sešitem i po otevření formuláře. Při kliknutí na tlačítko pro změnu cesty k databázi se zavolá procedura *TestConnection* s argumentem *True*. Kliknutím na tlačítko Zavřít se zavolá procedura *CloseApp*.




*Obrázek 6 - Ovládací prvky v listu Menu (zdroj: vlastní)*

### 4.5.2 Config

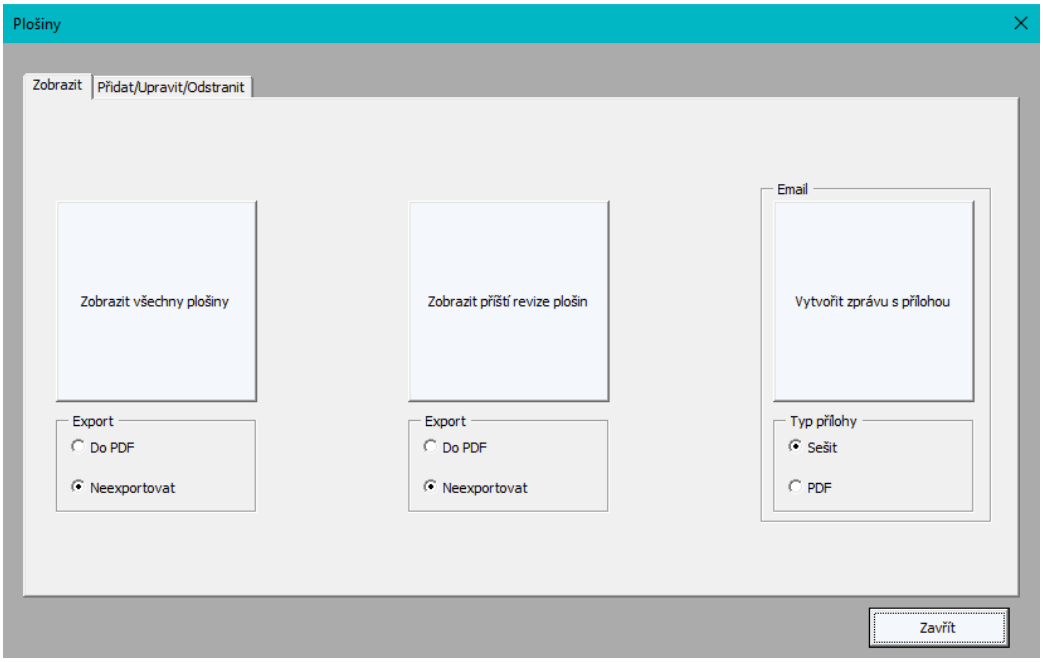
List Config je určen pro ukládání informací nezbytných pro běh programu. Uzamčením, skrytím a ošetřením před odstraněním je chráněn před manuální manipulací. V tomto listu jsou uloženy dvě hodnoty, a to cesta k souboru s daty a seznam listů chráněných před odstraněním oddělený středníky. K první hodnotě přistupuje dříve popsaná procedura *TestConnection*, ke druhé pak soukromá funkce *IsProtectedSheet* v modulu *SheetManager*. Tato funkce má návratovou hodnotu Boolean a vrací hodnoty *True/False* podle toho, jestli je list pojmenovaný parametrem *sheetName* obsažen v seznamu stálých listů. To je ověřováno pomocí vestavěné funkce *InStr*.

```
Private Function IsProtectedSheet(sheetName As String) As Boolean
    Dim protectedSheets As String
    protectedSheets = ThisWorkbook.Worksheets("Config").Range("B2")
    If InStr(protectedSheets, sheetName) = 0 Then
        IsProtectedSheet = False
    Else
        IsProtectedSheet = True
    End If
End Function
```

*Kód 37 - Ověření listů před odstraněním (zdroj: vlastní)*

## 4.6 Uživatelské formuláře

Kliknutím na tlačítka v listu Menu se uživateli zobrazí uživatelské formuláře pro přístup k záznamům v tabulkách. Pro každou tabulku byl vytvořen samostatný formulář. Tyto formuláře mají mnoho prvků shodných, tj. záložky Zobrazit, Přidat/Upravit/Odstranit a tlačítka pro zobrazení či export záznamů. Liší se pouze počtem, typem a chováním vstupních polí ve druhé záložce.




*Obrázek 7 - Záložka Zobrazit formuláře UserFormPlosinyMenu (zdroj: vlastní)*

Všechny tyto formuláře obsahují tlačítko pro zobrazení všech záznamů a vygenerování emailové zprávy s přílohou v podobě seznamu všech záznamů. Formulář Plošiny obsahuje navíc tlačítko pro zobrazení příštích revizí vypočtených podle data revize a intervalu revizí.

V záložce Přidat/Upravit/Odstranit mají všechny formuláře rozevírací seznam pro volbu konkrétního záznamu k úpravě. V těchto seznamech jsou na k výběru tyto údaje:

- Zákazníci: IČO
- Plošiny: Sériové číslo
- Výpůjčky: ID

Tyto seznamy mají, stejně jako všechny seznamy v tomto projektu, nastavenu hodnotu *frmDropDownList* u vlastnosti *Style*, což zajišťuje nemožnost do něj zadat vlastní text. Na počátku každého spuštění jsou z databáze vybrány identifikační údaje a těmi je poté seznam naplněn. K tomu může být ještě přidána položka „Nový“, která slouží pro přidání nového záznamu.

```
Public Sub FillCombobox(query As String, cb As combobox, numberOfCols As
Integer, addOptionNew As Boolean)
    cb.Clear
    cb.ColumnCount = numberOfCols
    dbManager.PullData query, "Temp"
    With ThisWorkbook.Worksheets("Temp")
        Dim lastRow As Long: lastRow = .Cells(.Rows.Count,
1).End(xlUp).row
        If lastRow = 1 Then
            SheetManager.DeleteSheet "Temp"
            Exit Sub
```

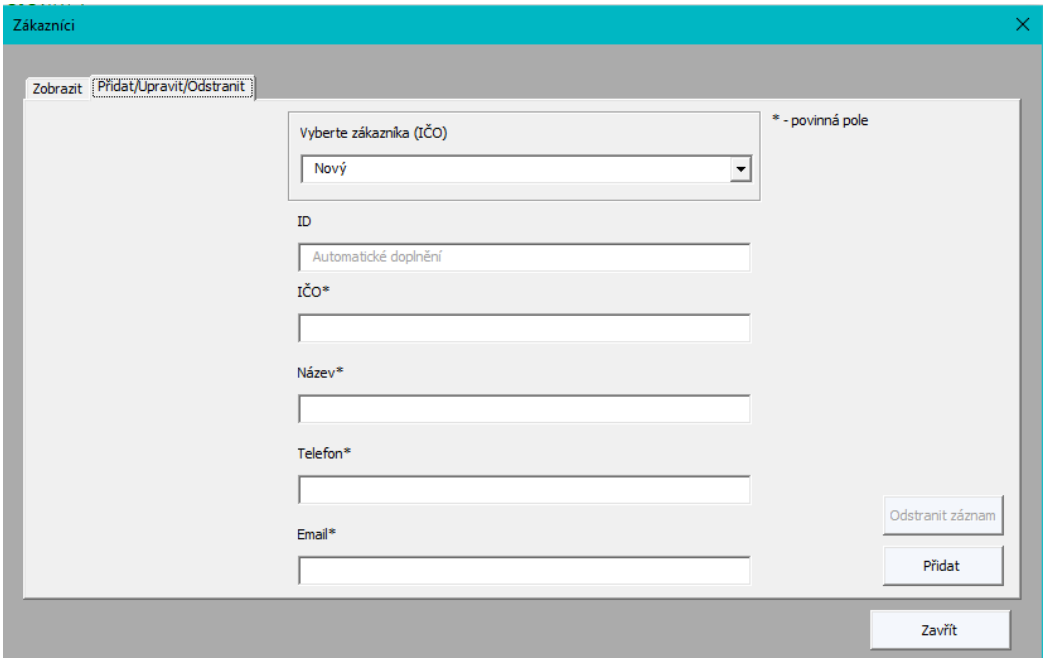
```
        End If
        Dim i As Long: i = 2
        Dim j As Integer: j = 2
        Dim N As Long
        If addOptionNew Then
            N = lastRow - 1
            ReDim CbRows(N, numberOfCols - 1) As Variant
            CbRows(0, 0) = "Nový"
            For i = 2 To lastRow
                For j = 0 To numberOfCols - 1
                CbRows(i - 1, j) = .Cells(i, j + 1)
                Next j
            Next i
            cb.List = CbRows
        Else
            N = lastRow - 2
            ReDim CbRows(N, numberOfCols - 1) As Variant
            For i = 2 To lastRow
                For j = 0 To numberOfCols - 1
                CbRows(i - 2, j) = .Cells(i, j + 1)
                Next j
            Next i
            cb.List = CbRows
        End If
    End With
    SheetManager.DeleteSheet "Temp"
End Sub
```

*Kód 38 - Univerzální procedura FillCombobox pro vyplnění seznamu položkami (zdroj: vlastní)*

Záložka Přidat/Upravit/Odstranit dále obsahuje vstupní pole podle počtu údajů a tlačítka Odstranit a Přidat/Upravit.

### 4.6.1 **Zákazníci**

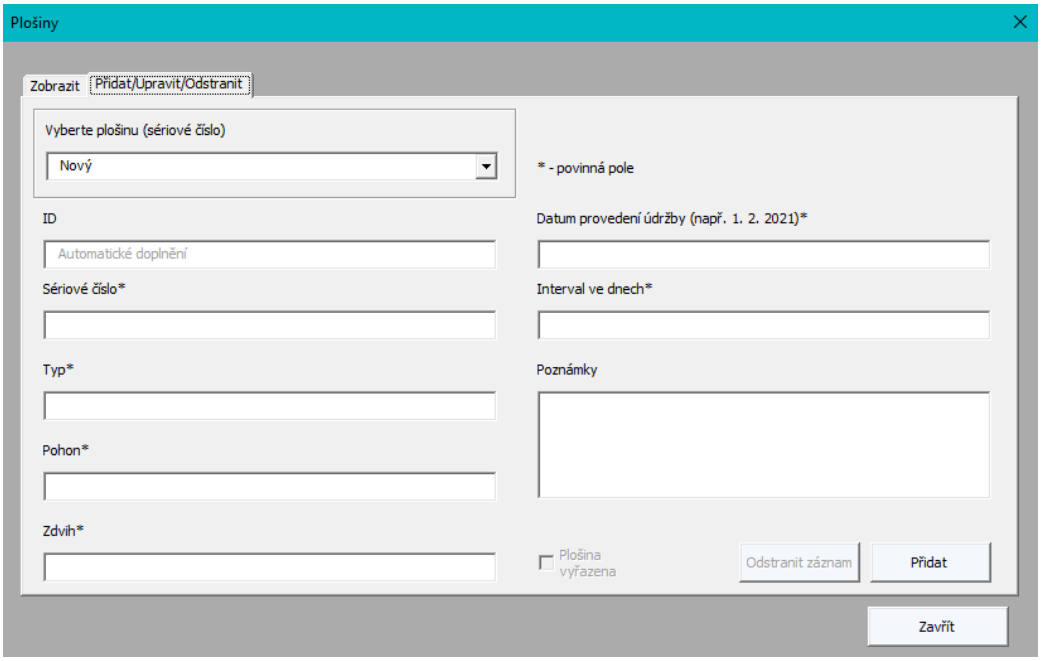
Formulář Zákazníci obsahuje nejméně vstupních polí ze všech formulářů. Kromě již dříve zmíněného seznamu s IČO obsahuje vstupní pole (*TextBox*) pro všechny údaje v záznamech. Pole *ID* je chráněno před úpravou a slouží pouze pro identifikaci záznamu, který má být poté upraven či odstraněn.




*Obrázek 8 - Vstupní pole formuláře UserFormZakazniciMenu (zdroj: vlastní)*

### 4.6.2 **Plošiny**

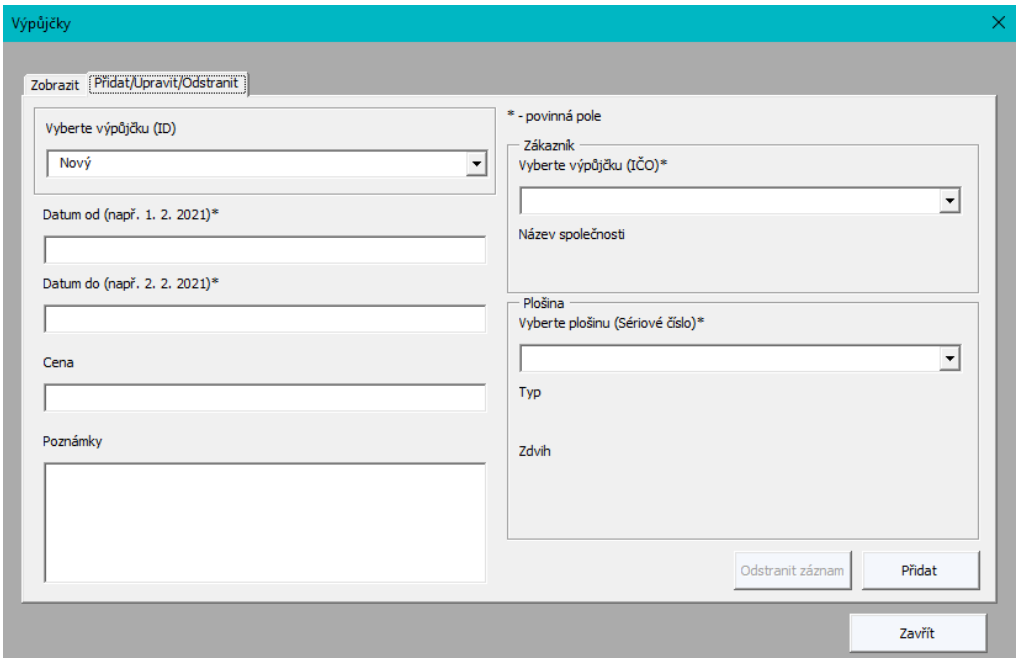
Záložka Přidat/Upravit/Odstranit u formuláře Plošiny obsahuje seznam se sériovými čísly pro výběr záznamu plošiny k úpravě či odstranění. Po výběru záznamu jsou jeho údaje vyplněny do textových polí, výjimku tvoří údaj o vyřazení plošiny, jehož hodnota je naplněna do zaškrtávacího pole (*CheckBox*).




*Obrázek 9 - Vstupní pole formuláře UserFormPlosinyMenu (zdroj: vlastní)*

### 4.6.3 Výpůjčky

Formulář Výpůjčky obsahuje kromě textových polí pro údaje Datum od, Datum do, Cena a Poznámky také rozevírací seznamy pro zobrazení zákazníka a plošiny. Výběr záznamu k úpravě se provádí pomocí seznamu s ID záznamy výpůjček. Rozevírací seznam pro výběr zákazníka zobrazuje údaje IČO a Název a rozevírací seznam pro výběr plošiny zobrazuje údaje Sériové číslo, Typ a zdvih. Tyto údaje se poté vypíšou pod příslušné seznamy. Seznam pro výběr plošiny je zpřístupněn až po vyplnění data začátku a konce vypůjčení z důvodu kontroly obsazenosti plošin.




*Obrázek 10 - Vstupní pole formuláře UserFormVypujckyMenu (zdroj: vlastní)*

## 4.7 Výpis záznamů

Výpis záznamů probíhá u všech záznamů stejným způsobem. Proces začíná kliknutím na tlačítko Zobrazit vše v záložce Zobrazit. Toto zachycuje událost *CommandButtonShowAll_Click*. Nejdříve se zavře formulář příkazem *Unload Me*. Poté se pomocí již dříve popsané procedury *PullData*, které je předán SQL dotaz a název listu, zkopírují záznamy do nově vygenerovaného dočasného listu. Poté je pomocí procedury *StyleSheet*, která přijímá název listu, vygenerovaná tabulka formátována pro dosažení přívětivějšího vzhledu. Procedura *StyleSheet* je univerzální pro všechny generované výpisy v aplikaci, tabulky se v ní formátují jednotlivě pomocí konstrukce *Select Case*.

```
Dim query As String
query = "SELECT Vypujcka.ID, Plosina.SerioveCislo, Plosina.Typ," _
    & "Zakaznik.ICO, Zakaznik.Nazev, Vypujcka.DatumOd," _
    & "Vypujcka.DatumDo, Vypujcka.Cena, Vypujcka.Pozn " _
```

```
    & "FROM Plosina, Vypujcka, Zakaznik " _
    & "WHERE Vypujcka.PlosinaID = Plosina.ID " _
    & "AND Vypujcka.ZakaznikID = Zakaznik.ID"
dbManager.PullData query, "Výpůjčky"
SheetManager.StyleSheet "Výpůjčky"
```

*Kód 39 - Část události CommandButtonShowAll_Click pro výpis výpůjček (zdroj: vlastní)*

| ID | Sériové číslo plošiny | Typ plošiny | IČO | Název společnosti | Vypůjčeno od | Vypůjčeno do | Cena vypůjčení | Poznámky |
|---|---|---|---|---|---|---|---|---|
| 2 | 4532187551 | EG2 | 45643645 | Beton:stavby s. r. o. | 01.02.2021 | 01.03.2021 | 15 600 | |
| 15 | 2435484512 | R5412 | 05687464 | ZZ a. s. | 15.03.2022 | 22.03.2022 | 9 200 | |
| 18 | 0254312158 | U20 | 54645647 | Mont-m s. r. o. | 03.03.2021 | 08.03.2021 | 4 300 | |

*Obrázek 11 - Vygenerovaná tabulka se záznamy výpůjček (zdroj: vlastní)*

## 4.8 Přidání záznamu a aktualizace záznamu

Přidání nového záznamu probíhá po vyplnění všech povinných údajů a kliknutí na tlačítko Přidat. Jako první se nastaví popis tohoto tlačítka na Přidat nebo Upravit podle toho, jestli je ve vyhledávacím seznamu hodnota „Nový“, nebo identifikátor záznamu. Dále se zkontrolují hodnoty těch vstupů, u kterých je tato operace nutná (povinné údaje a údaje typů číslo/datum). To, jestli jsou všechny údaje validní, testuje funkce *IsAnyOfInputInvalid*. Následně je proměnné *query* přiřazen řetězec s SQL dotazem pro přidání nebo aktualizaci záznamu. Pomocí procedury *ExecuteQuery* je nakonec dotaz předán databázi.

```
Private Sub CommandButtonAddUpdate_Click()
    If IsAnyOfInputInvalid Then
        MsgBox "Jeden nebo více vstupů jsou neplatné", _
            vbCritical, "Neplatný vstup"
        Exit Sub
    End If
    Dim query As String
    If Me.ComboBoxSearch.Value = "Nový" Then
        query = "INSERT INTO Zakaznik(Ico, Nazev, Tel, Email) VALUES ("""
_
            & Trim(Me.TextBoxICO.Value) & """, """ _
            & Trim(Me.TextBoxNazev.Value) & """," _
            & Trim(Me.TextBoxTel.Value) & ", """ _
            & Trim(Me.TextBoxEmail.Value) & """)"
    Else
        query = "UPDATE Zakaznik SET " _
            & "Ico=""" & Trim(Me.TextBoxICO.Value) _
            & """, Nazev=""" & Trim(Me.TextBoxNazev.Value) _
            & """, Tel=" & Trim(Me.TextBoxTel.Value) _
            & ", Email=""" & Trim(Me.TextBoxEmail.Value) _
            & """ WHERE ID=" & Me.TextBoxID
    End If
    dbManager.ExecuteQuery query
    Unload Me
End Sub
```

*Kód 40 - Událost pro přidání nebo aktualizaci záznamu zákazníka (zdroj: vlastní)*

O něco složitější je tento proces u záznamů výpůjček. Při otevření formuláře se vyplní rozevírací seznam Zákazník údaji IČO a Název. Tuto operaci provede dříve vytvořená

procedura *FillCombobox*. Ta naplní údaji *ICO* a *Nazev* dvourozměrné pole a pomocí metody vlastnosti List naplní daný seznam. Při výběru IČO zákazníka ze seznamu se nakonec vypíšou ostatní údaje do popisků pod ním.

Seznam plošin se naplní a aktualizuje společně po zadání validních údajů *DatumOd* a *DatumDo*. Do tohoto seznamu se vyplní pouze ty plošiny, které nejsou v zadaném rozsahu dat obsazeny a nejsou již vyřazeny.

```
Private Sub RefreshComboboxSearchPlatform()
    Dim query As String
    query = _
    "SELECT SerioveCislo, Typ, Zdvih" _
    & " From Plosina" _
    & " WHERE Vyrazeno=False" _
    & " AND ID NOT IN" _
    & " (" _
    & " SELECT DISTINCT PlosinaID FROM Vypujcka" _
    & " WHERE" _
    & " DateDiff(""d"", """ & Trim(Me.TextBoxDatumOd.Value) _
    & """, DatumDo)>0" _
    & " AND" _
    & " DateDiff(""d"", """ & Trim(Me.TextBoxDatumDo.Value) _
    & """, DatumOd)<0" _
    & ")"
    Common.FillCombobox query, Me.ComboBoxSearchPlatform, 3, False
End Sub
```

*Kód 41 - Procedura pro výběr volných plošin v zadaném intervalu (zdroj: vlastní)*

Jestliže má plošina naplánovanou revizi v zadaném rozsahu dat, vypíše tuto informaci do popisku pod rozevíracím seznamem.

```
lblCheckCaption = Common.GetCheckDate(.Value)
If lblCheckCaption <> vbNullString Then
    If DateDiff("d", Me.TextBoxDatumOd.Value, lblCheckCaption) >= 0 And _
    DateDiff("d", Me.TextBoxDatumDo.Value, lblCheckCaption) <= 0 _
    Then
        Me.LabelRevize.Caption = _
            "Pro tuto plošinu je naplánována revize " & _
        lblCheckCaption
    End If
End If
```

*Kód 42 - Část události ComboBoxSearchPlatform_Change pro testování naplánovaných revizí v rozsahu dat (zdroj: vlastní)*

Nakonec jsou po výběru plošiny, podobně jako u seznamu zákazníků, vypsány údaje *Typ* a *Zdvih* do popisků pod seznamem.

Při aktualizaci záznamu výpůjčky nelze měnit datum vypůjčení a vybranou plošinu. Pro změnu těchto údajů je nutné odstranit daný záznam a přidat nový.

## 4.9 Odstranění záznamu

Odstranění záznamu se provádí pomocí tlačítka „Odstranit záznam“, který je přístupný, když je vybrán záznam pomocí volby v seznamu s identifikátorem záznamu. Po kliknutí se zobrazí okno pro potvrzení akce a následně je záznam pomocí procedury *ExecuteQuery* odstraněn.

```
Private Sub CommandButtonDelete_Click()
    If MsgBox("Opravdu chcete odstanit záznam?", _
        vbYesNo + vbQuestion, "Odstranit záznam") = vbYes Then
        dbManager.ExecuteQuery "DELETE FROM Vypujcka WHERE ID=" _
            & Me.ComboBoxSearch.Value
        Unload Me
    End If
End Sub
```

*Kód 43 - Událost pro odstranění záznamu výpůjčky (zdroj: vlastní)*

## 4.10 Export dat

V aplikaci je možné exportovat výstupy do formátu PDF či jako emailovou přílohu. Mezi tyto výstupy patří:

- tabulky všech výpůjček, zákazníků a vlastněných plošin
- tabulka s termíny nadcházejících revizí plošin

Pro export výpisu záznamů slouží zaškrtávací pole (*Option button*) pod tlačítky pro výpis (viz. obrázek 7). Po následném kliknutí na přidružené tlačítko se nejdříve provedou stejné operace jako při výpisu záznamů. Zvolení zaškrtávacího pole pro export poté způsobí, že je volána procedura *ExportToPdf* s parametry *sheetName* a *sendAsEmail*. Ta exportuje list se záznamy do PDF buď do následně zvoleného adresáře, anebo jako přílohu.

```
Private Sub CommandButtonShowAll_Click()
    Unload Me
    dbManager.PullData "SELECT * FROM Zakaznik", "Zákazníci"
    SheetManager.StyleSheet "Zákazníci"

    If Me.OptionButtonExportListPDF.Value = True Then
        SheetManager.ExportToPdf "Zákazníci", False
        SheetManager.DeleteSheet "Zákazníci"
    End If
End Sub
```

*Kód 44 - Událost pro výpis záznamů zákazníků a export do PDF (zdroj: vlastní)*

V případě uložení souboru PDF do adresáře se nejdříve zobrazí vestavěné dialogové okno *GetSaveAsFilename* pro výběr umístění. Poté je list exportován pomocí jeho metody *ExportAsFixedFormat*. Vytvořený soubor poté tvoří tabulka z exportovaného listu a popisek s datem vytvoření, který byl listu dříve nastaven v proceduře *StyleSheet*. Ve proceduře *StyleSheet* byla také nastavena oblast tisku pro správné zobrazení tabulky v PDF souboru.

Seznam výpůjček - Export 17.02.2021

| ID | Sériové číslo plošiny | Typ plošiny | IČO | Název společnosti | Vypůjčeno od | Vypůjčeno do | Cena vypůjčení | Poznámky |
|---|---|---|---|---|---|---|---|---|
| 2 | 4532187551 | EG2 | 45643645 | Beton:stavby s. r. o. | 01.02.2021 | 01.03.2021 | 15 600 | |
| 15 | 2435484512 | R5412 | 05687464 | ZZ a. s. | 15.03.2022 | 22.03.2022 | 9 200 | |
| 18 | 0254312158 | U20 | 54645647 | Mont-m s. r. o. | 03.03.2021 | 08.03.2021 | 4 300 | |

*Obrázek 12 - Vygenerovaný PDF soubor s výpisem výpůjček (zdroj: vlastní)*

Vytvoření emailové zprávy je prováděno kliknutím na příslušné tlačítko v kartě Zobrazit a výběrem formátu. V události reagující na tuto akci je vytvořen výpis záznamů a zavolána procedura *ExportToPDf* (pro přílohu PDF), resp. *SendWs* (pro přílohu ve formě sešitu s listem). Nejprve je otestováno, jestli aplikace Microsoft Outlook je na pracovní stanici nainstalována a je možné ji otevřít. To je provedeno v proceduře *OpenOutlook*. Ta také v případě nespuštěné aplikace Outlook zajistí její spuštění pomocí příkazu *Shell "Outlook"*. Následně je budoucí příloha uložena do stejného adresáře, ve kterém je uložen sešit. Po samotném vytvoření zprávy a připojení přílohy je nakonec soubor z adresáře odstraněn procedurou *Kill*.

```
Public Sub ExportToPdf(sheetName As String, sendAsEmail As Boolean)
    If Not SheetExists(sheetName) Then Exit Sub
    Dim path As Variant

    If Not sendAsEmail Then
        path = Application.GetSaveAsFilename( _
             FileFilter:="PDF files, *.pdf", _
             Title:="Uložit PDF")
        If path = False Then Exit Sub
        ThisWorkbook.Worksheets(sheetName).ExportAsFixedFormat _
            Type:=xlTypePDF, _
            Filename:=path, _
            OpenAfterPublish:=True
    Else
        OpenOutlook
        If olApp Is Nothing Then Exit Sub
        path = ThisWorkbook.path & "\" & sheetName _
            & "_" & Day(Date) & "_" & Month(Date) & "_" & Year(Date) _
            & ".pdf"
        ThisWorkbook.Worksheets(sheetName).ExportAsFixedFormat _
            Type:=xlTypePDF, _
            Filename:=path
        Set olMail = olApp.CreateItem(olMailItem)
        With olMail
            .Attachments.Add path
            .Display
        End With
        Kill path
        Set olMail = Nothing
        Set olApp = Nothing
```

```
    End If
End Sub
```

*Kód 45 – Procedura ExportToPdf (zdroj: vlastní)*

```
Public Sub SendWs(sheetName As String)
    If Not SheetExists(sheetName) Then Exit Sub

    OpenOutlook
    If olApp Is Nothing Then Exit Sub

    Application.DisplayAlerts = False
    Application.ScreenUpdating = False

    Dim wbTemp As Workbook
    Set wbTemp = Application.Workbooks.Add
    ThisWorkbook.Worksheets(sheetName).Copy before:=wbTemp.Worksheets(1)

    wbTemp.SaveAs ThisWorkbook.path & "\" _

        & sheetName & "_" & Day(Date) & "_" _

        & Month(Date) & "_" & Year(Date) & ".xlsx"

    Set olMail = olApp.CreateItem(olMailItem)
    With olMail
        .Attachments.Add wbTemp.FullName
        .Display
    End With

    Dim wbTempPath As String: wbTempPath = wbTemp.FullName
    wbTemp.Close True
    Set wbTemp = Nothing

    Kill wbTempPath
    Application.DisplayAlerts = True
    Application.ScreenUpdating = True
End Sub
```

*Kód 46 - Procedura SendWs pro připojení sešitu k nové zprávě (zdroj: vlastní)*

## 4.11 Zabezpečení

Z důvodu distribuce aplikace uživatelům je vhodné zabezpečit stálé listy před neoprávněnou manipulací. Jejich uzamčení je provedeno při každém otevření souboru v události *Workbook_Open* (viz. kód 35). Současně je také možné znemožnit přístup do editoru kódu pomocí nastavení hesla v záložce *Tools -> VBAProject Properties*. V případě použití ochrany databáze heslem je možné k této databázi přistupovat pomocí úpravy Connection stringu.

# 5 Výsledky a diskuse

Vytvořená aplikace splňuje předem stanovené požadavky. Projekt je tvořen datovou vrstvou a klientem. Klient slouží pouze jako zprostředkovatel přístupu k záznamům. Záznamy v evidenci jsou uloženy na jednom místě, jsou rozděleny do samostatných částí a propojeny mezi sebou vazbami. Jsou automatizovány základní úkony, mezi které patří výpis, přidání, aktualizace a odstranění záznamů, při jejichž větším množství lze dosáhnout rychlejšího provedení těchto úkonů. Automatizováno bylo také exportování dat do různých formátů. Byly implementovány komponenty, které zabraňují zadání neplatných či chybných údajů do evidence. Velkým přínosem aplikace je automatizované ověřování obsazenosti plošin při zadávání nové výpůjčky.

Oddělení datové vrstvy od aplikace umožňuje jednoduché rozšíření funkčnosti aplikace do různých směrů. V případě požadavku by bylo možné evidenci rozšířit další informace, jako jsou např. fakturační údaje. Následně by bylo implementováno hromadné rozesílání faktur zákazníkům.

Také by mohly být vytvořeny komponenty pro vytváření specifických přehledů za dané časové období a jejich vizualizace. Pro vyšší zabezpečení by mohlo být implementováno zaznamenávání změn vykonaných uživateli nebo vymezení přístupu uživatelům jen do některých částí aplikace.

Jelikož je pro komunikaci se serverem využíván objektový model ADO, je možné v budoucnosti jednoduše přesunout datovou vrstvu do jiného databázového systému, jako je např. Microsoft SQL Server.

# 6 Závěr

Cílem práce byla analýza technologií využívaných k vytváření maker automatizujících úkony v aplikacích balíčku Microsoft Office a demonstrace vývoje aplikace pro řešení konkrétní situace. Toto bylo splněno.

Práce je rozdělena na teoretickou a praktickou část. V teoretické práci je shrnut historický vývoj tabulkových procesorů, představeny používané aplikace balíčku Microsoft Office a analyzovány možnosti vývoje maker v jazyce VBA.

V praktické části byl popsán postup při tvorbě aplikace sloužící k vedení záznamů o výpůjčkách pracovních plošin v podniku. Při vývoji bylo využito teoretických východisek nabytých ze zvolených zdrojů informací. Aplikace je tvořena serverovou částí uchovávající data a klientskou částí, která je zprostředkovatelem přístupu k nim. Byl kladen důraz na ověřování zadaných údajů. Dále aplikace využívá prostředků ostatních aplikací balíčku Microsoft Office pro přístup k datům a tvorbu emailových zpráv s přílohou. Navrženou aplikaci je možné dále rozšiřovat různými směry.

Výsledná aplikace využívá konstrukce a objekty jazyka VBA, které je možné využít pro další obdobné projekty.

# 7 Seznam použitých zdrojů


[1] POWER, D. J. A Brief History of Spreadsheets. *DSSResources.COM* [online]. c1995-2020 [cit. 2020-10-30]. Dostupné z: http://dssresources.com/history/sshistory.html. Version 3.6.

[2] WALKENBACH, John. *Microsoft Office Excel 2007: programování ve VBA*. 1. vydání. Brno: Computer Press, 2008. Programování (Computer Press). ISBN 978-80-251-2011-8.

[3] Excel Versions Explained. *The Smart Method* [online]. London: Smart Method Enterprises Ltd [cit. 2020-11-05]. Dostupné z: https://thesmartmethod.com/excel-versions-explained/

[4] MS Access - Overview. *Tutorialspoint* [online]. [cit. 2021-03-04]. Dostupné z: https://www.tutorialspoint.com/ms_access/ms_access_overview.htm

[5] BALLEW, Jolli, Ryan PERIAN, ed. What is Microsoft Outlook?. *Lifewire.com* [online]. [cit. 2020-11-09]. Dostupné z: https://www.lifewire.com/microsoft-outlook-4164620

[6] BALLEW, Jolli. What Is Microsoft Word?. *Lifewire.com* [online]. New York [cit. 2020-11-09]. Dostupné z: https://www.lifewire.com/microsoft-word-4159373

[7] VBA Language Specification Overview. *Microsoft Docs* [online]. [cit. 2021-02-17]. Dostupné z: https://docs.microsoft.com/en-us/openspecs/microsoft_general_purpose_programming_languages/ms-vbal/9d9f80f7-bdf8-4d24-b15d-6bb9552d7c22

[8] *What is VBA? The Excel Macro Language* [online]. [cit. 2020-12-13]. Dostupné z: https://www.automateexcel.com/vba/what-is

[9] NEWMAN, Chris. Will VBA Die in 2019?. *The Spradsheet Guru* [online]. [cit. 2020-12-13]. Dostupné z: https://www.thespreadsheetguru.com/blog/are-vba-macros-dead

[10] KRÁL, Martin. *Excel VBA: výukový kurz*. 1. vydání. Brno: Computer Press, 2010. ISBN 978-80-251-2358-4.

[11] Declaring variables. *Microsoft Docs* [online]. [cit. 2020-12-14]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/language/concepts/getting-started/declaring-variables

[12] Data type summary. *Microsoft Docs* [online]. [cit. 2020-12-22]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/language/reference/user-interface-help/data-type-summary

[13] Set statement. *Microsoft Docs* [online]. [cit. 2021-01-20]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/language/reference/user-interface-help/set-statement

[14] Controlling One Microsoft Office Application from Another. *Microsoft Docs* [online]. [cit. 2021-01-20]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/excel/concepts/working-with-other-applications/controlling-one-microsoft-office-application-from-another

[15] KELLY, Paul. The Ultimate Guide To Collections in Excel VBA. *Excel Macro Mastery* [online]. [cit. 2020-12-28]. Dostupné z: https://excelmacromastery.com/excel-vba-collections/

[16] Using arrays. *Microsoft Docs* [online]. [cit. 2020-12-28]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/language/concepts/getting-started/using-arrays

[17] VBA Dynamic Array (Redim & Redim Preserve). *Automate Excel* [online]. [cit. 2020-12-28]. Dostupné z: https://www.automateexcel.com/vba/dynamic-array-redim-preserve/

[18] VBA Course: Conditions. *Excel-pratique* [online]. [cit. 2020-12-27]. Dostupné z: https://www.excel-pratique.com/en/vba/conditions

[19] Looping through code. *Microsoft Docs* [online]. [cit. 2020-12-28]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/language/concepts/getting-started/looping-through-code

[20] While...Wend statement. *Microsoft Docs* [online]. [cit. 2021-01-19]. Dostupné z: https://docs.microsoft.com/cs-cz/office/vba/language/reference/user-interface-help/whilewend-statement

[21] Using Do...Loop statements. *Microsoft Docs* [online]. [cit. 2020-12-28]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/language/concepts/getting-started/using-doloop-statements

[22] Using For...Next statements. *Microsoft Docs* [online]. [cit. 2020-12-28]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/language/concepts/getting-started/using-fornext-statements

[23] Using For Each...Next statements. *Microsoft Docs* [online]. [cit. 2020-12-28]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/language/concepts/getting-started/using-for-eachnext-statements

[24] With statement. *Microsoft Docs* [online]. [cit. 2020-12-27]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/language/reference/user-interface-help/with-statement

[25] VBE Glossary. *Microsoft Docs* [online]. [cit. 2020-12-14]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/language/glossary/vbe-glossary

[26] Calling Sub and Function procedures. *Microsoft Docs* [online]. [cit. 2020-12-30]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/language/concepts/getting-started/calling-sub-and-function-procedures

[27] LAURENČÍK, Marek. *Programování v Excelu 2007 a 2010: záznam, úprava a programování maker*. Praha: Grada, 2011. ISBN 978-80-247-7410-7.

[28] Getting started with VBA in Office. *Microsoft Docs* [online]. [cit. 2021-01-17]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/library-reference/concepts/getting-started-with-vba-in-office

[29] GOMEZ, J.A. Excel VBA Object Model And Object References: The Essential Guide. *Power Spreadsheets* [online]. [cit. 2021-01-17]. Dostupné z: https://powerspreadsheets.com/excel-vba-object-model/

[30] MAPI Features and Architecture. *Microsoft Docs* [online]. [cit. 2021-01-26]. Dostupné z: https://docs.microsoft.com/en-us/office/client-developer/outlook/mapi/mapi-features-and-architecture

[31] Automating Outlook from a Visual Basic Application. *Microsoft Docs* [online]. [cit. 2021-01-26]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/outlook/concepts/getting-started/automating-outlook-from-a-visual-basic-application

[32] Using ActiveX Controls on Sheets. *Microsoft Docs* [online]. [cit. 2021-01-27]. Dostupné z: https://docs.microsoft.com/en-us/office/vba/excel/concepts/controls-dialogboxes-forms/using-activex-controls-on-sheets

[33] How to Add Additional Controls in Excel VBA. *Excel-VBA Solutions* [online]. 2021 [cit. 2021-01-27]. Dostupné z: https://www.excelvbasolutions.com/2020/01/Excel-vba-add-additional-controls.html

[34] Access connection strings. *ConnectionStrings.com* [online]. c2021 [cit. 2021-02-17]. Dostupné z: https://www.connectionstrings.com/access/

# 8 Přílohy

Přílohou k této práci je CD se soubory:

- app.xlsm
- db.accdb

Prvně jmenovaný soubor je klient aplikace určený ke spuštění uživatelem. Druhý soubor je serverová část aplikace, se kterou klient komunikuje.